

Explaining Supervised Learning Models: A Preliminary Study on Binary Classifiers

As the potential reach of artificial intelligence grows, it could provide predictions and suggestions but may leave an incomplete explanation of “reasoning.” This study explores general information that could be provided to system end users or novice analysts.

By Xiaomei Wang , Ann M. Bisantz, Matthew L. Bolton, Lora Cavuoto , & Varun Chandola

FEATURE AT A GLANCE:

The reach of artificial intelligence continues to grow, particularly with the expansion of machine learning techniques that capitalize on increased computing power. Such systems could have tremendous benefits by providing predictions and suggestions. However, they are limited by the fact that they offer incomplete explanations of their predictions to human decision makers. The objective of this work was to summarize general information that could help users make judgments about whether a system is trustworthy and whether the system’s training “makes sense.” A preliminary study was summarized to show the importance of iterative design and testing for visualizing explanations.

KEYWORDS:

black box, binary classification, machine learning, visualization, iterative design, decision making, trust

Machine learning (ML) is everywhere. People are eager to test its power by applying it to any data set they can find and for any topic they can think of. Because of the nature of ML, even system developers are incapable of tracking how many algorithms learn and generate results. Such methods are thus ultimately treated as black boxes.

Black box systems create many important questions for human–machine interaction: How do we tell people more about the system? How do we enable humans to make judgments about whether the black box or its prediction is trustworthy? How do we let humans compare different ML models? How do we visualize and communicate such information?

To develop trust in an intelligent system, systems users need information about both the quality of the results and how the results are derived (Swartout, 1983). There are two general approaches to interpretable ML (Lipton, 2016): developing new algorithms that are more transparent and providing post hoc interpretations to explain ML. Reviews on the latter method have been done on related work in social and behavioral science (Miller, 2019; Miller et al., 2017; Mittelstadt et al., 2019; Mueller et al., 2019). Most of the previous research has focused on explaining the mechanism of ML algorithms. In this article, we identified information that does not require an understanding of the mechanisms of the algorithms but still could help users understand the quality of supervised learning algorithms. We grouped information into three categories that were based on system input (what data were fed into the model), training (how and how well the model was trained), and output (the

prediction result and prediction quality). We provided an overview of an iterative design study that examined data visualization that could support users’ understanding of ML outputs (specifically algorithm accuracy and confidence).

EXPLAINING WHAT DATA WERE FED INTO THE MODEL

When a system provides a suggestion, the end user may want to know what the suggestion is based on. Relevant questions include the following:

- What features (input variables) are included to generate the predictions?
- What are the distributions of these features?
- Are the features correlated?
- What is the importance of a certain feature?
- What time span of data were used?

These questions are usually addressed by the system developers but are not always shared with the end users.

An important topic on this front is feature selection. Feature selection is the process of selecting a subset of relevant features to use in the final model training. In the context of ML, it is mostly used to reduce dimensionality, computation cost, redundancy, storage requirements, and training/utilization time (Guyon & Elisseeff, 2003). It is useful to share what features are included, how they were selected, and why they should be included or excluded with the end users.

Another related post hoc technique called black box auditing (Adler et al., 2016) can be used to decide the extent to which a specific

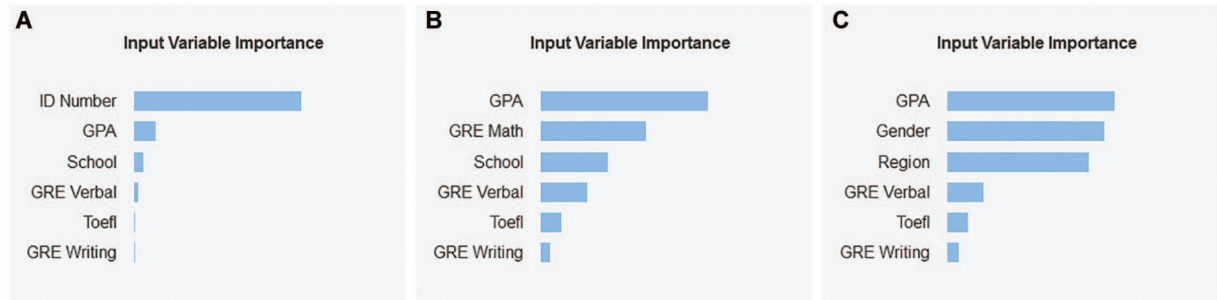


Figure 1. Examples of variable importance ranking for graduate admission decision making.

feature contributes to the accuracy (percentage of correct predictions) of a trained model. To quantify the direct effect of a feature, we can replace the feature by random noise and see how much the model accuracy drops. The resulting drop in accuracy can be considered the variable's importance.

The benefits of providing measures of importance for the input variables includes letting decision makers verify if the input makes sense as well as checking for the existence of bias. Consider graduate admission decision making as an example (Figure 1), a process of determining admission to a graduate program based on various parameters (e.g., test scores, prior academic performance). A poor example of using ML could involve an algorithm (illustrated in Figure 1A), where ID number is accidentally added as an input variable and provides the largest contribution to the model. Compared with the algorithm from Figure 1A, a different algorithm, shown in Figure 1B, which uses GPA (grade point average), GRE (Graduate Record Examination) math, and undergraduate school as its three most important variables, might be a more reasonable model. Furthermore, if a model shows variable ranking as in Figure 1C, one might want to check why gender and region became so important in the model and if that is because of bias in the past decisions.

One limitation of showing such information is that when the decision maker is not the system developer, he or she cannot choose to use a different set of features. Changing features would require retraining the models and could lead to dramatic changes in results. Another limitation is that if some dimension reduction techniques are used (e.g., principal components analysis), the features may no longer retain their original meanings. Thus, the combined values themselves need representations before being listed as shown in Figure 1.

Another way to test the system input is to do controlled experiments. One can accomplish this by feeding controlled input (either simulated or real examples) to the model and comparing the different outputs. The experiment would then perform a hypothesis test on the cause and effect relations inside the box. Datta et al. (2015) took this black box analysis perspective to explore Google ad settings. They treated the entire advertising ecosystem as the black box and simulated agents with different user profiles, ad settings, page views, and ad views as controlled inputs. They then tested how the

changes in the inputs affected the ads that were recommended to the agents. One example of their findings was that the gender of the agent significantly affected job-related ads. Specifically, setting the gender to female resulted in fewer instances of ads related to high paying jobs than when the gender setting was male. This study highlights how showing and comparing different outputs given controlled input (either simulated or real data) can affect how an end user will interpret how input variables affect outcome. If conditions do not allow such interactive experiments with the system, we could still provide some typical or simple cases to the end users to allow them to verify whether presented relationships are consistent with their own knowledge.

EXPLAINING HOW (WELL) THE MODEL IS TRAINED

Through a review of the existing techniques, we identified four aspects of model training that could be understood without needing to explain the mechanism of the model. They are stability, robustness/generalizability, parameter sensitivity, and training and inference times. When there are different models with similar performance, these measures can serve as additional criteria to help users to choose a more suitable model to use.

Stability

Stability is a measure of how an ML algorithm is perturbed by small changes to its inputs. A stable learning algorithm is one for which the prediction does not change much when the training data are modified slightly (Bousquet & Elisseeff, 2000).

One way to check stability is to plot a learning curve. If we randomly take samples from the data to be the training set, calculate the average prediction accuracy (or any other performance metric), and plot the results with sample size on the x -axis and the metric on the y -axis, the accuracy is expected to increase as the sample size increases. Plotting the learning curve shows the model stability by showing how much the model performance changes with changing training data sample size.

An article on sample size determination (Figueroa et al., 2012) indicated that there is a generic shape for learning curves: At first, the performance increases rapidly, then there

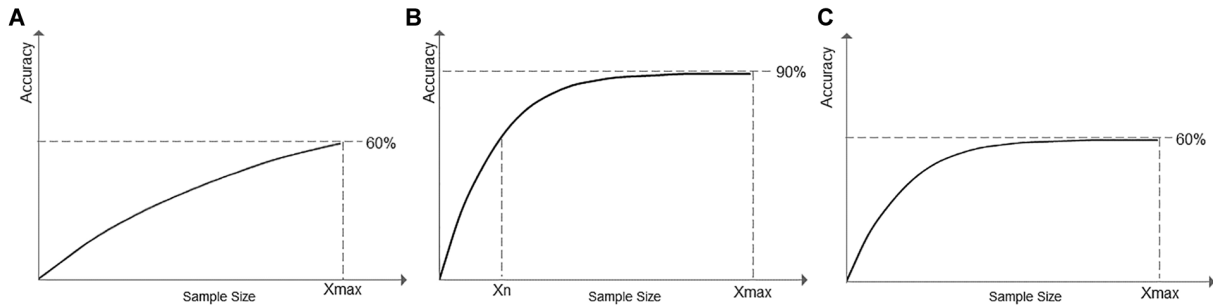


Figure 2. Examples of different learning curves.

is a turning point where the increase of performance is less rapid, and finally the algorithm reaches an efficiency threshold where the curve becomes flat.

Figure 2 shows some example learning curves. Figure 2B presents a case that is similar to a generic learning curve. For the curve in Figure 2A, with the maximum amount of data we have on hand, the curve is not flat. The x_{max} in Figure 2A might just be the x_n in Figure 2B. Thus, we could expect the model performance to improve with more data. For Figure 2C, the curve is already flat with the current sample, but the best accuracy we get is only 60%. It might indicate that the algorithm we chose is not suitable. If other algorithms also do not work well, then it might indicate that the input variables we used cannot explain enough of the variance in the output.

For models that keep absorbing new data, we can plot a similar curve with sample size (x -axis) replaced by time. This way, we can monitor the quality of the coming data and how the new data affect the model performance.

Robustness/Generalizability

Learning algorithm robustness can be understood as follows (Xu & Mannor, 2012): If a testing sample is “similar” to a training sample, then the testing error is close to the training error. Essentially, robustness reflects the degree to which a model usefully extends to data that were not used in training. Thus, robustness describes how well the model extends, or generalizes, to additional data. Generalizability is the opposite of overfitting, where the latter means that a model has good performance at training but bad performance at testing.

There are many different ways to test the robustness of a model. A frequently used method is the k -fold cross-validation method (Tan et al., 2019). With k -fold cross-validation, the whole data set is split into k subsets of equal size and each subset “takes turns” as the test set. The performance metrics are averaged over all k to yield an overall estimate. Each subset is used as training data $k - 1$ times and used as testing data 1 time.

To visualize robustness/generalizability, we could simply list the performance of the k tests and see the variance. Figure 3 shows two examples of 10-fold cross-validation that listed the accuracy of the tests. Both examples have average accuracy

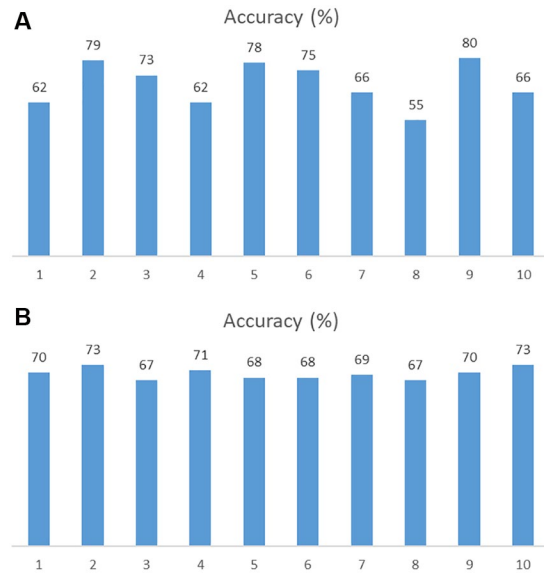


Figure 3. Example of 10-fold cross-validation.

of 70%, but the result in Figure 3A is less robust/generalizable than the one in Figure 3B. This is because, when both associated approaches are given similar testing samples, the results in Figure 3B showed steadier performance.

For models with low generalizability, we should check if the model is overfitted.

Hyperparameter Sensitivity

Hyperparameters in the context of ML are the values preset before the learning process starts. These differ from the parameters that are learned from data during the training process. Hyperparameter tuning is the process of finding the optimal values that maximize model performance. Hyperparameter sensitivity is how much the model performance changes by changing the values of hyperparameters.

During the hyperparameter tuning process, if we record the changes of the performance metric, we can see if the performance changes dramatically or slightly. Figure 4 shows two examples of hyperparameter tuning. The y -axis is a metric

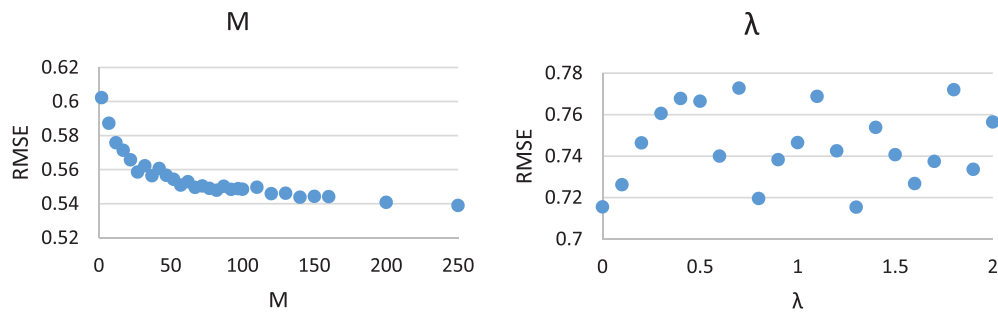


Figure 4. Example of parameter tuning.

called root mean square error to be minimized and the two x -axes are two different hyperparameters: one called M and one called λ . As M gets bigger, the error decreases and remains relatively steady. Thus, when M is large, the model is no longer sensitive to the change of M . For λ , we can see that there is no clear pattern and the error randomly jumps between 0.71 and 0.78, thus the performance does not reach a steady state. Thus, the model is always sensitive to the change of λ .

Unlike stability and robustness/generalizability, there does not exist a clear preference over hyperparameter sensitivity. However, a less sensitive algorithm might be preferred by less experienced users when it is easier to train and still produces good results (Lavesson & Davidsson, 2006).

Training Time and Inference Time

The time it takes to train a model can differ significantly between algorithms. Training time is also affected by the target accuracy, the amount of data, and the number of attributes of the problem. When the performance of two algorithms are comparable, the one with a longer training time might be undesirable.

While training time might not be interesting for the end users that are only using the trained model, the inference time (the time needed for the model to generate a prediction for a new case) could be vital for some applications, such as voice recognition, where you need a very short response time (Jiang et al., 2012). The inference time could be a filtering index when choosing algorithms for scenarios that require fast responses from the system.

EXPLAINING THE PREDICTION RESULT AND PREDICTION QUALITY

While prediction quality applies broadly across ML algorithms, in this section, we focus on classification algorithms where the output is a categorical variable.

Model Predictions May Not Map to Decision-Making Needs

For some classification problems, the prediction result is straightforward: Is the image showing a cat or a dog; is the

transaction fraud or normal; is the target on the radar friend or an enemy, and so on. However, for decision support problems, the prediction is not as straightforward. For such problems, the prediction outcome should always be translated or transformed to suggestions relevant for decision making. For example, an algorithm could provide a prediction that a stock will rise, but the actual suggestion for decision making is to buy (or hold) that stock. Also consider an algorithm that predicts that it will rain during the coming week. For most people, the actual suggestion is to bring an umbrella; but for agricultural purposes, the suggestion might be to turn off irrigation. It is thus vital to understand the human decision-making task so as to provide the information that is truly needed.

Prediction Quality: Historical Performance and Confidence

Ribeiro et al. (2016) developed a technique that uses local models to explain why classifiers give particular suggestions. Here we introduce metrics that can show the prediction quality of classifiers.

There are two different aspects to the quality (or uncertainty) of a prediction: historical performance and confidence.

Historical performance is evaluated for every ML model. The most common performance metric for a classifier is accuracy (see Sidebar “Performance Metrics”). The metrics apply only to the overall performance of the model on the known data set that was used to train it. It shows, to some extent, the amount of knowledge that is learned from the past data. However, it cannot tell you how “right” the model could be for a new unknown case.

The confidence of a model is how “sure” it is for a particular prediction. It is the estimated probability of the case in the predicted class. In binary classification, a case belongs to either class 0 or 1. If the model predicts a case x to belong to 0, then the confidence is $P(x \in 0)$. Confidence can be a good compensation for the historical performance because it is an estimation of the prediction quality for each new case, rather than the whole model.

While historical performance is an overall measure of the whole model, the confidence score is different for every single case. You can thus use this score to rank the cases. Suppose we trained a model from the past years’ decisions on graduate

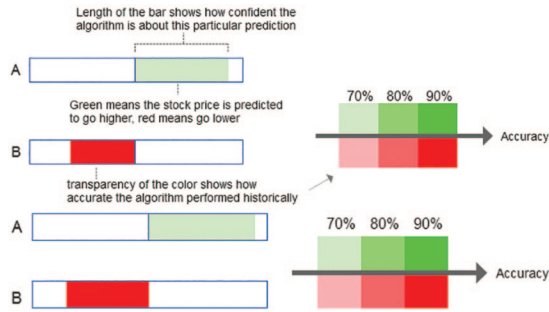


Figure 5. First version of the visualization. The top panel provides explanatory text that was provided to participants during training. The bottom panel shows the visualization without the explanation, as used during the experimental task.

admission decisions and used it for the future admission work. The model mimics how the admission committee valued an applicant in the past. Its output is a binary choice of whether to accept or reject a student. However, the optimal output of the system would be a ranking of all of the applicants, because the essence of graduate admissions is to pick the best applicants from the pool. We could calculate the probability score of $P(x = \text{accept})$ (which is equivalent to the confidence of predicting the case to belong to class “accept”) and output a rank for each applicant, rather than just an accept/reject suggestion.

An Iterative Design Study on Visualization Design for Binary Prediction Result and Prediction Quality

As suggested above, providing algorithms’ historical performance (accuracy) and confidence can help users understand the prediction quality of the algorithms. We designed two different visualizations, which could present these two variables for binary classification algorithms and used surveys to collect feedback. We used stock investment as the example decision-making task.

The first version of the design is shown in Figure 5. The prediction result is shown by the color (green—rise, red—fall) and whether the bar extends to the right (price will rise) or the left (price will fall). Accuracy and confidence are shown by the transparency of the color and the bar length, respectively. In Figure 5, algorithm A suggested that the stock price would rise, while algorithm B suggested the opposite. There was higher confidence for Algorithm A, as shown by the longer bar. Algorithm B has better historical performance because the color is deeper.

Seventy-nine individuals provided feedback on the first version of this visualization through a survey conducted on Amazon Mechanical Turk (approved by the University at Buffalo Institutional Review Board). Examining the pattern of results indicated that some participants appeared to be confused by the color coding. Regardless of the fact that the red/green colors corresponded to whether the stock price would fall or rise, participants associated the red color with

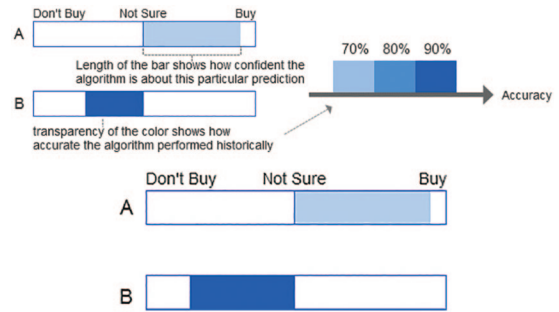


Figure 6. Second version of the visualization.

worse performing algorithms, and vice versa. Thus, participants tended to discount strong advice to avoid a stock.

We redesigned a second version of the visualization to address these issues, which resulted in better performance. In this second version (Figure 6), the color coding was replaced by simple marks on top of the bar. It avoided the “translation” (green → stock price will rise → I should buy), and instead gave the suggestion directly. It also avoided comparing transparencies among different colors.

Seventy-three participants provided their feedback on the second version of the visualization. This time, participants were able to understand the accuracy and confidence information. These results suggest that this visualization can be useful to support the explanation of binary classifiers’ outputs.

An important step in providing visual explanations of ML algorithms is to ensure that the visualizations themselves are interpretable. The improvement in performance across iterations, and the possible initial confusion regarding the red/green color scheme employed, reinforces the importance of iterative design and testing for complex visualizations. Design choices in the values of the graphical variables (color, transparency) can affect how the information presented is interpreted (Bisantz et al., 2009).

TAKEAWAYS

Supervised learning models could provide tremendous benefits by providing predictions and suggestions but suffer the disadvantage of inadequate explanation of those predictions to human decision makers. To help users understand the model training and model quality, we suggest the following techniques:

Explain what data were fed into the model by

- calculating and visualizing the input variable importance in a descending order
- explaining how the input variables were selected (why to include or exclude the variables)
- allowing experiments where users can test how controlled input variables affect the output variable
- providing some typical examples of test results so that users can verify these with their knowledge

Explain how the model is trained by recording and visualizing the following measures during training:

- **Stability:** How a model is perturbed by small changes to its inputs. Visualize this by drawing a plot to show how the model accuracy changes with the increase of the sample training set.
- **Robustness/generalizability:** How effective a model is while being tested on a new but similar data set. Visualize this by doing *k*-fold cross-validation and plotting the test accuracies.
- **Hyperparameter sensitivity:** How sensitive a model is with changes to its hyperparameters. Visualize this by drawing a plot that shows how the model accuracy changes with different values of the hyperparameter.
- **Training time and inference time:** How long it takes to train the model and how long it takes to infer prediction for a new case.

Explain the prediction result and prediction quality by

- mapping the prediction to the decision-making needs by explaining the actual meaning of the predicted value or translating it into decision-making suggestions
- calculating and visualizing model historical performance and prediction confidence

Care must be taken in designing visualization to ensure that the explanations themselves are interpretable. Iterative design and testing is important for complex visualizations.

MACHINE LEARNING

A widely recognized definition of an ML algorithm is given by Mitchell (1997): “A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*,” which in short, “computer programs that automatically improve with experience” (p. 2).

Supervised learning algorithms are ML algorithms that map input data to a desired output.

Binary classifiers are algorithms that predict the instances as belonging to one group or another.

PERFORMANCE METRICS

For binary classification problems, a case is either positive (P) or negative (N). The confusion matrix below shows the four different conditions of the result: true positive (TP), false positive (FP), false negative (FN), and true negative (TN).

		Predicted class	
		P	N
Actual class	P	TP	FN
	N	FP	TN

The performance metrics are defined as follows. Accuracy is the proportion of correct predictions among all predictions. On the contrary, error is the proportion of wrong predictions. Accuracy + Error = 1.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Error} = \frac{FP + FN}{TP + TN + FP + FN}$$

Accuracy and error can be misleading when the class distributions are imbalanced. Suppose there are 95% of noise and 5% of signal. When the algorithm predicts everything to be noise, the accuracy is 95% which is high, but it actually missed all the signals.

Thus, there are three better metrics to measure the unbalanced class problems.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\frac{2}{F_1} = \frac{1}{P} + \frac{1}{R}$$

$$F_1 = \frac{2TP}{2TP + FP + FN} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

REFERENCES

Adler, P., Falk, C., Friedler, S. A., Rybeck, G., Scheidegger, C., Smith, B., & Venkatasubramanian, S. (2016). Auditing black-box models for indirect influence. In *2016 IEEE 16th International Conference on Data Mining (ICDM)* (pp. 1–10). Institute of Electrical and Electronics Engineers. <https://doi.org/10.1109/ICDM.2016.0011>


Bisantz, A. M., Stone, R. T., Pfautz, J., Fouse, A., Farry, M., Roth, E., Nagy, A. L., & Thomas, G. (2009). Visual representations of meta-information. *Journal of Cognitive Engineering and Decision Making*, 3(1), 67–91. <https://doi.org/10.1518/155534309X433726>

Bousquet, O., & Elisseeff, A. (2000). Algorithmic stability and generalization performance. *Advances in Neural Information Processing Systems 13 (NIPS 2000)* (pp. 196–202).

Datta, A., Tschantz, M. C., & Datta, A. (2015). Automated experiments on ad privacy settings. *Proceedings on Privacy Enhancing Technologies*, 2015(1), 92–112. <https://doi.org/10.1515/popets-2015-0007>

- Figueroa, R. L., Zeng-Treitler, Q., Kandula, S., & Ngo, L. H. (2012). Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, 12, Article 8. <https://doi.org/10.1186/1472-6947-12-8>
- Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 1157–1182.
- Jiang, J., Teichert, A. R., Daumé, H., & Eisner, J. (2012). Learned prioritization for trading off accuracy and speed. *Advances in Neural Information Processing Systems 25 (NIPS 2012)*.
- Lavesson, N., & Davidsson, P. (2006). Quantifying the impact of learning algorithm parameter tuning. *American Association for Artificial Intelligence*, 1(1), 395–400.
- Lipton, Z. C. (2016). The myths of model interpretability. *KDD-98 Proceedings*. ICML Workshop on Human Interpretability in Machine Learning.
- Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267, 1–38. <https://doi.org/10.1016/j.artint.2018.07.007>
- Miller, T., Howe, P., & Sonenberg, L. (2017). *Explainable AI: Beware of inmates running the asylum or: How I learnt to stop worrying and love the social and behavioural sciences*. arXiv preprint arXiv:1712.00547.
- Mitchell, T. (1997). *Machine learning*. McGraw-Hill.
- Mittelstadt, B., Russell, C., & Wachter, S. (2019). Explaining explanations in AI. *Proceedings of the conference on fairness, accountability, and transparency* (pp. 279–288). <https://doi.org/10.1145/3287560.3287574>
- Mueller, S. T., Hoffman, R. R., Clancey, W., Emrey, A., & Klein, G. (2019). *Explanation in human-AI systems: A literature meta-review, synopsis of key ideas and publications, and bibliography for explainable AI*. arXiv preprint arXiv:1902.01876.
- Ribeiro, M. T., 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* (pp. 1135–1144). <https://doi.org/10.1145/2939672.2939778>
- Swartout, W. R. (1983). XPLAIN: A system for creating and explaining expert consulting programs. *Artificial Intelligence*, 21(3), 285–325. [https://doi.org/10.1016/S0004-3702\(83\)80014-9](https://doi.org/10.1016/S0004-3702(83)80014-9)
- Tan, P., Steinbach, M., Karpatne, A., & Kumar, V. (2019). *Introduction to data mining* (2nd ed.). Pearson Education.
- Xu, H., & Mannor, S. (2012). Robustness and generalization. *Machine Learning*, 86(3), 391–423. <https://doi.org/10.1007/s10994-011-5268-1>



Xiaomei Wang  is a PhD candidate in the Industrial and Systems Engineering Department of the University at Buffalo. She received her BS in industrial design from Xi’an Jiaotong University. Her research interests are in cognitive engineering, machine learning, human decision making, data analysis, and visualization. ORCID iD: <https://orcid.org/0000-0002-3432-0227>



Ann M. Bisantz, PhD, is a professor of industrial and systems engineering at the University at Buffalo, where she also serves as dean of undergraduate education for the university. She received her PhD in industrial and systems engineering/human machine systems from Georgia Institute of

Technology. Her research interests are in cognitive engineering, human decision making, human-computer interface design, and complex work system analysis. She is a fellow of the Human Factors and Ergonomics Society.



Matthew L. Bolton, PhD, is an associate professor of industrial and systems engineering at the University at Buffalo. He received his PhD in systems engineering from the University of Virginia. His research interests include systems engineering, human-automation interaction, formal methods, human behavior modeling, and human performance modeling. He is an associate editor of the IEEE Transactions on Human-Machine Systems.



Lora Cavuoto  PhD, is an associate professor of industrial and systems engineering at the University at Buffalo. She received her PhD in industrial and systems engineering from Virginia Tech. Her research focuses on quantifying indicators of fatigue development, understanding and modeling the effects obesity on physical capacity, and evaluating effective means for training motor skills. She is a scientific editor for Applied Ergonomics and an associate editor of Human Factors and Ergonomics in Manufacturing and Service Industries. ORCID iD: <https://orcid.org/0000-0003-4717-8378>



Varun Chandola, PhD, is an assistant professor in the Computer Science and Engineering Department and the Center for Computational Data Science and Engineering at the University at Buffalo. He completed his PhD from University of Minnesota, Department of Computer Science. His research is in the area of scalable anomaly detection and data mining for big graphs, temporal, and spatial data.



Copyright 2020 by Human Factors and Ergonomics Society. All rights reserved.
DOI: 10.1177/1064804620901641
Article reuse guidelines: sagepub.com/journals-permissions