DIPARTIMENTO DI METODI E MODELLI
PER L'ECONOMIA IL TERRITORIO E LA FINANZA
MEMOTEF

SAPIENZA
UNIVERSITÀ DI ROMA

ADVANCES IN THE COMBINATION OF SUPERVISED CLASSIFICATION METHODS:AN EXPERIMENTAL STUDY

Sabina Mazza

Working paper n.133

October 2014

Università degli Studi di Roma "La Sapienza"
Via del Castro Laurenziano 9, 00161 Roma (RM)
T (+39) 06 49766433 F (+39) 06 4957606
www.memotef.uniroma1.it

ADVANCES IN THE COMBINATION OF SUPERVISED CLASSIFICATION METHODS: AN EXPERIMENTAL STUDY

Sabina Mazza[1]

ABSTRACT

In this work we were interested in investigating the predictive accuracy of one of the most popular learning schemes for the combination of supervised classification methods: the Stacking Technique proposed by Wolpert (1992),consolidated by Ting and Witten, (1999) and Seewald (2002). In particular, we made reference to the StackingC (Seewald 2002) as a starting point for our analysis, to which some modifications and extensions were made. Since most of the research on ensembles of classifiers tends to demonstrate that this scheme can perform comparably to the best of the base classifiers as selected by cross-validation, if not better, this motivated us to investigate the performance of the Stacking empirically. An analysis of the results obtained by applying our Stacking scheme - which includes differences and characteristic implementations compared to what is proposed by the literature - to the dataset generated by means of an experimental design, does not lead us to believe that the Stackin is preferable in terms of performances to the use of the best single classifier. It always achieves good performances and is to be considered among the best. On the contrary, in the case of contaminated data, Stacking improves its performances noticeably, and generally appears to be very competitive, above all when the contaminations are more substantial.

Classification JEL: *C 13, C 14, C 38*
Keywords: *Supervised classification methods, Ensemble learning, Stacking, Meta-level learning, Cross-validation*

1. INTRODUCTION

In this chapter, first we describe the framework in which this work is set that is a combination of supervised classification methods, in particular the Stacking Technique. Then we explain the motivation, goals and purposes and the tools and methods used to achieve them. We finally conclude with the outline of the subsequent chapters.

1.1 *Overview*

Among those elements which may influence the precision and stability of a classification method are the size and quality of the data set used for the estimation. Even slight modifications to the data set may lead to the construction of different models.

[1] Sabina.mazza@uniroma1.it.
This paper is based on research work on “Combination of classification methods” supervised by Professor Giorgio Alleva.

In order to satisfy the need for models that are more stable and more precise in their predictions, various methods have been proposed by the literature, based on the combination of models from the same class, among which: Bagging (Breiman, 1996), Boosting (Freund and Schapire,1996), Random Forest (Breiman 2001), and on others based on the combination of predictions deriving from different supervised classification methods.

This approach is also known as an *ensemble of classifiers* in the supervised classification task. The trend of studies in this direction that starts with *Stacked Generalization* (Wolpert, 1992) is particularly interesting, and is consolidated by the proposals offered by *Stacking* (Ting and Witten, 1999) and *Stacking C* (Seewald 2002), which tackle and overcome crucial problems previously unsolved in continuity with the original theory.

This class of models aims to combine the predictions coming from a set of different supervised classification algorithms (*base-level classifiers*) by means of a *meta-level classifier* in order to improve performances. The main idea behind Stacking is to use the predictions of the base classifiers as attributes in a new training set that keeps the original class labels, and then combine them.

The presence of outliers in the dataset, could also alter the structure of the classification model, and cause the generation of predictions that might not be reliable.

1.2 *Goals of the Work*

The proposal consolidated in the stacking framework and the relative advances in the research on the elements that characterise this scheme for the combination of classifiers were the starting point for this work which intends to investigate this theme further. The idea is to explore in greater detail some aspects that seem to be less developed in the literature and could contribute to the introduction of further elements into the research, side by side with those critical elements already highlighted in the report.

Most of the research on ensembles of classifiers tends to demonstrate that Stacking can perform comparably to the best of the base classifiers as selected by cross-validation, if not better. It is to be hoped that we can expect that the final classifier produced by Stacking is able to achieve better performances in terms of accuracy than the best level-0 classifier. Otherwise the computational onus created by the complexity of the procedure would not be justified.

This has motivated us to investigate empirically the performance of the Stacking technique, also in terms of stability and strength, solving the problem of the combination of supervised classified methods by using a different approach which may be defined as innovative.

The research trend described has established the following objectives for this work:

- Evaluation of the base-level and meta-level classifiers in terms of their accuracy when there are modifications in the size of the data set.
- Evaluation of the effects caused by the presence of anomalous values in the data set on the performances of the base-level and meta-level classifiers and their comparison.
- Evaluation of the results of the simulation studies carried out to establish whether, and to what extent, the combination of classifiers makes it possible to improve performances compared to the use of a single classifier.

On what we might define as the traditional level, a Stacking scheme is proposed that has some differences compared to the well-known one, both in terms of characteristics that are already present and with regard to the introduction of innovative elements.
In particular, with regard to the assumption at the base of the theory that ”even small changes to the training set may cause large changes in the classifier induced by a learning algorithm”, that Breiman (1996) defined as “instability”, referring to instable algorithms as classification trees and, taking into account that little has been discovered about the relationship between training dataset parameter settings and the performance of base classifiers and meta-classifiers, we chose not to use well-known input datasets, contained in databanks and extensively represented when dealing with problems of supervised classification.

Since the topic of the choice of input data is, in our opinion, an important part of the study, we chose not to use well-known datasets but rather to carry out a wide-ranging simulation study that involved the generation of datasets with different characteristics for the modification of the quality and size of the estimate data. Moreover, taking into consideration the effects that the presence of atypical observations might have on the model too, a great deal of space has been assigned to the contamination design of the datasets which has introduced anomalous values with varying degrees of intensity, level and typology to allow us to explore this particular dimension of the problem, covered very little by the literature, and to give any relative indications.

Among the main differences from the predominant literature we find the choice to build the Stacking procedure entirely in a Matlab environment. The whole Stacking scheme has therefore been implemented in Matlab and built in the form of an interlinked process which is begun by the generation of the level-0 data in the case of the simulation study, and which includes a complete homogenisation of the procedures relative to each of its phases in order to guarantee uniformity and therefore comparability of the outputs returned at every step.
This allowed us to create a flexible and powerful work tool, although it is sometimes a little heavy in computational terms.
Since the Stacking algorithm, although included in other software (such as the open source software WEKA), did not exist in Matlab, a code for

creating a complex structure was completely implemented, which made possible:

- The organisation of a complex and extended experimental plan in order to carry out a wide-ranging simulation study
- The building of a set of base classifiers and the appropriate procedure of cross-validation for carrying out the fit, the assessment and the generation of predictions for the formation of the input dataset for the meta-learner
- The fit and the assessment of the meta-classifier and thus the procedures for the prediction combination and the homogeneous processing of the results with regard to the characteristics of each method
- The creation of suitable plots

It was necessary to carry out a process of homogenisation for each step of the procedures for all the classifiers, as we pointed out, which were chosen voluntarily with different characteristics, in order to obtain the same output that is indispensable for making the structure function and for the assessment and comparability of the results.
There have been some extensions and modifications to some algorithms compared to the implementation provided for in Matlab, respecting all the decision rules that preside over individual functioning. Using the various implementations and modifications of the default parameters will provide an indication for each classifier in Chapter 3 in the Section dedicated to their description.

1.3 *Outline of Chapters*

In the following Chapter 2 we present the main proposals for the Stacking framework, giving a great deal of space to Wolpert's and its main extensions.

In the first place the one provided by Ting and Witten which tackles and resolves crucial problems previously unsolved that Wolpert defined "black art":

- the choice of the type of attributes that represent meta-level input data for the combiner function. They propose using the outputs represented by the probability distributions that are derived from using a set of base-level classifiers as *level-1 data* instead of the predictions of single class values as proposed by Wolpert.
- the choice of a level-1 generaliser in order to obtain improved accuracy using the stacked generalization method. They recommend the use of MLR (Multi-response linear regression) as a meta- level classifier, as used by Breiman (1996a) in a Stacked regression learning scheme, and by Le Blanc and Tibshirami (1993). They believe that MLR is the best level-1 combiner when compared with other learning algorithms.

Then, the Ensemble scheme proposed by Seewald is illustrated, *StackingC*, which is based on reducing the dimensionality of the level-1 dataset not considering the entire probability distribution associated with each classifier as in Ting and Witten, (1999), but rather the dataset composed only of probability vectors expressed by each k base-level classifier on the belonging of the unit to a defined class.

Other proposals are presented that deal above all with the choice of the meta classifier such as that by Merz (1999) which proposes SCANN. This uses the Stacking strategies together with correspondence analysis to detect any correlations between the predictions of base-level classifiers, and as the meta-level combiner a nearest neighbor method is applied to predict unseen examples.

Those which envisage the use of different types of Meta decision trees as meta-classifiers or those such as the contribution of Dzeroski and Zenko (2004) who propose two extensions of Stacking, one using an extended set of meta-level features and the other using multi-response model trees to learn at the meta-level. Finally an interesting proposal from Reid and Grudic (2009) which demonstrates empirically that even with a linear combination function, regularization is necessary to reduce overfitting and increase predictive accuracy and propose different kind of regularizations.

In Chapter 3 we describe the proposed Stacking scheme, with particular attention to the traditional components of the Stacking process, by indicating the main differences between the proposed and the more well-known one. The whole procedure implemented in Matlab, the simulation study for 0-level input data, the contamination design, the extensions and modifications and the parameter settings implemented for each classifier and Stacking, are illustrated.

In Chapter 4 the empirical results obtained from the application of the proposed Stacking scheme to datasets generated by means of the experimental design are shown. In particular, in section 4.2 the results relative to the non-contaminated data are illustrated, in order to investigate the effects on the performance of the base classifiers and of Stacking in the presence of input datasets with different characteristics. In Section 4.3 the application was carried out on simulated and contaminated data to investigate whether the presence of outliers can affect the performances of the base classifiers and of Stacking.

In Section 4.4 are illustrated the results obtained using three different Stacking variants with different base-level classifier subsets built on different datasets.

Chapter 5 concludes this work with a summary of the results and an outline of future developments.

2. STACKING FRAMEWORK

In this chapter we first describe the Stacking framework and then we summarize the main results of several recent studies of the Stacking technique for the combination of supervised classification methods.

The trend of studies that starts with *Stacked Generalization* (Wolpert, 1992) is particularly interesting, and is consolidated by the proposals offered by *Stacking* (Ting and Witten, 1999) and *Stacking C* (Seewald 2002), which tackle and resolve crucial problems previously unsolved in continuity with the original theory.

2.1 Stacked Generalisation

The aim of this ensemble learning scheme, originally proposed by Wolpert (1992), is to combine the predictions coming from a set of different supervised classification algorithms (*level-0 models*) by means of a *meta-level classifier* in order to improve prediction accuracy (as opposed to learning accuracy) as much as possible.

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.

The predictions of *level-0 classifiers* represent the attributes in a new training set (*level-1 data*), which keeps the original class labels. Stacking thus utilizes a *meta-learner* (*level-1 model*) to combine the predictions from different base classifiers which were estimated via cross-validation on a single data set.

There follows a brief description of the logic and the functioning of the Stacking technique together with a diagram (figure 1) which take into account some of the considerations made by Ting and Witten (1999) on the Wolpert proposal.

Given a set of K learning algorithms, called *level-0 generalisers* by Wolpert, and a data set :

$$L = \{(y_n, x_n), n = 1, ..., N\} \qquad (1)$$

where y_n is the target value and x_n is a vector whose elements represent the values assumed by the variables for the *n-th* instance.

Let L be randomly split into J roughly equal-sized parts: $L_1, L_2, ..., L_J$.

We define :

L_j and $L^{(-j)} = L - L_j$ as the test and training set for the *j-th* fold of *J-fold* Cross Validation and, $M_k^{(-j)}$ a model for *k = 1,…K* is induced on the training set $L^{(-j)}$. Level-0 models.
For each vector x_n belonging to L_j, the test set for the *jth* cross-validation fold, let z_{k_n} be the prediction of $M_k^{(-j)}$ on x_n.

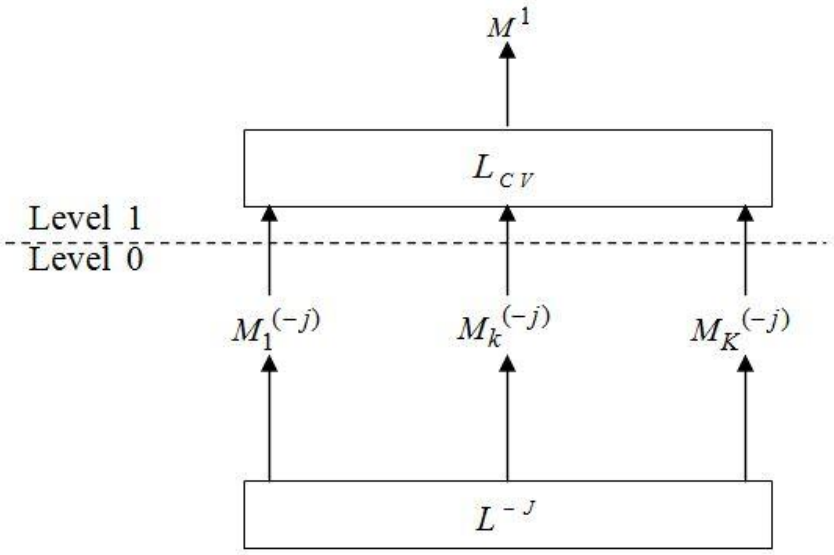

FIGURE 1. - *This figure illustrates the j-fold cross-validation process in level-0; the level-1 data set* L_{CV} *at the end of this process is used to produce the level-1 model* M^1 *(Ting and Witten, 1999).*

At the end of the cross-validation procedure, the dataset made up of the predictions of each K model using the terminology just introduced represents the level-1 data and is given by:

$$L_{cv} = \left\{(y_n, z_{1n}, ..., z_{Kn}), n = 1, ..., N\right\} \qquad (2)$$

The combiner function (*level-1 generaliser*) is then trained on this meta-level dataset to derive a Model M^1 (*level-1 model*) for y as a function of the predictions $(z_1, ..., z_k)$, whose output is the final classification of the units belonging to the input vector.

Formally, the final prediction function of *Stacked generalization* can be expressed by:

$$v_z(x) = \Sigma \alpha_k cz_z^k(x) \text{ for } k = 1, 2, ..., K \quad (3)$$

Where cz_z^k is a set of *k* predictors.

This is the model proposed by Wolpert (1992) and universally considered to be the base model of stacked generalisation. It has been revisited and studied in depth by several scientists such as Breiman (1996) who demonstrated the success of stacked generalization in the setting of ordinary regression and Le Blanc and Tibshirami (1993).

However, it is interesting to note that Wolpert himself believes that many aspects of stacked generalization are, at present, a kind of "black art", and, therefore have not yet been resolved.

These aspects will be dealt with and resolved subsequently in continuity with the original theory, as we will see in the following sections.

2.2 Stacking

Ting and Witten (1999) with their Stacking learning scheme shed light on the following aspects that Wolpert himself believed to be a kind of “black art” in Stacked generalisation:

- the type of attributes that should be used to form level-1 data,
- the type of level-1 generaliser in order to obtain improved accuracy using the stacked generalization method.

2.2.1 Meta level data

Ting and Witten (1999) have proposed settings for the meta classifier and the type of meta data to be used in the field of the problems of supervised classification such as the extension of the application of Stacked Generalization .

They propose using the outputs represented by the probability distributions that are derived from a set of base-level classifiers as *level-1 data* instead of the predictions of single class values as proposed by Wolpert.

When returning to the notation and to the reference scheme already used to describe Stacked generalization, if a generic model $M_k^{(-j)}$ is used to classify an instance *x* belonging to L_j, and $P_{ki}(x)$ is the *probability* that *x* belongs to the *i-th* class, the following vector:

$$\mathrm{P}_{kn} = (P_{k1}(x_n), \ldots, P_{ki}(x_n), \ldots, P_{kI}(x_n)) \quad (4)$$

represents the probabilities that the vector x_n belongs to the classes 1,..,*I*. assuming that classes have been returned by a single base-level classifier.

This gives the probabilities that the vector x_n belongs to the class 1,..,*I*. assuming that there are I classes and a set of k models with different bases. The level 1 dataset will be composed of the aggregation of the probability vectors generated by all k models:

$$L'_{CV2} = \{(y_n, \mathrm{P}_{1n}, ...\mathrm{P}_{kn}, ..., \mathrm{P}_{Kn}), n = 1, ..., N\} \qquad (5)$$

Compared to the previous ensemble scheme of Stacked generalization, the final new model will be M^2.

2.2.2 Meta-level classifier

Ting and Witten propose the use of MLR (Multi-response linear regression) as a meta- level classifier, as used by Breiman (1996a) in a Stacked regression learning scheme, and by Le Blanc and Tibshirami (1993). They believe that MLR is the best level-1 combiner when compared with other learning algorithms; it can represent a valid starting point in the search for the best method for meta-level learning to be used for problems in combining the supervised classification methods such as Stacking.

Linear regression can easily be used for classification in domains with numeric attributes. Indeed, *any* regression technique, linear or nonlinear, is suitable for classification.

MLR is an adaptation of a least squares linear regression. For a classification problem with I class values, *I* separated regression problems are fitted: for each class *I*, a linear equation *LRI* is constructed to predict a binary response variable, which has value one if the class value is *I* and zero otherwise. Given a new example *x* to classify, *LRI* (*x*) is calculated for all *j* , and the class *k* is predicted with maximum *LRk* (*x*).

MLR, therefore, learns a linear regression function for each class which predicts a degree of confidence in class membership and can, after normalization, be interpreted as class probability.

The output of the linear models, therefore, will have to be renormalized to yield a proper class probability distribution because the membership values they produce are not proper probabilities as they can fall outside the range 0 1.

Both Breiman (1996a) and LeBlanc & Tibshirani (1993) use the stacked generalization method in a regression setting and report that it is necessary to constrain the regression coefficients to be non-negative in order to guarantee that stacked regression improves predictive accuracy.

By modifying and simplifying Wolpert's hypothesis of Stacked generalization, seen in section 2.2, with regard to the final predictor: $v(x) = \Sigma \alpha_k v_k(x)$, the authors underline the need to enforce the non negativity of the coefficients $\{\alpha_k\}$, considering the hypothesis that the

different v_k, by making predictions about the same data, could be strongly correlated and there may be no guarantee that the final (stacked) predictor is near the range which might degrade the *generalisation* performance of this learning method.

Ting and Witten (1999) have shown that non-negativity constraints on coefficients are not necessary.

2.3 StackingC

Seewald (2002) proposed an extension of Stacking, called StackingC, based on reducing the dimensionality of the level-1 dataset independently of the number of classes and removing a priori irrelevant features. In order to overcome a weakness of Stacking (Ting and Witten, 1999) in problems with more than two classes. StackingC seems to display better performances in terms of accuracy compared to Stacking, especially for multi-class problems, while for two-class datasets the improvements are more moderate, while the reduction of the size of the features makes a gain in computational terms.
The proposed method includes the use as input for the level-1 classifier (each linear model is associated with each of the classes), not the entire probability distribution associated with each classifier as in Ting and Witten, (1999), but rather the dataset composed only of probability vectors expressed by each k base-level classifier on the belonging of the unit to a defined class (Figure 2). In the learning scheme StackingC, therefore, each linear model learns as input data only those partial class probabilities that it is trying to predict.

Classifier 1 a	Classifier 2 a	…	Classifier K a	Class = a
$P1_a(x_1)$	$P2_a(x_1)$		$PK_a(x_1)$	1
$P1_a(x_2)$	$P2_a(x_2)$		$PK_a(x_2)$	0
.	.		.	.
.	.		.	.
$P1_a(x_N)$	$P2_a(x_N)$		$PK_a(x_N)$	0

FIGURE 2. - *Level-1 data consisting only of partial probabilities given by each base-level classifier for class=a, k level-0 classifiers and N instances processed on the basis of the pattern proposed by Seewald (2002).*

The author maintains that the probability given by one classifier for only one class can be sufficient to guarantee the information necessary and also to ensure a good performance, because the sum of each class probability distribution has to be one, the probability of one class is one minus the probability of the other class.

The use of MLR (Multi-Response Linear Regression) as a meta-level classifier is confirmed. Seewald (2002) tries to use other combiner functions instead of MLR, such as LWR (Locally Weighted Regression) and MP5Prime, a model tree learner implemented in the WEKA open-source software (Waikato Environment for Knowledge Analysis) developed at the University of Waikato in New Zealand. Empirically he finds that for two-class datasets MLR is the best classifier, even if the differences are minimal.
The author believes that, in this case, the source of the improvement lies partially in the dimensionality reduction, but more importantly in the higher diversity of class models that are combined.

2.4 Related Work

There have been several studies on combining classification models, including of course those on the Stacking framework.
The purpose of most of this research has been to study in depth those aspects defined by Wolpert as “black art” and therefore a great deal of attention has been paid to the choices in terms of meta data and meta-level classifiers.
There are several interesting proposals and the main ones will be looked at in brief below.

Merz (1999) proposes a method called SCANN (Stacking Correspondence Analysis and Nearest Neighbour) that uses the strategies of Stacking and correspondence analysis detect any correlations between the predictions of base-level classifiers, because it is well known that the combination of different classifiers improves the accuracy performance if they are weakly correlated. The original meta-level feature space (the class-value predictions) is transformed into a space of uncorrelated features. As the meta-level combiner a nearest neighbor method is applied to predict unseen examples. The author compares SCANN with two other stacking schemes that have a Naïve Bayes classifier as a meta-learner and a back-propagation trained neural network. Merz applied SCANN in this work to classifiers that only return class value predictions and not class probability distributions as in Stacking.

Todorovski and Dzeroski (2000) introduced a new algorithm to be used as a level-1 classifier: the meta decision Trees (MTDs), whose leaves do not contain class value predictions. Instead the most appropriate base level classifier to be used for classifying the unit that falls in that leaf is indicated.

As first level dataset attributes they do not propose the use of probability distributions, but rather their characteristics, such as entropy and maximum probability, since they may be interpreted as estimates of the confidence of the classifier in its predictions.

Zenko et al. (2001) report that MDTs perform slightly worse compared to stacking with MLR. Overall, SCANN, MDTs, stacking with MLR and SelectBest seem to perform at about the same level. It would seem natural to expect that ensembles of classifiers induced by stacking would perform better than the best individual base-level classifier: otherwise the extra work of learning a meta-level classifier does not seem justified. The experimental results, however, do not show clear evidence of this.

Todorovski and Dzeroski (2003) report that stacking with MDTs makes it possible to exploit better than voting the differences between the base-level classifiers and has a better performance, especially in the hypothesis in which the mistakes made by the base level classifiers are uncorrelated. It is also superior when compared with SCANN, and the main ensemble methods of weak learners (especially decision trees) such as bagging and boosting.

Dzeroski and Zenko (2004) propose two stacking extensions with MLR, one using an extended set of meta-level features and the other using multi-response model trees instead of MLR as meta-classifiers. Firstly, the authors use the probabilities predicted for each class by each base classifier as meta-level features (as proposed by Ting and Witten) but augment with two additional sets of meta-level attributes: the probability distributions multiplied by the maximum probability and the entropies of the probability distributions. The results of their experiments show that there are no significant improvements when using only these two attributes (without the probability distributions), but when using all three sets of features at the same time, some improvements are noticeable. The second extension considers an alternative for MLR as meta-classifier, introducing Stacking with multi-response model trees, because model trees have been shown to perform better than MLR for classification via regression.

Reid and Grudic (2009) return to the need to insert constraints on coefficients; in fact they demonstrate empirically that with a linear combination function, regularization is necessary in order to improve accuracy and reduce overfitting. They propose using Ridge regression, lasso regression and elastic net regression because Stacking has a tendency to overfit, especially when highly correlated and well-tuned combination methods are used.

2.5 Discussion

We have outlined the main proposals of the literature that examine in depth and extend the Stacking Technique with particular attention paid to the choice of meta data and meta classifiers. In the work, as indicated, we will

use the Stacking C ensemble scheme as a starting point for our analysis, but we plan to focus our attention on an exploration of the parameters, as well as the choice of meta data and meta classifiers. We also focus on an aspect that has been covered much less by the studies and that in our view, however deserves special attention: the choice of the initial dataset. This is connected to the assumption that small changes in the dataset can lead to different models and that the presence of outliers might alter the parameters of the model.

3. ADVANCES IN THE STACKING SCHEME

3.1 Introduction

The proposals illustrated in the previous chapter for the Stacking framework and the relative progress made in the research on the elements that characterize such a scheme of classifier combination are a valid starting point for this work, which intends to investigate this topic further. The idea is to explore in more detail some of the aspects that seem less developed and could contribute to the introduction of further elements into the research, side by side with the critical elements already highlighted in this work.
In the following section some of the components of the Stacking technique will be explained, especially in the usual outlook, while in section 3.3, the elements that are the essential aspects for constructing our Stacking model will be introduced, with clarifications regarding the main differences compared to the traditional formulation, both in terms of modifications in the characteristics of elements already found and with regard to the introduction of innovative elements.

3.2 *Traditional elements of Stacking ensemble method*

As we have demonstrated several times, elements traditionally considered to be critical for dealing with problems in the combination of supervised classification methods, and in particular in the ensemble Stacking method, are represented by the choice of base classifiers, meta classifiers and also by the type of meta data to be used. In our opinion, another important aspect is the assessment of the classifiers that will be illustrated in the section 3.2.3.
These elements will be summarized below, but it should be clear that there will not be a thorough examination of this theme, since many other types of learning algorithms could be used to generate base classifiers, and other typologies of meta-classifiers, used to provide a final prediction, but usable for describing the components that will be inserted in the Stacking process built in this work and described in section 3.3.

3.2.1 *Base Classifiers*

The base classifiers that will be used to build the proposed Stacking scheme, are methods having different characteristics because the learning algorithms that generated them are different. We made a voluntary choice to use classifiers in the combination that have different predictive capacities and strengths together with different decision rules for investigating whether the combination is able to enhance the performances of the most capable and mitigate the weaknesses of the less able performers, and therefore Stacking can perform comparably to the best of the individual classifiers, if not better.

To make this treatment easier we can distinguish three categories among the algorithms that we will use such base classifiers in to the experimental set up:

- **Parametric methods**
  - Linear Discriminant Analysis
  - Quadratic Discriminant Analysis
  - Logistic Regression
  - Naive Bayes

- **Non-Parametric Methods**
  - Classification Tree
  - Support Vector Machine

- **Ensemble Methods**
  - Bagged Classification Tree
  - AdaBoost

The general formulations of the proposed algorithms will be summarised in Appendix, while in section 3.3.3 specific implementations carried out in a Matlab environment and relative to each algorithm will be illustrated.

3.2.2 *Meta-classifiers*

The most interesting of the proposed meta-classifiers are the following:

- Multi Response Linear regression (MLR)

- Ridge Regression

Multi-response Linear Regression is an adaptation of a least squares linear regression recommended (Ting and Witten 1999) for meta-level

learning while several learning algorithms are shown not to be suitable for this task.

For a classification problem with K class values, *K* separated regression problems are fitted: for each class *k* , a linear equation *LR*k is constructed to predict a binary response variable, which has value 1 if the class value is *k* , and 0 otherwise. Given a new example *x* to classify, *LRk* (*x*) is calculated for all *j* , and the class *k* is predicted with maximum *LRk* (*x*). MLR, therefore, learns a linear regression function for each class which predicts a degree of confidence in class membership and can, after normalisation, be interpreted as class probability. The output of the linear models, therefore, will have to be renormalized to yield a proper class probability distribution because the membership values it produces are not proper probabilities as they can fall outside the range 0-1.

By using the cross-validated predictions $f^{-k}(x)$ at *x*, using model *m*, applied to the dataset with the *i*th training observation removed. The stacking estimate of the weights is obtained from the least squares linear regression of y_i *on* $f_m^{-1}(x)$, *m=1,2,.....*M.

The stacking weights are given by:

$$w^{st} = \arg\min_{w} \sum_{i=1}^{N} \left[y_i - \sum_{m=1}^{M} w_m f_m^{-i}(x_i) \right]^2 . \qquad (6)$$

The final prediction then is $\sum_m w_m^{st} f_m(x)$.

Hastie et al. (2009) believe that better results can be obtained by restricting the weights to be nonnegative, and to sum to 1. This seems like a reasonable restriction if we interpret the weights as posterior model probabilities.

Ridge Regression

Ridge Regression, introduced by Hoerl and Kennard (1970), shrinks the regression coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squares

$$\beta^{ridge} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \qquad (7)$$

Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage. The coefficients are shrunk towards zero (and each other).

An equivalent way to write the ridge problem is:

$$\beta^{ridge} = \arg\min_{\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta j \right)^2 \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \le t \qquad (8)$$

which makes explicit the size constraint on the parameters. There is a one-to-one correspondence between the parameters *λ* in (7) and *t* in (8). When there are many correlated variables in a linear regression model, their coefficients can become poorly determined and exhibit high variance. By imposing a size constraint on the coefficients, as in (8), this problem is alleviated. the intercept β_0 has been left out of the penalty term. The solution to (7) can be separated into two parts, after reparametrization using *centered* inputs: each x_{ij} gets replaced by $x_{ij} - \bar{x}_j$.

We estimate β_0 by $y = \frac{1}{N} \sum_1^N y_i$.

The other coefficients get estimated by a ridge regression without intercept, using the centered x_{ij}. Henceforth we assume that this centering has been done, so that the input matrix X has *p* (rather than *p* + 1) columns.
Writing the criterion in (7) in matrix form,

$$RSS(\lambda) = (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta \qquad (9)$$

the ridge regression solutions are easily seen to be:

$$\beta^{ridge} = \left(X^T X + \lambda I \right)^{-1} X^T y \qquad (10)$$

where I is the $p \times p$ identity matrix. Notice that with the choice of quadratic penalty $\beta^T \beta$, the ridge regression solution is again a linear function of y. The solution adds a positive constant to the diagonal of $X^T X$ before inversion. This makes the problem nonsingular, even if $X^T X$ is not of full rank, and was the main motivation for ridge regression when it was first introduced in statistics (Hastie et al. 2009). Ridge Regression is recommended such a meta-combiner in a Stacking scheme by Le Blanc and Tibishirami (1993), Breiman (1996) and, recently, Reid and Grudic (2009).

3.2.3 *Classifiers Assessment*

The generalization performance of a learning method relates to its prediction capability on independent test data. In classification task, we are interested to assess the ability of a learning algorithm to generalize on

unseen data. It is common to measure a classifier's performance in terms of accuracy. Where:

Accuracy = 1 - generalization error rate

It is our choice to measure and compare the performances of the classifiers based on their prediction error rate. The error rate is the proportion of misclassified instances over a whole set of instances, and it measures the overall performance of the classifier.
Of course one can be interested in the likely future performance on new data, because the error rate on the training set is not likely to be a good indicator of future performance.

$$Tra_{err} = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f\left(x_i\right)\right) \qquad (11)$$

Any estimate of performance based on that data will be optimistic. Training error consistently decreases with model complexity, typically dropping to zero if we increase the model complexity sufficiently. However, a model with zero training error is overfitted to the training data and will typically generalize poorly. (Hastie et al. 2009).
Test error, or generalization error, is the prediction error on an independent test sample given by a classification method $f\left(X\right)$ that has been estimated from a training set. To predict the performance of a classifier on new instances, we need to evaluate its generalization error rate on a dataset that has not been part of the classifier's fit. The test data must not be used in any way to build the classifier.
When the amount of data for splitting in training and test set is limited, one of the simplest and most popular methods for estimating prediction error is *K*-fold cross-validation.
We first split the data into *K roughly* equal parts. Then for each *k* = 1*, . . . ,* *K*, we remove the *kth* part from our data set and fit a model $f^{-k}\left(x\right)$.
Let *Ck* be the indices of observations in the *k*th fold. The cross-validation estimate of the expected test error is:

$$CV\left(f\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f^{-k(i)}\left(x_i\right)\right). \qquad (12)$$

Overall, five- or tenfold cross-validation are recommended as a good compromise: see Breiman and Spector (1992), Kohavi (1995) and Guyon et al. (2006).

3.3 *Experimental set up of our Stacking proposal*

Our thesis proposes, with regard to the "traditional" approach, the following objectives:

- Evaluation of the base-level and meta-level classifiers in terms of their accuracy when there are modifications in the input data set

- Evaluation of the effects caused by the presence of anomalous values in the data set on the performances of the base-level and meta-level classifiers and their comparison

- Evaluation of the results of the simulation studies carried out to establish whether, and to what extent, the combination of classifiers makes it possible to improve performances compared to the use of a single classifier.

In order to achieve these objectives we built a Stacking scheme that includes some differences and characteristic implementations compared to what is proposed by the literature, and these will be specifically explained below for each element of the process.

3.3.1 *Software*

Among the main differences from the predominant literature we find the choice to build the Stacking procedure entirely in a Matlab environment. [MATLAB 7.12.0 (R2011a) and (R2011b)]. This allowed us to create a flexible and powerful work tool, although it is sometimes a little heavy in computational terms.

The whole Stacking scheme has therefore been implemented in Matlab and built in the form of an interlinked process which is begun by the generation of the level-0 data in the case of the simulation study, and which includes a complete homogenisation of the procedures relative to each of its phases in order to guarantee uniformity and therefore comparability of the outputs returned at every step.

Since the Stacking algorithm, although included in other software (such as the open source software WEKA), did not exist in Matlab, a code for creating a complex structure was completely implemented, which made possible:

- The organisation of a complex and extended experimental plan in order to carry out a wide-ranging simulation study.
- The building of a set of base classifiers and the appropriate procedure of cross-validation for carrying out the fit, the assessment and the generation of predictions for the formation of the input dataset for the meta-learner.

- The fit and the assessment of the meta-classifier, by means of an appropriate cross validation procedure, and thus the procedures for the prediction combination and the homogeneous processing of the results with regard to the characteristics of each method.
- The creation of suitable plots.

It was necessary to carry out a process of homogenisation for each step of the procedures for all the classifiers, as we pointed out, which were chosen voluntarily with different characteristics, in order to obtain the same output that is indispensable for making the structure function and for the assessment and comparability of the results.

Of course this has also led to homogeneity for the base classifiers in the procedures for the entire construction process for each one, to the generation of the predicted class labels and the relative computation of the prediction error (in this case cross validation error or extra-sample error which represents the fraction of the misclassified observations of the test total computed by the difference between the predicted class label and the true class label relative to the test set), which meant, earlier in the process, the implementation of the stratified k-fold cross-validation procedure, constructed in the same way for all the classifiers, including those for which this was not planned, which allowed us to achieve the same data partition in training sets and test sets.

Similarly, since we follow the approach of using probability distributions generated by base classifiers as metadata (because we believe that this is better than the predictions), but not all the selected base classifiers return class probabilities as output, we implemented the Matlab procedure for each classifier, in order to generate posterior probabilities (or to make transformation from predictions to probabilities), which are indispensable for creating meta-classifier input datasets, especially for those models which are not predicted by default. Thus the classifiers were not chosen on the basis of return output, but on the basis of their heterogeneity, which is a contribution to their knowledge of the phenomenon.

With regard to the decision rules and the typical characteristics of each algorithm, the following have been made available for each classifier, and implemented if not already present in Matlab :

- Posterior probabilities of training
- Training error rate
- Predicted class labels
- Posterior probabilities of testing obtained through ten-fold cross-validation
- Mechanisms for the partition and iteration of the dataset for cross validation
- Cross validation error rate

3.3.2 *Simulation study for 0-level input data*

The proposals from the literature reported so far have always used well-known datasets from the UCI learning repository for the building of Stacking schemes (Blake and Merz 1998).

With reference to what has been pointed out several times, that is to say that even small changes to the training set may lead to different models and, taking into account that little has been is known about the relationship between training dataset parameter settings and performances of base classifiers and meta-classifiers, we chose not to use well-known datasets, contained in databanks and used extensively for dealing with problems of supervised classification.

The part that deals with the data is, in our opinion, an important moment in our study, and taking this into consideration, we carried out a wide-ranging simulation study which led to the generation of datasets with different characteristics as follows:

$N = size\ of\ the\ population$

$k = number\ of\ classes$

$nj = size\ of\ the\ class \qquad (j = 1, ..., k)$

$v = number\ of\ features$

$\delta = degree\ of\ separation\ among\ the\ subpopulations$

$prior = prior\ probability\ to\ belong\ to\ a\ class$

$Niter = number\ of\ iterations$

For each dataset we generated two groups of *nj* observations; the first group consists of a *nxv* matrix generated from a standard multivariate normal population with a mean equal to μ for all variables and a covariance matrix Σ. The Second Group was also generated from a multivariate normal population, but with a mean equal to $(\mu + \delta)$.

We therefore imposed a different degree of separation δ between the groups.

Furthermore, taking into consideration the effects that atypical observations might have on the model too, we decided to build a well-organised and complex experimental contamination design that would allow us to explore this particular aspect of the problem, which has been paid very little attention by the literature, and draw some conclusions.

Contamination design

Typology: shift contamination
Level: proportion of contaminated data:
The contamination can be carried out on:

- only one class
- on both

Intensity: value of the constant to be added to the original data.

Both for datasets with and without contamination, each simulation in the Stacking process is repeated in a series of 100 trials. In each trial a dataset is generated by following the characteristics indicated in the section above.

3.3.3 *Base Classifiers*

The proposed Stacking scheme includes the use and subsequent combination of a set of 13 base-level classification methods generated by the 8 learning algorithms, as indicated in the subdivision of section 3.2.1, whose output will be used as input data for the meta-level algorithms.

The set is therefore composed of classifiers that have been generated by following several criteria:

- by applying different learning algorithms to a single data set
- by applying a single learning algorithm with different parameter settings to a single data set
- in the case of multiple classifiers by applying a single learning algorithm to the different variants of a dataset (bagging, boosting)

As we have already indicated, very few modifications were made intentionally to their default parameter settings and the exceptions will be included in the description of the single classifiers.
In the following part of this Section we illustrate the main parameter settings of the base classifiers used if they are different from the default ones.

- **Linear Discriminant Analysis and Quadratic Discriminant Analysis**

No significant modifications were made to their default parameter settings, but of course this is without considering what has been indicated for the organisation of the cross validation procedure (which holds for all the classifiers, excluding a version of Classification Tree and of Adaboost, as we will see later), which has made it possible to return the output predicted class labels and posterior probabilities homogeneously, and also to compute cross validation error.

- **Naive Bayes.**

Since the algorithm provides support for Gaussian and Kernel distribution, both were used in the experimental phase. In fact, it seems appropriate to use the Gaussian distribution for features that have normal distributions in each class, since for each dataset we had generated two groups of *nj* observations consisting respectively of a *nxv* matrix generated from a standard multivariate normal population. However, in the algorithm's training phase we also used a Kernel distribution that is appropriate for features that have a continuous distribution. Since this requires more computing time and more memory than the normal distribution and since in our case the results did not seem significantly better, we preferred to use the normal distribution of the proposed scheme.

- **Classification Tree**

These were used in the two versions:

- with pruning, which computes the optimal sequence of pruned subtrees (TRE)
- without pruning, which returns the decision tree that is the full one (TRE1)

- **Bagged Tree**

To estimate the prediction error of the bagged ensemble, instead of computing predictions for each tree on its out-of-bag observations, we average these predictions over the entire ensemble for each observation and then compare the predicted out-of-bag class value with the true class at this observation (as by default), and we built the cross validation procedure on the entire dataset.
We created an ensemble of 30 bagged decision trees.

- **Adaboost**

We use two ensemble algorithms:

- First, (ADAm) based on AdaBoostM1 (Freund and Schapire, 1996).
  The base classifier returns a discrete class label.
  Weak learner = tree.
  Number of ensemble learning cycles = 30

- For the second (ADA), we created a personalized function that extends the Matlab function "adaboost", with the implementation of the cross validation procedure, and, for calculating posterior

probabilities extends the calibration of the output of AdaBoost. MH proposed by Busa-Fekete et al. (2011) for multi-class problems.
Number of ensemble learning cycles=30

- **Support Vector Machine (LIBSVM)**

The Support Vector Machines (SVM), developed in the 1990s (Boser et al., 1992; Cortes and Vapnik,1995) are held to be among the most effective methods of supervised learning. They were implemented in the scheme proposed through LIBSVM by Chang and Lin (2011), one of the most widely used SVM software programs.
Four different implementations of the algorithm were created and for all of them the transformation of the design matrix was implemented in a sparse matrix, the procedure of common cross validation as for the other classifiers, together with the computing of posterior probabilities for extending SVM to give probability estimates (instead of only class labels as default).
The Kernel function was chosen as a reference: RBF (Gaussian) kernel:

$$K\left(x,x\right)=e^{-\gamma\left\|x-y\right\|^2},\ \gamma>0$$

The specific implementations for each version of the algorithm are summarised as follows:

- SVM
- Scaled SVM
  The authors recommend linear scaling. We have chosen to scale each attribute to the range [0,1]
- SVMbest
  A procedure of cross validation was implemented in order to choose the best parameters (C,γ) for an RBF kernel. Various pairs of (C,γ) values are tried and the one with the best cross validation accuracy is chosen. We recommend trying exponentially growing sequences of parameters C and γ to select good ones (e.g. $C=2^{-5},2^{-3},...,2^{15}$ $\gamma=2^{-15},2^{-13},...,2^{3}$). Although the grid search in cross validation is recommended, it means a great deal of computational time, at least with the values suggested.
- scaled SVMbest
  There are the implementations for SVM best and for scaled SVM.

3.3.4 Meta-classifiers

Since we believe that both MLR and ridge regression are valid algorithms for combining the outputs of base classifiers, we decided to use both of them with a mechanism that establishes in a mutually exclusive way the

application of linear regression in the hypothesis in which there is no multicollinearity for input matrix X, and of ridge regression when the dimensionality of the meta-feature space *L* (*L*=number of base classifiers) is greater than the effective rank of the input matrix.

- **MLR** (Multi-Response Linear Regression)
- **Ridge Regression**

The use of cross-validation on meta-data has also been envisaged for meta-classifiers, to build and then evaluate the meta-classifier, reduces the risk of overfitting and enables us to consider the estimate of the prediction error given by such a process as a generalization error of the Stacking scheme.
For this part of the work, therefore, we made extensive use of cross validation, since we use it in order to:

- build the classifiers (base and meta-level) from the training data
- estimate the prediction error of the base classifiers and the final model
- estimate the unknown tuning parameters (particularly for Ridge Regression and Support Vector Machine).

However, as we have mentioned before, we preferred not to proceed with an extreme tuning of the parameters with regard to the objective of investigating whether the Stacking with the combination of different methods is able to improve the performances of the classifiers, mitigating any bad performances, especially those of the “weakest” ones.

3.3.5 Evaluating and comparing Classification Methods

The generalization errors of the base classifiers for a given input dataset and of a meta-classifier (for a input dataset generated from partial class probability distributions from each base classifier) are estimate by averaging the result of 100 runs of ten-fold stratified cross validation . Cross validation is repeated 100 times using different random seeds of the data resulting in 100 different sets of folds. The same folds are used in all experiments to built all the base classifiers and to estimate their true errors.
It should always be remembered that even though we put together values relative to Stacking and base classifier errors in the tables and in the different plots, they are constructed using different typologies of input data.

A comparison can be made among Stacking schemes and for the single Stacking scheme just to establish whether or not Stacking is the better, worse or at least equal to the best base classifier.
Because of the variability and fluctuation of the cross validation error, the average does not seem to be sufficient and in addition other measures are calculated on the distribution of the cross validation errors for each classifier and averaged over the total of the iterations carried out.

- **Position indices**
  - Median
  - Percentage of best positioning

- **Indices of variability**
  - Standard deviation
  - Median of deviations from the median
  - Interquartile difference
  - Range

3.4 *Discussion*

The state of the art in the research of the Stacking framework was a valid starting point for this work. In this chapter we have illustrated the main characteristics of the proposed Stacking scheme, starting from the choice of implementing the whole Stacking scheme in the Matlab environment and built in the form of an interlinked process which is begun by the generation of the level-0 data in the case of the simulation study which led to the generation of datasets with different characteristics and, furthermore, taking into consideration the effects that atypical observations might have on the model too, to build a well-organised and complex experimental contamination design that would allow us to explore this particular aspect of the problem, to which very little attention has been paid by the literature, and to draw some conclusions. Of course, the implementation of the entire Stacking scheme required a complete homogenisation of the procedures relative to each of its phases in order to guarantee uniformity and therefore a comparability of the outputs returned at every step. Furthermore, the creation of a double procedure of cross-validation both for base and meta-classifiers (which are represented by MLR and Ridge Regression, the use of which is regulated by a mutually exclusive insertion mechanism where conditions of collinearity occur), made it possible to build and evaluate classifiers at a double level, thus reducing the risk of overfitting too. Together with cross validation error, other measures have been included, calculated on the error distribution of cross validation for each classifier and for Stacking. In the following chapter ample space will be given to the results of the application of the proposed scheme to the datasets generated by the experimental design, contaminated and non-contaminated, and on

real datasets to empirically verify their functioning. With a view to a further improvement of the entire proposed process, which is at the experimental stage, the research activity will be directed towards optimising performances and guaranteeing the reliability of the predictions for single classifiers by modifying the setting of the parameters used in this phase.

4.EXPERIMENTAL RESULTS

In this section there is a summary of the main results obtained from the application of the proposed Stacking scheme to datasets generated by means of the experimental design. In particular, in section 4.2 the results relative to the non-contaminated data are illustrated, in order to investigate the effects on the performance of the base classifiers and of Stacking in the presence of input datasets with different characteristics. In section 4.3 the application was carried out on simulated and contaminated data to investigate whether the presence of outliers can affect the performances of the base classifiers and of Stacking. In Section 4.4 are illustrated the results obtained using three different Stacking variants with different base-level classifier subsets built on different datasets.
In Appendix to Chapter 4 is illustrated the application of the proposed scheme to real data.

4.2 *Simulated data*

Based on the characteristics indicated in the previous chapter, datasets have been generated with the following characteristics, which make them different in terms of complexity and degree of separation among the groups:

N = 120; 200
n_1 = 60; 100
n_2 = 60; 100
v = 3; 5 ; 7 ; 10
δ = 0.5; 1; 1.5; 2; 2.5; 3

The experimental design for a fixed scheme with 13 base-level classifiers is used, to which we always refer for completeness. Subsequently, some examples of level-0 data will be reported for the base-level algorithms. Ten-fold stratified cross-validation was used on each dataset to build single methods and estimate the prediction error of all the base-level classifiers. Every trial process was repeated 100 times and an average of the results was calculated in order to find the error of average cross validation for each classifier, relative to each experiment.

The posterior probabilities of each classifier derived from the process of cross-validation form the meta-dataset for the meta-classifier.

Ten-fold stratified cross-validation is also used, repeated 100 times, for the meta classifiers, which are Linear regression and Ridge regression (mutually exclusive when hypotheses of multicollinearity recur).

We are interested in investigating empirically the performances of the single classifiers and of Stacking for datasets with different characteristics and above all if it is convenient to use Stacking in terms of improving performances instead of a single classifier, bearing in mind the necessary increase in computation.

It should always be remembered that even though we put together values relative to Stacking and base classifier errors in the tables and in the different plots, they are constructed using different typologies of input data.

A comparison can be made among Stacking schemes and for the single Stacking scheme just to establish whether or not Stacking is better, worse or at least equal to the best base classifier.

An analysis of the results obtained by applying the Stacking scheme to the set of the datasets generated by means of the experimental design does not lead us to believe that the prediction error of the Stacking scheme is to be considered lower than any other classifier or that, therefore, the Stacking scheme is preferable in terms of performances to the use of the best single classifier.

TABLE 1.- *Simulated Data.Measures of the performances of the classifiers and of the Stacking scheme. Average figures over 100 iterations. N=200, v=10,δ=0.5*

	Cross Validation Error	Median Cross Validation Error	Std. Deviation Cross Validation Error	Difference interquartile Cross Validation Error	Range Cross Validation Error	MAD Cross Validation Error	% Best positioning
Classifier							
LDA	0,2334	0,2300	0,0315	0,0450	0,1450	0,0200	12
QDA	0,2703	0,2700	0,0398	0,0550	0,2000	0,0300	0
TRE	0,3565	0,3550	0,0418	0,0500	0,2350	0,0250	0
TRE1	0,3580	0,3600	0,0386	0,0450	0,2050	0,0250	0
BAG	0,2799	0,2800	0,0347	0,0475	0,1750	0,0250	2
ADA	0,2767	0,2800	0,0357	0,0475	0,1900	0,0250	1
ADAm	0,2764	0,2750	0,0340	0,0475	0,1500	0,0250	0
NBA	0,2353	0,2350	0,0287	0,0400	0,1200	0,0200	12
SVM	0,2443	0,2400	0,0329	0,0425	0,1500	0,0225	7
SVMscaled	0,2239	0,2200	0,0315	0,0450	0,1400	0,0200	33
SVMb	0,2443	0,2400	0,0329	0,0425	0,1500	0,0225	0
SVMbscaled	0,3497	0,2650	0,1542	0,3025	0,4650	0,0650	9
GLM	0,2313	0,2300	0,0300	0,0450	0,1400	0,0200	8
STA	0,2314	0,2350	0,0361	0,0450	0,1850	0,0225	16

It always achieves good performances and is to be considered among the best, but it does not seem to be preferable for this type of application.

As we can see in an example summarised in Table 1, which shows the results of the application of the Stacking scheme to the dataset obtained from the experimental design characterised by $n_1 = n_2 = 60$, $v = 10$, $\delta = 0.5$. It should be noted that the best positioning is always in agreement with the lowest error.

We can appreciate a certain variability in the boxplots of error distribution in Figure 3.

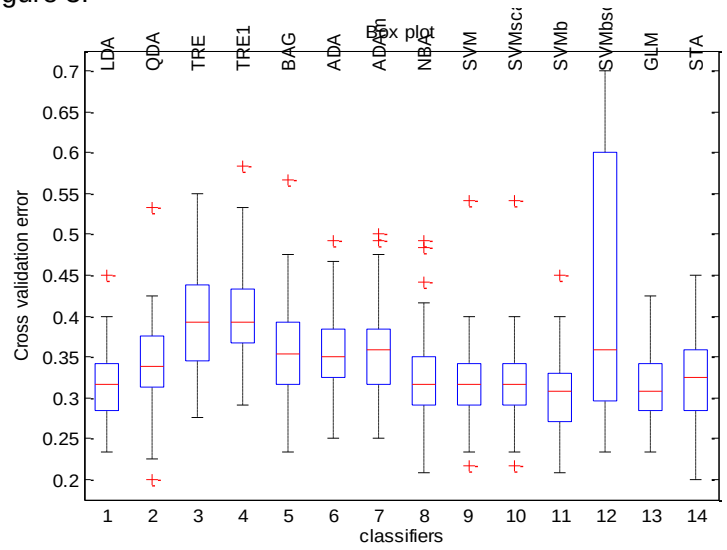

FIGURE 3. - *Simulated Data. Boxplots of error distribution on 13 base classifiers and Stacking scheme. N=200, v=10,δ=0.5*

On a more general level, Table 2 illustrates the results relative to the average cross validation error for each base classifier and for Stacking for input datasets with varying degrees of complexity and with different degrees of separation between the two groups.

The analysis was carried out on the entire set of the generated datasets, but for the sake of brevity we will only report the most important results. The behaviour of scaled SVM (which represents one of the implementations adopted for the Support Vector Machine described in Chapter 3 is particularly interesting, as it is has the lowest error among the classifiers, also in comparison with Stacking. Instead SVMbest does not achieve such moderate error levels and this pushes us to improve the cross validation procedure used for tuning the parameters.

TABLE 2. - *Simulated Data. Cross validation error for each base classifier and for Stacking for input datasets with varying degrees of complexity and with different degrees of separation between the two groups. N=120*

	δ=0.5				δ=3			
	n. variables				n. variables			
Classifier	3	5	7	10	3	5	7	10
LDA	0,3432	0,3145	0,2738	0,2396	0,0053	0,0003	0,0001	0,0000
QDA	0,3539	0,3392	0,3123	0,2948	0,0058	0,0003	0,0002	0,0000
TRE	0,4174	0,3958	0,3807	0,3602	0,0392	0,0409	0,0428	0,0418
TRE1	0,4243	0,3980	0,3761	0,3603	0,0381	0,0397	0,0425	0,0407
BAG	0,3921	0,3586	0,3258	0,2840	0,0197	0,0101	0,0035	0,0023
ADA	0,3785	0,3553	0,3213	0,2809	0,0201	0,0193	0,0199	0,0223
ADAm	0,3795	0,3557	0,3177	0,2863	0,0206	0,0048	0,0040	0,0072
NBA	0,3509	0,3236	0,2809	0,2476	0,0050	0,0003	0,0000	0,0000
SVM	0,3436	0,3188	0,2856	0,2431	0,0058	0,0005	0,0001	0,0000
SVMscaled	0,3347	0,3032	0,2657	0,2274	0,0044	0,0002	0,0000	0,0000
SVMb	0,3436	0,3188	0,2856	0,2431	0,0058	0,0005	0,0001	0,0000
SVMbscaled	0,4033	0,4252	0,4069	0,4119	0,0428	0,0052	0,0076	0,0000
GLM	0,3427	0,3129	0,2757	0,2428	0,0092	0,0009	0,0002	0,0000
STA	0,3545	0,3237	0,2846	0,2467	0,0073	0,0012	0,0001	0,0001

Thus, by using a set of classifiers containing scaled SVM, Stacking scheme could prove itself not to be competitive, as we can also see in Table 3 which summarises the performances of scaled SVM and the Stacking scheme on all the datasets generated by this part of the experimental design.

TABLE 3. - *Simulated Data. Comparison between the cross validation error given by scaled SVM and the Stacking scheme on the datasets generated by the experimental design. N=120, different level of degree of separation and number of variables.*

	Stacking scheme					SVMscaled			
	n. variables					n. variables			
δ	3	5	7	10	δ	3	5	7	10
0,5	0,3545	0,3237	0,2846	0,2467	0,5	0,3347	0,3032	0,2657	0,2274
1	0,2095	0,1515	0,1080	0,0691	1	0,1945	0,1388	0,0971	0,0608
1,5	0,1074	0,0576	0,0316	0,0153	1,5	0,0972	0,0470	0,0245	0,0093
2	0,0507	0,0187	0,0071	0,0019	2	0,0402	0,0124	0,0042	0,0006
2,5	0,0201	0,0035	0,0012	0,0001	2,5	0,0139	0,0018	0,0006	0,0000
3	0,0073	0,0012	0,0001	0,0001	3	0,0044	0,0002	0,0000	0,0000

On going back to Table 2 we notice that apart from the above-mentioned very good behaviour of scaled SVM, generally speaking the parametric classifiers

reach some of the best levels of accuracy, while "weak" classifiers, such as Classification Tree and the ensemble methods such as Bagged Tree and Adaboost achieve rather disappointing performances.

As far as the analysis of results relative to the measure that counts the number of times an algorithm performs better than the others (over 100 iterations) is concerned, Table 4 shows that for a low level of degree of separation, scaled SVM achieves the best relative positioning compared to the other classifiers, while with a higher degree of separation LDA achieves by far the best positioning.

TABLE 4. - *Simulated Data. Best positioning over 100 iterations for each classifier and Stacking scheme for different level of degree of separation and number of variables. N=120*

	δ=0.5				δ=3			
	n. variables				n. variables			
Classifier	3	5	7	10	3	5	7	10
LDA	16	14	17	15	75	98	99	100
QDA	4	4	3	1	9	1	0	0
TRE	2	0	0	0	1	0	0	0
TRE1	1	0	0	0	0	0	0	0
BAG	1	5	3	1	2	0	0	0
ADA	7	1	1	3	3	0	1	0
ADAm	3	7	6	2	0	0	0	0
NBA	12	9	8	8	4	0	0	0
SVM	9	8	4	15	0	0	0	0
SVMscaled	22	27	27	27	3	1	0	0
SVMb	0	0	0	0	0	0	0	0
SVMbscaled	5	5	7	7	1	0	0	0
GLM	6	6	13	11	1	0	0	0
STA	12	14	11	10	1	0	0	0

This result is also confirmed for N=200. For datasets with intermediate degrees of separation (≥1.5) the best positioning is always achieved by LDA, above all if associated (in the lower values) with a higher complexity due to the number of variables.
By comparing Table 2 and Table 4 (and more generally the whole set of results), we can observe that there is not always a correspondence for the classifiers between the best positioning achieved and the lowest value of cross validation error achieved and also because, since we are dealing with an average of 100 iterations, the variability of a classifier is significant in terms of the error returned which is often very high. It may therefore happen that classifiers with a higher variability can achieve better positioning.
This circumstance should lead to us to continue looking for more adequate measurements (at some point combining the use of more than one index) which are able to capture accuracy in the best possible way in terms of

estimating the prediction error returned by the single classifiers also in order to improve the comparison with the use of a more complex scheme like Stacking.

4.3 *Simulated contaminated data*

The second application of the Stacking procedure to simulated data was carried out on datasets generated with the same characteristics as the ones used in the previous section but they have undergone a contamination design which included a shift contamination with different levels of contaminated data, carried out on only one class or both and with different values of the constant to be added to the original data. Let's summarize the main characteristics of the contamination design:

Level (proportion) of contaminated data: 5%; 10%; 30%
Number of classes : one; both
Value of the constant to be added: +2; +4

They were generated by means of the contamination design and about 480 datasets were analysed.

In this section we are interested in investigating empirically the performances of the single classifiers and of Stacking for datasets upon which a contamination has been carried out and above all in seeing if it is convenient to use Stacking in terms of improving performances instead of a single classifier, bearing in mind the necessary increase in computation.

In the case of contaminated data Stacking improves its performances noticeably compared to what we have observed for non-contaminated data, in some cases also in comparison with scaled SVM, and generally appears to be very competitive, above all when the contaminations are more substantial. In a set of classifiers in which there was no SVM it would be the best for each of the analysed datasets. While scaled SVMbest achieves cross validation error values that are always very substantial and therefore are definitely not to be inserted with the current parameter setting (selected by means of cross validation) in a basic set of classifiers.

It will be advisable to improve the tuning of the parameters if the decision is taken to use it. It was inserted in order to make the analysis complete, but because of its very bad performances, it will not be taken into account in the comparison with the other classifiers. We will distinguish the hypotheses in which the contamination on the total of the observations is moderate (10%) from the hypotheses in which it is stronger (30%) and is carried out on one or both the classes. Table 5 illustrates the results relative to the average cross validation error for each base classifier and for Stacking for different levels of contamination with different degrees of separation between the

groups and for different number of variables of the input datasets. The contamination is only intended for one class: cont=+ 4. N=120.

When only one class is contaminated with a proportion of 10%, the behaviour of Stacking is very good and is only exceeded by three implementations of SVM. On the contrary, the effect of the contamination is quite substantial for Linear Discriminant Analysis and Logistic Regression, as well as for scaled SVMbest, which we have already said will no longer be inserted into the set of classifiers, as we prefer SVM best or scaled SVM.

TABLE 5. - *Simulated contaminated data. Cross validation error for 13 base classifier and for Stacking for different levels of contamination, different degrees of separation between the groups and for different number of variables of the input datasets. The contamination is only intended for one class: N=120.*

	10%				30%			
	δ=0.5		δ=3		δ=0.5		δ=3	
	n. variables				n. variables			
Classifier	3	10	3	10	3	10	3	10
LDA	0,5092	0,5007	0,1211	0,1220	0,5083	0,4667	0,4667	0,4625
QDA	0,3789	0,3377	0,1210	0,1098	0,5250	0,5667	0,2240	0,1714
TRE	0,3853	0,3502	0,1456	0,1397	0,3750	0,3167	0,2416	0,2161
TRE1	0,3903	0,3502	0,1426	0,1446	0,3833	0,3500	0,2426	0,2186
BAG	0,3631	0,2708	0,1131	0,0698	0,2583	0,2583	0,2098	0,1045
ADA	0,3845	0,3618	0,1450	0,1414	0,3083	0,2750	0,2513	0,2073
ADAm	0,3531	0,3066	0,1337	0,1051	0,2583	0,2750	0,2235	0,1672
NBA	0,3945	0,3928	0,1243	0,1009	0,5000	0,5000	0,2715	0,2334
SVM	0,3073	0,2201	0,0858	0,0363	0,2250	0,1250	0,1590	0,0597
SVMscaled	0,3022	0,2067	0,0788	0,0333	0,2417	0,1167	0,1541	0,0510
SVMb	0,3073	0,2201	0,0858	0,0363	0,2250	0,1250	0,1590	0,0597
SVMbscaled	0,4873	0,4733	0,1352	0,0930	0,4917	0,4750	0,4153	0,3868
GLM	0,5072	0,4994	0,1362	0,1597	0,5083	0,4667	0,4728	0,4646
STA	0,3248	0,2145	0,0876	0,0362	0,2417	0,1167	0,1681	0,0552

By increasing the degree of separation between the groups, we find quite uniform behaviour for the datasets with a moderate number of variables, with the exception of SVM and Stacking, while the application to a dataset with a higher number of variables greatly improves the performances of Naive Bayes, of the three best implementations of SVM and of Stacking.

We will see that by contaminating only one class with a level of 30% there will be a very clear effect on the four parametric base classifiers present in the set and in particular on the behaviour of NBA which, in a contamination hypothesis of 30% of the data in the presence of a low degree of separation between the groups will return a fixed error of 0.50 for every iteration. Stacking achieves an error level equal to that of the best classifier, which is scaled SVM. In the presence of a higher degree of separation (δ=3), the worst performance is given dramatically only by LDA and Logistic Regression.
Moving on to the hypothesis of contaminating two classes, in Table 6 we can see that with a moderate contamination and a low degree of separation between the classes, the four parametric classifiers in any case achieve a high level of error, although they are not among the worst.

TABLE 6. - *Simulated contaminated data. Cross validation error for 13 base classifier and for Stacking for different levels of contamination, different degrees of separation between the groups and for different number of variables of the input datasets. The contamination is intended for two class: N=120. Cont=+4.*

	10%				30%			
	δ=0.5		δ=3		δ=0.5		δ=3	
	n. variables				n. variables			
Classifier	3	10	3	10	3	10	3	10
LDA	0,4254	0,4583	0,0883	0,1094	0,4667	0,4583	0,3288	0,3033
QDA	0,4633	0,4873	0,0927	0,1843	0,4667	0,5750	0,3623	0,3552
TRE	0,4306	0,3973	0,1054	0,1195	0,4750	0,3833	0,1607	0,1610
TRE1	0,4303	0,3881	0,1036	0,1188	0,4417	0,3917	0,1564	0,1558
BAG	0,4002	0,3053	0,0736	0,0525	0,4250	0,3000	0,1352	0,0845
ADA	0,4109	0,3723	0,0904	0,0904	0,3917	0,3583	0,1605	0,1584
ADAm	0,3951	0,3547	0,0903	0,0748	0,4083	0,4000	0,1554	0,1368
NBA	0,4309	0,4373	0,0758	0,0509	0,5000	0,5000	0,3921	0,4523
SVM	0,3610	0,2357	0,0543	0,0268	0,3417	0,2250	0,0966	0,0383
SVMscaled	0,3479	0,2379	0,0489	0,0234	0,3667	0,2333	0,0927	0,0331
SVMb	0,3610	0,2357	0,0543	0,0268	0,3417	0,2250	0,0966	0,0383
SVMbscaled	0,4494	0,3653	0,2683	0,0712	0,5667	0,3750	0,2195	0,2116
GLM	0,4204	0,4524	0,0703	0,0873	0,4500	0,4500	0,2563	0,2689
STA	0,3532	0,2377	0,0580	0,0245	0,3667	0,2167	0,1020	0,0376

With a higher level of contamination for the two classes and a low degree of separation between the groups, they are always among the worst, and generally speaking none of the classifiers gives a good performance.
The behaviour of Stacking is always very interesting and overall is preferable for (δ=0.5, v=10), thus confirming the analogous result obtained also in the case of the contamination only of a single class in Table 5. In the case of a higher degree of separation (δ=3), the worst performance is achieved once again by Linear Discriminant Analysis and Logistic Regression.
Generally speaking, based on the outcome of the experiments carried out, it seems to be that the contamination contained by a single class causes a deterioration in the performances of LDA and Logistic Regression, while with a higher level of contamination this only happens with a high degree of separation between the groups. Where there is a low degree of separation, the worst performance will be achieved for all four of the parametric classifiers.

TABLE 7. - *Simulated contaminated data. Cross validation error for 13 base classifier and for Stacking.*

	10%				30%			
	δ=0.5		δ=3		δ=0.5		δ=3	
	n. variables				n. variables			
Classifier	3	10	3	10	3	10	3	10
LDA	0,4171	0,4443	0,0817	0,0861	0,4900	0,4659	0,3330	0,3199
QDA	0,4498	0,4807	0,0849	0,1329	0,4700	0,4792	0,3702	0,3470
TRE	0,4255	0,3592	0,0974	0,0943	0,4200	0,3419	0,1466	0,1357
TRE1	0,4215	0,3633	0,0981	0,0947	0,4450	0,3427	0,1483	0,1381
BAG	0,3990	0,2860	0,0684	0,0461	0,4750	0,2520	0,1230	0,0669
ADA	0,4017	0,3437	0,0884	0,0763	0,4000	0,3193	0,1451	0,1221
ADAm	0,3955	0,3307	0,0882	0,0656	0,4200	0,3226	0,1394	0,1148
NBA	0,4236	0,4400	0,0734	0,0505	0,5000	0,5000	0,3934	0,4556
SVM	0,3513	0,2299	0,0523	0,0234	0,3250	0,1992	0,0908	0,0330
SVMscaled	0,3489	0,2240	0,0495	0,0211	0,3400	0,1888	0,0891	0,0294
SVMb	0,3513	0,2299	0,0523	0,0234	0,3250	0,1992	0,0908	0,0330
bscaled	0,4375	0,3922	0,1483	0,0501	0,5650	0,5088	0,1987	0,1706
GLM	0,4130	0,4347	0,0649	0,0666	0,4800	0,4636	0,2562	0,2609
STA	0,3497	0,2220	0,0537	0,0231	0,3100	0,1850	0,0959	0,0316

The behaviour of Naive Bayes is particularly interesting, since at a high level of contamination, carried out both on one class and on both using the

hypothesis of a low degree of separation, it always obtains a value equal to 0.50 which remains the same for each iteration as can be seen in Table 5 and in Table 6, so using it is not very effective.

Table 7 summarises the results relative to the average cross validation error for each base classifier and for Stacking for different levels of contamination (10% and 30%), different degrees of separation between the groups, different numbers of variables of the input datasets. The contamination is carried out on both the classes. N=200. Cont= +4.
In comparison with non-contaminated data, we can observe that Stacking does not seem to be affected by contamination in the same way as scaled SVM does. In fact, Stacking is the best classifier in the hypothesis with a low degree of separation, both in the case of contamination at 10% (for v=10) and at 30%. Instead for all the other classifiers the effect is quite considerable, as can be seen in Figure 4 which can be compared with Figure 3 relative to the dataset generated by the experimental design but not contaminated.

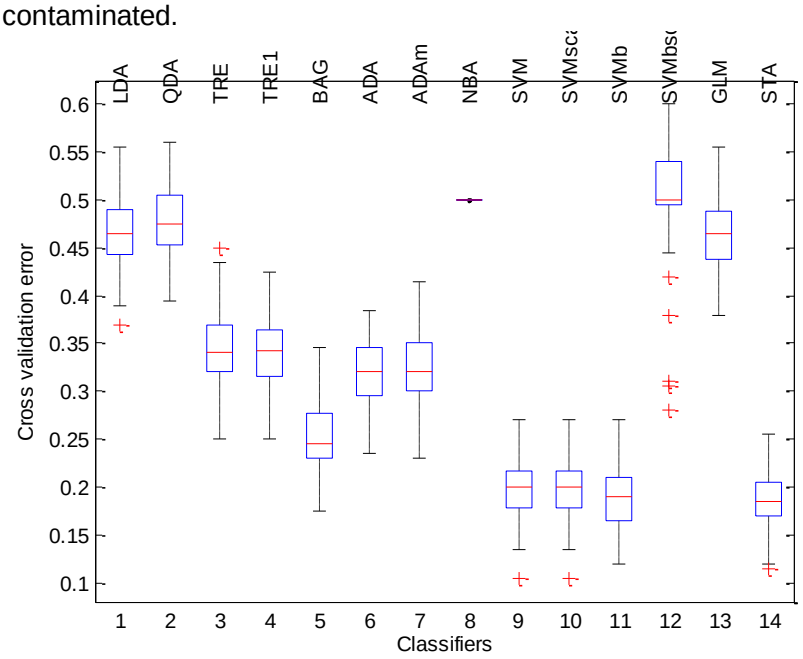

FIGURE 4. - *Simulated contaminated data. Boxplots of error distribution on 13 base classifiers and Stacking scheme. N=200, v=10,δ=0.5, cont=+4, level=30%.*

4.4 Are the number and type of Base Classifiers important?

An ensemble of classifiers consists of a set of different classification algorithms and a function to combine their individual outputs in order to improve accuracy.
Any classifier may be used in a Stacking learning scheme and any number of classifiers may be used.
For this work we have used a fixed scheme with 13 base-level classifiers.
In this Section, we are interested in investigating whether the performance of Stacking may be modified if we use different base-level classifier subsets on datasets with different characteristics.
To this end, we have created three different Stacking variants with different base-level classifier subsets built on different datasets.
The three subsets are differentiated according to the number of classifiers chosen and to their characteristics:

- subset I: composed of 3 classification algorithms : Tree Bagged, AdaboostM1 and SVMbest. The relative Stacking scheme is identified by **"STA3".**
- subset II: composed of 4 classification algorithms: 2 different variants of Classification Tree, Tree Bagged and AdaboostM1. The relative Stacking scheme is identified by "**STA4".**
- subset III: composed of 6 classification algorithms: Linear Discriminant Analysis,Quadratic Discriminant Analysis, Tree Bagged, AdaboostM1, Naïve Bayes Classifier and Logistic Regression. The relative Stacking scheme is identified by **"STA6".**

In the first subset there are classifiers that may be defined as "strong", since both Tree Bagged and Adaboost are themselves members of the ensemble generated by applying a single learning algorithm (a classification tree, which is considered as a "weak" classifier) and the Support Vector Machine (in its version with the selection of the best model through special cross validation) has proved itself to be fairly stable for predictions.
In contrast, in the second subset there are two "weak" learning algorithms and two very stable ones. The reason for this choice is to investigate whether a combination of different strength classifiers can still guarantee satisfactory performances.
For the third subset, which is much more numerous, four parametric classifiers were chosen, two of which are popular but different linear methods for classification tasks (Linear Discriminant Analysis and Logistic Regression). The main difference between them is in the way the linear function is fitted to the training data. Naïve Bayes, which is a simple method, often tends to outperform more sophisticated algorithms when the training set is small.

The experimental plan for the main set of base-level classifiers was used, to which we refer for completeness. Subsequently, some examples of level-0 data will be reported for the base-level algorithms. Ten-fold stratified cross-validation was used for all the base-level classifiers. As in the previous experiments, at every step of cross validation, one part of the available data was used to fit the model, and a different part was used to estimate individual prediction error.
Every trial process was repeated 100 times and an average of the results was calculated in order to find the error of average cross validation for each classifier and relative to each experiment.
The posterior probabilities of each classifier derived from the process of cross-validation form the meta dataset for the meta-classifier.
Ten-fold stratified cross-validation is also used, repeated 100 times, for the meta classifiers, which also in this case are Linear regression and Ridge regression (mutually exclusive when hypotheses of multicollinearity recur).
For completeness, in Appendix to Chapter 4 we show tables with the results of some experiments relating to the performances of the base-level classifier sets on the different dataset input and to the Stacking for different combination schemes. Any slight differences in the performances of some classifiers in the various schemes may be due to different partitions of the cross validation. Stacking turns out to be competitive and better too when compared with other classifiers, especially when there is greater complexity in the base models, or rather when the input datasets are characterised by one dimension larger in terms of observations and variables . The effect of these circumstances is however accentuated when there is a high degree of separation between the two populations.

TABLE 8. *Cross validation error rate for the different Stacking schemes built.*

	STA3	**STA4**	**STA6**
Base Classifier Input Dataset			
120_3_05	0.3547	0.3903	**0.3472**
120_10_05	**0.2323**	0.2798	0.2399
120_3_3	**0.0048**	0.0239	0.0066
120_10_3	**0**	0.0059	**0**
200_3_05	0.3498	0.3801	**0.3450**
200_10_05	**0.2268**	0.2732	0.2324
200_3_2	**0.0434**	0.0656	0.0462
200_10_2	**0.0004**	0.0092	0.0016
200_3_3	0.0058	0.0168	**0.0016**
200_10_3	**0**	0.0025	**0**

In Table 8 and in Figure 5 some results are summarised relating to the performances of the three different Stacking schemes built with base-level classifier sets of different sizes.

The different level-0 datasets in the first column refer to the data input with which the base-level classifiers were built and their prediction errors and probabilities were estimated (via cross-validation) and which form each Stacking scheme. The input data for the meta-classifier, therefore, are always the probabilities generated by base classifiers.
By observing the results, we can say first of all that the STA4 ensemble classification method is the worst for any kind of dataset. The weakness of two of the base-level classifiers was not sufficiently balanced out by the presence of the two ensemble classifiers, which are more stable. In this case, Stacking was unable to fully exploit the predictive capability of the stronger classifiers to compensate for the weakness of the other two.

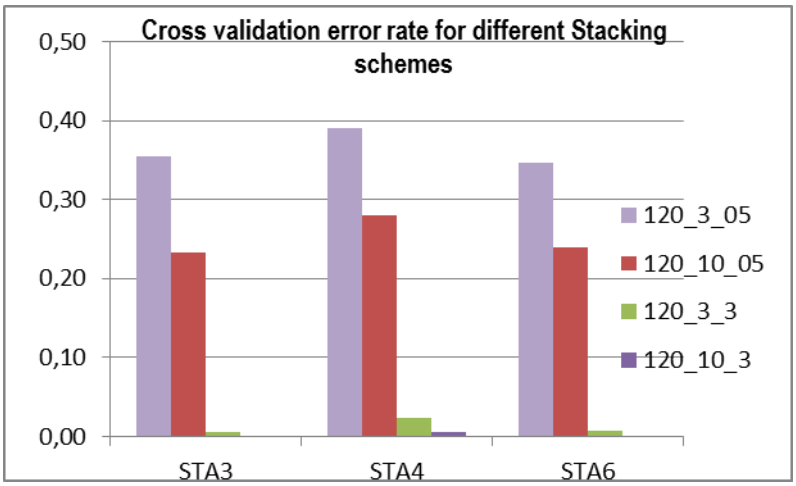

FIGURE 5. - *Cross validation error rate for the different Stacking schemes built and different input dataset.*

As far as the other two schemes are concerned, there is very little difference between them, although STA3 shows a higher success rate than the more complex STA6.
This might lead us to think that the choice from the start of high performance base-level classifiers could increase the predictive capacities of Stacking.
Furthermore, together with a greater complexity of a combination method built with a greater number of base-level classifiers, such as STA6, there is a better performance, also on level-0 datasets which express less complexity in terms of the number of variables and the degree of overlapping between the two populations in order to signify the weight expressed by the parametric base-level classifiers in their prediction.

However, a comparison shows that STA3 is more successful (although the differences are slight), thus representing the proposal with the best performances, which is also economical in its combination of the presence of classifiers with different characteristics with a lesser complexity of the model.

4.7 *Some remarks*

In this chapter we have summarized the main results obtained from the application of the proposed Stacking scheme to datasets generated by means of the experimental design and also real data (Appendix to Chapter 4). An analysis of the results relative to the set of the datasets generated by means of the simulation study, does not lead us to believe that the Stacking scheme is preferable in terms of performances to the use of the best single classifier. It always achieves good performances and is to be considered among the best, but it does not seem to be preferable for this type of application. In the case of contaminated data, Stacking improves its performances noticeably compared to what we have observed for non-contaminated data, in some cases also in comparison with scaled SVM, and generally appears to be very competitive, above all when the contaminations are more substantial. In a set of classifiers in which there was no scaledSVM it would be the best for each of the analysed datasets. The results obtained using three different Stacking variants with different base-level classifier subsets built on different datasets show that the choice from the start of high performance base-level classifiers could increase the predictive capacities of Stacking, and however, a comparison shows that the Stacking scheme with three base classifiers, is more successful (although the differences are slight). There would seem to be a confirmation of the circumstance that a greater complexity of the meta model does not improve results also for the applications to real data : in fact the best Stacking performances were achieved (at least in the examples analysed) with schemes with a lower number of base classifiers STA3 and STA6.
We can observe, moreover, that there is not always a correspondence for the classifiers between the best positioning achieved and the lowest value of cross validation error achieved and also because, since we are dealing with an average of 100 iterations, the variability of a classifier is significant in terms of the error returned which is often very high. It may therefore happen that classifiers with a higher variability can achieve better positioning. Because of the variability and fluctuation of the cross validation error, the average does not seem to be sufficient as a measurement and this circumstance should lead to us to continue looking for more adequate measurements (at some point combining the use of more than one index) which are able to capture accuracy in the best possible way in terms of estimating the prediction error returned by the single classifiers also in order

to improve the comparison with the use of a more complex scheme like Stacking.

5. CONCLUSIONS AND FUTURE DEVELOPMENTS

In this work we were interested to investigate the predictive accuracy of one of the most popular learning schemes for the combination of supervised classification methods: the Stacking Technique proposed by Wolpert (1992), in particular, we made reference to the StackingC ensemble scheme as a starting point for our analysis, to which some modifications and extensions were made.

We also focus on an aspect that has been covered much less by the studies and that in our view, however deserves special attention: the choice of the initial dataset. This is connected to the assumption that small changes in the dataset can lead to different models and that the presence of outliers might alter the parameters of the model.

We were interested in investigating this aspect and in extending certain distinct elements in the Stacking scheme, as well as examining some characteristics neglected by the literature and also in proposing some distinct elements of the Stacking scheme. Starting from the recent advances proposed by the literature for the Stacking framework, the whole Stacking scheme was therefore implemented in Matlab and built in the form of an interlinked process which is begun by the generation of the level-0 data in the case of the simulation study, and which includes a complete homogenisation of the procedures relative to each of its phases in order to guarantee uniformity and therefore comparability of the outputs returned at every step. At the moment, of course the scheme is in an experimental phase and will definitely have to be improved both in terms of the choice of classifiers to be inserted in the set and of the setting of its parameters in order to optimise performances.

Since we are interested in investigating empirically the performances of the single classifiers and of Stacking for datasets with different characteristics and above all if it is convenient to use Stacking in terms of improving performances instead of a single classifier, bearing in mind the necessary increase in computation, some applications of the proposed scheme were carried out both on simulated and real data.

An analysis of the results obtained by applying the proposed Stacking scheme to the set of the datasets generated by means of the experimental design does not lead us to believe that the prediction error of the Stacking scheme is to be considered lower than any other classifier or that, therefore, the Stacking scheme is preferable in terms of performances to the use of the best single classifier.

It always achieves good performances and is to be considered among the best, but it does not seem to be preferable for this type of application, such

as Linear Discriminant Analysis and Logistic Regression, while the behaviour of scaled SVM is particularly interesting, as it is has the lowest error among the classifiers, also in comparison with Stacking, LDA and Logistic Regression.

In a set of classifiers in which there was no SVM it would be the best for each of the analysed datasets. In the case of contaminated data Stacking improves its performances noticeably compared to what we have observed for non-contaminated data, in some cases also in comparison with scaled SVM, and generally appears to be very competitive, above all when the contaminations are more substantial and especially in the presence of strong contaminations also for both classes.
On the contrary, the effect of the contamination is quite substantial for Linear Discriminant Analysis and Logistic Regression. Generally speaking, based on the outcome of the experiments carried out, it seems to be that the contamination contained by a single class causes a deterioration in the performances of LDA and Logistic Regression, while with a higher level of contamination this only happens with a high degree of separation between the groups. Where there is a low degree of separation, or both classes are contaminated, the worst performance will be achieved for all four of the parametric classifiers.

A further element of interest in our research was whether the number and typologies of the algorithms chosen were important for the composition of the set of base classifiers, since the literature is quite unequivocal in maintaining that any classifier can be used in a Stacking scheme. Different Stacking schemes have been created for different input datasets. The main results obtained from the application of the different Stacking schemes show that the choice from the start of high performance base-level classifiers could increase the predictive capacities of Stacking, and however, a comparison shows that the Stacking scheme with three base classifiers, is more successful (although the differences are slight). There would seem to be a confirmation of the circumstance that a greater complexity of the meta model does not improve results also for the applications to real data : in fact the best Stacking performances were achieved (at least in the examples analysed) with schemes with a lower number of base classifiers STA3 and STA6.

As far as the application to real data is concerned, the analysis was carried out on different datasets, and the results of one of them are illustrated (Appendix to Chapter 4). It is a very complex dataset containing the results of the inspection surveys carried out by INPS (National Social Security Institute) on Italian companies in order to see if there was any off-the-book employment present, in and it achieves error values that are generally quite high for all classifiers, as was predictable given the level of overlap between the groups. It is easy to guess from the structure of the

data that there are anomalous observations which are in the group represented by the companies that declare an absence of any off-the-book employment. Any anomalous values could give rise to signals of the potential presence of off-the-book employment in those companies. In both cases, however, Stacking seems competitive when compared to the use of ensemble methods (Bagging and Adaboost), but not preferable in terms of performances to the use of the best single classifier apart the STA 6 scheme.

With a view to a further improvement of the entire proposed process, which is at the experimental stage, the research activity will be directed towards optimising performances and guaranteeing the reliability of the predictions for single classifiers by modifying the setting of the parameters used in this phase, and more generally by including:

- Extension of the experimental design both to verify further the results achieved and to insert other elements into the design (different processes of data generation, increasing the number of classes, different *prior* values).
- Introduction of more adequate measurements (at some point combining the use of more than one index) which are able to capture accuracy in the best possible way in terms of estimating the prediction error returned by the classifiers.
- Possible introduction of a weighting system into the method of meta-classification should we intend to combine several classifiers with very different performances in terms of accuracy.
- Extension of the methods proposed by the literature (Varma et al., 2006; Tibshirani et al., 2008) for the estimation and reduction of potential *bias* in cross validation error for the problem that is the object of the work.

APPENDIX to Chapter 3

This Appendix gives the traditional formalization of the four parametric algorithms used to induce the respective base classifiers. We recall, therefore, some notations and concepts commonly used for dealing with classification problems

- **Linear Discriminant Analysis**

In a $k-class$ classification problem we need to know the class posterior probabilities $\Pr(G \mid X)$ for optimal classification, where X is a casual p-dimensional variable, and G is a casual categorical variable that represents the to which an individual belongs.
The overall population is made up of K classes and we suppose $f_k(x)$ is the class-conditional density of X in class $G=k$, and let π_k be the prior probability of class k, with $\sum_{k=1}^{k} \pi_k = 1$.
Therefore, the density for the overall probability is:

$$f_k(x) = \sum_{k=1}^{K} f_k(x)\pi_k .$$

By application of the Bayes theorem:

$$\Pr(G = k / X = x) = \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{k} f_\ell(x)\pi_\ell} \qquad (1.1)$$

We will have to know or evaluate from the data π_k and $f_k(x)$. In a parametric context we can suppose the hypothesis that we model each class density as multivariate Gaussian $N_p(\mu_k, \Sigma_k)$, so that the result is:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}\left|\Sigma_k\right|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \qquad \text{for } k = 1,\ldots,K \qquad (1.2)$$

Linear discriminant analysis (LDA) arises in the special case when we assume that the classes have a common covariance matrix $\sum_k = \forall k$.
The *linear discriminant functions (LDA)*

$$\delta_k(x) = x^T \Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} + \log \pi_k \qquad (1.3)$$

are an equivalent description of the decision rule, with $G(x) = \arg\max_k \delta_k(x)$. Generally, we do not know the parameters of the Gaussian distributions. We will need to estimate them from the data as follows:

- $\pi_k = n_k / n$
- $\mu_k = \Sigma_{g_i=k} x_i / n_k$
- $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{g_i=k} = \left(x_i - \mu_k\right)\left(x_i - \mu_k\right)^T / \left(n - K\right)$

For a two-class problem there is a correspondence between Linear Discriminant Analysis and classification by linear least squares. The decision rule is assigned to class 2 if

$$x^T \hat{\Sigma}^{-1}\left(\mu_2 - \mu_1\right) > \frac{1}{2}\mu_2^T \hat{\Sigma}^{-1}\mu_2 - \frac{1}{2}\mu_1^T \hat{\Sigma}^{-1}\mu_1 + \log\left(n_1 / n\right) - \log\left(n_2 / n\right)$$

and class 1 otherwise. Suppose we code the targets in the two classes as *+1* an *-1* respectively.
With more than two classes, LDA is not the same as linear regression of the class indicator matrix, and it avoids the masking problems associated with that approach (Hastie et al., 1994).

- **Quadratic Discriminant Analysis**

If, on the other hand, the Σ_k are not assumed to be equal, from the expression of density (1.2) we then get *quadratic discriminant functions* (QDA),

$$\delta_k(x) = \frac{1}{2}\log|\Sigma_k| - \frac{1}{2}\left(x - \mu_k\right)^T \Sigma_k^{-1}(x - \mu_k) + \log \pi_k \qquad (1.4)$$

The estimates for μ_k for QDA are similar to those for LDA, while separate covariance matrices must be estimated for each class:

$$\hat{\Sigma}_k = \Sigma_{g_i=k}\left(x_i - \mu_k\right)\left(x_i - \mu_k\right)^T / \left(n_k - 1\right) \qquad (1.5)$$

Unlike the Linear Discriminant Analysis, QDA is closely linked to the Gaussian distributive hypothesis.

- **Naïve Bayes**

Naïve Bayes models are a variant of the previous case, and assume that each of the class densities are products of marginal densities; that is, they assume that given a class $G = j$, the features X_k are independent:

$$f_j\left(X\right) = \prod_{k=1}^{p} f_{jk}\left(X_k\right) \qquad (1.6)$$

This assumption is often not true, but it simplifies the estimation.

The individual class-conditional marginal densities f_{jk} can each be estimated separately using one-dimensional kernel density estimates.

This is in fact a generalization of the original *Naïve Bayes* procedures,which used univariate Gaussians to represent these marginals. Naïve Bayes uses a Laplacian estimate for estimating the conditional probabilities for each nominal attribute to compute f_{jk} .

For each continuous-valued attribute, a normal distribution is assumed in which case the conditional probabilities can be conveniently represented entirely in terms of the mean and variance of the observed values for each class.

- **Logistic Regression**

The specific form of the Logistic Regression model for the posterior probabilities $P_k(x)$ via linear functions in *x,* while at the same time ensuring that they sum to one and remain in [0, 1] if there are two classes is:

$$P(G=1 \mid X=x)=\frac{e^{(\beta_0+\beta^T x)}}{1+e^{(\beta_0+\beta^T x)}}$$

$$P(G=2 \mid X=x)=\frac{e^{(\beta_0+\beta^T x)}}{1+e^{(\beta_0+\beta^T x)}}$$

The *logit* transformation in terms of $P_k(x)$ is defined : $g(x)=\log\left[\frac{p}{(1-p)}\right]$

and the model has the form:

$$\log\frac{P(G=1 \mid X=x)}{P(G=2 \mid X=x)}=\beta_0+\beta^T x \qquad (1.7)$$

The importance of this transformation is that $g(x)$ has many of the properties of a linear regression model. The logit $g(x)$ is linear in its parameters, may be continuous, and may range $-\infty$ to $+\infty$, depending on the range of *x*.

- **Bagging** (Bootstrap aggregation)

Given a learning set $L = \{(y_n, x_n), n = 1, ..., N\}$, where the *y* are class labels, we assume to build a model on it obtaining the prediction $f(x)$ at input *x*.
Bootstrap aggregation (or *Bagging)* averages this prediction over a collection of bootstrap samples $\{L^{(B)}\}$ from L.

For each bootstrap sample $L^{(B)}$ we fit our model, that returns prediction $f^{(B)}(x)$. The bagging estimate is defined by:

$$f_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} f^{(B)}(x) \qquad (1.8)$$

If *y* is a class label, let the $f^{(B)}(x)$ vote to form $f_{bag}(x)$.
The $\{L^{(B)}\}$ form replicate data sets, each consisting of *N* cases, drawn at random, *but with replacement,* from *L*. Each (y_n, x_n) may appear repeated times or not at all in any particular $L^{(B)}$.
The $\{L^{(B)}\}$ are replicate data sets drawn from the bootstrap distribution approximating the distribution underlying L.

A critical factor in whether bagging will improve accuracy is the stability of the procedure for constructing f. If changes in *L*, i.e. a replicate *L*, produces small changes in f, then f_{bag} *B* will be close to f. Improvement will occur for unstable procedures where a small change in L can result in large changes in f. (Breiman 1996).

- **AdaBoost**

Boosting is a general method for improving the performance of any learning algorithm. Schapire introduced the first boosting algorithm in 1990. In 1995, Freund and Schapire introduced the AdaBoost algorithm. In this thesis, we refer to AdaBoostM1 (Freund and Schapire,1996).

The algorithm assumes a training set consisting of *m* instances $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ where x_i is a vector of attribute values and $y_i \in Y$ is the class label associated with x_i. The boosting algorithm call another unspecified learning algorithm (called WeakLearn) repeatedly in a series of rounds. The purpose of the boosting is to apply the weak learner to repeatedly modified version of the data, producing a sequence of weak classifiers h_t and the predictions from all of them combined through a weighted majority vote to give the final prediction that minimizes the error. On round *t*, therefore, the booster provides WeakLearn with a distribution D_t over the training set S and in response it computes a classifier which should correctly classify a fraction of the training set that has large probability with respect to . The process is carried out for $t = 1, 2, ..., T$ and in T the booster combines all weak classifiers into a final classifier h_{fin}.

Algorithm AdaBoost.M1

Input: $S = \{(x_1, y_1), ..., (x_m, y_m)\}$

WeakLearn

T (number of iterations)

Inizialize $D_1(i) = 1/m; i = 1, ..., m$

For t =1 to T

1. Fit a classifier h_t using WeakLearn and distribution D_t
2. Compute error of h_t: $\varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$

   If $\varepsilon > 0.5$ then $T \leftarrow t - 1$ exit Loop
3. $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$
4. Update distribution D_t

   $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & if\ h_t(x_i) = y_i \\ 1 & otherwise \end{cases}$$

   Z_t is normalization constant in order to be D_{t+1} a distribution

Output the final classifier

$$h_{fin}(x) = \arg\max_{y \in Y} \sum_{i:h_t(x_i) = y_i} \log \frac{1}{\beta_t}$$

APPENDIX TO CHAPTER 4

1. APPLICATION TO REAL DATA.

This part of the work is dedicated to the application of the proposed Stacking scheme to real data. The main results of the application of the Stacking Scheme to the dataset will be reported below.

First of all, in connection with what we covered in the previous section, that is to say the use of different Stacking schemes, we subjected the dataset to the three different Stacking schemes.

In the first scheme we always used a fixed set with 13 base classifiers **STA13,** which represents the scheme of reference that we applied for the simulation study.

On the basis of the experiments performed and the above study carried out for the thesis on the performance of different sets of classifiers, we also decided to use the set with 6 base classifiers **STA6** and the one with three classifiers, **STA3**, with the same composition features reported in the previous section.

We decided to do without STA4, which does not seem to give results that are worse overall compared to the others.

We always used linear least squares regression as meta-classifiers and always ridge regression with the mechanism of mutual exclusivity that has already been described.

1.2 *The dataset Off-the-book employment*

The application of the proposed Stacking Scheme to real data was carried out on a data sample taken from a dataset containing the results of the inspection surveys carried out by INPS (National Social Security Institute) on Italian companies in order to see if there was any off-the-book employment present. In its original version this dataset is extensive and rather complex, consisting of 14,651 records divided into two non-balanced groups, and 39 variables that are both continuous and qualitative.

The data used in the analysis are taken from 230 records extracted from the original dataset, respecting, where possible, the proportions of the data originally present in each group, and we have selected 4 variables.

Furthermore, because of the marked level of overlap between the two groups and the presence of collinearity between variables, these variables have been transformed by taking logs.

n_1 =126

n_2 =104

Identification of the variables used:

Org4 = No. paid days of unskilled workers per unit of total paid day

Dim8 = Total paid days
Pers13 = employee expenses per employee (Asia)
Pers16 = Productivity per employee
Label class = Absence/Presence of the off-the-book workforce during the last INPS inspection

Three different Stacking schemes were used for this case too:

Scheme 1) 3 base classifiers with 4 explanatory variables + LR/RR as mutually exclusive meta-classifiers
Scheme 2) 6 base classifiers with 4 explanatory variables + LR/RR as mutually exclusive meta-classifiers
Scheme 3) 3 base classifiers with 4 explanatory variables + LR/RR as mutually exclusive meta-classifiers

Since the dataset does not come from our simulation plan but refers instead to real data, and since there has also been a transformation of the original variables, it is interesting to observe Figure 1 which shows the scatter plot matrix.

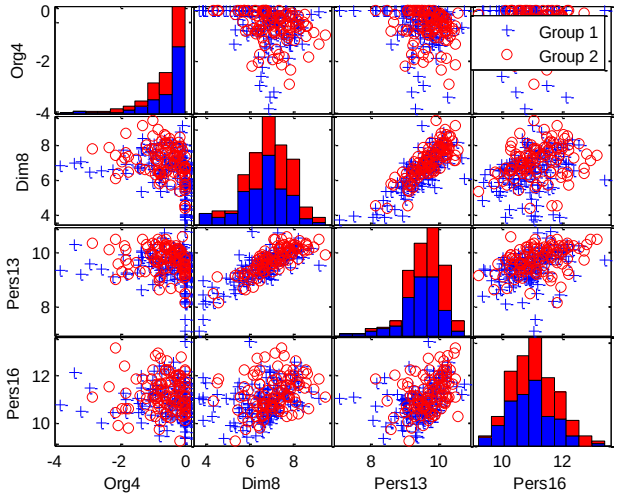

FIGURE 1. - *Off-the-book employment Data. Scatterplot matrix with bivariate scatters of the three variables and histograms on the main diagonal. Units in Group 1 are represented by blue crosses*

Both the conditions of extensive collinearity between the variables and of the degree of overlap between the groups seem to have improved compared to the original situation, although the overlap remains. There do not seem to be any variables that make a perfect separation possible between the two groups.

We now move on to the analysis of the main results for each proposed Stacking Scheme:

1) Scheme STA13

In the scheme made up of 13 base classifiers, as shown in Table 1 below, the average cross validation error stays quite high for all the classifiers, as was predictable given the level of overlap between the groups.
Having chosen to carry out the evaluation and comparison of the classifiers by estimating cross validation (medium in this case), but also with the aim of broadening the representation of the error distribution, we included the calculation of other indicators which could improve our knowledge and the plots shown above.

TABLE 1. - *Off-the-book employment Data. Measurements of the performances of the base classifiers and the Stacking scheme STA13 calculated with reference to the respective distribution of the cross validation errors. Average values for 100 iterations.*

	Cross Validation Error	Median Cross Validation Error	Std. Deviation Cross Validation Error	Interquartile Difference Cross Validation Error	Range Cross Validation Error	MAD Cross Validation Error	% Best positioning
Classifier							
LDA	0,4347	0,4348	0,0086	0,0087	0,0348	0,0043	2
QDA	0,4653	0,4652	0,0125	0,0174	0,0652	0,0087	0
TRE	0,4701	0,4696	0,0288	0,0413	0,1609	0,0217	0
TRE1	0,4642	0,4696	0,0307	0,0457	0,1435	0,0261	5
BAG	0,4772	0,4783	0,0224	0,0261	0,1087	0,0130	0
ADA	0,4360	0,4348	0,0192	0,0261	0,0913	0,0130	14
ADAm	0,4395	0,4348	0,0220	0,0348	0,1087	0,0152	16
NBA	0,4351	0,4348	0,0093	0,0087	0,0522	0,0043	2
SVM	0,4240	0,4217	0,0110	0,0130	0,0609	0,0087	17
SVMscaled	0,4236	0,4217	0,0129	0,0130	0,0565	0,0087	21
SVMb	0,4240	0,4217	0,0110	0,0130	0,0609	0,0087	0
SVMbscaled	0,4299	0,4304	0,0132	0,0217	0,0609	0,0087	9
GLM	0,4318	0,4304	0,0081	0,0130	0,0391	0,0043	0
STA	**0,4417**	**0,4435**	**0,0286**	**0,0391**	**0,1391**	**0,0217**	**14**

When comparing the performance achieved by Stacking with those of some base classifiers (Linear Discriminant Analysis, Adaboost, Support Vector Machine, and Logistic Regression) it does not seem to be competitive

compared to the use of a single classifier. However, it should be noted that, although Stacking has quite a modest result in terms of accuracy, it manages to achieve quite an interesting result in terms of best positioning (it was the best in 14% of the iterations) which is higher than classifiers characterised by lower average values for cross validation error. With regard to this, Figure 2 and Figure 3 show the presence of a certain variability in the error distribution for some classifiers on the total of the iterations.

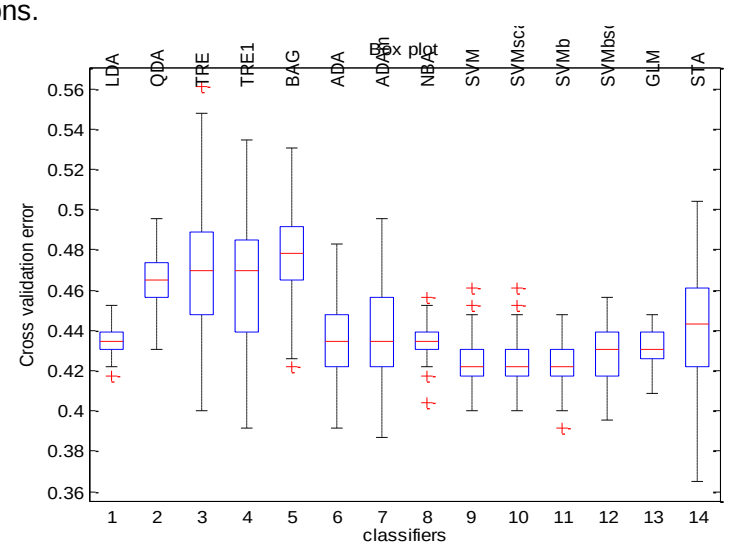

FIGURE 2. - *Off-the-book employment Data. Boxplots of error distribution of 13 base classifiers and Stacking scheme. Over 100 iterations.*

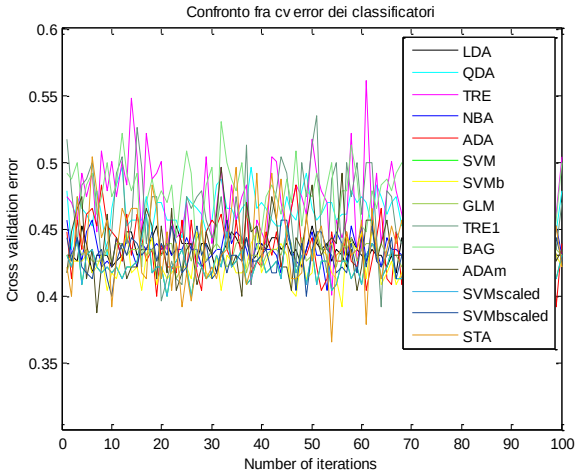

FIGURE 3. - *Off-the-book employment Data. Comparison of cross-validation error of 13 base classifiers and Stacking scheme. Over 100 iterations.*

In this sense, examples are represented by Stacking and by the Classifcation Tree, which achieve respectively the minimum and maximum values (compared to the other classifiers) of cross validation error in some iterations.

2) Scheme STA6

In the scheme with 6 base classifiers, the average cross validation error for Stacking is lower than any other single classifier and its performance is the best in terms of the number of times when it was the best classifier for the total of iterations carried out, as we can see in Table 2 below.

TABLE 2. - *Off-the-book employment Data. Measurements of the performances of six base classifiers and the Stacking scheme STA6, calculated with reference to the respective distribution of the cross validation errors. Average values for 100 iterations*.

	Cross Validation Error	Median Cross Validation Error	Std. Deviation Cross Validation Error	Interquartile Difference Cross Validation Error	Range Cross Validation Error	MAD Cross Validation Error	% Best positioning
Classifier							
LDA	0,4342	0,4348	0,0091	0,0130	0,0565	0,0043	16
QDA	0,4609	0,4609	0,0153	0,0174	0,1043	0,0087	2
BAG	0,4704	0,4696	0,0221	0,0261	0,1435	0,0130	2
ADA	0,4361	0,4348	0,0184	0,0217	0,1217	0,0130	22
NBA	0,4336	0,4348	0,0094	0,0130	0,0696	0,0043	15
GLM	0,4331	0,4348	0,0095	0,0130	0,0565	0,0087	8
STA	**0,4309**	**0,4304**	**0,0243**	**0,0304**	**0,1478**	**0,0174**	35

It would seem, therefore, that this combination of classifiers is the one that best expresses the predictive capacities of Stacking in terms of accuracy. In this case too, as shown in Figure 4, Stacking is characterised by a certain variability compared to the others, even though the average and median values for cross validation error are the lowest.

The fluctuations in cross validation error compared to some classifiers for the total number of iterations, especially for Stacking and Bagging, are shown in Figure 5.

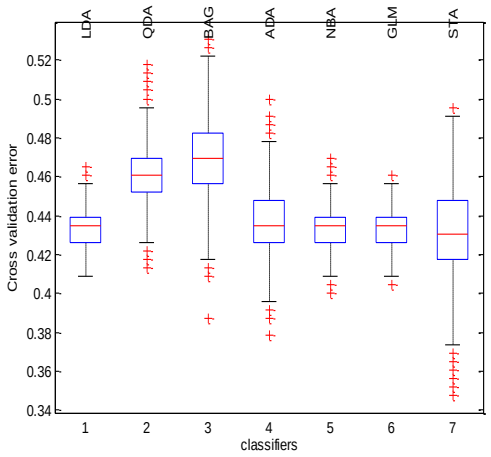

FIGURE 4. – *Off-the-book employment Data. Boxplots of error distribution of six base classifiers and STA6 scheme. Over 100 iterations.*

This variability is, of course, characterised by the fact of achieving minimum values for the former and maximum values for the latter, compared with the other classifiers which, on the contrary, appear to have quite moderate fluctuations.

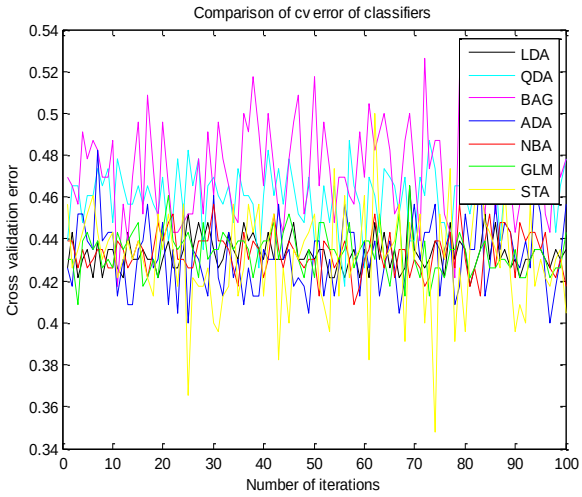

FIGURE 5. - *Off-the-book employment Data. Comparison of cross-validation error of six base classifiers and STA6 scheme. Over 100 iterations.*

3) STA3

In the scheme with three base classifiers, Stacking seems competitive when compared to the use of ensemble methods (Bagging and Adaboost), but not preferable to the use of the Support Vector Machine which has a lower error rate in the classification.

TABLE 3. - *Off-the-book employment Data. Measurements of the performances of three base classifiers and the Stacking scheme STA3, calculated with reference to the respective distribution of the cross validation errors. Average values for 100 iterations.*

	Cross Validation Error	Median Cross Validation Error	Std. Deviation Cross Validation Error	Interquartile Difference Cross Validation Error	Range Cross Validation Error	MAD Cross Validation Error	% Best positioning
Classifier							
BAG	0,4667	0,4652	0,0252	0,0304	0,1304	0,0174	1
ADA	0,4352	0,4348	0,0207	0,0261	0,0957	0,013	28
SVMb	0,4243	0,4261	0,0138	0,0174	0,0783	0,0087	51
STA	**0,4311**	**0,4304**	**0,0178**	**0,0174**	**0,1000**	**0,0087**	**20**

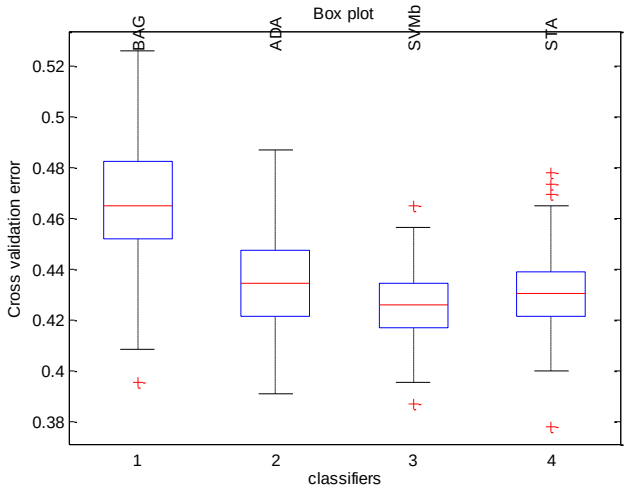

FIGURE 6. - *Off-the-book employment Data. Boxplots of error distribution of three base classifiers and STA3 scheme. Over 100 iterations.*

The variability of Stacking is more moderate compared to what we have seen in previous schemes and compared to ensemble methods, as illustrated in Figure 6 and Figure 7.

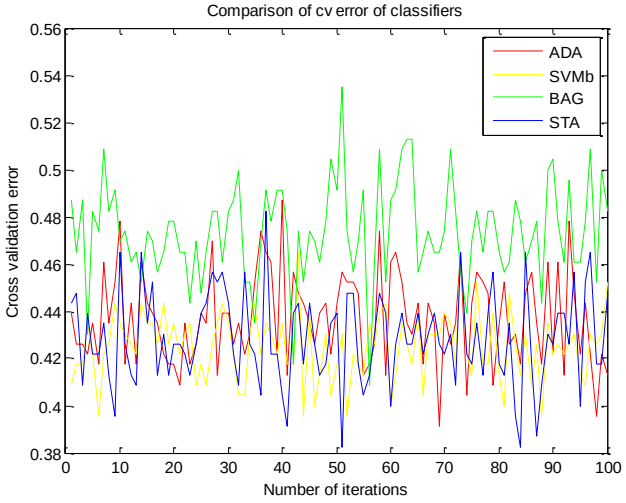

FIGURE 7. - *Off-the-book employment Data. Comparison of cross-validation error of three base classifiers and STA3 scheme. Over 100 iterations.*

If we summarise some of the main results obtained in terms of the Stacking scheme's performance, using subsets of base classifiers of different sizes and typologies, for real datasets and simulated data, we can then analyse the following Table 4.

TABLE 4.- *Off-the-book employment Data. Cross validation average error rate for different Stacking schemes and different base-level datasets.*

	STA3	STA6	STA13
Input Data 0-level			
120_3_05	0,3547	**0,3472**	0,3545
120_10_05	**0,2323**	0,2399	0,2467
120_3_3	**0,0048**	0,0066	0,0073
120_10_3	**0**	**0**	0,0001
200_3_05	0,3498	**0,3450**	0,3475
200_10_05	**0,2268**	0,2324	0,2314
200_3_2	**0,0434**	0,0462	0,0465
200_10_2	**0,0004**	0,0016	0,0016
200_3_3	0,0058	**0,0016**	0,0070
200_10_3	**0**	**0**	**0**
Off-the-book	0,4311	**0,4309**	0,4417

There would seem to be a confirmation of the circumstance that a greater complexity of the meta model does not improve results: in fact the best Stacking performances were achieved (at least in the examples analysed) with schemes with a lower number of base classifiers STA3 and STA6. We did not take the STA4 scheme into consideration, because with respect to a low second level complexity, it recorded very bad performances due to the weight of the “weak” component among its classifiers.
STA13 achieves the same level of performances as the other two schemes only with the hypothesis that both the degree of complexity and the degree of separation between the groups are at a maximum in the base-level input datasets.

REFERENCES

Boser , B.E., Guyon I., and Vapnik V. (1992), A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144-152. ACM Press

Blake, C. L., Keogh, E., Merz, C. (1998), UCI repository of machine learning databases. Department of Information and Computer Science, University of California at Irvine, Irvine CA.

Breiman, L., J. Friedman, R. Olshen, and C. Stone, (1984) *Classification and Regression Trees*. Boca Raton, FL: CRC Press.

Breiman, L., (1996), Bagging Predictors. *Machine Learning*, Vol. 24, No. 2, pp. 123-140.

Breiman, L., (1996a), Stacked Regressions. *Machine Learning*, Vol. 24, pp. 49-64.

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

Breiman, L. and Spector, P. (1992). Submodel selection and evaluation in regression: the X random case, *International Statistical Review* 60: 291–319.

Busa-Fekete R., Kégl B., Elteto t., Szarvas G. (2011), Ranking by calibrated AdaBoost. *JMLR: Workshop and Conference Proceedings 14*, 37-48.

Chang C. and Lin C., (2001), *LIBSVM, a library for support vector machines.* Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Cortes C., and Vapnik V. (1995), Support Vector network. *Machine Learning*, 20. 273-297.

Demsar J., Statistical Comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7, 1-30.

Dietterich, T. G. (1998), Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 107, 1895–1923.

Dietterich, T. G. (2000), Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems* (pp. 1–15). Berlin, Springer.

Dzeroski, S., and Zenko, B. (2004), Is combining classifiers better than selecting the best one? In *Proceedings of the Nineteenth International Conference on Machine Learning*, San Francisco: Morgan Kaufmann.

Dzeroski, S., and Zenko, B. (2002), Stacking with multi-response model trees. In *Multiple Classifiers Systems,Proceedings of the Third International Workshop*, Berlin, Springer

Freund, Y. and Schapire R.E. (1996), *Experiments with a new boosting algorithm, Proceedings of the International Conference on Machine Learning,* pages 148-156, Morgan Kaufmann, San Francisco.

Guyon I., et. al. (2006), Performance prediction challenge. *International Joint Conference on Neural Networks*. Vancouver, Canada

Hastie, T., Tibshirani, R., Friedman, J. (2003), *The Elements of Statistical Learning*. Springer, Heidelberg (2003)

Hastie T., Tibshirani R., and J. Friedman, (2009), *The elements of statistical learning* (2nd ed.). New York, Springer-Verlag.

Hoerl, A. E. and Kennard, R. (1970). Ridge regression: biased estimation for non orthogonal problems, *Technometrics* **12**: 55–67.

Kohavi R. (1995), A study of cross-validation and Bootstrap for accuracy estimation and model selection. *International Joint Conference on Artificial Intelligence*.

Kuncheva, L.I., (2004), *Combining Pattern Classifiers: Methods and Algorithms.* Wiley-Interscience, Hoboken.

LeBlanc, M., and Tibshirani R., (1993), Combining Estimates in Regression and Classification. In *Technical Report 9318*. Department of Statistics, University of Toronto.

Mardia, K., Kent, J. and Bibby, J. (1979). *Multivariate Analysis*, Academic Press.

Merz, C. J. (1999), Using correspondence analysis to combine classifiers. *Machine Learning, 36:1/2*, 33–58.

Reid S., and Grudic G. (2009), Regularized linear Models in Stacked Generalization. In Beneditksson J.A., et. al (Eds): *MCS 2009, LNCS 5519*, 112-121

Roli F., Giacinto G., and Vernazza G., (2001), Methods for designing multiple classifier systems. In MCS '01: *Proceedings of the Second International Workshop on Multiple Classifier Systems*, pages 78-87, London, UK, Springer-Verlag.

Schaffer, C., (1993), Selecting a classification method by cross-validation. *Machine Learning 13(1)* (1993) 135–143.

Seewald A.K. (2002), How to make Stacking better and faster while also taking care of an unknown weakness. In *Proceedings of the 19th International Conference on Machine Learning, ICML-2002*. Morgan Kaufmann Publisher, San Francisco.

Seewald A.K. (2002a), Exploring the Parameter State Space of Stacking, in *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM-2002)*, Maebashi City, Japan.

Tibshirami R.J., and Tibshirami R., (2009), A Bias correction for the minimum error rate in cross-validation. *The Annals of Applied Statistics*, Vol. 3, No. 2, 822-829.

Ting K. and Witten I., (1997), Stacked generalization: when does it work? In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Ting, K. M. and Witten, I. H. (1999), Issues in stacked generalization. *Journal of Artificial Intelligence Research 10*, pages 271-289.

Todorovski, L., & Dzeroski, S. (2000), Combining multiple models with meta decision trees. In *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 54–64). Berlin, Springer.

Todorovski, L., & Dzeroski, S. (2003), Combining classifiers with meta decision trees. *Machine Learning, 50:3*, 223–249.

Varma S., and Simon R., (2006), Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, 91

Witten I.H., Frank E. and Hall M., (2011), *Data Mining. Practical Machine Learning Tools and Techniques*. Third Edition. Morgan Kaufmann

Wolpert, D.H. (1992), Stacked Generalization. *Neural Networks*, Vol. 5, pp. 241-259, Pergamon Press.

Zenko, B., & Dzeroski, S. (2002), Stacking with an extended set of meta-level attributes and MLR. In *Proceedings of the Thirteenth European Conference on Machine Learning*, pp. Berlin, Springer.

Zenko, B., Todorovski, L., & Dzeroski, S. (2001), A comparison of stacking with MDTs to bagging, boosting, and other stacking methods. In *Proceedings of the First IEEE International Conference on Data Mining* (pp. 669–670). Los Alamitos, IEEE Computer Society.