# Analysable and Trustworthy Machine Learning Powered By Automated Reasoning

Hadrien Bride[2], Jacob Dong[3], Jin Song Dong[1,2], Zhé Hóu[2]

[1] School of Computing, National University of Singapore, Singapore
[2] Institute for Integrated and Intelligent Systems, Griffith University, Australia
[3] Dependable Intelligence

**Abstract.** The ability to learn from past experience and improve in the future in certain criteria and the ability to reason about the context of problems and extrapolate information from what is known are two important aspects of Artificial Intelligence. There have been various attempts at applying machine learning in automated reasoning but not the other way around. This paper aims to apply automated reasoning in machine learning and demonstrate that we can obtain valuable insights on the model in machine learning with the help of logic, reasoning, and formal methods. A major benefit of the proposed approach is that the user and the machine learner will be able to understand the reason behind the decision-making in machine learning, which is often as important as obtaining the best learning result. This understanding can then be used to reinforce user-specified requirements in the model and to improve the classification and prediction.

## 1 Introduction

Philip Wadler once wrote that "powerful insights arise from linking two fields of study previously thought separate" [7]. This paper, although does not provide a similar correspondence relation between two fields such as logic and computation, aims at finding an interesting application that combines machine learning and automated reasoning. There have been various attempts at applying machine learning in automated reasoning. For instance, the automated reasoning tool Sledgehammer, which is a subsystem of the proof assistant Isabelle/HOL [5], has a module named MaSh [3] which uses machine learning to rank the relevance of known facts in the proof context based on previous successful proofs and select a subset of facts that is estimated most helpful in proving the existing goals.

The other direction, i.e., applying automated reasoning in machine learning, has not seen an application as far as we are aware of. Recently, eXplainable Artificial Intelligence (XAI) has been gaining attention. The prestigious International Joint Conference on Artificial Intelligence (IJCAI) has been running a workshop specialised in this topic. From the automated reasoning community, Bonacina recently envisaged that automated reasoning could be the key to the advances of XAI and machine learning [1]. To this end, she posed several questions and challenges in this direction. Specifically,

> "How can we bridge the gap between the statistical inferences of machine learning and the logical inferences of reasoning, applying the latter to extract, build, or speculate and test, explanations of the former?" [1]

This paper addresses the above challenge by proposing a new concept called Model Analysis and Engineering for machine learning. First, we identify a suitable machine learning technique for logical analysis and reasoning. We then propose to use satisfiability modulo theories (SMT) solvers to perform analysis on classification and prediction models given by machine learning. Finally, we demonstrate a preliminary result using a new machine learning tool called Silas [4].

## 2 Decision Tree Based Machine Learning

To solve a problem one has to choose the right tool. Neural networks and deep learning have been exceptionally successful in analysing "unstructured data", e.g., images, speech, etc. However, there are two main reasons why we do not adopt such learning techniques: First, it is very hard to interpret the models produced by such machine learning techniques and even harder to perform analysis on them. Hence some people refer to such approaches as "black-boxes". Second, some reports [6] show that deep learning is not as successful and it provides lower accuracy than random forests or gradient boosting machines in "traditional" machine learning problems such as fraud detection and credit scoring. These problems are "traditional" in the sense that the data sets are "structured", i.e., they record values for important features that describe the problem.

To fully support the integration of automated reasoning techniques and to build towards a machine learning approach that is analysable and trustworthy, we have taken the following considerations into account:

**Interpretable:** The machine learning approach must generate models that can be understood and analysed.
**Performant:** The learning result should be comparable to the other state-of-the-art machine learning tools in terms of predictive performance.
**Time-efficient:** The approach must be reasonably fast in building models so that we can have the result and perform analysis in a timely manner.
**Space-efficient:** The targeted method should be memory efficient and big-data-friendly.

Given all the above requirements, we conclude that a decision-tree based machine learning approach fits within our application context. Specifically, an implementation of random forest or gradient boosting machine would be our preferred choice.

## 3 Model Analysis and Engineering

*Formulae extraction.* Suppose we are given a classification or prediction model that consists of set of decision trees, we first need to extract logical formulae from the trees. A decision tree in this context is a data structure in which every non-leaf node is associated with a logical formula that splits the data entries into two subsets. Assuming that $A, B, \cdots$ are the classes to be classified or predicted in a data set. Each leaf node contains a subset of data entries labelled by the classes. An algorithm such as majority voting is then needed to obtain the final decision. For instance, if a leaf node has 5 entries labelled with class $A$, 2 entries with $B$, and 1 entries with $C$, then we say this leaf node's decision is class $A$.

There are multiple ways to obtain logical formulae for model analysis. One can collect all the formulae from the root of a tree to a leaf node with decision *A*, and the *conjunction* of these formulae gives the reason why the subset of data entries in the leaf node are classified/predicted as *A*. We refer to the conjunction as the *branch formula*. There could be multiple leaf nodes whose decisions are all class *A*. The *disjunction* of all branch formulae which lead to class *A* represents the overall decision-making of the tree with respect to class *A*. We refer to this disjunction as the *decision formula* for class *A*. The decision formula for a class can then be used in the analysis to check inconsistencies and extract the core reason behind the decision-making. Given a set of decision trees, we can extract the decision formula for class *A* on each tree and perform analysis on the conjunction of multiple decision formulae.

*Model anlaysis and engineering.* Once the logical formulae are extracted, we can perform a number of analysis on the formulae using automated reasoning techniques.

**Maximum Satisfiable Subset (MSS):** To obtain the MSS, we assign a weight to each sub-formula, and try to maximise the accumulated weight in the MAX-SMT optimisation problem. This can be achieved in certain SMT solvers such as Z3 and MathSAT 5. The weight assigned to each formula can be optimised to reflect the predictive performance of the decision node or the decision tree. For instance, a decision node with more information gain may have more weight, and a decision tree with higher predictive accuracy may have more weight. The resulting MSS can give an indication of the core attributes and the range of the attributes that lead to the decision making of the classification and prediction model. Note that a similar analysis is to extract *maximal satisfiable subsets*, which is computationally cheaper, but we prefer the maximum subset because it may give more insight about the decision-making.

**Minimal Unsatisfiable Core (MUC):** Solvers such as Z3 provide a straightforward way to compute the MUC of a set of formulae. We can use this functionality to obtain the inconsistencies in the model and use this information to fine-turn the model by trimming the decision tree. This can form a recursive procedure in which we repeatedly find the MUCs in a decision tree and trim the tree accordingly until the tree becomes a consistent model. Another application is to use the MUC in boosting. A boosting algorithm usually consists of iterative learning steps in which weak classifiers are introduced and added to compensate the shortcomings of existing weak learners. The MUC can be effectively used as the shortcomings of multiple learners, because it represents the disagreements of multiple decision trees. We can then build weak classifiers around the MUC to boost the model.

**Model Verification:** In certain applications, the user may specify some requirements that a decision making procedure must satisfy. For instance, if machine learning is applied to classify whether a node in a network cluster is secure, our method may produce a logical condition for deciding network security. If the user has other security requirements that must be satisfied, we can use SMT solving and model checking to verify that the requirements hold in the learned condition. If this is not true, then the MUC analysis can pinpoint the reason why the learned condition fails and we can use the MUC to tweak the model by inserting decision nodes that reinforce the user requirements and obtain machine learning results that conform the user's specifications.

## 4 Discussion

The model analysis and engineering component for machine learning can provide several benefits to users at different levels: (1) The analysis can pinpoint the reason behind the classification and prediction. This will help the user (e.g., decision maker) understand what the key attributes are and how they lead to the result. Therefore the user can use the analytical information provided by this approach to make the final decision based on their discretion. Moreover, the analytical information can help transform the machine learning algorithm into a transparent process in which every decision can be inspected and verified. (2) The analysis can also provide the reason why some models have good performance while others have bad performance. Data scientists can use this information to improve the learning process and perform hyperparameter tuning. (3) Machine learners can use the MUC to fine-tune the models and improve classification and prediction results. They can also use the MSC to build new and consistent models that potentially have better results. (4) Model verification helps obtain machine learning results that conform with user-specified requirements. This is vital in providing a machine learning technique that can be trusted.

We have implemented the approach in this paper as a module for the machine learning tool Silas [4]. As an example on a diabetes data set [2], we are able to analyse a random forest model and obtain a set of "core reasons" behind each class (negative/ positive diabetes). By comparing the core reasons, we derive that $30 \leq \textit{age} \leq 34$ and $0 < \textit{number of times pregnant} \leq 2$ are among the key indicators for classifying positive diabetes, whereas $21 \leq \textit{age} \leq 22$ and $\textit{number of times pregnant} \leq 0$ strongly indicate negative diabetes. On the other hand, *2-hour serum insulin* is not a strong indicator for either classes, which implies that data scientists can perform certain feature engineering on the data set to improve the results.

## References

1. Maria Paola Bonacina. Automated reasoning for explainable artificial intelligence. In *AR-CADE Workshop (in association with CADE-26), Gothenburg, Sweden*, 2017.
2. Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.
3. Daniel Kühlwein, Jasmin Christian Blanchette, Cezary Kaliszyk, and Josef Urban. Mash: machine learning for sledgehammer. In *International Conference on Interactive Theorem Proving*, pages 35–50. Springer, 2013.
4. Dependable Intelligence Pty Ltd. Silas. https://depintel.com/silas/, 2018.
5. Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283. Springer Science & Business Media, 2002.
6. Szilard Pafka. A minimal benchmark for scalability, speed and accuracy of commonly used open source implementations of the top machine learning algorithms for binary classification. https://github.com/szilard/benchm-ml, 2018.
7. Philip Wadler. Propositions as types. *Communications of the ACM*, 58(12):75–84, 2015.