Interpretable Machine Learning Tools: A Survey

Namita Agarwal, Saikat Das
Department of Computer Science
The University of Memphis
Memphis, TN, USA
{nfnu, sdas1}@memphis.edu

***Abstract*— In recent years machine learning (ML) systems have been deployed extensively in various domains. But most ML-based frameworks lack transparency. To believe in ML models, an individual needs to understand the reasons behind the ML predictions. In this paper, we provide a survey of open-source software tools that help explore and understand the behavior of the ML models. Also, these tools include a variety of interpretable machine learning methods that assist people with understanding the connection between input and output variables through interpretation, validate the decision of a predictive model to enable lucidity, accountability, and fairness in the algorithmic decision-making policies. Furthermore, we provide the state-of-the-art of interpretable machine learning (IML) tools, along with a comparison and a brief discussion of the implementation of those IML tools in various programming languages.**

Keywords—machine learning, interpretable machine learning, open-source tools, interpretable machine learning tools survey.

I. INTRODUCTION

Machine learning (ML) models are being used extensively and have countless applications in almost every segment of human activity, such as health care, marketing, finance, logistics, banking, agriculture, and so on. ML models have become sophisticated and complex as a large amount of data is involved to solve problems. This complexity denotes the number of parameters of the ML model that impact the final prediction. The larger the computation complexity of these models, the more incomprehensible they become to humans. Hence, these black-box models need to be interpretable. In real-world scenarios, individuals need to recognize and understand how the data input affect the outcomes (predictions) of ML models. There are various tools designed to help people understand the intricate working of complex ML models. These open-source programming tools provide excellent executions of white-box, black-box, global, and local explanation strategies for ML models.

Model interpretation can be defined as the response generated by the decision-making function to explain the connection between input and output variables of the ML model. In many domains, interpretability cannot be fortified either because of legal requirements or because it leads to unbiased decisions, yet it is significant for users as it improves their faith in the system. Interpretability has multiple benefits, such as:

a) extricating interpretable patterns from pre-trained ML models,
b) identifying the reasons behind poor predictions,
c) increasing faith in model predictions,
d) detecting bias in ML models, and
e) creating safety catch to protect against overfitted models.

This paper aims to provide a review of the current state-of-the-art of interpretable machine learning (IML) tools, IML techniques, and related open-source software resources.

This article is composed as follows: In Section II, various interpretable techniques are explained. Section III describes open-source IML tools. In Section IV, a discussion on IML tools implemented in different programming languages (Python, R, and Java) is presented, and Section V concludes the paper.

II. INTERPRETABLE MACHINE LEARNING

The IML model can be defined as an extent to which a human can comprehend the decisions made by ML models in their decision-making process [1]. The importance of IML is due to the following characteristics:

- **Bias**: IML guarantee's predictions of ML models are not biased, and they do not discriminate against specific classes/groups.
- **Privacy**: Data confidentiality is ensured through model interpretation.
- **Robustness:** It ensures that predictions of the ML model have remained consistent and have no significant changes when few changes are made in data.
- **Causality**: It ensures only causal relationships are picked up.
- **Belief/Faith**: Individuals can trust the interpretable system easily compared to the black box system [2].

A. Model Interpretation Techniques

Interpretable techniques provide human-friendly explanations and reasoning behind the model outcomes/predictions. Several interpretable methods have been reported in the literature:

- Local Interpretable Model-Agnostic Explanations (LIME): A model agnostic method that generates sparse explanations using the significant local variables [3].
- Shapley Additive Explanations (SHAP): A coherent approach to describe the performance of any ML model. It is additive and locally accurate [4].
- Anchor: It explains the local behavior of ML models through rules of high-degree precision known as anchors [5].

978-1-7281-2547-3/20/$31.00 ©2020 IEEE

- Partial Dependence Plot (PDP): PDP shows the minimal impact that a couple of features have on the forecasted output of the ML model. Generally, it's a model agnostic method [6].
- Individual Conditional Expectation (ICE): A local and model agnostic method that focuses on specific instances. Instead of plotting an average, ICE displays one line per instance [7].
- Accumulated Local Effects (ALE): It represents the global behavior and describes how features have an impact on the predicted outcome of ML models [8].
- Permutation Feature Importance: It defines how features of the model contribute to the predictions in a concise way and is a model agnostic method [9].
- Global Surrogate: In this approach, an ML model (black -box) is trained with an interpretable ML model to approximate the ML model (black-box) prediction.
- Contrastive Explanation Method (CEM): A local method that produces PP and PN for a specific instance by indicating the presence of minimal features for any ML model [10].
- Layer-wise Relevance Propagation (LRP): A model specific method that identifies significant pixels by running a backward pass in the neural network [11].
- Explainable Neural Network (XNN) Explanation: They are structured neural networks designed to learn global interpretable features and rely on model-specific mechanisms [12].
- Monotonic Gradient Boosting Machines (GBMs): They enforce a straightforward relationship between input and output variables for certain ML models [13].
- Generalized Additive Models (GAMs): A generalized linear model and model-specific method in which the predictor relies linearly upon predictor variables and smooth functions of predictor variables [14].
- Supersparse Linear Integer Models (SLIMs): They are simple, model-specific, and require users to perform mathematical operations such as addition, subtraction, or multiplication of values with corresponding few input variables to generate correct and precise predictions [15].
- Decision Tree Surrogates: It is a fundamental ML model that provides a complete understanding of a complex ML model and gives insights about the internal working of it [16].
- Global Variable Importance: It explains the impact of the input variable's global contribution to model predictions and identifies the prominent variables. This method is both model specific and model agnostic [6].

III. Interpretable Machine Learning Tools

Numerous open-source tools that implement the interpretable techniques mentioned in the previous section are described here:

A. Alibi

In 2019, Seldon Technologies released the Alibi package [17], and it is a Python bundle intended to explain the ML models' predictions, guage assurance of decisions made by ML models, and at the end support the extensive capacity to analyze the performance of models concerning concept drift and algorithm bias. The focus of this library is to the give top-notch implementation of explanation methods (such as local, global) for ML models . It contains various algorithms that provide local explanations for ML model predictions, and they are summarized in Table I.

TABLE I. The Summary of Interpretation Methods Implemented in Alibi

Interpretability Techniques	Model	Scope	Classification	Regression
ALE	Model Agnostic	Global	√	√
Anchors	Model Agnostic	Local	√	×
CEM	Model Agnostic	Local	√	×
Kernel SHAP	Model Agnostic	Local Global	√	√
Tree SHAP	Model Agnostic	Local Global	√	√

Instance–specific scores such as trust score and linearity measure metric is provided by the tool to calculate model confidence for making a particular decision. The trust score is the ratio between the distance to the nearest class different from the predicted class and the distance to the predicted class. Linearity measure delivers a score quantifying how linear the model is around a test instance. The linearity score measures the model linearity around a test instance by feeding the model linear superpositions of inputs and comparing the output with the linear combination of output from predictions on single inputs.

B. Skater

It is a unified framework to help users build an IML system for real-world applications. Skater is designed to demystify both global and local learned structures of the ML model and is implemented in Python. It has adopted paradigms of object-oriented and functional programming as deemed necessary to provide scalability and concurrency while keeping code brevity in mind. This library is in the beta phase, and active development is taking place [18]. The interpretable techniques supported by Skater are mentioned in Table II.

TABLE II. The Summary of Interpretation Methods Implemented In Skater

Interpretability Techniques	Model	Scope	Classification	Regression
Feature Importance	Model Agnostic	Global	√	√
Partial Dependence Plots	Model Agnostic	Global	√	√
LIME	Model Agnostic	Local	√	√
LRP	Model Specific	Local	√	√
Tree Surrogates	Both	Local Global	√	√

Using the Skater package, users can identify the behavior of variable interaction, enhance domain knowledge, and evaluate the behavior of the ML model on a single instance of dataset or a complete data set. The drawback of the Skater tool is that it relies on different packages for implementing its interpretable methods. For example, to execute the LIME method, it relies on different packages such as NumPy and scikit-learn.

C. InterpretML

H. Nori et al. [19] developed InterpretML, an open-source Python package that incorporates interpretability techniques under one roof for ML. It is easy to use, provides flexibility, and interpretable glass-box models are trained with this kit to describe ML models. InterpretML has interactive dashboards that support data filtering and cohort creation capabilities to help users observe and understand the model performance for different variations of the dataset. Besides, with varied representations, local and global explanations are displayed on the dashboard, and this tool also has debugging capabilities. InterpretML mainly focuses on the interpretation techniques where it helps users understand the reasoning behind the model's predictions, as shown in Table III.

TABLE III. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN INTERPRETML

Interpretability Techniques	Model	Scope	Classification	Regression
LIME	Model Agnostic	Local	√	√
Partial Dependence	Model Agnostic	Global	√	√
Kernel SHAP	Model Agnostic	Local Global	√	√
Tree SHAP	Model Specific	Local Global	√	√

D. ELI5

ELI5 is a Python bundle which helps users to debug ML classifiers and explain their forecasts. Many inbuilt functions exist in ELI5 that is easy to use [20]. Characteristics of this tool like text highlighting and feature filtering can be reused. It supports model agnostic methods, but the LIME method only supports text classifiers. Table IV shows the summary of interpretable methods implemented in ELI5.

TABLE IV. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN ELI5

Interpretability Techniques	Model	Scope	Classification	Regression
LIME	Model Agnostic	Local	√	×
Permutation Feature Importance	Model Agnostic	Global	√	×

E. EthicalML-XAI

EthicalML-XAI library is constructed based on the eight standards for Responsible Machine Learning to explain AI at its core. It contains different instruments for the examination and assessment of data and models. More generally, the XAI library planned to utilize 3-phase machine learning, which includes analysis of data as phase one, evaluation of the ML model as phase two, and monitoring the production as phase three. This Python library is currently in an early stage of development [21]. The summary of the interpretation method supported by EthicalML – XAI is shown in Table VI.

TABLE V. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN ETHICALML -XAI

Interpretability Technique	Model	Scope	Classification	Regression
Permutation Feature Importance	Model Agnostic	Global	√	×

F. Anchor

This package is implemented in Python, R, and Java languages. It explains individual predictions of any ML classifier by finding high precision rules known as anchors. These anchors predict locally where the changes in the remaining feature values of an instance are not considered [5]. Anchors also address a shortcoming of local explanation method LIME, which proxy the local behavior of the model linearly.

There are some limitations of anchor tool, such as potentially conflicting anchors and overly specific anchors. In potentially conflicting anchors, two or more anchors with different predictions may lead to the same test instance. So, choosing an anchor becomes ambiguous. On the other hand, in overly specific anchors, predictions that are near to the boundary of the ML model's decision function or predictions of rare classes may require specific enough conditions, and thus their anchors may be complex and provide low coverage. The summary of the interpretable method is given in Table VI.

TABLE VI. THE SUMMARY OF INTERPRETATION METHOD IMPLEMENTED IN ANCHOR

Interpretability Technique	Model	Scope	Classification	Regression
Anchor	Model Agnostic	Local	√	×

G. Slim-Python

Slim-Python is a package to learn customized scoring systems for decision-making problems. SLIM is designed to learn the most accurate scoring system for a given dataset and set of constraints [15]. Table VII presents the summary of the interpretable technique executed in Slim-Python.

TABLE VII. THE SUMMARY OF INTERPRETATION METHOD IMPLEMENTED IN SLIM-PYTHON

Interpretability Technique	Model	Scope	Classification	Regression
SLIMs	Model Specific	Global	√	√

H. SHAP

Lundberg et al. [4] introduced Python bundle SHAP, and this library is based on game theory to explain the predictions of any ML model. SHAP uses the concept of Shapley values to score/rank features of the ML model. Shapley values consider all the possible predictions, for instance, using all the combinations of input variables. Because of this comprehensive approach, SHAP can guarantee properties like consistency and local accuracy, and their results are accurate and reliable. But,

SHAP value calculation is time-consuming and expensive as it checks for all the possible combinations. Also, SHAP has a wrapper class implemented in R known as Shapper. SHAP offers various Interpretability ML techniques, as shown in Table VIII.

TABLE VIII. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN SHAP

Interpretability Techniques	Model	Scope	Classification	Regression
Deep Explainer	Model Specific	Global	√	√
Kernel SHAP	Model Agnostic	Local Global	√	√
Tree SHAP	Model Specific	Local	√	√

I. LightGBM

LightGBM is a gradient boosting framework implemented in Python and R languages. It uses tree-based learning algorithms to support different interpretable methods and produces efficient and accurate results [22]. It handles massive data, requires low memory, and provides optimization with the implementation of parallel and GPU learning. Interpretation methods executed by LightGBM are shown in Table IX.

TABLE IX. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN LIGHTGBM

Interpretability Techniques	Model	Scope	Classification	Regression
Global Variable Importance	Both	Global	√	√
Tree SHAP	Model Specific	Local	√	√

J. eXtreme Gradient Boosting (XGBoost)

XGBoost is a distributed platform that executes ML algorithms under the gradient boosting framework. It is a lightweight algorithm and supports flexibility [23]. Also, XGBoost provides a parallel boosting mechanism that helps users solve many real-world decision-making problems quickly and accurately. This library is implemented in various programming languages such as Python, R, Julia, Java, C, C++, etc. It supports interpretable techniques, and they are presented in Table X.

TABLE X. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN XGBOOST

Interpretability Techniques	Model	Scope	Classification	Regression
Global Variable Importance	Both	Global	√	√
Tree SHAP	Model Specific	Local	√	√
Monotonic GBMS	Model Specific	Global	√	√

K. H2O.ai

H2O.ai is a scalable and distributed framework for ML. The H2O.library uses predictive modeling to give insights to users about the data quickly, and it is supported by different programming languages such as Java, Scala, Python, R, JSON, and Flow Notebook [24]. Interpretable techniques in H2O.ai are described for both regular and time-series experiments. These techniques do not support image type data. Also, it has an interactive dashboard where users can run their experiments and view/observe explanations on the interpretation model tab. This UI gives users the option to view local, global, and cluster-specific explanations. Therefore, it implements various interpretable algorithms in Table XI.

TABLE XI. THE SUMMARY OF INTERPRETABLE METHODS IMPLEMENTED IN H2O.AI

Interpretability Techniques	Model	Scope	Classification	Regression
Monotonic GBMS	Model Specific	Global	√	√
PDP	Model Agnostic	Global	√	√
Global Variable Importance	Both	Global	√	√
Shapley Values	Model Agnostic	Local Global	√	√
LIME	Model Agnostic	Local	√	√
ICE	Model Agnostic	Local	√	√
Decision Tree surrogates	Model Agnostic	Global	√	√
Permutation Feature Importance	Both	Global	√	√

L. LIME

LIME is a Python and R library that supports explaining individual predictions for text classifiers, images, and tabular data. LIME is fast and can explain any ML algorithm, with multiple classes [3]. The output of LIME (local interpretability) helps individuals to determine which feature changes have a considerable impact on the prediction. The implementation of LIME is not very simple as kernel settings for every application should be changed to get accurate explanations for models' output. Also, linear models are used to explain the local behavior of the ML model, so this approach might not be advantageous to explain the complex behavior of the ML model. It lacks stability and does not provide consistent explanations. Table XII represents the summary of the interpretable method implemented in this tool.

TABLE XII. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN LIME

Interpretability Technique	Model	Scope	Classification	Regression
LIME	Model Agnostic	Local	√	√

M. DALEX

DALEX package radiates any ML model and helps to explore and explain its complex behavior. Numerous methods present in the DALEX package helps users understand the connection between the features and predicted outcome of the ML model. Implemented interpretable methods assist individuals to test the model at a single instance and dataset level [25]. This tool executed in R language provides useful approaches to compare results across multiple ML models. It

does not support multinomial classes. The model agnostic methods supported by this library are listed in Table XIII.

TABLE XIII. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN DALEX

Interpretability Techniques	Model	Scope	Classification	Regression
ALE	Model Agnostic	Global	√	√
PDP	Model Agnostic	Global	√	√
Variable Importance	Model Agnostic	Global	√	√
LIME	Model Agnostic	Local	√	√
SHAP	Model Agnostic	Local	√	√

N. Statistical Machine Intelligence and Learning Engine (SMILE)

SMILE is a notable library for a wide range of ML tasks. It has efficient memory usage and covers every aspect of ML. It is fast and provides NLP, data visualization, and statistical techniques. SMILE is self-contained and includes only the standard Java library [27]. The interpretability technique implemented by this library is shown in Table XIV.

TABLE XIV. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN SMILE

Interpretability Technique	Model	Scope	Classification	Regression
Tree SHAP	Model Agnostic	Local Global	√	√

O. iml

iml developed by Cristopher et al. [26], integrates all approaches in one place, utilizes a similar syntactic structure, and provides consistent features and outputs. A human-friendly and class-based method is provided by this tool to develop interpretable ML models. These interpretable models provide explanations to users, which are easy to comprehend. This R library executes various interpretability techniques, as shown in Table XV.

TABLE XV. THE SUMMARY OF INTERPRETATION METHODS IMPLEMENTED IN IML

Interpretability Techniques	Model	Scope	Classification	Regression
LIME	Model Agnostic	Local	√	√
Shapley value	Model Agnostic	Local	√	√
PDP	Model Agnostic	Global	√	√
ALE	Model Agnostic	Local	√	√
Variable Importance	Model Agnostic	Global	√	√
Tree Surrogates	Model Agnostic	Global	√	√
Global Surrogates	Model Agnostic	Global	√	√
ICE	Model Agnostic	Local	√	√
Permutation Feature Importance	Model Agnostic	Global	√	√

P. AI Explainability 360 (AIX 360)

It is a Python package that supports the interpretability and explainability of datasets and provides a flexible, familiar programming interface. This tool kit intended for high-stake applications of ML models contains several algorithms and provides explanations for ML models in different ways [32].

In data explanation, users need to understand the characteristics and features of data before any ML model is applied. Data explanations were provided by implementing DIP-VAE and ProtoDash algorithms. On the other hand, in model explanation, there are several ways to explain the ML model, such as local versus global, directly interpretable versus post-hoc explanation, etc. For global directly interpretable models, the algorithms implemented are Boolean Decision Rules and Generalized Linear Rule models. For the Global post hoc explanation, the ProfWeight technique is implemented. For Local directly interpretable models, the Teaching AI to Explain its Decisions (TED) technique is implemented. Local post hoc explanation supports CEM, LIME, SHAP, and ProtoDash methods. Besides, it also supports explainability metrics known as faithfulness and monotonicity.

Q. Glmnet

Glmnet package is fast, implemented in R programming language, and fits a generalized linear model through a probability function known as penalized maximum likelihood. Also, the algorithm fits other regression models, such as poisson, linear, multinomial, and logistic models [31]. This package is also available in other languages such as Python and MATLAB.

R. Variable Importance Plots (vip)

vip package helps users to visualize feature impact both locally and globally by constructing Variable Importance Plots. It offers a consistent interface to compute variable importance for various kinds of ML models. This library provides both model-specific and model agnostic variable importance measures, and it is implemented in the R language [29].

S. PDP

PDP is used for constructing partial dependence plots and individual conditional expectation curves [28]. The PDP implemented in R language is very adaptable in delivering various sorts of PDP, and it has numerous features, for example, a choice to show advancement bars, the choice to alleviate the dangers related to extrapolation, and alternatives to develop PDPs in parallel. The two significant functions implemented by PDP are: a) partial function which is used for computing partial dependence functions and individual conditional expectations from various fitted model objects, and b) plot partial function that builds lattice-based PDP and ICE curves.

T. PDPBox

The objective of this library is to determine the effect of input features on model prediction for any ML algorithm using PDPs [30]. It supports multi-class and two-variable interaction PDPs. Similar to ICEBox but executed on the Python programming

platform. Also, it is a solution for handling complex mutual dependency among features.

U. ICEBox

This repository presents the R code for Individual Conditional Expectation (ICE) plots similar to Partial Dependence Plots. ICE curves show the functional relationship between input and output variables of any supervised ML algorithm for individual observations, and users can visualize those graphs and understand the reasoning behind the predicted ML output [33].

IV. DISCUSSION

An overview of various interpretability techniques along with their associated tools is described in the above sections to help understand the reasoning behind the ML models' predictions and outcomes. Methods such as PDP and ICE curves are easy to implement and intuitive when compared to ALE plots. PDP plots and ICE curves do not perform well when features are strongly correlated. LIME executes faster when compared to SHAP as it checks all the combinations. SHAP is the only consistent additive feature attribution method as it guarantees properties like consistency and local accuracy.

Interpretable techniques are associated with tools so that interpretation of ML models can be executed and explained efficiently. A few tools are used only for constructing plots, and they are vip, PDP, PDPBox, and ICEBox. Some libraries are in the developmental stages, and not every method in the framework can be extended to all domains. For example, the LRP method in the Skater package can only be applied to images but not to any other data type. Based on model agnostic methods implemented in various frameworks, iml is the only framework that offers all interpretable approaches in one place and simplifies the study and interpretation of ML models. Another tool AIX360 is distinct from all other IML tools, as it provides explainability metrics.

A. Programming Language support

In recent years, several open-source programming tools have been developed to describe ML models and to provide a high-level implementation of them and explainability. IML tools are classified as language-independent or language-dependent based on their availability on different programming platforms. XGBoost is considered as language-independent as it is available on all the programming platforms, which gives a choice for users to implement this library in their preferred programming language. Some tools are language-dependent as they only can be executed on specific platforms, and this restricts the flexibility to users. For example, iml is implemented in R. In Table XVI, we summarize various interpretable techniques along with their associated tools implemented in different languages.

TABLE XVI. THE SUMMARY OF IML METHODS ALONG WITH THEIR ASSOCIATED TOOLS IMPLEMENTED IN PYTHON, R AND JAVA LANGUAGES

Interpretability Techniques	**IML Tools - Python Implementation**	**IML Tools – R Implementation**	**IML Tools - Java Implementation**
LIME	LIME, Skater, ELI5, InterpretML, AIX360	LIME, Dalex, iml, h2o.ai	h2o.ai
SHAP	SHAP, InterpretML, Alibi, h2o.ai, AIX360	Dalex, XGboost, lightGBM, shapper, h2o.ai	h2o.ai, Smile, XGBoost
ICE	PyICE Box, InterpretML, h2o.ai	iml, ICEBox, PDP	h2o.ai
ALE	Alibi	iml, Dalex	×
LRP	Skater	×	×
PDP	PDP, PDPBox, Skater, InterpretML	PDP, Dalex, iml, h2o.ai	h2o.ai
Permutation Feature Importance	SHAP, EthicalMl -xai, Alibi, h2o.ai	Dalex, iml, vip, h2o.ai	h2o.ai
Decision Tree Surrogate	Skater, InterpretML, h2o.ai	iml, h2o.ai	h2o.ai
Monotonic GBMS	h2o.ai, InterpretML, XGBoost	h2o.ai, XGBoost	h2o.ai, XGBoost
GAMS	h20.ai	GAM, h2o.ai, glmnet	h2o.ai
Global Variable Importance	h2o.ai, Skater, XGBoost, InterpretML, lightGBM, SHAP	Dalex, iml, ShapleyR, XGBoost, lightGBM, vip, shapper	h2o.ai, XGBoost
SLIMS	Slim - Python	×	×
XNN Explanations	Skater, SHAP	×	×
CEM	Alibi, AIX 360	×	×
Anchors	Anchor, Alibi	Anchors	Anchor
Global Surrogate	□	iml	×

V. CONCLUSION

We have presented an overview of interpretable approaches, along with their related software resources to describe enigmatic and unclear ML models. These tools assist in constructing interpretable models. This survey led to the conclusion that, given the various tools being developed for IML, not every kit offers a single interface, and every IML tool has its limitations. We have also summarized the IML methods and showcased how these tools were implemented in various programming languages.

Considering this tools survey as baseline, we plan to use various tools to enrich our existing supervised [34] and unsupervised [35] ensemble ML framework. In addition, we plan to expand our research focusing on explaining detected DDoS attacks, selecting critical features compared with EnFS method [36], and monitoring [37] secure network (both offline and cloud [38]) using various IML tools.

REFERENCES

[1] Molnar, Christoph. "A guide for making black box models explainable." URL: https://christophm. github. io/interpretable-ml-book (2018).

[2] Hall, Patrick. An introduction to machine learning interpretability. O'Reilly Media, Incorporated, 2019.

[3] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "" Why should I trust you?" Explaining the predictions of any classifier." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.

[4] Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions." Advances in neural information processing systems. 2017.

[5] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Anchors: High-Precision Model-Agnostic Explanations." AAAI. Vol. 18. 2018.

[6] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

[7] Goldstein, Alex, et al. "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation." Journal of Computational and Graphical Statistics 24.1 (2015): 44-65.

[8] Apley, Daniel W., and Jingyu Zhu. "Visualizing the effects of predictor variables in black box supervised learning models." arXiv preprint arXiv:1612.08468 (2016).

[9] Fisher, A., C. Rudin, and F. Dominici. "Model class reliance: Variable importance measures for any machine learning model class, from the "Rashomon" perspective. arXiv 2018." arXiv preprint arXiv:1801.01489.

[10] Dhurandhar, Amit, et al. "Explanations based on the missing: Towards contrastive explanations with pertinent negatives." Advances in Neural Information Processing Systems. 2018.

[11] Bach, Sebastian, et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation." PloS one 10.7 (2015): e0130140.

[12] Vaughan, Joel, et al. "Explainable neural networks based on additive index models." arXiv preprint arXiv:1806.01933 (2018).

[13] XGBoost, "Monotonic Constraints" https://xgboost.readthedocs.io/en/latest/tutorials/monotonic.html. [Online; accessed 20-September-2020].

[14] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. No. 10. New York: Springer series in statistics, 2001.

[15] Ustun, Berk, and Cynthia Rudin. "Supersparse linear integer models for optimized medical scoring systems." Machine Learning 102.3 (2016): 349-391.

[16] Craven, Mark, and Jude W. Shavlik. "Extracting tree-structured representations of trained networks." Advances in neural information processing systems. 1996.

[17] Klaise, Janis, et al. "Alibi: Algorithms for monitoring and explaining machine learning models." URL https://github. com/SeldonIO/alibi (2020).

[18] Github, "Oracle Skater" https://github.com/oracle/Skater. [Online; accessed 20-September-2020].

[19] Nori, Harsha, et al. "Interpretml: A unified framework for machine learning interpretability." arXiv preprint arXiv:1909.09223 (2019).

[20] Github, "TeamHG-Memex, Eli5" https://github.com/TeamHG-Memex/eli5. [Online; accessed 20-September-2020].

[21] Github, "xai-eXplainable AI" https://ethicalml.github.io/xai/index.html. [Online; accessed 20-September-2020].

[22] Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." Advances in neural information processing systems. 2017.

[23] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.

[24] Cook, Darren. Practical machine learning with H2O: powerful, scalable techniques for deep learning and AI. " O'Reilly Media, Inc.", 2016.

[25] Biecek, Przemysław. "DALEX: explainers for complex predictive models in R." The Journal of Machine Learning Research 19.1 (2018): 3245-3249.

[26] Molnar, Christoph, Giuseppe Casalicchio, and Bernd Bischl. "iml: An R package for interpretable machine learning." Journal of Open Source Software 3.26 (2018): 786.

[27] Github, "Smile-Statistical Machine Intelligence and Learning Engine" https://haifengl.github.io/quickstart.html. [Online; accessed 20-September-2020].

[28] Greenwell, Brandon M. "pdp: An R Package for Constructing Partial Dependence Plots." R J. 9.1 (2017): 421.

[29] Github, "koalaverse,vip" https://github.com/koalaverse/vip. [Online; accessed 20-September-2020].

[30] Github, "SourceCat,PDPbox" https://github.com/SauceCat/PDPbox. [Online; accessed 20-September-2020].

[31] Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent." Journal of statistical software 33.1 (2010): 1.

[32] Arya, Vijay, et al. "One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques." arXiv preprint arXiv:1909.03012 (2019).

[33] Goldstein, Alex, et al. "Package 'ICEbox'." (2017).

[34] Das, Saikat, et al. "DDoS intrusion detection through machine learning ensemble." 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, 2019.

[35] Das, Saikat, Deepak Venugopal, and Sajjan Shiva. "A Holistic Approach for Detecting DDoS Attacks by Using Ensemble Unsupervised Machine Learning." Future of Information and Communication Conference. Springer, Cham, 2020.

[36] Das, Saikat, et al. "Empirical Evaluation of the Ensemble Framework for Feature Selection in DDoS Attack." 2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom). IEEE, 2020.

[37] Das, Saikat, and Sajjan Shiva. "CoRuM: collaborative runtime monitor framework for application security." 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). IEEE, 2018.

[38] Das, Saikat, Ahmed M. Mahfouz, and Sajjan Shiva. "A Stealth Migration Approach to Moving Target Defense in Cloud Computing." Proceedings of the Future Technologies Conference. Springer, Cham, 2019.