CIGI 2021
Quality of sawmilling output predictions according to the size of the lot - The size matters!

VINCENT MARTINEAU[1,2], JONATHAN GAUDREAULT[1,2], MICHAEL MORIN[1,3], STEVE VALLERAND[4]

[1] FORAC Research Consortium
[2] Department of Computer Science and Software Engineering
[3] Department of Operations and Decision Systems
Université Laval, Québec, QC, Canada
vincent.martineau.1@ulaval.ca
jonathan.gaudreault@ift.ulaval.ca
michael.morin@osd.ulaval.ca

[4] FPInnovations, Québec, QC, Canada
steve.vallerand@fpinnovations.ca

***Résumé* – Lors de l'évaluation de modèles d'apprentissage automatique supervisé, on considère généralement le rendement de prédiction moyen obtenu sur les tests individuels comme mesure de choix. Toutefois, lorsque le modèle est destiné à prédire quels produits du bois seront obtenus lors du sciage de certains billots, c'est généralement la performance pour un lot complet qui importe. Dans cet article, nous montrons l'impact de cette nuance en termes d'évaluation du modèle. En fait, la qualité d'une prédiction (globale) s'améliore considérablement lorsque l'on augmente la taille des lots, ce qui offre un solide soutien à l'utilisation de ces modèles en pratique.**

***Abstract* – When comparing supervised learning models, one generally considers the average prediction performance obtained over individual test samples. However, when using machine learning to predict which lumber products will be obtained when sawing logs, it is usually the performance over the entire lot that matters. In this paper, we show the impact of this by evaluating a model performance for various batch sizes. The quality of a (global) prediction improves tremendously when batch size increases, which offers a strong support for the use of such models in practice.**

***Mots clés* - Simulation de débitage, évaluation de modèles d'apprentissage supervisé, application d'apprentissage automatique, industrie des produits forestiers.**
***Keywords* – Sawing Simulation, Supervised Learning Models Evaluation, Machine Learning Application, Forest products industry.**

1 INTRODUCTION

For many reasons, the forest product industry needs forecasting what its inventory would allow producing. There exist needs for performance forecasting, configuration decisions and wood allocation between several plants. It has been shown that accurately forecasting sawmills output can improve profitability due to better decision-making [Morin et al. 2020]. Tools for such evaluations include sawmill simulators, such as the Optitek [FPInnovations, 2014]. These tools are designed to simulate the transformation process of a log at a given plant [Lin et al. 2011]. However, simulation is computationally expensive with large volume of woods, for companies owning multiple plants or when testing multiple sawmill configurations. As a result, supervised learning has been recently proposed to improve or replace slow simulators by a fast relation from the space of the logs characteristics to the space of the sawmill outputs, i.e. lumbers [Morin et al. 2015].

Morin et al. (2020) showed that, although supervised learning models produced for that particular problem are only approximate, they remain more than useful for decision-making, e.g. to make decision regarding wood allocation between sawmills. In this paper, we explain why seemingly low performance on individual logs does not hinder a models' usefulness for decision-making. As demonstrated by our experiments, the errors in the prediction for some given logs are partially compensated by the errors on other logs as the batch size increases. In addition, sophisticated models, i.e. models trained using a machine learning procedure, proved useful and more accurate than a simple average for all batch sizes and especially for smaller batches. This explains the good performance of such models for decision-making, especially on the wood allocation problem [Morin et al. 2020]. It makes these models especially promising in the specific use cases where a simulator tends to be too slow, i.e. on large amount of data.

The paper is organized as follows. We first review the related literature and concepts in Section 2. In Section 3, we present the experimental process and data supporting our contributions, interpret the precision, the recall and the F1-score performance

measure in the context of our machine learning problem, and present the results. We conclude in Section 4.

2 RELATED CONCEPTS AND LITERATURE

In the following subsections, we introduce general notions regarding sawing simulation as well as supervised learning for sawing simulation.

2.1 Sawing Simulators

Sawing simulators make it possible to anticipate which products a log will generate when processed at a given sawmill. These tools generally take three inputs, namely a sawmill model (described in a suitable formal language), a feasible product list, and a virtual description of the log.

The *sawmill model* details the equipment, the links between the machines as well as their capacities. The *feasible product list* describes the lumbers that can be produced by the factory along with their value. Log sawing by-products, such as sawdust and shavings, are also associated with values. A product value can either be based on its true market value, or it can reflect its importance w.r.t. the current needs of the company. Finally, a *virtual log* is a representation of a log that needs to be cut, often a three-dimensional scan. It is generally modeled as a point cloud approximating the surface of the log [Thomas 2013].

Simpler representations, such as *parametric descriptions*, might be used in some studies. Parametric descriptions are vectors of characteristics such as the length of the log, its strong diameter, its weak diameter, its curvature, its tapering, and its volume. Other characteristics, such as the log species, could be introduced in the parametric description.

Given a sawmill model, a feasible product list and a virtual log, the sawing simulator performs a simulation of all the feasible cuts to choose the one leading to an optimal yield for this log, i.e. the simulator finds a set of lumbers maximizing the total value of the transformed log [Todoroki 1990]. An example of a sawing simulation result can be seen in Figure 1.

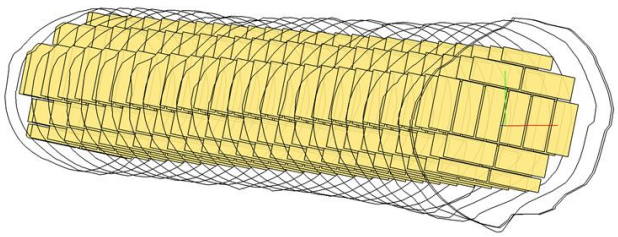

Figure 1. A virtual log and the optimal basket of products obtained by the sawmill simulation.

2.2 Supervised Learning for Sawing Simulation

Although there exist some machine learning model using the three-dimensional log scans as input (e.g. [Selma 2018]), we focus in this paper on models using the parametric description of a log. Both types of models provide as output a vector of counts for each feasible lumber products. This vector of counts describes the *basket of products* one should obtain if processing the log at the modeled sawmill.

Supervised learning algorithms aims to build a mapping from a feature vector space to a label space using a training set of examples consisting of pairs of inputs and outputs. In our context, the features are the logs parametric descriptions, and the labels are the vectors of counts representing the baskets of products.

In supervised learning, there are two main types of output labels, namely discrete (and finite), and continuous. The former leads to what we call a classification problem whereas the latter leads to a regression problem. In what follows, we describe how basket of products, which are vectors of counts, can be encoded as labels in both the classification and the regression cases.

2.2.1 Classification for Sawing Simulation

A class is a discrete label. For classification purposes in the sawing simulation context, it is possible to consider each individual basket of products encountered in the training set, i.e. combination of lumber products, as a specific class.

The drawback of this approach is that each specific combination needs to be encountered in the data. Therefore, it is possible that some feasible combinations of products, i.e. some feasible baskets, cannot be predicted by the model. One example would be a class that is present in the test dataset and not in the training dataset. Conceptually, one of the main benefits of the approach is that all the predictions are guaranteed to be feasible for at least one existing log (as they have been seen at least once in the data).

2.2.2 Regression for Sawing Simulation

When performing a regression in our context, a regression model individually predicts the number of products of each feasible lumber from the feasible product list (i.e. each position in the vector count).

Contrary to classification it is now possible to predict any basket of products even if they were not part of the train set. For each of the feasible product, the model provides a real number. The drawback of the approach is that there are no such things as decimal lumber counts in practice. This might be a problem when using the model for applications that need predictions for individual logs. However, in many applications, there is few need to obtain precise counts and we aim at evaluating a global prediction, i.e. the prediction on a batch of logs. As a result, we are using the real values provided by the model directly to compute the performance on a batch of logs.

This approach comes with some challenges. One way to address the problem would be to create one model per product output. The problem with this approach is that it assumes independent outputs. In the case of our application, individual entries in the count vector are related to one another. That is, each predicted product might take the space needed for another. For that reason, we used a single model trained to predict the entire vector of counts. The specific procedure depends on the chosen learning algorithm.

2.2.3 Learning Algorithms for Sawing Simulation

In the literature, multiple learning algorithms have been evaluated in the context of sawing simulation. In this paper, we focus on three machine learning algorithms that were proved efficient for that problem by Morin et al. (2015), namely k-nearest-neighbors (KNN) [Fix and Hodges Jr. 1951], decision tree [Breiman 1996], and random forest [Breiman 2001]. We experiment with both the regression and the classification version of these algorithms.

2.2.3.1 k-Nearest-Neighbors

The KNN algorithm is based on the idea that similar logs should have similar product baskets. During training, the algorithm will

use a data structure that makes it easy to compare the distance between two logs. In this study, we use the Euclidian distance between the parametric description of two logs. The algorithm chose the *k* nearest logs.

In classification mode the algorithm performs a majority vote to determine the class whereas in regression mode it averages the basket of the nearest *k* logs. Although we tested multiple values of *k*, a value of one allowed us to achieve good results in training while maintaining a good generalization for examples that were not seen in the training phase. Recent applications of KNN has shown that it performs well on unbalanced data, e.g. [Cai 2020]. Since our dataset is highly unbalanced, it made the algorithm a promising choice in the context of our application.

2.2.3.2 Decision Tree

The decision tree learning algorithm uses the training examples to generate a tree-based model where the nodes are simple decision rules based on the input features. For prediction, an unseen example is passed through the tree and the model output is determined when it reaches a leaf.

In classification mode, each leaf represents the class to be predicted which is determined by the examples the training set that reached that leaf during training (by a majority vote). In the case of regression, we use a variation of the algorithm to predict multiple output values. Although, it would be possible to create a set of independent trees, product counts in the basket are dependent. We therefore used a single tree that can predict the entire basket at once such as described in [Borchani 2015]. Recent applications of the decision tree algorithm assessed it as a simple, efficient, and easy to interpret model [Lan 2020].

2.2.3.3 Random Forest

The random forest algorithm builds multiple decision trees. It does so by sampling the training set N times. Each sample is used to create a single tree [Schonlau 2020]. For prediction, the data are passed through each tree and the predictions are combined. In the case of classification, a majority vote is performed whereas in regression the trees outputs are averaged.

3 EXPERIMENTS

Our experiments are intended to show the impact of the size of the lot on the quality of the prediction for the prediction models using the KNN, the decision tree and the random forest learning algorithms.

In this section, we present the industrial data used for this study (3.1), the details of the model-building phase (3.2), and the evaluation metrics we use to compare the quality of the models (3.3). Finally, we analyze and compare the performance of our models on three aspects. First, by exploring the average prediction quality on individual logs (3.4). Second, by comparing the performance on batches of increasing size (3.5). Third, by highlighting the most common source of error in predictions (3.6).

3.1 Data

Data was provided by FPInnovations. Our dataset contains a total of 2235 logs. All the logs from this dataset are known to produce a non-empty basket of products. To train the supervised learning models, we used the following log characteristics: length, strong diameter, weak diameter, curvature, taper, and volume.

We used the Optitek sawing simulator [FPInnovations, 2014] to virtually transform each of these 2235 logs and obtain the actual basket of products. A total of 85 different sawing products are present in those vectors leading with 85 entries.

We obtained 1188 different baskets of products which leads to 1188 classes when considering the classification version of the problem. Considering the high number of classes relative to the number of examples in our dataset, it is expected that several classes will have a low representation. For example, there are roughly 30% of classes that are only present once in the dataset while one of the classes is present 58 times. This particularity of the dataset implies that certain classes in the training set will not be present in the test set (and vice-versa).

This class distribution is particularly complex for classification models because it is expected that these models do not try predicting classes that have not been seen in training. However, models should predict similar classes that are present at training. In this situation even if the classes are different, it is possible that these two classes have several common elements.

It is important to mention that the logs were not chosen for the purpose of having a balanced input or output. This means that some lengths or diameters are more common than others and some only appear once. The same thing is true for the basket of products, hence the classes.

Figure 2 shows a histogram of the length of the logs in the complete dataset. The figure allows us to see that some lengths are much more frequent and that some lengths are very rare and have only 1 example (5.43 meters). Furthermore, we see that

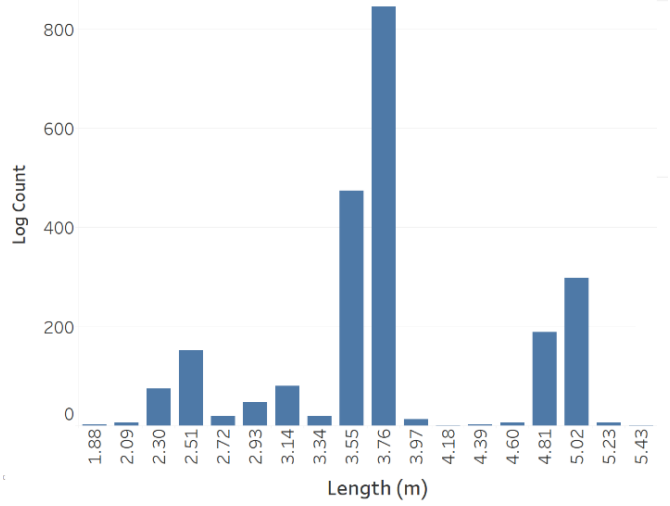

Figure 2. Distribution of logs by length (m) in the dataset

around 800 logs have a length of 3.76 meters. This distribution is expected to complicate the task of fitting a model on the data.

3.2 Model Training

To carry out the experiments, the logs in the dataset is randomly separated so that 85% of the logs are used for training, i.e. to fit a model, and 15% are used to form a test dataset to evaluate the performance of the model on unseen data. We created 10 such partitions which will allow the experiments to be repeated 10 times for each of the algorithms.

For all splits, we first trained models using the KNN, the decision tree, and the random forest algorithms on the training set. Furthermore, to establish a base case reference, we also consider a *dummy* model which systematically predicts the

average basket of products as seen in the train set. It should be noted that the dummy model prediction procedure is akin to simple forecasting heuristics such as using the historical average of the production.

3.3 Performance Measure

To evaluate the performance of the learned models, we need to assess how the basket of products they predict (vector *y*) compares to the actual basket of products (vector *b*) generated by the simulator. Vectors *y* and *b* may either correspond to the quantities obtain from the sawing of a given log or of a whole batch of logs.

In what follows, we use the well-known *precision*, *recall* and *F1-score* metrics to evaluate the models [Van Rijsbergen 1979, Powers 2020]. We recall that precision and recall are calculated using the number of true positives (*TP*), the number of false positives (*FP*), and the number of true negatives (*TN*) [Van Rijsbergen 1979]. In our context, we define *TP*, *FP*, and *TN* as follows:

- *TP* represents the number of lumbers that are present in both vectors.
- *FP* is the number of lumbers predicted but not actually produced.
- *FN* is the number of lumbers that were manufactured but not predicted.

Given a predicted basket *y* and an actual basket *b*, the precision is defined as

$$precision(y,b) = \frac{TP(y,b)}{TP(y,b) + FP(y,b)},$$

and the recall is defined as

$$recall(y,b) = \frac{TP(y,b)}{TP(y,b) + FN(y,b)}.$$

Finally, the F1 score is computed as the harmonic mean between precision and recall which leads to the following equation:

$$\mathrm{F1}(y,b) = \frac{2TP(y,b)}{2TP(y,b) + FP(y,b) + FN(y,b)}.$$

It should be noted that by evaluating a set of models built using a learning algorithm and comparing their average results to the average results of models built using another learning algorithms we are able to determine which algorithm generates the best models either for individual logs or for a batch of logs.

3.4 Performance on Individual Logs

This section is used to demonstrate the quality of our prediction for individual logs. In the case of individual log predictions, we present the performance metrics for individual predictions averaged on a batch of unseen logs and multiple replications where a replication consists of the following three steps (as described in Section 3.2). First, we partition the dataset into a training set and into a test set. Second, we use each learning algorithm to build a model based on the data of the training set. Third we evaluate each of the generated models on the test set by using it to make a prediction of the basket of each individual log it contains. The quality of the prediction for individual logs in terms of a specific metric, e.g. the F1-Score, is then averaged.

Figure 3 reports the average F1-Score for the different models we are using in this research. We are comparing the decision tree, KNN and random forest models to the dummy predictor.

All models perform much better than the dummy prediction. This result can be explained by the large variety of feasible basket of products in our dataset and their uneven distribution. This characteristic of the dataset proves to be problematic for the dummy model.

Among all the tested learning algorithms, KNN shows the worst performance. As it was shown by Selma et al. (2015), similar logs could have very different basket of products which hinders nearest neighbors approaches efficiency. Although, the difference between KNN and the dummy predictor is 28%.

Tree-based machine learning algorithms (decision tree and random forest) performed best. This is in line with the result of Morin et al. (2015). What is interesting is that the regression and classification models have similar performance. This result highlights the possibility to achieve good prediction even if the training dataset does not contain at least one example of each feasible basket of products.

Using regression also provides an advantage over classification. For our experiments, the number of classes is defined by the number of different basket of products present in our dataset. One problem with classification is that small training datasets are very likely not to contain all feasible classes. As a result, a classification model output is likely to be wrong on at least some of the products in the baskets. In regression, this is less of an issue since the model can output an unseen basket.

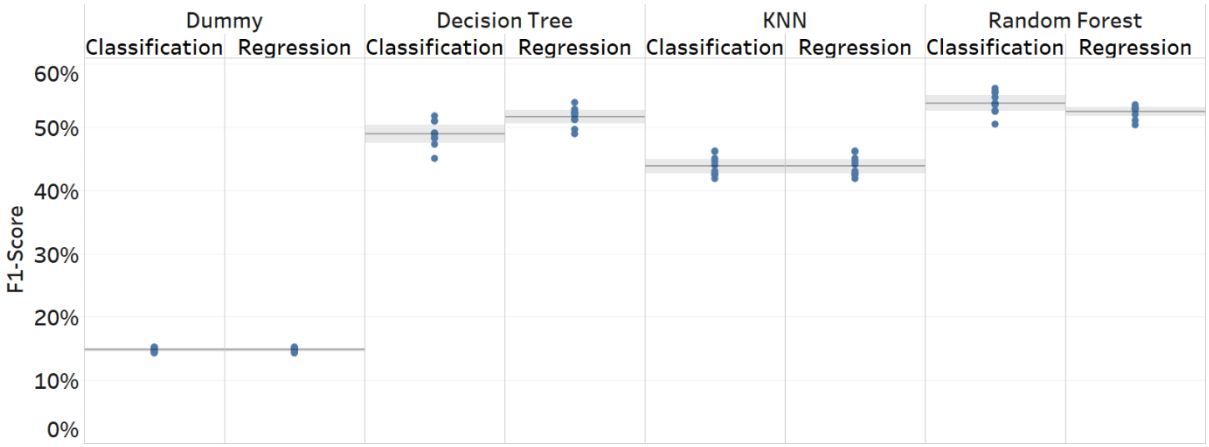

Figure 3. Average F1-score for individual predictions on the test set (95% confidence intervals; 10 replications; individual replication scores are shown in blue)

Finally, the model with the best performance is the random forest in classification. In this case the algorithm is performing 39% better than the dummy predictor.

3.5 Performance for Batches

In this section, we explore the effect of the size of the batches on the quality of the forecasts. We also validate our initial hypothesis that the errors made by a model when predicting the output of a sawmill for an unseen log is compensated for by errors made on other logs. Finally, we confirm that sophisticated models are useful for all tested batch sizes and especially on smaller batches where a simple average does not achieve a sufficient performance.

Figure 4 reports the F1-score for the prediction of the global basket of products associated with the whole test dataset. We can see a significative improvement for all models. All models improve by at least 40% in terms of F1-score. The dummy predictor has the largest improvement (approx. 76%). This result is not surprising because with a large enough sample the average should improve to be closer to a perfect prediction. This establishes a base case for the other algorithms.

It should be noted that the random forest algorithm in classification is no longer the best performer and the accuracy of its prediction is only 89%. Not only is this model no longer the best performer, but it is now our worst performer. KNN has a F1-score of about 90%, which is lower than that of the dummy predictor. The best performer is random forest in regression with a F1-score of 93%.

Figure 5 illustrates more precisely how the performance is affected by the size of the batch. We kept the same test dataset but report global results for randomly chosen subsets of different sizes.

We notice that the performance curve is logarithmic. The batch size does not need to be huge to see dramatic performance improvement. This renders the models useful for a wide range of applications involving small batches of logs such as log piles, e.g. for wood allocation purposes. Companies can therefore achieve interesting results in decision-making on such problems using such models although they appeared to be only approximate for individual predictions.

In classification all models perform better than dummy for smaller batch size; the smaller the batch is, the greater the difference. For example, decision tree will perform 42% (in terms of F1-score) better than dummy for batch size of 1. At batch size of 50, we can see that dummy is beating the KNN algorithm and will do better than random forest at batch size of 110. With our experiment dummy never reach the level of prediction of the decision tree, but with a larger test dataset it is possible that dummy would yield better result than the decision tree classifier.

For regression, the picture is different. Again, for small batches all our algorithms perform better than the dummy predictor and again the dummy predictor performs better than KNN (starting at a batch size of 50). However, this time decision tree and random forest are doing better than dummy for batch size of 330. Not only do these models perform better, but random forest has a prediction that is 3% over the dummy in terms of F1-score.

This clearly supports the fact that the regression version of random forest dominates the other approaches and would be a safe choice whatever is the size of the lot.

Figure 6 details the F1-score, precision, and recall metrics according to the size of the batch for the best classification model (decision tree) and the best regression model (random forest). Recall and precision are relatively close to each other regardless of the size of the lot. For both models, recall and precision tends to both increase with batch size. However, we can point that the difference is much smaller for decision tree in classification mode. The distance between precision or recall and the F1 score is of about 0.2%. For random forest in regression mode, the spread is larger, and the difference is slightly over 1%. We also notice that the difference tends to be larger for smaller batch sizes than for larger batch sizes.

Figure 7 shows the 95% confidence intervals around the F1-score average for the various batch size. The spread of the 95% confidence interval diminishes significantly as batch size increases. This is especially true for small batches, e.g. when increasing the size of the batch from 1 to 10 logs. Of course, for small batches, it is possible to make a prediction that is wrong that has a large impact on the overall prediction for the batch. The fact that the interval is narrowing as the size of the batch increases confirms our hypothesis that errors tend to be canceled.

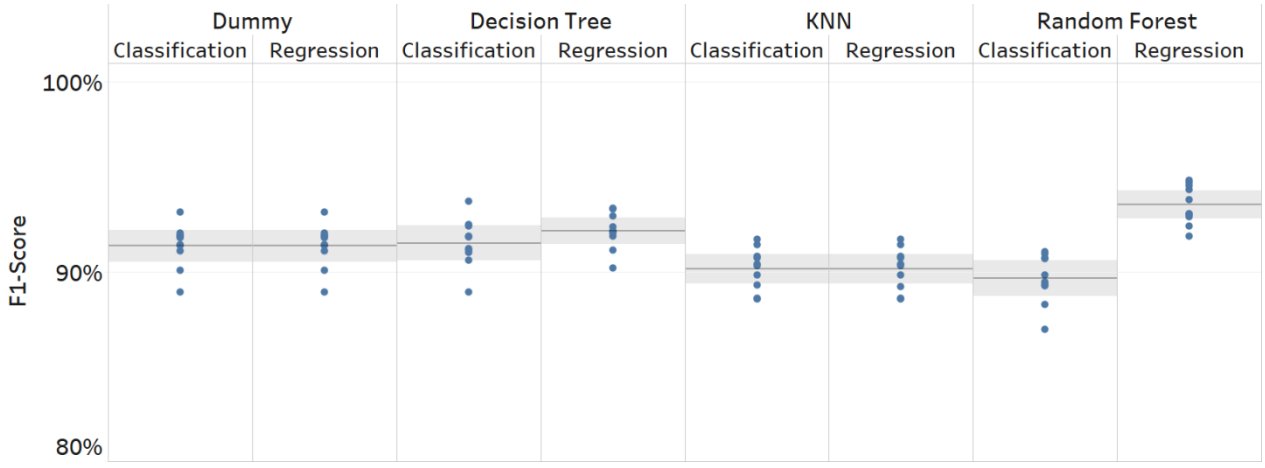

Figure 4. Average F1-score for global predictions on the test set along (95% confidence intervals; 10 replications; individual replication scores are shown in blue)

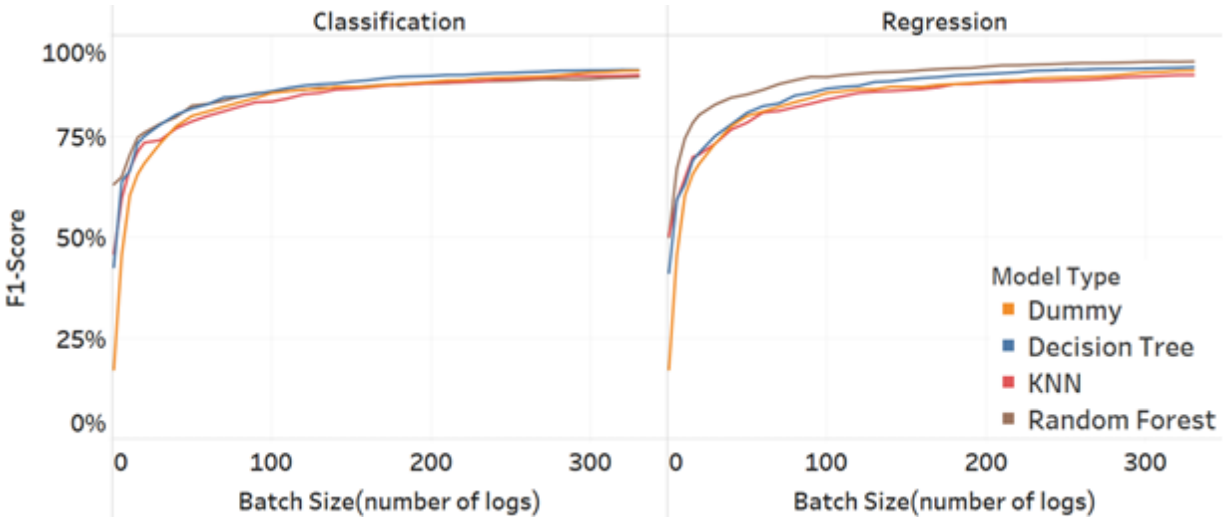

Figure 5. F1-score (in %) for KNN, Decision Tree, Random Forest, and Dummy as a function of the size of the batch (number of logs)

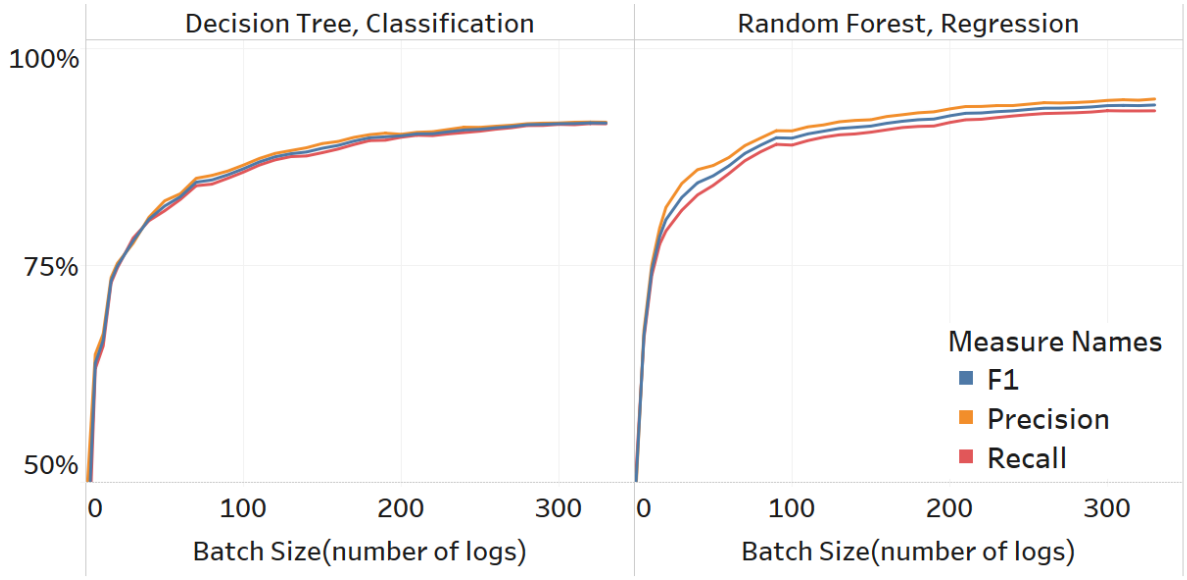

Figure 6. F1-score, precision, and recall (in %) as a function of the size of the batch (number of logs) for decision tree in classification and random forest in regression

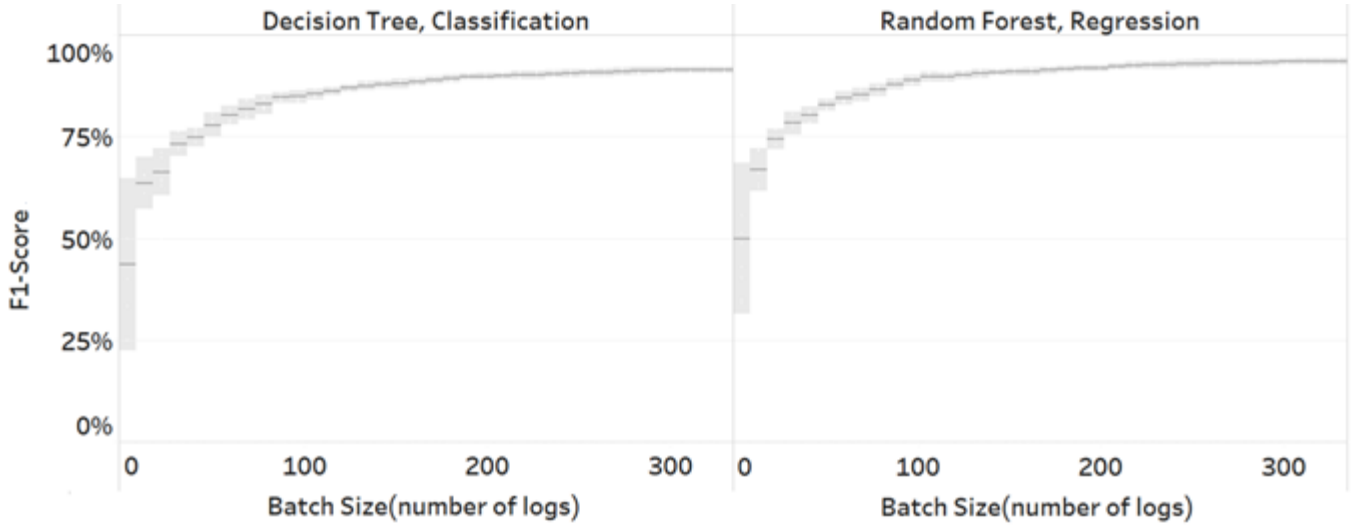

Figure 7. 95% confidence intervals for F1-score as a function of the batch size (number of logs)

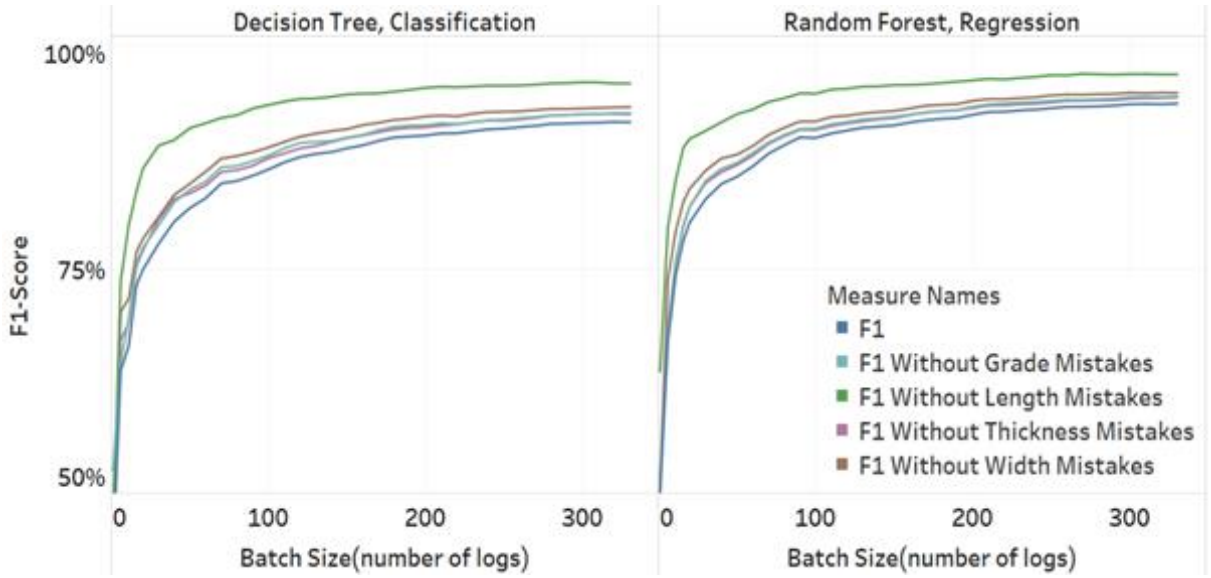

Figure 8. F1-score with and without single attribute mistakes for all types of error as a function of the size of the batch (number of logs)

3.6 Sources of error

Each given type of lumber product has a *length*, a *width*, a *thickness*, and a *grade* which, taken together, fully define the product. We call these characteristics the *lumber attributes*. By artificially ignoring errors on a specific lumber attribute, we artificially increase the score of the original model. Let us suppose we choose ignoring the length errors. Then, the magnitude of the score increment reflects the number of times the original model predicted the wrong length for a lumber without making a mistake on the other attributes.

The idea behind this analysis is to visualize "where" our models produce errors, but also to understand how to improve them in the long run. If, for instance, a model that predicts products that have the right width, length, and thickness is frequently misled on the grade, then it may be possible that the parametric data does not provide enough information to properly predict the grade. In this case it may be necessary to add information in the parametric description of the logs to improve the model.

Figure 8 reports the F1-score when we do not consider a specific type of mistakes. It shows that the length is the main source of error in our predictions. This is especially true in the case we are using a decision tree classifier as the curve without length mistakes is clearly above the others.

The curve representing the errors along the length is the curve that detaches most from the other curves. This suggests a higher error rate on length compared to other lumber attributes. Of course, other attributes also appear as sources of errors when inspecting the other curves, but the magnitude of the gain when ignoring them is less.

4 CONCLUSION

We showed how, in the context of supervised learning for sawing simulation approximation, the quality of a model can not only be quantified by its performance on individual logs, but also by its performance on batches. These results confirm that prediction errors on one log can be compensated for by errors on other logs, but also shows why more sophisticated models are preferable than a simpler model based on the historical average, especially for small batches.

Furthermore, we presented metrics that can be easily interpreted in an industrial context in the forest-product industry. Since those metrics make it easier to understand what is happening with the models in terms of over and under prediction of lumber counts, it becomes possible to explore the sources of errors in the models and interpret them in the context of the application. Those analyses pave the way to further model improvements.

5 ACKNOWLEDGEMENT

The authors would like to thank the FORAC Research Consortium and its partners. Our gratitude goes as well to the Natural Sciences and Engineering Research Council of Canada (NSERC) who provided funding for this research.

6 REFERENCES

Andersson, K., & Luttropp, C. (1997). Design for disassembly: Computer aid for separating surfaces and sorting borders. *Schriftenreihe WDK*, 351-354.

Borchani, H., Varando, G., Bielza, C., & Larranaga, P. (2015). A survey on multi-output regression. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 5(5), 216-233.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1996). Classification and Regression Trees. Belmont, CA: Chapman and Hall.

Breiman, L. (2001). Random Forests. Machine Learning, 45 (1): 5–32.

Cai, L., Yu, Y., Zhang S., Song, Y., Xiong, Z. and Zhou, T., "A Sample-Rebalanced Outlier-Rejected k-Nearest Neighbor Regression Model for Short-Term Traffic Flow Forecasting," in *IEEE Access*, vol. 8, pp. 22686-22696, 2020.

Fix, E., & Hodges Jr., J. L. (1951). Discriminatory Analysis – Nonparametric Discrimination: Consistency Properties. Tech. Rep., DTIC Document.

FPInnovations (2014). Optitek 10. User's Manual.

Lan, T., Hu, H., Jiang, C., Yang, G., & Zhao, Z. (2020). A comparative study of decision tree, random forest, and convolutional neural network for spread-F identification. *Advances in Space Research*, *65*(8), 2052-2061.

Lin, W., Wang, J., & Thomas, R. E. (2011). A three-dimensional optimal sawing system for small sawmills in central Appalachia. In: Proceedings, 17th central hardwood forest conference; 2010 April 5-7; Lexington, KY; Gen. Tech. Rep. NRS-P-78. Newtown Square, PA: US Department of Agriculture, Forest Service, Northern Research Station: 67-76. (Vol. 78, pp. 67-76).

Morin, M., Paradis, F., Rolland, A., Wery, J., Gaudreault, J., & Laviolette, F. (2015). Machine learning-based metamodels for sawing simulation. In 2015 Winter Simulation Conference (WSC) (pp. 2160-2171). IEEE.

Morin, M., Gaudreault, J., Brotherton, E., Paradis, F., Rolland, A., Wery, J., & Laviolette, F. (2020). Machine learning-based models of sawmills for better wood allocation planning. International Journal of Production Economics. 222. 107508.

Powers, David M. W. (2020). "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". Journal of Machine Learning Technologies. 2 (1): 37–63.

Selma, C., Bril El Haouzi, H., Thomas P., Gaudreault J., & Morin M. (2018) An Iterative Closest Point Method for Measuring the Level of Similarity of 3D Log Scans in Wood Industry. In: Service Orientation in Holonic and Multi-Agent Manufacturing. Studies in Computational Intelligence, vol 762. Springer, Cham.

Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, *20*(1), 3-29.

Stock, G. N., Greis, N. P., & Kasarda, J. D. (1998). Logistics, strategy and structure. *International Journal of Operations & Production Management*.

Thomas, R.E. (2013). RAYSAW: A log sawing simulator for 3D laser-scanned hardwood logs. In: Proceedings, 18th Central Hardwood Forest Conference; 2012 March 26-28; Morgantown, WV; Gen. Tech. Rep. NRS-P-117. Newtown Square, PA: US Department of Agriculture, Forest Service, Northern Research Station: 325-334. (Vol. 117, pp. 325-334).

Todoroki, C. (1990). AUTOSAW system for sawing simulation. New Zealand Journal of Forestry Science.

Van Rijsbergen, C. J. (1979). Information Retrieval (2nd ed.). Butterworth-Heinemann.