Data Science Project: Arbitrated ensembling

Regression and Classification problems

Ostap Kharysh: `ostap.kharysh@etudiant.univ-rennes1.fr`
Ignacio Regana: `ignacio.regana@etudiant.univ-rennes1.fr`

ISTIC - IT & Electronics Department
Master EIT Digital in Data Science

January 18, 2022

Contents

1 Scope

The Ensemble method appeared as a technique for leveraging the decision of group of regressors. With this work we look through the performance of Arbitrated Ensemble learning for regression (including time series regression) and introduce an approach of Ensemble learning for classification. In our report we use 8 estimators for regression and classification problems. For each of the problems we select 4 publicly available datasets. The approach we use for error measurements and results obtained suggests that Arbitrated Ensemble method could be used as a technique for optimal decision making for both regression and classification problems.

2 Introduction

The basic approach to create a classification or regression model is to build it from training data containing both the target variable and the predictor variables. A model constructed in this way then allows us to predict the target variable from the predictor variables of any other structured data set such as the training data.

We can go a step further in the construction of machine learning models by means of the technique known as ensemble. This deals with "models of models": in other words, models built not directly from the data in a training dataset, but from the predictions of several models built from the training data.

That is, we first train a few traditional models (first level models); and then from the predictions of the first level models we build a second model (second level model). We usually stop there, but we could continue to build new levels of models.

In this section we will have a broader overview of what is Arbitrated Ensemble and in which cases the utilization of such approach is worthwhile. For our explanations we will consult with the research of Cerqueira, Vitor and Torgo, Luís and Pinto, Fábio and Soares [1], [2] which is the basis for our report.

2.1 Arbitrated Ensemble

The Arbitrated Ensemble technique appeared as a response to reduce the influence of non-stationary and time-evolving structures of the date for time series forecasting. The idea which puts a ground for the Arbitrated Ensemble is to combine several forecasting models. As a result there appeared approaches that analyse the errors that model produce at prediction and provide a constant weight for each of the model based on its performance, except for the inputs. Consequently, the final results were obtained by multiplying constant weights for each prediction result of the models and taking the sum of each parts. This way one could achieve an optional result for the prediction based on the error analysis. One of the famous approaches that originated from this idea was stacking [3], where the deduction of the biases of the statistical models of a learning set result of approaches of prioritizing: taking only output of the best performing model *winner-takes-all*, cutting-off the lowest performing models, etc. The major boundary of all these approaches is the assumption that each of the models performs unchanged along the whole dataset, hence, considering mean error of the model to find the best prediction result is sufficient. In contrast to this strong assumption of static behavior of the prediction model there appeared an approach of Arbitrated Ensemble that enabled a dynamic analysis of the model behavior. It provides a meta-learner strategy that enables to utilize different levels of predictors expertise across the input space, so that for each of the observation a dynamic weighting is used provided by error learning model that is further elaborated in **Architecture of Arbitrated Ensemble** part with a Figure 1.

2.2 Motivations of AE

About classical ensembly, some of the advantages of using ensemble learning methods is that they reduce the probability of overfitting to the data during the training phase and decrease both variance error (the results provided by the meta-classifier will depend less on the peculiarities of the training data) and bias error (by combining classifiers, the particularities of the training data set are better learned). All this is because, if a combination of varied classifiers is constituted from a set of training instances, they can provide complementary information regarding the patterns underlying the data and, therefore, higher accuracy in classifying new examples. However, due to the complexity of constructing this learning technique, the combination of learning methods has the disadvantage of increasing their processing time since they do not train a single classifier, but many. In order to obtain better performance when performing prediction tasks, ensemble methods are used as a machine learning technique that consists of a set of classifier (or regression) methods, in such a way that the predictions made by each of them are combined to classify a new instance (they are aggregated to form a single classifier); they are therefore called meta-classifiers. In this way, learning is not carried out on a single classifier, but on a set of classifiers. In general, this technique performs better than separate classification models and, for this to happen, the classifiers of which this meta-classifier is composed have to be accurate (having a lower error rate than performing a random prediction) and varied (making different errors when classifying new instances).

Moreover, Arbitrated Ensembling is, in essence, a weighted average of the available models. However, these weights, rather than being random or static, are dynamically estimated using the loss of these available models.
Thus, the better an individual available model performs (the lower its error), the higher the weighting assigned to it in the overall forecast. To obtain these errors and thus the weights, it is important to keep in mind that the meta-learning approach to adaptively combine forecasting models that specializes them through regression, time series and classification. We assume that different forecasting models have different areas of specialization and varying relative performance. Especially in the case of time series problems, it is very rare to find individual models that perform consistently well over the time series. There are times when a given model performs well, while there are times when other models do better.

The goal of Arbitrated Ensembling is to take advantage of this localized experience of the individual models in these individual fixed-length moving window and generate a combined forecast for the next time pocket.

2.3 Architecture of Arbitrated Ensemble

The base learners M produce the predictions $\hat{y}_i$ i ε{1, . . . , m} for the next value of the dataset. In parallel, the meta learners Z produce the weights w of each base learner according to their predictions of error, $\hat{e}_i$. The final prediction $\hat{y}$ is computated using a weighted average of the predictions relative to the weights.

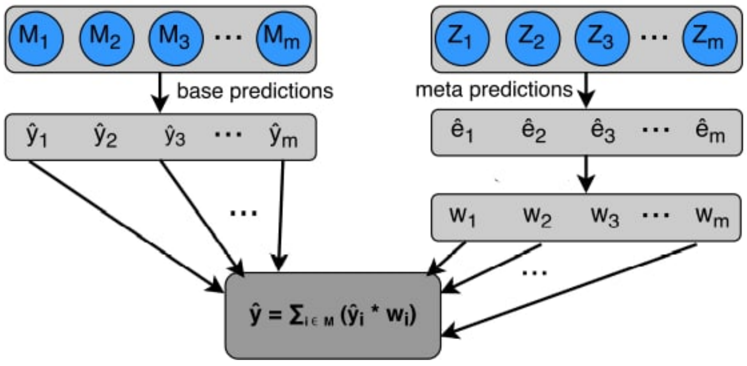

Figure 1: Graph of the Arbitrated Ensemble workflow for a new prediction.

- **Input data level.** Given the training set T, an assembly can train each of its m base models on the same set T, and this can be a subset of T.

- **Variable level.** With respect to the different p variables of each x_i ϵ T, each base classifier can be trained using the whole set of variables or with only a subset of it. In some techniques, such as random subspace methods, each base classifier is trained on a random subset of variables to yield less correlated models, thus increasing diversity.

- **Model level.** Given the wide range of statistical learning models that can be used as base classifiers, the possibilities when combining them are endless. However, two main strategies can be defined.
  In one, an ensemble is created using the same model m times, varying only the parameters between them. For example, using m k-NN models by modifying the parameter k or m decision trees with different stopping criteria and different pruning levels. In the other, m different methods are assembled; a k-NN, a linear model, etc.

- **Combination level.** Once the result of the m base models is available, the possibilities for combining them are divided into two main types: trainable and untrainable assemblies. However, in this project we only focus on the trainable models. Within the untrainable ones are the techniques that directly use the output of the m models to obtain the final classification, for example, unweighted committee voting. Trainables, however, are ensemble techniques that require the construction of a meta-model, using the outputs of the base models to provide the training set T with extra information.
  In addition, one can also distinguish between combination methods, which simply need the class predicted by the m base classifiers, and those that need the support that each base model m assigns to each class.

3 Arbitrated Ensemble for regression

3.1 Data

For regression problem we have selected 2 types of datasets: 4 regression datasets (with attributes used for prediction the respnonce variable) and 4 time-series datasets (with data points ordered in time where previous elements are used to predict the future once). In the tables below one could find the the names, number of attributes, size of the dataset and the link where to obtain it. The *Bike Sharing* dataset is used in both examples as it as hourly time series dataset with an attributes associated to the bike rental records.

Regression Datasets			
Name	Attributes	# instances	Source
Cancer	14	3047	link
Bike Sharing	15	17379	link
Insurance (Charge)	6	1338	link
Real Estate	6	415	link

Table 1: The datasets selected for the regression problem along with the number of attributes, amount of instances and the data source

Time series Datasets		
Name	# instances	Source
Bike Sharing	17379	link
IBM stock price	1008	link
Ozon	518	link
Temperature	1461	link

Table 2: The datasets selected for the time series problem along with the amount of instances and the data source

As one can see, we selected datasets with different attribute counts and of different sizes. We believe that attempting to run our analysis on such data allows to explore the performance of the Arbitrated Ensemble in different situation and, ideally, find out in which cases it has a higher efficiency by capturing the best models for specific observations.

3.2 Estimators used

In this section we explain the 8 different regression learning algorithms stored in a dictionary, which had parameters whose values could be modified to obtain the best possible result. We selected 8 models as we believe this amount is sufficient to analyse the performance of the ADE model, saving the computing time.

- **Support Vector Regression (SVR)** is a variant of the Support Vector Machine analysis model used for classification. However, with this variant the support vector model is used as a regression scheme to predict values. It is a set of supervised learning algorithms directly related to classification and regression problems where from a set of training data or samples and with labeled classes an SVM is trained to build the model that predicts the class of a new sample. A tolerance margin (epsilon) is established near the vector in order to minimize the error taking into account that part of the error is tolerated.

- **Decision tree (DTR)** learning is a method of approximating an objective function which is represented by a decision tree. It is a classifier expressed as a recursive partition of the instance space consisting of nodes, branches and leaves. Thus, when classifying an instance, it starts from the root node and, depending on the values of the predictor variables of that instance, the example goes through the relevant branches associated with those values until it reaches a leaf, which has a class assigned to it that will be used to classify the instance. Therefore, the branches represent the conjunctions of features that lead to those class labels. max_depth explains the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than the minimum number of samples.

- A **Random Forest (RFR)** model consists of a set of individual decision trees, each trained with a slightly different sample of the training data (generated by bootstrapping). The prediction of a new observation is obtained by aggregating the predictions of all the individual trees that make up the model. This implies that each tree is trained on slightly different data. In each individual tree, the observations are distributed by bifurcations (nodes) generating the tree structure until a terminal node is reached. The prediction of a new observation is obtained by aggregating the predictions of all the individual trees that make up the model. max_depth explains the maximum depth of the tree. If none, then nodes are expanded until all leaves are pure or until all leaves contain less than the minimum number of samples.

- **Linear regression (LR)** is a statistical method that attempts to model the relationship between a continuous variable and one or more independent variables by fitting a linear equation. It is called simple linear regression when there is only one independent variable and multiple linear regression when there is more than one. Depending on the context, the modeled variable is known as the dependent variable or response variable, and the independent variables as regressors, predictors or features. The intercept explains whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations.

- **Lasso (LAS)** is a regularization applied to linear regression that penalizes the sum of the absolute value of the regression coefficients and has the effect of forcing the coefficients of the predictors to tend to zero. Since a predictor with zero regression coefficient does not influence the model, lasso manages to exclude the least relevant predictors. fit_intercept explains whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations.

- **Bayes Ridge (BAY)** is a regularization applied to linear regression that penalizes the sum of the squared coefficients and has the effect of reducing proportionally the value of all the coefficients of the model but without them reaching zero. The main advantage of applying ridge over ordinary least squares (OLS) is variance reduction. Generally, in situations where the relationship between the response variable and the predictors is approximately linear, least squares estimates have little bias but can still suffer from high variance (small changes in the training data have a large impact on the resulting model).

- **ElasticNet (ELAS)** includes a regularization that combines lasso and ridge penalties. A frequently used strategy is to assign almost all the weight to the lasso penalty to select predictors and a little to the ridge to give some stability in case some predictors are correlated.

- **Automatic Relevance Determination (ARD)** fits the weights of a regression model, using an ARD prior. The weights of the regression model are assumed to be in Gaussian distributions. Also estimate the parameters lambda (precisions of the distributions of the weights) and alpha (precision of the distribution of the noise). The estimation is done by an iterative procedures (Evidence Maximization).

3.3 ADE configuratation and errors calculation

In this subsection we explain how we gather and process the errors of the base-learners and utilize them to train the Meta-learner as central part of ADE - arbitrated dynamic ensemble.

Dataset sampling (chunking) As introduced by Cerqueira et al. [1], sampling is used to learn the the behavior of estimators and balance error-learning of the meta-learner. This way we split the dataset in the K predetermined equal chunks and starting from the first chunk we train the base learners and predict with them the next chunk. The prediction errors than are stored. The instances of the chunk are used to train the meta-learner to predict the obtained prediction errors. Than we iteratively enlarge the first chunk with the next one and use it for training the base models. Then the next portion of the errors are learned by the meta learner. This process continues till all the chunks *1-K* are used to predict the chunk K. All in all, this approach enlarges the training set for the meta-learner and enables it to learn from both undertrained and overtrained results of the base-learners.

Prioritization vs. Weighting Now as we know how our meta-learning model trains on the errors the base-learners produce on the dataset, we move to the stage how can one select the final output of the ensemble. In this study we use 2 approaches for ADE to produce an optimal prediction:

- Prioritization - for each observation, ADE selects the prediction result of the the model with the lowest predicted error by the meta-learnear.
- Weighting - for each observation the prediction of error for each base-learner is produced by the meta-learner. The lower the error the higher influence the base-learner has on the final result.

In case of prioritization we basically discharge the output of other base-learners and take only one that is expected to be the best one. In contrary, weighting gives a chance to each of the base-learner to contribute for the final result. Our scientific interest here is to find an evidence which of the approaches produces better results.

On the other hand, for errors weighting we decided to reuse the approach of *exponential weighting* [1] and implement our solution which we call the *inverse weighting.* To begin with, the errors are represented as squared distances of the prediction made by the base-learners and the true response value from the training set. The *exponential weighting* approach can be represented as:

$$w^j{}_t = \frac{exp(-\hat{e_t}^j)}{\sum_{j\epsilon M} exp(-\hat{e_t}^j)}, \tag{1}$$

where $\hat{e_t}^j$ is the meta-learner prediction of error that base-learner made, w_t^j is a weight assigned for a base-learner prediction result and M is a committee of base-learners.

In contrast to *exponential weighting* of errors, we designed the approach of *inverse weighting* which begins with :

$$p_{jt} = \frac{\hat{e_t}}{\sum_{j\epsilon M} \hat{e_t}^j}, \tag{2}$$

where $\hat{e}^j$ is the meta-learner prediction of error that base-learner made, p^{jt} is the proportion error the base-learner holds in respect to other base-learners. As we are seeking to penalize those base-learners that have relatively big errors and benefit from those who have relatively low errors. For this reason we use inversing. And to balance out the values so that they still keep summing up to 1 we introduce the following calculations:

$$p_{1j}{}^{-1}x_t + p_{2j}{}^{-1}x_t + p_{3j}{}^{-1}x_t + ... + p_{nj}{}^{-1}x_t = 1, \tag{3}$$

where x_t is a balancing parameter for base-learner errors proporitons for the instance t, ${p_{jt}}^{-1}$ the inverse of the proportion error of a base-learner for the instance t.
After the above mentioned steps we receive the balanced proportions

$$For\ each\ instance\ t\ : w_{jt} = p_{jt}^{-1} x_t, \tag{4}$$

where for t is an instance of the dataset.
Finally, we produce the weighted prediction by ADE of the base-learners:

$$\hat{y}_t^{ADE} = \sum_{j\ \epsilon_M} \hat{y}_t^j w_t^j, \tag{5}$$

where $\hat{y}_t^{ADE}$ is a balanced prediction of the base-learners weighted by the proportions derived from the predicted errors, $\hat{y}_t^j$ - prediction made by base-learner j for instance t.

As a result, we demonstrated 2 possible approaches of error evaluation for producing the final output. In our experiment we will compare the performance of those methods and will try to find out if one of the error analysis methods presented here performs better than another one.

3.4 Results

In this chapter, we provide the results that we obtained with our regression and time series ADE predictions. For comparison we show table of the base learner prediction errors on the competition dataset and a table of prediction errors made by different ADE models separately for regular regression problems and time series regression. The experiment is conducted on competition dataset which is 30% of the whole data available from the original dataset. Accordingly, 70% is used for the training purpose.

Regular Regression First off all, we provide the description for 8 Arbitrated Dynamic Ensemble models we designed for comparing with base-learners. We selected the random forest regressor (RFR) as the meta-learning model due to its complexity (combines numbers of decision trees) and good performance results on the datasets we present in our report. $Ens3$ is ADE with chunk size = 3 ($beta$ = 3) that uses inverse weighting as base-learners result selection, $ExEns3$ utilizes the same approach but uses exponential weighting for results determination. With $Ens3P$ and $ExpEns3P$ we follow the same corresponding approaches, but the result is derived by selecting the output of the base-learner with the lowest predicted error by meta-learner. The other 4 ($Ens5, Ens5P, ExEns5, ExEns5p$) follow the same design but the chuck size used here equals to 5 ($beta = 5$).

Based on the results obtained we can see the top performing model is Random Forest Regressor, except from 1 case where Linear Regressor (LR) and Lasso Regressor (LAS) showed a better result leaving Random Forest Regressor (RFR) the third- most worst performers for the $Insurance$ dataset. Looking trough the results obtained for different variation of ADE one could claim that they are not the best selection for this problem. Nevertheless, if we scrutinize the outcomes the Ensembles produced one could see that

Regression: Base-learners								
Dataset	SVR	RFR	DTR	LR	LAS	ELAS	BAY	ARD
Cancer	817.30	**477.36**	586.67	520.72	519.38	719.79	528.71	509.76
Bike Sharing	23306	**13118**	13881	22150	22159	26643	22152	22154
Insurance	1.64	1.33	1.35	**1.2713**	**1.2713**	1.30	1.2765	1.28
Real Estate	60.10	**58.51**	60.02	71.88	71.74	74.40	71.26	105.82

Table 3: The mean prediction error of the base-learners on the regression competition datasets. Base -learners: SVR - Support vector regressor, RFR- Rando forest regressor, DTR - Decision Tree Regressor, LR - Linear regressor, LAS - Lasso regressor, ELAS - ElastincNET, BAY - Bayes Ridge Regressor, ARD - Automatic relevance determination.

$ExEns3P$ is very close to the RFR result for $Cancer$ dataset and is the second-best performer on it. The results of $Ens5$ for $Insurance$ and $RealEstate$ are really close to the best-performer of the base-learner tables. To sum-up, one could infer here that meta-model being arbitrager in weighting is able to predict the models with better accuracy and benefit from it generating a worthwhile results. On the other hand, the usage of prioritization is not justified.

Regression: ADE								
Dataset	Ens3	Ens3P	ExEns3	ExEns3P	Ens5	Ens5P	ExEns5	ExEns5P
Cancer	582.04	519.29	604.34	**503.87**	626.60	708.31	584.01	708.31
Bike Sharing	22344	23719	26655	23719	**21292**	23508	26654	23508
Insurance	1.32	1.63	3.24	1.63	**1.29**	1.6484	3.24	1.6484
Real Estate	63.65	70.45	64.36	70.45	**60.71**	72.25	63.37	72.25

Table 4: The mean prediction error of the Arbitrated Dynamic Ensemble on the regression competition datasets. Meta-learners: Ens3 - ADE(inverse weighting, 3 chunks), Ens3P - ADE(inverse weighting, 3 chunks with prioritization), ExEns3 - ADE(exponential weighting, 3 chunks), ExEns3P - ADE(exponential weighting, 3 chunks with prioritization), Ens5 - ADE(inverse weighting, 5 chunks), Ens5P - ADE(inverse weighting, 5 chunks with prioritization), ExEns5 - ADE(exponential weighting, 5 chunks), ExEns5P - ADE(exponential weighting, 5 chunks with prioritization).

Time-series regression In this part, we use the same approaches as above, but instead of attributes of the class we use the response values up to the lag of 10. So we use 10 consecutive observations to predict the 11-th one. Without surprise, none of the ADE performs the best on the dataset. Here the Random Forest Regressor and Linear Regression has a draw in being the best ones. At the same time, we can observe again that ADE is able to learn and predict which models can perform better to accumulate a meaningless result with middle up to top performance results.

All in all in this chapter we explored the performance of ADE with different settings and could claim that with a given set of base-learners and design approach the Arbitrated Dynamic ensemble performance is in between 50% up to 95% percentile when competing with the base-learners. Needless to say, that increasing the amount of chunks allowed

Regression for Time series: Base-learners								
Dataset	SVR	RFR	DTR	LR	LAS	ELAS	BAY	ARD
Bike Sharing	26904	**8302**	9589	11711	11711	11792	11712	11710
IBM	44.38	27.46	58.88	**16.30**	16.31	16.56	16.35	16.35
Ozone	543.5	**358.5**	569.8	373.4	373.5	386.3	375.0	389.1
Temperature	22.74	17.86	21.06	**16.81**	16.85	16.97	16.85	16.87

Table 5: The mean prediction error of the base-learners on the time-series competition datasets. Base-learners: SVR - Support vector regressor, RFR- Rando forest regressor, DTR - Decision Tree Regressor, LR - Linear regressor, LAS - Lasso regressor, ELAS - ElastincNET, BAY - Bayes Ridge Regressor, ARD - Automatic relevance determination.

the meta-learning model to fit better the errors the base-learner model produced, hence receive better result in competition dataset.

In addition it is important to point out that priority selection approach for regression problems appeared to be not effective as a weighting technique. Similarly, it is noticeable that the approach of inverse weighting appeared to be better selection in the prediction result generation for a meta-learner than exponential weighting.

For a more appealing visual perception of the results visit Appendix A.1 where we visualize the results with a help of Nemenyi test.

Regression for Time series: ADE								
Dataset	Ens3	Ens3P	ExEns3	ExEns3P	Ens5	Ens5P	ExEns5	ExEns5P
Bike Sharing	11250	19070	11734	19731	**10869**	22414	19145	19378
IBM	17.11	119.50	16.94	119.50	**16.35**	44.29	16.33	44.29
Ozone	413.0	703.4	**396.3**	703.4	562.8	610.2	608.1	610.2
Temperature	20.92	26.03	24.45	26.03	**20.37**	23.07	22.56	23.07

Table 6: The mean prediction error of the Arbitrated Dynamic Ensemble on the time-series competition datasets. Meta-learners: Ens3 - ADE(inverse weighting, 3 chunks), Ens3P - ADE(inverse weighting, 3 chunks with prioritization), ExEns3 - ADE(exponential weighting, 3 chunks), ExEns3P - ADE(exponential weighting, 3 chunks with prioritization), Ens5 - ADE(inverse weighting, 5 chunks), Ens5P - ADE(inverse weighting, 5 chunks with prioritization), ExEns5 - ADE(exponential weighting, 5 chunks), ExEns5P - ADE(exponential weighting, 5 chunks with prioritization).

4 Arbitrated Ensemble for classification

In this part we decided to expand further the research of Cerqueira et al. [2] and explore the possibilities of using Arbitrated Dynamic Ensemble. This part demonstrate a novel approach of how we utilize the same philosophy for classification problems.

4.1 Data

For the classification problems we work with 4 publicly available datasets. In the tables below one could find the the names, number of attributes, size of the dataset and the link where to obtain it.

As one can see, for the research we prepared 2 datasets for binary classification and 2 for multinomial classification with different attribute count and different size. We keep here the same idea as we present in the regression part: explore the performance of Ensemble under different conditions to find the cases where ADE is the most efficient.

Classification Datasets				
Name	Attributes	Classes	# Instances	Source
Car	6	3	1728	link
Obesity	16	4	2211	link
Chess	36	2	3196	link
Tic-tac-toe	9	2	958	link

Table 7: The datasets selected for the classification problem along with the number of attributes, amount of instances and the data source

4.2 Estimators used

For the development of this section we use 8 different learning algorithms for classification stored in a dictionary, which had parameters whose values could be modified to obtain the best possible result. The amount of models is selected subjectively based on the amount of models used for base-learners in regression part.

- **Logistic regression (MLR)** is a statistical model used in machine learning to describe the relationships that exist between a set of predictor variables and a class variable in order to estimate the probability that an instance belongs to a certain class. In its original formulation it is a binary classifier (dichotomous prediction), but it can be generalized to multiclass classification. The objective for training the multinomial logistic regression model is that it estimates a high probability for the class to which each of the instances belong and a low probability for the others. For the multinomial class the loss minimised is the multinomial loss fit across the entire probability distribution, even when the data is binary. In order to optimize the function we use ab solver, which consists in subroutines for large-scale bound constrained optimization. We set the maximum number of iterations taken for the solvers to converge.

- **Support vector machines (SVM)** are a supervised learning algorithm that can be used for both classification and regression problems; however, it is mainly used in classification problems. In this algorithm, each instance (of the training sample) is represented as a point in n-dimensional space, with the value of each feature mapping to the value of a given coordinate. Next, given this set of training examples, each belonging to a class, we train an SVM to build a model that predicts the class of a new sample by constructing a hyperplane that separates the classes of the training data and maximizes the margin between those classes (maximizes the distance between the closest points of each class to the optimal separation hyperplane, called support vectors) in n-dimensional space.

- The **Stochastic Gradient Descent Classifier (SGD)** model implements a simple stochastic gradient descent learning routine that supports different loss functions and penalties for classification.

- One type of model that is widely used for supervised classification is **Bayesian networks**. Bayesian networks are probabilistic graphical models that allow the joint probability distribution of a set of random variables to be represented in a simple, compact, accurate and understandable way. In Bayesian classifiers, the objective is to assign the most probable class to a given instance, defined by a set of values of the predictor variables. In probabilistic terms, a test instance is assigned the class label with the highest a posteriori probability. On the other hand, the naive Bayes classifier is one of the simplest Bayesian classifiers. Its structure is based on setting the class variable without a parent, the predictor variables as child vertices of the class variable and without any dependencies between the predictor variables; it assumes that, given the class variable, the predictor variables are conditionally independent.
  The **Multinomial Naive Bayes classifier (multNB** is suitable for classification with discrete features. The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work. Like MultinomialNB , this classifier is suitable for discrete data. The difference is that while MultinomialNB works with occurrence counts, the **Bernoulli Naive Bayes classifier (bernNB)** assumes that all the features are binary such that they take only two values.

- **k-Nearest Neighbours (KNN)** is based on the idea of identifying the group of k objects in the training dataset that is closest to a new object to be classified, so that this new case is assigned the most frequent class label in that group of k objects. This supervised classification method is based on three main components: a set of labeled training data, a distance metric to compute the distances between different objects, and the number k of nearest neighbors. Thus, using the corresponding distance metric, the distance between the new case to be classified and the labeled cases is calculated to find out which k instances are closest to the new object. Once this step is executed, the most repeated class in the group of k closest cases is calculated and assigned to the new object.

- **AdaBoost (ADA)**, short for adaptive boosting, is an algorithm that can be used in conjunction with other learning algorithms to improve their performance. AdaBoost works by choosing a base algorithm and iteratively improving it by taking into account the incorrectly classified cases in the training set.

4.3 Errors calculations

Contrarily to the regression errors, the classification does not normally use distance measures as the classes are being compared rather than distances. Nevertheless, in our approach we decided to combine the distance measures with class predictions. That is why here we use the random forest regressor (RFR) as the meta-learning model as we do in regression part. Here we will explain the process:

- First of all, we create a dictionary of every class of response values of the dataset. Normally, each statistical model predicts each of the possible classes with some probability and selects the one that holds the highest probability. In our cases we are interested in probability for the class which is correct. To find out which probability the base-learner assigned to the class we apply the function *predict_proba* from *sklearn*. This way we obtain the prediction score (probability of selecting a class) of each base-learner for the correct class. The prediction score for each base-learner is than substracted from 1. This way we receive the error score (propability of not selecting the right class). This error is used to apply for meta-learning model training as a response variable.
- For each instance of the dataset the meta-learner produces its prediction of wrong class decision for each base-learner. Having received the prediction of errors, we end up with an option to choose between 2 strategies, prioritization or regression:
  - Prioritization - for each observation, ADE selects the prediction result of the model with the lowest predicted error by the meta-learner.

    $$\hat{y}_t^{ADE} = c_j,\ where\ \hat{e}_{jt} = \min(\hat{E}_{j\epsilon M}) \tag{6}$$

    where $\hat{e}_{jt}$ is the smallest predicted error of a base-learner j by meta-learner for instance$t$, c_t - class associated with the smallest predicted error, $\hat{E}_{j\epsilon M}$ is a set of all prediction errors of base-learners produced by meta-learner, y_t^{ADE} - class selected by ADE for instance t
  - Averaging - each observation the meta-learner produce the prediction of error for each base-learner. For each instance we create a dictionary where we insert all possible classes and equalize each to 0 score. When each of the base-learners predict the class we add the predicted error to the class score. Than each class score is divided by the amount of times the error was added there. The class which holds the lowest average prediction error by base-learners is assigned to be the result of ADE class prediction. It could be described in the equation as following:

    $$\hat{y}_t^{ADE} = c,\ where\ \hat{e}_{tc} = \min\left(\frac{\sum_{j\epsilon MC} \hat{e}_{jc}}{N_{MC}}, for\ each\ c\ where\ c\ \epsilon\ C\right), \tag{7}$$

    where c is a class out of all possible classes C and e_{tc} is the smallest error score of the class after the division of the sum of errors of base-learners that predicted this class $\sum_{j\epsilon MC} \hat{e}_{jc}$ by the number of base-learners that predicted this class N_{MC}, $\hat{y}_t^{ADE}$ prediction result calculated by ADE for instance t.

4.4 Results

This part shows the result that is obtained from the second part of the report which is dedicated to classification results. Here we use 4 classification datasets and produce 2 tables: 1 for base-learners and 1 for ADE following the same approach to explore the results obtain as in the regression part. The experiments were conducted on competition dataset which is 30% of the original dataset, using 70% for training the base-learners and obtaining error results for meta-learners.

To begin with, we provide the description of 4 Arbitrated Dynamic Ensemble models that we implemented to compete against the base-models. *Ens*3 is ADE with chunk size of 3 ($beta = 3$) uses meta-learner predicted errors for the base-models for each observation. The classes predicted by base-learners and average predicted error for each class is taken into account. The class with the lowest possible errors is selected as a final result.*Ens*3*P* we follow the same approach but select the result of the base-learner with the lowest predicted error by meta-learner. In addition to the explained ADE models, we have similar *End*5 and *Ens*5*P* respectively that are different only amount in amount of data chunks used for meta-learning ($beta = 5$).

Classification: Base-learners								
Dataset	MLR	SVM	SGD	RFC	multNB	bernNB	KNN	ADA
Car	0.917	**0.973**	0.905	0.967	0.855	0.863	0.946	0.795
Obesity	0.831	0.534	0.556	**0.958**	0.611	0.556	0.910	0.317
Chess	**0.959**	0.946	0.956	0.970	0.849	0.850	0.924	0.957
Tic-tac-toe	**0.982**	0.975	0.979	0.972	0.677	0.666	0.968	0.857

Table 8: The accuracy of the base-learners on the classification competition datasets. Base-learners: MLR - Multinomial logistic regression, SVM - Support vector machine, SDG - Stochastic gradient descent classifier, RFC - Random forest classifier, multNB-Multinomial naive bayes, bernNB - bernuolli naive bayes, KNN - K-nearest neighbors, ADA - AdaBoost

In contrast to Regression results part here we evaluate the accuracy of the class prediction. This means - the higher value achieved the better. Based on the results obtained in the base-learners classification, Multinomial Logistic Regressor (MLR) takes a lead having a the best performance for *Chess* and *Tic-tac-toe* dataset, Support Vector Machines (SVM) is the best for *Car* class predictions and Random Forest Classifier (RFC) is the most accurate for *Obesity* type predictions.

Now, let us compare the results of base-learners with the ADE results. As we can see in Ensemble results there is a clear domination of prioritization approach with *Ens*5*p* having the best prediction for *Car*, *Obesity*, *Chess* and *Ens3p* for *Tic-tac-toe* leaving the averaging ensembles approach behind in terms of accuracy. Furthermore, the accuracy result of *Ens5P* is higher than the highest accuracy of the base-learner for the predictions of *Chess* dataset.

Classification: ADE				
Dataset	Ens3	Ens3P	Ens5	Ens5P
Car	0.761	0.797	0.855	**0.963**
Obesity	0.446	0.902	0.798	**0.906**
Chess	0.871	0.965	0.912	**0.967**
Tic-tac-toe	**0.982**	**0.982**	0.975	0.947

Table 9: The accuracy of the Arbitrated Dynamic Ensemble on the classification competition datasets. Meta-learners - Ens3 - ADE(averaging, 3 chunks), Ens3P - ADE(3 chunks with prioritization), Ens5 - ADE(averaging, 5 chunks), Ens5P - ADE(5 chunks with prioritization).

In conclusion, one could identify the same evidence for the meta-learner model as it is identified in the regression part. The ability of Arbitrated ensemble to manage the selection of the results could provide a significant generalization of the base-learners for predictions. Although the performance of the ensemble with error averaging cannot demonstrate profound results for classification, the prioritization can, which is directly opposite to the results of ADE obtained in the regression part.

For a more appealing visual perception of the results visit Appendix A.2 where we visualize the results with a help of Nemenyi test

5 Summary

With this study we explored the possibility of using the Arbitrated Dynamic Ensemble as a tool to obtain optimal decision for both regression and classification problems. We are especially proud of our achievement in *inverse weighting* calculations that proved to more efficient in errors calculations for base-learners than the original exponential weighting. In addition, we provided an evidence that prioritization for selecting the best predictor for the observation is not efficient for regression problems, but is competitive for classification.

Nevertheless, taking into account the obtained results, we cannot claim that utilizing given predictive models as base-learners the ADE can result in predictions that could outperform the base-learners. On the other hand, certainly, our development of ADE has a potential for improved prediction. This could be achieved by increasing the amount of base learners used and discharging a certain amount of the least-performing once.

The implementation related to our report is freely available on *GitHub*.

A Appendix

A.1 Regular regression results with Nemenyi test

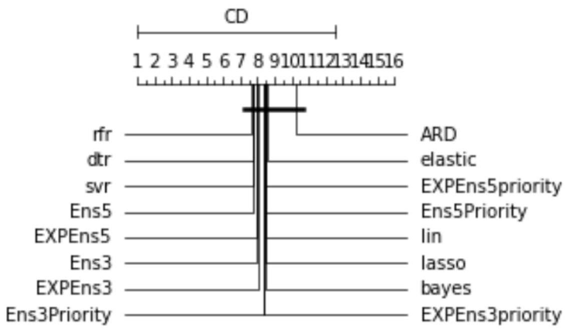

Figure 2: Mean errors for Regression with Real estate dataset

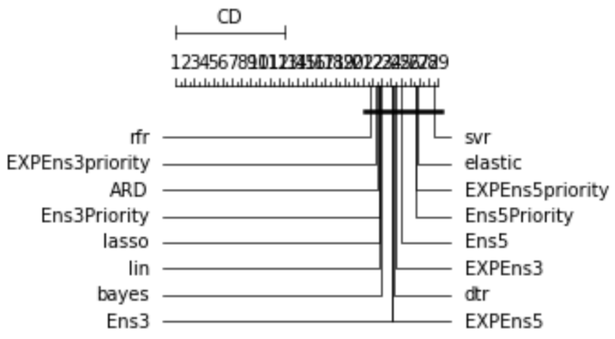

Figure 3: Mean errors for Regression with Cancer dataset

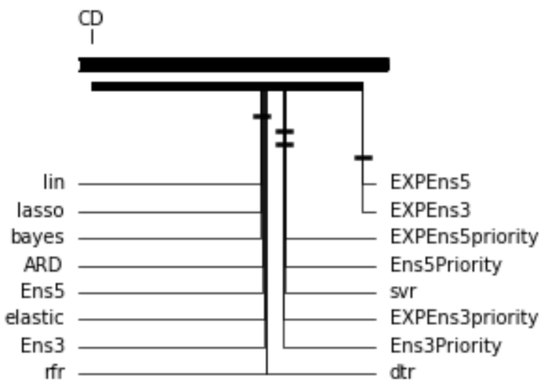

Figure 4: Mean errors for Regression with Insurance charge dataset

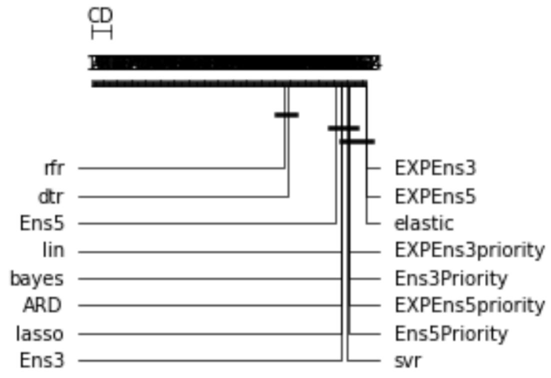

Figure 5: Mean errors for Regression with Bike sharing dataset

A.2 Regression results for time series with Nemenyi test

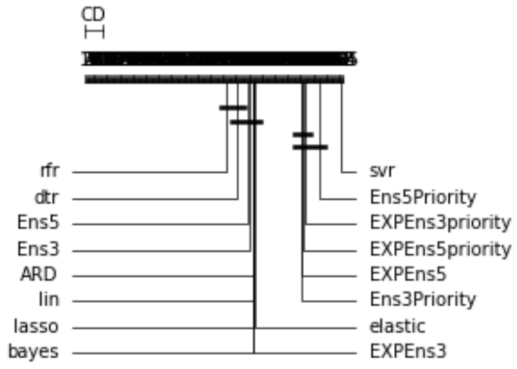

Figure 6: Mean errors for Time-series Regression with Bike sharing dataset

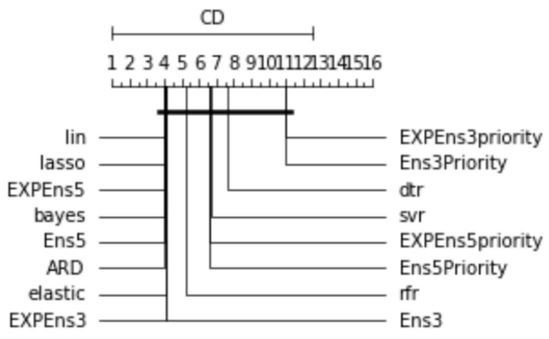

Figure 7: Mean errors for Time-series Regression with IBM dataset

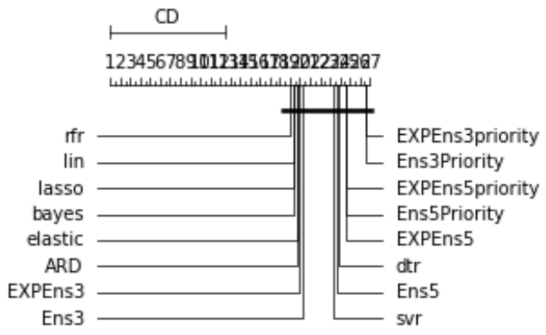

Figure 8: Mean errors for Time-series Regression with Ozone dataset

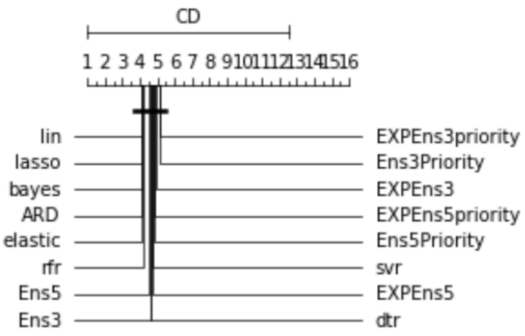

Figure 9: Mean errors for Time-series Regression with Temperature dataset

References

[1] Vítor Cerqueira, Luís Torgo, Fábio Pinto, and Carlos Soares. Arbitrated ensemble for time series forecasting. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 478–494. Springer, 2017.

[2] Vitor Cerqueira, Luís Torgo, Fábio Pinto, and Carlos Soares. Arbitrage of forecasting experts. *Machine Learning*, 108(6):913–944, 2019.

[3] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.