An Interactive Visual Tool to Enhance Understanding of Random Forest Predictions

Ram B. Gurung, Tony Lindgren and Henrik Boström

Abstract Random forests are known to provide accurate predictions, but the predictions are not easy to understand. In order to provide support for understanding such predictions, an interactive visual tool has been developed. The tool can be used to manipulate selected features to explore *"what-if"* scenarios. It exploits the internal structure of decision trees in a trained forest model and presents this information as interactive plots and charts. In addition, the tool presents a simple decision rule as an explanation for the prediction. It also presents the recommendation for reassignments of feature values of the example that leads to change in the prediction to a preferred class. An evaluation of the tool was undertaken in a large truck manufacturing company, targeting the fault prediction of a selected component in trucks. A set of domain experts were invited to use the tool and provide feedback in post-task interviews. The

Ram B. Gurung · Tony Lindgren
Department of Computer and System Sciences (DSV), Stockholm University
Postbox 7003, SE-164 07 Kista, Sweden
✉ gurung@dsv.su.se
✉ tony@dsv.su.se

Henrik Boström
KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science
Electrum 229, 164 40 Kista, Sweden
✉ bostromh@kth.se

Archives of Data Science, Series A
(Online First)
KIT Scientific Publishing
Vol. 6, No. 1, 2020

DOI: 10.5445/KSP/1000098011/08
ISSN 2363-9881

result of this investigation suggests that the tool indeed may aid in understanding the predictions of a random forest, and also allows for gaining new insights.

1 Introduction

Machine learning models are getting widespread adoption in many domains where they are used to optimize operations by making useful predictions. Most of the state-of-the-art machine learning models are good at making accurate predictions, but typically have limited capabilities to explain their reasoning. Although the need for interpretable machine learning models has been an active research topic for quite some time, it has received serious attention only recently (Lipton, 2018), in particular due to changed legislation, e.g. the General Data Protection Regulation (GDPR). The recent adoption of the GDPR by the European Parliament gives the right to citizens to demand an explanation of decisions made by automated decision makers.

Research on model interpretability is being conducted along various dimensions, such as model-agnostic vs. model-specific approaches, interpreting whole models (global interpretation) vs. single predictions (local interpretation) (Guidotti et al., 2018). This paper focuses on explaining single predictions made by a random forest model (Breiman, 2001). A random forest is a set of (diverse) decision trees. When a test instance is predicted by a single decision tree, it will follow a path from the root of the tree to one of the leaf nodes. For each internal node, the subsequent node is chosen depending on the outcome of a test condition. The prediction is formed at the leaf from the training instances having reached that leaf, e.g., by using the most frequent label as the prediction. Hence, the logic behind each prediction can be easily traced, and also understood, at least if the features are understandable and the paths are not too long. However, since a random forest consists of a large number of trees, each prediction is formed from a large set of such paths, where the outcome is combined, e.g. by averaging, from the individual predictions. Hence, even if a single decision tree is interpretable, this does not entail interpretability of a forest of such trees. In order to provide some insights in their inner workings, random forest implementations usually provide some ranking of the features according to their discriminatory power by assigning them a *variable importance* score. However, this type of information is rather generic and does not help to explain a single prediction.

The main goal of this study is to investigate how a user of random forest may be supported to better understand its predictions. In order to achieve this, an interactive visual tool was developed, where a user can manipulate selected features of a test instance to explore "*what-if*" scenarios. Rather than presenting all the paths followed in a forest when making a prediction, it derives the aggregated information of these paths which are presented through dynamic plots. In order to investigate the effectiveness of such tools, a case study was undertaken at a truck manufacturing company (Scania CV AB). A small group of domain experts were requested to perform specific tasks using the tool and their experiences in doing so were gathered through interviews.

In the next section, we present some related work on explaining black box models. In Section 3, we outline the functionalities of the tool and describe a typical workflow of using it in Section 4. We then describe the design of the user evaluation in Section 5 and present and discuss the results in Section 6. Finally, we summarize the main conclusions and point out directions for future research in Section 7.

2 Related Work

Research on interpretability in machine learning has recently been re-vitalized with many recent publications on explaining black-box models. A survey of explanation methods is presented in Guidotti et al. (2018) which, together with the book on interpretable machine learning by Molnar (2019), provides a good overview of the field.

Random forest models are considered as black-box models. In order to address the opacity problem of random forests, Breiman (2001) introduced a so-called *variable importance* score which is calculated by measuring the effect of permuting the values of each feature on predictive performance. In this way, *variable importance* provides an indication of which features have a (relatively) high impact on the predictive performance. But the importance measure cannot explain the way in which the features impact the predictions. Partial dependency plots can be used to study how changes for some particular features impact the predictions, e.g. as in the visual tool proposed in Krause et al. (2016). The *inTree* (interpretable tree) framework, proposed by Deng (2019), uses frequent itemset mining to allow for interpreting random forests, where each path in a tree is represented by an itemset. Frequent itemsets are extracted

to be used as an interpretable classifier. A method proposed by Friedman and Popescu (2008) also extracts tree paths, but instead uses them as features in a linear model. This model acts as an interpretable approximation of the original model. Model approximation can also be achieved by building a simpler model, such as a decision tree, from a large, artificially generated data set along with predictions from the complex model, such as a random forest, that is being approximated (see e.g. Lindgren (2015)).

In addition to interpreting a model, another relevant task for interpretability is understanding its prediction. *LIME* (Ribeiro et al., 2016) is one of the most popular model agnostic approaches for this purpose. It randomly generates examples in the neighborhood of the test example and puts weights on them according to their proximity. A simple linear model is then built in this neighborhood using the weighted examples, which can be be used to show the importance of features as an explanation. LIME has been further improved with anchoring techniques which provide sufficient reasons for a model to make a certain prediction with confidence. Similarly, *DALEX*, a framework introduced in Biecek (2018), provides ways of explaining model predictions complemented with various plots. Lundberg and Lee (2017) introduced an unified framework named *SHAP* that can relate and compare various interpretable methods. There have been other approaches of explaining predictions of an instance, by comparing it to the closest instance from another class (Laugel et al., 2018). Similar to this, but specific to tree ensembles, is an actionable feature tweaking approach which suggests changes in some features of the example being predicted in order to change the original prediction (Tolomei et al., 2017).

3 Artifact

The proposed tool is a web-based interface developed using the R development environment (`www.r-project.org/`). The *Shiny* R package is one of the major components of the tool. The tool can be used for any random forest model trained on any classification data set. In this paper, the *Pima Indian Diabetes* data set, which is publicly available through the UCI Machine Learning Repository (Dua and Graff, 2017), is used to illustrate various features of the tool. Figure 1 shows the main interface of the tool, which provides information about the trained model, the test example being predicted and the predicted class

probabilities, where the class labels 1 and 0 denote diabetic and non-diabetic patients, respectively.

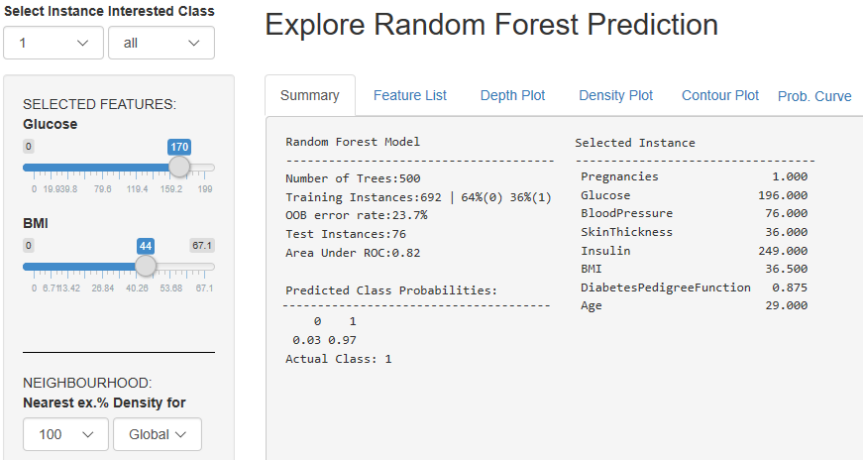

Figure 1: User interface showing the main dashboard of the tool.

3.1 Feature Ranking

The prediction made by a random forest is an aggregate result of the predictions made by each of its base trees. In each base tree, the test example follows one of the paths starting from a root to a leaf node. The path (branch) in the tree followed by the example is traced which constitutes a set of test conditions evaluated on some features. As shown in Figure 2, each extracted path is disintegrated into independent units in the format *(feature, split value, depth)*. Each unit represents a feature used to route the test example further down the path, a threshold (cutoff) value of the feature used for evaluating the test condition, and the depth of an intermediate node, where this evaluation is performed relative to the root node. A large bag of such units are obtained when all the paths (branches) are fragmented.

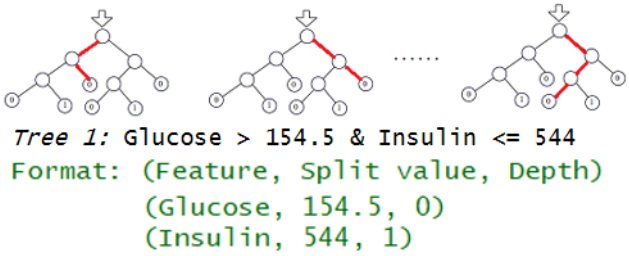

Figure 2: Disintegrating tree path into bag of *(feature,value,depth)* units.

From this large bag, information such as frequency of the features, threshold (cutoff) value used most frequently, the depth where the feature was used most frequently and the various depths that a feature was used in threshold values are extracted. Here, the frequency of a feature may give an indication of its importance. For a prediction of a specific test example, features are ranked according to their frequencies, as shown in Table 1. The importance of features for a given test example are compared based on how deep down the path they are used. Features that are used more frequently closer to the root nodes are considered to be more important. As shown in Figure 3, a bubble chart is used to compare features, where the size of each bubble corresponds to the frequency of a feature at a particular depth.

By disintegrating the extracted paths, we destroy the inherent information about interaction among various features used along the paths. Such information could in the future be preserved and presented in some meaningful way.

Table 1: Features ranked according to their frequency.

	Features	**Type**	**Frequency**
1	Glucose	numerical	918
2	BMI	numerical	550
3	DiabetesPedigreeFunction	numerical	420
4	Age	numerical	313
5	Insulin	numerical	250

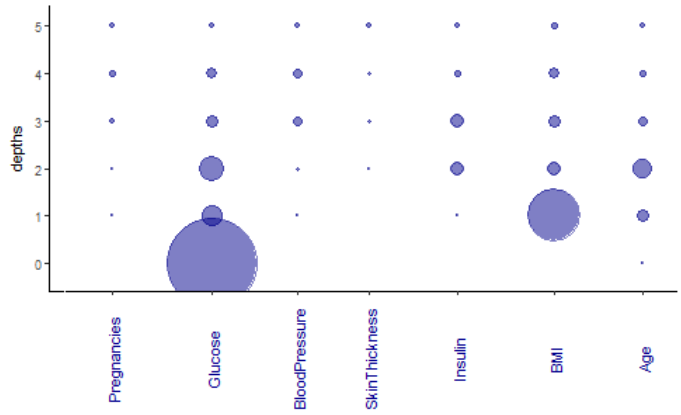

Figure 3: Bubblechart of features with node depth.

3.2 Feature Specific Plots

From the ranked feature list (Table 1) and the bubble chart (Figure 3), users get an overview of the importance of all (relevant) features specific to a single prediction. Further, users are allowed to select some features for deeper exploration. For example, as shown in the top sub-plot of Figure 4, the feature *Glucose* is selected. The plot shows the class-specific distributions of *Glucose* in the training data, with a vertical line representing the value of *Glucose* in the selected test example. On top of it, the density of threshold values (cutoff points) of *Glucose* used in various trees to route the test example is presented. This plot gives an indication of what values of the feature *Glucose* are frequently used as threshold value which are represented as peaks. If the current value of the selected feature for the test example lies around a peak, a small change of the feature value can lead to changes in the prediction as the test example is likely to change paths in many of the base trees. In the tool, stability of the prediction for the test example is examined using a sensitivity plot, as shown in the bottom sub-plot of Figure 4, where the predicted class probabilities are plotted for a range of values of the selected feature, e.g. *Glucose*.

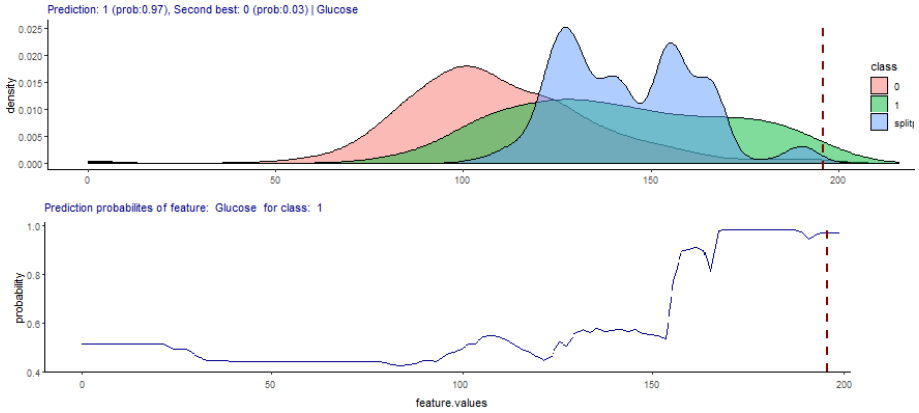

Figure 4: Distribution of cutoff points (top) and feature sensitivity (bottom).

In addition to a density plot, the tool deploys information about the node depth to investigate what threshold values of the features are used more often at what depth in the paths (branches) and present it as contour plot, as shown in Figure 5. The feature cutoff values selected more often closer to the root node (at depth 0) are more significant.

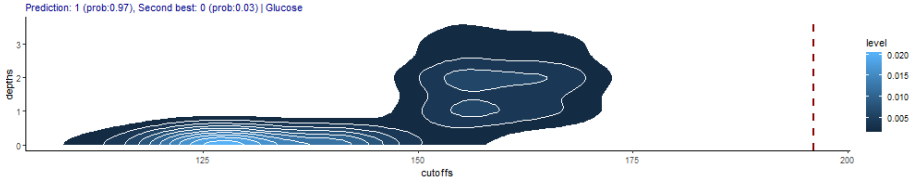

Figure 5: Contour plot showing cutoff points used at specific depths.

3.3 Explanation of Prediction Using Local Surrogate Tree

Although it is normally very difficult to explain the complex decision boundaries of the whole model, it may still be possible to look into a specific region for a simpler explanation. Such a specific region could be a neighbourhood (or locality) around the test example that is being examined. A simple surrogate

model, such as a decision tree, which is easy to interpret can then be used to imitate the behavior of the random forest model within that locality.

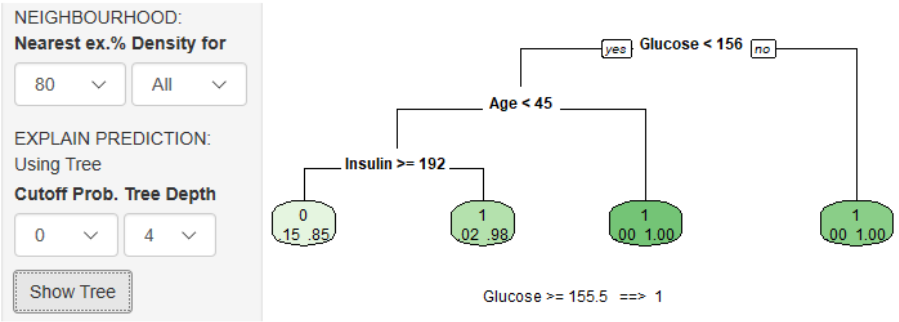

Figure 6: Local explanation using tree.

In the tool, the locality of the test example is determined by using the random forest model itself. In a decision tree, two examples may be considered similar if they fall into the same leaf node. So, all training examples that share leaf node(s) with the test example are selected. Very similar training examples tend to occur multiple times as they share leaf nodes in multiple base trees. The training examples are assigned weights based on their frequency. A local surrogate tree is built using these weighted training examples where the original class labels are replaced with the random forest model predicted class labels. Once the tree is built, an explanation of the prediction for a particular example is obtained from the branch it follows in this newly built tree. As shown in Figure 6, for the patient with details presented in Figure 1, the surrogate tree model predicts the patient as diabetic. It provides reasons for that prediction: The level of *Glucose* is higher than 156. However, occasionally, all instances in the neighborhood are predicted to be of the same class and a meaningful surrogate tree can hence not be grown. This limitation needs to be addressed in the future.

3.4 Recommending Feature Adjustment for the Desired Prediction

In addition to knowing why a model predicts a test example to be of a certain class, in some cases, one would also like to know what changes should be made to the example in order for the model to change its prediction. For example, a patient who is predicted to be diabetic may want to know what changes in her lifestyle would make the model predict her as healthy. Similarly, an applicant whose loan request is denied may also would like to know what changes in his application would grant him the loan. The tool can inform users about changes to be made in order for a particular example to be classified as belonging to a desired class.

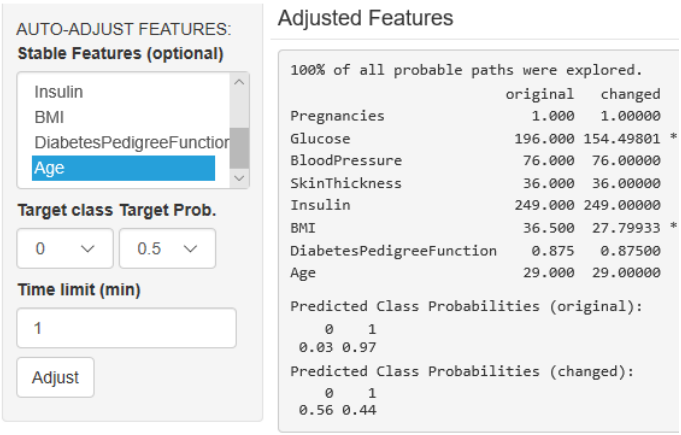

Figure 7: Minimal cost feature adjustment.

The tool implements the actionable feature tweaking algorithm proposed by Tolomei et al. (2017). The algorithm has been slightly adapted to accommodate some additional functionalities including the possibility for users to select features that they want to keep intact. This is useful as some features cannot be manipulated in reality such as *Age*. Users can also specify the lower limit for the predicted class probability of a target class after the suggested changes are performed.

The algorithm first selects all the base trees that do not predict a test example according to a user-preferred class. In these trees, all the paths that lead to leaf nodes that predict a user-preferred class label are extracted. For each path, based on test conditions at each intermediate nodes, the algorithm tries to adjust corresponding feature values such that the test example will take such a path. The adjustment that requires minimal changes, but also changes the overall forest's prediction to a user preferred class is obtained. The complexity of the algorithm in terms of search time for an optimal adjustment depends on the size and number of trees in the forest. The tool therefore allows user to set a time limited search. All possible paths in the trees are then randomly explored and the optimal adjustment found within the allocated time is returned. For example, as shown in Figure 7, the patient was originally predicted to be diabetic with *97%* certainty. Keeping the feature *Age* intact, the tool is allowed to search for optimal adjustment within in 1 minute of search time that would make the model predict it as non-diabetic with at least *50%* certainty. The tool suggests that if the patient lowers his *Glucose* level from 196 to 154 and *BMI* value from 36.5 to 28, the model will predict her as non-diabetic with *56%* certainty.

4 Workflow

The user interface for the tool has a dashboard with left sidebar for user controls and a main panel at the center. All the functionalities mentioned before are arranged as tabs in the main panel.

1. **Example Selection and Prediction Details:**
   A drop-down list at the top-left sidebar allows the user to select a new test example. Details of the selected example are displayed in the main panel under *Summary* tab as shown in Figure 1. This tab also displays details of the trained model: Number of trees, number of examples and class distribution in training data, out-of-bag error rate, number of examples left out for testing purpose, and the area under ROC curve (AUC) value as model's performance metric on the test set. Model predicted class probabilities and the actual class label of the selected test example are shown.

2. **Ranking and Selection of Useful Features:**
   By exploring the paths in the random forest trees that the selected test example follows, features that are used frequently are ranked under the *Feature List* tab as shown in Table 1. For comparing features with respect to both frequency and depth in the path where it was evaluated, a bubblechart as shown in Figure 3 is displayed under *Depth Plot* tab. From the ranked feature list, users can click on the respective rows to select interesting features for further exploration. Selected features with respective slider-input control appear on the sidebar under the *Select Features* title.

3. **Investigating Selected Features:**
   For the selected features, a class-specific distribution of feature values in the training data is displayed under the *Density Plot* tab. Functionality for plotting the distribution of threshold values of the selected features atop the class-specific distribution as shown in Figure 4 is available under the *Splitpoint Density* tab. The feature specific sensitivity of model predicted probability as shown in Figure 4 is available under *Sensitivity* tab. A contour plot for showing what threshold value of the feature is used frequently at what tree depth as shown in Figure 5 can be obtained under the *Contour Plot* tab.

4. **Managing Local Neighborhood and Surrogate Tree:**
   Users can manage the size of the local neighborhood by limiting to a certain percentage of very close neighbors using a drop-down list. The local neighbors are used to grow a surrogate tree as shown in Figure 6. The tree is used for explaining the prediction. Users can limit the maximum depth of the tree and the probability cutoff threshold for assigning class labels as prediction to neighbors. The tree structure is plotted in a new pop-up window. The tool also suggests other test examples very similar to the current one.

5. **Suggesting Optimal Feature Changes:**
   Users can search for the model's suggestion for optimal changes that result in model predicting the test example to a user specified class label. Users can specify the target class, the predicted class probability, the limit for search time and the list of features that should not be changed. The best result within the allocated time is displayed in a new pop-up window

where original and changed feature values are placed side by side along with the original and changed class probabilities as shown in Figure 7.

5 Evaluation

In order to evaluate the usability of the tool, a case study was conducted in one of the large truck manufacturing companies in Sweden, Scania CV AB. The tool was used by a research unit within the company that is managing their flexible maintenance program for a large fleet of trucks. The research unit is interested in deploying data-driven predictive models for predicting imminent risk of failure for important components in trucks. In addition to prediction, they wanted to understand what the model bases its prediction on. This is important in order to understand what caused the failure so that similar failures can be avoided in the future. Tools such as ours could be useful in this regard which is why we decided to conduct our case study at this research unit in Scania.

A random forest model was built on operational data obtained from around six thousand trucks operating in Denmark. The operational data for each truck includes features that described their overall operation. Only trucks built after 2010 were included in the data set. The objective of the trained model was to predict whether a particular component of a truck would survive three months in the future. We decided to choose the brake-pad as the component and the task was to predict whether the brake-pad needs to be changed. In reality, a sensor for measuring the thickness of a brake-pad exists which can be easily used to determine whether the pad needs change. The reason for choosing this component for our case study is to make sure that the user can easily corroborate the information provided by the tool with what is expected and make the evaluation trustworthy.

We decided to conduct post-task interviews separately with five domain experts who all had some background in data science. All the participants had at least a year of experience in the domain of heavy duty trucks operation. Domain expertise and some data science background were the only selection criteria. We had some reflection about how Krause et al. (2016) evaluated their tool which was very similar to ours. Unlike our approach, they had regular meetings with a group of participants over a period of four months, in the future this is something we also would like to do with our tool. The interview session in our case was scheduled for a duration of around two hours each. Each participant

was informed about the purpose of his participation and asked to read and sign the consent form. The tool was introduced as a way of helping them understand random forest prediction. Each participant was then given a simple task to be performed by using the tool under the interviewer's supervision when needed. The task was to select specific test instance, note the model prediction, determine the important features, determine what values of the selected features were frequently used, find explanation for the prediction and finally to find suggested changes to change the current prediction. Later, the participant was allowed to further explore the tool on their own. After the participant had explored the tool, an open interview session was conducted, discussing about the various aspects of the tool such as user friendliness, ease of use, informativeness, usefulness of the plots, trust-worthiness of the explanations, future scope, limitation of the tool and suggestions for improvements. The audio for the interview session was recorded with consent from the participant.

6 Results and Discussion

In general, participants considered that the tool would help them to better understand the prediction of a random forest model. They also mentioned that the tool provides users with an easy to use interactive interface. One participant even mentioned that the tool was not as complex as it sounded, especially after he started exploring and using it on his own. Self exploration of the tool resulted in new revelations in some cases. Regarding user friendliness, a second participant found the input controls on the left sidebar to be overwhelming and confusing. He further suggested an improvement by limiting the display of controls that are relevant only to the selected tab in the main panel while hiding the rest. Regarding the targeted user of the tool, almost all of the participants believed that it would be useful to users with some data science background. Some participants even mentioned that this tool should be used by data scientists together with engineers or domain experts so that data scientists can relay the information obtained from the tool to the engineers who can then relate such knowledge to the field of their expertise. Many participants preferred the density plot of threshold values and sensitivity plot as they valued information about how stable the prediction is relative to the current value of the selected feature. Also the bubble chart and the list of ranked features were appreciated while some functionalities, such as the contour plot had a difficult time justifying their relevance.

Regarding the scope of the tool, one participant mentioned that he would like the tool to be extended for regression and survival forests as well. He also mentioned that it would have been an added advantage if it was possible to investigate multiple instances together. He gave a scenario where some trucks in a fleet encountered a problem and they might have the same root cause. So, being able to group such problematic trucks together for investigating root causes would probably be useful. Regarding the role of such tools in building trust while adopting less transparent machine learning models like random forests, participants believe that such tools certainly help, but also mentioned that it takes time. One participant explained building trust in black-box models in terms of maturity level and he believes that employees should be trained to have some basic knowledge in data science.

Participants judged that the tool could be useful for them in terms of various functionalities it offers to explain the prediction by a random forest model. On the other hand, they also suggested that some issues of user friendliness and aesthetics could be improved. They also appreciated its simple workflow, decent performance and informativeness. From the interviews, it was observed that the tool can help users to further understand random forest predictions and aid in building trust in black-box machine learning models.

7 Conclusion and Future Work

Predictions made by black-box machine learning models such as random forests are often accurate but difficult to understand. Therefore, we designed an interactive visual tool that explores a prediction made by a random forest model and presents the information through plots, charts and tables with the objective of helping the users to understand the reasons for the prediction. Such tools also allow users to get familiarized with machine learning models by providing an easy to use interface thereby eventually helping in building trust in the models. The tool was evaluated at a large truck manufacturing company by interviewing domain experts about their experience of using the tool and its future prospects. From these interviews, it was discovered that they found the tool to be very useful in terms of the wide range of supports it provides to explore the predictions.

However, there are some limitations in the current version of the tool. Its functionality is limited to classification problems. We expect to extend it to handle regression and survival analysis in the future. As of now, we disregard the interaction among features occurring in the same tree path, but this could be a useful information. One possible way of handling it could be to have a matrix of features indicating how often two features co-occur in the same path. The support for handling categorical features is limited. Similarly, while growing a local surrogate tree, if the model predicts a test example to be of a certain class with high certainty, often all the examples in the neighborhood tend to belong to same class. In such cases, the local surrogate tree cannot be grown. We plan to address these issues in the future. Finally, the evaluation of the tool was done as post-task interviews. Before an interview, the tool was introduced which might influence the interview result. Therefore, for a better evaluation, we believe that the user should use it repeatedly over a prolonged period. The current setting only allows for capturing immediate reactions, and an extended study period would allow for observing more in depth experiences.

Acknowledgements This work has been funded by Scania CV AB and the Vinnova program for Strategic Vehicle Research and Innovation (FFI)Transport Efficiency.

References

Biecek P (2018) DALEX: Explainers for Complex Predictive Models. ArXiv e-prints. URL: `https://arxiv.org/abs/1806.08915`. ArXiv:1806.08915.

Breiman L (2001) Random Forests. Machine Learning 45(1):5–32. DOI: 10.1023/A:1010933404324.

Deng H (2019) Interpreting Tree Ensembles With inTrees. International Journal of Data Science and Analytics 7:277–287. DOI: 10.1007/s41060-018-0144-8.

Dua D, Graff C (2017) UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. URL: `http://archive.ics.uci.edu/ml`.

Friedman JH, Popescu BE (2008) Predictive Learning via Rule Ensembles. The Annals of Applied Statistics 2(3):916–954. DOI: 10.1214/07-AOAS148.

Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D (2018) A Survey of Methods for Explaining Black Box Models. ACM Computing Surveys (CSUR) 51(5):93. DOI: 10.1145/3236009.

Krause J, Perer A, Ng K (2016) Interacting with Predictions: Visual Inspection of Black-box Machine Learning Models. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16), Association for Computing Machinery (ACM), New York (USA), pp. 5686–5697. DOI: 10.1145/2858036.2858529.

Laugel T, Lesot M, Marsala C, Renard X, Detyniecki M (2018) Comparison-Based Inverse Classification for Interpretability in Machine Learning. In: Medina J, et al. (eds.), Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations (IPMU2018), Springer, Cham (Switzerland), Communications in Computer and Information Science, Vol. 853, pp. 100–111. DOI: 10.1007/978-3-319-91473-2_9.

Lindgren T (2015) Model Based Sampling: Fitting an Ensemble of Models Into a Single Model. In: Arabnia HR (ed.), Proceedings of 2015 International Conference on Computational Science and Computational Intellingence (CSCI), Institute of Electrical and Electronics Engineers (IEEE), Piscataway (USA), pp. 186–191. DOI: 10.1109/CSCI.2015.27.

Lipton ZC (2018) The Mythos of Model Interpretability. Queue 16(3):30–31–30:57.

Lundberg SM, Lee SI (2017) A Unified Approach to Interpreting Model Predictions. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds.), Advances in Neural Information Processing Systems 30 (NIPS2017). Curran Associates, Inc., New York (USA), pp. 4765–4774.

Molnar C (2019) Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. URL: `https://christophm.github.io/interpretable-ml-book/`.

Ribeiro MT, Singh S, Guestrin C (2016) "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16), Association for Computing Machinery (ACM), New York (USA), pp. 1135–1144. DOI: 10.1145/2939672.2939778.

Tolomei G, Silvestri F, Haines A, Lalmas M (2017) Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery (ACM), New York (USA), pp. 465–474. DOI: 10.1145/3097983.3098039.