24th COBEM - 2017

24th ABCM International Congress of Mechanical Engineering
December 3-8, 2017, Curitiba, PR, Brazil

COBEM-2017-1600
INVESTIGATING THE USE OF EXTREMELY RANDOMIZED TREES, GRADIENT BOOSTING MACHINE, K-NEAREST NEIGHBORS AND THEIR ENSEMBLE APPLIED TO FAULT DETECTION

Luís Felipe Nogoseke
Gabriel Herman Bernardim Andrade
Marco Antonio Reichert Boaretto
Electrical Engineering Graduate Program, Federal University of Paraná, Curitiba, Paraná, Brazil
luis_nogoseke@hotmail.com, gabrielhbandrade@gmail.com, marco.boaretto@hotmail.com

Leandro dos Santos Coelho
Industrial and Systems Engineering Graduate Program, Pontifical Catholic University of Paraná, and
Electrical Engineering Graduate Program, Federal University of Paraná, Curitiba, Paraná, Brazil
leandro.coelho@pucpr.br

Abstract. *The idea of ensemble methodology is to build a predictive model by integrating multiple models. It is well-known that ensemble methods (called ensemble classifiers or multiple classifier systems) can be used for improving prediction performance in classification tasks. In recent years, machine learning and pattern recognition approaches have been a promising tool in the field of fault diagnosis. The fault detection is an approach to catch any abnormal events of the system quickly or in advance and notify the problems to the system user. The main contribution of this paper is to verify the effectiveness of using ensemble learning combining extremely randomized trees, gradient boosting machine, k-nearest neighbors and artificial neural networks for fault detection case studies related to signal processing and mechanical systems. The ensemble approach with a stacking method reached a better performance when compared with the other machine learning techniques applied alone in both of the two fault study cases problems. Proving that the combination of different techniques with low accuracy scoring, can result in a final model that overachieves the overall performance of the techniques applied alone.*

Keywords: *Machine learning, Extremely randomized trees, Gradient boosting machine, k-Nearest neighbors, Fault detection.*

1. INTRODUCTION

In the modern industrial production process, different ways and techniques for reducing the risks and accidents which may result in public damage and large economic losses have been always concerned. The monitoring of an industrial process is quite necessary to ensure the safety of producing. With the ever-increasing complexity of industrial processes, more and more process data are collected, and the machine learning based methods have been widely used to realize process monitoring in recent years (Wang et al., 2015).

To automatically detect faulty components, machine learning algorithms can be adopted. Machine learning algorithms use data to construct a model that can detect different conditions using regressors and/or classifiers. Data used to train models are features which are constructed and extracted by an expert from raw data. The classification task is based on a classification models (classifiers) that are induced from an exemplary set of preclassified patterns. An algorithm which constructs the model is called inducer and an instance of an inducer for a specific training set is called a classifier.

Recently, ensemble learning techniques have received much attention. The main idea behind the ensemble methodology is to weigh several individual classifiers, and combine them in order to obtain a classifier that outperforms every one of them (Rokach, 2010). The combination of many classifiers in an ensemble is a well-known method of increasing the quality of recognition and classification tasks. Utilization of learner ensemble technique improves the generalization performance of learning system in comparison with that of single learner. Moreover, it enables an overall

system easier to modify, and then perform more difficult tasks than any of its components. Ensemble methods have been used in multiple research fields such as computational intelligence, statistics, engineering and machine learning.

A typical ensemble method for classification tasks contains the following building blocks: (i) *training set*: a labeled dataset used for ensemble training; (ii) *base inducer*: the inducer is an induction algorithm that obtains a training set and forms a classifier that represents the generalized relationship between the input attributes and the target attribute. (iii) *diversity generator*: this component is responsible for generating the diverse classifiers, and (iv) *combiner*: the combiner is responsible for combining the classifications of the various classifiers.

Given the potential usefulness of ensemble methods, it is not surprising that a vast number of methods are now available to researchers and practitioners (details are presented in (Ren et al., 2016)). The aim of this paper is to provide the effectiveness of using ensemble learning combining random forest, gradient boosting machine and support vector machine for fault detection problems related to mechanical systems.

The remainder of the paper is organized as follows. Section 2 provides the fundamentals of machine learning its definitions and different types of task. Section 3 refers to the information of the different ML approaches used in this paper. The information of the two case studies adopted in this paper is provided in Section 4 and Section 5 present the results of the experiments performed in this paper. Finally, in Section 6 the conclusion is presented, with the view of the authors on the overall results and future works.

2. FUNDAMENTALS OF MACHINE LEARNING (ML)

In the past few decades, there's been a huge increase in the quantity of generated and stored information, due to the popularization of technology and the increase in computational power along the years. All this material can be considered a real gold mine with many relevant information hidden inside of petabytes of stored data. This mined information has been shown useful as big companies started to realize the importance of finding patterns in this data, in order to be able to predict its customer's behavior, a social network can analyze people's likes and posts to display more suitable advertisement, for example. We are now at a point where data analysis cannot be done manually due to the amount of the data. This has driven us to the use of computational tools that are able to analyze big data and learn how to extract information automatically.

Learning is a complex phenomenon and, like intelligence, covers a broad range of processes that are difficult to define by themselves. It can be defined by simple phrases such as "to gain knowledge" or "to understand", but a few steps are needed to do so, such as acquire a new declarative knowledge, develop the cognitive skills that it requires through instruction or practice, modeling the new information in a general, broad way and discover new facts through observation and experimentation. The core of Machine Learning (ML) is to make computers be able to walk through these steps and produces more positive outcomes with increasingly precise predictions.

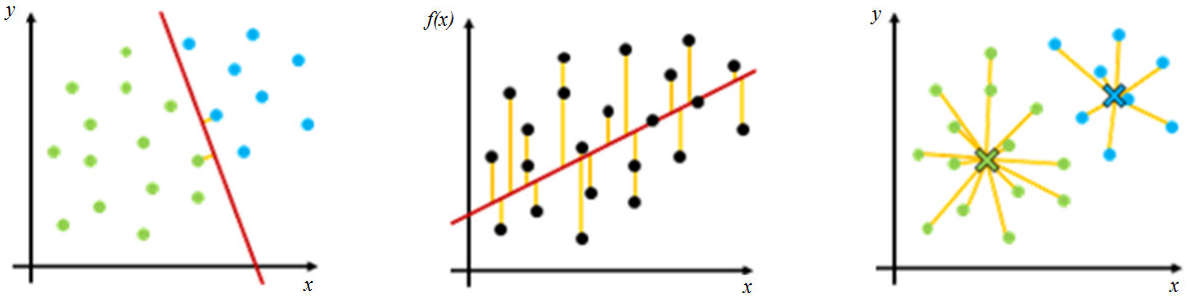
ML arose to, as Samuel said, give "computers the ability to learn without being explicitly programmed". We say that a machine learns whenever it changes its structure or program in such a manner that its expected future performance improves (Samuel, 1959).The learning is done always based on some sort of observation or data, such as examples, direct experience or instructions, using, then, past experiences in order to do better in the future. That is why ML is so suitable for working with huge amounts of data, the more information is fed into the system, more precisely, accurately and correctly it will output. As a core subfield of Artificial Intelligence, ML focuses in studying and constructing systems that can learn automatically by themselves from the data, minimizing human dependency. In other words, construct algorithms that are capable to identify patterns according to its ability to analyze and model past experiences.

From all that has been said, it is possible to draw Mitchell's formal definition of a ML algorithm: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" (Mitchell, 1997).

The tasks that are typically assigned to ML algorithms can be classified into three vast categories, regarding the nature of the learning process used. We may input into the system a set of data that is presented as examples of inputs and their desired outputs, given by an "oracle", which represents the ground true output for that sample of data. In this case the algorithm has the task to identify a general rule that fits to this set of data and maps the sets of input data to outputs. This is called Supervised Learning, where the ML system is explicitly being "taught" how to behave against the data. In contradiction, Unsupervised Learning is the case where no true output values are given (the patterns in the data may be unknown), expecting that the algorithm, on its own, identify a structure on the given set of input data, generating a model that allows it to classify new sets of data. As a third stand, there is Reinforced Learning, in which there is no correct input/output pairs of data given, but, instead, the algorithm receives a feedback in form of rewards or punishments for actions taken while it takes actions, exploring the search space of solutions.

ML tasks can also be classified regarding their output. In classification tasks, the data set is divided into two or more distinct classes, in which the learner system must generate a model that assigns new unseen input sets to one (or more) of this classes. Most of ML problems fall in this class, since they are more common in real life, such as face detection, spam filtering and OCR (optical character recognition). Regression is a kind of task where the algorithm has to generate a model of the data in order to be able to predict an output continuous value, rather than a class label. Both of this group of tasks is usually faced through Supervised Learning approaches. And last, we have Clustering tasks, which, similarly

to classification, there is a set of input data that is intrinsically divided into groups, however there is no information about such grouping beforehand, making this typically an unsupervised task. A simple visual example of the difference of these tasks is shown in Fig. 1. Classification aims to create borderlines between the tasks, Regression tries to adjust a line that better fits all the data and Clustering looks for patterns that identify different groups in the data and separates it in meaningful ways.



(a) classification (b) regression (c) clustering

Figure 1. Visual representation of three core ML tasks.

3. DESCRIPTION OF THE ADOPTED ML APPROACHES

ML applications range throughout so many different areas of study that, along the years, many different techniques have emerged in order to try to solve the problems it faces. Since the problems of unlike fields of study that are treated by ML are diverse in their format, data types and desired output, the developed algorithms applicable to these problems have different functionalities and work based on different methodologies. One may have from rather simple approaches, such as regression or instance based algorithms, in which the relationship between variables is modeled and via a similarity measure a best match is selected and a prediction is made, to complex ones, such as Evolutionary Algorithms or Artificial Neural Networks, inspired in phenomena derived from nature behavior and biological science observations.

When treating about ML, it is stated that there is no one algorithm works best for every problem. For example, you can't say that neural networks are always better than decision trees or vice-versa. There are many factors at play, such as the size and structure of the dataset. The answer to what algorithm should be used is dependent to various factors. It depends on the size, quality, and nature of the data. It depends on what the desired output answer should be and in what format one may want it. It depends also on the computational power and time available to progress the task. Even the most experienced data scientists can't tell which algorithm will perform best before trying them.

Of course, there are algorithms that are more suitable for a certain type of problem than other (mainly dependent on the type of task represented by the problem), which is where testing and evaluating the performance of distinct classes of algorithms comes into play. In this paper, we evaluate the performance of a stacking ensemble approach that uses Extremely Randomized Trees (Extra-Trees) (Geurts et al., 2006), Gradient Boosting Machine (GBM) (Friedman, 2001), k-Nearest Neighbors (KNN) (Cover and Hart, 1967) and Artificial Neural Networks (ANN) (Rosenblatt, 1957) applied to the machine fault detection problem.

3.1 Extra-Trees

Random forest (RF) was proposed by Breiman (2001). It is a combination of many decision trees that creates using bootstrapping technique developed by (Ho, 1995), coming from the learning dataset samples of the predictors and choosing randomly at each node. One of its advantages is that it produces an estimation of classification accuracy based on the so called out-of-bag cross-validation method (bagging or bootstrap aggregation). It is usually assumed that such estimation is not biased and may be used instead of validation based on an external data-set or a cross-validation external to the algorithm.

The decision tree is a non-linear and non-parametric supervised classification algorithm where the nonterminal nodes indicate the features and terminal nodes are outcomes. Random forests are an ensemble of decision tree classifiers (Breiman, 2001), with acknowledged advantages such as superior classification accuracy, ability of handling high dimensional data without attribute selection, and the feasibility to be conducted by parallel computing.

Its main idea is to train a series of decision trees (Classification And Regression Tree, CART) (Breiman et al., 1984) in which each node attribute is selected from a random set of attributes and the trees are built on bootstrap samples of the data. In general, the training of decision trees for both classification and regression problems starts from the root node, and binarily divide the nodes into branches until it reaches the leaf, where the nodes represent the test over a

feature, the branch the values of the result of the test and the leaf represents the class. A basic representation of a classification tree is shown in Fig. 2.

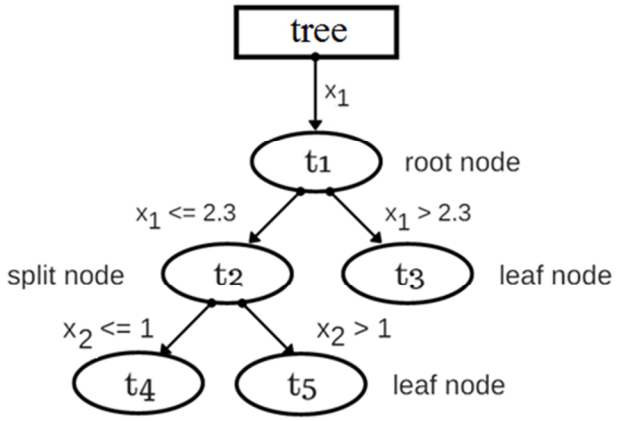


Figure 2. Example of a classification tree

Each tree fits to each sample generating a set of classifiers, which via majority vote each classifier vote for its predicted class label, as shown in Fig. 3, and the label with the most votes is used to classify the observation (Fawagreh et al., 2014). The combination of many bagged predictors by the RF technique result in more accurate classifications with the ability to model complex interactions among predictor variables (Cutler et al., 2007).

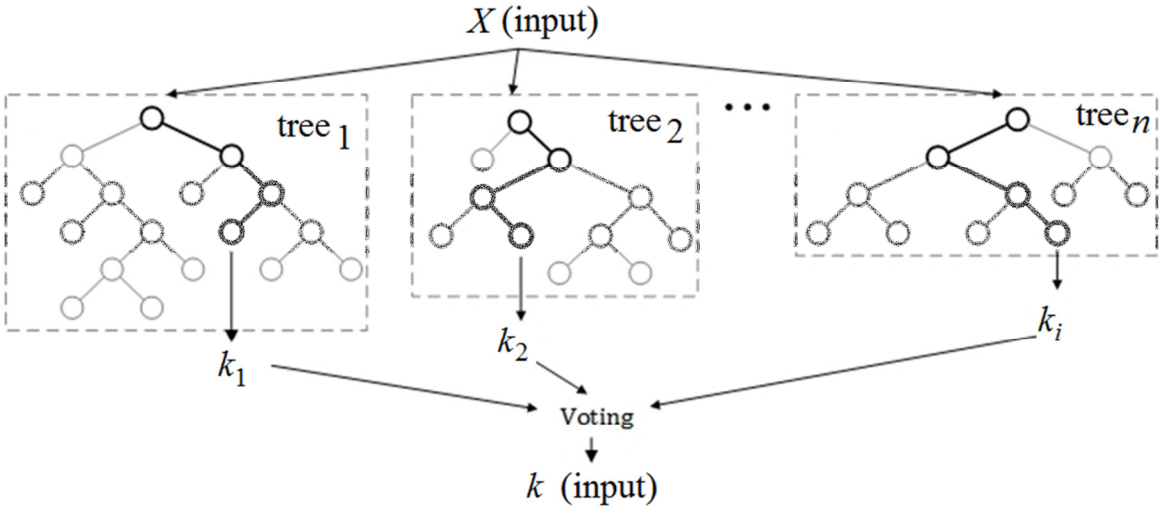


Figure 3. RF ensemble representation

The Extra-Tree algorithm proposed by Geurts et al. (2006), builds an ensemble of unpruned decision or regression trees according to the classical top-down procedure, like in an ordinary random forest. Its two main differences with other tree-based methods are that it splits nodes by choosing cut-points fully at random and that it uses the whole learning sample (rather than a bootstrap replica) to grow the trees. Instead of computing the locally *optimal* feature/split combination, the top-down splitting in the tree learner is randomized, for each feature under consideration, a random value is selected for the split. This value is selected from the feature's empirical range.

3.2 Gradient boosting machine

Gradient boosting machine (GBM) (Friedman, 2001) is an improved boosting algorithm for regression and classification problems. The basic theory of GBM is to produce a prediction model constructed by an ensemble of weak learners or base learners, typically decision trees, and each tree is grown sequentially by using the information from previously grown trees.

The major difference between bagging and boosting methods is that the boosting method strategically resamples the training data to provide the most useful information for each consecutive model. The adjusted distribution during each step of training is based on the error produced by the previous models.

Unlike the bagging method where each sample is uniformly selected to produce a training dataset, the probability of selecting an individual sample is not equal for the boosting algorithm. Samples that are misclassified or incorrectly estimated have more chances to be selected with higher weight. Therefore, each newly created model places emphasis on the samples that have been misclassified by previous models (Zhang and Haghani, 2015).

Gradient boosting machine (GBM) produces a competitive, highly robust, interpretable procedures for both classification and regression (Friedman, 2001). In other words, GBM is an ensemble of boosted regression trees, regression trees differs from decision trees on the fact that regression trees contains a continuous score on each of the leaf (Chen and Guestrin, 2016). GBM uses a gradient-descent method to build a tree which decrease the objective on the direction of the gradient, in consequence data-points which are hard to classify gain influence during the training (Keck, 2016).

GBM works with the idea of additive training, by combining several weak learners to develop a strong learner (Urraca et al., 2017).

3.3 K-Nearest Neighbors

The conventional kNN algorithm is a simple and well-known instance-based method in pattern recognition proposed by Cover and Hart (1967) (Cover and Hart, 1967) that simply uses the *k* training samples (observations) that are closest to (nearest neighbors) the test sample according to a distance metric to classify it. It is called a lazy learning algorithm because the generalization occurs only when a new observation beyond the training data needs to be classified. The kNN can be seen as a nonparametric classification technique based on an empirical Bayes decision rule that can achieve high classification accuracy in problems that have unknown and non-normal distribution.

Its learning process consists on saving all the training instances with its class' labels, to classify an unknown instance the classifier ranks the instance's neighbors among the training instances and use the class label of *k* most similar neighbors to predict the class of the new instance (Tan, 2006).

The distance between the new instance and the *k* nearest neighbors is measured and the closest neighbors indicates the class for the new instance, as shown in Fig. 4, where *k* is a parameter set in the beginning of the algorithm that represents the number of neighbors that will be compared. The value of *k* is preferable to be an odd number since an even number of neighbors can result in a tie decision.

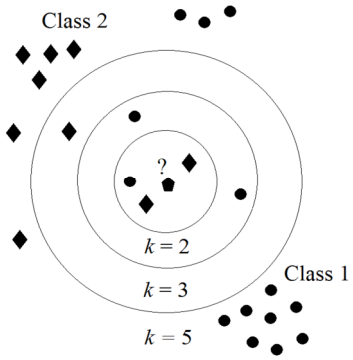


Figure 4. Example of a distance comparison related to *k* value in the kNN algorithm

The kNN algorithm and its variants have been widely used in the literature to solve real problems. For example, the kNN classifier has been applied to feature selection (Park and Kim, 2015) and dimensionality reduction (Ingram and Munzner, 2015).

3.4 Artificial Neural Network

Based on the studies that developed the first mathematical model of the biological neuron performed by (McCulloch and Pitts, 1943), the first ANN was developed by Rosenblatt in 1957 (Rosenblatt, 1957) with the goal to solve pattern recognition problems.

ANNs are "digitalized" representations of the human brain, with ability to learn from training to recognize patterns and perform other task as in classification and regression. The neuron body consists of its nucleus, dendrites, axons and its axon terminals, as can be seen in Fig. 5.

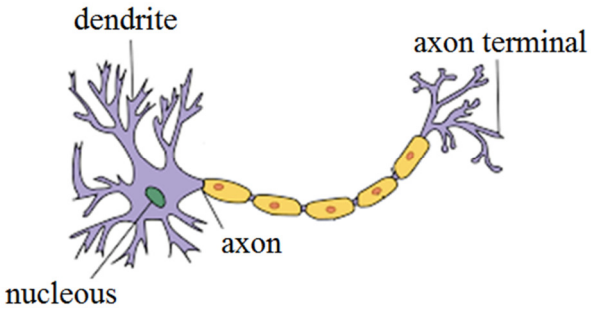


Figure 5. Biological neuron

There are several ANN architectures on the literature as the *feedforward neural network* (Tahmasebi and Amirkabir, 2011), *feedback neural network* (Zamarren and Gonza, 2005), *recurrent neural network* (Rojas, 1996) and *self-organizing maps.*(Kohonen, 1990).

The Multi-Layer Perceptron (MLP), consists of layers of connected neurons that has one or more hidden layers on its architecture, Fig. 6. The hidden layer consists of the transformation where the input data is projected into a linearly separable space, the most common transformation functions are *tanh* and *sigmoid* functions.

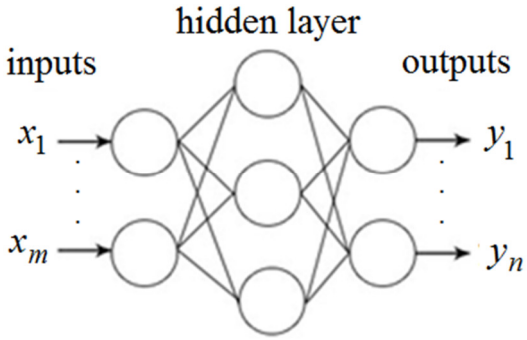


Figure 6. MLP structure

Although the ANNs have a high adaptability rate in order to identify new patterns that haven't being seen and are easy to implement, when not well tuned can present a high probability of overfitting and low generalization accuracy.

3.5 Ensemble approach

Ensemble methods use a combination of multiple models to obtain better performance. The ensemble approach was proposed by Hansen and Solomon who demonstrated that combining multiple neural networks, trained independently, raises significant the model capacity of generalization (Hansen and Solomon, 1990).

The basic structure of an Ensemble model is shown in Fig. 7, base classifiers are trained and a combination technique is chosen to make the prediction based on the output of the base classifiers.

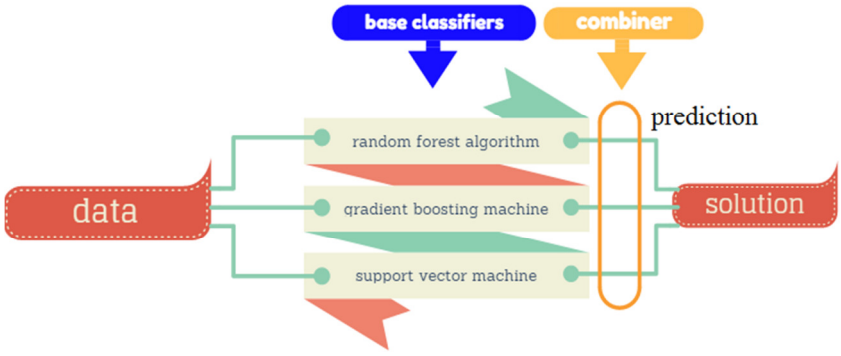


Figure 7. Basic ensemble model

Krogh e Vedelsby (1995) proved that the error of an ensemble can be divided in two components, one that represents the average error of the individual components and one that measures the decorrelation of the components. The conclusion of their work is that the good performance of an ensemble depends on the individual components

having high accuracy and being as different from each other as possible (Krogh and Vedelsby, 1995). On his work, Dietterich (2000) claims that precision and diversity are the sufficient and necessary conditions for an ensemble to have better performance than any of its individual components alone (Dietterich, 2000).

One efficient method of combining different models that is proven to improve the model's accuracy is the stacking method. Stacking has already won several competitions from the site Kaggle (a platform which hosts machine learning competitions).

Different from bagging and boosting, stacking use the concept of meta-learner. First a set of base learners is used to learn part of the dataset that is left for training, then these same base learners make predictions on the other part of the dataset that is left to testing. A higher-level learner, called meta-learner is trained using the predictions from the previous step as input.

The stacking approach used on this project in order to increase model's accuracy is shown in Fig. 8. For the base learners were used the KNN, GBM and Extra-Trees algorithms, and for the meta-learner a ANN with an MLP structure containing 50 neurons on its hidden layer were used.

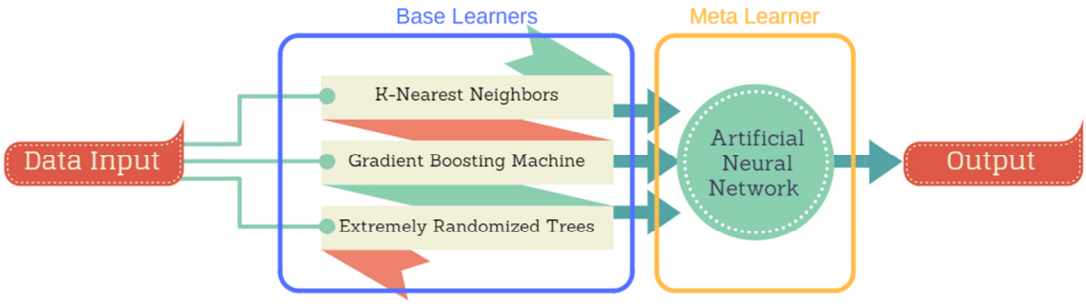


Figure 8. Representation of the stacking approach used on this work

4. DESCRIPTION OF CASE STUDIES

The primary goal of the evaluation was to analyze how well do ML techniques perform when applied to the task of detecting and identifying faulty components in industrial processes. For this we approached two different datasets, which represent distinct situations, introducing different features and cases. Both the evaluated datasets are publicly available through UCI Machine Learning Repository (Lichman, 2013).

4.1 Robot Execution Failures dataset

The execution of robotic tasks with robustness represents a difficult learning problem. In this dataset the target class indicates whether there was a collision during the transfer of a part, the possible values for the target are: normal, front collision, back collision, collision to the right and collision to the left. The attributes are forces and torques measured during a window of 315 *ms* after a failure detection, and are all integer-valued. The total number of attributes is 90 and there is a total of 463 examples (Lichman, 2013).

The robot execution failures problem includes 5 distinct datasets, each of them defining a different learning problem: LP1: failures in approach to grasp position, LP2: failures in transfer of a part, LP3: position of part after a transfer failure, LP4: failures in approach to ungrasp position, LP5: failures in motion with part. The examples and their real classes are distributed among the datasets as shown in Tab. 1.

Table 1. Feature information and class distribution of the robot execution failures.

Dataset	Number of Instances	Classes
LP1	88	4 classes (1 = 24%; 2 = 19 %; 3 = 18%; 4 = 39%)
LP2	47	5 classes (1 = 43%; 2=13 %; 3 = 15%; 4 = 11%; 5 = 19%)
LP3	47	4 classes (1 = 43%; 2=19 %; 3 = 32%; 4 = 6%)
LP4	117	3 classes (1 = 21%; 2 = 62 %; 3 = 18%)
LP5	164	5 classes (1 = 27%; 2=16 %; 3 = 13%; 4 = 29%; 5 = 16%)

4.2 Steel Plates Faults dataset

The Steel Plates Faults case study introduces data records of superficial faults on stainless-steel leaves. The faults are classified in one of seven possibilities: Pastry, Z_Scratch, K_Scatch, Stains, Dirtiness, Bumps and Other_Faults. For each record 27 distinct numeric attributes that represent the geometry and contour of the fault are analyzed. The data acquisition goal was to train machine learning for automatic pattern recognition. The dataset includes 1941 instances, which have been labeled by different fault types. Table 2 presents the class distribution of the dataset (Lichman, 2013).

Table 2. Steel Plates Faults dataset class distribution.

Output Class	Number of Instances
Pastry	158
Z_Scratch	190
K_Scratch	391
Stains	72
Dirtiness	55
Bumps	402
Other Faults	673

5. RESULTS ANALYSIS

The experiments were performed on the following system: a Windows 10, 64bits OS, CPU (Computer Processing Unit) Intel core i7-4700HQ 2.4 GHz processor, Random Access Memory (RAM) 8 GB and Python 3.5.2 environment.

When working with multi-class classification problem in order to validate the model performance, the common hold-out approach where the original dataset is split in two parts one for training and one for testing, can present some issues depending on how the dataset is arranged, in some cases one of the parts could contain labeled observations that do not exist in the other, resulting in a low accuracy performance given the dataset distribution. One approach that can counter-act this matter, is by using *k*-fold cross-validation (cv). The k-fold cv, consists on dividing the original dataset in *k* equal parts, then leaving one part for testing and the remaining *k-1* for training. This procedure is repeated *k* times until all of the *k* parts where left for testing and validated once, the final accuracy score of the *k*-fold cv consist on the average of all accuracy scores of the *k* parts that were left for testing, resulting in a final model that has been able to see the dataset as a whole.

In order to validate the models for both case studies, a 10-fold cv algorithm was used in each technique, where the accuracy of the model was used as performance measure criteria for each algorithm. The accuracy of the model consists on the rate of the sum of all the correct predictions over the sum of all predictions, ranging from 0 to 1, where 0 means the worst accuracy and 1 the best accuracy.

Table 3. Results from the application of the ML approaches in both case studies.

Technique	Robot Execution Failures			Steel Plate Faults		
	Mean	Standard Deviation	Time (s)	Mean	Standard Deviation	Time (s)
Extra-Trees	0.9993	0.0007	26. 9544	0.7862	0.0383	14.8528
GBM	0.9994	0.0008	290.5881	0.7956	0.0377	146.4469
KNN	0.9993	0.0006	169.5949	0.7604	0.0354	0.2317
Ensemble	0.9997	0.0001	490.0025	1	0	180.2586

From Tab. 3, the results of the ML techniques applied for both cases present the mean, standard deviation and the processing time in seconds for the 10-fold cross-validation score. Tables 4 and 5 show the resultant confusion-matrix for the ensemble approach on both case studies.

Table 4. Confusion-Matrix of the ensemble application for the Robot Execution Failure Case

Feature	Back collision	Normal	Left collision	Front collision	Right collision
Back collision	32721	0	0	0	0
Normal	0	19	0	0	0
Left collision	0	2	6	0	1
Front collision	0	0	1	1	4
Right collision	0	0	0	1	4

Table 5. Confusion-Matrix of the ensemble application for the Steel Plate Faults Case

Feature	Pastry	Z_Scratch	K_Scatch	Stains	Dirtiness	Bumps	Other_Faults
Pastry	158	0	0	0	0	0	0
Z_Scratch	0	190	0	0	0	0	0
K_Scatch	0	0	391	0	0	0	0
Stains	0	0	0	72	0	0	0
Dirtiness	0	0	0	0	55	0	0
Bumps	0	0	0	0	0	402	0
Other_Faults	0	0	0	0	0	0	673

For the Robot Execution Failures case, all techniques including the ensemble approach achieved a high value of accuracy, the ensemble approach reached 99.97% although very close to the values resultant from the techniques applied alone, the ensemble produced the highest performance. As the results of the ensemble have a low but existent error of 0.03% its confusion-matrix from Tab. 4 presents some false negatives and false positives resultant from the misclassified observations.

For the Steel Plate Faults case the ensemble approach clearly shows a higher accuracy value than the other techniques applied alone, reaching a 100% of accuracy where GBM being the best score of the three techniques applied alone only achieve 79.56%. As a result of 100% of accuracy, as shown by Tab. 5, there is no misclassified observations at all, proving to be an efficient method for improving the accuracy of classifier models when applied on fault detection problems.

6. CONCLUSION AND FUTURE RESEARCH

Ensemble methods consist in using multiple models to obtain better predictive performance than that could be obtained from any of the constituent models. This technique has been proved very effective in many applications. In this paper, the performance of three different machine learning techniques that present different approaches applied alone and also in an ensemble form using stacking and an ANN as meta learner, was evaluated on two case studies regarding fault detection, robotic execution failure and steel plate faults.

All of the three techniques alone had good performance results, however the ensemble application reached higher values of accuracy. Where for the robotic execution failure problem the ensemble application reached 99.97% against 99.94% of the best technique applied alone, and for the steel plate faults the ensemble reached an amazing 100% of accuracy against only 79.56% of the best technique applied alone.

Ensemble applications, that make use of meta-learner concept as stacking, proved to be an efficient approach for combining weak learners into a final strong learner, improving accuracy of the models and providing good and stable results.

An important part when working with datasets is preparing the data to be used by the machine learning algorithms, dealing with missing values, normalizing, removing redundant information, creating new variables, among others. Future research would include trying to automate this process of data preparation so that it would be possible to clean and prepare the data without great specific knowledge about the dataset.

7. ACKNOWLEDGEMENTS

The authors would like to thank National Council of Scientific and Technologic Development of Brazil - CNPq (grants: 404659/2016-0 and 303908/2015-7-PQ), and “Fundação Araucária” (grant number: 117/2014) for the financial support of this work.

8. REFERENCES

Breiman, L., 2001, “Random forests”. *Machine Learning*, Vol. 45, p. 5-32.

Chen, T., and Guestrin, C., 2016. “XGBoost : Reliable large-scale tree boosting system”. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’16)*, San Francisco, CA, USA, p. 785-794.

Cover, T., and Hart, P., 1967. “Nearest neighbor pattern classification”. *IEEE Transactions on Information Theory*, Vol. 13, p. 21-27.

Cutler, D.R., Edwards, T.C., Beard, K.H., Cutler, A., Hess, K.T., Gibson, J., and Lawler, J.J., 2007. “Random forests for classification in ecology”. *Ecology*, Vol. 88, p. 2783-2792.

Fawagreh, K., Gaber, M. M., and Elyan, E., 2014. “Random forests: from early developments to recent advancements”. *Systems Science & Control Engineering*, Vol. 2, p. 602-609.

Friedman, J.H., 2001. “Greedy function approximation: A gradient boosting machine”. *Annals of Statistics*, Vol. 29, p. 1189-1232.

Geurts, P., Ernst, D., and Wehenkel, L., 2006. "Extremely randomized trees". *Machine Learning*, Vol. 63, p. 3-42.

Ho, T.K., 1995. "Random decision forests". In *Proceedings of the 3rd International Conference Document Analysis and Recognition*. Montreal, Quebec, Canada, p. 278-282.

Keck, T., 2016. "FastBDT: A speed-optimized and cache-friendly implementation of stochastic gradient-boosted decision trees for multivariate classification". p. 1-16. Retrieved from http://arxiv.org/abs/1609.06119

Kohonen, T., 1990. "The self-organizing map". *Proceedings of the IEEE*, Vol. 78, p. 1464-1480.

McCulloch, W.S., and Pitts, W., 1943. "A logical calculus of the ideas immanent in nervous activity". *The Bulletin of Mathematical Biophysics*. Vol. 5, p. 115-133.

Mitchell, T.M., 1997. *Machine Learning*. McGraw-Hill Science/Engineering/Math, Princeton, NJ, USA.

Ren, Y., Zhang, L., and Suganthan, P.N., 2016. "Ensemble classification and regression: Recent developments, applications and future directions". *IEEE Computational Intelligence Magazine*. Vol. 11, p. 41-53.

Rojas, R. (1996). "The Hopfield model". *Neural Networks*, In: Mean Field Models for Spin Glasses. Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge / A Series of Modern Surveys in Mathematics, vol. 55. Springer, Berlin, Heidelberg, p. 338-372.

Rokach, L., 2010. "Ensemble-based classifiers". *Artificial Intelligence Review*. Vol. 33, p. 1-39.

Rosenblatt, F., 1957. "The perceptron: a perceiving and recognising automation". Report No. 85-460-1, Cornell Aeronautical Laboratory, Buffalo, NY, USA.

Samuel, A.L., 1959. "Some studies in machine learning using the game of checkers". *IBM Journal of Research and Development*, Vol. 3, p. 210-229.

Tahmasebi, P., and Amirkabir, A.H., 2011. "Application of a modular feedforward neural network for grade estimation application of a modular feedforward neural network for grade estimation". *Natural Resources Research*, Vol. 20, p. 25-32.

Tan, S., 2006. "An effective refinement strategy for KNN text classifier". *Expert Systems with Applications*, Vol. 30, p. 290-298.

Urraca, R., Martinez-de-Pison, E., Sanz-Garcia, A., Antonanzas, J., and Antonanzas-Torres, F., 2016. "Estimation methods for global solar radiation: Case study evaluation of five different approaches in central Spain". *Renewable and Sustainable Energy Reviews*, Vol. 77, p. 1098-1113.

Wang, X., Feng, H., and Fan, Y. (2015). "Fault detection and classification for complex processes using semi-supervised learning algorithm". *Chemometrics and Intelligent Laboratory Systems*. Vol. 149, p. 24-32.

Zamarren, M., and Gonza, P. A. (2005). "Prediction of hourly energy consumption in buildings based on a feedback artificial neural network". *Energy and Buildings*, Vol. 37, p. 595-601.

9. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.