Explainable Machine Learning Models of Consumer Credit Risk[*]

Randall Davis,[†] Andrew W. Lo,[‡] Sudhanshu Mishra,[§] Arash Nourian,[¶] Manish Singh,[‖] Nicholas Wu,[§] Ruixun Zhang[§]

January 12, 2022

Abstract

In this paper, we 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. We analyze the explainability of these models for various stakeholders: loan companies, regulators, loan applicants, and data scientists, incorporating their different requirements with respect to explanations. For loan companies, we generate explanations for every model prediction of creditworthiness. For regulators, we perform a stress test for extreme scenarios. For loan applicants, we generate diverse counterfactuals to guide them with steps to reverse the model's classification. Finally, for data scientists, we generate simple rules that accurately explain 70-72% of the dataset. Our work is intended to accelerate the adoption of ML techniques in domains that would benefit from explanations of their predictions.

Keywords: Machine Learning; Interpretability; Explainable AI; Credit Lending; Inductive Logic Programming; Optimal Trees; LIME; SHAP; Counterfactual.

JEL codes: C55; C45; G0; G2; G51; C69.

[*]Research support from the MIT Laboratory for Financial Engineering is gratefully acknowledged. We thank the participants of the 2022 North American Winter Meeting of the Econometric Society (January 7-9), discussant, Ansgar Walther, and session chair, Mao Ye, for discussion and helpful suggestion.

[†]MIT Computer Science and AI Laboratory;

[‡]Corresponding author; MIT Sloan School of Management; MIT Laboratory for Financial Engineering; MIT Computer Science and Artificial Intelligence Laboratory; 100 Main Street, E62–618, Cambridge, MA 02142, alo-admin@mit.edu.

[§]MIT Laboratory for Financial Engineering

[¶]Fair Isaac Corporation

[‖]smanish@mit.edu, MIT Computer Science and AI Laboratory; MIT Laboratory for Financial Engineering

1 Introduction

The total U.S. household debt at the end of the fourth quarter of 2020 is estimated to be \$14.56 trillion, with 189.6 million new credit accounts opened within the prior 12 months.[1] Given this massive amount of household debt, even a low delinquency rate can significantly affect the operation of the financial system. This potential impact makes the study of credit default risk an important real-world classification 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 gauge the credit-worthiness of individuals have been described in the literature, which extends as far back as the 1940s (Chapman, 1940).

Recent advances in computing, innovative algorithms, and an explosion in the quantity of data have contributed to the growing success of complex nonlinear machine learning models, sometimes known as “deep” learning models. Unlike traditional machine learning 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 machine learning methods. ML models have been explored for diverse applications in economics and finance (Gogas and Papadimitriou, 2021). 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 (Sirignano, Sadhwani, and Giesecke, 2016; Kvamme et al., 2018; Chen, Guo, and Zhao, 2021), credit card risk (Butaru et al., 2016), consumer credit risk (Jiang et al., 2021), and fair credit allocation (Tantri, 2021).

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 (2018) created a right to explanation, whereby a user may ask for an explanation of an algorithmic decision that was made about them (Goodman and Flaxman, 2017).

[1]Source: New York Fed Consumer Credit Panel/Equifax

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

In this work, we first model a credit risk forecast on a real-world home equity line of credit (HELOC) dataset released by the Fair Isaac Corporation (FICO). We developed a wide range of ML models, including interpretable rule-based models (e.g., inductive logic programming and optimal trees) and black-box ML models (e.g., neural networks and random forests). We compare these models and find that neural networks outperform other linear and nonlinear models, reaching 74.75% accuracy. We also find that simple rules can explain 70-72% of the dataset.

The "explainability" of a model means the ability to give answers to the different stakeholders involved in the decision-making process (Croxson, Bracke, and Jung, 2019). In this work, we identify the different stakeholders involved in credit risk management—loan companies, regulators, loan applicants, and data scientists—and provide explanations of the models according to their needs.

For loan companies, we generate interpretable explanations for every prediction using two post hoc explainability tools, Local Interpretable Model-Agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP), that are able to provide reasons for a loan denial. We modify the methodology of the LIME algorithm to incorporate the constraints on our data. The ability to generate explanations makes a model transparent about its functionality, in this case, informing loan companies about the relationships it has learned. For example, we identify that having a higher average age of line of credit leads to a decrease in the probability of default. These explanations also assist in finding representative borrowers from the past, for example, to find an accurate applicant from past to aid loan officers.

For regulators, we suggest a method of investigating potential regarding fairness of these models. Regulators may also be involved in investigating the model's functionality in extreme scenarios. In anticipation of this issue, we provide a methodology to perform stress tests to help the regulators analyze the model's behavior in extreme scenarios.

Since loan applicants are interested in getting approval for a loan, we generate a diverse set of suggestions that can help individuals deemed non-creditworthy become creditworthy. For individuals who already have been approved for loans, we are able to generate suggestions that can help them remain creditworthy. These counterfactual suggestions are generated by incorporating constraints based on specific items of domain knowledge, which makes these suggestions more usable in practice. We generated suggestions successfully for 99% of the individuals in our testing dataset.

Finally, for data scientists, we are able to summarize the dataset using a few simple rules that can help them understand the relationships and structure in the dataset. Understanding these relationships may provide a data scientist with the insights to develop better models in the future. Using inductive logic programming, we are able to explain 70% of the dataset using a single rule, and can explain up to 72% of the dataset using two simple rules.

Our work demonstrates that the functionality of black-box ML models can be explained to a range of different stakeholders, if the right tools are applied to the task, unlocking the future potential of applying AI to improve credit modeling. New explainable AI tools will be able to help stakeholders make counterfactual predictions and explain a model's output, leading to the ability to answer many "what-if" questions. This includes stress tests for extreme scenarios, and informing applicants about the factors which will improve their creditworthiness. More generally, it demonstrates the importance and potential of explainable AI to affect traditional fields like credit modeling significantly.

The remainder of this paper is structured as follows. Section 2 discusses the relevant literature. Section 3 describes the different models used in our paper, while Section 4 describes the dataset used in the analysis. Section 5 evaluates and compares different models. The explainability for various stakeholders is discussed in Section 6. Section 7 summarizes our findings and concludes the paper.

2 Literature Review

A range of statistical and operational research methods has been used over the long history of credit scoring models (Thomas, 1999). Most recently, machine learning techniques have been adopted for credit risk models. Lessmann et al. (2015), Thomas (1999), Breeden (2020), and Gogas and Papadimitriou (2021) give detailed surveys of ML methods for credit risk forecasting. However, Gogas and Papadimitriou (2021) also highlight the shortcomings of ML with regard to the explainability of the models for fintech applications.

An extensive literature has been developed about explainable AI for models in healthcare, computer vision, and natural language processing (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. (2018) use an interpretable two-layer additive risk model for a home equity line of credit dataset. A two-layer additive risk model is the 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. In this paper, we show that a neural network outperforms the two-layer additive risk model proposed in Chen et al. (2018).

Another category of solutions includes post hoc methods. Bussmann et al. (2020) use Shapley values to explain tree-based ensemble models and apply correlation networks to group the borrowing companies using the derived explanations. Similarly, Hadji Misheva et al. (2021) use the post hoc methods of Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (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 neural networks to predict consumer default, and use SHAP to provide model explanations. Other methods that use SHAP for explaining credit risk models include Ariza-Garzón et al. (2020) and 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. Other methods of post hoc interpretability include layerwise relevance propagation and activation analysis of hidden units of a neural network (Ponomareva and Caenazzo, 2019). Rudin and Shaposhnik (2019) propose a minimum set cover problem to generate a rule-based summary of a machine learning model on the home equity line of credit dataset. Similarly, Martens et al. (2007) propose a rule extraction method from a trained support-vector machine (SVM) model for credit scoring.

While these methods generate explanations of model outcomes, they are not tailored specifically for the different stakeholders involved in the credit risk pipeline. 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[2], but it remains incomplete, and does not use the state-of-the-art methods that we employ in this work.

In this paper, we present a methodology of generating explanations of black-box models for different stakeholders, a direction that has not been well explored previously. 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.

3 Machine Learning 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 non-creditworthy. A creditworthy applicant is likely to repay their financial obligation, while a non-creditworthy 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, $x_1, x_2, ...x_k$, for a borrower i, the task is to predict a variable y_i, that is, the probability of default, $Pr(Default)$. The features x_i describes 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.

[2] `https://aix360.mybluemix.net/data`

We frame the classification as a supervised learning problem in which we train a function f that can approximate the relationship $y_i = f(x_1, x_2..x_k)$. To train the function f, we use variety of ML models, including optimal classification trees, random forests, inductive logic programming, and neural networks. We describe these models next.

3.1 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 non-expert classifiers which 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.

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.

3.2 Inductive Logic Programming

Inductive logic programming (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.

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, a $N \times R \times 2n$ tensor, and interpret $\text{softmax}(\Pi[i, j])$ 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. However, these operators 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, $\widehat{\vee}(x) = 1 - A(1 - x)$, where A is the continuous extension of $\hat{\wedge}$. 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^*:

$$\hat{y} = \widehat{\bigvee_i} \widehat{\bigwedge_j} (X^* \cdot \text{softmax}(\Pi[i, j]))$$

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:

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

In our work, 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.

Using the inductive logic programming model, we can thus learn simple interpretable rules that can be used for classification and summarizing datasets.

3.3 Optimal Classification Trees

Decision trees are constructed in a top-down greedy way using the CART algorithm (Breiman et al., 1984) as described in Section 3.1. 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. However, optimal trees 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 creation of 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.

3.4 Neural Networks

Neural networks (NN) are supervised learning models loosely inspired by the biological networks of the human brain.

A 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 neural network. It is worth noting that logistic regression is a special case of a 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:

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

Here, w_{ij} is the weight from input x_i to hidden unit h_j. The weights w_{ij} can be learned by minimizing the loss function using optimizers like stochastic gradient descent.

The weights of the neural network create complex and nonlinear interactions between input features, which do not have human-interpretable meanings. However, the complex nonlinear interactions learned by NNs lead in general to better model performance. Moreover, the decision boundaries learned by NNs can be both high-dimensional and extremely nonlinear. Thus, learned features may have varying significance at different points of the feature space.

4 Data

For our task, we used a home equity line of credit (HELOC) dataset provided by FICO. A HELOC is a revolving loan in which the collateral is the borrower's equity in their house. Like a credit card, a HELOC is available for a set time frame during which a borrower can withdraw money as needed.

The FICO dataset contains 10,459 borrowers who were granted HELOCs during a two-year application window from March 2000 to March 2002. In March 2003, a year after the application window had closed, a performance snapshot was captured, and the risks of the borrowers were evaluated. In this dataset, an applicant might have used their HELOC for a duration between one and three years, depending on the time of their approval. The dataset contained 5,459 non-creditworthy records, while the remaining 5,000 records were deemed creditworthy. In addition to the binary target variable of credit risk classification, each credit applicant is characterized by 23 predictor features, 21 continuous and 2 categorical. Further information about these features, and important terms relevant to the dataset, are provided in Appendix B.2.

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 weight of evidence (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^{th} 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 clear-cut formula to define the binning schema, and choosing it manually can be sub-optimal.

Weight of evidence (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:

$$WoE_i = [\ln(\frac{\text{Relative Frequency of Goods}}{\text{Relative Frequency of Bads}})] * 100 \tag{1}$$

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 non-creditworthy individuals in bin_i to the total number of non-creditworthy individuals.

Intuitively, the WoE value of a bin provides a measure of its predictive ability to separate creditworthy and non-creditworthy 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. As we will see in Section 5, models trained on WoE encoded data have a better performance than other methods.

5 Evaluation

In this section, we evaluate the different models described in Section 3. These machine learning models, including optimal trees, random forests, and neural networks, are trained by recoding all the features using weight of evidence encoding. An inductive logic programming model is trained using the methodology described in section 3.2. We also evaluate the neural network model trained on one-hot encoding version of the dataset. The details of these implementations and best-performing models are included in Appendix C for reproducibility.

We include the two-layer additive risk model in our evaluation (Chen et al., 2018), which was the best performing model of the FICO Data Challenge[3]. It partitions features into different subgroups, combining scores from different subgroups into a global model score. Its feature subgroups are generally interpretable, as they are created through the intervention of an expert in the field. The model resembles a two-layer sparse neural network. 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 $K = 5$, i.e., 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 Table C.1 in the appendix.

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 Table

[3]http://dukedatasciencefico.cs.duke.edu/

C.1, 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 Table 1 illustrates, we find that the NN model trained on WoE data 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 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) that 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 (1 or 2) of simple rules from inductive logic programming 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 NN model, we will see in Section 6 that they are useful for explainability for the different stakeholders involved.

Using ILP, we obtain a single simple rule, that is, "if $ExternalRiskEstimate$ is smaller than 72, then classify borrowers as non-creditworthy," achieves an accuracy of 70.65%. $ExternalRiskEstimate$ is a condensed version of the borrower's credit risk, a metric similar to the FICO score. 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 machine learning models using different features about a borrower's credit history can increase the accuracy of the delinquency forecast by 4 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 non-defaulters (that is, false negatives), they will suffer losses. Similarly, if credit companies deny loans to borrowers who can repay (that is, false positives), they will lose business. Given the enormous size of the mortgage business in the United States, even a small increase in model performance can create a substantial impact. 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.

Models	Test Accuracy	Test AUC	False Positive Rate	False Negative Rate
ILP (1 rule)	70.65	70.60	30.37	28.41
ILP (2 rules)	71.01	70.84	33.02	25.28
Optimal Trees (Interpretable)	72.28	74.18	28.25	27.21
Optimal Trees (Black Box)	74.12	74.46	29.16	22.85
Random Forest (140 Trees)	73.77	79.82	29.71	23.01
Two Layer Additive Risk	74.12	81.01	30.31	**21.77**
NN (one-hot)	74.10	80.59	**25.16**	26.60
NN (WoE)	**74.75**	**81.40**	28.31	22.42

Table 1: 5-Fold cross-validation performance of ML models. The NN trained on WoE data performs the best with a mean test accuracy of 74.7% and AUC of 81.4%. A higher AUC implies the NN model is better able to distinguish between the creditworthy and non-creditworthy 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, while the two-layer additive risk model has the minimum false negative rate when the threshold used to classify the model's output is 0.5. All the reported values are in percentages.

6 Explaining Machine Learning Models

We have shown in Section 5 that the neural network model outperforms every other model tested. However, the non-interpretable nature of neural networks 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 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, while 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.

6.1 Interpretability for Loan Companies: Opening the Black Box

Loan companies use machine learning 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.

LIME

LIME is a model-agnostic technique that approximates the decision boundary of a model at a particular data point using a linear approximation[4]. 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.

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

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

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 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.

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 vs 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 resembles the characteristics of the original dataset.

Constraint Violation. There are 12 constraints relevant to the HELOC dataset (6 relational constraints and 6 value constraints). We determined these via discussion with FICO personnel. Some examples of these constraints are, "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 credits." The 5,000 perturbed data points are scanned for violations of these constraints. The results for all constraints are shown in Table 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, we conclude that we use more realistic data in our modified LIME for generating LIME approximations.

In the remainder of the paper, we use this modified LIME for our analysis. Also, whenever we use the term LIME, we refer to our modified LIME.

Index	Constraint	Original LIME Violations	Modified LIME Violations
1	All feature values interpreted quantitatively must be non-negative	4383	4024
2	PercentLOCNeverDelq ≤ 100	1198	1183
3	PercentInstLOC ≤ 100	1	2
4	PercentLOCWBalance ≤ 100	340	296
5	FracRevLOCLimitUse ≤ 100	62	75
6	FracInstLOCUse ≤ 100	509	506
7	NumLOC90PlusDaysDelq $\leq$ NumLOC60PlusDaysDelq	1656	655
8	NumLOCReqLast6MExPastWeek $\leq$ NumLOCReqLast6M	2095	388
9	NumLOC60PlusDaysDelq $\leq$ NumTotalLOC	191	239
10	NumLOC90PlusDaysDelq $\leq$ NumTotalLOC	192	233
11	NumLOCNotDelq $\leq$ NumTotalLOC	2257	1915
12	NumLOCInLast12M $\leq$ NumTotalLOC	249	255

Table 2: 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.

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

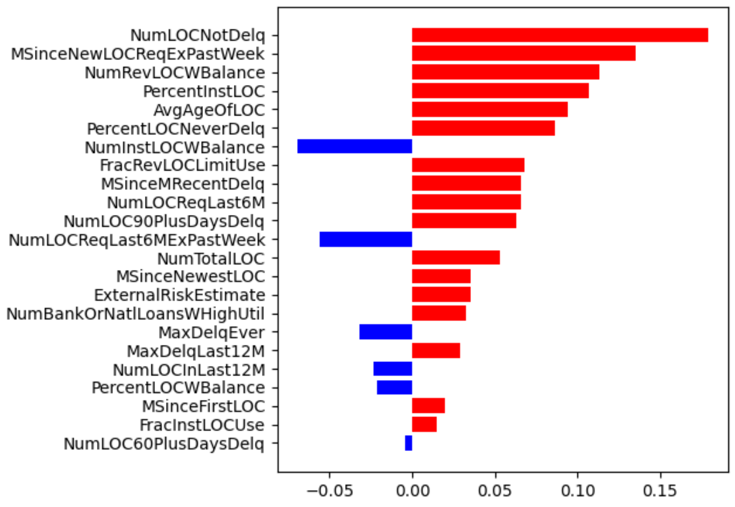
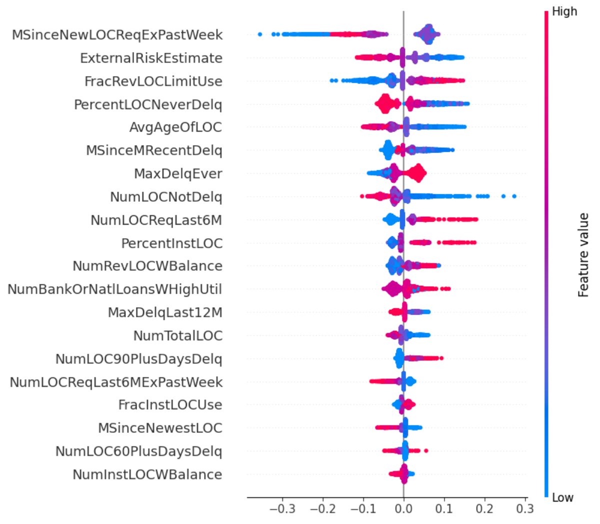


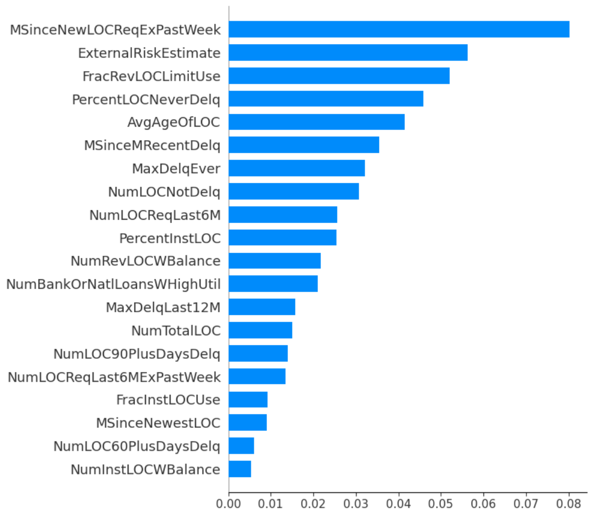
Figure 1: The LIME model approximation for a random data point. A unit increase in features with a red horizontal line will increase the probability of default. Similarly, a unit increase in features with a blue horizontal line will decrease the probability of default. The contribution of each feature in the model explanation can be obtained by multiplying each feature coefficient by the feature value.

in the corresponding feature value, given that other feature values are held constant. Figure 1 shows an example of a LIME explanation. A unit increase in the values of features with coefficients shown in red lines will increase the probability of default, while a unit increase in the values of features with coefficients shown in blue lines will decrease the probability of default.

After obtaining the feature importance for all the data points, they can be aggregated to find the overall feature importance. In Figure 2, 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. From Figure 2a, we can observe the nonlinear nature of the classifier. 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 probability of default, while having values around the median contributes to an increase in the probability of default. Hence, by aggregating the feature importance, we are able to find the relationship learned by the model. These generated explanations and the overall feature importance also aid in discovering biases in the model.



(a) Feature contribution for all sample in test set



(b) Average feature contribution (absolute value) for all samples

Figure 2: Feature importance obtained by aggregating LIME explanations for all samples in the test set. In Figure (a), the color-coding (blue-red) represents the value of the feature. For example, for external risk estimate, the larger values (in red) contribute to the decreasing probability of default, while the smaller values (in blue) contribute to the increasing probability of default. Figure (b) is obtained by aggregating the feature importance as obtained for all points. Figure (b) illustrates that the three foremost 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.

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 1973 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 1973 data points, 1935 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 where the classification of the NN model and LIME do not match and are different significantly, the linear approximation cannot be trusted. In order to generate explanations for these points, we discuss another method.

SHapley Additive exPlanations (SHAP)

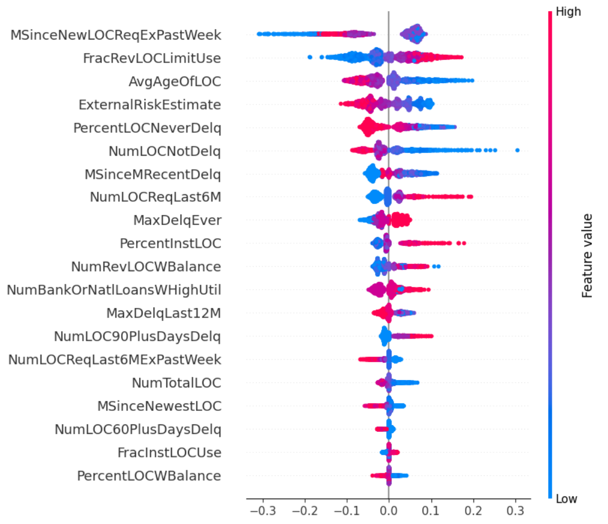
SHAP is an explainable AI method with an economic foundation. It performs the Shapley value decomposition of the model output, giving the contributions of every feature at a data point. Like LIME, it is also a model-agnostic method.

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.

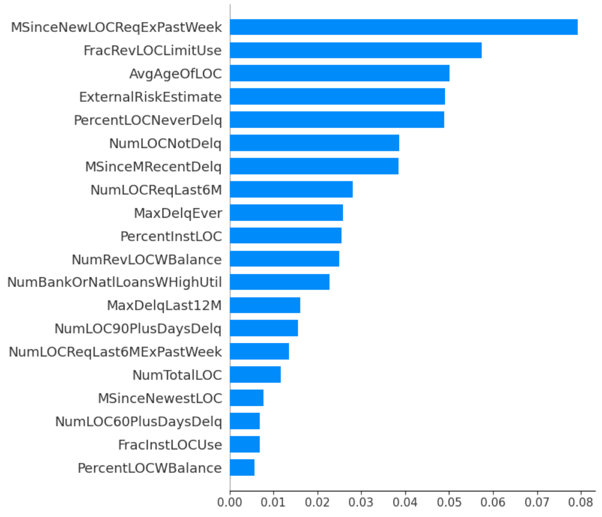
We use the Kernel SHAP implementation from Lundberg and Lee (2017). The Kernel SHAP procedure computes the Shapley values by running a weighted-least-squares regression whose solution is the Shapley values of features. To explain a point x_i, the different points used in the linear regression are obtained by selecting a subset of features from x_i, and the remaining subset of features not selected are replaced with values from background data points (that is, 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 contribution. Using the described weighted-least-squares regression, we obtain Shapley values for the data point 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 the points in the test sample in Figure 3, 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 lines 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 the LIME and SHAP. This may be due to the fact that not all

LIME approximations are locally accurate.



(a) Feature contribution for all samples in test set.



(b) Average feature contribution (absolute value) for all samples.

Figure 3: Feature importance (Shapley values) obtained by aggregating Kernel SHAP explanations for all the samples in the test set. In Figure (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 probability of default, while the smaller values (in blue) contribute to the increasing probability of default. We obtain the overall feature importance (as illustrated in Figure (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 lines 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.

Both LIME and SHAP have their weaknesses. For example, in addition to feature importance, we can obtain locally linear approximations using LIME (as obtained in Figure 1), which are not available in SHAP. These local approximations of the decision boundary are necessary for other applications, as will become clear in the next section. On the other hand, the explanations using LIME might not satisfy important properties like local accuracy and consistency that Shapley values allow.

From the computational perspective, different runs of LIME produce slightly different explanations because of the randomness in sampling local data points. Consequently, it can potentially suffer from instability issues. On the other hand, computing Shapley values is not scalable for datasets with a large number of features. In such scenarios, LIME approximations can be computed efficiently. Therefore, both methods (SHAP and LIME) should be used in practice depending on the use case, carefully and judiciously.

Finding Representative Samples

Another application of model explanations is to find the most representative data point for a particular data point analyzed. It can be useful in scenarios where loan companies might require representative data points from the past to explain the model prediction of a particular loan candidate. For example, the borrower x_i might default in the future because it is similar to x_j and x_k who defaulted in past.

The k-nearest neighbors algorithm (kNN) applied to the data naively can generate the most similar points in the sample. However, if we apply kNN to the original feature space, the less significant features will end up contributing to the distance calculations between the data points. To avoid this problem, every data point can be represented by a feature importance vector (for instance, a vector of Shapley values), and kNN can be applied to the feature importance vector. Using SHAP will ensure that less important features have smaller feature contributions (that is, smaller Shapley values). Hence, the contributions of less important features to the distance calculation in the kNN will be minimal. This modification leads to finding a more representative data point for each new data point being analyzed.

6.2 Interpretability for Regulators: Model Fairness and Stress Testing

In the previous subsection, we generated explanations for model predictions. Apart from concerns about the transparency of the credit approval algorithm, government regulations mandate that these algorithms must be fair. Concerns about the fairness of a model arise when the model gives importance to features like race, religion, or gender that are specified in law. Our HELOC dataset does not contain any such features overtly. Nevertheless, the fairness of a black box model can be examined (if features were present) using the individual and overall feature contributions that are obtained from LIME and SHAP.

Another important aspect of interpretability for regulators is stress-testing the models.

The credit risk models used by companies need to withstand appropriate stress testing. This involves simulating extreme scenarios and analyzing the model's behavior in response to generally extreme macroeconomic conditions.

Extreme macroeconomic conditions cannot be simulated in our model due to its lack of macroeconomic features. However, we demonstrate how the interpretability methods can help in stress testing a black-box model for extreme customers. The same methodology can be generalized for extreme macroeconomic conditions. 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 truncated 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 Table 3.

Regulators are interested in verifying the validity of the model's output and analyzing its behavior. We preprocessed features for these extreme customers using weight of evidence 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 $Pr(default) = 0.53$, which implies that the person may or may not default with almost equal probability. Since this data point is sampled randomly, we cannot comment on the accuracy of the prediction. For case 2, the model's prediction is $Pr(default) = 0.75$, which implies that, when there is no information about the person on file, the person is likely to default, hence preventing lenders from approving credit for such individuals with no information in their credit file.

To understand the behavior of the model in the proximity of extreme points, we use the LIME model approximation. For cases 1 and 2, we present the LIME approximation in Figure 4. In both cases, the LIME model approximation prediction is close to the original model prediction. Hence, we conclude that the approximations can be trusted. This model approximation can be used by regulators to learn more about the model's behavior in an extreme scenario.

In this section, we demonstrated the use of explainable AI methods for generating expla-

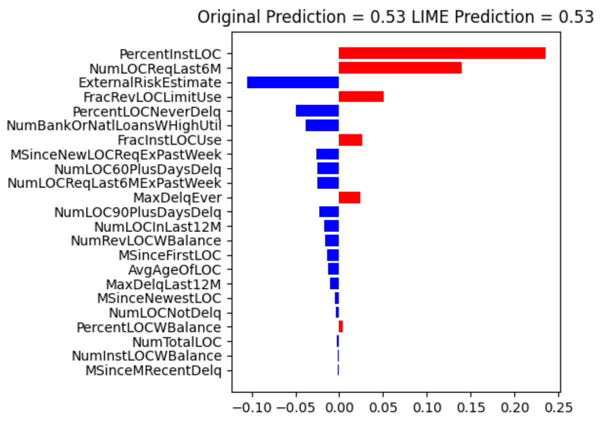
Feature	Data Pt 1	Data Pt 2
ExternalRiskEstimate	131	-9
MSinceFirstLOC	935	-9
MSinceNewestLOC	99	-9
AvgAgeOfLOC	467	-9
NumLOCNotDelq	89	-9
NumLOC60PlusDaysDelq	6	-9
NumLOC90PlusDaysDelq	5	-9
PercentLOCNeverDelq	100	-9
MSinceMRecentDelq	115	-9
MaxDelqLast12M	16	-9
MaxDelqEver	13	-9
NumTotalLOC	125	-9
NumLOCInLast12M	6	-9
PercentInstLOC	88	-9
MSinceNewLOCReqExPastWeek	35	-9
NumLOCReqLast6M	13	-9
NumLOCReqLast6MExPastWeek	12	-9
FracRevLOCLimitUse	71	-9
FracInstLOCUse	100	-9
NumRevLOCWBalance	23	-9
NumInstLOCWBalance	25	-9
NumBankOrNatlLoansWHighUtil	21	-9
PercentLOCWBalance	100	-9
Model Prediction	0.53	0.75

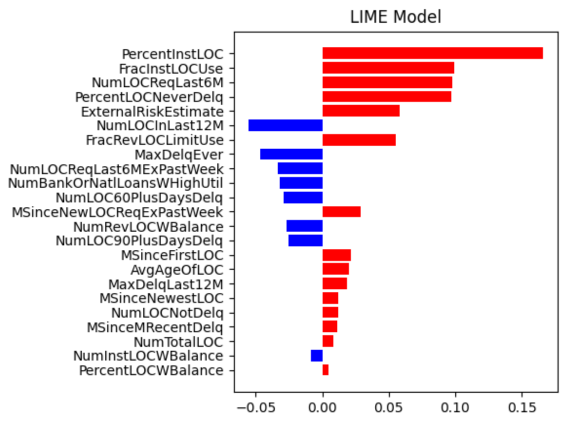
Table 3: Extreme Customers (data points). The first point is sampled, assuming features are multivariate Gaussian. The second point is obtained by setting all feature values to -9 (no bureau record or investigation).

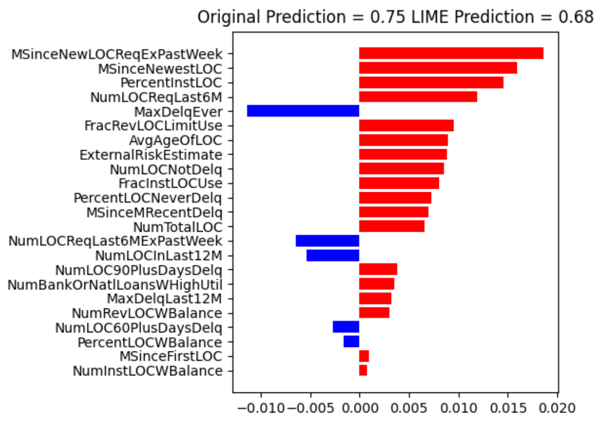
nations in model fairness testing and stress testing that may be able to assist regulators in their duties. Next, we discuss the use of model interpretability for loan applicants.

6.3 Interpretability for Loan Applicants: Counterfactual Suggestions

Loan applicants are interested in two major aspects of model interpretability. The first is to learn the reason behind their denial or approval, and the second is how to modify the features that might change their classification in the future. Explanations for the reason behind a denial or approval can be obtained from the techniques discussed in the previous


(a) Case 1: Feature Contribution.


(b) Case 1: LIME Model


(c) Case 2: Feature Contribution

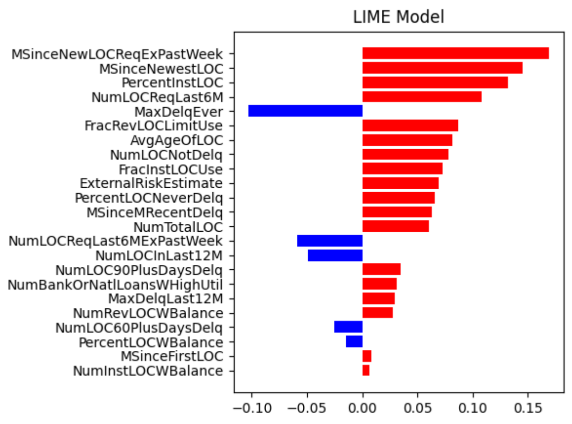

(d) Case 2: LIME Model

Figure 4: LIME explanations for the two extreme scenarios (case 1: an extreme sample, case 2: no information in file). The regulators obtain the linear approximation using the LIME model in the proximity of an extreme sample. In both cases, the LIME approximation is close to the model's output. Hence, we conclude the linear model approximations are reliable. The LIME models are $\Sigma(FeatureValue) * (LIME - Coefficient)$. The LIME coefficients are represented by horizontal lines. We can observe that for extreme case 1, the model gives its maximum importance to the percentage of lines of credit that are installment lines of credit, while for extreme case 2, the model gives its highest importance to the months since the newest request for a new line of credit excluding those requested in the past week. The above analysis can be extended to different extreme conditions.

sections. In this section, we discuss the methodology of generating instructions for reversing the model's classification.

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 classification (that is, the model prediction for the counterfactual should be reversed from the original decision), proximity (that is, the counterfactual should be close to the original data point), and diversity (that is, there should be multiple different counterfactuals from which an individual be 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:

$$L = \sum_{i=1}^{k} loss(f(c_i), y) + \lambda_1 \sum_{i=1}^{k} dist(c_i, y) - \lambda_2 * diversity(c_1, c_2 ... c_k)$$

where the first part is the reverse classification loss, the second part is the proximity to the original data point, and the third part is the diversity component. λ_1 and λ_2 are the weights for proximity and diversity components, respectively.

We study a few illustrative examples and analyze the properties of the counterfactuals generated using DiCE.[5]

Applicants classified as non-creditworthy 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 non-creditworthy 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, since 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. Table 4 shows an example of the set of suggestions to become

[5]We use the publicly available implementation of the DiCE algorithm, found at `https://github.com/interpretml/DiCE`.

creditworthy for an individual who is deemed non-creditworthy. All of the suggestions are possible to implement, though some might be difficult. In cases where it is hard for an individual to implement certain changes, he or she may be able to choose from a diverse set of counterfactuals to reverse the classification.

In Table 4, the first counterfactual suggests increasing the months since the applicant's most recent delinquency to more than four years. However, this requires an individual to wait for four years while making payments for all his lines of credit. In this case, the individual may instead choose to follow another set of steps, as suggested by the third counterfactual: increasing the percent of lines of credit never delinquent (by decreasing the total 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 non-creditworthy, and individuals can follow the steps which 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 non-creditworthy. Table 5 presents one such example. The applicant in Table 5 might face a credit denial if he or she takes 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 line of credit in the six months prior to applying for a line of credit. These steps result in the probability of default 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 may decrease their creditworthiness.

To gauge the ability of DiCE to generate counterfactuals, we run it for all test data points (total: 1973) and count the number of points for which the counterfactuals were successfully found. In Table 6, we show that increasing the proximity constraint leads to a decrease in the average number of feature changes in the counterfactuals obtained. We also observe that increasing the proximity constraint leads to a decrease in the number of successfully generated counterfactuals. An optimal algorithm to generate counterfactuals for an individual thus is to start with a high value for the proximity constraint and slowly decrease it until a counterfactual is found.

Using DiCE and related counterfactual generating methods, we can generate suggestions

Feature	Original Value	New Value
MSinceMRecentDelq	$0-4$	$48-$
NumLOCReqLast6M	$2-4$	$1-2$
$Pr(Default)$	0.63	0.44
AvgAgeOfLOC	$75-98$	$98-$
NumTotalLOC	$9-14$	$0-1$
NumLOCReqLast6M	$2-4$	$1-2$
$Pr(Default)$	0.63	0.43
PercentLOCNeverDelq	$0-82$	$98-$
NumLOCReqLast6M	$2-4$	$1-2$
Pr(Default)	0.63	0.31

Table 4: The steps toward creditworthiness suggested to an individual deemed non-creditworthy. Following these steps, an individual should be able to decrease the probability of default predicted by the machine learning model. We present three diverse counterfactuals. In the first, it suggests changing the months since most recent delinquency from the value of 0-4 months to greater than 48 months, which means that person should not be delinquent for the next 4 years. In the second counterfactual, one of the suggestions is to decrease the total number of lines of credit to 0-1 (by closing accounts). Because some suggestions may be difficult for an individual to follow, we provide multiple diverse counterfactuals. The individual can follow the third counterfactual, which suggests decreasing the number of lines of credits to increase the percentage of lines of credit never delinquent, and decrease the number of lines of credit requests in the most recent six months prior to applying for a new line of credit. Providing multiple diverse counterfactuals allows the option of selecting the most convenient option to an individual.

for the loan applicants, as discussed above. One limitation of the counterfactual generation algorithm is the inability to take feature relationships into consideration. For example, if one of the generated counterfactuals suggests decreasing the number of the total lines of credit, then decreasing number of lines of credit also leads to the changes in the features that depend on the total number of lines of credit. However, this type of relationship is ignored in the present method. Solving this problem would involve the intervention of an expert who is familiar with the relationship between the model's features, and can encode them as constraints in the counterfactual generation optimization process. We leave this for future studies.

Feature	Original Value	New Value
AvgAgeOfLOC	$60 - 75$	$4 - 29$
NumLOCNotDelq	$12 - 17$	$4 - 6$
NumLOCReqLast6M	$0 - 1$	$1 - 2$
FracRevLOCLimitUse	$13 - 29$	$77-$
$Pr(Default)$	0.09	0.58

Table 5: The steps that can make a creditworthy-deemed individual non-creditworthy, as suggested by the algorithm. In simple English, if the individual decreases the number of lines of credit not currently delinquent, opens up multiple new lines of credit leading to a decrease in the average age of their lines of credit, made multiple requests for lines of credit in the past six months, and increases the fraction of their revolving line of credit, then the individual will be deemed non-creditworthy.

Proximity Constraint(λ_1)	Mean # of Feature Changes	Loss Value (Distance)	# of success counterfactual found
0.5	3.96	0.054	1961
1.5	3.35	0.045	1909
5.0	2.66	0.035	1854

Table 6: Proximity Analysis. On increasing the proximity constraint (λ_1), the number of changes required for generating a successful counterfactual decreases, and the number of data points for which successful counterfactuals are generated also decreases. An optimal algorithm to generate counterfactuals for an individual is to start with a larger proximity constraint and slowly decrease it until a counterfactual is found.

6.4 Interpretability for Researchers and Data Scientists: Simple Rules to Summarize the Dataset

One property of interpretability of particular interest to data scientists is the summarization of the data and the model. A global view of the data and the model is able to give the 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.

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 section

(LIME and SHAP), data scientists can obtain the most important features and know their contribution to the model's overall prediction. For example, Figure 3 illustrates that the feature, "months since the newest request for a new line of credit (excluding those requested in 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 Figure 3a, we can observe that higher values of the fraction of all revolving line of credit limits in use increases the probability of default.

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 Table 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 Figure 5. 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 (that is, be classified as non-creditworthy) 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 inductive logic programming 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 non-creditworthy and creditworthy groups is able to achieve an accuracy of 70.65%.

In addition to ILP, there exist multiple other rule-finding methods in the literature. However, the ILP approach stands out in terms of its simplicity and ability to learn more effective

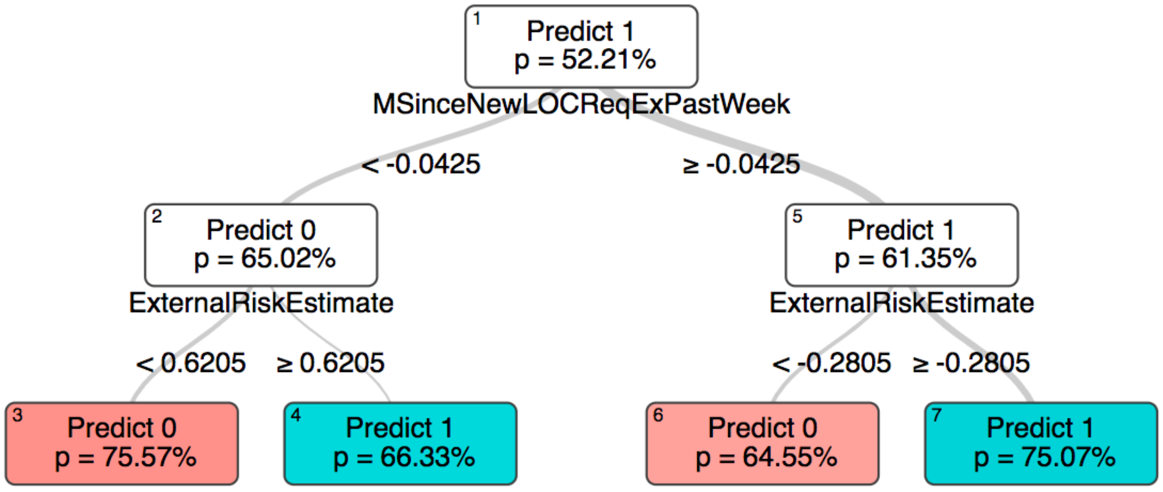


Figure 5: Explainable Optimal Tree Model used to learn simple rules summarizing the dataset. The rules states that a person will default on a loan (that is, be classified as non-creditworthy) if the number of months since a new line of credit has been requested (excluding the past week) is greater than 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.

rules. For example, 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 (BRCG) method described in Dash, Günlük, and Wei (2018) gives a set of rules to describe a dataset, but it requires dividing feature values into different bins before learning the rules. Data-binning limits the quality of the generated rules, since the algorithm might not select the threshold for the binning of features that is the optimal threshold for the rule.

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

7 Conclusion

In this paper, we have examined several different machine learning models, and used suitable tools to create explanations of their function according to the different needs of stakeholders involved in credit risk management. We have used state-of-the-art interpretable machine learning techniques, including Local Interpretable Model-Agnostic Explanations (LIME), SHapley Additive exPlanations (SHAP), and Diverse Counterfactual Explanations (DiCE), adapting 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, but can produce pragmatically valid suggestions and explanations.

In our results, we demonstrated that with the right tools, even black-box machine learning 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 machine learning models in credit risk management, but also provides borrowers, lenders, and data scientists with answers to questions that they may desire from ML models.

Interpretable models have the additional benefit of being convincing and easy to accept. Interpretable models may also help domain experts troubleshoot the inner workings of a complex model, which in turn will make it more accurate and 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 machine learning 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 work 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 along these lines should extend this explanability 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 explanability tools to the traditional factors used for credit risk forecast 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 this model's capabilities, in order to help quantify the model's overall fairness, response to stress testing, and the monetary impact due to the superior performance of black-box machine learning models.

References

Addo, P., D. Guegan, and B. Hassani. 2018. Credit risk analysis using machine and deep learning models. *Risks* 6:38–.

Albanesi, S., and D. F. Vamossy. 2019. Predicting consumer default: A deep learning approach. Working Paper, National Bureau of Economic Research.

Ariza-Garzón, M. J., J. Arroyo, A. Caparrini, and M.-J. Segovia-Vargas. 2020. Explainability of a machine learning granting scoring model in peer-to-peer lending. *Ieee Access* 8:64873–90.

Bertsimas, D., and J. Dunn. 2017. Optimal classification trees. *Machine Learning* 106:1039–82.

Bracke, P., A. Datta, C. Jung, and S. Sen. 2019. Machine learning explainability in finance: an application to default risk analysis. Bank of England working papers 816, Bank of England.

Breeden, J. L. 2020. Survey of machine learning in credit risk. *Available at SSRN 3616342* .

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and regression trees*. Monterey, CA: Wadsworth and Brooks.

Bussmann, N., P. Giudici, D. Marinelli, and J. Papenbrock. 2020. Explainable machine learning in credit risk management. *Computational Economics* 1–14.

Butaru, F., Q. Chen, B. Clark, S. Das, A. W. Lo, and A. Siddique. 2016. Risk and risk management in the credit card industry. *Journal of Banking & Finance* 72:218–39.

Caruana, R., and A. Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, 161–8.

Chapman, J. M. 1940. Factors affecting credit risk in personal lending. In *Commercial Banks and Consumer Instalment Credit*, 109–39. NBER.

Chen, C., K. Lin, C. Rudin, Y. Shaposhnik, S. Wang, and T. Wang. 2018. An interpretable model with globally consistent explanations for credit risk. *arXiv preprint arXiv:1811.12615* .

Chen, S., Z. Guo, and X. Zhao. 2021. Predicting mortgage early delinquency with machine learning methods. *European Journal of Operational Research* 290:358–72.

Croxson, K., P. Bracke, and C. Jung. 2019. Explaining why the computer says 'no'. *FCA* 5:31–.

Dash, S., O. Günlük, and D. Wei. 2018. Boolean decision rules via column generation. *arXiv preprint arXiv:1805.09901* .

Duan, J. 2019. Financial system modeling using deep neural networks (dnns) for effective risk assessment and prediction. *Journal of the Franklin Institute* 356:4716–31.

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

Evans, R., and E. Grefenstette. 2018. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research* 61:1–64.

Gogas, P., and T. Papadimitriou. 2021. Machine learning in economics and finance. *Computational Economics* 57:1–4.

Goodman, B., and S. Flaxman. 2017. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine* 38:50–7.

Hadji Misheva, B., A. Hirsa, J. Osterrieder, O. Kulkarni, and S. Fung Lin. 2021. Explainable ai in credit risk management. *Credit Risk Management (March 1, 2021)* .

He, K., X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–8.

Jiang, J., L. Liao, X. Lu, Z. Wang, and H. Xiang. 2021. Deciphering big data in consumer credit evaluation. *Journal of Empirical Finance* 62:28–45.

Khandani, A. E., A. J. Kim, and A. W. Lo. 2010. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance* 34:2767–87.

Krizhevsky, A., I. Sutskever, and G. E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems* ISSN 10495258. doi:http://dx.doi.org/10.1016/j.protcy.2014.09.007.

Kvamme, H., N. Sellereite, K. Aas, and S. Sjursen. 2018. Predicting mortgage default using convolutional neural networks. *Expert Systems with Applications* 102:207–17.

LeCun, Y., Y. Bengio, and G. Hinton. 2015. Deep learning. *nature* 521:436–44.

Lessmann, S., B. Baesens, H.-V. Seow, and L. C. Thomas. 2015. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research* 247:124–36.

Lundberg, S. M., and S.-I. Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., *Advances in Neural Information Processing Systems 30*, 4765–74. Curran Associates, Inc.

Ma, X., J. Sha, D. Wang, Y. Yu, Q. Yang, and X. Niu. 2018. Study on a prediction of p2p network loan default based on the machine learning lightgbm and xgboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications* 31:24–39.

Martens, D., B. Baesens, T. Van Gestel, and J. Vanthienen. 2007. Comprehensible credit scoring models using rule extraction from support vector machines. *European journal of operational research* 183:1466–76.

Molnar, C. 2020. *Interpretable machine learning.* Lulu. com.

Mothilal, R. K., A. Sharma, and C. Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 607–17.

Muggleton, S. 1991. Inductive logic programming. *New generation computing* 8:295–318.

Obermann, L., and S. Waack. 2016. Interpretable multiclass models for corporate credit rating capable of expressing doubt. *Frontiers in Applied Mathematics and Statistics* 2:16–
.

Ponomareva, K., and S. Caenazzo. 2019. Interpretability of neural networks: A credit card default model example. *Available at SSRN 3519142* .

Qadi, A. E., N. Diaz-Rodriguez, M. Trocan, and T. Frossard. 2021. Explaining credit risk scoring through feature contribution alignment with expert risk analysts. *arXiv preprint arXiv:2103.08359* .

Ribeiro, M. T., S. Singh, and C. Guestrin. 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, San Francisco, CA, USA, August 13-17, 2016*, 1135–44.

Rudin, C., and Y. Shaposhnik. 2019. Globally-consistent rule-based summary-explanations for machine learning models: Application to credit-risk evaluation. *Available at SSRN 3395422* .

Siddiqi, N. 2012. *Credit risk scorecards: developing and implementing intelligent credit scoring*, vol. 3. John Wiley & Sons.

Simonyan, K., and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICRL)* ISSN 09505849. doi:10.1016/j.infsof.2008.09.005.

Sirignano, J., A. Sadhwani, and K. Giesecke. 2016. Deep learning for mortgage risk. *arXiv preprint arXiv:1607.02470* .

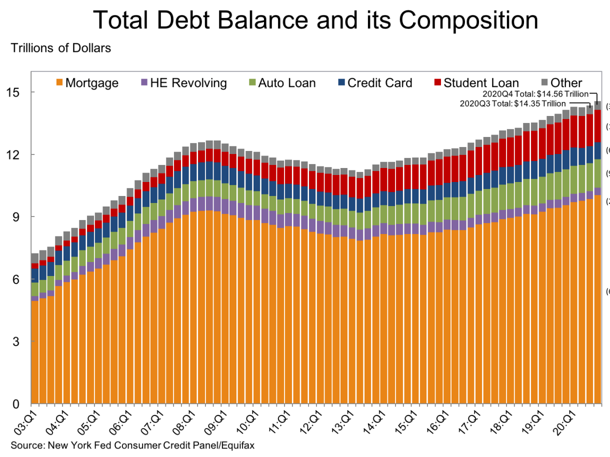
Tantri, P. 2021. Fintech for the poor: Financial intermediation without discrimination. *Review of Finance* 25:561–93.

Thomas, L. C. 1999. *A survey of credit and behavioural scoring: Forecasting financial risk of lending to consumers.* Citeseer.

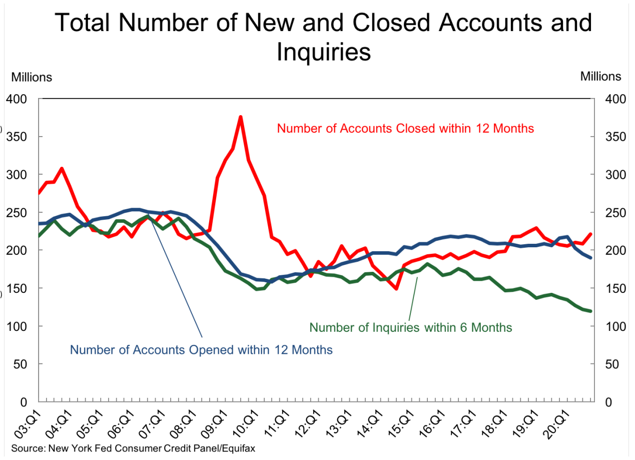
Wei, D., S. Dash, T. Gao, and O. Gunluk. 2019. Generalized linear rule models. In K. Chaudhuri and R. Salakhutdinov, eds., *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, 6687–96. PMLR.

A Household Debt Statistics

Figure A.1a shows the total household debt balance in the United States and its composition over time. We observe that the household debt balance reached an all-time high at the end of the fourth quarter of 2020, with a value of \$14.56 trillion. The overall trend line of the debt balance is increasing. Figure A.1b shows the total number of new accounts opened over time, along with the number of inquiries and number of accounts closed. The total number of accounts opened in the last single-year period measured is 189.6 million. This enormous debt balance and number of new accounts opened shows the importance of credit risk management. The data and the Figure A.1 were obtained from the New York Fed Consumer Credit Panel/Equifax.



(a)



(b)

Figure A.1: Household Debt. The plots were obtained from the quarterly report on household debt and credit released by the New York Fed.

B Dataset

B.1 Glossary of Relevant Credit Modeling Terms

The following definitions 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 fixed loan amount and a fixed 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.

B.2 Explanation of Predictor Features

In addition to the binary target variable (Risk Classification), each credit applicant is characterized by 23 predictor features, 21 continuous and 2 categorical. These are:

1. A condensed version of the borrower's credit risk computed by FICO using all credit bureau information (ExternalRiskEstimate)

2. Months since the very first 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 ever been 60 or more days delinquent (NumLOC60PlusDaysDelq)

7. Number of lines of credit ever been 90 or more days delinquent (NumLOC90PlusDaysDelq)

8. Percentage of lines of credit 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 6 months (NumLOCReqLast6M)

17. Number of requests for new lines of credit in the last 6 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)

B.3 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 machine learning 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. 331 of these data points are labeled as non-creditworthy, 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 are either a VIP and do not need to be investigated, or if they have no bureau record at all, i.e., 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, she 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, since she will be moved from a desirable value of the feature to a less desirable one. To handle these special values, we used binning techniques.

B.4 Data Visualization

Before using the dataset to train machine learning models, we analyze its properties using a few exploratory data visualization techniques.

Figure B.1 visualizes the 23x23 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 request for new lines of credit in the past 6 months (NumLOCReqLast6M) and the number of request for new lines of credit in the past 6 months excluding the past week (NumLOCReqLast6MExPastWeek)

C Model Implementation Details

The class distribution of the dataset in the five-fold cross-validation is presented in Table C.1. By visual inspection, it illustrates that the dataset is reasonably balanced.

We give details of the implementation for the different ML models used in our evaluation.

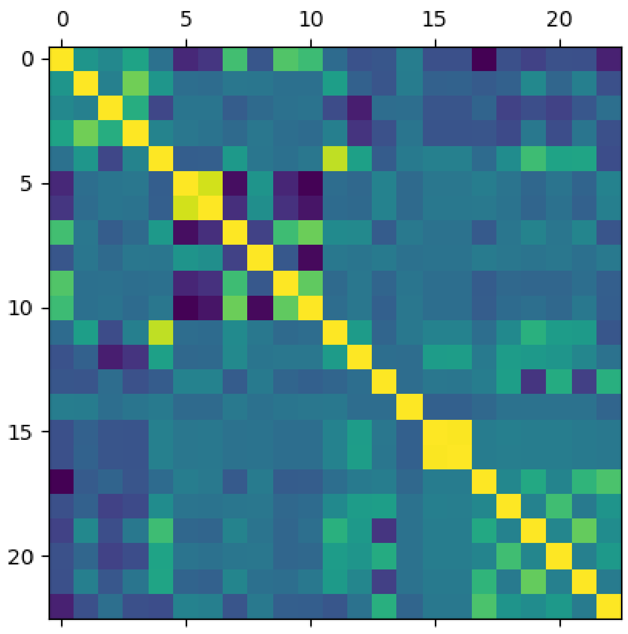


Figure B.1: 23 x 23 Correlation Matrix. The lighter shades represents highly correlated feature pairs. As expected, the diagonal has the lightest shade because it represents the correlation of a feature with itself.

For optimal trees, we used the Julia implementation available from the Interpretable AI website [6]. We performed a grid search over a depth from 1 to 10, and the number of features used for the deciding split at each node from the set 1,2,3,4,5,10,15,20,23. 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[7]. A grid search was performed on the number of estimators (trees) from 1 to 150. The best performing model consisted of 140 trees. Other parameters were set to the provided default values.

The neural network (WoE) model was implemented using the scikit-learn implementation of MLPClassifier. We performed a grid search over the different architectures and the L2-

[6]`https://docs.interpretable.ai/v1.0/`

[7]`https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

Fold Number	Count of $y_i = 1$ (Train)	Count of $y_i = 0$ (Train)	Count of $y_i = 1$ (Test)	Count of $y_i = 0$ (Test)
1	4118	3770	1010	963
2	4088	3801	1040	932
3	4094	3795	1034	938
4	4111	3778	1017	955
5	4101	3788	1027	945

Table C.1: Five-fold cross-validation data distribution. $y_i = 1$ corresponds to the data point (i.e., individual) who defaulted. The dataset is fairly balanced.

regularization constant. The best-performing model had three hidden layers consisting of 5 units each, and a L2-regularization penalty constant of 0.5. Other parameters were set to the default values.

The neural network (one-hot) model was implemented using tensorflow-keras. This was done to ensure the availability of gradients from the model for the counterfactual generating algorithm (DiCE). The best-performing model had one hidden layer, with the number of nodes=20. It was trained using the Adam optimizer with a learning rate of 0.01. The activation type used was the Rectified Linear Unit (ReLU), and a L1-regularization penalty of 0.001 was used.