Machine learning interpretability for a stress scenario generation in credit scoring based on counterfactuals

Expert Systems With Applications 202 (2022) 117271

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Machine learning interpretability for a stress scenario generation in credit scoring based on counterfactuals

Andreas C. Bueff [a,*], Mateusz Cytryński [b], Raffaella Calabrese [b], Matthew Jones [c], John Roberts [c], Jonathon Moore [c], Iain Brown [d,e]

[a] School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom
[b] Business School, University of Edinburgh, 29 Buccleuch Place Edinburgh, EH8 9JS, United Kingdom
[c] Nationwide Building Society, Nationwide Building Society Headquarters, Pipers Way, Swindon, SN3 1TA, United Kingdom
[d] SAS, Wittington House, Henley Rd, Medmenham, Marlow SL7 2EB, United Kingdom
[e] University of Southampton, Hartley Library B12, University Rd, Highfield, Southampton SO17 1BJ, United Kingdom

ARTICLE INFO

Keywords:
OR in banking
Interpretable ML
Credit scoring
Stress scenario

ABSTRACT

To boost the application of machine learning (ML) techniques for credit scoring models, the blackbox problem should be addressed. The primary aim of this paper is to propose a measure based on counterfactuals to evaluate the interpretability of a ML credit scoring technique. Counterfactuals assist with understanding the model with regard to the classification decision boundaries and evaluate model robustness. The second contribution is the development of a data perturbation technique to generate a stress scenario.

We apply these two proposals to a dataset on UK unsecured personal loans to compare logistic regression and stochastic gradient boosting (SBG). We show that training a blackbox model (SGB) as conditioned on our data perturbation technique can provide insight into model performance under stressed scenarios. The empirical results show that our interpretability measure is able to capture the classification decision boundary, unlike AUC and the classification accuracy widely used in the banking sector.

1. Introduction

With the growing prevalence of machine learning (ML) usage in the financial sector, there has been a growing interest in its applications in credit scoring. However, financial institutions are still reluctant to use ML models due to the 'blackbox problem', i.e. these techniques are so complex that the impact of their predictions are often difficult to explain and validate (Rudin, 2019). This problem is also known as the accuracy–explainability trade-off. The Bank of England (BoE) and the Financial Conduct Authority (FCA) conducted a survey on 106 UK financial institutions and determined that two thirds of respondents are already using ML in their business, even if with a limited number of use cases (Bank of England, 2019).

This paper is particularly focused on the use of ML techniques for credit scoring models, the quantitative method used by financial institutions to classify potential customers as good or bad borrowers (Thomas, Crook, & Edelman, 2019). This approach is based on a quantitative score that uses an application scorecard at the loan origination stage. The scorecard is used to record all data items that have predictive power related to the applicant's default risk. This data consists of characteristics about the borrower and information regarding previous relationships with the bank. Banks can develop their own internal quantitative models to translate this information into an individual score which is associated with a default probability of the borrower.

When the last financial crisis unfolded more than a decade ago, financial institutions discovered that most of their blackbox algorithms used to estimate this score were based on flawed assumptions. Therefore, financial regulators decided that additional controls were needed and introduced regulatory requirements for modelling risk management in the banking sector. For example, in April 2011 the Board of Governors of the Federal Reserve System in the US published a document (the Board of Governors of the Federal Reserve System, 2011) that states that banks have to prove that they understand the models they are using. The Financial Stability Board (FSB) in 2017 pointed out that ML is likely to bring many challenges to financial stability due to the lack of model explainability (Financial Stability Board, 2017).

* Corresponding author.
E-mail addresses: andreas.bueff@ed.ac.uk (A.C. Bueff), mateusz.cytrynski@gmail.com (M. Cytryński), raffaella.calabrese@ed.ac.uk (R. Calabrese), Matthew.Jones3@nationwide.co.uk (M. Jones), John.Roberts2@nationwide.co.uk (J. Roberts), Jonathon.Moore@nationwide.co.uk (J. Moore), iain.brown@sas.com (I. Brown).

https://doi.org/10.1016/j.eswa.2022.117271
Received 25 March 2021; Received in revised form 9 February 2022; Accepted 13 April 2022
Available online 26 April 2022

The European General Data Protection Regulation (GDPR) (Voigt & Bussche, 2017) provides the right to an explanation of algorithm decisions. To guarantee this right, financial regulators require that the models used by financial institutions for computing the capital requirements should be easily understood. This guideline has been followed by the European Banking Authority that defined a model explainable "when its internal behaviour can be directly understood by humans (interpretability) or when explanations (justifications) can be provided for the main factors that led to its output" (European Banking Authority (EBA), 2020). Both these regulations create a demand for explainability of ML models in the banking sector.

The two primary aims of this paper are to provide a proposal for making ML models easier to explain and to propose a method to generate stress scenarios for cross sectional data. First, we suggest the use of counterfactuals (Johansson, Shalit, & Sontag, 2016) in understanding the internal workings of the model so that banks can be compliant with financial regulations. Particularly, we analyse the robustness of credit scoring models based on ML techniques to understand the minimum data perturbation to misclassify a borrower. Second, we suggest a data perturbation technique that can be used for generating stress scenarios in a cross-sectional context.

Performance of traditional credit score models, using logistic regression, typically perform well in an economic downturn, in terms of their ability to classify the worst borrowers worse than the best borrowers i.e. the model's discrimination or 'rank ordering' is not materially impacted. However, these models can deteriorate from an accuracy perspective and the probability of default at a given score is no longer as accurate as it should be. As ML models have not been used for the purpose of credit scoring in financial services for long, there is no deep understanding of how they will perform under the same conditions. The purpose of this paper is to explore this dynamic.

We apply our two methodological proposals to a dataset on 61,239 UK unsecured personal loans issued from June 2014 to July 2015. We generate different stressed scenarios from this dataset and we apply both the logistic regression and the stochastic gradient boosting (SGB) techniques as scoring models. Our first set of experiments demonstrate the impact on model performance when trained on augmented data and show that model performance is in fact negatively impacted when the default ratio is increased. Our second set of experiments use our counterfactual distance metric to compare logistic regression with SGB across each dataset feature followed by a comparison on the impact of feature constraints when applied to counterfactual generation. Results here indicate that the decision boundary is smaller with some black-box models, and that the application of constraints provides insight into feature value ranges which correlate with model misclassification. The last set of experiments combine the two approaches to bring together the study of stressed scenarios via augmentation of the data and isolating weaknesses in the classification decision boundary using counterfactuals.

The remainder of this paper is structured as follows. Section 2 discusses related literature on the explainability of ML techniques in the banking sector. Section 3 provides an overview of counterfactuals and presents our proposal. A novel approach to generate synthetic data for stress scenario is presented in Section 4. Section 5.1 describes the data used for the experiments and explains the setting of the problem. Finally, Section 5 shows the results of applying the proposed methods to the previously stated problems.

2. Literature review

The usage of ML techniques by financial institutions and Fintech companies has gained significant popularity over recent years (Bank of England, 2019). Based on our knowledge, one of the first uses of ML methods in credit scoring dates back to 1996, when Henley and Hand (1996) obtained that a K-Nearest Neighbour classifier is more accurate than linear and logistic regression, decision trees and graphs. Since then, a large variety of machine learning techniques has been applied to credit scoring showing that these methods are often more accurate than traditional regression models (Brown & Mues, 2012; Kraus, Feuerriegel, & Oztekin, 2020; Lessmann, Baesens, Seow, & Thomas, 2015; Paleologo, Elisseeff, & Antonini, 2010).

Analogously to the analysis conducted by the Bank of England (Bracke, Datta, Jung, & Sen, 2019), in this paper we chose the stochastic gradient boosting (SGB) (Friedman, 2002) as a machine learning technique as it provides the highest predictive accuracy when compared to other methods such as random forest and support vector machines. However, it should be mentioned that SGB is opaque in its decision making, rendering it a non-interpretable ML technique. Chang, Chang, and Wu (2018) apply SGB to a highly imbalanced data (10% of defaulted loans) provided by a financial institution in Taiwan from 2009 to 2016 showing that SGB achieves the highest discriminant accuracy compared to those obtained by logistic regression, support vector machines, and neural networks. Addo, Guegan, and Hassani (2018) also find that SGB achieves highest performance results among logistic regression, random forest, and neural networks with various number of hidden layers. As their dataset is highly imbalanced with 1% of defaulted loans, the authors apply an oversampling technique called SMOTE. Addo et al. (2018) point out the importance of transparency and interpretability of credit scoring models, however they do not explore it further. Moreover, Letham, Rudin, Mccormick, and Madigan (2015) and Paleologo et al. (2010) emphasize that not only accuracy but also interpretability are relevant characteristics for credit scorecards.

Even if some authors have interchangeably used the terms interpretability and explainability, they are two different concepts. The former is mainly focused on describing the prediction in a way that is understandable by humans, while the latter aims to explain the underlying mechanics of a particular decision obtained by the algorithm (Gilpin et al., 2018). As the main aim of this paper is to suggest a technique that financial institutions can use to explain their internal scoring models to regulators, we focus our analysis on interpretability. Furthermore, analogously to a regulatory approach (Bracke et al., 2019), we are interested in credit scoring prediction and not in explaining the causes of loan defaults.

Existing techniques on interpretability can be divided into two main categories: local and global approaches. The latter analyse the model as a whole, for example to analyse possible bias detection. On the other hand, local interpretability tries to identify how different features influenced a particular prediction of the model, e.g. a lending decision for a particular applicant. Analogously to a regulatory approach, we choose in this paper to focus on global interpretability. Classically, global interpretability seeks to understand how a model came to a decision and is based on a general perspective of the input features and the learned coefficients such as weights, parameters, and network architecture. Identifying the importance of features and their interactions are explanations that can be considered global. Our contribution provides global interpretability via analysis of the decision boundary based on the model input and counterfactual generation (Molnar, 2019). Alternatively, our proposal is also able to provide an instrument for local interpretability, for example why a particular loan application was rejected, as Grath et al. (2018) showed in their analysis.

Counterfactuals are example-based explanations that describe causal situations in the form of '*if A had not happened, B would not have happened either*' (Johansson et al., 2016). Exemplar usage of applying counterfactuals to tree based models can be found in Tolomei, Silvestri, Haines, and Lalmas (2017). Where financial institutions are concerned, counterfactuals can be used on an individual level to assess what would have to happen to change the prediction from default to non-default or vice versa. It not only provides better understanding of model predictions, but can also be used as a suggestion to the client on what they could do to get the loan next time they apply for it.

Similar interpretable methods include LIME and SHAP, which a can be used for both local and global interpretability via feature importance. LIME builds local surrogate linear models for the purpose of model interpretability, where the feature weight coefficients on the local linear model can be interpreted as feature importance (Ribeiro, Singh, & Guestrin, 2016). While our proposed approach uses counterfactuals to model the decision boundary, our focus is on robustness and not necessarily feature importance. SHAP (Shapley Additive exPlanations) is a model agnostic value estimation method for explaining individual predictions, where SHAP learns local explanations by utilizing from game theory, so called Shapley Values (Lundberg & Lee, 2017). In a recent effort, Giudici and Raffinetti (2021) have extended SHAP to further address global explainability goals by incorporating Lorenz decompositions. The authors integrate the Lorenz Zonoid model accuracy tool, which is related to the AUROC measure, as such provides a metric that incorporates model performance and model explainability. Our contribution also proposes a new metric; however, our focus is on the use of counterfactuals to calculate the decision boundary conditioned on perturbations to the dataset to gauge model robustness to various scenarios.

There are few studies in the literature that analyse interpretability in credit risk management. Chen, Janizek, Lundberg, and Lee (2020)ranked the features of a credit scoring model available in the LendingClub dataset based on their Shapley value. Then the authors computed the change in the scoring model's predicted log odds of default after imputing each feature (up to 10 features) to the mean. Interventions on the features ranked based on the Shapley values computed on the imputed features lead to a substantial decrease of the predicted likelihood of default. Bussmann, Giudici, Marinelli, and Papenbrock (2020) proposed to apply correlation networks to Shapley values to cluster the predictors in ML models used for credit scoring. They applied this approach to data on small and medium enterprises showing that good and bad borrowers can be clustered based on these predictors.

The Bank of England (Bracke et al., 2019) applied a global approach to explain SGB using the Shapley value, a game theoretic concept that tells how much each feature contributes to the final prediction. They compare logistic regression and SGB on UK mortgage data during the 2015–2017 period. Bracke et al. conclude that the most important features found by the Shapley value – loan-to-value ratio and current interest rate – are in line with the relevant literature. Another main finding is that explanations for ML models depend on the input region considered, and that they should be tested extensively for several potential states of the world.

The conclusion by the Bank of England naturally leads to the notion of stress testing, a simulation technique used to test the resilience of scoring models against possible future economic downturn. Stress testing is becoming very important in the risk evaluation of banks and represents a key technique for risk management and capital decisions for financial institutions, as recognized by the Financial Services Authority (Financial Services Authority, 2008). Most of the approaches available in the literature for stress testing use dynamic models, such as survival models (Bellotti & Crook, 2013; Wang, Crook, & Andreeva, 2020).

To apply a dynamic approach, we need data from multiple years. For this paper we have access to a cross-sectional dataset extracted by Nationwide in 2014–2015, for this reason we cannot apply a dynamic model. We instead propose a method to perturb the data based on the k-Nearest Neighbours algorithm (Henley & Hand, 1996) and, therefore, generate stress scenarios to test the scorecard. We highlight that this proposal is not considered a stress testing approach as it does not capture the dynamic behaviour of the economic cycle. Since default rates tend to increase under downturn economic conditions (Hall, 2010), we can manipulate the default ratio as a proxy of different states of the world. It is crucial to note that we did not perform the perturbation using sampling techniques such as SMOTE (Baesens, Rösch, & Scheule, 2016; Chawla, Bowyer, Hall, & Kegelmeyer, 2002) as the method incorporates randomness in generating synthetic instances, which we aim to avoid as we sought to maintain a degree of consistency with the original data. Ultimately, it was a design choice to reject using a stochastic approach, such as SMOTE, for the stress scenario exercise to avoid adding an additional source of complexity to the problem. While the generation of balanced synthetic data has demonstrated success in the past with SMOTE (Patil, Framewala, & Kazi, 2020), SMOTE can potentially generate unseen feature values in a stochastic manner, thus we sought alternative methods in data generation. Therefore, we settled on using k-Nearest Neighbours for the perturbation procedure as original data values from the population of applicants can be reused and changes to the outcome can be made to meet the stress criteria. The exact details of our k-Nearest Neighbours approach and the counterfactual methodology can be found below.

3. Counterfactuals and robustness

It requires mentioning, robustness in statistics refers to the study of learning on corrupted data, such that data statistics can still be recovered. We acknowledge the statistical definition, however we use the ML definition of robustness which is defined by the minimum perturbations needed to craft an adversarial example for a ML model to misclassify a datapoint. In essence, robustness in ML refers to adversarial examples for a trained model, in the test set potentially, and robustness in statistics refers to the handling of adversarial examples prior to training. We note, as we associated global interpretability with comprehension of model robustness, we clarify that robustness is a reflection of a model's decision boundary. That when analysed conditioned on model input, provides global interpretability via understanding of the relationship between model performance and input features. Whether a model is sensitive or resilient to adversarial examples, such information provides a global perspective on model performance.

Counterfactuals generated from a blackbox machine learning model can be harnessed to create a counterfactual distance scoring metric to pinpoint areas of weakness in the model. These areas would pertain to values within the feature space. Counterfactuals have provided means of interpreting machine learning models in the past (Guidotti et al., 2019, 2018; Sharma, Henderson, & Ghosh, 2019), and have been useful in explaining the classification of individuals. Recent advances in counterfactual research have used genetic algorithms to create synthetic populations of instances in the neighbourhood of a datapoint. Taken as a whole, counterfactuals generated with genetic algorithms and machine learning models can express the beliefs of the model conditioned on the data. The score proposed is the metric distance of the counterfactuals and the original datapoint, and so provides a numerical understanding of how well a model performs given certain neighbourhoods in the data.

Applying constraints to counterfactual generation allows a modeller to analyse subpopulations of the data constrained by specific data values. To the best of our knowledge the proposed method is the first systemic analytical framework using counterfactuals to derive a new evaluation metric for model confidence. While Sharma et al. propose a similar counterfactual based metric with optional constraints, our approach expands on the use of the metric by specifying constraints so that we can analyse the robustness of individual features as well as specified continuous ranges for individual features on the model (Sharma et al., 2019). Importantly, Sharma et al. utilize constraints to prevent the generation of counterfactuals that are infeasible or statistically unlikely, whereas we apply constraints in order to interpret model performance more succinctly through inspection of constrained inputs. The granularity of our constraints provides a means of broader interpretation of the model, and as we later make use of generating synthetic data, we can better infer model performance under specified conditions.

As we want to observe the performance of the model given augmented feature values with known classification, we found that counterfactuals can assist in generating datasets contingent on the model and constraints, which can lead to misclassification (i.e. expose feature values which potentially lead the model to make errors). The following methods use counterfactuals generated on subsets of the test population, which fall under a particular feature constraint, to determine the robustness of these subsets with respect to the model. The goal with this method is to complement sampling methods which provide a given blackbox model with biased data, such that we can observe the performance of a machine learning model trained on multiple augmented datasets. The following section provides an overview of the blackbox approach in generating counterfactuals and how they are utilized in calculating the counterfactual distance metric referred to now as the *robustness score* as well as our algorithm incorporating the *robustness score* in analysis of model weak points using constraints.

3.1. Counterfactuals in machine learning

A counterfactual is defined as a generated datapoint that is as close to the input datapoint as possible but which the models gives a different outcome (Sharma et al., 2019). To illustrate, had a user been denied a loan then a counterfactual statement could take the form, *"Had your income been $5000 greater per year and your credit score 30 points higher your loan would be approved"*. Counterfactuals explain model results to a user by providing them actionable ways of changing their behaviour to obtain favourable predictions. Simply put, counterfactuals explain which data features should be changed (and by what amount) to change a specific classification predicted by a model to a different predicted classification.

In past works, counterfactuals have been used for local explanations (Guidotti et al., 2019, 2018; Sharma et al., 2019), but also have the capacity to be used to identify model decision boundaries in particular data neighbourhoods. By neighbourhood, we are referring to points in the dataset where feature values are broadly similar. Counterfactuals for a datapoint can explain the decision boundary in that neighbourhood of datapoints. By taking counterfactuals for multiple datapoints, with say similar feature values, counterfactuals can explore the decision boundary for those datapoints conditioned on their feature values. For example, a grouping of users denied loans with incomes lower than $2000 per year, and keeping the counterfactual constrained to this feature value, may receive a counterfactual *"On average, had their credit score been 70 points higher, they would have been approved for the loan"*.

As counterfactuals can define decision boundaries by providing generated datapoints close to the input datapoint. The distance of the counterfactuals to the original point reflects how robust the model is to adversarial examples. If the counterfactuals across classes are further away from the input instances on average for say model A compared to another model B, we can say that the first model A is less likely to make a false prediction. Taken another way, if we find groupings of datapoints with similar feature values, we can take the average distance of the decision boundary to see how easy it is to fool the model given datapoints within the similar feature value ranges.

3.2. Genetic algorithm

Genetic algorithms (GAs) provide positive benefits in counterfactual generation including generating points on linear and non-linear models. They also allow for constraints to be added which allows for exploration of the decision boundary of particular sub-groups in the data, as well as restricting feature changes and/or generating counterfactuals on specific feature ranges. Key to understanding how a GA can generate counterfactuals for an instance (x), we need to define the distance function and the fitness function. In this body of work, the primary means of generating counterfactuals via a GA follows the LORE pipeline as introduced by Guidotti et al. (2019).

3.2.1. Distance function

The distance function determines the metric difference between an instance (x) and a counterfactual (c). The model presented by Guidotti et al. (2019) LORE, provided a distance function $d(x, c)$ for mixed data (categorical and continuous). As we normalize the data in our experiments, we treat all features as continuous thus our distance function is the normalized Euclidean distance, where the distance function maps the feature space consisting of m attributes to a value between 0 and 1. In Eq. (1) we normalize the euclidean by relying on the total sum of the variances between our two instances in question, as we are comparing the distance between our original instance and the counterfactual, whereas we normalize the variance by the size of the instance vector (m). The mean of our instance vector (x) is represented by $(\hat{x})$.

$$NormEuclid(x, c) = \frac{1}{2}\frac{Var(x - c)}{(Var(x) + Var(c))}, \quad \text{where} \quad Var(x) = \frac{\sum_{i=1}^{m}(x_i - \hat{x})^2}{m} \tag{1}$$

$$d(x, c) = NormEuclid(x, c), \text{ where } \quad d : X^m \rightarrow [0, 1] \tag{2}$$

While we operated with only continuous data, the discrete case is a possibility for future research. As many datasets are mixed discrete continuous, the distance function in Guidotti et al. (2019) also provides a method for handling discrete features.

$$d(x, c) = \frac{h}{m} SimpleMatch(x, c) + \frac{m - h}{m} NormEuclid(x, c) \tag{3}$$

Where a given dataset contains h categorical features and $m - h$ continuous features. The $SimpleMatch$ function is the weighted sum of simple matching coefficients for categorical features.

3.2.2. Fitness function

Two fitness functions are used in the GA, one fitness function looks for instances (c') similar to (x) but not equal to (x) for which the model predicted the same classification. The second fitness function assists in generating instances similar to (x), again not equal to (x), where the model predicts a different classification.

$$\begin{aligned} fitnessEqual(x, c', M) &= \mathbb{I}_{M(x)=M(c')} + (1 - d(x, c')) - \mathbb{I}_{x=c'}, \\ fitnessNotEqual(x, c', M) &= \mathbb{I}_{M(x)\neq M(c')} + (1 - d(x, c')) - \mathbb{I}_{x=c'} \end{aligned} \tag{4}$$

In the following equations, $\mathbb{I}$ is the indicator function for a conditional statement, $\mathbb{I}_{true} = 1$ and $\mathbb{I}_{false} = 0$. The term $(1 - d(x, c'))$ determines the similarity of (x) and (c'). $\mathbb{I}_{M(x)=M(c')} = 0$ determines whether the predicted output produced by model M predicts the same class for (x) and (c') with the indication of 0 signifying the classifications are different, while $\mathbb{I}_{M(x)=M(c')} = 1$ indicates the predicted classifications are the same. $\mathbb{I}_{x=c'}$ determines whether (x) and (c') are the same as this would hurt the overall fitness for both fitness functions. Alternatively, $\mathbb{I}_{M(x)\neq M(c')} = 1$ indicates that model classification on the two instances is different, and for the sake of calculating robustness, we are concerned with this case. Ideally, the first indicator term $\mathbb{I}_{M(x)\neq M(c')}$ returns 1 and we add the similarity measure, which approaches 1 the closer the counterfactual is to (x) as the distance function is scaled between 0 and 1, with the final measure $\mathbb{I}_{x=c'}$ subtracting 1 from the fitness valuation if the potential counterfactual is identical to the original instance.

While the GA has the capacity to generate two populations on an instance (x) using $fitnessEqual$ and $fitnessNotEqual$, in the context of developing our robustness metric, we only analyse the counterfactual populations generate by $fitnessNotEqual$, so the concern is to identify potential instances such that $\mathbb{I}_{M(x)\neq M(c')} = 1$. Ideally we seek potential counterfactuals where $fitnessNotEqual(x, c', M) \geq 1$ and so the GA seeks to maximize this result.

3.2.3. Genetic algorithm pipeline

The GA relies on the fitness functions and the fitness function relies on the distance function. The fitness function evaluates which counterfactuals are similar to (x) using the distance function. The classification of the counterfactuals predicted by the model M is used as well in the fitness function. The fitness function then assigns a metric determining how fit a given counterfactual is. This process is reflected in the *evaluation function*. A *selection function*, selects a population produced by the evaluation function with the highest fitness scores.

A *crossover function*, is then applied on the selected fittest population of counterfactuals. Crossover of counterfactual feature values is based on a user defined probability value (γ). Here, two-point crossover is used which select 2 parents and crossover features at random by swapping the crossover feature values of the parents.

The next method for counterfactual generation with a GA is the *mutation function*. A subset of the generated counterfactuals from the crossover function are mutated based on a user defined probability (μ). The selected subset of counterfactuals for mutation are transformed by replacing the feature values at random according to the distribution of feature values.

The GA loops for a number of generations G where (1) the fittest counterfactuals are selected, (2) the fittest counterfactuals are augmented and return a new population using the crossover function, and then (3) a subset of the new population is selected and feature values are manipulated via the mutation function. (4) The final step passes the new counterfactual population into the evaluation function to determine the fitness of individual counterfactuals and the loop continues. In algorithm 1, we can see the pipeline with the relevant functions used in generating a genetic neighbourhood for a specific instance.

Algorithm 1: GeneticAlg($x, fitness, M, N, G, \gamma, \mu$)

Input: x - instance to explain, M - model, $fitness$ - fitness function, N - population size, G - # of generations, γ - crossover probability, μ - mutation probability
Result: **c** -neighbours of x
$P_0 \leftarrow \{x|\forall 1 \cdots N\}$; $i \leftarrow 0$;
$evaluate(P_0, fitness, M)$
while $i < G$ **do**
  $P_{i+1} \leftarrow select(P_i)$
  $P'_{i+1} \leftarrow crossover(P_{i+1}, \gamma)$
  $P''_{i+1} \leftarrow mutate(P_{i+1}, \mu)$
  $evaluate(P''_{i+1}, fitness, M)$
  $P_{i+1} = P''_{i+1}$
  $i \leftarrow i + 1$
end
c $\leftarrow P_G$
return c

3.3. Framework and robustness metric

As stated, the idea of our method is to use counterfactuals for features to determine the *robustness score* on datapoints with particular feature value intervals which lead to misclassification. We can isolate datapoints with particular feature value ranges and use the *robustness score* to indicate if the model is weak in predicting the classification. The CERTIFAI model introduced by Sharma et al. introduced the use of counterfactuals as a means to generate a robustness metric as well as the concept of applying constraints, and here we take a similar approach for calculating the robustness metric with constraints, with the difference being how we apply the scoring metric with constraints for broader interpretability (Sharma et al., 2019).

The output of this pipeline would include feature ranges where the model has an increased likelihood of misclassification. This information can then be passed to a sampling method to generate a synthetic dataset to observe the prediction ability of the model given perturbed feature values. Inversely, this method could be used to observe performance of a model trained on biased data (eg. increased presence of defaults) and use constraints in the generation of counterfactuals to observe the robustness of specific neighbourhoods in the data that are defined by feature value intervals. The pipeline calculating the robustness of subpopulations in the data is as follows:

1. As we are interested in looking at particular feature ranges, we create a subpopulation derived from the test set with a specified interval range. Assuming we have a feature f_i which has feature values in the range $a \leq v \leq b$ where (v) is a given value in the range. We could specify an interval $f_i \in [\alpha, \beta]$ where $a \leq \alpha$ and $\beta \leq b$. In the context here, we would perform equal-width binning on the features and identify each subpopulation that falls into a given interval. As some values may have a higher frequency we would need to limit the constrained subsets from the test set in order for their size to be generally equal and avoid bias.
2. **Constraints**: First we would need to apply the feature range constraints to the GA, as this will return counterfactuals with the explicit feature range we wish to analyse. So given a particular feature with a specified feature range $f_i \in [\alpha, \beta]$ that we wish to constrain in the generation of counterfactuals, after selecting real datapoints from the test set with our specified feature range, we apply our specified feature range as a constraint C_i into the GA pipeline. This constraint would be applied to the mutation function, as this method selects values from feature range distributions but would now be limited to our constraint. The current counterfactual and neighbourhood population generated for x is defined by P_i.

$$P_i = mutate(P_i, \mu, C_i) \tag{5}$$

3. **Robustness**: Once we generate counterfactuals using just the *fitnessNotEqual* function, as we want to identify the decision boundary for our constrained subset, we calculate the *robustness score* which is the expected distance (or average normalized Euclidean distance) between input instances and their corresponding counterfactuals. As we are looking at the *robustness score* across all instances in the subset and constrained counterfactuals, we are able to identify the size of the decision boundary. The smaller the score, the closer the decision boundary is between instances in the subset and their counterfactuals, thus identifying what feature values lead to model misclassification. In calculating average *robustness score* for a single instance (x) we normalize the sum of the Euclidean distance for all generated counterfactuals by N_c, which is the final number of counterfactuals generated for an instance x. As our proposal seeks to identify the average robustness for multiple instances $\mathbf{x}$, we normalize the sum of our *robustness score* across all instances in $\mathbf{x}$ by $N_{\mathbf{x}}$, which reflects the number of instances in ($\mathbf{x}$).

$$RobustScore = \mathbb{E}[d(\mathbf{x}, \mathbf{c})] \tag{6}$$

$$RobustScore_x = \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{1}{2} \frac{Var(x - c_i)}{(Var(x) + Var(c_i))} \tag{7}$$

$$RobustScore_{\mathbf{x}} = \frac{1}{N_{\mathbf{x}}} \sum_{j=1}^{N_{\mathbf{x}}} \sum_{i=1}^{N_c} \frac{1}{2} \frac{Var(x_j - c_i)}{(Var(x_j) + Var(c_i))} \tag{8}$$

Essentially, the pipeline will loop through all features and isolate a feature range to constrain on, then (1) create a new subset from the test set based on the constrained feature range, (2) then apply the feature range constraint in the GA for generating counterfactuals, (3) and followed by returning the average *robustness score* of these constrained counterfactuals.

3.4. Robustness of machine learning models and areas of weakness

Putting all the methods together, we arrive at system for analysing specific neighbourhoods in the data with the following algorithm, see Algorithm 2. Two binning methods can be implemented, the primary binning method implemented was equal-frequency binning where the number of instances is roughly equal for each bin interval, and equal-width binning which divides the feature value range into equal partitions. In order to analyse a model with regard to feature regions partitioned by (k), we iterate over all features and use a *binning function* to identify constraints (C). The specific constrained feature range (ϕ_i) for a given feature f_i is then passed to the *subpopulation function* to get the subset of instances falling within the feature range constraint. The subset of constrained instances are then used by Algorithm 1 to generate counterfactuals which are then used in calculating the *robustness score* for the given feature range.

Algorithm 2: NeighbourRobustness($b, M, m, N, G, \gamma, \mu, k, D$)

Input: M - model, m - # of continuous features , N - population size, G - # of generations, γ - crossover probability, μ - mutation probability, k - number bins, D -dataset
Result: $\mathbf{S}$ - robustness values for the model
$F \leftarrow \{f | \forall 1 \cdots m\}$;
for $f_i \in F$ **do**
  $C = binning(f_i, k)$; $score_i \leftarrow 0$
  for $\phi_j \in C$ **do**
    $D^{'} = subpopulation(D, \phi_j)$
    for $x \in D^{'}$ **do**
      $\mathbf{c} =$ GeneticAlg($x, fitnessNotEqual, M, N, G, \gamma, \mu$)
      $score_i = \mathbb{E}[d(x, \mathbf{c})]$
      $score = score + score_i$
    end
    $\mathbf{S}(i, j) \leftarrow \frac{score}{|D^{'}|}$
  end
end
return $\mathbf{S}$;

3.5. Primary advantages of the robustness metric

The *robustness scores* for feature ranges provide an interpretable metric for analysing the strength of a given model. Concretely, it is interpreted that features with larger *robustness scores* are performing better in the sense that the generated counterfactuals are 'further away', and can be inferred that those value ranges, in general, produce a more reliable prediction by the model. For example, given a feature say, $f_i \leftarrow Income$ with 3 bins produced thus providing 3 constraint ranges, with bin values of $[6.7, 18.3, 4.9]$ corresponding with ranges of $[v \leq 1000), (1000 < v \leq 50000), (50000 < v]$, we see that Bin 2 is more robust than Bin 1 and 3. As the distance metric calculated counterfactuals with an average distance of 18.3 whereas the other bins have lower scores, thus instances within the value range $(1000 < v \leq 50000)$ generate counterfactuals further from the original on average. This signifies that the model is more confident about classifications for features in that range (not necessarily what the classification is, just that the model is confident in its decision). These values also tell us that in the cases where the model is classifying instances with $f_i = 100$ or $f_i = 100000$ we should be cautious as the decision boundary in these ranges is less robust. The *robustness score* tied with constrained counterfactuals provides a means to pinpoint feature vulnerabilities in machine learning. The *robustness score* conditioned on discrete ranges for all features provides an additional analysis metric that helps modellers identify which features and more explicitly which ranges are associated with increased likelihoods of misclassification with regard to a model.

Robustness has been investigated in conjunction with financial applications in the past. Petropoulos et al. proposed a robustness metric based on model likelihood on discretized portions of data with particular groupings pertaining to actual classifications (i.e. loan defaults) (Petropoulos, Siakoulis, Stavroulakis, & Klamargias, 2019). The approach is similar in regards to measuring confidence of the model, especially with respect to comparisons with other models, however our framework for robustness is far more granular in that (Petropoulos et al., 2019) are limited to observing groupings of so called score values, while we can analyse performance on individual feature ranges. Other works which investigated model robustness and credit scoring often focused on the construction of a more robust model (Abdou, 2009; Luo, 2020) or used previously defined metrics unrelated to counterfactuals (Vieira, Barboza, Sobreiro, & Kimura, 2019).

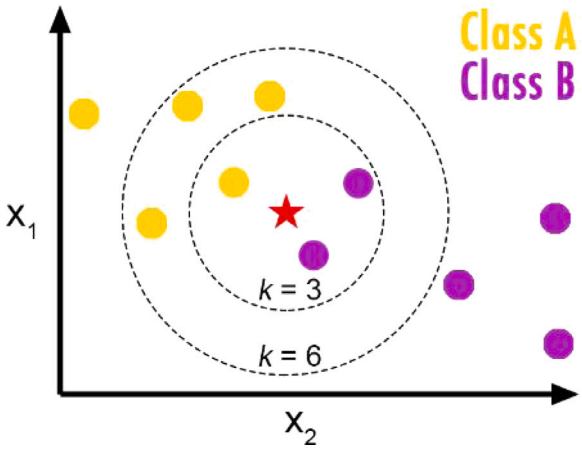

Fig. 1. kNN example for different number of neighbours for two dimensional space.

Counterfactuals have rarely been utilized with respect to credit scoring. McGarth et al. implemented counterfactuals for the use in local interpretations of a models decision (Chen et al., 2018), and instead of using a genetic algorithm for generating counterfactuals they made use of optimization on a loss function with the Nelder–Mead algorithm. Overall, past works have used counterfactuals to assist with financial applications primarily with regard to interpretation of model decisions at the local level (Chen et al., 2018; Grath et al., 2018; Wachter, Mittelstadt, & Russell, 2017; White & d'Avila Garcez, 2019), or for an analysis on model fairness (Coston, Mishler, Kennedy, & Chouldechova, 2020; Kusner, Loftus, Russell, & Silva, 2017; Wang, Sridhar, & Blei, 2019; Zhao, Coston, Adel, & Gordon, 2019), but none, as far as we have observed, use counterfactuals as adversarial examples to analyse model performance on feature values. While some works have used counterfactuals to analyse predictive weaknesses of a model (Sokol & Flach, 2019), none take advantage of generated counterfactuals for constrained regions in the data for analysing local robustness.

4. Data perturbation for stress scenario generation

In the following section, we explain our proposal for generating stressed scenarios from a cross-sectional dataset. We consider the particular restriction where only a dataset for a given year is available so we cannot apply a dynamic model, as explained in Section 2. We suggest the use of the k-Nearest Neighbours classifier to generate synthetic data with different default rates or feature values of clients to represent the economic downturn. The following subsections describe the proposal in details.

4.1. k-Nearest Neighbours

K-Nearest Neighbours (kNN) is a machine learning algorithm that classifies a new datapoint based on a similarity measure (Henley & Hand, 1996). As the name suggests, it is used to classify a datapoint based on the classification of its nearest neighbours. The general idea of this procedure for a two label classification can be seen in Fig. 1 below.

This is exactly the setting for our case — in credit default prediction problems, the clients are usually divided into two classes, defaulted

and non-defaulted. The figure above simplifies this problem to two dimensions. When a new point is added, we first need to choose the number of neighbours that we will be considering for classification (usually represented by k), as well as selecting a distance metric used as a similarity measure. A popular one is the Euclidean distance metric seen in Eq. (9).

$$d(\vec{p},\vec{q}) = \sqrt{\Sigma_{i=1}^{n}(p_i - q_i)^2}, \quad i = 1,2,\dots,n \tag{9}$$

where $\vec{p}$ and $\vec{q}$ are observations and n is the number of features in the data. Under the chosen distance metric, we check which points are the closest to the new datapoint, and count them. Then, based on this count, the point is assigned to a certain class. For example, suppose that we are interested in checking 5 nearest neighbours, and we see that 2 out of these 5 neighbours are defaulted (class 1), whereas 3 did not (class 0). The kNN algorithm would then conclude that the new datapoint should be classified as non-defaulted.

An extension of this procedure involves taking into account the distance between each neighbour and the currently considered point by calculating the exact distance between two points and multiplying each neighbour's value (1 for default and 0 for non-default) by the distance's inverse. It ensures that closer neighbours weigh more than the ones that are further away. In particular, the general equation for point $\vec{p}$ looks as follows:

$$\frac{\Sigma_{r=1}^{k} class \times \frac{1}{d(\vec{p},\vec{r})}}{\Sigma_{r=1}^{k} \frac{1}{d(\vec{p},\vec{r})}}, \quad r = 1,2,\dots,k \tag{10}$$

where k is the number of nearest neighbours used.

As we describe below, we use the kNN algorithm for two main tasks: varying the default rates and the applicant's features in the original sample to create synthetic datasets that represent different stressed scenarios.

4.2. Framework for generating default rate

Default rates tend to be higher during economic downturns as often people have less money to cover their ongoing loans (Mian & Sufi, 2009). Therefore, in order to assess how the model would perform during such events, we change the classification of certain loans to the opposite — in this case, from non-defaulted to defaulted. The pipeline for changing the classification is as follows:

1. **Obtain default probabilities.** Apply kNN with the chosen number of neighbours on every datapoint, using Eq. (10), in order to get a probability $p : \ 0 \leq p \leq 1$ of every point belonging to a given class based on its neighbours. It can help to conclude whether a point is surrounded by other points from the same class or not. It is all done in m-dimensional space, where m is the number of features in the out-of-time set.
2. **Find questionable observations.** Conclude which observations lie close to the decision boundary according to the kNN algorithm. In principle, the closer the observation is to $p = 0.5$, the more questionable it is because at this point we are at least sure (provided that we would not have this information in the first place) to which class such observation would belong. Depending on the final default rate that we are aiming for, we can increase the range, for example for $0.4 \leq p \leq 0.6, 0.3 \leq p \leq 0.7$ etc.
3. **Change the classification.** From the observations obtained above, choose the ones that were initially classified as non-defaulted and classify them as defaulted, which in effect increases the default rate from the initial out-of-time set.

4.3. Framework for applicant's features

A similar logic can be applied for changing the independent variables in the original sample. The key in this case is to find observations that would be least likely to change their classification due to the changes in the feature values. Combined with the approach for increasing default rates, it leads to a creation of completely new synthetic out-of-time sets, depending on the hyperparameters used. The procedure for finding certain observations consists of 2 steps and is very similar to the ones described above. It mainly differs by step (2), in which we aim to find observations whose calculated probability of default p is either as close to 0 as possible (for the non-defaulted case) or as close to 1 as possible (for the defaulted case). Having found n such observations, we need to follow the steps outlined below.

1. **Calculate the ranges for each variable.** Find the minimum $\min f_j$ and maximum $\max f_j$ of each feature f_j, $j = 1,2,\dots,m$ from the set of all observations x_i, $i = 1,2,\dots,n$.
2. **Generate new observations.** For each feature f_j of every observation x_i, generate a new value from the range $\min f_j$, $\max f_j$ and replace the original value with it.

In the end, a newly created set has observations with perturbed features and unchanged classifications. Optionally, the classification could be perturbed as well using the procedure from the previous subsection.

5. Empirical results

5.1. Data description

We apply the proposed methods to a dataset provided by Nationwide Building Society on 61,239 UK unsecured personal loans issued from June 2014 to July 2015. It contains applicants' characteristics at the time at which they apply for loans and sample weights that should be applied to each observation. The reported percentage of default is 6.8%, which represents 4168 loans. The dependent variable is coded as 1 if the borrower is considered as default and 0 otherwise. For confidential reasons, we cannot share the description of the independent variables for this dataset.

The logistic regression and the SGB are estimated on 50 explanatory variables. We train the models on the data observed between June 2014 and May 2015 (in-sample) and assess the quality of the models on the out-of-time sample from June to July 2015. We obtain 41,711 loans in the training set, 10,101 in the out-of-sample set and 9427 in the out-of-time set.

In this section we summarize the results that we obtained by applying data perturbation and counterfactuals to the described dataset. We follow the analysis structure conducted by the Bank of England (Bracke et al., 2019) on explainability for scoring models by comparing the logistic regression (logit) and the SGB (Friedman, 2002) methods. Hyperparameters were not tuned for models — we rely on scikit-learn (Pedregosa et al., 2011) and XGBoost (Chen, He, Benesty, Khotilovich, & Tang, 2015) default parameter values (version 0.90).

5.2. Data perturbation

Perturbing the data can help financial institutions in performing a stress scenario analysis, instead of stress testing (Bellotti & Crook, 2013), when data is collected over a short period of time and, therefore, macroeconomic variables cannot be used. The percentage of defaulted loans in the Nationwide dataset is 6.8%.

Higher probabilities of default for perturbed out-of-time sets are obtained by changing the classification of good observations inside the probability intervals. These probabilities are obtained using kNN, as explained in Section 4, and take an approximate runtime of 87 s to generate. They appear as follows (out-of-sample and out-of-time, respectively):

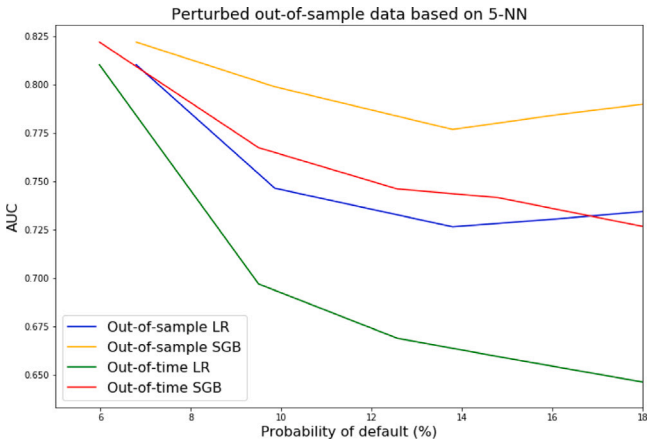

Fig. 2. AUC against PD for perturbed datasets.

- [0.25, 0.75] - the probability of default increases to 9.9 and 9.4%
- [0.15, 0.85] - the probability of default increases to 13.8 and 12.6%
- [0.10, 0.90] - the probability of default increases to 16.1 and 14.8%
- [0.06, 0.94] - the probability of default increases to 20.0 and 19.0%

Fig. 2 below shows how *AUC* decreases as the probability of default in the perturbed out-of-time set increases for both logistic regression and SGB. The same can be said about the perturbed out-of-sample set. However, it can be seen that the drop for out-of-sample set is much smaller for both models. One of the reasons for this might be due to a higher initial probability of default of the out-of-sample set, which is closer to the probability of default of the in-sample set (7%). Since out-of-sample set seems to have characteristics more similar to the in-sample set than the out-of-time set, the results on the perturbed data are also better for it.

The drop in performance for both models is the most significant for the initial increase in probability of default. This is due to the overlap of datapoints now found with both classes (non-default and default). As we perturbed the data to have higher ratio of defaulters, the distributions for the two classes now have a degree of overlap. Essentially, ML techniques have a harder time classifying datapoints when feature values are of an ambiguous class association (Prati, Batista, & Monard, 2004). Testing on perturbed data (with higher default ratios) results in more false positives and false negatives thus reducing the AUC value. Overall, the drop is higher for the logistic regression showing that it is less robust than SGB. The initial datapoints in the figure reflect AUC values for logistic regression and SGB prior to perturbation of the Nationwide dataset (baseline performance is also stated in Table 1).

Note, as we use kNNs in the perturbation process, the value of k significantly determines the appearance of the new decision boundary. At higher values of k we can expect the decision boundary to smooth out, while at lower values of k we can expect a more flexible decision boundary (James, Witten, Hastie, & Tibshirani, 2013). As increasing k also impacts the ratio of increased defaulters, we can expect a decrease in model performance as the rate of false positives and false negatives increase. Importantly, As our method relies on the returned probabilities p based on the setting of k for determining which points to set to default, it is expected that with alternative settings of k we would rely on different probabilities as the decision boundary smooths out at higher k values. In our experiments, we only selected a small valuation of k to observe an increase in defaulters ($k = 5$) as our primary concern was experimenting with the various probability ranges. As the exploration of higher k values has a notable impact on the decision boundary, we propose this as an avenue for future directions and further exploration of the proposed work.

Apart from investigating how the performance of the model is affected, we also checked whether the explanations of decisions made by the model are the same for the original and perturbed data. For this task, we used the data obtained by changing applicant's features as explained in Section 4.3. In particular, we perturb the features of non-defaulted applicants, for which the value obtained in Eq. (10) is smaller than 0.05. We have used Shapley values, and in particular their variation called Shapley Additive Explanations (SHAP) (Lundberg & Lee, 2017), which measure the average impact on model output by variable magnitude. The comparison for the original out-of-time set and the augmented one are presented in Fig. 3 below where these values represent the magnitude of variable contribution to the outcome. It

Table 1
AUC for Logit and SGB models on the Nationwide dataset.

Dataset	Model	
	Logit	SGB
Nationwide	0.81	0.82

Table 2
Average robustness scores for selected features on the Nationwide dataset. Approximate running time for Logit was 319 s, SGB was 365 s.

Nationwide	Model	
	Logit	SGB
variable01	5.51 ± 1.72	10.1 ± 2.45
variable02	4.88 ± 1.30	10.1 ± 2.41
variable03	7.02 ± 1.84	10.1 ± 2.42
variable04	5.92 ± 3.09	9.90 ± 2.51
variable05	7.76 ± 1.91	9.84 ± 2.51
variable06	6.45 ± 2.78	9.11 ± 2.79
variable07	5.16 ± 3.31	10.4 ± 2.33
variable08	5.27 ± 1.16	10.3 ± 2.43
variable09	7.50 ± 1.97	10.0 ± 2.45
variable10	5.72 ± 3.55	9.71 ± 2.56
variable11	6.97 ± 1.78	9.78 ± 2.57
variable12	7.31 ± 3.30	9.68 ± 2.56
variable13	6.27 ± 3.84	9.94 ± 2.49
variable14	6.49 ± 2.11	9.47 ± 2.66
variable15	6.21 ± 2.49	10.2 ± 2.44
variable16	5.25 ± 2.01	9.83 ± 2.54
variable17	5.44 ± 1.52	9.84 ± 2.54
variable18	6.18 ± 2.23	7.40 ± 1.99
variable19	8.44 ± 2.48	10.2 ± 2.41
variable20	7.62 ± 4.32	10.0 ± 2.45
variable21	6.90 ± 3.29	10.2 ± 2.36
variable22	6.10 ± 2.14	11.1 ± 2.19
variable23	7.62 ± 2.17	7.31 ± 2.02
variable24	5.81 ± 3.25	9.85 ± 2.53
variable25	6.77 ± 1.21	9.85 ± 2.53
variable26	7.71 ± 3.07	9.91 ± 2.51
variable27	6.76 ± 3.59	9.99 ± 2.46
variable28	6.43 ± 2.32	10.1 ± 2.44
variable29	5.42 ± 1.96	10.2 ± 2.35
variable30	5.72 ± 1.21	10.6 ± 2.24
variable31	7.93 ± 1.41	10.0 ± 2.45
variable32	8.06 ± 3.59	9.91 ± 2.51
variable33	8.24 ± 3.28	12.3 ± 2.53
variable34	7.36 ± 3.45	10.0 ± 2.46
variable35	6.36 ± 3.30	10.4 ± 2.31
variable36	8.55 ± 2.23	9.94 ± 2.49
variable37	6.55 ± 4.52	9.15 ± 2.78
variable38	7.87 ± 2.46	9.92 ± 2.50
variable39	6.34 ± 3.17	9.93 ± 2.49
variable40	8.08 ± 2.27	10.6 ± 2.24
variable41	5.79 ± 3.76	10.2 ± 2.36
variable42	7.92 ± 1.83	9.21 ± 2.75
variable43	5.96 ± 4.29	10.0 ± 2.46
variable44	7.62 ± 1.93	9.83 ± 2.54
variable45	7.54 ± 2.46	10.1 ± 2.41
variable46	7.88 ± 3.37	11.3 ± 2.21
variable47	4.62 ± 3.71	9.89 ± 2.51
variable48	6.09 ± 1.01	10.6 ± 2.25
variable49	9.34 ± 1.71	9.77 ± 2.57
variable50	6.68 ± 5.37	10.0 ± 2.46

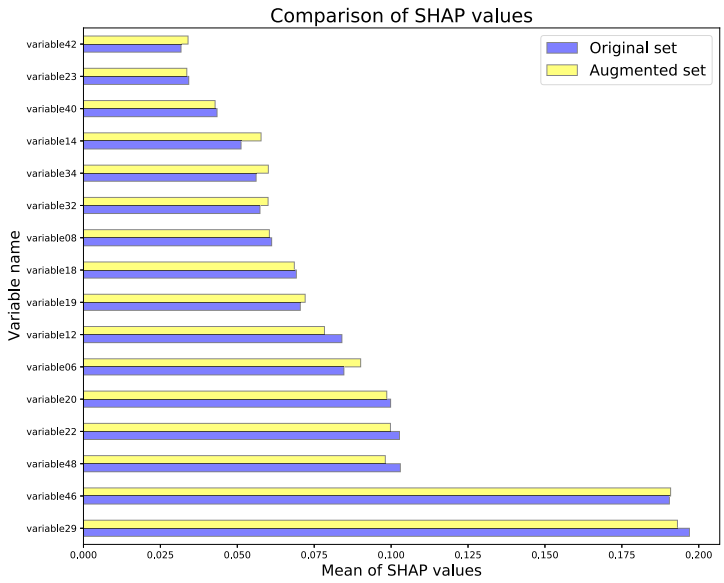

Fig. 3. Shapley values for the original out-of-time set.

can be seen that the predictive power of respective variables does not change significantly after perturbing the original data.

5.3. Counterfactual robustness

We evaluate the *robustness score* by looking at the average robustness of individual features and comparing the scores between models, we then look at the robustness of a model where counterfactuals are generated with individual feature ranges acting as constraints. We train blackbox models on the Nationwide credit application dataset with the goal here being to provide another performance metric to compare models, analyse model robustness on individual features, and to further analyse model robustness on particular feature ranges. As we see this as an important contribution to global interpretability analysis of ML models in the financial sector.

In Table 1 we can see the predictive power of the respective blackbox classifiers. In general, common metrics of model performance are model accuracy and the AUC. Initial observations indicate that SGB is the superior model for credit scoring with respect to both metrics.

5.3.1. Model robustness on individual features

To assess the robustness of individual features in a dataset, we generate counterfactuals on individual instances using only the value range of the selected feature in the genetic algorithm. In essence, any counterfactuals of an alternate classification for an instance x_i will be generated based on only the possible values of a feature f_i leaving all other values in x_i the same. For each feature, K-means clustering is utilized to select a sub-population that best represents the data. This provides a population of datapoints that is general and constrains counterfactual generation on the range of values of our selected feature. Counterfactuals are generated on this subset and from the population of counterfactuals we calculate the *robustness score*. The average *robustness scores* for each feature, trained with logistic regression and SGB, can be seen in Table 2.

The average feature *robustness score* for logistic regression and SGB models trained on the Nationwide dataset can be seen in Table 2. We see a similar pattern where individual feature robustness is better performing with SGB than logistic regression. The Nationwide dataset demonstrates on average, a larger overall *robustness score* among individual features. The logistic regression model demonstrates less certainty among the various features. We find the logistic regression model's most robust feature is `variable49` while the least robust is `variable02`. The SGB model is more consistent with feature *robustness scores* than with the logistic regression model. The most robust features are `variable22`, `variable46`, and `variable33`. The less robust features are `variable18` and `variable23` which comparatively are more robust with the SGB model than logistic regression. As an example, if a modeller had two new entries, customers with similar values for `variable33` and wildly different values for `variable23` and both entries have different classifications for the SGB model then further analysis of the entries is required to assess the predicted classifications.

5.3.2. Robustness with binned feature ranges as constraints

In the following evaluation we take the SGB model and observe the *robustness score* when constraints are applied to the individual feature ranges as seen in Algorithm 2. Here we compare the difference in *robustness scores* given 2 and 3 bins across all features in the Nationwide dataset. By increasing the bin count we can isolate particular feature ranges that may be less robust to perturbations and help the modeller understand and isolate areas of weakness in the model. The primary method of binning is equal-frequency binning, however we did encounter features which failed to create 3 bins with equal-frequency. In these cases, we defaulted to using equal-width binning.

From Tables 4 and 3 we see again the benefits of increased binning of individual features when training the SGB model on the Nationwide dataset. In looking at the 2 binned constraint, we see that feature `variable01` appears robust across both bin ranges 1 and 2. In the case of 3 bins, the corresponding feature has an average *robustness score* of 11.3, 15.8, and 5.28 for bins 1,2, and 3 respectively. The implication here is that the feature ranges seen with 2 bins does not capture the areas of less robustness that can be observed with increased binning. As we see, bin 3 with 3 bin partitioning is less robust for feature `variable01`. This increased partitioning of the feature ranges shows potential problem areas not seen with 2 bins. The range corresponding to bin 3 for feature `variable01` can be identified in Table 5. This tells the modeller that this range is an area of caution for model prediction.

We expect different measures of robustness for differing bin sizes, as increasing the bin size allows for more granular measures of particular feature ranges. In essence, if a specific feature value has a correlation with adversarial attacks, more discrete binning could potentially capture that fact with a corresponding low *robustness score*. The presence of outliers can impact the *robustness score* as well, as constraints for the GA will include outlier values if they are in fact the highest or lowest

Table 3
Average robustness scores for the Nationwide dataset on 3 bin feature constrained counterfactuals. Approximate running time for 3 binned SGB was 565 s.

Nationwide	3 bins			
	bin 1	bin 2	bin 3	avg.
variable01	11.3 ± 2.21	15.8 ± 2.64	5.28 ± 1.42	10.8
variable02	13.1 ± 2.54	9.59 ± 2.51	12.2 ± 1.51	11.6
variable03	9.1 ± 1.14	6.32 ± 0.80	9.34 ± 2.82	8.25
variable04	14.0 ± 4.14	11.8 ± 2.87	8.59 ± 1.11	11.4
variable05	8.42 ± 1.90	11.9 ± 1.92	1.65 ± 0.16	7.32
variable06	10.3 ± 2.10	9.81 ± 1.88	1.65 ± 0.16	7.28
variable07	9.92 ± 2.37	12.2 ± 1.77	1.66 ± 0.16	7.95
variable08	9.86 ± 2.29	11.4 ± 1.59	1.65 ± 0.16	7.66
variable09	10.2 ± 2.11	12.3 ± 1.78	1.65 ± 0.16	8.09
variable10	13.2 ± 2.12	11.8 ± 3.36	8.67 ± 1.40	11.2
variable11	10.3 ± 0.27	8.47 ± 1.95	13.8 ± 2.40	10.8
variable12	14.1 ± 2.66	11.7 ± 3.65	9.18 ± 1.32	11.7
variable13	10.4 ± 2.17	4.98 ± 0.86	3.14 ± 0.26	6.17
variable14	10.3 ± 2.17	4.28 ± 0.69	3.15 ± 0.26	5.93
variable15	0.0 ± 0.0	4.98 ± 2.34	5.92 ± 0.84	3.63
variable16	3.93 ± 0.53	0.0 ± 0.0	8.31 ± 2.28	4.08
variable17	0.64 ± 0.44	5.81 ± 1.65	6.06 ± 0.81	4.17
variable18	0.87 ± 0.78	8.02 ± 1.38	12.1 ± 1.99	7.02
variable19	0.03 ± 0.03	4.18 ± 2.31	6.23 ± 0.84	3.48
variable20	8.71 ± 2.02	1.03 ± 0.08	0.0 ± 0.0	3.25
variable21	7.60 ± 2.67	16.0 ± 2.21	1.26 ± 0.11	8.30
variable22	8.75 ± 2.10	7.78 ± 1.08	6.90 ± 0.29	7.81
variable23	9.20 ± 1.91	0.0 ± 0.0	7.01 ± 0.12	5.40
variable24	6.41 ± 1.08	0.0 ± 0.0	8.91 ± 2.51	5.10
variable25	4.63 ± 4.40	6.61 ± 0.90	7.55 ± 1.54	6.26
variable26	8.66 ± 2.23	11.6 ± 1.73	16.3 ± 0.80	12.2
variable27	0.0 ± 0.0	0.0 ± 0.0	5.92 ± 0.84	1.97
variable28	14.6 ± 0.68	10.5 ± 1.18	10.3 ± 2.30	11.8
variable29	4.65 ± 0.77	4.38 ± 0.44	9.70 ± 2.54	6.24
variable30	0.0 ± 0.0	0.0 ± 0.0	5.92 ± 0.84	1.97
variable31	3.82 ± 3.82	0.0 ± 0.0	5.92 ± 0.84	3.24
variable32	2.41 ± 0.47	0.0 ± 0.0	8.24 ± 224	3.55
variable33	1.99 ± 0.31	8.91 ± 1.48	8.60 ± 1.88	6.50
variable34	2.29 ± 0.25	9.81 ± 1.24	8.35 ± 2.16	6.82
variable35	2.72 ± 0.23	9.22 ± 1.15	8.90 ± 1.99	6.95
variable36	0.0 ± 0.0	6.77 ± 0.75	7.55 ± 1.54	4.77
variable37	11.2 ± 3.03	9.33 ± 2.00	10.7 ± 3.15	10.4
variable38	10.4 ± 2.38	9.59 ± 1.35	17.1 ± 0.24	12.3
variable39	6.31 ± 0.94	16.3 ± 0.12	8.48 ± 1.74	10.3
variable40	13.5 ± 1.45	0.0 ± 0.0	10.8 ± 4.01	8.12
variable41	1.41 ± 0.75	6.87 ± 1.48	6.06 ± 0.81	4.78
variable42	7.32 ± 1.04	13.1 ± 0.29	8.81 ± 2.01	9.75
variable43	10.7 ± 1.46	10.3 ± 1.27	17.7 ± 2.25	12.9
variable44	1.28 ± 0.16	0.0 ± 0.0	8.91 ± 2.35	3.39
variable45	0.0 ± 0.0	7.57 ± 1.59	12.2 ± 1.99	6.58
variable46	1.91 ± 0.18	3.23 ± 0.45	8.42 ± 1.82	4.52
variable47	2.17 ± 2.17	2.04 ± 0.52	6.23 ± 0.84	3.48
variable48	8.17 ± 1.54	9.82 ± 1.49	17.4 ± 0.82	11.8
variable49	12.4 ± 2.56	0.42 ± 0.07	0.0 ± 0.0	4.26
variable50	14.2 ± 2.45	10.3 ± 2.11	1.06 ± 0.05	8.58

Table 4
Average robustness scores for the Nationwide dataset on 2 bin feature constrained counterfactuals. Approximate running time for 2 binned SGB 503 s.

Nationwide	2 bin		
	bin 1	bin 2	avg.
variable01	11.1 ± 3.03	13.5 ± 2.24	12.3
variable02	6.52 ± 1.74	16.4 ± 2.62	11.4
variable03	11.8 ± 1.61	13.5 ± 3.29	12.6
variable04	10.7 ± 2.62	10.4 ± 2.64	10.6
variable05	12.1 ± 5.64	8.65 ± 1.54	10.4
variable06	9.65 ± 3.06	8.16 ± 1.7	8.91
variable07	8.99 ± 4.38	9.13 ± 1.26	9.06
variable08	7.88 ± 3.83	10.0 ± 0.92	8.98
variable09	15.2 ± 10.6	8.98 ± 1.44	12.1
variable10	10.5 ± 3.30	15.2 ± 2.51	12.9
variable11	15.1 ± 1.79	12.7 ± 2.16	13.8
variable12	16.0 ± 2.03	15.0 ± 2.51	15.5
variable13	15.2 ± 3.40	5.57 ± 0.80	10.3
variable14	13.1 ± 3.09	5.89 ± 0.86	9.52
variable15	0.0 ± 0.0	12.01 ± 3.13	6.01
variable16	12.3 ± 2.21	16.5 ± 6.89	14.4
variable17	15.1 ± 2.04	8.53 ± 1.62	11.8
variable18	11.1 ± 2.81	4.44 ± 0.98	7.81
variable19	12.1 ± 3.35	1.97 ± 0.20	7.05
variable20	15.2 ± 2.23	0.0 ± 0.0	7.61
variable21	11.1 ± 3.28	1.57 ± 0.23	6.36
variable22	10.7 ± 2.08	7.75 ± 1.26	9.24
variable23	5.98 ± 1.23	13.2 ± 6.47	9.61
variable24	10.8 ± 1.26	9.95 ± 3.76	10.4
variable25	0.0 ± 0.0	12.0 ± 3.13	6.01
variable26	11.6 ± 1.68	13.9 ± 3.54	12.7
variable27	0.03 ± 0.03	12.0 ± 3.13	6.02
variable28	15.7 ± 2.64	12.7 ± 3.06	14.2
variable29	4.34 ± 0.68	14.1 ± 3.14	9.21
variable30	0.0 ± 0.0	12.9 ± 3.18	6.45
variable31	14.3 ± 2.73	12.0 ± 3.13	13.2
variable32	10.2 ± 1.01	11.1 ± 2.91	10.6
variable33	10.8 ± 1.35	8.3 ± 2.63	9.56
variable34	9.96 ± 1.27	6.08 ± 2.32	8.02
variable35	7.69 ± 0.92	7.53 ± 2.79	7.61
variable36	0.18 ± 0.18	12.0 ± 3.13	6.09
variable37	12.8 ± 2.39	12.0 ± 3.13	12.4
variable38	8.13 ± 1.54	17.3 ± 2.78	12.7
variable39	12.1 ± 1.83	13.1 ± 3.07	12.6
variable40	11.0 ± 1.19	9.96 ± 3.60	10.4
variable41	14.2 ± 2.50	16.1 ± 6.81	15.1
variable42	9.76 ± 1.78	7.64 ± 2.51	8.70
variable43	12.9 ± 2.90	12.5 ± 2.91	12.7
variable44	1.66 ± 0.09	8.59 ± 3.37	5.12
variable45	11.1 ± 2.81	4.00 ± 0.80	7.58
variable46	2.02 ± 0.21	7.48 ± 2.66	4.75
variable47	12.1 ± 3.35	1.86 ± 0.24	6.99
variable48	9.79 ± 1.66	13.6 ± 3.08	11.7
variable49	11.7 ± 3.06	12.9 ± 1.10	12.3
variable50	11.3 ± 3.07	4.51 ± 0.55	7.92

value for the feature range. This is a consequence of binning being performed by equal-frequency/equal-width binning. The use of K-means clustering does prevent selection of outliers for the sub-populations used in calculating the *robustness score* for a given feature range, and the fitness functions by design would avoid using outlier values in generating counterfactuals as they would be significantly different from the instances in the subpopulation.

5.4. Counterfactual robustness and stress scenario testing

The following experiments seeks to combine the counterfactual *robustness score* with the data perturbation methodology using kNNs. The purpose here is to generate possible stress scenarios on the data by increasing the ratio of bads (default rate) in the data and perturbing features in order to observe model performance during what could be a financial crisis. The counterfactual *robustness score* provides insight into what features are most at risk and which value ranges modellers should directly observe in future cases. The models evaluated here is exclusively SGB, using the same hyper-parameters as used in Experiments 5.3.1 with training being done on the Nationwide dataset. As in Section 5.3, SGB models are evaluated on both perturbed Nationwide datasets with increased default rate only and on datasets with increased default rates with perturbations in feature values. For stress scenario testing, the GA generates counterfactuals based on the reasoning of the trained model and takes as input the test set. In order to analyse the robustness of the model, we first train the model on the perturbed datasets with increased bad rates and/or perturbed features, followed by testing on unperturbed datasets (original Nationwide dataset). By following this pipeline, we can observe model behaviour given data representing stress scenarios, and the counterfactuals generated are based on stressed model's reasoning on the test set. The *robustness score* in this context is a reflection of stressed model reasoning on non-stressed data.

The following evaluation is done on 4 separate versions of the Nationwide dataset. Using the kNNs, we increase the positive bad rate

Table 5
Feature ranges for the Nationwide dataset.

Nationwide	3 bins		
Value ranges	bin 1	bin 2	bin 3
variable01	$-1.77e^{-15} < x \leq 6.0$	$6.0 < x \leq 15.0$	$15.0 < x \leq 172$
variable02	$1.0 < x \leq 3.0$	$3.0 < x \leq 5.0$	$5.0 < x \leq 24.0$
variable03	$-998.0 < x \leq 0.0$	$0.0 < x \leq 37.0$	$37.0 < x \leq 109$
variable04	$-4.82e^{4} < x \leq -8.01e^{3}$	$-8.01e^{3} < x \leq -3.23e^{3}$	$-3.23e^{3} < x \leq 3.30e^{5}$
variable05	$0.0 < x \leq 196.71$	$-196.71 < x \leq 1326.5$	$1326.5 < x \leq 3.34e^{5}$
variable06	$0.0 < x \leq 211.13$	$211.13 < x \leq 1487.06$	$1487.06 < x \leq 3.34e^{5}$
variable07	$0.0 < x \leq 236.61$	$236.61 < x \leq -1.37e^{3}$	$-1.37e^{3} < x \leq 3.34e^{5}$
variable08	$0.0 < x \leq 263.30$	$263.30 < x \leq 1552.48$	$1552.48 < x \leq 3.34e^{5}$
variable09	$-8.52e^{3} < x \leq 187.84$	$187.84 < x \leq 1.49e^{3}$	$1.49e^{3} < x \leq 3.34e^{5}$
variable10	$-5710.0 < x \leq -340.0$	$-340.0 < x \leq -91.58$	$-91.58 < x \leq 0.0$
variable11	$-9.99e^{5} < x \leq -721.06$	$-721.06 < x \leq -341.81$	$-341.81 < x \leq 0.0$
variable12	$-1.77e^{-15} < x \leq 7.99$	$7.99 < x \leq 19.0$	$19.0 < x \leq 172.0$
variable13	$0.0 < x \leq 3.64$	$3.64 < x \leq 333.25$	$333.25 < x \leq 3.34e^{5}$
variable14	$0.0 < x \leq 4.09$	$4.09 < x \leq 382.84$	$382.84 < x \leq 3.34e^{5}$
variable15	$0.99 < x \leq 4.0$	$4.0 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable16	$-1.00e^{6} < x \leq -6.63e^{5}$	$-6.63e^{5} < x \leq -3.27e^{5}$	$-3.27e^{5} < x \leq 9214.00$
variable17	$0.99 < x \leq 4.0$	$4.0 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable18	$0.99 < x \leq 4.0$	$4.0 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable19	$0.99 < x \leq 3.99$	$3.99 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable20	$-1.00e^{6} < x \leq 1.65e^{5}$	$1.65e^{5} < x \leq 1.33e^{6}$	$1.33e^{6} < x \leq 2.49e^{6}$
variable21	$1.0 < x \leq 8.0$	$8.0 < x \leq 9.0$	$9.0 < x \leq 10.0$
variable22	$-9.99e^{5} < x \leq 100.0$	$100.0 < x \leq 145.0$	$145.0 < x \leq 701.0$
variable23	$-1.00e^{6} < x \leq -6.66e^{5}$	$-6.66e^{5} < x \leq -3.33e^{5}$	$-3.33e^{5} < x \leq 1.0$
variable24	$-1.00e^{6} < x \leq -6.63e^{5}$	$-6.63e^{5} < x \leq -3.27e^{5}$	$-3.27e^{5} < x \leq 9333.0$
variable25	$0.99 < x \leq 4.0$	$4.0 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable26	$-9.99e^{5} < x \leq 2.0$	$2.0 < x \leq 6.0$	$6.0 < x \leq 87.0$
variable27	$0.99 < x \leq 4.0$	$4.0 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable28	$-9.99e^{5} < x \leq 89.0$	$89.0 < x \leq 118.0$	$118.0 < x \leq 9972.0$
variable29	$-9.99e^{5} < x \leq 10.0$	$10.0 < x \leq 56.0$	$56.0 < x \leq 463.0$
variable30	$0.99 < x \leq 3.99$	$3.99 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable31	$-1.00e^{6} < x \leq -6.66e^{5}$	$-6.66e^{5} < x \leq -3.33e^{5}$	$-3.33e^{5} < x \leq 52.0$
variable32	$1.00e^{6} < x \leq -6.66e^{5}$	$-6.66e^{5} < x \leq -3.33e^{5}$	$-3.33e^{5} < x \leq 92.0$
variable33	$-9.99e^{5} < x \leq 0.0$	$0.0 < x \leq 1.0$	$1.0 < x \leq 28.0$
variable34	$-9.99e^{5} < x \leq 0.0$	$0.0 < x \leq 1.0$	$1.0 < x \leq 90.0$
variable35	$-9.99e^{5} < x \leq 0.0$	$0.0 < x \leq 1.0$	$1.0 < x \leq 90.0$
variable36	$0.99 < x \leq 4.0$	$4.0 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable37	$-9.99e^{5} < x \leq 0.0$	$0.0 < x \leq 1.0$	$1.0 < x \leq 19.0$
variable38	$-9.99e^{5} < x \leq 144.0$	$144.0 < x \leq 303.66$	$303.66 < x \leq 15335.0$
variable39	$-9.99e^{5} < x \leq 96.0$	$96.0 < x \leq 100.0$	$100.0 < x \leq 9678.0$
variable40	$-1.00e^{6} < x \leq -6.65e^{5}$	$-6.55e^{5} < x \leq -3.11e^{5}$	$-3.11e^{5} < x \leq 3.28e^{4}$
variable41	$0.99 < x \leq 4.0$	$4.0 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable42	$-9.99e^{5} < x \leq 95.0$	$95.0 < x \leq 104.0$	$104.0 < x \leq 9570.0$
variable43	$-9.99e^{5} < x \leq 19.0$	$19.0 < x \leq 43.0$	$43.0 < x \leq 596.0$
variable44	$-1.00e^{6} < x \leq -6.66e^{5}$	$-6.66e^{5} < x \leq -3.33e^{5}$	$-3.33e^{5} < x \leq 71.99$
variable45	$0.99 < x \leq 4.00$	$4.00 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable46	$-9.99e^{5} < x \leq 1.0$	$1.0 < x \leq 2.0$	$2.0 < x \leq 51.0$
variable47	$0.99 < x \leq 3.99$	$3.99 < x \leq 7.0$	$7.0 < x \leq 10.0$
variable48	$-9.99e^{5} < x \leq 2.0$	$2.0 < x \leq 5.0$	$5.0 < x \leq 86.0$
variable49	$-0.024 < x \leq 8.0$	$8.0 < x \leq 16.0$	$16.0 < x \leq 24.0$
variable50	$0.0 < x \leq 38.0$	$38.0 < x \leq 102.0$	$102.0 < x \leq 440.0$

to 12% and 18%. KNNs are again used on the biased datasets to perturb features based on extracting the most robust ranges for variables for which an observation should have a certain classification. We use these robust ranges to randomize feature values for these observations and hence arrive at a new, perturbed out-of-time sample.

5.4.1. Baseline

In Table 5 The following ranges are provided for each feature. The two values in each bin indicate the lower and upper bound for that bin interval. The initial baseline is seen in Table where the bad rate in the data is initially 6%. The average *robustness scores* for 3 binned intervals can be observed in Table 3.

5.4.2. Stress scenarios

Nationwide 12% bad rate. In Table 6 we see the impact of increased bad rate has on the Nationwide dataset. We note that features `variable12` and `variable48` become more robust when more observations are labelled as bad. We also see certain features such as `variable15` and `variable31` become less robust with model prediction.

Nationwide 12% bad rate with perturbed features. In Table 7 we see the result of feature perturbation on the Nationwide data with 12% bad rate. It can be observed that the most notable change is a higher robustness for feature `variable36` at bin interval 1. Given that, the overall impact of feature data perturbation is minimal with this bias ratio for credit default.

Nationwide 18% bad rate. In Table 8 it can be observed that features `variable04` (increased robustness) and `variable31` (decreased robustness) change significantly from the baseline Nationwide dataset (see Table 3). The remaining features see little fluctuation in their average *robustness scores*.

Nationwide 18% bad rate with perturbed features:. The impact of the perturbed feature values on the Nationwide dataset with 18% bad rate is minimal with few actual changes in average robustness. Cases of increased robustness can be seen with feature `variable18`, as seen in Table 9.

Overall we notice that the robustness of certain features fluctuates with increased bad rate as well as with perturbations on the data.

Table 6
Average robustness scores for the Nationwide dataset with 12% bad rate. Approximate running time for 3 binned SGB was 559 s.

Nationwide (12%)	3 bins			
	bin 1	bin 2	bin 3	avg.
variarble01	11.1 ± 2.14	13.2 ± 3.18	6.13 ± 1.15	10.1
variable02	11.2 ± 2.64	9.69 ± 2.46	12.2 ± 1.51	11.0
variable03	9.10 ± 1.14	6.32 ± 0.80	12.0 ± 3.82	9.15
variable04	14.1 ± 4.11	9.79 ± 2.85	7.73 ± 1.54	10.5
variable05	8.68 ± 1.98	10.2 ± 1.67	1.65 ± 0.16	6.86
variable06	10.2 ± 2.04	11.9 ± 1.93	1.65 ± 0.16	7.95
variable07	8.11 ± 2.09	12.2 ± 1.77	1.66 ± 0.16	7.34
variable08	7.09 ± 1.53	9.45 ± 1.71	1.65 ± 0.16	6.06
variable09	10.2 ± 2.10	12.3 ± 1.78	1.65 ± 0.16	8.09
variable10	13.2 ± 2.12	9.18 ± 0.98	8.67 ± 1.40	10.3
variable11	10.3 ± 0.27	9.04 ± 1.66	13.8 ± 2.40	11.1
variable12	12.1 ± 3.10	14.2 ± 3.46	10.2 ± 0.35	12.1
variable13	10.3 ± 2.17	4.31 ± 0.90	3.14 ± 0.26	5.94
variable14	10.3 ± 2.17	4.28 ± 0.69	3.15 ± 0.26	5.93
variable15	0.96 ± 0.96	2.41 ± 0.89	5.92 ± 0.84	3.10
variable16	4.48 ± 1.60	0.0 ± 0.0	8.31 ± 2.28	4.26
variable17	0.89 ± 0.42	5.59 ± 1.73	6.06 ± 0.81	4.18
variable18	2.09 ± 1.39	8.17 ± 1.35	12.1 ± 1.99	7.48
variable19	$3.28e^{-05} \pm 3.28e^{-05}$	2.31 ± 0.45	6.23 ± 0.84	2.85
variable20	8.71 ± 2.02	1.03 ± 0.08	0.0 ± 0.0	3.25
variable21	7.60 ± 2.67	16.0 ± 2.21	1.26 ± 0.11	8.30
variable22	8.77 ± 2.08	7.78 ± 1.08	6.90 ± 0.29	7.82
variable23	9.20 ± 1.91	0.0 ± 0.0	7.00 ± 0.12	5.40
variable24	6.41 ± 1.08	0.0 ± 0.0	8.91 ± 2.51	5.10
variable25	0.21 ± 0.18	6.60 ± 0.94	7.55 ± 1.54	4.78
variable26	7.49 ± 1.72	11.6 ± 1.73	16.3 ± 0.80	11.8
variable27	0.01 ± 0.01	0.0 ± 0.0	5.92 ± 0.84	1.98
variable28	12.8 ± 1.87	10.5 ± 1.18	9.72 ± 2.40	11.0
variable29	4.65 ± 0.77	4.38 ± 0.44	8.97 ± 2.76	6.00
variable30	1.10 ± 1.10	0.0 ± 0.0	5.92 ± 0.84	2.34
variable31	0.0 ± 0.0	0.0 ± 0.0	5.92 ± 0.84	1.97
variable32	2.29 ± 0.53	0.0 ± 0.0	8.24 ± 2.24	3.51
variable33	1.99 ± 0.31	8.91 ± 1.48	8.60 ± 1.88	6.50
variable34	4.62 ± 2.35	9.81 ± 1.24	8.69 ± 1.99	7.71
variable35	2.45 ± 0.39	9.22 ± 1.15	8.62 ± 1.96	6.76
variable36	3.42 ± 3.41	6.64 ± 0.84	7.55 ± 1.54	5.87
variable37	8.63 ± 1.89	9.33 ± 2.00	10.7 ± 3.15	9.56
variable38	12.6 ± 2.01	8.54 ± 1.37	17.1 ± 0.24	12.7
variable39	6.31 ± 0.94	16.3 ± 0.12	8.48 ± 1.74	10.3
variable40	13.5 ± 1.45	0.0 ± 0.0	10.7 ± 4.04	8.08
variable41	2.73 ± 2.08	6.98 ± 1.42	6.06 ± 0.81	5.26
variable42	7.32 ± 1.04	13.1 ± 0.29	8.80 ± 2.01	9.75
variable43	10.6 ± 1.42	10.3 ± 1.27	17.7 ± 2.25	12.8
variable44	1.28 ± 0.16	0.0 ± 0.0	8.90 ± 2.35	3.39
variable45	0.47 ± 0.47	6.96 ± 1.72	12.1 ± 1.99	6.54
variable46	1.91 ± 0.18	3.23 ± 0.45	8.42 ± 1.82	4.52
variable47	0.15 ± 0.15	2.25 ± 0.47	6.23 ± 0.84	2.87
variable48	11.2 ± 2.56	9.82 ± 1.49	17.4 ± 0.82	12.8
variable49	12.3 ± 2.56	0.42 ± 0.07	0.03 ± 0.03	4.27
variable50	14.6 ± 1.84	10.4 ± 2.06	1.06 ± 0.05	8.72

Table 7
Average robustness scores for the Nationwide dataset with 12% bad rate and feature perturbations. Approximate running time for 3 binned SGB was 571 s.

Nationwide (12%)	3 bins			
Perturbed features	bin 1	bin 2	bin 3	avg.
variable01	11.1 ± 2.19	13.2 ± 2.59	6.13 ± 1.27	10.9
variable02	11.2 ± 2.94	9.69 ± 2.51	12.2 ± 1.51	11.3
variable03	9.10 ± 1.14	6.32 ± 0.80	12.0 ± 3.08	8.70
variable04	14.1 ± 3.57	9.79 ± 2.89	7.73 ± 1.55	12.0
variable05	8.68 ± 2.06	10.2 ± 1.92	1.65 ± 0.16	7.21
variable06	10.2 ± 6.36	11.9 ± 2.01	1.65 ± 1.37	9.12
variable07	8.11 ± 2.26	12.2 ± 1.50	1.66 ± 0.16	6.97
variable08	7.09 ± 2.13	9.45 ± 1.59	1.65 ± 0.16	7.77
variable09	10.2 ± 1.96	12.3 ± 1.78	1.65 ± 0.16	7.72
variable10	13.2 ± 2.12	9.18 ± 1.24	8.67 ± 1.40	10.4
variable11	10.3 ± 0.27	9.04 ± 1.66	13.8 ± 2.40	11.1
variable12	12.1 ± 2.66	14.2 ± 3.40	10.2 ± 0.35	12.8
variable13	10.3 ± 2.17	4.31 ± 0.86	3.14 ± 0.26	6.17
variable14	10.3 ± 2.17	4.28 ± 0.69	3.15 ± 0.26	5.93
variable15	0.96 ± 0.14	2.41 ± 1.42	5.92 ± 0.84	3.25
variable16	4.48 ± 0.61	0.0 ± 0.0	8.31 ± 2.28	3.82
variable17	0.89 ± 0.48	5.59 ± 1.54	6.06 ± 0.81	4.56
variable18	2.09 ± 1.24	8.17 ± 1.81	12.1 ± 1.99	6.61
variable19	$3.28e^{-5} \pm 0.78$	2.31 ± 2.10	6.23 ± 0.84	4.49
variable20	8.71 ± 2.02	1.03 ± 0.08	0.0 ± 0.0	3.25
variable21	7.60 ± 2.67	16.0 ± 2.21	1.26 ± 0.11	8.30
variable22	8.77 ± 1.94	7.78 ± 1.08	6.90 ± 0.29	8.53
variable23	9.20 ± 1.91	0.0 ± 0.0	7.00 ± 0.12	5.40
variable24	6.41 ± 1.08	0.0 ± 0.0	8.91 ± 2.51	5.10
variable25	0.21 ± 0.0	6.60 ± 0.44	7.55 ± 1.54	5.17
variable26	7.49 ± 1.70	11.6 ± 1.73	16.3 ± 0.80	12.7
variable27	0.01 ± 3.38	0.0 ± 0.0	5.92 ± 0.84	3.11
variable28	12.8 ± 0.68	10.5 ± 1.18	9.72 ± 2.30	11.8
variable29	4.65 ± 0.77	4.38 ± 0.44	8.97 ± 2.54	6.24
variable30	1.10 ± 0.01	0.0 ± 0.0	5.92 ± 0.84	1.97
variable31	0.0 ± 0.15	0.0 ± 0.0	5.92 ± 0.84	2.02
variable32	2.29 ± 0.77	0.0 ± 0.0	8.24 ± 2.24	3.83
variable33	1.99 ± 0.31	8.91 ± 1.48	8.60 ± 1.88	6.50
variable34	4.62 ± 0.25	9.81 ± 1.24	8.69 ± 1.99	6.93
variable35	2.45 ± 0.23	9.22 ± 1.15	8.62 ± 1.96	6.85
variable36	3.42 ± 0.0	6.64 ± 0.89	7.55 ± 1.54	4.93
variable37	8.63 ± 1.81	9.33 ± 2.00	10.7 ± 3.15	10.1
variable38	12.6 ± 2.01	8.54 ± 1.60	17.1 ± 0.24	12.6
variable39	6.31 ± 0.94	16.3 ± 0.12	8.48 ± 1.74	10.3
variable40	13.5 ± 1.45	0.0 ± 0.0	10.7 ± 4.09	8.03
variable41	2.73 ± 0.53	6.98 ± 1.27	6.06 ± 0.81	4.76
variable42	7.32 ± 1.28	13.1 ± 0.29	8.80 ± 2.01	9.33
variable43	10.6 ± 1.42	10.3 ± 1.27	17.7 ± 2.25	12.8
variable44	1.28 ± 0.16	0.0 ± 0.0	8.90 ± 2.35	3.39
variable45	0.47 ± 3.12	6.96 ± 1.66	12.1 ± 1.99	8.13
variable46	1.91 ± 0.18	3.23 ± 0.45	8.42 ± 1.82	4.52
variable47	0.15 ± 0.0	2.25 ± 0.56	6.23 ± 0.84	2.73
variable48	11.1 ± 1.97	9.82 ± 1.49	17.4 ± 0.82	11.6
variable49	12.3 ± 2.56	0.42 ± 0.08	0.03 ± 0.0	4.24
variable50	14.6 ± 1.84	10.4 ± 2.04	1.06 ± 0.05	8.73

In some cases significantly, however overall the SGB model retains robustness across the majority of features.

6. Conclusion

The implementation of ML techniques in the financial sector is still limited due to the lack of interpretability. To address this challenge, our paper proposes to merge two distinct approaches, that being counterfactuals generated from blackbox models and targeted data perturbation using kNNs, together creating an approach to perform stress scenario analysis for scoring models. The two approaches can be used separately or in conjunction.

The data perturbation methodology uses kNNs to identify entries which are in close proximity to entries of another classification for class reassignment. This method can be used to generate synthetic data that represent economic downturn when a limited time series is available. The generation of counterfactuals provides an added perspective into data regions that are robust to change or are weak to feature value perturbations. The two approaches combined provide a key stress scenario mechanism for blackbox model analysis.

We test these proposals on a UK dataset on unsecured personal loans. We show that our data perturbation method decreases AUC as the default ratio increases. We also demonstrate the insight of the robustness score derived from counterfactuals, where using counterfactual constraints enables us to compare the robustness of logistic regression and SGB across each feature in the Nationwide dataset. Our results demonstrate that SGB remains more robust to data perturbations than the industry benchmark of logistic regression. We also investigated the stressed scenario datasets provided by our data perturbation method and were able to identify the robustness of discrete feature ranges across all features. Our results indicate that with increasing bad rates, SGB still remains robust to perturbations that signify stress scenarios.

Table 8
Average robustness scores for the Nationwide dataset with 18% bad rate. Approximate running time for 3 binned SGB was 581 s.

Nationwide (18%)	3 bins			
	bin 1	bin 2	bin 3	avg.
variable01	16.9 ± 7.04	13.2 ± 3.18	5.21 ± 1.45	11.8
variable02	12.8 ± 2.60	9.63 ± 2.49	12.2 ± 1.51	11.5
variable03	9.10 ± 1.14	6.32 ± 0.80	10.6 ± 3.08	8.70
variable04	16.5 ± 3.58	11.8 ± 2.86	7.73 ± 1.54	12.0
variable05	8.93 ± 1.79	13.9 ± 3.48	1.65 ± 0.16	8.17
variable06	8.11 ± 2.04	11.9 ± 1.93	1.65 ± 0.16	7.23
variable07	10.3 ± 2.13	12.2 ± 1.77	1.66 ± 0.16	8.10
variable08	10.1 ± 2.13	11.4 ± 1.59	1.65 ± 0.16	7.77
variable09	9.90 ± 2.31	12.3 ± 1.78	1.65 ± 0.16	7.96
variable10	13.2 ± 2.12	9.18 ± 0.98	8.16 ± 1.27	10.1
variable11	10.3 ± 0.27	9.04 ± 1.66	13.8 ± 2.40	11.1
variable12	14.1 ± 2.66	18.3 ± 3.84	10.2 ± 0.35	14.2
variable13	10.3 ± 2.17	4.98 ± 0.86	3.14 ± 0.26	6.17
variable14	10.3 ± 2.17	4.28 ± 0.69	3.15 ± 0.26	5.93
variable15	1.64 ± 1.64	10.4 ± 6.07	5.92 ± 0.84	6.01
variable16	3.85 ± 0.51	0.0 ± 0.0	8.31 ± 2.28	4.05
variable17	1.46 ± 0.97	7.35 ± 1.30	6.06 ± 0.81	4.96
variable18	0.02 ± 0.02	7.96 ± 1.47	12.1 ± 1.99	6.72
variable19	0.97 ± 0.85	5.95 ± 3.65	6.23 ± 0.84	4.38
variable20	8.71 ± 2.02	1.03 ± 0.08	0.0 ± 0.0	3.25
variable21	7.60 ± 2.67	16.0 ± 2.21	1.26 ± 0.11	8.30
variable22	8.77 ± 2.08	7.78 ± 1.08	6.90 ± 0.29	7.82
variable23	9.20 ± 1.91	0.0 ± 0.0	7.01 ± 0.12	5.40
variable24	6.41 ± 1.08	0.0 ± 0.0	8.91 ± 2.51	5.10
variable25	1.32 ± 1.32	6.15 ± 1.09	7.33 ± 1.64	4.93
variable26	8.46 ± 2.32	11.6 ± 1.73	16.3 ± 0.80	12.1
variable27	0.46 ± 0.46	0.0 ± 0.0	5.92 ± 0.84	2.13
variable28	12.7 ± 1.93	10.5 ± 1.18	10.3 ± 2.30	11.2
variable29	4.65 ± 0.77	4.38 ± 0.44	9.70 ± 2.54	6.24
variable30	0.01 ± 0.01	0.0 ± 0.0	5.92 ± 0.84	1.97
variable31	0.08 ± 0.06	0.0 ± 0.0	5.92 ± 0.84	2.00
variable32	2.69 ± 0.44	0.0 ± 0.0	8.24 ± 2.24	3.64
variable33	1.99 ± 0.31	8.91 ± 1.48	8.60 ± 1.88	6.51
variable34	2.29 ± 0.25	9.81 ± 1.24	8.66 ± 2.01	6.92
variable35	2.72 ± 0.23	9.22 ± 1.15	8.62 ± 1.96	6.85
variable36	0.0 ± 0.0	8.10 ± 1.16	7.55 ± 1.54	5.22
variable37	10.3 ± 1.81	9.33 ± 2.00	10.7 ± 3.15	10.1
variable38	12.6 ± 2.00	10.2 ± 1.73	17.1 ± 0.24	13.3
variable39	6.31 ± 0.94	16.3 ± 0.12	8.48 ± 1.74	10.3
variable40	13.5 ± 1.45	0.0 ± 0.0	10.8 ± 3.99	8.13
variable41	0.48 ± 0.34	7.13 ± 1.35	6.06 ± 0.81	4.56
variable42	8.39 ± 1.76	13.1 ± 0.29	8.80 ± 2.01	10.1
variable43	10.6 ± 1.42	10.3 ± 1.27	17.7 ± 2.25	12.8
variable44	1.28 ± 0.16	0.0 ± 0.0	8.91 ± 2.35	3.39
variable45	1.19 ± 1.12	7.62 ± 1.58	12.1 ± 1.99	7.00
variable46	1.91 ± 0.18	3.23 ± 0.45	8.42 ± 1.82	4.52
variable47	0.0 ± 0.0	1.81 ± 0.37	6.23 ± 0.84	2.68
variable48	9.29 ± 1.97	9.82 ± 1.49	17.4 ± 0.82	12.1
variable49	12.3 ± 2.56	0.42 ± 0.07	0.0 ± 0.0	4.26
variable50	14.6 ± 1.84	10.5 ± 2.04	1.06 ± 0.05	8.73

Table 9
Average robustness scores for the Nationwide dataset with 18% bad rate and feature perturbations. Approximate running time for 3 binned SGB was 573 s.

Nationwide (18%)	3 bins			
Perturbed features	bin 1	bin 2	bin 3	avg.
variable01	9.01 ± 2.34	13.2 ± 3.18	6.34 ± 1.35	9.54
variable02	12.8 ± 2.63	9.58 ± 2.51	12.2 ± 1.51	11.5
variable03	9.10 ± 1.14	6.32 ± 0.80	9.64 ± 3.44	8.35
variable04	19.4 ± 5.00	11.7 ± 2.88	8.08 ± 1.32	13.1
variable05	13.5 ± 4.48	7.37 ± 1.61	1.65 ± 0.16	7.51
variable06	8.07 ± 2.08	11.9 ± 1.93	1.65 ± 0.16	7.21
variable07	9.92 ± 2.37	12.2 ± 1.77	1.66 ± 0.16	7.95
variable08	6.05 ± 1.44	11.4 ± 1.59	1.65 ± 0.16	6.39
variable09	9.87 ± 1.90	10.7 ± 1.57	1.65 ± 0.16	7.42
variable10	13.2 ± 2.12	9.18 ± 0.98	8.67 ± 1.40	10.3
variable11	10.3 ± 0.27	9.04 ± 1.66	13.8 ± 2.40	11.1
variable12	12.3 ± 3.17	14.1 ± 3.39	10.2 ± 0.35	12.2
variable13	10.3 ± 2.17	4.46 ± 0.84	3.14 ± 0.26	5.99
variable14	10.3 ± 2.17	3.46 ± 0.72	3.15 ± 0.26	5.66
variable15	0.0 ± 0.0	2.71 ± 0.82	5.92 ± 0.84	2.88
variable16	3.85 ± 0.51	0.0 ± 0.0	8.31 ± 2.28	4.05
variable17	0.97 ± 0.56	7.47 ± 1.58	6.06 ± 0.81	4.83
variable18	2.84 ± 2.84	8.42 ± 1.26	12.1 ± 1.99	7.81
variable19	0.72 ± 0.70	1.97 ± 0.54	6.23 ± 0.84	2.97
variable20	8.71 ± 2.02	1.03 ± 0.08	0.0 ± 0.0	3.25
variable21	7.60 ± 2.67	16.0 ± 2.21	1.26 ± 0.11	8.30
variable22	10.9 ± 1.94	7.78 ± 1.08	6.90 ± 0.29	8.53
variable23	9.20 ± 1.91	0.0 ± 0.0	7.01 ± 0.12	5.40
variable24	6.41 ± 1.08	0.0 ± 0.0	8.91 ± 2.51	5.10
variable25	0.0 ± 0.0	7.79 ± 1.27	7.55 ± 1.54	5.11
variable26	10.6 ± 1.70	11.6 ± 1.73	16.3 ± 0.80	12.9
variable27	0.18 ± 0.18	0.0 ± 0.0	5.92 ± 0.84	2.03
variable28	14.6 ± 0.68	10.5 ± 1.18	10.3 ± 2.30	11.8
variable29	4.65 ± 0.77	4.38 ± 0.44	9.31 ± 2.71	6.11
variable30	3.40 ± 3.40	0.0 ± 0.0	5.92 ± 0.84	3.11
variable31	0.0 ± 0.0	0.0 ± 0.0	5.92 ± 0.84	1.97
variable32	2.38 ± 0.48	0.0 ± 0.0	8.24 ± 2.24	3.54
variable33	1.99 ± 0.31	8.91 ± 1.48	8.60 ± 1.88	6.50
variable34	3.57 ± 1.31	9.81 ± 1.24	8.69 ± 1.99	7.36
variable35	2.72 ± 0.23	9.22 ± 1.15	8.62 ± 1.96	6.85
variable36	0.0 ± 0.0	6.23 ± 1.04	7.55 ± 1.54	4.59
variable37	10.3 ± 1.81	9.33 ± 2.00	10.7 ± 3.15	10.1
variable38	10.4 ± 2.38	7.48 ± 1.88	17.1 ± 0.24	11.6
variable39	6.31 ± 0.94	16.3 ± 0.12	8.48 ± 1.74	10.3
variable40	13.5 ± 1.45	0.0 ± 0.0	10.7 ± 4.03	8.09
variable41	2.87 ± 1.91	8.07 ± 1.46	6.06 ± 0.81	5.67
variable42	7.32 ± 1.04	13.1 ± 0.29	8.80 ± 2.01	9.75
variable43	8.82 ± 1.80	10.3 ± 1.27	17.7 ± 2.25	12.29
variable44	1.28 ± 0.16	0.0 ± 0.0	8.90 ± 2.35	3.39
variable45	0.0 ± 0.0	7.51 ± 1.62	12.1 ± 1.99	6.56
variable46	1.91 ± 0.18	3.23 ± 0.45	8.42 ± 1.82	4.52
variable47	0.48 ± 0.47	2.28 ± 0.46	6.23 ± 0.84	3.00
variable48	8.39 ± 2.30	9.82 ± 1.49	17.4 ± 0.82	11.8
variable49	12.3 ± 2.56	0.49 ± 0.09	0.0 ± 0.0	4.28
variable50	12.3 ± 2.47	12.1 ± 2.93	1.06 ± 0.05	8.48

CRediT authorship contribution statement

Andreas C. Bueff: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Data curation. **Mateusz Cytryński:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Data curation, Visualization. **Raffaella Calabrese:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Matthew Jones:** Supervision, Resources, Conceptualization. **John Roberts:** Supervision, Resources, Conceptualization. **Jonathon Moore:** Supervision, Resources, Conceptualization. **Iain Brown:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

Funding was provided by EPSRC IAA PIII055.

References

Abdou, H. (2009). Genetic programming for credit scoring: The case of Egyptian public sector banks. *Expert Systems with Applications*, *36*, 11402–11417. http://dx.doi.org/10.1016/j.eswa.2009.01.076.

Addo, P. M., Guegan, D., & Hassani, B. (2018). Credit risk analysis using machine and deep learning models. *Risks*, *6*(2), 38.

Baesens, B., Rösch, D., & Scheule, H. (2016). *Credit risk analytics: measurement techniques, applications, and examples in SAS* (1st ed.). (p. 512). John Wiley & Sons Inc..

Bank of England (2019). *Machine learning in UK financial services*: *Bank of England working paper*, (pp. 1–36).

Bellotti, T., & Crook, J. (2013). Forecasting and stress testing credit card default using dynamic models. *International Journal of Forecasting*, *29*(4), 563–574.

Bracke, P., Datta, A., Jung, C., & Sen, S. (2019). Machine learning explainability in finance: An application to default risk analysis. *Risk Management & Analysis in Financial Institutions EJournal*, 1–44.

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, *39*(3), 3446–3453.

Bussmann, N., Giudici, P., Marinelli, D., & Papenbrock, J. (2020). Explainable machine learning in credit risk management. *Computational Economics*, *57*, 203–216.

Chang, Y.-C., Chang, K.-H., & Wu, G.-J. (2018). Application of extreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. *Applied Soft Computing*, *73*, 914–920.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: SYnthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*(1), 321–357.

Chen, T., He, T., Benesty, M., Khotilovich, V., & Tang, Y. (2015). *Xgboost: extreme gradient boosting*: *R package version 0.4-2*, (pp. 1–4).

Chen, H., Janizek, J. D., Lundberg, S., & Lee, S.-I. (2020). True to the model or true to the data? arXiv preprint arXiv:2006.16234.

Chen, C., Lin, K., Rudin, C., Shaposhnik, Y., Wang, S., & Wang, T. (2018). An interpretable model with globally consistent explanations for credit risk.. CoRR abs/1811.12615 URL: http://dblp.uni-trier.de/db/journals/corr/corr1811.htmlabs-1811-12615.

Coston, A., Mishler, A., Kennedy, E. H., & Chouldechova, A. (2020). Counterfactual risk assessments, evaluation, and fairness. *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 582–593. http://dx.doi.org/10.1145/3351095.3372851.

European Banking Authority (EBA) (2020). *Report on big data and advanced analytics*: *EBA report, January*, (pp. 11–47).

Financial Services Authority (2008). *Stress and scenario testing*: *Consultation Paper 08/24, UK*.

Financial Stability Board (2017). Artificial intelligence and machine learning in financial services. *Market Developments and Financial Stability Implications*, *1*, 1–40.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th international conference on data science and advanced analytics (DSAA)* (pp. 80–89). IEEE.

Giudici, P., & Raffinetti, E. (2021). Shapley-Lorenz explainable artificial intelligence. *Expert Systems with Applications*, *167*, Article 114104. http://dx.doi.org/10.1016/j.eswa.2020.114104, URL: https://www.sciencedirect.com/science/article/pii/S0957417420308575.

Grath, R. M., Costabello, L., Van, C. L., Sweeney, P., Kamiab, F., Shen, Z., et al. (2018). Interpretable credit application predictions with counterfactual explanations. ArXiv abs/1811.05245.

Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., & Turini, F. (2019). Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, *34*(6), 14–23.

Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., & Giannotti, F. (2018). Local rule-based explanations of black box decision systems. CoRR abs/1805.10820 arXiv:1805.10820 URL: http://arxiv.org/abs/1805.10820.

Hall, R. E. (2010). Why does the economy fall to pieces after a financial crisis? *Journal of Economic Perspectives*, *24*(4), 3–20.

Henley, W., & Hand, D. J. (1996). A K-nearest-neighbour classifier for assessing consumer credit risk. *Journal of the Royal Statistical Society: Series D (the Statistician)*, *45*(1), 77–95.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in R*. Springer, URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.

Johansson, F., Shalit, U., & Sontag, D. (2016). Learning representations for counterfactual inference. In *International conference on machine learning* (pp. 3020–3029).

Kraus, M., Feuerriegel, S., & Oztekin, A. (2020). Deep learning in business analytics and operations research: Models, applications and managerial implications. *European Journal of Operational Research*, *281*(3)), 628–641.

Kusner, M. J., Loftus, J. R., Russell, C., & Silva, R. (2017). Counterfactual fairness. arXiv:1703.06856.

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *1*(16), 124–136.

Letham, B., Rudin, C., Mccormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and Bayesian analysis: building a better stroke prediction model. *The Annals of Applied Statistics*, *9*(3), 1350–1371.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems* (pp. 4765–4774).

Luo, C. (2020). A comprehensive decision support approach for credit scoring. *Industrial Management & Data Systems*, *120*, 280–290.

Mian, A., & Sufi, A. (2009). The consequences of mortgage credit expansion: Evidence from the US mortgage default crisis. *Quarterly Journal of Economics*, *124*(4), 1449–1496.

Molnar, C. (2019). Interpretable machine learning.

Paleologo, G., Elisseeff, A., & Antonini, G. (2010). Subagging for credit scoring models. *European Journal of Operational Research*, *201*, 490–499.

Patil, A., Framewala, A., & Kazi, F. (2020). Explainability of SMOTE based oversampling for imbalanced dataset problems. In *2020 3rd international conference on information and computer technologies (ICICT)* (pp. 41–45). http://dx.doi.org/10.1109/ICICT50521.2020.00015.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Petropoulos, A., Siakoulis, V., Stavroulakis, E., & Klamargias, A. (2019). A robust machine learning approach for credit risk analysis of large loan level datasets using deep learning and extreme gradient boosting. In *IFC Bulletins Chapters 49*.

Prati, R., Batista, G. E. A. P. A., & Monard, M. C. (2004). Class imbalances versus class overlapping: An analysis of a learning system behavior. In *MICAI* (pp. 312–321).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. arXiv:1602.04938.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206.

Sharma, S., Henderson, J., & Ghosh, J. (2019). CERTIFAI: counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. CoRR abs/1905.07857 arXiv:1905.07857 URL: http://arxiv.org/abs/1905.07857.

Sokol, K., & Flach, P. A. (2019). Counterfactual explanations of machine learning predictions: Opportunities and challenges for AI safety.

the Board of Governors of the Federal Reserve System (2011). Supervisory guidance on model risk management. *Federal Reserve System*.

Thomas, L., Crook, J., & Edelman, J. (2019). *Credit scoring and its application* (2nd ed.). Siam.

Tolomei, G., Silvestri, F., Haines, A., & Lalmas, M. (2017). Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 465–474).

Vieira, J., Barboza, F., Sobreiro, V., & Kimura, H. (2019). Machine learning models for credit analysis improvements: Predicting low-income families' default. *Applied Soft Computing*, *83*, Article 105640. http://dx.doi.org/10.1016/j.asoc.2019.105640.

Voigt, P., & Bussche, A. v. d. (2017). *The EU general data protection regulation (GDPR): A practical guide* (1st ed.). Springer Publishing Company, Incorporated.

Wachter, S., Mittelstadt, B. D., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. CoRR abs/1711.00399 arXiv:1711.00399 URL: http://arxiv.org/abs/1711.00399.

Wang, Z., Crook, J., & Andreeva, G. (2020). Reducing estimation risk using a Bayesian posterior distribution approach: Application to stress testing mortgage loan default. *European Journal of Operational Research*, [ISSN: 0377-2217] *287*(2), 725–738.

Wang, Y., Sridhar, D., & Blei, D. M. (2019). Equal opportunity and affirmative action via counterfactual predictions. arXiv:1905.10870.

White, A., & d'Avila Garcez, A. (2019). Measurable counterfactual local explanations for any classifier. arXiv:1908.03020.

Zhao, H., Coston, A., Adel, T., & Gordon, G. J. (2019). Conditional learning of fair representations. ArXiv abs/1910.07162.