Article

Responsible Machine Learning

with Interpretable Models, Post-hoc Explanation, and Disparate Impact Testing

Navdeep Gill [1,†], Patrick Hall [1,3,†,*], Kim Montgomery [1,†], and Nicholas Schmidt [2,†]

[1] H2O.ai
[2] BLDS, LLC
[3] George Washington University
* Correspondence: phall@h2o.ai; nschmidt@bldsllc.com
† All authors contributed equally to this work.

Version December 2, 2019 submitted to Information

Abstract: This text outlines a viable approach for training and evaluating machine learning (ML) systems for high-stakes, human-centered, or regulated applications using common Python programming tools. The accuracy and intrinsic interpretability of two types of constrained models, monotonic gradient boosting machines (MGBM) and explainable neural networks (XNN), a deep learning architecture well-suited for structured data, are assessed on simulated data with known feature importance and discrimination characteristics and on realistic, publicly available mortgage data. For maximum transparency and the potential generation of personalized adverse action notices, the constrained models are analyzed using post-hoc explanation techniques including plots of partial dependence (PD) and individual conditional expectation (ICE) and global and local gradient-based or Shapley feature importance. The constrained model predictions are also tested for disparate impact (DI) and other types of discrimination using straightforward group fairness measures. By combining innovations in interpretable models, post-hoc explanation, and discrimination testing with accessible software tools, this text aims to provide a template workflow for important ML applications that require high accuracy and interpretability and minimal discrimination.

Keywords: Machine Learning; Neural Network; Gradient Boosting Machine; Interpretable; Explanation; Fairness; Disparate Impact; Python

---

0. Introduction

ML models can be inaccurate and unappealable black-boxes, even with the application of newer post-hoc explanation techniques [1].[1] ML models can perpetuate and exacerbate discrimination [2], [3], [4], [5]. ML models can be hacked, resulting in manipulated model outcomes or the exposure of proprietary intellectual property or sensitive training data [6], [7], [8]. The authors make no claim that the interdependent issues of opaqueness, discrimination, or security vulnerabilities in ML have been solved, even as singular entities, much less as complex intersectional phenomena, e.g. the fairwashing or scaffolding of biased models with ML explanations or the privacy harms of ML explanations [9], [10], [11]. However, this text is an attempt to address the technological aspects of these vexing problems with interpretable models, post-hoc explanation, and DI and discrimination testing implemented in widely available, free, and open source Python tools. Section 1 describes methods and materials, including simulated and collected training datasets, interpretable and constrained model architectures, post-hoc explanations used to create an *appealable* decision-making framework, tests for

[1] See: "When a Computer Program Keeps You in Jail".

DI and other discrimination, and public and open source software resources. In Section 2, interpretable and constrained modeling results are compared to less interpretable and unconstrained models and post-hoc explanation and discrimination testing results are also presented for interpretable models. Section 3 then discusses some nuances of the presented modeling, explanation, and discrimination testing methods and results. Finally, Section 4 closes this text with a brief outline of proposed additional steps to increase human trust and understanding in ML and also touches on the authors' future work.

1. Materials and Methods

To provide a sense of accuracy differences, this text compares the performance of more interpretable constrained ML models and less interpretable unconstrained ML models on simulated data and on collected mortgage data. The simulated data, based on the well-known Friedman datasets and with known feature importance and discrimination characteristics, is used to gauge the validity of interpretable modeling, post-hoc explanation, and discrimination testing techniques [12]. The mortgage data is sourced from the Home Mortgage Disclosure Act (HMDA) database.[2] Post-hoc explanation and discrimination testing techniques are applied to constrained, interpretable models trained on the mortgage data to provide a more realistic template workflow for future users of these methods and tools. Unconstrained gradient boosting machines (GBMs) (e.g. [13], [14]) and artificial neural networks (ANNs) (e.g. [15], [16], [17], [18]) are deemed uninterpretable black-box ML models herein.

Because black-box ML models can be difficult to understand, trust, and appeal, even after the application of post-hoc explanation techniques, explanation analysis and discrimination testing are applied only to the constrained interpretable ML models [1], [9], [10]. Here, MGBMs[3] and XNNs ([19] [20]) will serve as those more interpretable models for subsequent explanatory and discrimination analysis. Presented explanation techniques include PD, ICE, and Shapley values [14], [21], [22], [23]. PD, ICE, and Shapley values provide direct, global, and local summaries and descriptions of constrained models without resorting to the use of intermediary and approximate surrogate models. Discussed discrimination testing methods and metrics include ... . All discussed materials and methods are implemented with open source Python packages and code and are available in the software resources associated with this text. Detailed descriptions of notation, training data, ML models, post-hoc explanation techniques, discrimination testing methods, and software resources are organized in Section 1 as follows:

Notation: spaces, datasets, & models – §1.1
Training data: simulated data & collected home mortgage data – §1.2 and §1.3
ML models: constrained, interpretable MGBM & XNN models – §1.4 and §1.5
Post-hoc explanation techniques: PD, ICE, & Shapley values – §1.6 and §1.7
Discrimination testing methods: – §1.8
Software resources: GitHub repository for this text; utilized & useful Python packages – §1.9

1.1. Notation

To facilitate descriptions of data and modeling, explanatory, and discrimination testing techniques, notation for input and output spaces, datasets, and models is defined.

1.1.1. Spaces

- Input features come from the set $\mathcal{X}$ contained in a P-dimensional input space, $\mathcal{X} \subset \mathbb{R}^P$. An arbitrary, potentially unobserved, or future instance of $\mathcal{X}$ is denoted $\mathbf{x}$, $\mathbf{x} \in \mathcal{X}$.

---
[2] See: Mortgage data (HMDA).
[3] As implemented in `XGBoost` or `h2o`.

- Labels corresponding to instances of $\mathcal{X}$ come from the set $\mathcal{Y}$.
- Learned output responses of models are contained in the set $\hat{\mathcal{Y}}$.

1.1.2. Datasets

- The input dataset $\mathbf{X}$ is composed of observed instances of the set $\mathcal{X}$ with a corresponding dataset of labels $\mathbf{Y}$, observed instances of the set $\mathcal{Y}$.
- Each i-th observation of $\mathbf{X}$ is denoted as $\mathbf{x}^{(i)} = [x_0^{(i)}, x_1^{(i)}, \dots, x_{P-1}^{(i)}]$, with corresponding i-th labels in $\mathbf{Y}$, $\mathbf{y}^{(i)}$, and corresponding predictions in $\hat{\mathbf{Y}}$, $\hat{\mathbf{y}}^{(i)}$.
- $\mathbf{X}$ and $\mathbf{Y}$ consist of N tuples of observations: $[(\mathbf{x}^{(0)}, \mathbf{y}^{(0)}), (\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(N-1)}, \mathbf{y}^{(N-1)})]$.
- Each j-th input column vector of $\mathbf{X}$ is denoted as $X_j = [x_j^{(0)}, x_j^{(1)}, \dots, x_j^{(N-1)}]^T$.

1.1.3. Models

- A type of ML model g, selected from a hypothesis set $\mathcal{H}$, is trained to represent an unknown signal-generating function f observed as $\mathbf{X}$ with labels $\mathbf{Y}$ using a training algorithm $\mathcal{A}$: $\mathbf{X}, \mathbf{Y} \xrightarrow{\mathcal{A}} g$, such that $g \approx f$.
- g generates learned output responses on the input dataset $g(\mathbf{X}) = \hat{\mathbf{Y}}$, and on the general input space $g(\mathcal{X}) = \hat{\mathcal{Y}}$.
- The model to be explained and tested for discrimination testing is denoted as g.

1.2. Simulated Data

1.3. Mortgage Data

The training data contains 33 total features and 144,000 rows, each representing a unique loan, and a fold identifier to ensure consistent 5-fold cross-validation accuracy and error measurements across different types of models. Consumer finance and loan descriptors are used for training. Demographic features are not used in model training. The mortgage test data contains 36,000 loans.

1.4. Monotonically Constrained Gradient Boosting Machine

MGBMs constrain typical GBM training to consider only tree splits that obey user-defined positive and negative monotonicity constraints. The MGBM remains an additive combination of B trees trained by gradient boosting, T_b, but each tree learns a set of splitting rules that respect monotonicity constraints, Θ_b^{mono}.

$$g^{\text{mono}}(\mathbf{x}) = \sum_{b=1}^{B} T_b\left(\mathbf{x}; \Theta_b^{\text{mono}}\right) \tag{1}$$

As in unconstrained GBM, Θ_b^{mono} is selected in a greedy, additive fashion by minimizing a regularized loss function that considers known target labels, $\mathbf{y}$, the predictions of all subsequently trained trees in the MGBM, $g_{b-1}^{\text{mono}}(\mathbf{X})$, and a regularization term that penalizes complexity in the current tree, $\Omega(T_b)$. For the b-th iteration, the loss function, $\mathcal{L}_b$, can generally be defined as:

$$\mathcal{L}_b = \sum_{i=0}^{N-1} l(y^{(i)}, g_{b-1}^{\text{mono}}(\mathbf{x}^{(i)}), T_b(\mathbf{x}^{(i)}; \Theta_b^{\text{mono}})) + \Omega(T_b) \tag{2}$$

In addition to $\mathcal{L}_b$, g^{mono} training is characterized by additional splitting rules and constraints on tree node weights. Each binary splitting rule, $\theta_{b,j,k} \in \Theta_b$, is associated with a feature, X_j, is the k-th split associated with X_j in T_b, and results in left and right child nodes with a numeric weights, $\{w_{b,j,k,L}, w_{b,j,kR}\}$. For terminal nodes, $\{w_{b,j,k,L}, w_{b,j,kR}\}$ can be direct numeric components of some g^{mono} prediction. For two values of some feature X_j, $x_j^\alpha \leq x_j^\beta$, where the prediction for each value results in $T_b(x_j^\alpha; \Theta_b) = w_\alpha$ and $T_b(x_j^\beta; \Theta_b) = w_\beta$, Θ_b is restricted to be positive monotonic w.r.t. X_j by the following rules and constraints.

1. For the first and highest split in T_b involving X_j, any $\theta_{b,j,0}$ resulting in the left child weight being greater than the right child weight, $T(x_j;\theta_{b,j,0}) = \{w_{b,j,0,L}, w_{j,0,R}\}$ where $w_{b,j,0,L} > w_{b,j,0,R}$, is not considered.
2. For any subsequent left child node involving X_j, any $\theta_{b,j,k\geq 1}$ resulting in $T(x_j;\theta_{b,j,k\geq 1}) = \{w_{b,j,k\geq 1,L}, w_{b,j,k\geq 1,R}\}$ where $w_{b,j,k\geq 1,L} > w_{b,j,k\geq 1,R}$, is not considered.
3. Moreover, for any subsequent left child node involving X_j, $T(x_j;\theta_{b,j,k\geq 1}) = \{w_{b,j,k\geq 1,L}, w_{b,j,k\geq 1,R}\}$, $\{w_{b,j,k\geq 1,L}, w_{b,j,k\geq 1,R}\}$ are bound by the parent set of node weights, $\{w_{b,j,k-1,L}, w_{b,j,k-1,R}\}$, such that $\{w_{b,j,k\geq 1,L}, w_{b,j,k\geq 1,R}\} \leq \frac{w_{b,j,k-1,L}+w_{b,j,k-1,R}}{2}$.
4. (1) and (2) are also applied to all right child nodes, except that for right child nodes $\{w_{b,j,k\geq 1,L}, w_{b,j,k\geq 1,R}\} \geq \frac{w_{b,j,k-1,L}+w_{b,j,k-1,R}}{2}$.

Note that for any one X_j and $T_b \in g^{\text{mono}}$ left subtrees will alway produce lower predictions than right subtrees, and that any $g^{\text{mono}}(\mathbf{x})$ is an addition of each T_b output, with the application of a monotonic logit or softmax link function for classification problems. Moreover, each tree's root node corresponds to some constant node weight that by definition obeys monotonicity constraints, $T(x_j^{\alpha};\theta_{b,0}) = T(x_j^{\beta};\theta_{b,j,0}) = w_{b,0}$. Together these additional splitting rules and node weight constraints ensure that $g^{\text{mono}}(x_j^{\alpha}) \leq g^{\text{mono}}(x_j^{\beta}) \ \forall \ x_j^{\alpha} \leq x_j^{\beta} \in X_j$. For a negative monotonic constraint, i.e. $g^{\text{mono}}(x_j^{\alpha}) \geq g^{\text{mono}}(x_j^{\beta}) \ \forall \ x_j^{\alpha} \leq x_j^{\beta} \in X_j$, left and right splitting rules and node weight constraints are switched.

Herein, two g^{mono} models are trained. One on the simulated data and one on the mortgage data. In both cases, positive and negative monotonic constraints for each X_j are selected using the sign of the Pearson correlation between each X_j and y, random grid search is used to determine other hyperparameters, and five-fold cross validation and test partitions are used for model assessment. For exact parameterization of the MGBM models, see the software resources referenced in Subsection 1.9.

1.5. Explainable Neural Network

XNNs are an alternative formulation of additive index models in which the ridge functions are neural networks [19]. XNNs also bare a strong resemblance to generalized additive models (GAMs) and so-called explainable boosting machines (EBMs or GA^{2}M), i.e. GAMs which consider main effects and a small number of 2-way interactions and incorporate boosting in their training [14], [24]. Hence, XNNs enable users to tailor interpretable neural network architectures to a given prediction problem and to visualize model behavior by plotting ridge functions. XNNs are composed of a global bias term, μ_0, K individually specified neural networks, n_k with scale parameters γ_k, and the inputs to each n_k are themselves a linear combination of modeling inputs, $\sum_{j=0}^{J} \beta_{k,j} x_j$.

$$g^{\text{XNN}}(\mathbf{x}) = \mu_0 + \sum_{k=1}^{K} \gamma_k n_k (\sum_{j=1}^{J} \beta_{k,j} x_j) \tag{3}$$

g^{XNN} is comprised of 3 meta-layers:

1. The first and deepest meta-layer, composed of K linear $\sum_j \beta_{k,j} x_j$ hidden units, is known as the *projection layer* and is fully connected to each input feature, X_j.
2. The second meta-layer contains K hidden and separate n_k ridge functions, or *subnetworks*. Each n_k is a neural network, which can be parameterized to suite a given modeling task. To facilitate direct visualization, the input to each subnetwork is the 1-dimensional output of its associated projection layer hidden unit, $\sum_j \beta_{k,j} x_j$.
3. The output meta-layer, called the *combination layer*, is another linear unit comprised of a global bias term, μ_0, and the K weighted 1-dimensional outputs of each subnetwork, $\gamma_k n_k(\sum_j \beta_{k,j} x_j)$. Again, subnetwork output is restricted to 1-dimension for visualization purposes.

Here, each g^{XNN} is trained by mini-batch stochastic gradient descent (SGD) on the simulated data and mortgage data. Each g^{XNN} is assessed in five training folds and in a test data partition. L_1 regularization is applied to both the projection and combination layers to induce a sparse and interpretable model, where each n_k subnetwork and corresponding combination layer γ_k are ideally associated with an important X_j or combination thereof. The g^{XNN} models are highly sensitive to weight initialization and manual feature selection is informed by Shapley feature importance from the g^{mono}. For more details regarding g^{XNN} training, see the software resources in Subsection 1.9.

1.6. Partial Dependence and Individual Conditional Expectation

PD plots are a widely-used method for describing and plotting the average predictions of a complex model g across some partition of data $\mathbf{X}$ for some interesting input feature X_j [14]. ICE plots are a newer method that describes the local behavior of g for a single instance $\mathbf{x} \in \mathcal{X}$ [21]. PD and ICE can be overlaid in the same plot to compensate for known weaknesses of PD (e.g. inaccuracy in the presence of strong interactions and correlations [21], [25]), to identify interactions modeled by g, and to create a holistic global and local portrait of the predictions of a complex model for some X_j [21].

Following Friedman *et al.* [14] a single feature $X_j \in \mathbf{X}$ and its complement set $\mathbf{X}_{(-j)} \in \mathbf{X}$ (where $X_j \cup \mathbf{X}_{(-j)} = \mathbf{X}$) is considered. $\text{PD}(X_j, g)$ for a given feature X_j is estimated as the average output of the learned function $g(\mathbf{X})$ when all the observations of X_j are set to a constant $x \in \mathcal{X}$ and $\mathbf{X}_{(-j)}$ is left unchanged. $\text{ICE}(x_j, \mathbf{x}, g)$ for a given instance $\mathbf{x}$ and feature x_j is estimated as the output of $g(\mathbf{x})$ when x_j is set to a constant $x \in \mathcal{X}$ and all other features $\mathbf{x} \in \mathbf{X}_{(-j)}$ are left untouched. PD and ICE curves are usually plotted over some set of constants $x \in \mathcal{X}$, as displayed in Section 2.

1.7. Shapley Values

Shapley explanations are a class of additive, locally accurate feature contribution measures with long-standing theoretical support [22], [26]. Shapley explanations are the only possible locally accurate and globally consistent feature contribution values, meaning that Shapley explanation values for input features always sum to $g(\mathbf{x})$ for some $\mathbf{x} \in \mathcal{X}$ and that Shapley explanation values can never decrease in magnitude for some x_j when g is changed such that x_j truly makes a stronger contribution to $g(\mathbf{x})$ [22], [23]. Unfortunately, many non-consistent explanation methods can result in drastically different global and local feature importance values across different models trained on the same data or even for refreshing the same model with augmented training data [27]. Consistency and accuracy guarantees are perhaps a factor in the growing momentum behind Shapley values as a candidate technique for generating consumer-specific and personalized adverse action notices for automated ML-based decisions in highly-regulated settings such as credit lending [28].

For some instance $\mathbf{x} \in \mathcal{X}$, Shapley explanations take the form:

$$g(\mathbf{x}) = \phi_0 + \sum_{j=0}^{j=\mathcal{P}-1} \phi_j \mathbf{z}_j \tag{4}$$

In Equation 4, $\mathbf{z} \in \{0,1\}^{\mathcal{P}}$ is a binary representation of $\mathbf{x}$ where 0 indicates missingness. Each ϕ_j is the local feature contribution value associated with x_j and ϕ_0 is the average of $g(\mathbf{X})$. Local, per-instance explanations using Shapley values tend to involve presenting a ranking of x_j by ϕ_j values or simply presenting a set of the x_j associated with the largest ϕ_j for some $\mathbf{x}$. Global explanations are typically the absolute mean of the ϕ_j associated with a given X_j across the observations in some partition of data $\mathbf{X}$. Each ϕ_j is a weighted combination of model scores on the $g_x(\mathbf{x})$ with x_j, $g_x(S \cup \{j\})$, and the model scores without x_j, $g_x(S)$, for every subset of features S not including j, $S \subseteq \mathcal{P} \setminus \{j\}$.

$$\phi_j = \sum_{S \subseteq \mathcal{P} \setminus \{j\}} \frac{|S|!(\mathcal{P} - |S| - 1)!}{\mathcal{P}!} [g_x(S \cup \{j\}) - g_x(S)] \tag{5}$$

Where g_x incorporates the mapping between $\mathbf{x}$ and the binary vector $\mathbf{z}$.

Shapley values can be estimated in different ways, many of which are intractable for datasets with large $\mathcal{P}$. Tree SHAP is a specific implementation of Shapley explanations that relies on traversing internal decision tree structures to efficiently estimate the contribution of each x_j for some $g(\mathbf{x})$ [23]. Tree SHAP has been implemented efficiently in popular gradient boosting libraries such as `h2o`, `LightGBM`, and `XGBoost`, and Tree SHAP is used to calculate accurate and consistent global and local feature importance for MGBM models in this text.

1.8. Discrimination Metrics and Test Description

1.9. Software Resources

Python code to reproduce the results presented in this text are available at: https://github.com/h2oai/article-information-2019. The authors primarily make use of the datatable, h2o, matplotlib, pandas, scikit-learn, seaborn, and shap packages for data manipulation, modeling, and reporting results.

2. Results

2.1. Simulated Data Results

2.1.1. Unconstrained Model Fit Assessment

2.1.2. Constrained Model Fit Assessment

2.1.3. Interpretability and Post-hoc Explanation Results

2.1.4. Discrimination Testing Results

2.2. Mortgage Data Results

2.2.1. Unconstrained Model Fit Assessment

2.2.2. Constrained Model Fit Assessment

2.2.3. Interpretability and Post-hoc Explanation Results

2.2.4. Discrimination Testing Results

3. Discussion

3.1. The Burgeoning Ecosystem of Interpretable Models

3.2. Impact of Discrimination Testing on Model Use and Adoption

3.3. Viable Discrimination Remediation Approaches

3.4. Intersectionality of Interpretability, Fairness, and Security in ML

4. Conclusion

Author Contributions: N.G., GBM and MGBM assessment and results; P.H., primary author; K.M., ANN and XNN implementation, assessment, and results; N.S., secondary author, data simulation and collection, and discrimination testing.

Funding: This research received no external funding.

Acknowledgments: Wen Phan for work in formalizing our notation.

Conflicts of Interest:

Abbreviations

The following abbreviations are used in this text: ANN – artificial neural network, DI – disparate impact, GBM – gradient boosting machine, ICE – individual conditional expectation, MGBM – monotonic gradient boosting machine, ML – machine learning, PD – partial dependence, SGD – stochastic gradient descent, US – United States, XNN – explainable neural network.

References

1. Rudin, C. Please Stop Explaining Black Box Models for High Stakes Decisions. *arXiv preprint arXiv:1811.10154* **2018**. URL: https://arxiv.org/pdf/1811.10154.pdf.
2. Barocas, S.; Hardt, M.; Narayanan, A. *Fairness and Machine Learning*; fairmlbook.org, 2019. URL: http://www.fairmlbook.org.
3. Feldman, M.; Friedler, S.A.; Moeller, J.; Scheidegger, C.; Venkatasubramanian, S. Certifying and Removing Disparate Impact. Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015, pp. 259–268. URL: https://arxiv.org/pdf/1412.3756.pdf.
4. Dwork, C.; Hardt, M.; Pitassi, T.; Reingold, O.; Zemel, R. Fairness Through Awareness. Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. ACM, 2012, pp. 214–226. URL: https://arxiv.org/pdf/1104.3913.pdf.
5. Buolamwini, J.; Gebru, T. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. Conference on Fairness, Accountability and Transparency, 2018, pp. 77–91. URL: http://proceedings.mlr.press/v81/buolamwini18a/buolamwini18a.pdf.
6. Barreno, M.; Nelson, B.; Joseph, A.D.; Tygar, J. The Security of Machine Learning. *Machine Learning* **2010**, *81*, 121–148. URL: http://people.ischool.berkeley.edu/~tygar/papers/SML/sec_mach_learn_journal.pdf.
7. Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M.K.; Ristenpart, T. Stealing Machine Learning Models via Prediction APIs. 25th {USENIX} Security Symposium ({USENIX} Security 16), 2016, pp. 601–618. URL: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_tramer.pdf.
8. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017, pp. 3–18. URL: https://arxiv.org/pdf/1610.05820.pdf.
9. Aïvodji, U.; Arai, H.; Fortineau, O.; Gambs, S.; Hara, S.; Tapp, A. Fairwashing: the Risk of Rationalization. *arXiv preprint arXiv:1901.09749* **2019**. URL: https://arxiv.org/pdf/1901.09749.pdf.
10. Slack, D.; Hilgard, S.; Jia, E.; Singh, S.; Lakkaraju, H. How Can We Fool LIME and SHAP? Adversarial Attacks on Post-hoc Explanation Methods. *arXiv preprint arXiv:1911.02508* **2019**. URL: https://arxiv.org/pdf/1911.02508.pdf.
11. Shokri, R.; Strobel, M.; Zick, Y. Privacy Risks of Explaining Machine Learning Models. *arXiv preprint arXiv:1907.00164* **2019**. URL:https://arxiv.org/pdf/1907.00164.pdf.
12. Friedman, J.H.; others. Multivariate Adaptive Regression Splines. *The annals of statistics* **1991**, *19*, 1–67. URL: https://projecteuclid.org/download/pdf_1/euclid.aos/1176347963.
13. Friedman, J.H. Greedy Function Approximation: a Gradient Boosting Machine. *Annals of statistics* **2001**, pp. 1189–1232. URL: https://projecteuclid.org/download/pdf_1/euclid.aos/1013203451.
14. Friedman, J.; Hastie, T.; Tibshirani, R. ***The Elements of Statistical Learning***; Springer: New York, 2001. URL: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf.
15. Recht, B.; Re, C.; Wright, S.; Niu, F. HOGWILD: A Lock-free Approach to Parallelizing Stochastic Gradient Descent. Advances in neural information processing systems, 2011, pp. 693–701. URL: https://papers.nips.cc/paper/4390-hogwild-a-lock-free-approach-to-parallelizing-stochastic-gradient-descent.pdf.
16. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. *arXiv preprint arXiv:1207.0580* **2012**. URL: https://arxiv.org/pdf/1207.0580.pdf.
17. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the Importance of Initialization and Momentum in Deep Learning. International Conference on Machine Learning, 2013, pp. 1139–1147. URL: http://proceedings.mlr.press/v28/sutskever13.pdf.
18. Zeiler, M.D. ADADELTA: an Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701* **2012**. URL: https://arxiv.org/pdf/1212.5701.pdf.
19. Vaughan, J.; Sudjianto, A.; Brahimi, E.; Chen, J.; Nair, V.N. Explainable Neural Networks Based on Additive Index Models. *arXiv preprint arXiv:1806.01933* **2018**. URL: https://arxiv.org/pdf/1806.01933.pdf.
20. Yang, Z.; Zhang, A.; Sudjianto, A. Enhancing Explainability of Neural Networks Through Architecture Constraints. *arXiv preprint arXiv:1901.03838* **2019**. URL: https://arxiv.org/pdf/1901.03838.pdf.

21. Goldstein, A.; Kapelner, A.; Bleich, J.; Pitkin, E. Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation. *Journal of Computational and Graphical Statistics* **2015**, *24*. URL: https://arxiv.org/pdf/1309.6392.pdf.
22. Lundberg, S.M.; Lee, S.I. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*; Guyon, I.; Luxburg, U.V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; Garnett, R., Eds.; Curran Associates, Inc., 2017; pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.
23. Lundberg, S.M.; Erion, G.G.; Lee, S.I. Consistent Individualized Feature Attribution for Tree Ensembles. In *Proceedings of the 2017 ICML Workshop on Human Interpretability in Machine Learning (WHI 2017)*; Kim, B.; Malioutov, D.M.; Varshney, K.R.; Weller, A., Eds.; ICML WHI 2017, 2017; pp. 15–21. URL: https://openreview.net/pdf?id=ByTKSo-m-.
24. Lou, Y.; Caruana, R.; Gehrke, J.; Hooker, G. Accurate Intelligible Models with Pairwise Interactions. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2013, pp. 623–631. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.352.7682&rep=rep1&type=pdf.
25. Apley, D.W. Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *arXiv preprint arXiv:1612.08468* **2016**. URL: https://arxiv.org/pdf/1612.08468.pdf.
26. Shapley, L.S.; Roth, A.E.; others. *The Shapley value: Essays in Honor of Lloyd S. Shapley*; Cambridge University Press, 1988. URL: http://www.library.fa.ru/files/Roth2.pdf.
27. Molnar, C. ***Interpretable Machine Learning***; christophm.github.io, 2018. URL: https://christophm.github.io/interpretable-ml-book/.
28. Bracke, P.; Datta, A.; Jung, C.; Sen, S. Machine Learning Explainability in Finance: an Application to Default Risk Analysis **2019**. URL: https://www.bankofengland.co.uk/-/media/boe/files/working-paper/2019/machine-learning-explainability-in-finance-an-application-to-default-risk-analysis.pdf.