Douglas Merrill
CEO ZestFinance
Testimony to the House Committee on Financial Services AI Task Force
June 26, 2019

Chairman Foster, Ranking Member Hill, and members of the task force, thank you for the opportunity to appear before you to discuss the use of artificial intelligence in financial services.

My name is Douglas Merrill. I'm the CEO of ZestFinance, which I founded ten years ago with the mission to make fair and transparent credit available to everyone. Lenders use our software to increase loan approval rates, lower defaults, and make their lending fairer. Before ZestFinance, I was Chief Information Officer at Google. I have a Ph.D. in Artificial Intelligence from Princeton University.

The use of artificial intelligence in the financial industry is growing in areas like credit decisioning, marketing, and fraud detection. Today I will discuss a type of AI — machine learning (a.k.a ML) — that discovers relationships between many variables in a dataset to make better predictions. Because ML-powered credit scores substantially outperform traditional credit scores, companies will increasingly use machine learning to make more accurate decisions. For example, customers using our ML underwriting tools to predict creditworthiness have seen a 10% approval rate increase for credit card applications, a 15% approval rate increase for auto loans, and a 51% increase in approval rates for personal loans — each with no increase in defaults.

Overall, this is good news and it should be encouraged. Machine learning increases access to credit especially for low-income and minority borrowers. Regulators understand these benefits and, in our experience, want to facilitate, not hinder, the use of ML.

At the same time, ML can raise serious risks for institutions and consumers. ML models are opaque and inherently biased. Thus, lenders put themselves, consumers, and the safety and soundness of our financial system at risk if they do not appropriately validate and monitor ML models.

Getting this mix right—enjoying ML's benefits while employing responsible safeguards—is very difficult. Specifically, ML models have a "black box" problem; lenders know only that an ML algorithm made a decision, not why it made a decision.

Without understanding why a model made a decision, bad outcomes will occur. For example, a used-car lender we work with had two seemingly benign signals in their model. One signal was that higher mileage cars tend to yield higher risk loans. Another was that borrowers from a particular state were slightly less risky than those from other states. Neither of these signals raises redlining or other compliance concerns. However, our ML tools noted that, taken together, these signals predicted a borrower to be African-American and more likely to be denied. Without visibility into how seemingly fair signals interact in a model to hide bias, lenders will make decisions which tend to adversely affect minority borrowers.

There are purported to be a variety of methods for understanding how ML models make decisions. Most don't actually work. As explained in our White Paper and recent essay on a technique called SHAP, both of which I've submitted for the record, many explainability techniques are inconsistent, inaccurate, computationally expensive, or fail to spot discriminatory outcomes. At ZestFinance, we've developed explainability methods that render ML models truly transparent. As a result, we can assess disparities in outcomes and create less-discriminatory models. This means we can identify approval rate gaps in protected classes such as race, national origin and gender and then minimize or eliminate those gaps. In this way, ZestFinance's tools decrease disparate impacts across protected groups and ensure that the use of machine learning-based underwriting mitigates, rather than exacerbates, bias in lending.

Congress could regulate the entirety of ML in finance to avoid bad outcomes, but it need not do so. Regulators have the authority necessary to balance the risks and benefits of ML underwriting. In 2011, the Federal Reserve, OCC, and FDIC published guidance on effective model risk management.[1] ML was not commonly in use in 2011, so the guidance does not directly address best practices in ML model development, validation and monitoring. We recently produced a short FAQ, which we've also submitted for the record, that suggests updates to bring the guidance into the ML era. Congress should encourage regulators to set high standards for ML model development, validation and monitoring.

We stand upon the brink of a new age of credit. An age that is fairer and more inclusive, enabled by new technology — machine learning. However, "brink" can also imply the edge of a cliff; without rigorous standards for understanding why models work, ML will surely drive us over the edge. Every day that we wait to responsibly implement ML keeps tens of millions of Americans out of the credit market or poorly treated by it. Thank you for your time and attention.

[1] https://www.occ.treas.gov/news-issuances/bulletins/2011/bulletin-2011-12a.pdf

Douglas Merrill
CEO & Founder
ZestFinance
(Former Chief Information Officer of Google)

Douglas Merrill is the CEO and founder of ZestFinance, a Los Angeles-based financial services technology company that uses machine learning and data science to predict credit risk and make more accurate underwriting decisions. Zest helps lenders deploy fully explainable machine learning models and to make their lending fairer and more inclusive. Backed by some of Silicon Valley's most prominent venture capitalists, Zest's partners include Discover Financial Services, Ford Motor Co., Synchrony Financial, one of Turkey's largest banks, and Baidu.

In 2009, Douglas started ZestFinance with a hypothesis: Google-like algorithms could be applied to make consumer credit more transparent, available to more people, and significantly less expensive. ZestFinance's team of data scientists and mathematicians are united by a unique mission: to make fair and transparent credit available to everyone.

Prior to founding Zest, Douglas was the Chief Information Officer of Google for six years. Douglas led an organization of 15,000 staff, oversaw all aspects of internal engineering and technology, and drove multiple strategic efforts, including Google's IPO auction in 2004.

Douglas also served as Senior Vice President of Infrastructure and HR Strategy at Charles Schwab. In academia, Douglas was an Information Scientist at the RAND Corporation, where he conducted highly classified research for several branches of the U.S. armed services.

Douglas holds a Ph.D. in artificial intelligence from Princeton and is the author of *Getting Organized in the Google Era: How to Get Stuff Out of Your Head, Find It When You Need It, and Get It Done Right.* His academic publications include articles in The Journal of the Learning Sciences, Cognition and Instruction, Reliable Distributed Systems, and a paper in the book series Lecture Notes in Computer Science.

Beyond The Black-Box: A Better Framework For Explainable AI

By Evan Kriminger, Mark Eberstein, Sean Kamkar, Jose Valentin, Douglas Merrill
ZestFinance

November 2018

1 Table Of Contents

2 Introduction

Machine learning (ML) is a subset of artificial intelligence that focuses on the design of systems that can learn from and make decisions and predictions based on large information sets. It has become the standard for producing powerful data models that automate decision-making, often in high-stakes use cases. Its effectiveness has been proven in diverse fields such as natural language processing, robotics, recommendation engines, finance, and healthcare. The research community continues to substantiate the superior predictive power of these new algorithms over traditional methods such as logistic regression. Unlike status quo methods, ML models accommodate non-linearities, multivariate interactions, and generalize well to new datasets all within a single model -- improving accuracy and reducing complexity and risk.

Despite the clear benefits of machine learning, the use of logistic regression models continue to be the norm, especially in risk- and prediction-related business such as credit and underwriting. There are several reasons for this. One is that financial institutions (FIs) do not have the in-house expertise required to build, train, and deploy advanced ML models. More user-friendly ML modeling tools will help close this knowledge gap. The more imposing obstacle to adoption is regulatory and business risk. The Federal Reserve, the Office of the Comptroller of the Currency, and the Federal Deposit Insurance Corporation have all issued guidance dictating clear and documented model risk management: how and why a model that an FI has put into production arrives at the results. Explainable machine learning models should be the standard for FIs not only to meet regulatory requirements but also to illustrate their decision-making process to clients and business stakeholders.

In this paper, we define explainability in terms of the problems it solves, the principles on which it is based, and the way in which it conveys information about the model. We present an overview of methods in use in the market, along with techniques for evaluating the quality of explanations each technique delivers. Popular explainability methods have systematic shortcomings: they are often computationally expensive, restricted to certain classes of models, and they suffer from failure modes, which could lead to catastrophic outcomes such as race-based discrimination. Solving these issues is the focus of a considerable research effort, with the goal of efficiently explaining expressive models such that the explanations provide an accurate picture of the model's behavior in a human-interpretable form. Enabling the safe application of modern machine learning techniques is the key to revolutionizing high-stakes business problems like credit underwriting.

3 What is explainability?

To explain a model is to relate the model's decisions to the input data on which its decisions are based. This is a notably vague definition compared to the performance goal of machine learning, which is to make highly accurate predictions in a range of high-stakes applications, such as consumer finance and healthcare. While a useful optimization criterion for modeling might be classification error, the success of explaining a model is a more qualitative outcome. What does it mean for a model to be properly explained? To approach the difficult task of explaining machine learning models, the problem setting must be well-established. Which type of model is to be explained? What form does the information provided by the explanation take? What are the desired outcomes for interpreting this model? These questions dictate the principles of a satisfactory explainability approach.

3.1 Conveying explanations

The typical end-product of an explainability tool reveals the contributions or influence of each input feature towards the model prediction. In the context of machine learning, features are individual input columns. This feature-level explanation is considered local if it applies to a single input sample or global if it describes feature contributions over all samples. A different approach is to use example data points. Exemplar-level approaches explain a prediction in terms of the training input which led to that prediction. Thus, explanations can exist at the level of features, individual samples, or the model as a whole.

For example, in an autonomous vehicle, explaining the decision to turn involves generating a heat map over the input image revealing which features in the image led to that decision. An English to Spanish translation model can be explained by highlighting which English words led to each Spanish word in the translation. For credit underwriting, the decision to reject a loan applicant is explained by highlighting the fields in the loan application which led to the rejection decision. This may also include the features with the most important contributions, both positive and negative, to the applicant's score.

The interpretation of model behavior at a particular input is often less meaningful than explaining behavior at the input relative to a point of reference. This referential form of explainability is more intuitive for certain application domains. In credit underwriting, returning the reasons for a rejection are more informative in the context of an accepted applicant. The customer and regulators are less interested in the explanation of the model's exact probability output than the explanation of why the applicant was rejected compared to an accepted reference applicant.

3.2 Interpreting a model

Explanations reveal how a model uses its inputs to make predictions. With feature-level explanations, this is a breakdown of each feature's contribution to the output. However, it is only in the case of linear models that the feature contributions are exactly the linear model coefficients for each feature. For nonlinear models, the explanation does not convey information as transparently. It is therefore important to assure that the explanation of a complex model can be trusted to depict model behavior accurately.

There are a few defining properties of a reliable explanation, which aid in the development and evaluation of explainability techniques:

Consistency. A consistent explainer should not rely on meticulously tuned parameters and should provide reasonable results for a wide range of parameters. It should not be prone to large random fluctuations between repeated runs of the program. Logically equivalent models should yield the same explanations. Similar inputs should receive similar explanations.

Accuracy. An explainer should be fully representative of the true dynamics and behavior of the model. The problem of interpreting complex machine learning often leads to some simplifying assumptions. For example, the proxy model explanation technique assumes that a simple, interpretable model can serve as a proxy to explain a more complicated model, as long as their input-output behavior is similar enough. The proxy model assumption is reasonable only if a high-level interpretation of the model is needed, such as the feature importance across all inputs. For explaining individuals, the proxy is going to produce very different results from the target model. Other assumptions such as monotonicity and independence of the input variables cannot be guaranteed and result in explanations which do not reflect reality.

One tangible target for the accuracy of an explanation is the sensitivity of a model, which refers to how a model's output is affected by a small perturbation to its input. Variables which are declared to be important by an explanation should have a significant impact on the model's output when perturbed.

Interpretability. Explanation values are not normalized quantities such as with probabilities, and amplitudes of explanations may vary greatly between methods. If the desired end product of an explanation is the relative feature importance, then this is not an issue, but providing values in interpretable units is preferable. Some explainers provide an attribution to the features which sums to the model's output. Thus, the explanation can be directly interpreted as each feature's contribution to the output. Explanations are harder to interpret in isolation, so some techniques are referential. This means that they explain the prediction for a particular input in reference to the model's treatment to a baseline input. For many problems a referential explanation is natural. For example, in credit underwriting, the decision to deny an applicant credit may be explained relative to a reference-approved applicant.

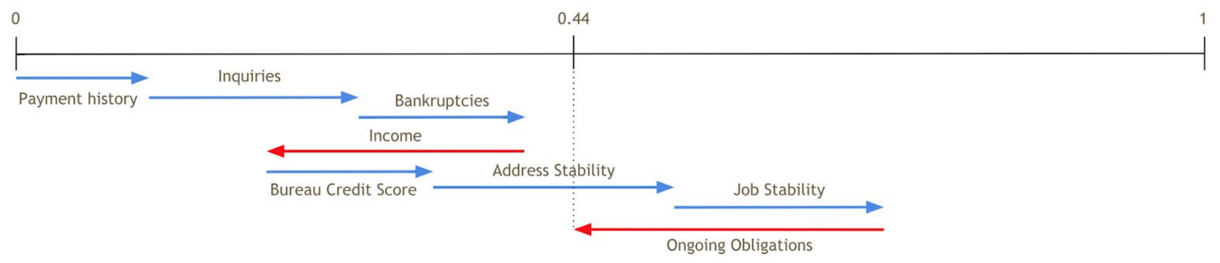


Figure 1: A credit model's prediction of 0.44 decomposed into contributions from the input features.

Practicality. Explainability techniques can be quite computationally expensive. Explainers must scale well with the complexity of the model and the size of the dataset if they are to offer real-time explanations. An important consideration is the types of models which a technique is capable of explaining. Black-box methods can explain any model, while others focus on trees or neural networks. Ensembles of models offer robustness and improved performance, yet interpreting heterogenous submodels poses a difficult task for explainability.

4 Importance of explainability

The predictive ability of a machine learning model is only one component of a complete explanation. Predictions optimize a particular objective function, such as mean squared error, but this is an incomplete picture of the overall success of the model. A complete explanation needs to address numerous regulatory concerns that the standard model validation process takes for granted, as it was developed for more easily interpretable logistic regression-based models. In order to put machine learning models into production for everyday use, their explainability methods have to meet the following regulatory model validation requirements [Doshi-Velez and Kim, 2017]:

- **Fairness:** It is necessary to identify unlawful or inappropriate model biases. For example, some features may be a proxy for race or gender. A model should not be overly reliant on these features, and not treat individuals from protected classes differently than those from an unprotected class with comparable inputs.
- **Safety:** Identify failure modes which may be extrapolated from the features the model uses in making its decision. Such failure modes may not be obvious in summary statistics derived from validation sets. The model may be getting good results from obviously flawed logic or poor results due to unobvious changes in input data. Consider the stopping decisions an autonomous vehicle makes for pedestrians. It is important to interpret the model's decisions to ensure that the pedestrian is the driving force of the stop, and not just the presence of signs or crosswalks. Otherwise, the model may not respond to pedestrians in unprotected crossing situations.

- **Objective misalignment:** Although the model's objective function may be a proxy for a business objective, the model must be analyzed to ensure it is adequately addressing its intended purpose. For example, optimizing a diet to lower cholesterol may not produce a healthy diet if the dieter chooses nonfat, but high sugar foods.
- **Security:** A model that is not well-understood could be exploitable by attackers. The model can be manipulated if easily modifiable features can change the outcome. Confirming that the model does not rely on such "gameable" features helps ensure its security.
- **Model health monitoring:** In production, the model's performance could degrade considerably if the characteristics of the population diverge from the training set. It may be difficult to acquire ground truth target values for more recent examples to detect such a change. Additionally, in more critical applications, monitoring may need to be real-time. Using explainability to monitor model health ensures that the functionality of the model is the same as it was during training. This is a more direct test for the model's health than detecting shifts in the input data alone, with no regard to the model.

4.1 Stakeholders

Explainability impacts many parties, and understanding how these impacts play out is an important step in designing machine learning products.

- **Regulators** have laid out model risk management criteria. While regulatory guidance was based on old techniques, that guidance still applies to ML models in production. Unfortunately, you cannot meet regulatory requirements for ML models without explainability. In the European Union, for instance, the General Data Protection Regulation (GDPR), as of 2018, grants a "right to explanation" for users subject to automated or AI-based decision-making. In the U.S., state regulators are also moving in the direction of demanding greater explainability, and federal regulators are actively reviewing the use of alternative data and advanced mathematical techniques in automated decision-making.
- **Consumers** also benefit from receiving explanations of decisions that can have a profound impact on their lives. Automated decisions can be troubling, and clear explanations of how an automated process arrived at a decision can provide confidence that they were treated fairly, and offer a path of correct behavior towards a different outcome. At ZestFinance, our goal is to expand the availability of fair and transparent credit, which requires providing consumers with interpretable decisions.
- **Management** must assess the business impacts and the risks associated with deploying machine learning models. Explaining models puts algorithmic decisions in a context that is consistent with the business logic of human decision makers. Explainability tools may themselves be an important product offering alongside machine learning models. Business leadership has more options for what to include in a complete machine learning product.

- **Data scientists** must consider explainability in their modeling decisions. This is an emerging field, with an ever-evolving set of techniques and empirical results. Research and development efforts may need to focus more on explainability than the actual modeling itself. Developments in explainability enable data scientists to build more powerful models and provide a valuable tool for debugging and validation.

4.2 Explainability in credit

In consumer credit underwriting, explainability is just as important as the model itself. The need for explainability is explicit for credit underwriting systems, as specified by the Equal Credit Opportunity Act (ECOA) Regulation B and Fair Credit Reporting Act (FCRA). These regulations require that lenders supply *adverse action* notices, which inform the applicant of the reasons for the denial of credit. The law establishes the basis on which applicants cannot be denied credit. For example, discrimination is prohibited on the basis of race, sex, age, national origin, or marital status, i.e., the *disparate impact* of particular classes.

The legal requirement of interpretability has led the industry to be dominated by simple, inexpressive models such as logistic regression. Expanding credit to more good borrowers (without added risk) can only happen with the wider adoption of machine learning. This can only happen by applying novel explainability techniques. In the following sections, we describe the current explainability methods and their performance ability, followed by a comparison to Zest's explainability method and performance.

5 Explainability techniques

Methods for explainability can be categorized based on the type of model they explain and the criteria which these explanations seek to satisfy. The explanation can also convey information in different forms, as described earlier.

Black-box explainability techniques derive explanations solely from input-output behavior, without considering the model internals. Black-box techniques have the benefit of versatility and apply to any class of model. The tradeoff for this versatility is the difficulty in accurately characterizing model behavior without utilizing any specific information from the model. Black-box techniques are also often computationally expensive, which roughly stems from the need to enumerate many test cases in an attempt to fully explain the decision space. White-box techniques exploit the structure of the model in interpreting its decisions. While white-box techniques have more information available to produce explanations, this comes at the expense of losing the flexibility of black-box methods.

5.1 Intrinsic explainability

Two model types are considered to be inherently explainable: linear models and decision trees. For this reason, they have been workhorses in regulated industries. A linear model makes predictions of the form

$$f(x) = a_1x_1 + a_2x_2 + ... + a_nx_n + \varepsilon$$

Where x_i is the ith feature on the input, a_i is the corresponding coefficient (weight), and ε is the bias term. Explaining a linear model is simple because the contributions of each feature are by definition additive. Each feature contributes its value weighted by the coefficient associated with it, i.e. a_ix_i. The coefficients of the linear model serve as measures of feature importance, and these do not change across all inputs. Thus, local and global explainability are the same for linear models. For classification problems, with a discrete number of classes, modelers use a logistic regression model that takes the form:

$$f(x) = \sigma(a_1x_1 + a_2x_2 + ... + a_nx_n + \varepsilon)$$

In logistic regression, the linear model is transformed by the nonlinear logistic function, which scales the output to behave like a class probability. Since this function is monotonic, logistic regression models can be explained by their underlying linear model.

Decision trees arrive at classification decisions by following decision paths determined by querying individual features. Each node in a decision tree is associated with a test that a certain feature is above a given threshold, with the result of this test determining the next node. Leaf nodes in the decision tree represent decisions.

Predictions made by a decision tree are the result of very explicitly stated conditions on a handful of features. Explaining a decision tree is as simple as returning the decision path. This is a local explanation, but a global summary of model behavior can be derived from measures of the frequency with which each feature is used in the decisions.

Other simple classifiers may be considered explainable. The k-nearest neighbor's classifier assigns inputs to the class of the nearest input in the training set, and thus inherently offers exemplar-level explanations. The naive Bayes classifier uses the independence assumption to represent the class posterior probability as the product of likelihoods for each feature. The feature likelihoods represent the relative importance of each feature towards the model's prediction.

5.2 The challenge of explainability

While simple models offer out-of-the-box explainability, unlocking more powerful models requires advanced explainability methods. The most expressive models with the greatest potential performance gains provide no innate interpretability. The classifiers discussed above do not scale to high dimensional data such as images, making them unusable for many real-world problems. Ensembled models offer robustness, but explaining them carries the added difficulty of ensuring that the submodel explanations are compatible. The relative ordering of feature-level explanations may be the only meaningful information that can be extracted, which prohibits the comparison of explanations between models.

5.3 Univariate perturbations

Explainability of a black box model requires understanding how the model responds to its inputs. Consider the task of local explainability. A natural approach is to perturb a given input and observe the effect on the output. If the model is highly sensitive to the perturbation, then the features involved were important to the prediction. Fully characterizing the model may require testing any possible perturbation, which is computationally intractable and cannot return a concise explanation of the model function. The most common procedure is to observe the effect that a single feature has on the model output, and use this a measure of that feature's importance.

One type of perturbation is to remove a feature entirely, which is the basis for Leave-one-covariate-out (LOCO) [Lei et al. 2018] feature importance. The impact of removing feature x_i is $f(x) - f(x_{-i})$, where x is the original input and x_{-i} is the input without feature i. Another approach is to add noise to a feature rather than removing it entirely. Permutation feature importance (PMI) [Breiman 2001] is a global explainability method in which, for each feature, the values of that feature are shuffled across all input samples. The intuition is that if a feature is not being incorporated into the model's decision, then replacing it with arbitrary values will not affect performance.

Univariate perturbation approaches are simple to implement and model-independent but suffer from a few important drawbacks. The process of perturbing each feature individually is computationally demanding. The model must be evaluated for each feature, and for each perturbation that is required to get a sufficient estimate of the model impact.

Not only are univariate perturbation approaches computationally intensive, they often also yield wildly inaccurate results. Measuring only univariate variable effects does not explain model behavior that depends on variable interactions. For example, in credit underwriting, the impact that an applicant's income has on their score depends on the loan amount. For a small loan, greatly increasing an applicant's income is probably not going to have a large impact on their score. A permutation of income, even a large increase, might appear irrelevant to credit. This is likely to be wrong.

One must take care when perturbing inputs that the perturbed input still makes sense. Consider a variable such as a car's down payment. Permutation feature importance could, by substituting a down payment for a Porsche on a cheaper Ford Fusion, end up evaluating the model on applications with down payments that are greater than the loan amount itself. Not only do these "made up" data points fall well outside of the space of data that the model was trained on, they violate the fundamental logic of the problem domain.

5.4 Visualization

Univariate or bivariate feature importance can be demonstrated graphically. Partial Dependence Plots (PDPs) [Friedman 2001] compute the model output for fixed values of features under study, averaged over all input samples. This produces a curve (1D) or heatmap (2D) of how a feature's values generally influence the output. PDPs lose information by averaging over all inputs, which again ties into the problem of variable interaction. Independent Conditional Expectation (ICE) [Goldstein et al. 2015] plots attempt to alleviate this issue by plotting multiple response curves, which represents a split of the input samples by conditioning on certain variables. While useful, these tools provide graphical insight into the effect of a feature rather than true model explanations. In addition, such plots require human interpretation for each feature, which is hard for large numbers of features.

5.5 Proxy models

A complicated model can sometimes be sufficiently approximated by a simpler explainable model. If this proxy model can be shown to behave closely to the original model, then its explanation may be similar as well. Student-teacher [Buciluǎ et al. 2006] learning is one way to generate simple proxy models. The teacher is trained on the given task and used to generate predictions on the whole dataset. The student model is a less complex model that is trained on the same input data but with the teacher's predictions as its target. The intuition here is that the predictions of the teacher provide a more easily learnable target for the simple model with less expressive power. While this paradigm makes sense for models of the same type, such as neural networks, differing only in architecture, it is unlikely that an inherently explainable model such as logistic regression or a decision tree would be able to accurately represent a powerful teacher. This can be clearly seen by trying to fit a linear proxy model to a parabola $y = x^2$. Even if two models make the same predictions on a set of examples, this is not a strong argument that they use the data in the same way.

Local Interpretable Model-Agnostic Explanations (LIME) [Ribeiro et al. 2016] notes that simple, explainable models can locally approximate complex models. Explaining a model, therefore, involves building an explainable model at the point of interest. In LIME, data is sampled in the vicinity around the test point and used to train a proxy model (logistic regression or decision tree) with the samples weighted inversely proportional to their distance from the test point. LIME provides black-box explainability and has empirically shown promising results, but suffers from some practical drawbacks. The process of sampling and training a proxy model for each input is computationally expensive. Sampling itself may be difficult in high-dimensional datasets where it is difficult to choose appropriate metrics and parameters to define a local neighborhood.

5.6 Gradient methods

Another approach to explainability is to capture the sensitivity of a model's output to its inputs. For differentiable models, such as neural networks, this amounts to taking the gradient of the output with respect to the input [Simonyan 2013]. The simple case of a linear model reveals the effectiveness of this approach. Given the model,

$$f(x) = a_1x_1 + a_2x_2 + \ldots + a_nx_n + \varepsilon ,$$

take the gradient with respect to a feature x_i

$$\partial f/\partial x_i = a_i ,$$

to reveal that the sensitivity of the model to x_i is simply the coefficient a_i which weights that feature.

While the gradient applies to arbitrary differentiable models and corresponds to an intuitive definition of explainability, it suffers from a few drawbacks. Backpropagation is used to pass gradients from the output of the network to the inputs. Due to the use of nonlinear activation functions which rectify and clip the signals, neural networks have many "flat" regions in which the value of the gradient is zero. A zero gradient suggests that the factor does not matter, which could be correct, but complicated nonlinear models do actually extract information from flat regions. For this reason, plain gradients are not a perfect solution for propagating contribution through a network. Recent research [Kindermans et al. 2017] addresses this issue with techniques to pass gradients through flat regions. Many of these techniques are specific to certain architectures, activation functions, and application domains.

6 Evaluating explanations

The quality of an explanation is difficult to evaluate relative to the well-defined goal of model accuracy in supervised machine learning. Evaluating explanations is easier for visual machine learning tasks, such as object recognition, as the quality of the explanation can be compared to a human's understanding of the important aspects of a scene. Other problem domains lack an obvious intuitive explanation and a human oracle may not be available for every input of interest. Automated evaluation of explanation quality is important to advance the field of explainability. For business applications, metrics can ensure the reliability of the explanation.

Aside from the lack of a ground truth, explaining a model is an ambiguous problem. If an explanation seems suspicious, there are two possibilities:

1. The model is behaving erratically, and the explainer is accurately describing this behavior.
2. The explainer is not properly describing model behavior.

Deciding which of these two possibilities is truth is hard even for tasks where useful features are inherently known. This is problematic. A model explainability technique should be able to accurately describe model behavior. What is needed are means of assessing the accuracy and utility of an explainability approach for a given task.

The process of evaluating an explainer is therefore not as simple as ensuring that its explanations put more weight on generally important features. In the literature, explainability techniques are often proposed along with fundamental properties that they satisfy. Confidence in the explainer is derived by showing, either theoretically or experimentally, that it satisfies the properties of a good explainer. Another approach to evaluating explainability techniques is based on the true goal of model interpretability, which is the information it conveys to the human user. A variety of experiments can be devised to test how explanations assist a user in the desired task.

6.1 Comparison metrics

The ability to compare two explanations is useful for evaluation purposes. Metrics for explainability are vital in research and development as well, providing a means of measuring improvement or regression from the state of the art. This is similar to distance metrics which compare two input samples. For two explanations to be similar, they must rank the features similarly in importance. Ranks may be compared with Spearman's rank correlation coefficient [Pirie 2004], a nonparametric estimator of correlation in the orderings of two variables. Two explanations which produce the same ordering of feature importance receive a Spearman's coefficient of +1. The rank of the least important features is not very meaningful because many features contribute trivially and their relative ordering is noise.

Spearman's coefficient weights the entire ordering equally, which may result in an explainability metric that does not align with human intuition. In top-*k* intersection, the top-*k* features for each explanation are computed and the number of features in common is used as a measure of similarity. This is more in line with the human perception of which features are driving a decision. The parameter k can be set by determining the number of features after which the contribution values fall to a trivial level.

6.2 Properties of a good explanation

The following properties, while not an exhaustive list, provide an extensive framework for comparing and evaluating explainability methods. These were developed through research and development at ZestFinance and curated from the literature on explainability.

- **Sensitivity to hyperparameters**
  In supervised learning, the selection of hyperparameters can be conducted methodically through a carefully designed cross-validation process. Explainability methods also have hyperparameters, but lack of ground truth values preclude the same cross-validation process. Many practitioners resort to simply observing the top features from the training set in aggregate. With this difficulty in validation, it is important that explainers not be overly sensitive to their hyperparameter values. The property can be tested by perturbing hyperparameters and observing the change in the explanations.

- **Variance**
  Explainers that are not deterministic will produce different explanations each time they are run, even on the same input sample. Small variations in the amplitudes of the feature attributions are permissible, but if the rank ordering of features changes between runs this creates serious issues in the reliability of the explainer.

- **Smoothness**
  If two input samples are extremely similar and are scored identically by the model, then one would expect the explanations for each of these inputs to be similar as well. Without such smoothness, explanations appear random and unreliable. To evaluate the smoothness of an explainer, take the nearest neighbor samples with the same predictions, and measure how close their explanations are. It is expected that if two samples are close in input space, then their explanations are highly correlated.

- **Precision**
  Complex machine learning models make their decisions based on complicated relationships between their variables. It is important that an explainer has the power to follow the subtle behavior of a model. Some explainability methods seek to simplify the problem with proxy models, but if the proxy model is too simple, the details of local behavior will be missing in the explanations. One way to test for the precision of an explainer is to look at the diversity of features reported in the top-*k*. An explainer with no locality, such as a global linear proxy model, will always return the same features.

- **Accuracy**
  If a given variable or feature is important to a model's output, the explanation should report it as important. Establishing the accuracy of feature impact is a principle that guides many of the commonly used explainability methods such as permutation

importance and partial dependence plots. The ability to change the prediction of a model by changing the most important features is, therefore, a very intuitive measure of the explainer's accuracy. It would be disconcerting to a denied credit applicant, for example, if the most important factors in the decision could not, in fact, change their score.

- **Treatment of correlated variables**
  It is difficult to assign credit to correlated features even in linear models. Perturbation-based explainability techniques may miss the impact of these features entirely by perturbing them in isolation. The desired attribution amongst correlated variables is not clear and depends heavily on what the model is doing, however, the behavior of an explainer should be understood and consistent. A related problem is with one-hot encodings of categorical variables. If one of the categorical values is a particularly important predictor, then it is preferable to attribute the indicator of this value. Because only one of the one-hot indicators may be 1 for a given input, an explainability method may decide to weight the less important indicator, since its value being 1 implies that the truly important indicator is 0.

- **Robustness to outliers**
  Explainability methods may grow unstable near the edges of the data space. This is a concern for methods like LIME which are based on samples from the distribution of the data. Ensuring consistency for outliers is important since explanations ensure that the model is using reasonable features for inputs in regions where the model did not have much training data.

- **Computational cost**
  For production systems, explanations may need to be real-time or near real-time. Thus, an explainer should be able to efficiently explain both single inputs and batches of inputs. Explainability techniques vary greatly in their computational demands. Brute force computation of some techniques is completely intractable, while a gradient, for example, can be evaluated at the same time as a model prediction.

6.3 User testing

An explanation is only useful if people can use it to understand a model and act based on the model's insights. This utility should be verifiable by experiment. For example, modelers should be able to use the explanations to predict which of two given models will generalize better, or use more robust logic in generating predictions from its inputs. A human user should be able to identify irregularities in a model and predict inputs which would be misclassified. If the explanations are accurately describing the model, the user will be able to make these predictions.

7 Two experiments show limits to current explainability techniques

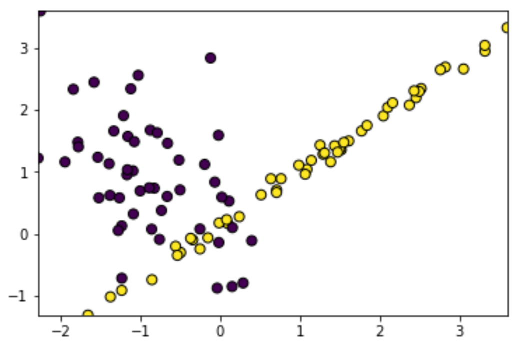
Explainability in machine learning is still very much an emerging field, lacking thorough testing and comparison procedures. In this paper, we perform two experiments to better understand the limits of the ability to explain supervised learning models using current techniques. The first experiment compares three popular explainers on a tree model trained on a toy classification dataset to visualize the mechanics of the algorithms, understand the effect of model hyperparameters, and diagnose potential failure modes. In the second experiment, we evaluate the explanations of a neural network model using a more robust Lending Club loan application dataset.

Experimental evaluation of explainability methods has overwhelmingly focused on vision problems, as saliency maps can be compared by visual inspection. There has been less experimentation on other domain-specific machine learning applications such as credit underwriting and natural language processing. The tabular data of problems like credit underwriting lacks the local structure between neighboring columns that pixels have in image data. Categorical variables behave very differently than continuous variables, which poses additional difficulties.

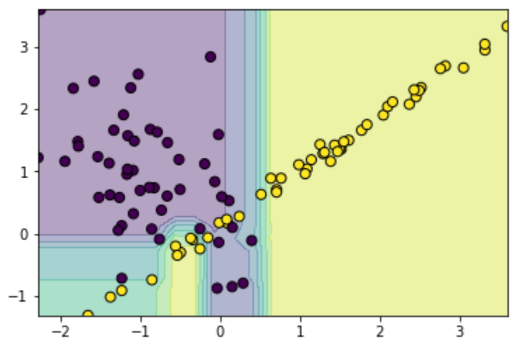
The results of the two experiments in this paper offer insight into the implementation of explainability methods and their performance for credit problems. We develop an extensive set of tests to evaluate the stability and validity of explainers. It is shown that popular explainers require extensive parameter tuning, and may produce unreliable results. Many explainers do not scale well computationally with the number of rows or columns of the dataset and are thus unsuitable for production. The ZAML explainer is evaluated and is shown to satisfy all the desired properties of a reliable explainability technique.

7.1 Experiment One: Implementation issues of LIME

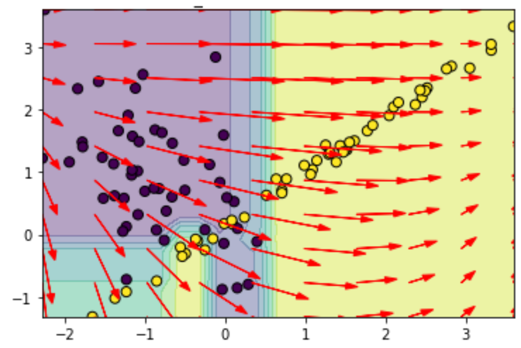
To demonstrate the performance issues of LIME, we begin with a simple 2D classification problem. This toy dataset is shown in the figure below. The purple dots represent class 0 and the yellow dots class 1. The goal of the model is to label each input as belonging to one of the two classes.



An XGBoost classifier model is trained on this data. In the plot below, a heat map of the model output reveals the decision surface of the model.

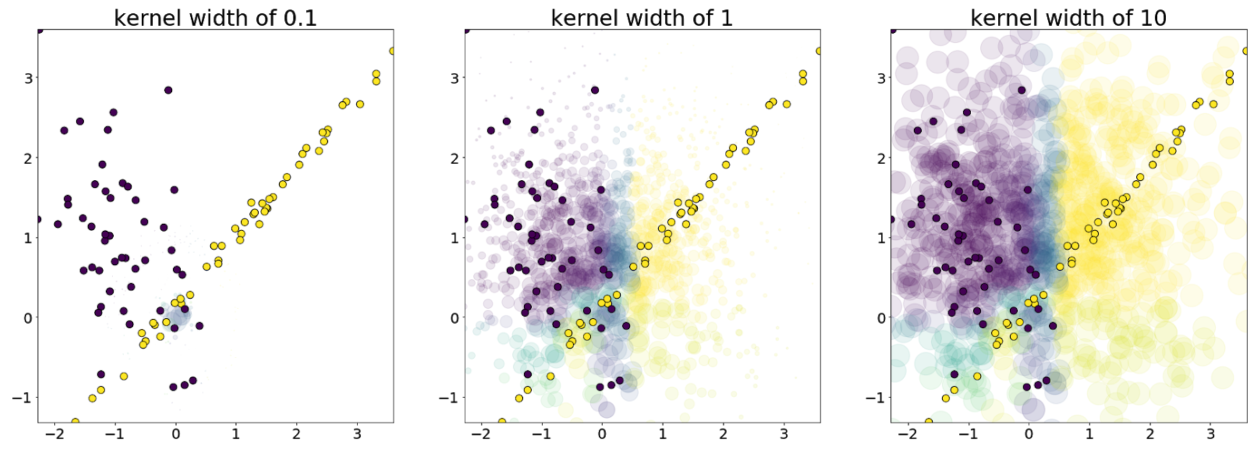


LIME can be used to understand the importance each feature plays in the model's decision in different regions of the data space. The LIME explanations are computed at evenly spaced grid points, and the resulting explanation is displayed as a red arrow. The horizontal component of the red arrow corresponds to the explanation of the *x* feature, and the vertical component is the explanation of the *y* feature.



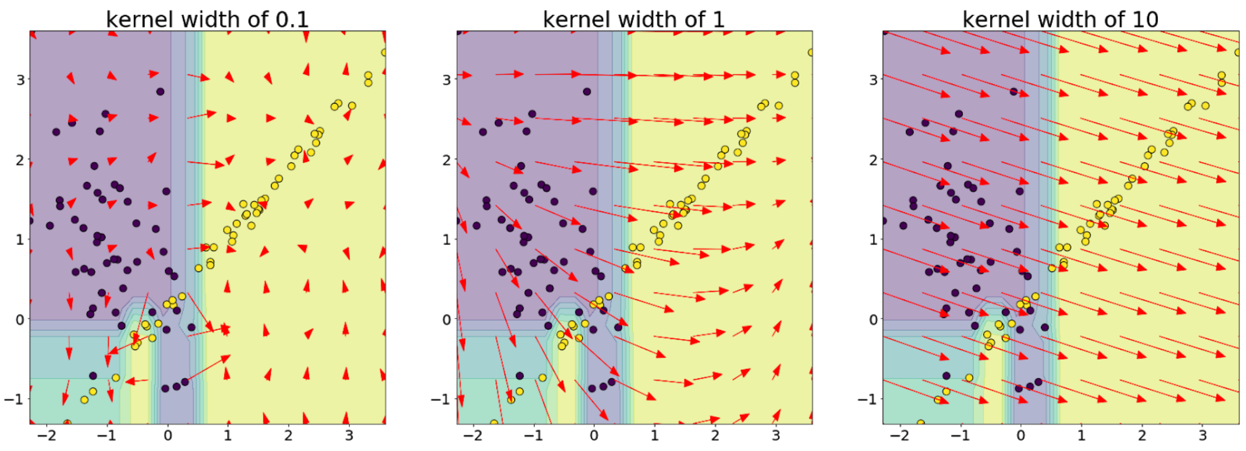
The nature of LIME's explanations depends heavily on its two main parameters. The first parameter is the kernel size, which determines the size of the local neighborhood around a point from which the linear proxy model is built. As the kernel size grows large, LIME approximates the model globally with a single linear model. The second parameter is the number of samples that are used in estimating the proxy model. Drawing more samples results in a more stable explanation, but increases the computation time.

To examine the effect of the kernel size, the figures below capture the neighborhoods LIME would use to explain a prediction at (0,0) given different sizes. The sampled neighborhood points are plotted beneath the data point, colored according to the prediction of the XGBoost model. The size of each neighborhood point is proportional to its weight in the estimation of the linear proxy model. The color represents the score given to each point by the model. Open source implementations of LIME choose a relatively high kernel size, as this ensures that samples from all classes will be present in the neighborhood. However, as can be seen from the figure on the right, if the kernel size is too large, the locality of the proxy model is lost.

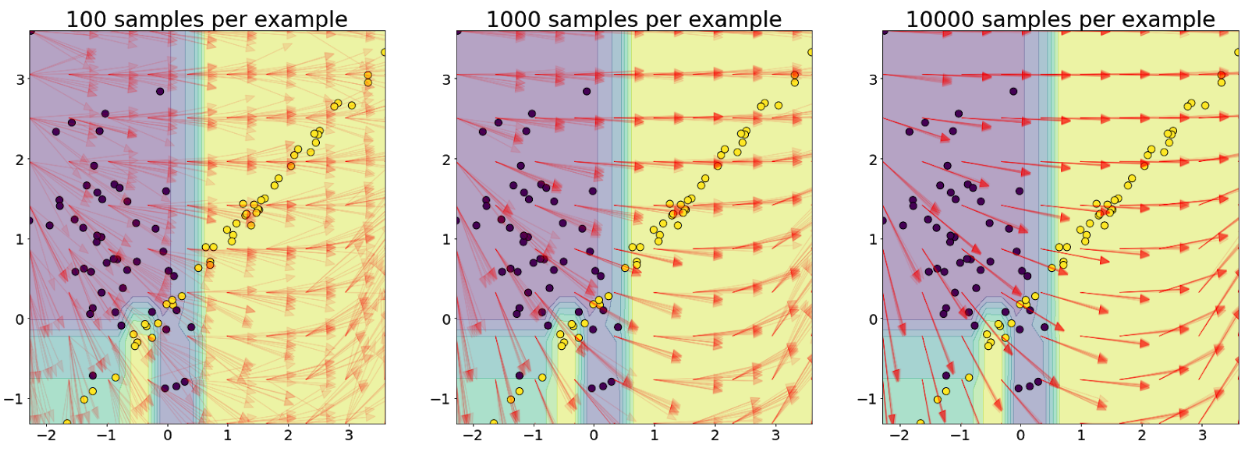


The effects of the kernel size on the resulting LIME explanations (the coefficients of the linear proxy models) are seen in the figure below. The arrow plot may be interpreted as the slope of the linear proxy model at that particular point, thus pointing in the direction of increasing model scores. For the smallest kernel size, the explanations point towards the closest prediction boundary. This provides a more precise description of model behavior, but in sparse regions of the data space, behavior becomes erratic. For samples where x > 1, the x-coordinate is entirely responsible for the prediction of class 1. The LIME neighborhood is so small that it does not contain a reasonably weighted sample from class 0 this far from the boundary. Thus, when LIME is truly local, the predictions become unreliable for many regions of the data space.

When the kernel size is 1, LIME provides a reasonable explanation of samples near the simplest part of the classification boundary (y > 1). The larger neighborhood results in a smooth explanation space, but the finer details in the lower left corner are lost. For an extremely large kernel size of 10, LIME has lost all locality, and each proxy model is the same linear model fit on the unweighted dataset.



The effect of the number of samples that are drawn for the purpose of estimating a proxy model is illustrated in the figure below. With a kernel size of 1, LIME is repeatedly run 10 times, and the resulting explanation arrows are plotted to show the volatility under different numbers of neighborhood samples. With only 100 samples to estimate each proxy model, the explanations vary wildly, to the point where either feature could be considered more important, depending on the draw. As the number of samples increases, the LIME explanations approach determinism. There is a tradeoff between the volatility and computation time of LIME. The proxy model must be learned for each sample that is to be explained. In this simple 2-dimensional problem, 1,000 samples suffice for a reasonable explanation, but as the dimensionality of the data grows, this might become a concern.



One of the more difficult details in implementing LIME is drawing samples from the distribution of the data. Generative models [Goodfellow et al. 2016] address the problem of drawing accurate samples from the distribution of the data but are amongst the most difficult models to train in machine learning. The authors of LIME simply estimate the univariate means and standard deviations for each feature and draw each feature independently from these independent normal approximations. For some types of data, this may be a reasonable approximation, but this will pose problems, especially with categorical data types, as will be shown in the next section.

From this experiment, it is clear that LIME is very sensitive to its parameters. Even in 2 dimensions, a very large number of generated samples is needed to get stable performance. This will worsen in higher dimensions. The kernel width parameter is easier to estimate for a visualizable dataset, but will also be a much more difficult problem for real datasets. For real data, the Gaussian assumption for the sampling distribution may be unreasonable.

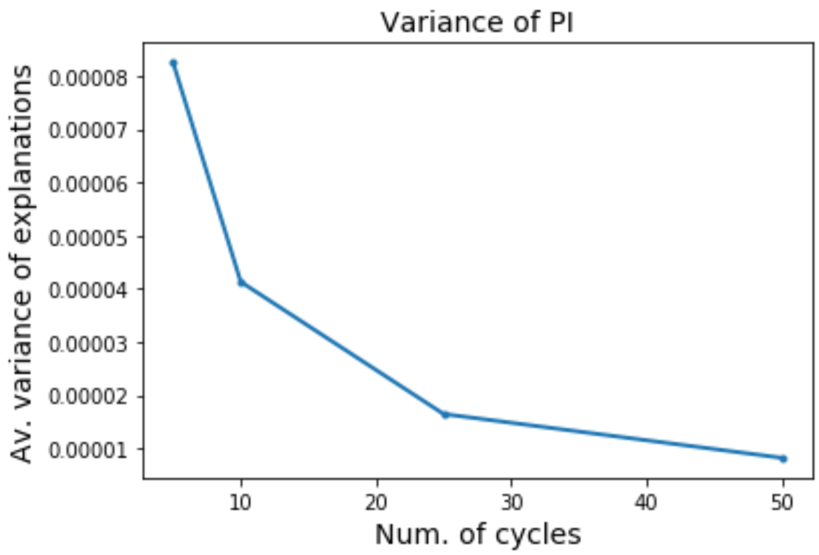
7.2 Experiment Two: Explaining performance on real credit data

In our second experiment, we went beyond the limitations presented for toy datasets by attempting to explain a model built on a large, publicly available 2007-2011 data set from the online loan marketplace The Lending Club. The data included 42,538 loan applications and their payment status. Records with a loan status of "Charged Off" or "Fully Paid" were selected, with these statuses serving as the classification target. Feature engineering was deliberately kept simple. Commonly used variables were selected by hand, categorical variables were one-hot encoded, and missing values were imputed with the mean, while adding a binary column to indicate if the value was missing. This resulted in 39,088 input records with 35 dimensions. The first 35,000 records were used for training and the remaining 4,088 were reserved for evaluating explainers. The data was standardized by subtracting the mean of each column and dividing by the standard deviation.

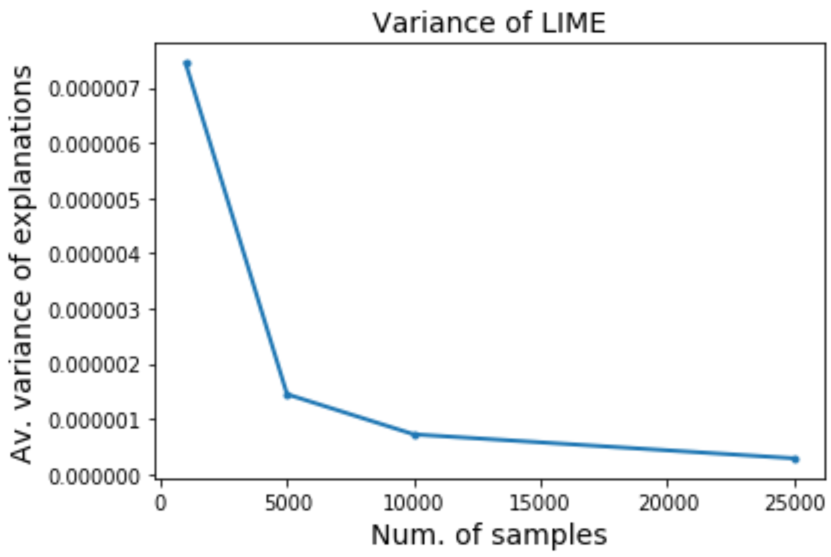
We built a neural network model with 4 dense layers with ReLU activations and dropout regularization. The model had a test set performance of 0.703 AUC. Three explainers were run on this model: permutation importance (PI), LIME, and the ZAML explainer (ZAML), which is based on vector calculus. We evaluate their performance here based on the properties described in the "Evaluating explanations" section of this paper

Variance

Permutation importance (PI) and LIME are stochastic algorithms, while the ZAML explainer is deterministic. While some randomness in an explanation is acceptable, an explainer that provides different feature rankings across subsequent runs is undesirable. The randomness in PI originates from the shuffling of the columns to determine model sensitivity to that feature. The number of times each column is shuffled is a parameter called *cycles*, and it controls the tradeoff between determinism of the explanation and computational cost. In the plot below, the explanations from PI are generated for the Lending Club test set, and the variance across 10 runs is computed for each feature level attribution. We plot the average variance over all inputs and columns as a function of the *cycles* parameter. By increasing the cycles, the variance of PI can be greatly reduced at the expense of increased computational cost. Each additional cycle requires *D* additional model evaluations, where *D* is the dimensionality of the feature space.

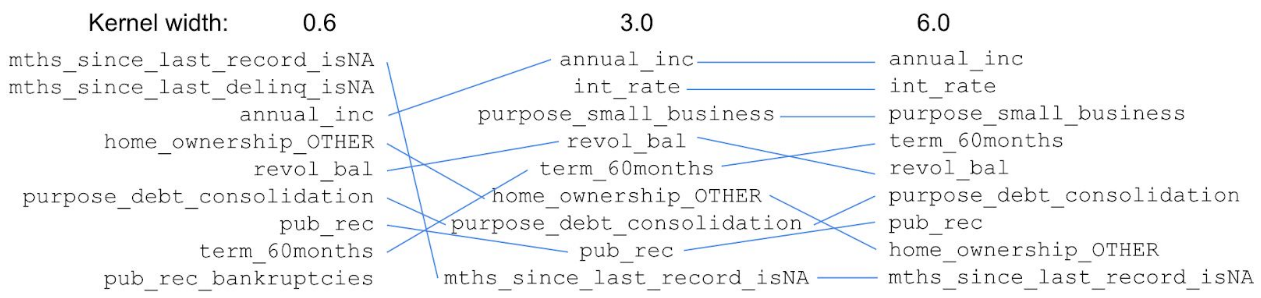


Randomness in LIME arises from the neighborhood sampling procedure. As with PI, there is a tradeoff between determinism and computational cost, which is controlled by the number of samples. Estimating a proxy model with more samples has increased cost, but results in a more stable explanation. In the plot below, the average variance of explanations over 10 runs is shown for LIME as a function of the number of samples. For a low number of samples, the variance is very large relative to permutation impact.



Sensitivity to hyperparameters

Aside from the parameters that affect randomness, explainers have additional hyperparameters that control the nature of the explanations. LIME has the kernel width, which dictates the effective neighborhood size around the point of interest. The kernel width greatly affects the resulting explanation. In the figure below, the top 9 most important features are shown for increasing kernel widths.

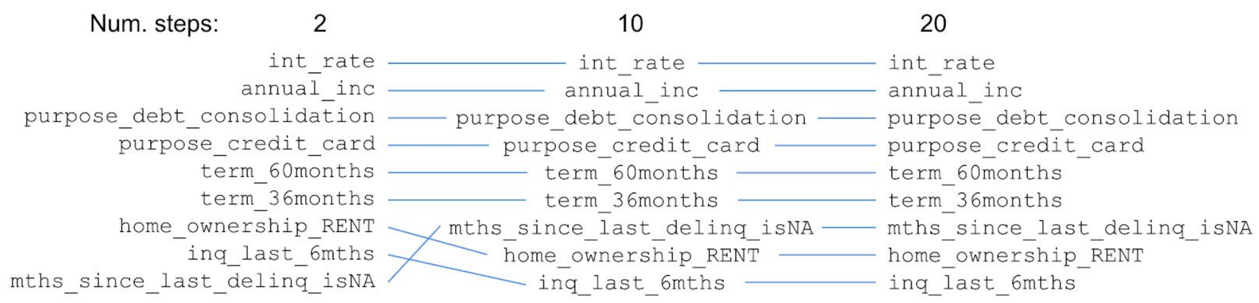


In addition to the kernel width, the sampling procedure itself is a design choice in LIME. Sampling a realistic local neighborhood around a point of interest is a very difficult task and common implementations of LIME model each feature as a univariate Gaussian random variable. This means that LIME explains the prediction of a model based on a neighborhood of samples which may be highly unrealistic.

The first row in the table below is an actual applicant in the Lending Club dataset, followed by 9 rows of application data generated by LIME as the neighborhood. The highly correlated variables of `loan_amnt` (loan amount) and `installment` are no longer correlated. Additionally, variables like `annual_inc` (annual income) may contain negative values. The model was not trained on data like these samples and they do not provide reliable insight about its behavior.

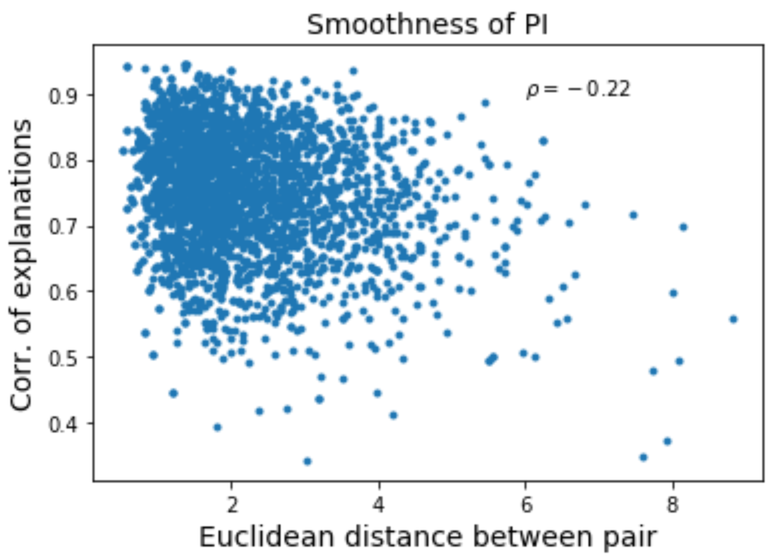
loan_amnt	int_rate	installment	annual_inc	dti	delinq_2yrs
2000.0	0.0662	61.41	50000.00	8.11	0.0
10338.0	0.1071	76.78	-30125.32	21.46	-1.0
17928.0	0.1565	152.43	92680.99	13.69	-0.0
5794.0	0.1322	493.70	79560.33	14.50	0.0
15998.0	0.0395	558.82	79275.55	14.03	0.0
11432.0	0.1101	-125.05	16307.60	18.46	-0.0
13813.0	0.1080	776.08	-36931.54	-1.40	1.0
2498.0	0.0356	426.58	20338.34	5.87	0.0
10847.0	0.1776	229.98	36622.50	20.42	-0.0
-7156.0	0.1369	489.08	27108.77	8.50	-0.0

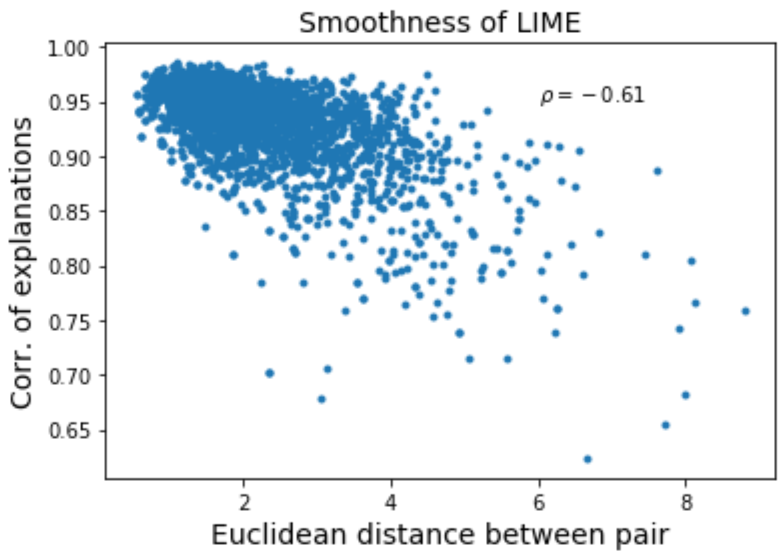
The ZAML explainer involves the computation of a path integral along the manifold of the data and is approximated with discrete steps. The number of steps determines the quality of the approximation. Taking the top 9 most important features for the number of steps set to 2, 10, and 20 reveals that the explanation is not overly sensitive to this parameter.

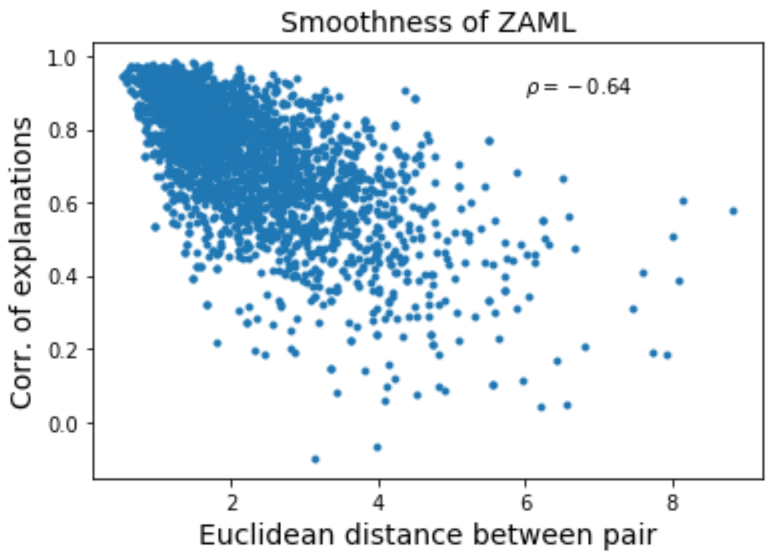


Smoothness

If two inputs to a model are very similar and receive the same scores from the model, then one would expect the explanations of these inputs to be similar as well. This property is known as smoothness. To measure smoothness, we find the nearest neighbor for each test applicant (if it received the same score from the model). For each pair of neighboring inputs, the distance between the two is calculated. Then the explanations of the two samples are compared with Spearman’s rank correlation coefficient. For a consistent explainer, if the distance between a pair of inputs is very small, then the explanation will have a Spearman’s coefficient near 1.


Smoothness of PI
$\rho = -0.22$
Corr. of explanations
Euclidean distance between pair
0.4
0.5
0.6
0.7
0.8
0.9
2
4
6
8


Smoothness of LIME
$\rho = -0.61$
Corr. of explanations
Euclidean distance between pair
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
2
4
6
8

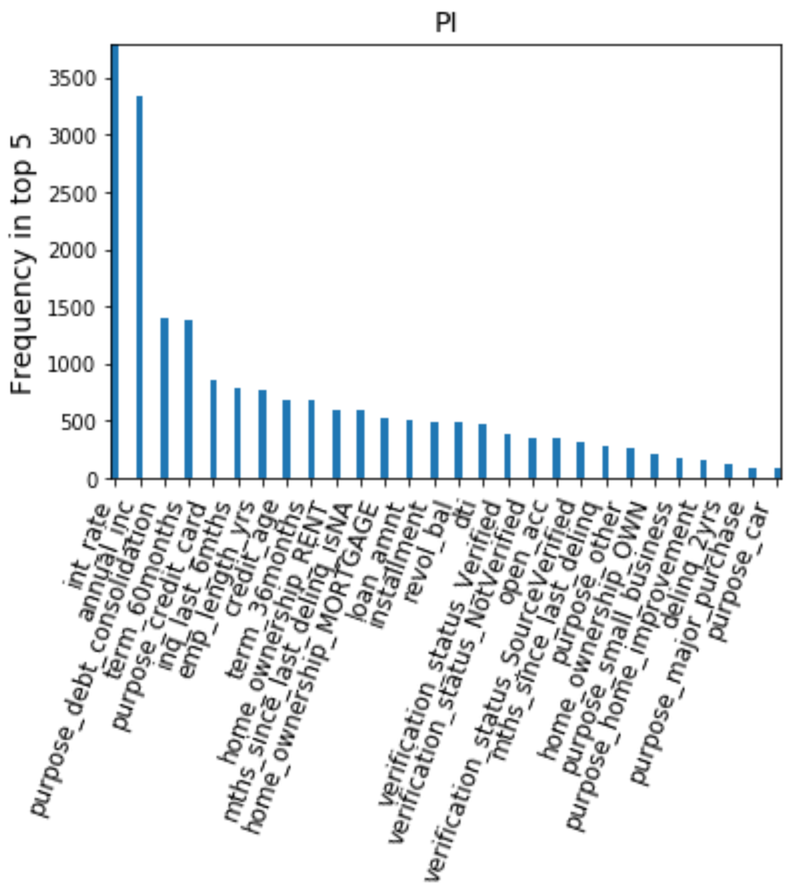


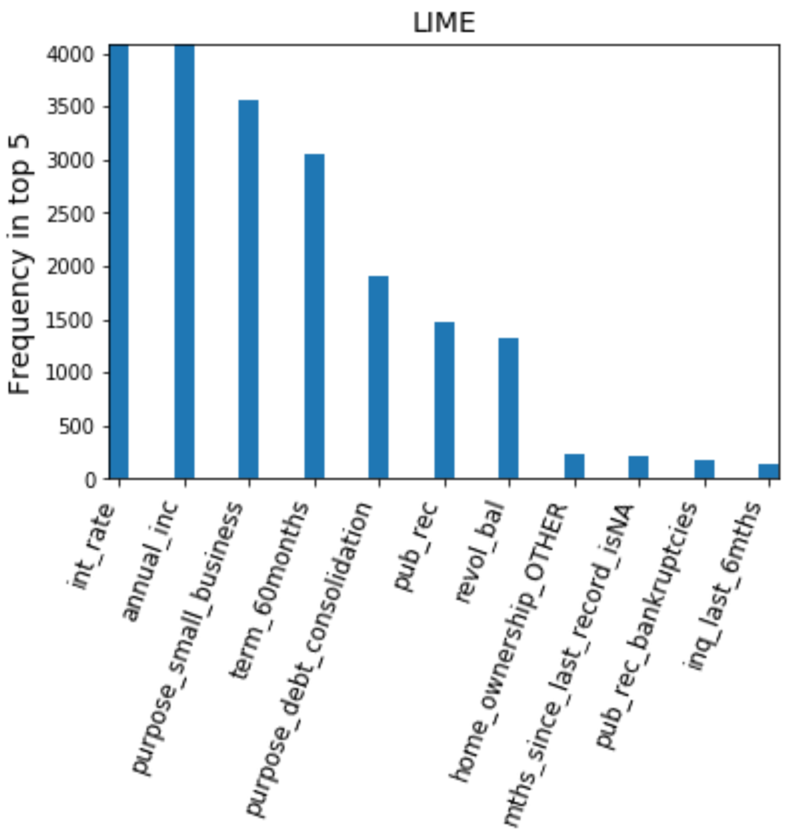
In the plots above, the ZAML explainer displays the strongest negative correlation between the distance of a sample pair and the similarity of their explanations, followed by LIME. For permutation importance, similar samples can result in quite different explanations. From a business perspective, smoothness in the space of explanations provides a sense of fairness in the model's decisions. Discrepancies in model outcomes for two similar scenarios creates dissonance in the human interpretability of the model's functionality.

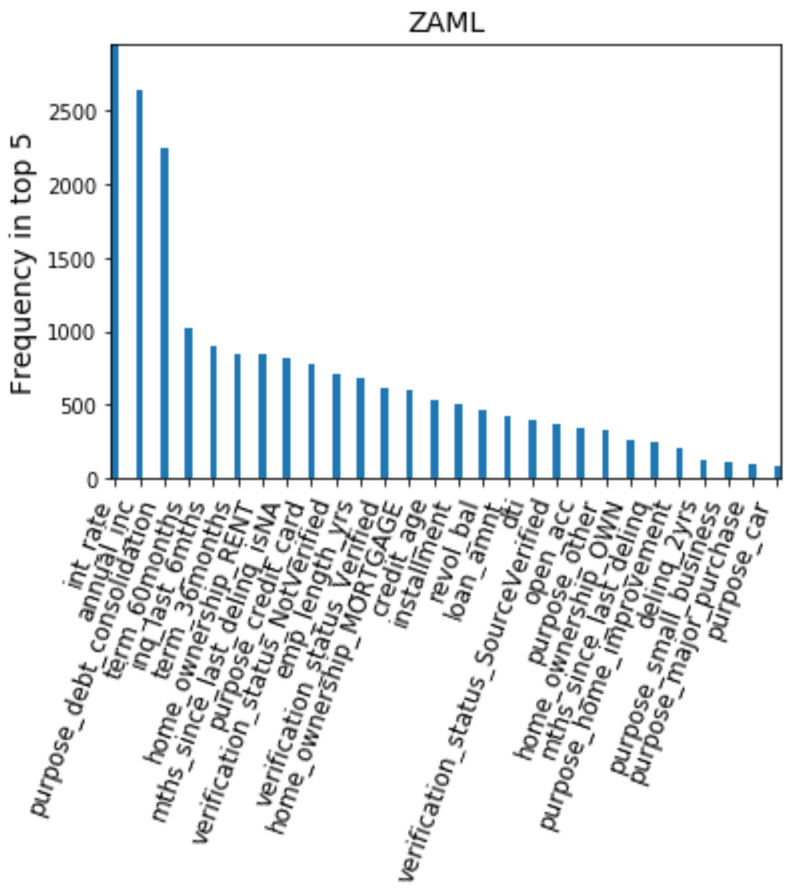
Precision

As we mentioned earlier, the precision of an explanation is based on how varied the explanations are across the data space. Modern machine learning models get their predictive power because they can model complex local behavior, extracting signals among different features in different regions of the data space. A global linear proxy model returns the same explanation for every input and is thus not a good candidate for explaining a nonlinear model like the one we built off the Learning Club data. To show the relative precision of the various techniques, we took the top 5 features for each sample in the testing set, and build a histogram of how often each feature appears in the top 5.

These histograms are shown below. ZAML and permutation impact show a few features which almost always are in the top 5. In fact, these methods have significant overlap in their explanations. The frequency of less common features decays gradually. LIME only selects 2 features with significant frequency: the interest rate and the flag indicating if a public record is available. The rest of the features receive very little attribution and essentially fluctuate as noise.


PI
Frequency in top 5
0
500
1000
1500
2000
2500
3000
3500
int_rate
annual_inc
purpose_debt_consolidation
term_60months
purpose_credit_card
inq_last_6mths
emp_length_yrs
credit_age
term_36months
home_ownership_RENT
mths_since_last_delinq_isNA
home_ownership_MORTGAGE
loan_amnt
installment
revol_bal
dti
verification_status_Verified
verification_status_NotVerified
open_acc
verification_status_SourceVerified
mths_since_last_delinq
purpose_other
home_ownership_OWN
purpose_small_business
purpose_home_improvement
delinq_2yrs
purpose_major_purchase
purpose_car


LIME
Frequency in top 5
4000
3500
3000
2500
2000
1500
1000
500
0
int_rate
annual_inc
purpose_small_business
term_60months
purpose_debt_consolidation
pub_rec
revol_bal
home_ownership_OTHER
mths_since_last_record_isNA
pub_rec_bankruptcies
inq_last_6mths



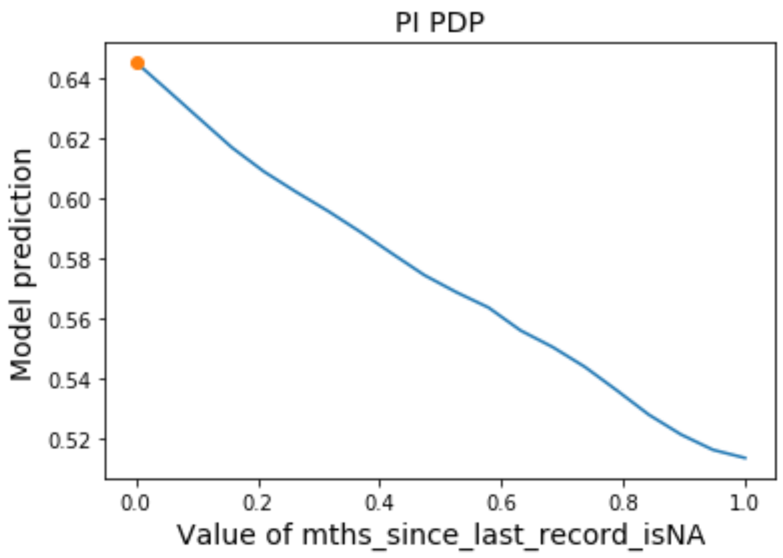
Why does LIME produce such unvaried explanations? The kernel width parameter must be set relatively high in order to keep a sufficient number of samples in the local neighborhood from which the linear proxy model is estimated. If the neighborhood size is too small, particularly with higher dimensional data, the linear model estimation will be ill-posed. Thus, a larger kernel width is favored, and the proxy model no longer accurately represents the neural network, but rather a highly smoothed version of it with no complex local behavior.

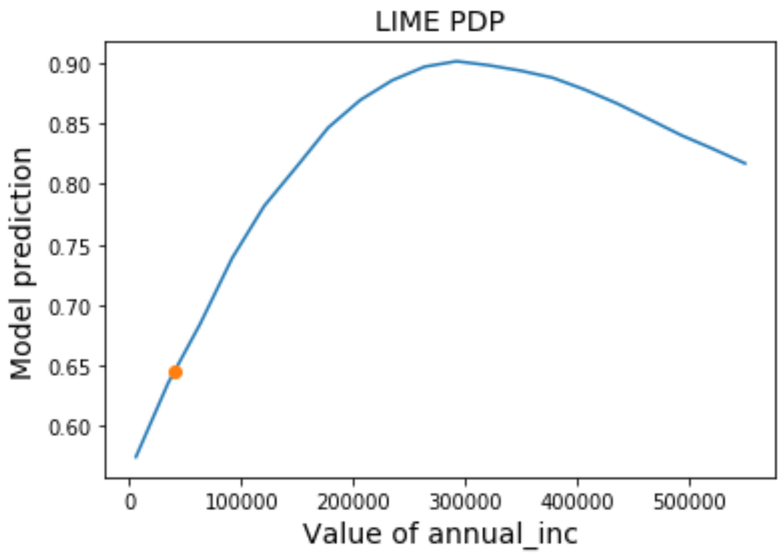
Accuracy

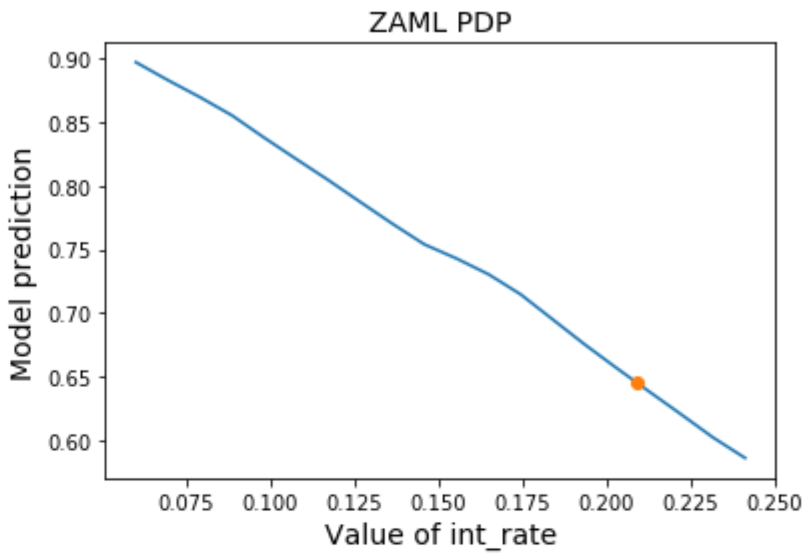
The accuracy of an explanation is especially important in the credit domain as the Fair Credit Reporting Act requires that all credit denials come with reasons that accurately describe the key factors that led to an applicant being denied. The adverse action notice provided to the customer must describe key factors a consumer can change in order to improve their likelihood of being approved. Accurate reasons are therefore required to comply with the law in the United States and many other jurisdictions.

To evaluate the accuracy of an explainer, consider the top feature for each loan applicant. We replicated the input, but replaced the top feature with every possible value in its range. The model was evaluated with these synthetic inputs to determine the maximum possible change in score from the original input. If this value was small, then the top feature was not really influential in the model's decision.

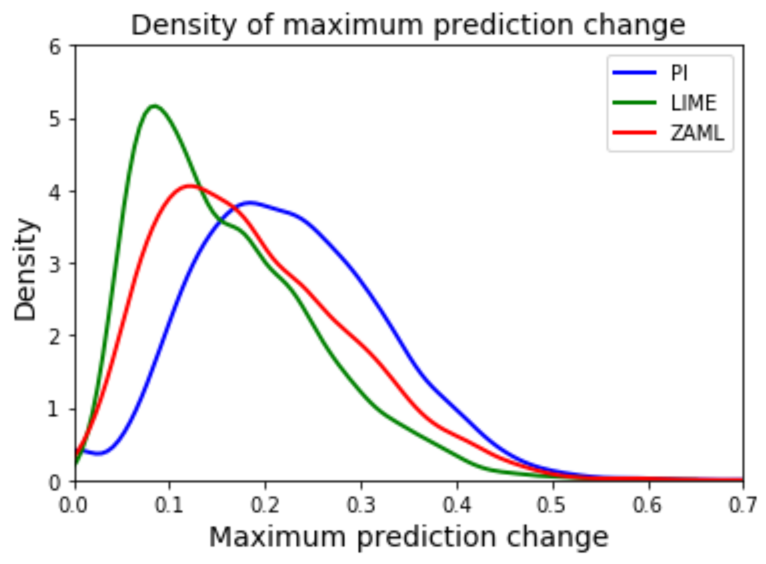
For a single input, this can be visually interpreted from a partial dependence plot (PDP). The x-axis represents the value of the feature of interest, and the y-axis represents the resulting model prediction. In the plots below, the impacts of the top features predicted by the 3 explainers are shown for a single input. In this case, all explainers produce a meaningful explanation.


PI PDP
Model prediction
0.64
0.62
0.60
0.58
0.56
0.54
0.52
0.0
0.2
0.4
0.6
0.8
1.0
Value of mths_since_last_record_isNA


LIME PDP
Model prediction
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
100000
200000
300000
400000
500000
Value of annual_inc



To compare the accuracies of the 3 explainers, we show the distributions of the maximum impacts of the top features for the test set. Permutation importance produces a top feature with the highest impact, which is a reasonable outcome considering this test is essentially what the permutation importance uses to produce its explanation. The ZAML explainer, while built from completely different principles, produces reasonably impactful top features. The top feature of LIME is not impactful. The LIME technique fails to identify the most impactful variables.



8 Conclusion

The techniques used today to explain linear models are not safe to use for machine learning, nor are many of the leading methods created to explain machine learning models. For explaining a neural network model, like the one we built using Lending Club data, LIME was not a reliable explainability technique. Sampling a local neighborhood becomes increasingly difficult as the number of features grows. Setting hyperparameters for LIME is unfortunately too much of an art given the sensitivity of the results to its parameters and the lack of ground truth. It is possible that for many problems, a single neighborhood size parameter will not work globally. The randomness in the algorithm can only be mitigated by drawing a large number of neighborhood samples, which increases computational costs.

Permutation importance and ZAML both present reasonable explanations, which is supported by the fact that their top features can be manipulated to greatly affect the model prediction. Both methods have significant overlap in the features they identify as important, which is additional evidence in the correctness of their results. The diversity of top 5 features shows that they both capture local behavior, rather than a single global explanation. ZAML provides a more consistent explanation in the sense that similar inputs receive similar explanations. It is also a deterministic algorithm, which is an important property, especially in credit, where fluctuations between two runs of the algorithm may be hard to defend.

Computational cost is an important property of explainers for systems that must run in near real-time. The cost of LIME is essentially fitting a linear model for each input that is to be explained, a task that quickly becomes intractable when a model has even just hundreds of variables. The cost of least-squares regression depends on the number of samples in the local neighborhood and the dimensionality of the data.

Permutation importance and the ZAML explanation can be compared more directly. For a dataset with *D* features, permutation importance requires *D*cycles* model evaluations to explain each input. The ZAML explanation requires model evaluations equal to the *number_of_steps* parameter. Experimentally, we observed stable explanations for permutation importance with about 50 cycles, and for ZAML with 10 steps. In a dataset with 100 features for example, PI will take about 500 times longer than ZAML, and this gap only increases with the dimensionality of the data.

9 Explainability At Zest

Zest has developed over the last few years wholly new model explainability methods that provide accurate and repeatable explanations at a row, segment, and global level for ensembles of heterogeneous ML models. The explainability is achieved through methods that cover differentiable and non-differentiable models. We have solved for numerical precision issues and provided methods for combining explanations of diverse submodels. This capability, when paired with ZAML's analysis, monitoring, and automated documentation enables lenders to safely and quickly apply advanced ML models in credit underwriting and in other regulated applications where consistency, accuracy, and performance is paramount.

10 References

1. Doshi-Velez, Finale, and Been Kim. "Towards a rigorous science of interpretable machine learning." arXiv preprint arXiv:1702.08608 (2017).
2. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)
3. Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.
4. Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.
5. Goldstein, Alex, et al. "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation." Journal of Computational and Graphical Statistics 24.1 (2015): 44-65.
6. Lei, Jing, et al. "Distribution-free predictive inference for regression." Journal of the American Statistical Association (2018): 1-18.
7. Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006.
8. Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, 2016.
9. Shapley, Lloyd S. "A value for n-person games." Contributions to the Theory of Games 2.28 (1953): 307-317.
10. Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions." Advances in Neural Information Processing Systems. 2017.
11. Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." arXiv preprint arXiv:1312.6034 (2013).
12. Kindermans, Pieter-Jan, et al. "PatternNet and PatternLRP–improving the interpretability of neural networks." stat 1050 (2017): 16.
13. Pirie, W. "Spearman rank correlation coefficient." Encyclopedia of statistical sciences 12 (2004).
14. Goodfellow, Ian, et al. Deep learning. Vol. 1. Cambridge: MIT press, 2016.

10.1 Internal References & Code

https://github.com/Katlean/Experimental/blob/master/users/mfe/experiments/simulations/Demo%20Explainer%20Issues.ipynb

Why Lenders Shouldn't 'Just Use SHAP'
To Explain Machine Learning Credit Models

Everyone wants to solve AI's black box problem: the dilemma of understanding how a machine learning (ML) computer model arrives at its decisions. The hard part is figuring out the influence of each of the hundreds or thousands of variables interacting in nearly infinite combinations to derive an outcome in an ML model.

In 2017 two computer scientists from the University of Washington published a technique for generating fast and practical explanations of a particular kind of ML called tree-based models (specifically, a variant called XGBoost). The algorithm's authors named their work SHAP, for Shapley additive explanations, and it's been used hundreds of times for coding projects.

The Shapley name refers to American economist and Nobelist Lloyd Shapley, who in 1953 first published his formulas for assigning credit to "players" in a multi-dimensional game where no player acts alone. Shapley's seminal game theory work has influenced voting systems, college admissions, and scouting in professional sports. Shapley Values work well in machine learning, too. The catch is that they're expensive to compute. In a game or model with just 50 variables you're already looking at considering more options than there are stars in the universe.

That's where SHAP comes in. SHAP approximates Shapley values quickly by cleverly using the tree structure of XGBoost models, speeding up the explanation time enough to make it practical to assign credit to each variable. Some banks and lenders eager to use machine learning in credit underwriting or other models are asking themselves, "Why not just use SHAP to power my explanation requirements?"

Fair question. The answer? Because that would be irresponsible for a bunch of reasons. For credit and finance applications, bridging from off-the-shelf SHAP to a safe application takes a lot of care and work, even if you just want to explain XGBoost models. Credit risk models must be treated particularly carefully because they highly regulated and significantly impact consumers' lives. When a consumer is denied credit, the Fair Credit Reporting Act of 1970 requires accurate and actionable reasons for the decision so that consumers can repair their credit and re-apply successfully.

SHAP is a practical solution for some use cases of ML, but in credit underwriting, it just doesn't hold water on its own. Here are a few reasons why we've faced serious challenges in our attempts to apply SHAP in credit risk -- and why we had to invent something new.

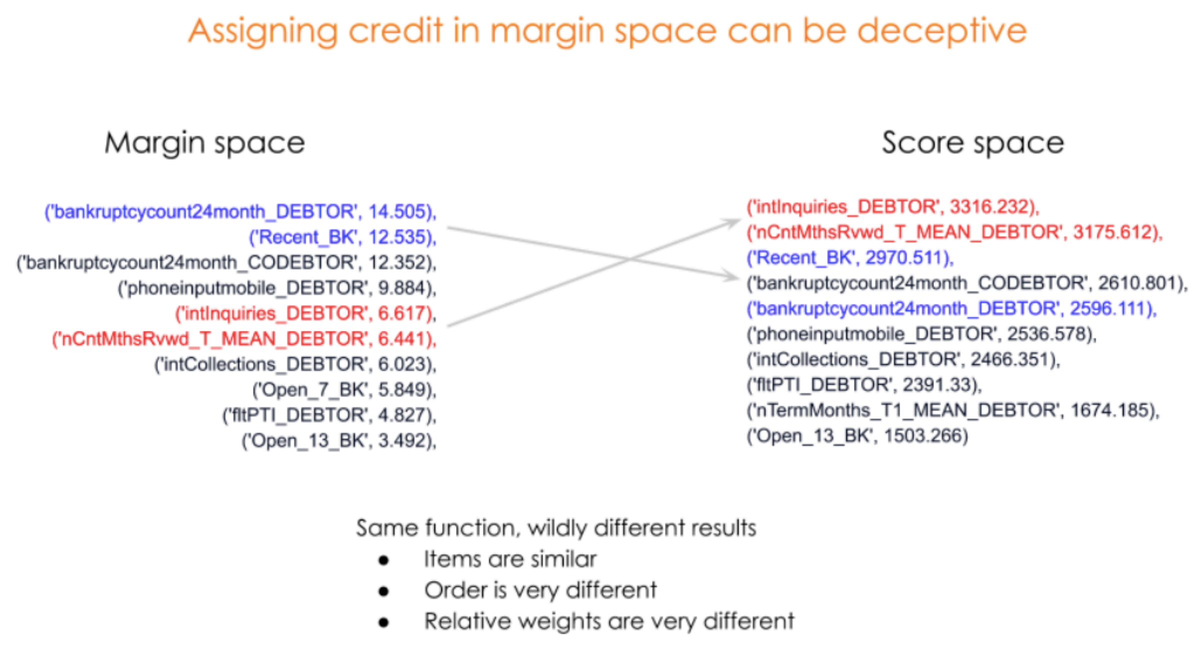
Score space vs margin space - these details really matter

Lending businesses want to be able to set a target and approve, say, 20% of applicants. That means the business wants a function that outputs numbers between 0 and 100, where 0 is the worst and 100 is the best, and for which exactly 20% of all scores lie above 80, 30% of all scores lie above 70%, and so on. This well-defined output is said to be in “score space."

The score space is very different mathematically from the credit model’s actual output, which is said to be in “margin space.” Margin space numbers fall in a narrow range from 0 to 1. In general, the relationship between the model’s actual output in margin space and the acceptance threshold in score space is extremely non-linear, and you have to transform the model’s output to generate the number the lending business wants. Don’t worry, you’re not the only one that struggles to keep track: we do too, and while technical, the margin space/score space transition really matters.

The problem with SHAP is that, because of the way it computes its Shapley values, it really only works in margin space. If you compute the set of weighted key factors in margin space, you'll get a very different set of factors and weights than if you compute them in score space, which is where banks derive their top five explanations for rejecting a borrower. Even if you are using the same populations and are only looking at the transformed values, you will not get the same importance weights. Worse, you likely won’t end up with the same factors in the top five.

The table below shows how this plays out for a real applicant for an auto loan. The reasons returned to the rejected borrower were dramatically different when translated from margin space to score space. If you skipped this important step, and just used SHAP out of the box, you would have thought the main reason for denial was the bankruptcy count. But the real top reason for denial, in score space, was the number of credit inquiries. A consumer relying on reasons generated by margin space attribution would be misled. Getting this wrong could have devastating consequences to consumers seeking to access financing for their first house or car, who rely on denial reasons to improve their ability to access credit. It could also cause a lender to run afoul of fair lending and fair credit rules.



Assigning credit in margin space can be deceptive

Margin space

```
('bankruptcycount24month_DEBTOR', 14.505),
('Recent_BK', 12.535),
('bankruptcycount24month_CODEBTOR', 12.352),
('phoneinputmobile_DEBTOR', 9.884),
('intInquiries_DEBTOR', 6.617),
('nCntMthsRvwd_T_MEAN_DEBTOR', 6.441),
('intCollections_DEBTOR', 6.023),
('Open_7_BK', 5.849),
('fltPTI_DEBTOR', 4.827),
('Open_13_BK', 3.492),
```

Score space

```
('intInquiries_DEBTOR', 3316.232),
('nCntMthsRvwd_T_MEAN_DEBTOR', 3175.612),
('Recent_BK', 2970.511),
('bankruptcycount24month_CODEBTOR', 2610.801),
('bankruptcycount24month_DEBTOR', 2596.111),
('phoneinputmobile_DEBTOR', 2536.578),
('intCollections_DEBTOR', 2466.351),
('fltPTI_DEBTOR', 2391.33),
('nTermMonths_T1_MEAN_DEBTOR', 1674.185),
('Open_13_BK', 1503.266)
```

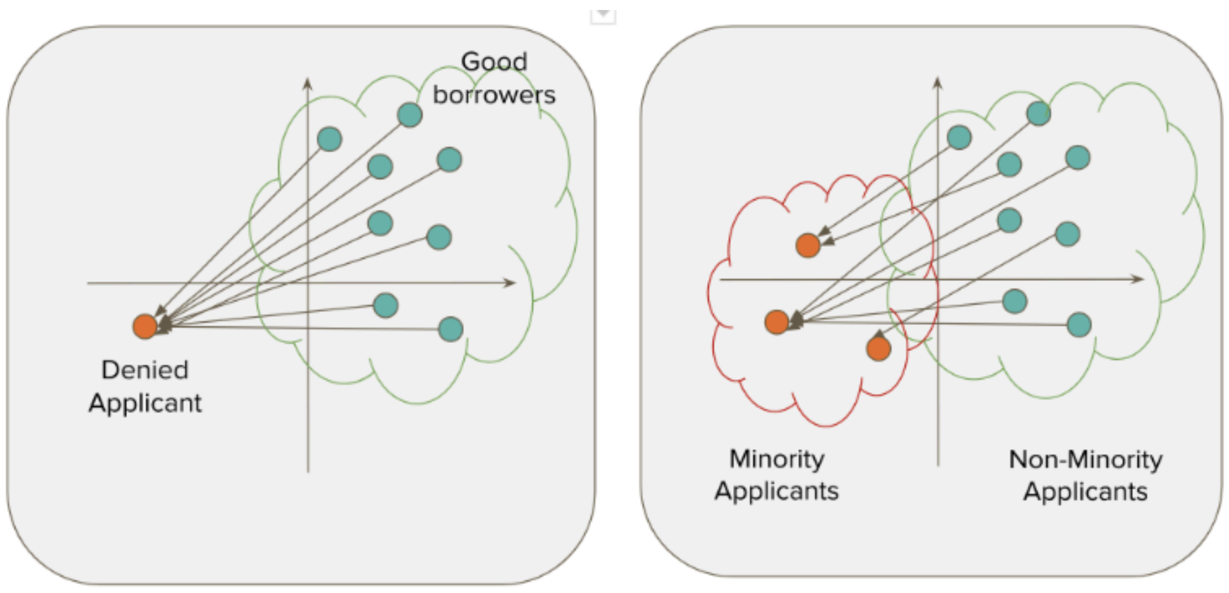
Same function, wildly different results

- Items are similar
- Order is very different
- Relative weights are very different

Why does this happen? Because SHAP derives its values by looking at all the results of taking a path down each tree in the model, and it assumes that the sum of the values along a set of paths down a tree gives you the score -- basically, you can compute the score with only the data in the trees. That's not true when you transform into score space; the transformation destroys that structure. SHAP can also have trouble recovering even a simple model's internal structure, as we'll explain in the last point.

Explanation by reference

SHAP computes variable importance globally, which means it shows how the model behaves for every applicant (in margin space) with respect to the overall model itself. In credit risk modeling, it is often required to understand an applicant's score in terms of another applicant or applicant population, that is, with respect to a reference population. For example, when lenders compute the reasons an applicant was rejected (for adverse action notices), they want to explain the applicant's score in terms of the approved applicants. When they do disparate impact analysis lenders want to understand the drivers of approval rate disparity. This requires comparing the feature importance for the population of, say, white non-Hispanic male applicants to protected groups and performing a search for less discriminatory alternatives. These are illustrated in the diagrams below.



Left: Adverse action requires comparing the denied applicant to good borrowers.

Right: Fair lending analysis requires comparing minority applicants with non-minority applicants.

There are many details you need to get right in this process, including the appropriate application of sample weights, mapping to score space at the approval cut-off, sampling methods, and accompanying documentation. Out of the box, SHAP doesn't allow you to easily do this.

You want to use modeling methods other than XGBoost

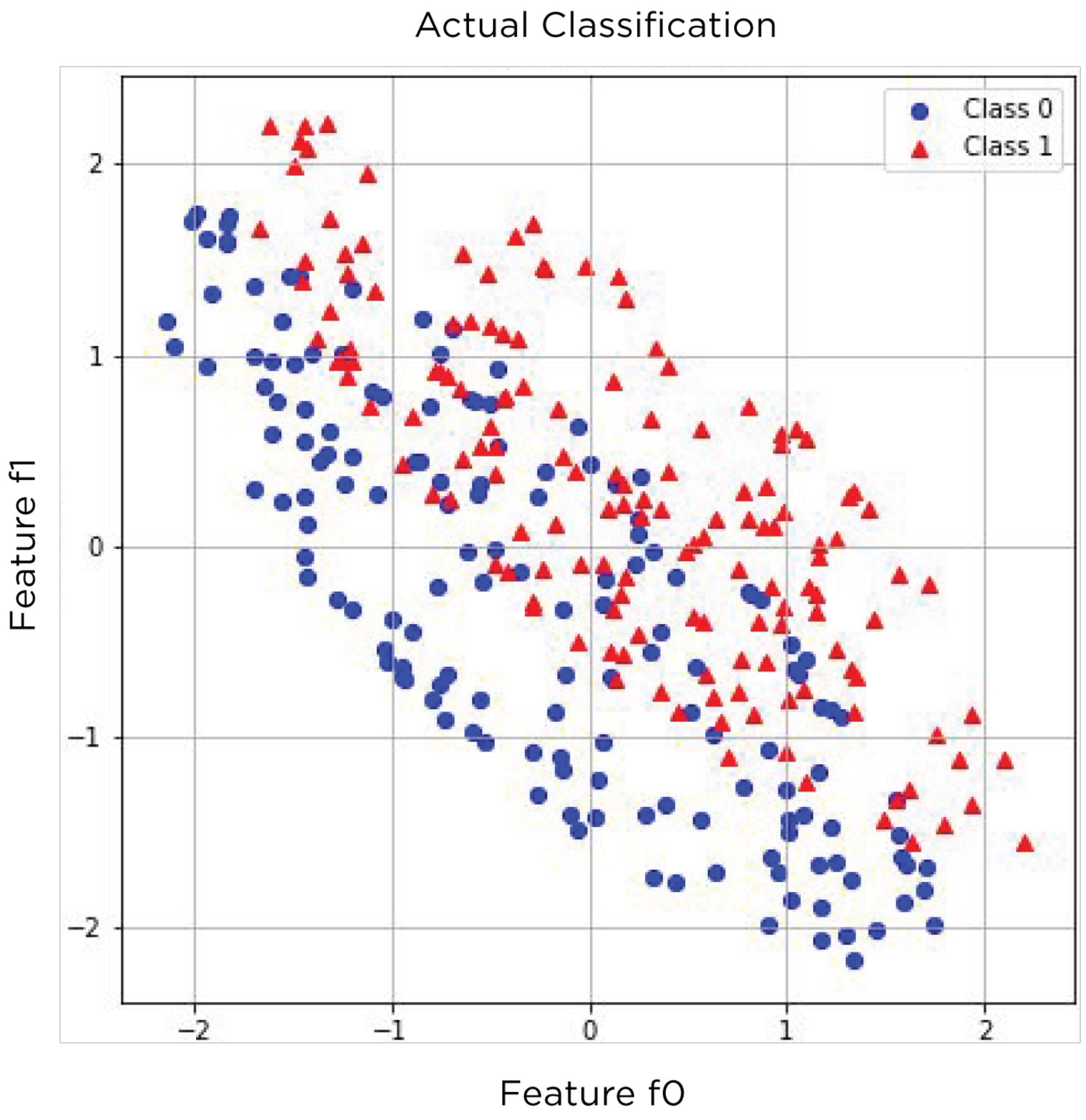
Using SHAP is hard enough because it outputs values in margin space that you have to correctly map into score space. But it has other important limitations. Although SHAP provides fast explanations for gradient-boosted tree models, there are many other mechanisms for building scoring functions, including many alternative forms of tree models such as random forests and extremely random forests, not to mention other implementations of gradient boosting such as LightGBM.

You may also want to use continuous modeling methods such as radial basis function networks, Gaussian mixture models, and, perhaps most commonly, deep neural networks. The current implementation of SHAP cannot explain any of the other types of tree models, and cannot explain any continuous model outside a small collection, and only by importing algorithms other than SHAP.

What's more, SHAP cannot explain ensembles of continuous and tree-based models, such as stacked or deeply stacked models that combine xgboost and deep neural networks. In our experience (and the experience of others), these types of ensembled models are more accurate and stable over time. That's why we built ZAML to explain a much wider variety of model types, enabling you to use world-beating ensembled models to drive your lending business.

Even on a simple XGBoost model, SHAP fails to uncover the underlying geometry

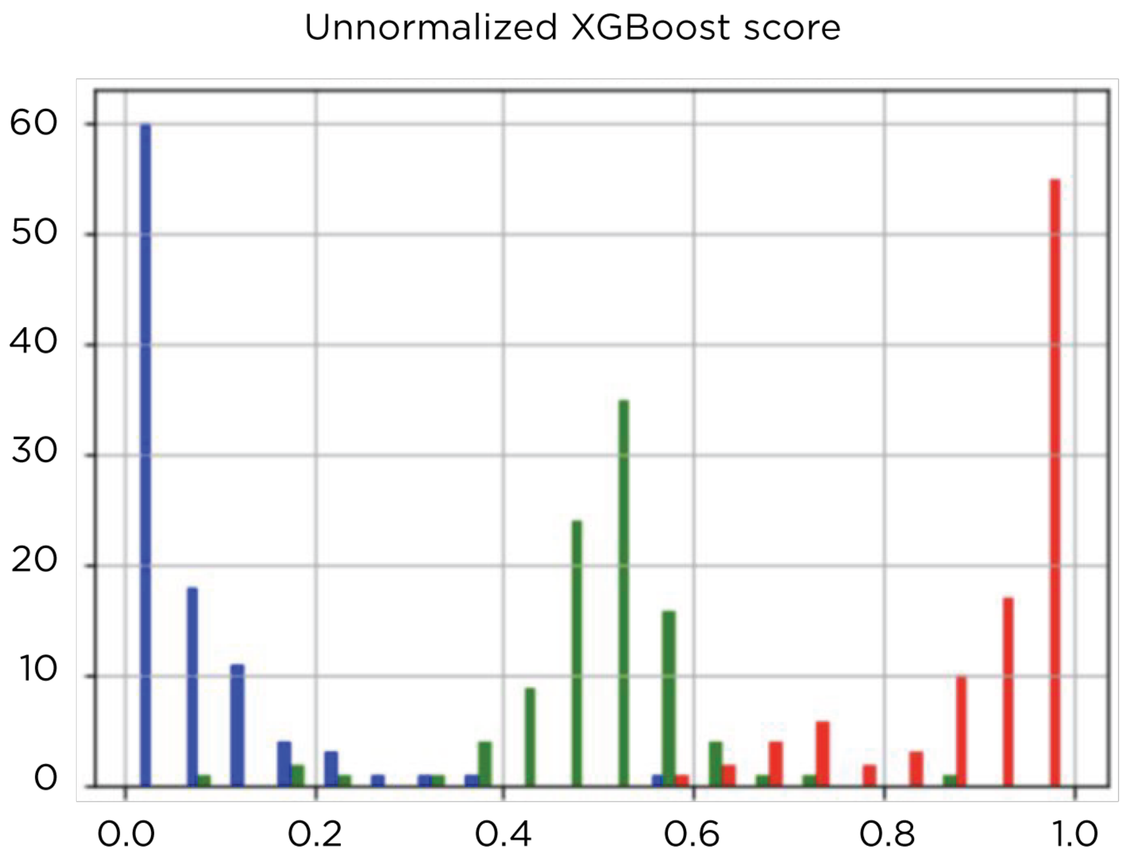
Machine learning models are effectively geometric entities: they embody the idea that things near to one another will tend to be mapped to the same place and then produce systems which reflect that structure. A good example of this is the ovals dataset, a two-dimensional dataset consisting of a set of points drawn uniformly from two overlapping ovals with the same number of points drawn from each. The ovals from which the points are drawn are arranged roughly vertically in the chart below, and the model is trained to predict membership in one or the other oval given the coordinates of a point. For convenience, the oval with a greater y value is arbitrarily assigned the target value 1 and the oval with the less y value is assigned the value 0.



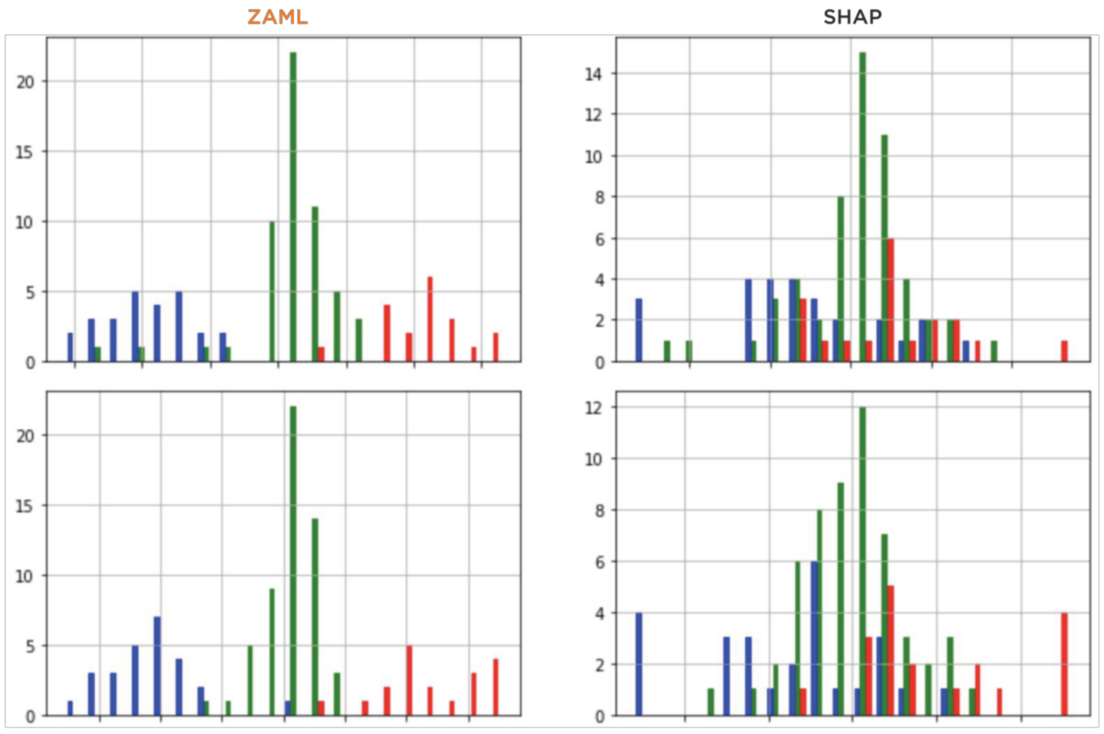
When viewed geometrically, this dataset is inseparable: points in the overlapping region are equally likely to have been drawn from either of the two ovals and so no classifier can predict membership for any such point.

Intuitively, one would expect a classification function defined for the ovals dataset to correspond to three regions: a region of points belonging only to the upper oval, a region of points common to the two ovals, and a region of points belonging only to the lower oval. We trained an XGBoost model on a random sample of half of the ovals dataset, and looked at the model's predictions on the other half.

The chart below shows the model's predictions, and we can see the three regions we expected. The blue represents scores the model assigned to the bottom region, the green the middle, and the red, the top. As you can see the model produces nicely separated outputs.



We should see that same separation when we look at the explainer outputs. One would intuitively expect that items in the upper region will have average attributions which are relatively large and positive, items in the common area to have attributions which are relatively close to zero, and items in the lower region to have average attributions which are relatively large and negative. If the explainer doesn't reflect this structure, it isn't really explaining the model, and probably shouldn't be trusted. To investigate that question, we compared SHAP attribution weights with the attribution weights generated by Zest's ZAML software in the charts below.



Let’s walk through what we’re seeing. The left column shows the feature importance for each model prediction, as assigned by ZAML. The right column shows the feature importance for each model prediction as generated by SHAP. The top row is the feature importance for f0, the x coordinate in our ovals dataset. The bottom row is the feature importance for f1, the y coordinate. The blue, green and red colors correspond to the bottom, middle and top regions, respectively.

As you can see, ZAML readily separates the top, middle, and bottom regions -- notice how the blue green and red bars are all nicely separated in the charts in the left column -- while SHAP, shown on the right, gets them all jumbled up. The results suggest that SHAP may not be the right tool to use off the shelf for the rigorous and regulated requirements of credit underwriting.

We did not expect these results when we first saw them, and frankly we thought they were wrong. After careful review by multiple teams inside and outside the company, however, we’re confident they’re not. Look for a scientific paper describing our algorithm, a mathematical proof of its correctness and uniqueness, and other empirical results to be published soon. In the meantime, if you care about getting your model explanations right, feel free to reach out to us.

SHAP was a giant leap forward in model explainability. The use of a game-theoretic framework to explain models is powerful and creative. Nonetheless, as the above analyses show, you really need more than just the out-of-the-box SHAP to provide the kind of accurate explanations required for real-world credit decisioning applications.

Even on a simple XGBoost model, SHAP can provide inaccurate explanations, and care must be taken to map into score space correctly and to mitigate numerical precision issues, when computing explanations by reference. Before diving head first into ML explainability with SHAP, it is important to understand its limitations and determine whether or how you will address those limitations in your ML application. Credit decisions make lasting impacts on people's lives and getting the explanations right matters.

FAQs ZestFinance has received on the Application of the Federal Agencies' Banking Supervisory Guidance on Model Risk Management to Machine Learning Models

NOTE TO READERS: ZestFinance, Inc. helps lenders transition from conventional underwriting methods to machine learning-based underwriting. In the course of our work, we often receive questions from executives at financial institutions about the applicability of the federal banking agencies' Supervisory Guidance on Model Risk Management (the Guidance) to machine learning models.[1] The Guidance clearly applies to machine learning-based underwriting models. Machine learning models, however, differ from traditional models in ways that raise unique issues regarding their evaluation, testing, and documentation under the Guidance. The FAQs below reflect the questions Zest receives most frequently on this issue. The accompanying answers set forth Zest's current views on best practices for the responsible adoption of machine learning models consistent with the Guidance, as well as the goals of ensuring safety and soundness in the financial system, increasing access to credit, and minimizing fair lending and other compliance risk.

SUMMARY

The financial services industry is increasing its adoption of machine learning (ML) for a range of applications. ML models are powerful at predicting outcomes because they can consider more data than traditional models and apply sophisticated mathematical techniques to evaluate multiple variables and the relationships between them, and continually refine and improve their underlying algorithms to enhance performance and predictive power on an ongoing basis. ML technologies have the potential to bring unbanked and underbanked consumers into the financial system, enhance access to responsible credit, and contribute positively to the overall safety and soundness of the financial system. Increased predictive power, however, comes with increased

[1] The Guidance was issued by the Board of Governors of the Federal Reserve System (FRB) and the Office of the Comptroller of the Currency (OCC) in 2011, and adopted by the Federal Deposit Insurance Corporation (FDIC), with technical conforming changes, in 2017.

complexity. The use of these advanced new modeling techniques in financial services raises important issues of risk management.

The Guidance establishes a framework for effective model risk management that focuses on appropriate development, documentation, validation, and governance standards for models used by financial institutions. The guidance applies to all modern modeling techniques, including machine learning models. However, the Guidance was adopted in 2011, largely before ML was common, and thus does not address ML models. While the principles articulated in the Guidance remain sound and appropriate for all models, certain particulars and examples in the Guidance do not reflect the way ML models function. Different validation approaches, built largely upon current approaches, are more effective at meeting the Guidance's goals when using ML models.

These FAQs cover the application of the Guidance to the use of ML models by financial institutions and describes methods for complying with key aspects articulated in the Guidance given the unique risks posed by ML techniques. The questions below do not address all aspects of the Guidance; instead, they are the questions Zest is most frequently asked by executives at financial institutions considering the use of machine learning. The accompanying answers represent Zest's current thinking on how financial institutions may use ML models responsibly and consistent with the Guidance.

SECTION III: OVERVIEW OF MODEL RISK MANAGEMENT

For new machine learning models, can lenders use techniques for model risk management different from those outlined in the Guidance that they determine to be more appropriate for such models?

- Yes. The Guidance is not prescriptive, but illustrative. The selection of model risk management techniques should be based, in part, on the type, complexity, and functional attributes of the model. ML models operate differently than traditional models; thus, it is appropriate to consider alternative approaches to model risk management.

Is it acceptable to use machine learning models in high stakes financial services decision-making?

- Yes. Nothing in the Guidance precludes the use of machine learning models. The Guidance applies to a financial institution's use of any "model," which it defines as a "quantitative method, system, or approach that applies statistical, economic, financial, or mathematical theories, techniques, and assumptions to process input data into quantitative estimates." ML models fit squarely within this definition.

- Fairness, anti-discrimination, and safety and soundness goals tend to support the use of more predictive models, including machine learning models. As many as fifty million Americans have incomplete or inaccurate credit bureau data. Millions of these consumers are denied access to credit by lenders using conventional, static credit scoring techniques because those models often inaccurately predict default risk. Machine learning models are resilient to incomplete data, able to consider more variables, and capable of creating models that more accurately assess credit risk. Consequently, ML's enhanced predictive power has the potential to safely expand access to credit while reducing losses and systemic risk.

- However, ML-based credit risk models must be validated, documented, and monitored using methods appropriate to the modeling approach selected in order to comply with the principles articulated in the Guidance. As discussed below, conventional validation approaches are not sufficient to evaluate ML models. ML model developers and institutions should take care to conform their practices to the principles in the Guidance regarding Model Development Implementation and Use, and Model Evaluation and Verification standards using techniques robust enough to assess and explain the performance of ML models.

SECTION IV: MODEL DEVELOPMENT, IMPLEMENTATION, AND USE

Can you use as many variables as desired in a model?

- Yes. The Guidance does not address, or limit, the number of variables that may be used in a model, and nothing in the Guidance suggests that fewer variables necessarily decreases risk. ML models can consider many more variables than traditional methods, which is a key reason why ML models often provide greater predictive power, and deliver superior results, compared to traditional models.

- The same data review and documentation practices outlined in the Guidance still apply to ML models even though ML models consider many more variables than traditional models. As the Guidance indicates, "there should be rigorous assessment of data quality and relevance, and appropriate documentation. Developers should be able to demonstrate that such data and information are suitable for the model."

Can model developers analyze vastly more variables and still comply with the Guidance?

- Yes. As the Guidance states: “Developers should be able to demonstrate that such data and information are suitable for the model and that they are consistent with the theory behind the approach and with the chosen methodology. If data proxies are used, they should be carefully identified, justified, and documented.”
- ML models consider hundreds or even thousands of variables, so it may be impractical to manually review all of them. An automated variable review may be the most effective way to support comprehensive analysis and documentation of the data and the model. Automated variable review methods should identify and document data issues that could raise questions about the predictive power, fairness, and safety and soundness of a model. Notably, variables should be reviewed for unexpected and/or inconsistent distributions, mappings, and other data degradation issues that can lead to model misbehavior. In connection with reviewing data variables, ML models will detect patterns and relationships among variables that no human would detect. This continuously evolving multivariate analysis is what makes any assessment of the data during the development phase problematic. The Guidance calls for documentation of these review methods and descriptions of the assumptions and theoretical basis for their use.

SECTION V: MODEL VALIDATION

What methods are permissible for assessing the soundness of an ML model?

- The Guidance does not prescribe any specific method for validating any model, including a machine learning model. Nonetheless, the Guidance sets out a core framework for effective model validation: evaluation of conceptual soundness, ongoing monitoring, and outcomes analysis.
- Regarding soundness, certain conventional evaluation methods described in the Guidance would, if applied to ML models, be ineffective and would likely produce misleading results. For example, one of the testing methods identified by the Guidance is sensitivity analysis. Common implementations of sensitivity analysis include exploring all combinations of inputs and permuting these inputs one-by-one (univariate permutation) in order to understand the influence of each variable (or a combination thereof) on model scores. Exploring all combinations of inputs (exhaustive search) is computationally infeasible for most ML models. Univariate permutation (permuting inputs one-by-one), while more computationally tractable, yields incorrect results for ML models that capture and evaluate multivariate interactions.
- Effective ML model evaluation techniques should be efficient and tractable, and designed to test the how ML models actually work. Such techniques should also

assess the impact of multivariate interactions because ML models evaluate such interactions. Appropriate methods of evaluating ML models include techniques derived from game theory, multivariate calculus, and probabilistic simulation.

How do the Guidance's monitoring standards apply to ML models?

- The Guidance calls for ongoing model monitoring: "Such monitoring confirms that the model is appropriately implemented and is being used and is performing as intended..." The Guidance further states: "Many of the tests employed as part of model development should be included in ongoing monitoring and be conducted on a regular basis to incorporate additional information as it becomes available."
- A thorough approach for monitoring ML models should include:
    - Input distribution monitoring: Recent model input data may be compared with model training data to determine whether incoming credit applications are significantly different from model training data. The more that live data differs from training data, the less accurate the model is likely to be. This data comparison is typically done by looking at variable distributions and ensuring recent data is drawn from a similar distribution as occurred in the model training data. For ML models, multivariate input variable distributions should be monitored to identify input data where combinations of values that were unlikely to appear together during model development are now occurring in production. Systems for monitoring model inputs should trigger alerts to monitors or validators when they spot anomalies or shifts that exceed pre-defined safe bounds.
    - Missing input data monitoring: Comprehensive model monitoring should include monitoring for missing input data. Model input data comes from a variety of sources, some of which is retrieved over networks from third parties. Data sources could become unavailable in production. A complete model monitoring program should monitor and trigger alerts to monitors and validators when the rate of missing data, and its impact on model outputs and downstream business outcomes, exceed pre-defined thresholds.
    - Output distribution monitoring: Model outputs should be monitored by comparing distributions of model scores over time. Monitoring systems should compute statistics that establish the degree to which the score distribution has shifted from the scores generated by the model in prior periods such as those contained in training and validation data sets.

- Execution failure monitoring: Error and warning alerts generated during model execution can indicate flaws in model code that may affect model outputs. Such alerts should, therefore, be closely monitored, the causes of such alerts should be investigated and identified, and appropriate remediation should be implemented where necessary.

- Latency monitoring: Model response times should be monitored to ensure model execution code and infrastructure meet the latency requirements of applications and workflows that rely on model outputs. Models that perform slowly or with unreliable execution time may cause intermittent timing issues, which can result in the generation of inaccurate scores. Establishing clear latency objectives and pre-defined alert thresholds should be part of a comprehensive model monitoring management program.

- Economic performance monitoring: A complete ML model monitoring solution should include business dashboards that enable analysts to configure or pre-define alert triggers on key performance indicators such as default rate, approval rate, and volumes. Substantial changes in these indicators can signal operational issues with model execution and, at a minimum, should be investigated and understood in order to manage risk.

- Reason code stability: Reason codes explain the key drivers of a model's score. Reason code distributions should be monitored because material changes to the distributions can indicate a change in the character of the applicant population or even in the decision-making logic of the ML model.

- Fair lending analysis: Machine learning models can develop unintended biases for a variety of reasons. Relatedly, like any model, ML models can result in disparities between protected classes. To ensure that all applicants are treated fairly and in a non-discriminatory manner, it is important to monitor loan approvals, declines, and default rates across protected classes. Historically, this monitoring has been done far after the fact. Because of the possibility of bias and the advanced predictive fit of ML models, monitoring of these models should occur in real time.

Should model monitoring include automation?

- Yes. The Guidance states: "monitoring should continue periodically over time, with a frequency appropriate to the nature of the model." Given the complexity of ML models, automated model monitoring, which can run concurrently with model operations, is essential to meet the expectations set by the Guidance, especially when combined with multivariate input monitoring and alerts. Changes to input and output distributions should be monitored in real time to identify problems promptly and reflected in periodic reports.

How should model outcomes be analyzed?

- As the Guidance recommends, model outcomes should be thoroughly understood prior to adoption and deployment of any new model, including ML models. Because machine learning models can consider many more data points than traditional models, traditional tools such as manual review of partial dependence plots can be cumbersome or inaccurate. Such tools can also miss crucial aspects of ML model behavior, such as the influence of variable interactions. In addition to understanding fully how a model arrives at a score, it is important to understand the swap sets generated by switching to a new model: that is, which applicants will now be approved (swap-ins) and which will now be denied (swap-outs). While the quantity of applicants is important, so is the quality of applicants. Outcomes validation methods should include an examination of the distribution of values for all model attributes of swap-ins and swap-outs, as well as a comparison with populations already accepted and with known credit performance.

SECTION VI: GOVERNANCE, POLICIES, AND CONTROLS

How do the Guidance documentation requirements apply to ML models?

- As the Guidance states, "documentation of model development and validation should be sufficiently detailed so that parties unfamiliar with a model can understand how the model operates, its limitations, and its key assumptions."
- Meeting the requirement for thorough documentation of advanced modeling techniques can be challenging for model developers because ML models can process many more variables than traditional models, ML algorithms often have many tunable parameters, ML "ensembles" can join both many variables and many tunable parameters, and all of these must be thoroughly documented so the model can be reproduced.
- These issues largely do not apply to logistic regression-based underwriting models, which are easier to understand and explain but less predictive.
- In the case of ML models, documenting how a model operates, its limitations, and its key assumptions requires using explainability techniques that accurately reveal how the model reached its decisions and why.

- Entities should ensure that they use explainability methods that accurately explain how a model operates. Most commonly used explainability methods are unable to provide accurate explanations. For example, some methods (e.g., LOCO, LIME, PI) look only at model inputs and outputs, as opposed to the internal structure of a model. Probing the model only externally in this way is an imperfect process leading to potential mistakes and inaccuracies. Similarly, methods that analyze refitted and/or proxy models (e.g., LOCO and LIME), as opposed to the actual final model, result in limited accuracy. Explainability methods that use "drop one" or permutation impact methods (e.g., LOCO and PI) rely on univariate analysis, which fails to properly capture feature interactions and correlation effects. Finally, methods that rely on subjective judgement (e.g., LIME) create explanations that are both difficult to reproduce and overly reliant on the initial judgement. These errors in explanation cause model accuracy to suffer. Even slight inaccuracies in explanations can lead to models that discriminate against protected classes, are unstable, and/or produce high default rates. Models that rely on mathematical analyses of the underlying model itself, including high-order interactions, and do not need subjective judgement are appropriate explainability methods.

  Should model documentation include automation?

- Yes. Although the Guidance is silent on whether model documentation may be generated automatically, automated model documentation is the most practical solution for ML models. ML model development is complex, and operationalizing and monitoring ML models is even harder. It is not feasible for a human, unaided, to keep track of all that was done to ensure proper model development, testing and validation. There are tools to automate model documentation for review by model developers, compliance teams, and other stakeholders in the model risk governance process. Given the number of variables in ML models, automated documentation is likely to provide a higher degree of accuracy and completeness than manual documentation. In general, participants in model risk management should not rely upon manually generated documentation for ML models.

Are there other best practices for ML model risk management?

- Yes. The Guidance makes clear that the quality of a bank's model development, testing, and validation process turns in large part on "the extent and clarity of documentation." Therefore, model documentation should be clear, comprehensive, and complete so that others can quickly and accurately revise or reproduce the model and verification steps. Documentation should explain the business rationale for adopting a model and enable validation of its regulatory compliance.

- Records of model development decisions and data artifacts should be kept together so that a model may be more easily adjusted, recalibrated, or redeveloped when conditions change. Such artifacts include development data, data transformation code, modeling notebooks, source code and development files, the final model code, model verification testing code, and documentation.
- Model documentation should be clear, comprehensive, and complete so that others can quickly and accurately revise or reproduce the model and verification steps. Documentation should explain the business rationale for adopting a model and enable validation of its regulatory compliance.