Explainable Machine Learning Models of Consumer Credit Risk

Randall Davis, Andrew W. Lo, Sudhanshu Mishra, Arash Nourian, Manish Singh, Nicholas Wu, and Ruixun Zhang

Randall Davis
is a full professor in the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology in Cambridge, MA.

KEY FINDINGS

- The authors identify and summarize the explainability requirements for different stakeholders involved in credit risk management.
- The authors identify the right tools from the literature that can help in answering necessary explainability questions for different stakeholders in a systematic way.
- The authors modify the existing explainability tools and generate better explanations for different stakeholders.

ABSTRACT

In this work, the authors create machine learning (ML) models to forecast home equity credit risk for individuals using a real-world dataset and demonstrate methods to explain the output of these ML models to make them more accessible to the end user. They analyze the explainability for various stakeholders: loan companies, regulators, loan applicants, and data scientists, incorporating their different requirements with respect to explanations. For loan companies, they generate explanations for every model prediction of creditworthiness. For regulators, they perform a stress test for extreme scenarios. For loan applicants, they generate diverse counterfactuals to guide them with steps toward a favorable classification from the model. Finally, for data scientists, they generate simple rules that accurately explain 70%–72% of the dataset. Their study provides a synthesized ML explanation framework for all stakeholders and is intended to accelerate the adoption of ML techniques in domains that would benefit from explanations of their predictions.

The total US household debt at the end of the fourth quarter of 2020 was estimated to be $14.56 trillion, with 189.6 million new credit accounts opened within the prior 12 months.[1] The total amount of home equity lines of credit (HELOCs) at the same time was around $300 billion.[2] Although small in relative terms compared with the total US household debt, this number is still signi□cant economically, and HELOCs currently account for the majority of the banking industry's home equity portfolios. Given this massive amount of household debt, even a low delinquency rate can signi□cantly affect the operation of the business and the □nancial system as a whole. This potential impact makes the study of credit default risk an important real-world task.

Lenders generally use consumer credit ratings to grant and structure the terms of credit to consumers. To compute these credit ratings, a variety of factors that

[1] Source: New York Fed Consumer Credit Panel/Equifax.
[2] Source: FRED Economic Data.

gauge the creditworthiness of individuals have been described in the literature, which extends as far back as the 1940s (Chapman 1940). One popular example is the FICO score created by the Fair Isaac Corporation, which is used by over 90% of top lenders when making lending decisions.

Recent advances in computing, innovative algorithms, and an explosion in the quantity of data available have contributed to the growing success of complex nonlinear machine learning models, including what are known as deep learning models. Unlike traditional machine learning (ML) techniques such as logistic regression and decision trees, deep learning models may have an extraordinary number of parameters. They are able to automatically learn nonlinear representations and interactions of input features from large datasets, a process that helps them to achieve superior performance compared to other ML methods. ML models have been explored for diverse applications in economics and finance (Gogas and Papadimitriou 2021). In particular, they are widely used for a variety of different credit risk applications, including peer-to-peer lending (Ma et al. 2018; Duan 2019), mortgage risk (Kvamme et al. 2018; Chen, Guo, and Zhao 2021; Sadhwani, Giesecke, and Sirignano 2021), credit card risk (Butaru et al. 2016), consumer credit risk (Jiang et al. 2021), fair credit allocation (Tantri 2021), sovereign default risk (Dim et al. 2022), and a large pool of loans (Sirignano and Giesecke 2019).

However, regulatory compliance is a major obstacle in the adoption of black-box models for credit risk modeling. In the United States, the Fair Credit Reporting Act of 1970 mandates that lenders must be able to disclose up to four key factors that adversely affected the credit score of a rejected consumer. More recently, the European Union's General Data Protect Regulation of 2018 creates a right to explanation, whereby users may ask for an explanation of an algorithmic decision that was made about them (Goodman and Flaxman 2017). Similarly, the European Artificial Intelligence Act of 2021 imposes the requirement of global explainability for artificial intelligence (AI) algorithms.

Apart from regulatory compliance, different stakeholders have different requirements for explanation and transparency. For example, a loan applicant may want to know the possible steps that they could follow to make them creditworthy. Similarly, loan companies may need to provide explanations for their decisions regarding the creditworthiness of applicants, whereas system developers may need to understand the specific features and relationships that underpin their models.

In this article, we first model a credit risk forecast on a real-world HELOC dataset released by the Fair Isaac Corporation (FICO). This dataset is widely used to study interpretability (see, for example, Rudin and Shaposhnik 2019 and Chen et al. 2022). We developed a wide range of ML models, including interpretable rule-based models (e.g., inductive logic programming [ILP] and optimal trees) and black-box ML models (e.g., neural networks [NNs] and random forests). We compare these models and find that NNs outperform other linear and nonlinear models, reaching an accuracy of 74.75% on a task to predict the creditworthiness label provided by FICO. We also find that properly constructed explainable rules can reach an accuracy of up to 70%–72%. Our focus in this article is the explainability of ML models. As such, we do not view these models as optimal in terms of prediction accuracy, and they are
-5.3 (c)-10.52 a, we1511.9 (, we66 (r)-4..9 (e)u7.9 (r)-12.4 (t)-33 (e)-6.1 (d)1)5.3 (s op-9.5 (r)-3
tuneaic3e7 preabi 1li (t)-3..5 (y o)-0.8 (o)-2.5 (r)- 0.52 a,.2 (e m)-9.9 (r)0.7 (r)-46.1 (t)-3..5 (y o)(,.2

LITERATURE REVIEW

A range of statistical and operational research methods has been used over the long history of credit scoring models (Thomas 2000). Many existing models learn relationships between risk factors and borrower behavior, either using logistic regression (Campbell and Dietrich 1983; Cunningham and Hendershott 1984; Elul et al. 2010; Agarwal et al. 2011) or the Cox proportional hazards model (Green and Shoven 1983; Deng, Quigley, and Van Order 2000; Stanton and Wallace 2011). Some studies find nonlinear relationships between the risk factors and mortgage risk (Elul et al. 2010; Foote et al. 2010; Agarwal, Chang, and Yavas 2012).

More recently, ML techniques have been adopted for credit risk models. For example, Sadhwani, Giesecke, and Sirignano (2021) develop a deep learning model and find significant nonlinearities in mortgage delinquency, foreclosure, and prepayment risk. Kvamme et al. (2018) predict mortgage default using convolutional NNs. Lessmann et al. (2015), Thomas (2000), Breeden (2020), and Gogas and Papadimitriou (2021) give detailed surveys of ML methods for credit risk forecasting. Gogas and Papadimitriou (2021) also highlight the shortcomings of ML with regard to the explainability of the models for fintech applications.

On the other hand, an extensive literature has been developed about explainable AI in health care, computer vision, and natural language processing (Adadi and Berrada 2018; Molnar 2020). Even though these techniques were developed for other domains, some have been adopted for explaining credit risk forecast models.

A variety of solutions have been proposed to deal with the shortcomings of ML models used in credit risk forecasting. Interpretable ML models have been used by Khandani, Kim, and Lo (2010) (decision trees for a consumer credit risk model) and Obermann and Waack (2016) (a multiclass rule-based model for corporate credit ratings). Similarly, Dumitrescu et al. (2021) propose an interpretable penalized logistic tree regression model for credit scoring, while Chen et al. (2022) use an interpretable two-layer additive risk model for a HELOC dataset. Some studies have explored the use of network-based (Giudici, Misheva, and Spelta 2020) or latent factor (Ahelegbey, Giudici, and Misheva 2019) models for peer-to-peer lending, which can be more interpretable than traditional deep learning models. A two-layer additive risk model was an award-winning model in the recent FICO data science challenge. For these models, interpretability generally comes at the cost of performance compared to black-box models. For example, Caruana and Niculescu-Mizil (2006) found that random forests outperform decision tree classifiers, but random forests are not interpretable. In this article, we show that an NN outperforms the two-layer additive risk model proposed in Chen et al. (2022), which we use as a benchmark. Most of these explainable ML models are developed for applications involving structured data.

There is also a rapidly growing literature on post hoc interpretable ML methods. These methods can generally be used on both structured and unstructured data. Sadhwani, Giesecke, and Sirignano (2021) use first-order derivatives and cross partial derivatives of the fitted transition probability with respect to features to study marginal and interaction effects. Bussmann et al. (2021) use Shapley values to explain tree-based ensemble models and apply correlation networks to group the borrowing companies using the derived explanations. Similarly, Misheva et al. (2021) use LIME and SHAP to obtain local and global explanations for models trained on a Lending Club dataset. Albanesi and Vamossy (2019) propose a deep learning–based approach that combines the outputs of tree-based ensemble models and NNs to pre-

Bracke et al. (2019). More recently, Qadi et al. (2021) propose a human-in-the-loop ML method that combines a post hoc explanation from SHAP with explanations from credit risk experts. Rudin and Shaposhnik (2019) propose a minimum set cover problem to generate a rule-based summary of an ML model on the HELOC dataset. Similarly, Martens et al. (2007) propose a rule extraction method from a trained support vector machine for credit scoring. There are also some post hoc interpretability methods, including accumulated local effects (ALE) and leave one covariate out (LOCO) (Lei et al. 2018; Apley and Zhu 2020), that can be used for finding the global relationship between input features and output. Finally, Ponomareva and Caenazzo (2019) propose the layerwise relevance propagation and activation analysis of hidden units of an NN.[3]

Although these methods generate explanations of model outcomes, they are not tailored specifically for the different stakeholders involved in the credit risk pipeline. For example, many of the aforementioned methods do not provide local explainability that is necessary for many real-world applications. Because interpretability for one party may not always translate to interpretability in the eye of another party, none of the existing methods can be generalized for all stakeholders. In addition, they do not embed the specific constraints of the dataset in their methodology of generating explanations, which may lead to explanations that are not practical or useful. Interpretability for different stakeholders has been introduced in a demo by IBM,[4] but it remains incomplete and does not use the state-of-the-art methods that we employ in this study.

In this article, we present a methodology that generates explanations of black-box models for different stakeholders, a direction that has not been well explored previously. In some cases, we improve existing methods by making necessary modifications for our purposes, whereas in others we develop new models. In the process of generating these explanations, we embed constraints to ensure that explanations and suggestions are meaningful and can be adopted for real-world use.

ML METHODS

The fundamental goal of credit scoring is to determine the creditworthiness of an individual. Simply put, it is a binary classification task that labels credit applicants as creditworthy or noncreditworthy. A creditworthy applicant is likely to repay their financial obligation, whereas a noncreditworthy applicant is not.

We frame the consumer credit risk classification problem as predicting the probability of default for a borrower. More specifically, given a set of features, $\vec{x}_i = x_{1i}, x_{2i} \ldots. x_{ki}$, for a borrower i, the task is to predict a variable y_i, that is, the probability of default, $Pr(Default)$. The features $\vec{x}_i$ describe the borrower's credit history, for example, the number of lines of credit and the number of times the borrower defaulted in the past, among others. In practice, y_i is determined by long-standing credit scoring models, which are characterized by decisions such as whether the applicant had 90+ days delinquency in the two years after the opening of a new line of credit.

We frame the classification as a supervised learning problem in which we train a function f that can approximate the relationship $y_i = f(\vec{x}_i)$. To train the function, f, we use a variety of ML models, including optimal classification trees, random forests, ILP, and NNs. We describe these models next.

[3] There are also other interpretability methods, such as those discussed in Giudici and Raffinetti (2021) and Giudici, Mezzetti, and Muliere (2003), that are yet to be applied to credit risk management.

[4] https://aix360.mybluemix.net/data.

Random Forests

Random forests are an ensemble supervised learning technique. This technique aggregates multiple outputs from a set of predictors—in this case, multiple decision trees—in the belief that this will produce a more accurate classifier.

The key idea behind random forests is that a high-performing classifier can be constructed from a set of nonexpert classifiers that are decision trees. A single decision tree is a supervised learning method that predicts the value of a target variable by learning simple if–then decision rules. It is constructed using the classification and regression tree (CART) algorithm (Breiman et al. 1984). Each node in the decision tree is a condition on the value of a single feature that splits the data into two subsequent branches. CART recursively identifies the feature–value pair that best minimizes the tree's Gini impurity, a metric of the disorderliness of the labels of a set of data points.

Random forests are trained via a method called bootstrap aggregation, or bagging. The training data points are first randomly assigned into n groups with replacement, where n corresponds to the number of decision trees. Individual decision trees are fitted to a randomly chosen set of features in each group. To classify a new data point, the random forest aggregates the predictions from each of its constituent decision trees and uses the majority vote as its classification. The random sampling of data points and features ensures that the resulting decision trees are uncorrelated. Thus, by aggregating their independent predictions, random forests are able to reduce variance and improve generalizability.

However, random forests are difficult to interpret. For individual data points, each decision tree gives its if–else conditions that lead to a classification, but when these conditions are combined over the many trees of an ensemble, interpreting these conditions is impossible. This makes random forests effectively a black-box model.

ILP

ILP (Muggleton 1991) involves using first-order logic to represent and explain data. The dataset can be represented by a finite set of rules or clauses. In this section, the ILP developed is data-driven and clauses are learned from the input data for the ILP model.

ILP requires that we specify the number of rules, N, a given maximum rule size, R, and the dimensional n binary input vector X. We construct Π, an $N \times R \times 2n$ tensor, and interpret softmax($\Pi[i,j]$) as a probability distribution for the jth term in the ith rule; that is, we obtain a $2n$-sized discrete probability distribution over the n features and their negations.

The rules are learned in a disjunctive normal form that consists of clauses with $\hat{\wedge}$ (AND) and $\hat{\vee}$ (OR) conditions. The AND and OR operators require binary operands. These operators, however, must be extended to continuous operands to learn clauses from data. We use the product as a continuous extension of $\hat{\wedge}$, while a continuous extension of $\hat{\vee}$ is obtained from DeMorgan's law, $\hat{\vee}(x) = 1 - A(1 - x)$, where A is the continuous extension of $\hat{\wedge}$ (product as described earlier). From parametrization and using these continuous extensions of $\hat{\vee}$ and $\hat{\wedge}$ on $[0, 1]$, we can compute an approximated label $\hat{y}$ for an input vector X by concatenating X with its $1 - X$ to get a $2n$-vector X^*:

$$\hat{y} = \hat{\vee}_i \, \hat{\wedge}_j \, (X^* \cdot \text{softmax}(\Pi[i, j]))$$

where i ranges over the number of rules and j ranges over the size of a rule.

The binary cross-entropy loss between $\hat{y}$ and ground truth y can be minimized using stochastic gradient descent in order to learn the probability distributions in Π. Once the model is trained, we use the probability distribution Π to obtain a set of

logical rules in disjunctive normal form, for all n from 1 to N and r from 1 to R, using the following expression:

$$\text{Target} \leftarrow \vee_{n=1}^{N} \left(\wedge_{r=1}^{R} \text{argmax}_k \left(\Pi[n, r, k] \right) \right)$$

In our study, X is a numerical vector with each dimension corresponding to a different feature value. We first rank-normalize it to change its range to $[0, 1]$ and use the transformation T to binarize it: $T(x) = \sigma(a(X - b))$, where σ is the sigmoid function, a is the fixed scale parameter, and b is the parameter learned during optimization. We use stochastic gradient descent to learn the clauses based on the input data. Using the ILP model, we can thus learn simple interpretable rules that can be used for classification and summarizing datasets.

Optimal Classification Trees

Decision trees are constructed in a top-down greedy way using the CART algorithm (Breiman et al. 1984), as described in the "Random Forests" section. At every node, the split is decided locally without the knowledge of other nodes. This makes decision trees only one-step optimal, leading to poor performance when classifying unseen points.

Optimal trees (Bertsimas and Dunn 2017) are a variant of decision trees that are learned in a globally optimal manner. Optimal trees decide their split in one step, with knowledge of all other splits. The tree learning process is modeled as a mixed-integer optimization problem, which can be solved using fast available optimizers. Because optimal trees are constructed in a globally optimal fashion, they perform better than decision trees, and have all the advantages of other decision trees in terms of explainability. Optimal trees, however, become difficult to explain if they are very deep, and the rules learned at every node are complex. In addition, the number of variables involved in optimization for the creation of an optimal tree model is a linear function of dataset size and an exponential function of maximum depth. In general, mixed integer optimization does not scale well with a large number of variables. As a result, deeper optimal trees take a longer time to train on large datasets compared to decision trees, which limits their use for Big Data problems.

NNs

NNs are supervised learning models loosely inspired by the biological networks of the human brain. An NN is composed of three types of layers: an input layer, a number of hidden layers, and an output layer. An input layer relays the input features into the model, the hidden layers act as the computational engine, and the output layer generates the final model prediction. The input to each hidden unit is a linear combination of the units of the preceding layer. The hidden unit then computes its output by mapping its input through an activation function. A nonlinear activation function is commonly used to create nonlinear interactions between the units of the NN. It is worth noting that logistic regression is a special case of an NN, with one hidden layer containing one hidden unit with a sigmoid activation function.

In NNs, the value of each hidden unit can be computed as

$$h_j(x) = f\left(w_j + \sum_{i=0}^{n} w_{ij} \cdot z_i \right)$$

Here, w_{ij}

a one-hot encoding version of the dataset. The details of these implementations and best-performing models are included in the "Model Implementation Details" section of the appendix, for reproducibility.

We include the two-layer additive risk model in our evaluation (Chen et al. 2022), which was the best-performing model of the FICO Data Challenge.[7] It partitions features into different subgroups, combining scores from different subgroups into a global model score. Its feature subgroups are generally interpretable because they are created through the intervention of an expert in the field. The model resembles a two-layer sparse NN. This model serves as a baseline for the other black-box models.

We evaluate these models using K-fold cross-validation. In K-fold cross-validation, the input dataset is randomly partitioned into K equal subsets. In each run, one of the subsets is chosen as the test set and the remaining as the training set. For this analysis, we choose $K = 5$, that is, in each run of cross-validation, the training set contains 80% of the dataset (7,898 points), and the test set contains the remaining 20% (1,973 points). The class distributions of the train–test sets are included in Exhibit A1 in the appendix. The standard errors of the performance measures are obtained by bootstrap sampling of the dataset (keeping a train/test split of 80%/20%) followed by retraining of the model and estimating the performance metric for 1,000 iterations. The performance measures for 1,000 iterations can be used to obtain standard errors.

We evaluate the models using different metrics: accuracy, area under the curve (AUC), false positive rate (type I error), and false negative rate (type II error). The metrics are averaged across all five cross-validation datasets. The AUC measures the ability of a classifier to distinguish between the classes. Our dataset is fairly balanced, as is illustrated in Exhibit A1, hence the accuracy is also a good measure of goodness of fit along with the AUC. All models predict the *Pr*(*Default*) values for every sample. To classify the different samples, we used a threshold of 0.5.

As Exhibit 1 illustrates, we find that the NN model trained on WoE data[8] has the best performance, with an accuracy of 74.75%. It outperforms the NN trained on one-hot encoded data, implying that the WoE-encoded features have more

EXHIBIT 1

Fivefold Cross-Validation Performance of ML Models

Models	Test Accuracy	Test AUC	False Positive Rate	False Negative Rate	Precision
ILP (one rule)	$70.71_{(0.92)}$	$70.64_{(0.93)}$	$31.13_{(1.38)}$	$27.57_{(1.20)}$	$71.56_{(1.24)}$
ILP (two rules)	$70.92_{(0.94)}$	$70.62_{(0.94)}$	$36.98_{(1.48)}$	$\mathbf{21.75}_{(1.21)}$	$69.59_{(1.21)}$
Optimal Trees (interpretable)	$72.28_{(1.03)}$	$74.18_{(1.74)}$	$28.25_{(4.26)}$	$27.21_{(2.99)}$	$71.59_{(2.16)}$
Optimal Trees (black box)	$74.12_{(0.46)}$	$74.46_{(0.50)}$	$29.16_{(2.63)}$	$22.85_{(2.78)}$	$73.54_{(1.14)}$
Random Forest (140 trees)	$73.77_{(0.01)}$	$79.82_{(0.01)}$	$29.71_{(0.02)}$	$23.01_{(0.02)}$	$73.61_{(0.02)}$
Two-Layer Additive Risk	$74.12_{(0.88)}$	$81.01_{(0.83)}$	$30.31_{(1.28)}$	$21.77_{(1.17)}$	$73.63_{(1.15)}$
NN (one-hot)	$74.10_{(0.88)}$	$80.59_{(0.84)}$	$\mathbf{25.16}_{(2.50)}$	$26.60_{(2.25)}$	$74.06_{(1.53)}$
NN (WoE)	$\mathbf{74.75}_{(0.98)}$	$\mathbf{81.40}_{(0.84)}$	$28.31_{(1.56)}$	$22.42_{(1.37)}$	$\mathbf{74.70}_{(1.30)}$

NOTES: The NN trained on WoE data performs the best with a mean test accuracy of 74.75%, precision of 74.70%, and AUC of 81.40%. A higher AUC implies the NN model is better able to distinguish between the creditworthy and noncreditworthy classes compared to other models. All rule-based classifiers have smaller values of AUC compared to the others. The NN (one-hot) has the minimum false positive rate, whereas two rules generated using ILP has the minimum false negative rate. All the reported values are in percentages. The values in parentheses are the standard errors of the estimates.

[7] http://dukedatascience.co.cs.duke.edu/.

[8] "Data Preprocessing" in the appendix provides a summary of the WoE feature preprocessing.

information than one-hot encoded features. The two-layer additive risk model performs second best, with an accuracy of 74.12%. This model is easy to interpret; however, it requires the input and involvement of experts who know about the constraints in the dataset and the relationship between its different features (in order to divide features into different subgroups), which might not always be available for various applications.

On the other hand, the rule-based models, optimal trees (black box), and random forests suffer from the problem of explainability. For random forests, analyzing the ensemble of 140 trees at a time is intractable. Similarly, the optimal trees (black-box) model has a single deep tree with multiple features deciding every split, which is difficult to interpret. If we train a shallow optimal tree, its accuracy drops to 72.28%, but the model is easy to interpret and analyze. We also obtain a small set (one or two) of simple rules from ILP that are easy to understand, but they come at the cost of a decrease in accuracy. Even though interpretable rule-based models do not perform as well as the NN model, we will see in "Explaining ML Models" that they are useful for explainability for the different stakeholders involved.

Using ILP, we obtain a single simple rule—"If *ExternalRiskEstimate* is smaller than 72, then classify borrowers as noncreditworthy"—which achieves an accuracy of 70.71%. *ExternalRiskEstimate* is a condensed version of the borrower's credit risk, a metric similar to the FICO score.[9] In general, companies use the credit score and a threshold associated with it to decide an individual's creditworthiness; in our case, it is the *ExternalRiskEstimate* with a threshold of 72. We observe that the use of ML models using different features about a borrower's credit history can increase the accuracy of the delinquency forecast by four percentage points compared to the naïve use of credit scores.

Credit companies are naturally interested in using models with the best performance. If credit companies give loans to borrowers by misclassifying defaulters as nondefaulters (i.e., false negatives), they will suffer losses. Similarly, if credit companies deny loans to borrowers who can repay (i.e., false positives), they will lose business. A 4% improvement in accuracy may not seem big, but given the enormous size of the mortgage business in the United States, even a small increase in model performance can create a substantial economic impact.[10] Moreover, a more accurate model can lead to more borrowers having access to loan opportunities in aggregate.[11] However, regulatory practice and the black-box nature of the models prevent them from harnessing these benefits. This motivates us to analyze the explainability of these different models.

EXPLAINING ML MODELS

We have shown in the "Evaluation" section that the NN model outperforms every other model tested. However, the noninterpretable nature of NNs limits its adoption. In this section, we analyze the explainability of models in order to help their wider adoption for different applications.

The multiple parties involved in credit risk management require different explanations for different purposes. For instance, loan applicants who are denied loans

[9] Although this may not be an interpretable feature by itself, we keep this feature in the analysis because it is widely used and our goal is to generate an explanation methodology given a dataset and model irrespective of features included in the study.

[10] In practice, with more data available by real lenders, the improvement in model performance should be even bigger than our proof-of-concept study.

[11] See, for example, the model of Fuster et al. (2022).

are interested in finding the reason for the denials and suggestions that can make them more creditworthy. Data scientists, on the other hand, are more interested in understanding the data, whereas regulators demand fairness from the models and analyze the model's behavior in extreme scenarios. We characterize the kinds of explanation that are appropriate for the following end users: loan companies, data scientists, loan applicants, and regulators.

Interpretability for Loan Companies: Opening the Black Box

Loan companies use ML models to evaluate the creditworthiness of a borrower. Regulations require the loan companies to give a set of reasons for every denial of the application. Consequently, loan companies are interested in finding the factors that contribute to the creditworthiness of the individual. In addition, they require explanations for every prediction of the model. These explanations make the model transparent, and they assist in finding the most representative samples (borrowers from the past) for a data point (a new borrower). We use state-of-the-art methods to generate explanations for our model predictions, including LIME (Ribeiro, Singh, and Guestrin 2016) and SHAP (Lundberg and Lee 2017). We generate explanations for the best-performing NN model. Our key contributions in this section include applying and comparing existing models (LIME and SHAP) to generate explanations for loan companies, modifying LIME to improve the validity of linear approximations in our context, and using the *k* nearest neighbor (kNN) algorithm on the explanation to find representative samples for a data point.

LIME

LIME is a model-agnostic technique that approximates the decision boundary of a model at a particular data point using a linear approximation.[12] This linearity makes the LIME model interpretable. The approximation is constructed by training a locally weighted linear regression model in the neighborhood of the data point of interest. The coefficients of the regression can be used to justify the data point's classification. Because the features are on the same scale (WoE encoded), the coefficients are comparable.

Constructing a locally weighted linear regression model involves sampling and perturbing the data points that are used for training. One of the shortcomings of LIME is that the perturbed data points sampled by LIME may be invalid. For example, imagine a dataset with two features A and B, with a constraint that $A < B$. Sampling each feature's value independently, as is done in LIME's original algorithm, may produce perturbed data points where this constraint is violated. We address this shortcoming of LIME by modifying the algorithm to take into account the potential interdependencies of different input features.

This modification changes the methodology of sampling a data point of the LIME algorithm. In its original implementation, a data point with *n* features is sampled by independently sampling each of its *n* features from univariate normal distributions. In our modified implementation, a data point is sampled directly from a joint multivariate normal distribution across all features. This allows perturbations to be informed by the correlation of features. The multivariate normal distribution is still centered on the mean of each feature value, but the standard deviation along each feature is determined by the correlation matrix of the input data.

[12] The key idea behind LIME is that every segment of the decision boundary begins to look linear at increasingly smaller scales.

We first evaluate our modified LIME with respect to the validity of the perturbed data points. As noted, LIME samples perturbed points in the neighborhood of the input data point. We generate a set of 5,000 perturbed data points from several univariate normal distributions centered on the feature means and a single multivariate normal distribution centered similarly. We measure the quality of the sampled points by two metrics.

Correlation. The correlation between features of the perturbed data points should be similar to that of the features in the training dataset. We compute the mean-squared error (MSE) between the correlation matrix of the training data and that of the perturbed points generated by both implementations of LIME. Let us call these values $MSE^{original}$ and $MSE^{modified}$, respectively. We find that $MSE^{modified}$ is much smaller than $MSE^{original}$ (0.000 versus 0.053). This is expected because the correlation matrix of the training data is an input to the multivariate normal distribution that samples perturbed points in our modified implementation. Our perturbed data more closely resemble the characteristics of the original dataset.

Constraint violation. There are 12 constraints relevant to the HELOC dataset (six relational constraints and six value constraints). We determined these via discussions with FICO personnel. Examples of these constraints include "All feature values with percentage units should be smaller than 100" and "The number of lines of credit not delinquent must be less than the total number of lines of credit." The 5,000 perturbed data points are scanned for violations of these constraints. The results for all constraints are shown in Exhibit 2. We see that the modified implementation of LIME produces fewer violations than the original implementation in 7 out of the 12 constraints. Moreover, a perturbed point sampled by the modified implementation has, on average, 1.954 constraint violations versus 2.627 for the original implementation. Hence, the modified LIME does provide an improvement in terms of generating more realistic data points.

We also compute the average goodness of fit of the linear model in LIME for all test data points. We find that the average R^2 of the modified LIME is 90%, compared to 88% for the original LIME. Comparing correlation, constraint violation, and goodness of fit, we conclude that the modified LIME leads to better surrogate models. In the

EXHIBIT 2

Constraint Violation

Index	Constraint	Original LIME Violations	Modified LIME Violations
1	All feature values interpreted quantitatively must be non-negative	4,383	4,024
2	PercentLOCNeverDelq ≤ 100	1,198	1,183
3	PercentInstLOC ≤ 100	1	2
4	PercentLOCWBalance ≤ 100	340	296
5	FracRevLOCLimitUse ≤ 100	62	75
6	FracInstLOCUse ≤ 100	509	506
7	NumLOC90PlusDaysDelq ≤ NumLOC60PlusDaysDelq	1,656	655
8	NumLOCReqLast6MExPastWeek ≤ NumLOCReqLast6M	2,095	388
9	NumLOC60PlusDaysDelq ≤ NumTotalLOC	191	239
10	NumLOC90PlusDaysDelq ≤ NumTotalLOC	192	233
11	NumLOCNotDelq ≤ NumTotalLOC	2,257	1,915
12	NumLOCInLast12M ≤ NumTotalLOC	249	255

NOTES: This exhibit shows a comparison of constraint violations among 5,000 perturbed points sampled from the original and modified implementations of LIME. The points sampled from the modified implementation of LIME have fewer violations in 7 out of the 12 constraints.

remainder of the article, we use this modified LIME for our analysis, which we refer to as the LIME model for simplicity.

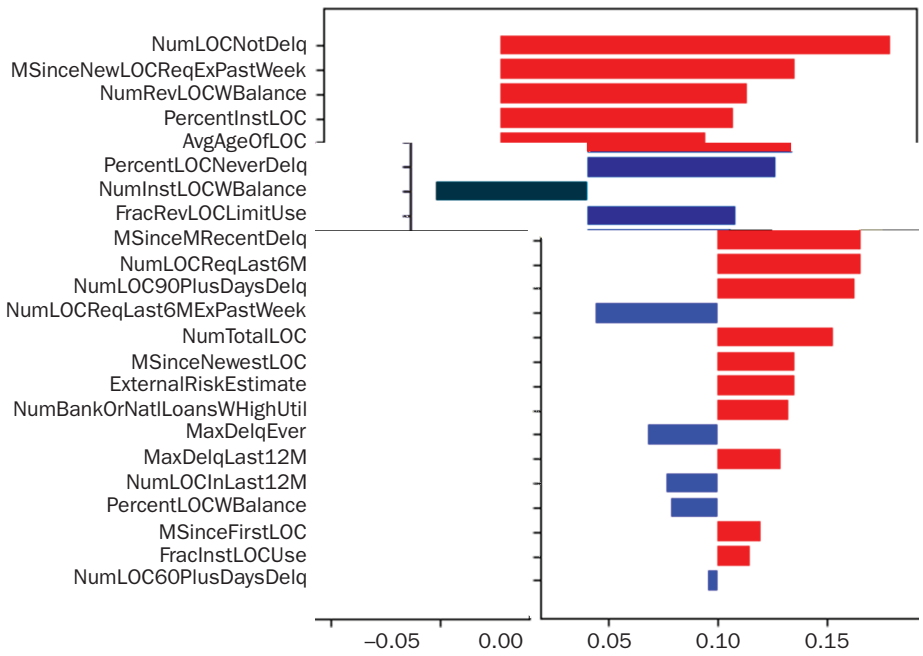
Explaining Model Predictions (LIME)

LIME learns a linear surrogate model. As a result, for each data point, the coefficients of its linear regression can be interpreted as the change in the output produced by a unit change in the corresponding feature value, given that other feature values are held constant. Exhibit 3 shows an example of a LIME explanation. A unit increase in the values of features with coefficients shown in red lines will increase the noncreditworthy probability, whereas a unit increase in the values of features with coefficients shown in blue lines will decrease the noncreditworthy probability.

After obtaining the feature importance for all the data points, they can be aggregated to find the overall feature importance. In Exhibit 4, we look at the global feature importance for the model. The top three most significant features are the months since the newest request for a new line of credit (excluding those requested in the past week), the external risk estimate, and the fraction of all revolving line of credit limits in use. It is worth emphasizing that the external risk estimate is an important feature of the model but not the most important one. This can be understood by observing the nonlinear nature of the classifier in Exhibit 4, Panel A. For example, for the feature "months since the newest request for a new line of credit (excluding those requested in the past week)," having small and large values both contribute to a decrease in the noncreditworthy probability, while having values around the median contributes to an increase in the noncreditworthy probability. Such a nonlinear relationship would be

EXHIBIT 3

The LIME Model Approximation for a Random Data Point

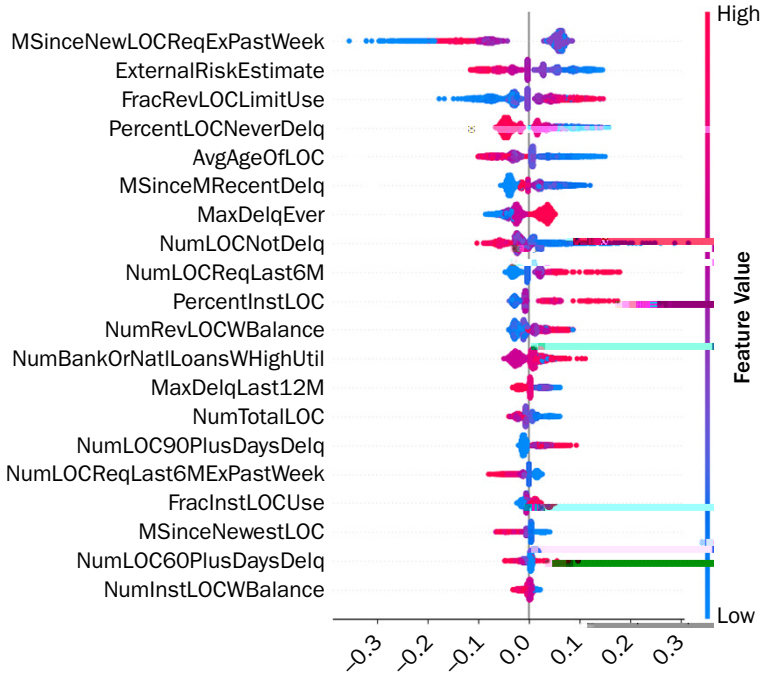


NOTES: A unit increase in features with a red horizontal line will increase the noncreditworthy probability. Similarly, a unit increase in features with a blue horizontal line will decrease the noncreditworthy probability. The contribution of each feature in the model explanation can be obtained by multiplying each feature coefficient by the feature value.

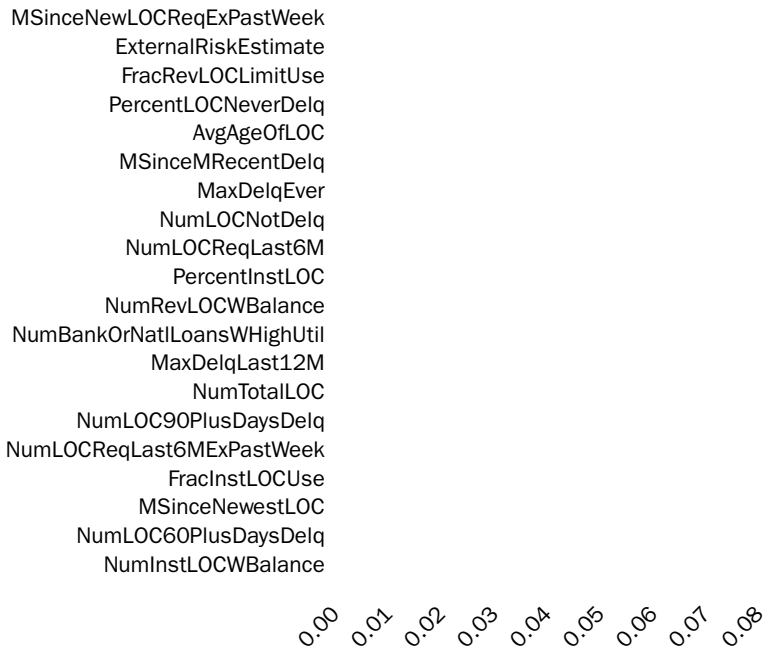
EXHIBIT 4

Feature Importance Obtained by Aggregating LIME Explanations for All Samples in the Test Set

Panel A: Feature Contribution for All Samples in the Test Set



Panel B: Average Feature Contribution (absolute value) for All Samples



NOTES: In Panel A, the color coding (blue-red) represents the value of the feature. For example, for the external risk estimate, the larger values (in red) contribute to the decreasing noncreditworthy probability, whereas the smaller values (in blue) contribute to the increasing noncreditworthy probability. The contributions are defined relative to 0.5 (equal probability of default/nondefault). The *x*-axis is the contribution of the individual feature in the output of the model relative to 0.5. For example, for a data point, if the contribution of all other features is zero and the contribution of the external risk estimate is −0.2, the model's output will be 0.5 − 0.2 = 0.3. Panel B is obtained by aggregating the feature importance as obtained for all points, which illustrates that the three most significant features are months since the newest request for a new line of credit (excluding those requested in the past week), the external risk estimate, and the fraction of all revolving line of credit limits in use.

difficult to capture using traditional models. These generated explanations and the overall feature importance also aid in discovering biases in the model.

The quality of LIME explanations depends on the fidelity of the LIME model. We evaluate the fidelity of the approximations produced by our modified implementation of LIME for the best-performing NN. We do this by constructing an approximation at each of the 1,973 points in the test dataset and computing the fraction of times the NN and the LIME approximations produce the same classification. We find that out of 1,973 data points, 1,935 points have the same classification for the linear LIME approximation and the NN model being analyzed. The high fidelity of the LIME approximations shows that LIME is able to generate valid linear approximations that can be trusted for a large number of data points.

However, for the few cases in which the classification of the NN model and LIME do not match and are significantly different, the linear approximation cannot be trusted. In order to generate explanations for these points, we discuss another method.

SHAP

SHAP is an explainable AI method with an economic foundation. It performs the Shapley value decomposition of the model output, giving the contributions of every feature at a data point. Like LIME, it is also a model-agnostic method.

Shapley values have several good properties that satisfy a number of important criteria, including local accuracy (i.e., the model explanation matches the original prediction), handling of missing data (i.e., if the feature is absent, its contribution to the model prediction will be zero), and consistency (i.e., if the model changes in a way that leads to larger marginal contributions for a feature, the Shapley values also increase). Shapley values also incorporate the interactions between features in the process of calculating feature importance, making SHAP a more reliable method for interpretability than LIME.

We use the kernel SHAP implementation from Lundberg and Lee (2017). The kernel SHAP procedure computes the Shapley values by running a weighted least squares regression whose solution is the Shapley values of features. To explain a point $\vec{x}_i$, the different points used in the linear regression are obtained by selecting a subset of features from $\vec{x}_i$, and the remaining subset of features not selected are replaced with values from background data points (i.e., from training data). The weights of the linear regression are decided by the size of the sampled subset. For example, a subset of one feature has the maximum weight because it provides the maximum information about that feature's contribution. Using the described weighted least squares regression, we obtain Shapley values for the data point $\vec{x}_i$. It is worth highlighting that the kernel SHAP implementation relies on taking subsets of features, that is, $2^{Numberoffeatures}$. As a result, kernel SHAP does not scale well.

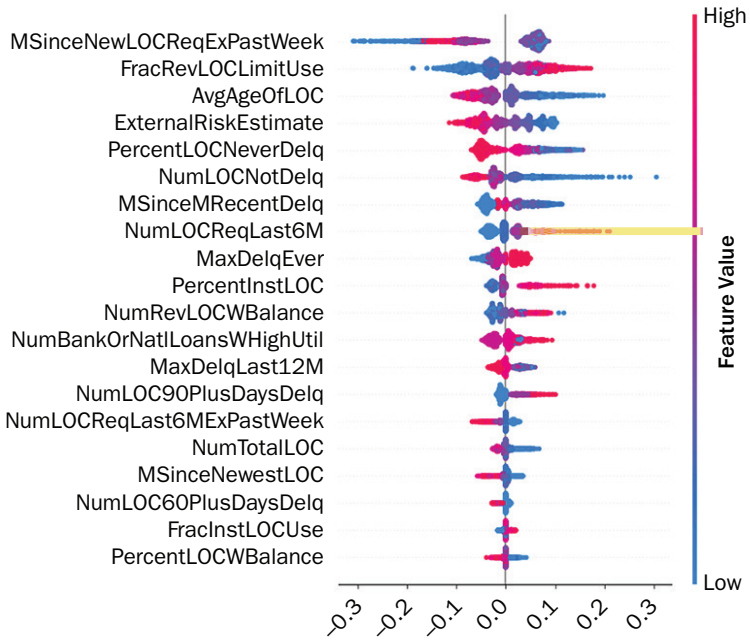
Using SHAP, we obtain explanations for the classification of all the data points in the test set. Similar to LIME, the Shapley values for features can be aggregated for all test points to find the overall impact of a feature on the model. We present the feature importance for all data points in the test sample in Exhibit 5, which shows that the months since the newest request for a new line of credit (excluding those requested in the past week), the fraction of all revolving line of credit limits in use, and the average age of lines of credit are the top three most important features for the model. We observe a minor difference in the feature contribution obtained from LIME and SHAP. This may be due to the fact that not all LIME approximations are locally accurate.

It is also worth emphasizing that both LIME and SHAP have their weaknesses, which may affect the explanations in the credit risk domain. For example, in addition

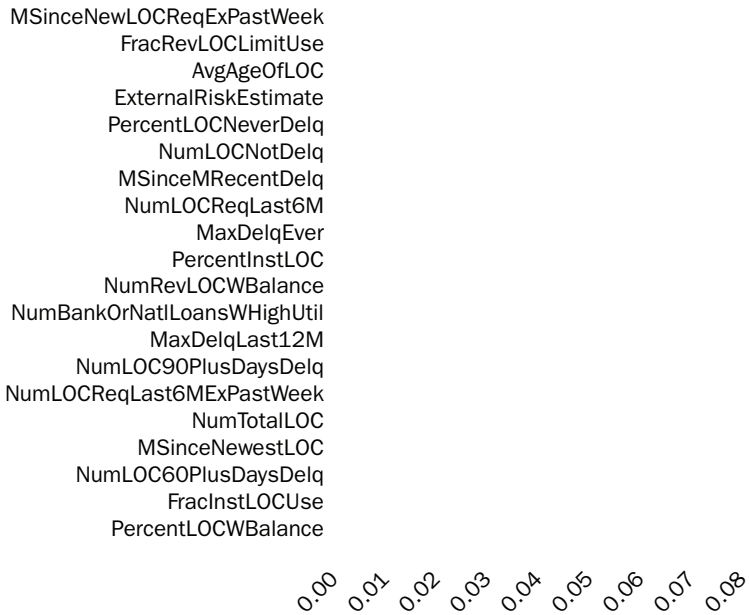
EXHIBIT 5

Feature Importance (Shapley Values) Obtained by Aggregating Kernel SHAP Explanations for All the Samples in the Test Set

Panel A: Feature Contribution for All Samples in the Test Set



Panel B: Average Feature Contribution (absolute value) for All Samples



NOTES: In Panel A, the color coding (blue-red) represents the value of the feature. For instance, the larger values (in red) for the external risk estimate contribute to the decreasing noncreditworthy probability, whereas the smaller values (in blue) contribute to the increasing noncreditworthy probability. The contributions are defined relative to 0.5 (equal probability of default/nondefault). The *x*-axis is the contribution of the individual feature in the output of the model relative to 0.5. For example, for a data point, if the contribution of all other features is zero and the contribution of the external risk estimate is −0.2, the model's output will be 0.5 − 0.2 = 0.3. We obtain the overall feature importance (as illustrated in Panel B) by aggregating Shapley values for all test points. The most important features are the months since the newest request for a new line of credit (excluding those requested in the past week), the fraction of all revolving line of credit limits in use, and the average age of lines of credit. These are largely consistent with the LIME feature importance. Small discrepancies can be attributed to the inaccurate local approximations of LIME.

to feature importance, we can obtain locally linear approximations using LIME (as shown in Exhibit 3), which are not available in SHAP. These local approximations of the decision boundary are necessary for other applications, which will become clear in the next section. However, the explanations using LIME might not satisfy important properties like local accuracy and consistency that Shapley values allow. In addition, we have incorporated feature correlations in the sampling of data points for learning the local linear model in LIME, which leads to fewer constraint violations in the input feature vector, hence creating more realistic data points. However, in SHAP such feature correlations are not considered when computing Shapley values.

From the computational perspective, different runs of LIME produce slightly different explanations because of the randomness in s (e)2 ()e4.7 ()-8.1 (c)3.7 (d)--10l(i) [(S)-2 (H)- [illegible]

We use two extreme customers in our stress testing. In the first case, an extreme customer is obtained by sampling points from a multivariate normal distribution with extreme values. While sampling, some feature values may no longer satisfy the feature-specific constraints; for them, we manually truncate the feature values. In the second case, we use a customer whose feature values are all −9, that is, there is no bureau record or investigation on file about this customer. We have multiple such data points in the dataset that are not included for training or testing purposes. The extreme points used in our analysis are included in Exhibit 6.

Regulators are interested in verifying the validity of the model's output and analyzing its behavior. We preprocessed features for these extreme customers using the WoE encoding, which leads to extreme values being assigned to one of the bins. Due to this binning, the model was able to produce valid predictions. For case 1, the output is *Pr*(*Defe-Td(()TjEMC*

EXHIBIT 7

LIME Explanations for the Two Extreme Scenarios (case 1: an extreme sample, case 2: no information in file)

NOTES: The regulators obtain the linear approximation using the LIME model in the proximity of an extreme sample. In both cases, the LIME approximation is cloae8 (i)-5.8 ((i)-6e27yi)-3.6.3 (r7eE m)-13ra6 (e)1(o)-0.p(o)-o9.2u.7 (a6 (e)1(0)-8.F1 (e□-4.9 (l)-7)-35.6 (y o)-738 (f a)-9.6

models in the literature and make appropriate modi□cations to the existing models to generate realistic instructions for reverse classi□cation.

Applicants are interested in counterfactuals that provide information about the steps that might change the decision of the model. Consequently, the counterfactuals being generated should have the following properties: reverse classi□cation (i.e., the model prediction for the counterfactual should be reversed from the original decision), proximity (i.e., the counterfactual should be close to the original data point), and diversity (i.e., there should be multiple different counterfactuals from which an individual is able to choose).

We generate counterfactuals using the state-of-the-art method, diverse counterfactual explanations (DiCE) (Mothilal, Sharma, and Tan 2020). This method learns the counterfactuals, c_i, for a data point y optimizing over the requirements of the counterfactual. In particular, it learns the c_i's by minimizing the following loss:

((

where the rst part is the reverse classi cation loss (cross entropy for our application), the second part is the proximity to the original data point (average L_1 distance), and the third part is the diversity component (determinant of kernel matrix given by the distance between counterfactuals). λ_1 and λ_2 are the weights for proximity and diversity components, respectively. The exact details of the implementation of different loss function components can be found in Mothilal, Sharma, and Tan (2020).

We study a few illustrative examples and analyze the properties of the counterfactuals generated using DiCE.[13] Counterfactuals can be generated using other algorithms as demonstrated in Gomez et al. (2020). We use DiCE because it implements the diversity functionality while generating counterfactuals. We analyze the importance of diversity next.

Applicants classi ed as noncreditworthy are interested in the steps that can make them creditworthy. The steps suggested must be practical enough for an applicant to implement. Some features are inherently impossible for individuals who are deemed noncreditworthy to improve. For example, the maximum delinquency ever in days cannot be decreased: It is an event that happened in the past and cannot be changed. Likewise, the total number of lines of credit established cannot be increased because the person has been turned down from opening a new line of credit.

In our system, the features that are impossible to modify are incorporated as constraints in the optimization of loss L. Exhibit 8 shows an example of the set of suggestions to become creditworthy for an individual who is deemed noncreditworthy. All of the suggestions are possible to implement, though some might be dif cult. In (e)-2.2.5 (a)0 73 BDC 0 59 (f)-42

lines of credit), along with decreasing the number of lines of credit requests made in the six months prior to applying for a new line of credit. Similarly, diversity-constrained counterfactual suggestions can be made for all individuals who are deemed noncreditworthy, and individuals can follow the steps that are most convenient to their circumstances.

Likewise, the people who are classified as creditworthy would like to maintain their creditworthy status. Hence, they would like to know which actions to avoid that might make them noncreditworthy. Exhibit 9 presents one such example. The applicant in Exhibit 9 might face a credit denial if they take steps such as opening up new lines of credit, decreasing the lines of credit that are not currently delinquent, increasing the fraction of revolving lines of credit, and making new requests for a line of credit in the six months prior to applying for another line of credit. These steps result in the noncreditworthy probability, as suggested by the model increasing from 0.09 to 0.58, which may lead to a denial of credit. From these counterfactuals, the applicant thus knows the steps not to take that otherwise may decrease their creditworthiness.

To gauge the ability of DiCE to generate counterfactuals, we run it for all test data points (total: 1,973)

researcher an idea about any possible problems with the model. It can additionally help them to present a summary of the model to managers or regulators. In this section, we discuss methods that may help data scientists to demystify these black-box models for their particular needs. The key contribution in this section includes applying data-driven ILP and decision trees to generate a dataset summary, which is not discussed in earlier sections.

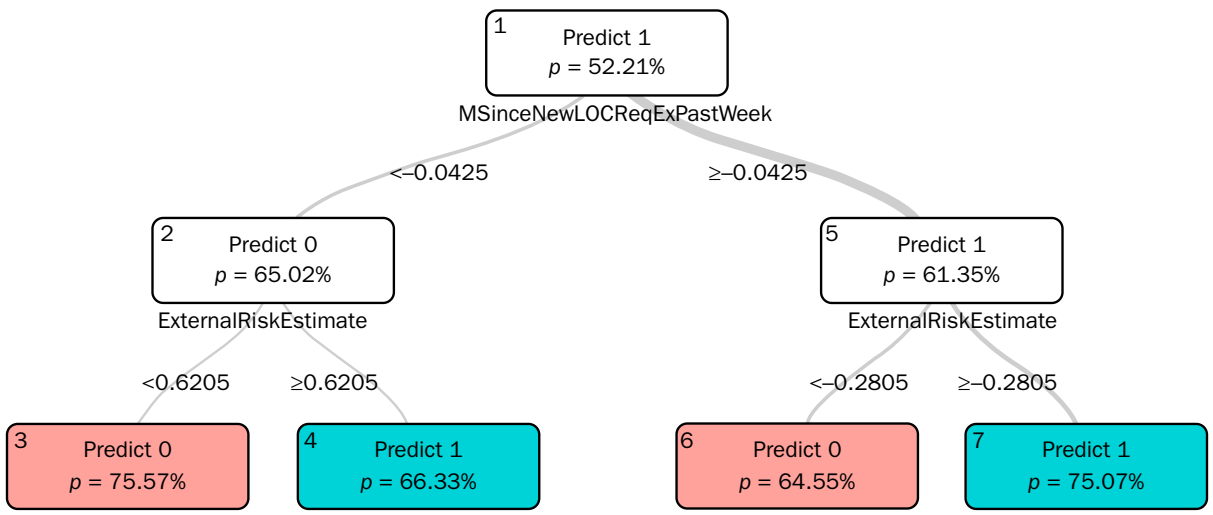
We have already discussed the first aspect of interpretability, the summarization of the model. A good summary of the model includes determining the most important features that contribute to the model's prediction. Using the methods described in the previous sections (LIME and SHAP), data scientists can obtain the most important features and know their contribution to the model's overall prediction. For example, Exhibit 5 illustrates that the feature "months since the newest request for a new line of credit (excluding those requested in the past week)" is the most important feature in the model. The model summary also includes information about the relationship between the input and output of the model. Additionally, from Exhibit 5, Panel A, we can observe that higher values of the fraction of all revolving line of credit limits in use increase the noncreditworthy probability.

Another important aspect for researchers is the summarization of the dataset. It involves discovering the intrinsic relationships within the dataset. To this end, we aim to learn a set of simple rules that can summarize the dataset. Tree-based classification approaches can be used to learn such rules. A tree classifier is a set of if–else statements that determines the classification of a data point. As illustrated in Exhibit 1, the best optimal tree classifier with our dataset achieves an accuracy of 74.12%. However, the complexity of rules obtained from the best optimal trees classifier makes these rules difficult to analyze and interpret.

To obtain rules that are simple to understand, the decision tree must be constrained to a smaller depth, with fewer features at every node. Consequently, we use a simpler and interpretable optimal tree to obtain more easily analyzed rules. This simple optimal tree is shown in Exhibit 11. Here, we show two simple rules that can achieve an accuracy of 72.28%. The rules state that a person will default on a loan

EXHIBIT 11

Explainable Optimal Tree Model Used to Learn Simple Rules Summarizing the Dataset



NOTE: The rules state that a person will default on a loan (i.e., be classified as noncreditworthy) if the number of months since a new line of credit has been requested (excluding the past week) is greater than 1 and the external risk estimate is less than 75, or the number of months since a new line of credit has been requested (excluding the past week) is less than or equal to 1 and the external risk estimate is less than 68.

(i.e., be classified as noncreditworthy) if the number of months since a new line of credit has been requested (excluding those requested in the past week) is greater than 1, and the external risk estimate is less than 75, or if the number of months since a new line of credit has been requested (excluding those requested in the past week) is less than or equal to 1, and the external risk estimate is less than 68.

We use ILP to learn an even simpler set of rules with a small decrease in accuracy compared to the rules generated from the optimal trees model. As discussed earlier, ILP generates rules that are composed of a single condition; for example, using the rule *ExternalRiskEstimate* < 72 to classify people into noncreditworthy and creditworthy groups can achieve an accuracy of 70.65%.

In addition to ILP, there exist multiple other rule-finding methods in the literature. The ILP approach, however, stands out in terms of its simplicity and ability to learn more effective rules. For example, anchors (Ribeiro, Singh, and Guestrin 2018) can be used for generating rules based on data points that are in close proximity. The logistic rule regression/generalized linear rule model described in Wei et al. (2019) is another rule-based model, but it leads to multiple rules that are difficult to analyze together. Similarly, the Boolean rule column generation method described in Dash, Günlük, and Wei (2018) gives a set of rules to describe a dataset, but it requires users to divide feature values into different bins before learning the rules. Data binning limits the quality of the generated rules because the binning algorithm may not select the optimal thresholds for the rule.

Using the methods discussed earlier, data scientists should be better able to summarize datasets and demystify models. Obtaining a good summary of the dataset and an understanding of the model will help in the creation of better classifiers.

CONCLUSION

In this article, we examine several different ML models and use suitable tools to create explanations of their respective functions according to the needs of different stakeholders involved in credit risk management. We use state-of-the-art interpretable ML techniques, including LIME, SHAP, and DiCE, and adapt them to our use case. We demonstrate the importance of domain-specific knowledge in order to explain these black-box models. These domain-specific constraints must be obtained from experts in the field, and they can produce pragmatically valid suggestions and explanations.

Our results demonstrate that, with the right tools, even black-box ML models are able to answer a series of important questions for credit risk modeling. These questions include: Why does the model classify a data point in a certain way? What small changes in feature values could reverse the model's classification of an individual? How does the model behave in extreme scenarios? What relationships did the model learn? Are the models biased? What is the minimal summary of the dataset?

Answering these questions not only fulfills the legal requirements specified by regulators for the use of ML models in credit risk management but also provides borrowers, lenders, and data scientists with answers to questions that they may desire from ML models. Improvements in the interpretability of ML models can not only accelerate their adoption but also help domain experts troubleshoot their inner workings, which in turn drives the model to iterate toward higher accuracy and be tailored to its domain.

Many problems in finance and economics have a common mathematical representation and internal statistical structure and may therefore benefit from our framework to interpret the black-box ML models used to analyze them. These include

loan defaults, mortgage prepayments, Federal Reserve rate decisions, corporate merger and acquisition decisions, asset return maximization, and insurance claims, among others. Our study is a step in the direction of bridging the gap between these black-box models and their use in a real-world setting.

Future work can extend our explainability analysis by incorporating second-order effects into the explainability algorithms (LIME and DiCE) in order to generate more practical counterfactual suggestions and explanations. It may also be of interest to compare the insights derived from these explainability tools to the traditional factors used for credit risk forecasting by involving an expert in the process. Finally, large real-world datasets, in particular those including macroeconomic and demographic features and the size of loan requests, should be used to extend the model's capabilities, in order to help quantify its overall fairness, response to stress testing, and the monetary impact due to the superior performance of black-box ML models.

APPENDIX

DATASET DETAILS

Glossary of Relevant Credit Modeling Terms

The following de□nitions provide helpful context for the description of the dataset's features.

1. Line of credit: An agreement to provide credit
2. Revolving line of credit: A line of credit with a maximum amount that the borrower can choose to use each month. The most common example is a credit card.
3. Installment line of credit: A line of credit with a □xed loan amount and a □xed monthly payment. A mortgage is a common example.
4. Delinquent: A line of credit is delinquent if its payments are not made in a timely manner.
5. Utilization: The amount still owed divided by the total amount borrowed; the fraction of available credit currently in use.

Explanation of Predictor Features

In addition to the binary target variable (risk classi□cation), each credit applicant is characterized by 23 predictor features, 21 continuous and 2 categorical. These consist of the following:

1. A condensed version of the borrower's credit risk computed by FICO using all credit bureau information (ExternalRiskEstimate)
2. Months since the very □rst line of credit was established (MSinceFirstLOC)
3. Months since the newest line of credit was established (MSinceNewestLOC)
4. Average age in months of all existing lines of credit (AvgAgeOfLOC)
5. Number of lines of credit not currently delinquent (NumLOCNotDelq)
6. Number of lines of credit that have ever been 60 or more days delinquent (NumLOC60PlusDaysDelq)
7. Number of lines of credit that have ever been 90 or more days delinquent (NumLOC90PlusDaysDelq)
8. Percentage of lines of credit that have never been delinquent (PercentLOCNeverDelq)
9. Number of months since the most recent delinquency (MSinceMRecentDelq)
10. Maximum delinquency in days in the past year (MaxDelqLast12M)

11. Maximum delinquency ever in days (MaxDelqEver)
12. Total number of lines of credit established (NumTotalLOC)
13. Number of lines of credit established in the past year (NumLOCInLast12M)
14. Percentage of lines of credit that are installment lines of credit (PercentInstLOC)
15. Months since the newest request for a new line of credit excluding those requested in the past week (MSinceNewLOCReqExPastWeek)
16. Number of requests for new lines of credit in the last six months (NumLOCReqLast6M)
17. Number of requests for new lines of credit in the last six months excluding those requested in the past week (NumLOCReqLast6MExPastWeek)
18. Fraction of all revolving credit limits in use (FracRevLOCLimitUse)
19. Fraction of all installment lines of credit in use (FracInstLOCUse)
20. Number of revolving lines of credit with outstanding balances (NumRevLOCWBalance)
21. Number of installment lines of credit with outstanding balances (NumInstLOCWBalance)
22. Number of bank loans and national loans (a subset of all revolving trades) with an outstanding balance of at least 75% of the credit limit (NumBank/NatlLoansWHighUtil)
23. Percentage of lines of credit with outstanding balances (PercentLOCWBalance)

Data Cleaning

Our dataset contains special values, negative integers that are interpreted symbolically and do not hold any numeric significance. As a result, we cannot directly feed them into our ML models. We either have to drop them or encode them appropriately. A large fraction of the data points in our HELOC dataset contains at least one special value (7,957 of the 10,459 data points). Hence, dropping all such data points is infeasible.

In addition, the dataset has 588 records that solely contain the special value −9 for all feature values. Three hundred thirty-one of these data points are labeled as noncreditworthy and 266 as creditworthy. This is a problem for any model because the same input vector will produce opposite target labels. This happens because a borrower will receive a special value if they do not need to be investigated or if they have no bureau record at all, that is, they have no credit history. Such data points are dropped in our analysis.

We find special values are concentrated in 9 of our 23 input features. A standard technique for dealing with special values is to replace them with the mean values of the respective feature. A simple example illustrates that this is not a meaningful approach for our dataset. If a borrower has never had a delinquency, they will have the −7 (condition not met) special value for the feature "months since most recent delinquency." Clearly, replacing the feature value with the mean is not correct because they will be moved from a desirable value of the feature to a less desirable one. To handle these special values, we used binning techniques, which are described in the next section.

Data Preprocessing

The records in our dataset contain special values, which require a careful approach. We deal with special values by discretizing continuous features into bins. The advantage of binning is that special values can be treated as a separate bin, and any outliers can be consolidated. Once a binning schema has been decided, a feature can be represented using one-hot encoding and WoE encoding.

In one-hot encoding, a feature that contains n bins can be represented as an n-dimensional vector f. If a feature value belongs to bin_i then the value of its k dimension, $f_k = 1$ if $k = i$ and 0 otherwise, for $k \in [0, n)$. The drawbacks of one-hot encoding are that bins are treated as unordered categories, and sparsity is introduced. Sparse features

can result in overfitting and biased parameters in a model if the training dataset is small. In addition, for continuous variables, there is no clearcut formula to define the binning schema, and choosing it manually can be suboptimal.

WoE encoding is a popular statistical technique used in the credit rating industry (Siddiqi 2012). It is used to automatically recode the values of continuous and categorical predictor variables into discrete bins and to assign each bin a WoE value. The bins are determined such that they will produce the largest differences with respect to the WoE values. Additionally, monotonicity constraints can be specified to ensure that WoE values are strictly increasing or decreasing in feature values.

The formula for WoE encoding is derived from entropy theory and the information value. For bin_i, the WoE value can be computed as follows:

$$\text{WoE}_i = \left[\ln\left(\frac{\text{Relative frequency of goods}}{\text{Relative frequency of bads}}\right)\right] \cdot 100$$

where the relative frequency of goods is defined as the ratio of the number of creditworthy individuals in bin_i to the total number of creditworthy individuals, and the relative frequency of bads is defined as the ratio of the number of noncreditworthy individuals in bin_i to the total number of noncreditworthy individuals.

Intuitively, the WoE value of a bin provides a measure of its predictive ability to separate creditworthy and noncreditworthy applicants. An important benefit of WoE encoding is that it can be used to treat missing values and outliers without introducing sparsity. As WoE values are on the same scale, we can use them to compare the univariate effects of bins on the target variable within a feature or across all features. Its drawback, like most binning techniques, is that it results in a loss of information. However, models trained on WoE encoded data have a better performance than other methods.

Data Visualization

Before using the dataset to train ML models, we analyze its properties using a few exploratory data visualization techniques. Exhibit A2 visualizes the 23×23 correlation matrix of our dataset, identifying higher correlation values with lighter shades. We find three pairs of features with correlations greater than 0.8.

1. The total number of lines of credit (NumTotalLOC) and the number of lines of credit that are not currently delinquent (NumLOCNotDelq)
2. The number of lines of credit that have been 60+ days delinquent (NumLOC60PlusDaysDelq) and the number of lines of credit that have been 90+ days delinquent (NumLOC90PlusDaysDelq)
3. The number of requests for new lines of credit in the past six months (NumLO

The optimal tree (black box) is the best-performing model of all the parameter combinations. Its parameters are depth = 2 and the number of features = 10. The optimal tree (interpretable) model corresponds to a model depth = 2 and a number of features = 1.

We used the scikit-learn implementation of the random forest classifier.[15] A grid search was performed on the number of estimators (trees) from 1 to 150. The best-performing model consisted of 140 trees.

The NN (WoE) model was implemented using the

ACKNOWLEDGMENTS

We thank the participants of the 2022 North American Winter Meeting of the Econometric Society (January 7–9), the Retail Credit Research Seminar at the Federal Reserve Bank of Philadelphia, and in particular, Ansgar Walther (discussant) and Mao Ye (session chair) for discussion and helpful comments. We also thank Jayna Cummings for editorial assistance. Ruixun Zhang's research is supported by the National Key R&D Program of China (2022YFA1007900), the National

Chapman, J. M. 1940. "Factors Affecting Credit Risk in Personal Lending." In *Commercial Banks and Consumer Instalment Credit*, pp. 109–139. Cambridge, MA: National Bureau of Economic Research.

Chen, S., Z. Guo, and X. Zhao. 2021. "Predicting Mortgage Early Delinquency with Machine Learning Methods." *European Journal of Operational Research* 290 (1): 358–372.

Chen, C., K. Lin, C. Rudin, Y. Shaposhnik, S. Wang, and T. Wang. 2022. "A Holistic Approach to Interpretability in Financial Lending: Models, Visualizations, and Summary-Explanations." *Decision Support Systems* 152: 113647.

Croxson, K., P. Bracke, and C. Jung. "Explaining Why the Computer Says 'No.'" Research paper, FCA, 2019.

Cunningham, D. F., and P. H. Hendershott. "Pricing FHA Mortgage Default Insurance." Technical report, National Bureau of Economic Research, 1984.

Dash, S., O. Günlük, and D. Wei. 2018. "Boolean Decision Rules via Column Generation." *arXiv* 1805.09901.

Deng, Y., J. M. Quigley, and R. Van Order. 2000. "Mortgage Terminations, Heterogeneity and the Exercise of Mortgage Options." *Econometrica* 68 (2): 275–307.

Dim, C., K. Koerner, M. Wolski, and S. Zwart. "Hot Off the Press: News-Implied Sovereign Default Risk." Technical report, EIB Working Papers, 2022.

Duan, J. 2019. "Financial System Modeling Using Deep Neural Networks (DNNS) for Effective Risk Assessment and Prediction." *Journal of the Franklin Institute* 356 (8): 4716–4731.

Dumitrescu, E., S. Hué, C. Hurlin, and S. Tokpavi. "Machine Learning or Econometrics for Credit Scoring: Let's Get the Best of Both Worlds." Working paper hal-02507499, HAL, 2021.

Elul, R., N. S. Souleles, S. Chomsisengphet, D. Glennon, and R. Hunt. 2010. "What 'Triggers' Mortgage Default?" *American Economic Review* 100 (2): 490–494.

Foote, C., K. Gerardi, L. Goette, and P. Willen. 2010. "Reducing Foreclosures: No Easy Answers." *NBER Macroeconomics Annual* 24 (1): 89–138.

Fuster, A., P. Goldsmith-Pinkham, T. Ramadorai, and A. Walther. 2022. "Predictably Unequal? The Effects of Machine Learning on Credit Markets." *The Journal of Finance* 77 (1): 5–47.

Giudici, P., M. Mezzetti, and P. Muliere. 2003. "Mixtures of Products of Dirichlet Processes for Variable Selection in Survival Analysis." *Journal of Statistical Planning and Inference* 111 (1–2): 101–115.

Giudici, P., B. H. Misheva, and A. Spelta. 2020. "Network Based Credit Risk Models." *Quality Engineering* 32 (2): 199–211.

Giudici, P., and E. Raf□netti. 2021. "Shapley–Lorenz Explainable Arti□cial Intelligence." *Expert Systems with Applications* 167: 114104.

Gogas, P., and T. Papadimitriou. 2021. "Machine Learning in Economics and Finance." *Computational Economics* 57 (1): 1–4.

Gomez, O., S. Holter, J. Yuan, and E. Bertini. 2020. "ViCE: Visual Counterfactual Explanations for Machine Learning Models." In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pp. 531–535. New York, NY: Association for Computing Machinery.

Goodman, B., and S. Flaxman. 2017. "European Union Regulations on Algorithmic Decision-Making and a 'Right to Explanation'" *AI Magazine* 38 (3): 50–57.

Green, J. R., and J. B. Shoven. "The Effects of Interest Rates on Mortgage Prepayments." Working paper 1246, National Bureau of Economic Research, 1983.

Jiang, J., L. Liao, X. Lu, Z. Wang, and H. Xiang. 2021. “Deciphering Big Data in Consumer Credit Evaluation.”

Siddiqi, N. 2012. *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*, vol. 3. Hoboken, NJ: John Wiley & Sons.

Sirignano, J., and K. Giesecke. 2019. "Risk Analysis for Large Pools of Loans." *Management Science* 65 (1): 107–121.

Stanton, R., and N. Wallace. 2011. "The Bear's Lair: Index Credit Default Swaps and the Subprime Mortgage Crisis." *The Review of Financial Studies* 24 (10): 3250–3280.

Tantri, P. 2021. "Fintech for the Poor: Financial Intermediation without Discrimination." *Review of Finance* 25 (2): 561–593.

Thomas, L. C. 2000. "A Survey of Credit and Behavioural Scoring: Forecasting Financial Risk of Lending to Consumers." *International Journal of Forecasting* 16 (2): 149–172.

Wei, D., S. Dash, T. Gao, and O. Gunluk. 2019. "Generalized Linear Rule Models." In *Proceedings of the 36th International Conference on Machine Learning*, edited by K. Chaudhuri and R. Salakhutdinov, pp. 6687–6696, Proceedings of Machine Learning Research, Volume 97. Cambridge, MA: PMLR.