Explainable Machine Learning for Credit Lending

CMPE257 Machine Learning Term Project

Praveena Manikonda
Kai Kwan Poon
Courtney Nguyen

San Jose State University
Professor Ming-Hwa Wang

May 17th, 2020

Acknowledgments

We would like to express our gratitude to the essential workers supporting the world as COVID-19 threatens the lives of many at the time of writing this paper. We also thank Dr. Ming-Hwa Wang, Department of Applied Data Science for his lectures on Machine Learning and imparting current industry knowledge.

We would also like to thank our family members for their support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

Black box algorithms make predictions based on large-scale input training data. Although they repeatedly demonstrate high accuracy rates and often tout a high degree of confidence, they are not always concordant with logical human explanations. Oftentimes, they are not even capable of providing a reason for the predictive output. For high-stakes decisions such as credit lending, consumers want to know exactly why a computer chose to approve or deny their loan application. Unfortunately, it is difficult to protest against a denied application, default risk, or other financial maladies based on skewed training data, unintended biases, or input errors when the black box is not fully understood by the agent or in some cases, even the developers themselves. To shed light on the black box, there is growing interest in a new field of study called 'Explainable Machine Learning.' This subfield aims to clarify the classification rules behind the final prediction or recommendation. These models are built post-hoc or after the deployment of the first black model, such that the decision-making process in the first model can be understood by humans. To explore a human intuitive solution relative to credit lending, we describe and use two advanced explainable machine learning techniques in the field: LIME and SHAP. In this paper, we deploy LIME and SHAP to provide users a logical explanation as to why their credit decision was made, and use the results to assess whether or not the decisions are valid, or if the explanation is even viable. This allows consumers to understand why or why not a particular prediction was made, and provide more understanding of when the algorithm succeeds or when it fails.

Keywords: explainable AI, Lime, SHAH, logistic regression, credit lending

1. Introduction

1.1. Objective

Today, some numerous models or algorithms use statistical techniques to learn how to classify data or predict outcomes in 'artificial intelligence' or 'machine learning.' Deep learning, or neural networks, go further into mimicking how human brains function, and is "one of the most common iterations of machine learning" [10]. Neural networks can get so complicated that they are completely unreadable, except for the input and outputs. The logical leaps that occur between input features ultimately make a final prediction that can be far beyond the human capacity to understand. This is the premise of the black box system.

There are three main components in these black box systems: an input, a model or algorithm, and an output. The term 'black box' is given to these algorithms because the inner workings, or what happens in between the input and output, is a complete mystery. Briefly, an input is put into the model and a series of hidden layers go on to group the inputs based on patterns discovered by the algorithm. The final layer is the output or the decision that the system makes.

With the proliferation of artificial intelligence and machine learning algorithms in the world, it is getting more and more important for human beings to be comfortable with using these perplexing systems. However, the usages are limited in highly regulated spaces that require logical explanations for final decisions. Explainable

machine learning can provide insight into the specific reasons why a particular outcome was recommended by the elusive black box.

For this paper, we envision an explainable system that, based on the client's financial history as an input, can inform the credit lending agent and explain exactly how and why the black box decided to approve or deny loans to clients. The agent can then review decision-making processes or reports to ensure that no inappropriate biases were influenced by typographical errors or skewed input data.

1.2. Identifying the Problem

For any company to not identify nor take advantage of prosperous opportunities from their data is a naive business mistake. Consequently, financial institutions employ computer technology and mathematical models in several applications to support the enormous amount of people using their services. In the case of credit lending, banks and lenders must find ways to achieve net returns over and above what was invested in loan applicants to turn a profit.

Unlike humans, computers dispassionately execute commands based on vital, front-end human input. In other words, decisions are made based on data, rather than persuasion or emotions. Computational decision making is possible due to decades of thorough research, software programming, and system design. Although these systems are generally validated iteratively before deployment, human mistakes in the design can result in catastrophic failures down the road. For example, the quantitative models that banks used to test securitized mortgages failed to capture the possibility of a mass mortgage default such as the one that fueled the 2008 US financial crisis

[10]. Likewise, an air quality prediction model by BreezoMeter (used by Google during the California wildfires of 2018) predicted that the air quality was ideal for outdoor activities, at the same time residents discovered a layer of ash on their cars outside [2]. Rapid decision making allows financial institutions to capitalize on servicing people with low risk, faster than ever before. However, grave mistakes tend to impact the consumer more so than the institution. In using computer software to automate the approval of credit lending, it is important to be confident in the black-box algorithm. The output of the computerized credit lending system is only as good as the black-box model and its input. Therefore, the quality of data that is gathered from primary sources is of vital importance, as is the "cleaning and storage" of the data.

Several features work in tandem to calculate a credit score such as loan payments, loan length, duration of payments, and credit history. Although banks have an age-old system in place to calculate one's credit score, there are too many unknowns that can potentially mislead people rather than put them on a positive trajectory in terms of financial health. Because the internal details of the entire process are largely unknown, the problem requires that the model's explanation of why it makes certain predictions make sense to the consumer. When the output predicts or makes high-stakes decisions that can alter the course of someone's fate, the value of a black-box algorithm can exponentially increase if the prediction is actionable.

Explainable machine learning is important for credit lending and subsequently, financial institutions because two of the core tenants of any successful business are accountability and trust. Banks need to trust the AI system recommendations, and that they are accurate and were not developed using biased data.

1.3. Project Relevance to Machine Learning

This topic is exploring real-world applications of Machine Learning to help problems related to credit lending. Therefore, we will be working on several concepts and challenges reviewed in our lectures. We will be exploring anomaly detection for the iterative process that is building a useful model using the CRISP-DM framework. We will be emulating the steps involved in identifying and understanding a business problem, using data to fit a model, removing outliers, and feature engineering to improve the solutions currently used in the field. With an understanding of domain knowledge via research, we will be creating a model that is based on data and makes sense for human applications.

1.4. An Evaluation of Other Solutions

Machine learning applications are limited in their applicability to non-critical sectors since the ML predictions are difficult to understand and explain. For applications where a lack of explainability is not acceptable, machine learning algorithms have not been widely deployed. For instance, for some high-reliability applications such as nuclear plants, avionic computers, or self-driven cars, the controlling system needs to ascertain that the system will remain stable and in control no matter the new circumstances the system encounters. In the case of self-driven cars, some famous accidents have shown that self-driven Tesla cars have been involved in accidents where the self-driven ML algorithm didn’t stop the car and killed the passenger because the algorithms didn’t make the right decision for a new unseen situation.

Other factors can render the use of ML difficult in other industries. For instance, in the banking industry, laws and regulations such as the “Equal Credit Opportunity Act” prevent lenders from discriminating against customers based on their race [9]**,** and banks exposed themselves against heavy fines for not complying. More importantly, in addition to not discriminating, banks need to be able to explain and show evidence that their models are not discriminating in court during lawsuits. In a famous case, Well Fargo in 2018 has been suited by the cities of Sacramento and Philadelphia for discriminating against Latino and the African-American communities. Therefore, rendering an explainable model is critical for the company’s profit and protect against lawsuits in this industry. But what is the best solution if explainability is required, should an ML engineer use an explainable model such as logistic regression? Is there a way to use the latest ML model while inserting an explainability into the model?

1.5. Advantages of Our Solution

In the quest of producing explainable ML algorithms, Ribeiro and al. have introduced the novel idea of making any algorithm explainable locally. By rendering a complex algorithm comprehensible to humans in a localized area of the data set, the complexity of the model is greatly reduced and easier to understand. For improving the explainability, the novel approach relies on interpretable components defined as ‘d,’ where their value gives insights about the reasons the algorithm is making its predictions.

Why is this approach better? Without this technique, the second-best alternative is to rely on a model that is generally explainable, which is logistic regression. As well documented in the literature, ML and DL algorithms are able to extrapolate more information from the same dataset by incorporating really complex interaction terms into their models. Unfortunately, by introducing this complexity in the model, the model becomes less interpretable and explainable. In front of this dilemma, the novel approach allows keeping the added benefit of having a more complex model while keeping a localized explainability of the model.

1.6. Scope of Investigation

Model Accuracy and Explainability have tradeoffs. As you can see in the graph below which shows a family of ML algorithms starting from Linear Regression to Neural Networks. On the far left are algorithms like Neural Networks, Boosted Trees that are highly accurate but not very explainable. And on the far right, we've algorithms like Linear Regression and Logistic Regression that are highly explainable but not as accurate.

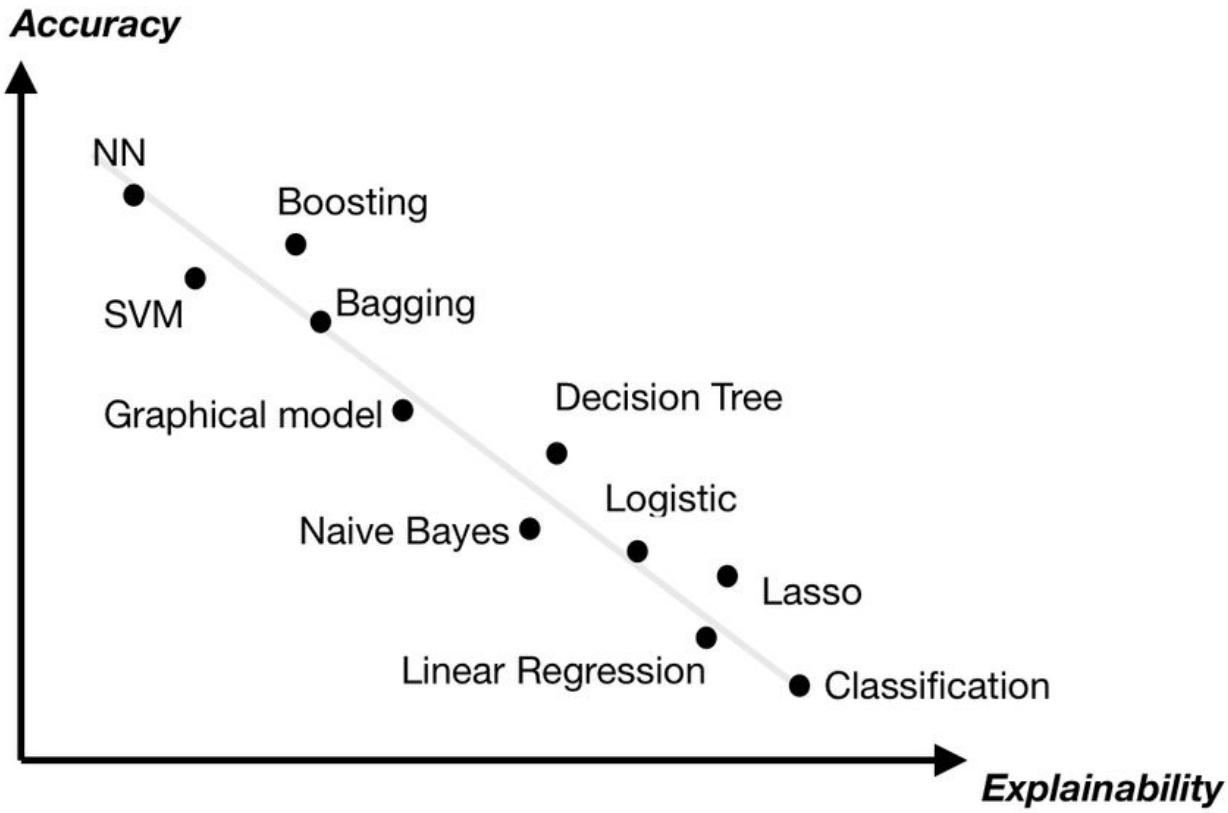


Figure 1: Trade-off between Model Accuracy and Explainability

Since Explainability is a must-have for Credit Lending because of regulations, most of the financial institutions are still using simple Linear Models. However, they suffer from low accuracy and this causes the business to potentially be making sub-optimal business decisions. For a very large bank, 1% point improvement in model accuracy could result in winning revenues in the order of 100s of millions of dollars. Therefore the need to train models that are both accurate and explainable is more prevalent there.

In this paper, we will investigate ML techniques that can optimize for both simultaneous accuracy as well as explainability thus can help Credit Lending not lose business value at the cost of transparency.

2.Theoretical Basis & Literature Review

2.1. Mathematical Basis

Our paper explains how an explainable machine learning algorithm could help in the credit lending industry. In the credit lending industry, successfully classifying customers according to the probability of being unable to pay or pay the loan is a very essential step. This is also where machine learning algorithms become handy in this functionality. Loan fraud detection is a process to predict, measure, detect the default rate of a loan. In addition, the lending company has to take into account that the cash flow utility from the opportunity gains and costs if the loan is incorrectly misclassified or is accepted. The company has to measure the risk they could have exposed then assign a customized interest rate according to the risk level they could have exposed. In order to correctly measure the risk the company exposes, the company mainly uses the credit risk models for evaluating personal credit loans application. By using the credit risk models, the company could estimate the expected loss by using the probability of default (PD), loss given default (LGD), exposure at default (EAD), recovery rate (RR=1-LGD). Probability of default (PD) is the probability of default of a customer over a year period. Exposure at default (EAD) is the amount of outstanding loan when it is considered as defaulted. Loss given default (LGD) is the ratio of loss on exposure due to the default of a customer. To sum up, the expected loss is calculated as follows:EL = PD × LGD × EAD = PD × (1 − RR) × EAD which summarizes the expected loss in dollar amount if the company takes on the loan applicant. In this paper, we mainly focus on how to predict the probability of the default

(PD) model and its calculation. In PD model calculation, companies use logistic regression, decision tree and Neural Network (Recurrent Neural Network) to estimate the probability of default. The applicants are evaluated by their 5C aspects: the character of the borrower, capacity of repayment, collateral assets packed with the loan, conditions of the market, and capital of the borrowers own and possess. First, the company prepares the data internally and externally. Then the data will be passed to the model, such as logistic regression to estimate the probability of default of that particular customer then an associated scorecard will be assigned. In the end, the score will calibrate with the rating which includes economic factors and regulatory factors in order to make the final decision on their loan application. Therefore, accurately predicting the probability of default of a loan customer is essential for the credit lending industry.

2.2. Theoretical Background

The importance of having an explainable and accurate probability of default rate model is very high in the credit lending industry. In the PD modeling process, the company may expose a model risk as well if the PD model is not good. “Essentially, all models are wrong, but some are useful” (George Box, 1987). Models are usually imperfect, and its imperfection could be dealt with by further investigation. However, most of the PD models are considered as black-box models. By definition, a black box is a device, system, or object which takes in input data and generates output, without any knowledge of its internal workings and that could lead to model risk. Model risk is basically a type of risk when a model is used to compute quantitative information and

the model fails or performs poorly which leads to adverse outcomes for the company. The risks could be the probability of default rate is incorrectly predicted sequentially the company wrongly classifies the customer and the associated interest rate, so that the company suffers a financial loss and opportunity lost. Without the ability to interpret the features in the model, the model could not be trusted. How do the company know the models are working well for them? In the paper “Why should I trust you?” It states the importance of being able to explain the methodology of generating the output in order to gain trust and reliability from the user. This also applies to the credit lending industry. Especially in the financial service industry, there are a few regulations about risk management, such as Basel II, Base III, and Solvency II. These regulations regulate the capital levels the banks should maintain. The new Basel Capital Account (Basel II) and Basel III require comprehensive disclosure by banks whose internal processes are under supervisory review and evaluation periodically. The financial institutions have to comply with Basel II to minimize their market risk by maintaining sufficient capital within companies. This is the main reason why it is so important to estimate the probability of default of the loan applicants accurately and to have the model explainable. If a bank is over-lending or taking a high-risk loan, the bank is exposing a high market risk then it fails to comply with the regulation. On top of that, the market or economic situation changes every day, the market risk is highly associated with the economic factor which implies the model might require some adjustments to adapt to the current circumstance. In order to be able to make adequate changes in the model to adapt to the current economic environment,

understanding the importance of the features and being able to interpret the features and the outcome are very crucial.

2.3. Related Research Solutions

Explainable AI or ML is a new and rapidly changing field. There are no doubt many other models to come as the field develops. In this section, we briefly review other techniques currently being used.

1. RETAIN (Reverse Time Attention) is another classification algorithm developed in 2016 for healthcare predictive modeling with interpretability. Because healthcare, much like credit lending, involves high-stakes decision making with sometimes fatal consequences, RETAIN was developed for the purpose of utilizing the accuracy from recurrent neural networks (RNN) and the interpretability from traditional models such as logistic regression. [11] It was developed in a study to predict heart failure at a rate comparable to other models using clinical trials and identifies the particular data that leads to the prediction [11]. In Choi et al.'s study, the model was able to take advantage of "14 million patients… over an 8 year period" [11]. The scalability and prediction accuracy were demonstrably promising without sacrificing the interpretability previously unique to traditional models.

2. The xAUC Metric is proposed by N. Kallus and A. Zhou for ranking disparities, or "a good ranking function that ranks positively labeled examples above negative examples"[1] in bipartite ranking tasks, or binary classifications tasks for the purpose of calculating prediction risk scores. 'xAUC' is built upon the popular performance/accuracy AUC metric used for binary classification. In the article, xAUC

is defined as the cross-area-under-the-curve, which measures the probability that a randomly chosen unit from the “positive group” is ranked higher than a randomly chosen unit from the “negative group” [1] High scores denote correct rankings. The magnitude of xAUC represents the “across-subgroup rank-accuracies”[1]. Cross-group ranking accuracies are carefully considered throughout this study, as there are a number of racial implications that the authors are trying to unveil in their fairness in criminal recidivism studies.

3. In the paper of “E.T.-RNN: Applying Deep Learning to Credit Loan Applications” and “Classification Model for Detecting and Managing Credit Loan Fraud Based on Individual-Level Utility Concept”, both papers describe how the companies apply Neural Network (Recurrent Neural Network) and logistic regression and Utility Sensitivity Classification to predict the probability of default of a customer. Both papers elaborate well on the methodology they apply to the credit application process. They also explain well the requirements from the model and the applications. For example, the input data and the processes of how the algorithm works. The papers clearly iterated how they dealt with the overfitting problem. However, both of the papers could not explain the importance of each feature in their models and the basis of how the model generates the output and the logic behind. Even though the author has created a baseline for his model to compare and to cross-check, it is not sufficient in the credit lending industry. This is where the Explainable Machine Learning algorithms are very important to deploy on top of those black-box models.

Regardless of the method, our surveillance of related research is clear: explainable machine learning has the potential to launch businesses into higher

efficiencies, with more transparency, accountability, accuracy, and reliability with carefully designed models. In turn, better decision-making will result in improved differentiation, trust between consumers and businesses, and ultimately help companies succeed. Our goal is to improve the explainable systems that will make it easier for humans to work with, thus leading to man-machine collaborations that will perform better than just computers or humans alone.

2.4. Advantages & Disadvantages of Current Research

In the section above, it is clear that several studies are underway to reap the benefits from both RNNs and traditional models that are highly interpretable. There are a number of efforts to increase the accountability and trust of AI systems in general. As discussed in this paper, Explainable Machine Learning is one such solution. The great advantage of these algorithms is that it potentially unveils hidden insights that humans are not necessarily capable of seeing. Furthermore, the speed by which computers can make or act on an application is far faster than humans ever could. Although rigorous scientific research and testing underpin each decision, the problem with this process is that it is not transparent. Although black boxes can be a powerful tool, one of the biggest drawbacks is that it cannot tell you why it made a certain decision or recommendation. Current AI systems cannot explain how it comes to a particular decision, even if the degree of confidence is over 90%. This can be permissible for harmless applications such as computational advertisements or Google searches, but for high-stakes decisions, the consequences of making a bad prediction can be life-altering. Therefore, the adoption of black-box algorithms is much

slower to develop in regulated, high-stakes decision-making fields. Explainable machine learning algorithms can help with progress as it is necessary to service millions of people, whether it is in credit lending or medical decisions.

On the contrary, although there are hidden layers that may be gleaned from a black-box algorithm, there are a number of disadvantages to using them. Firstly, there is a potential loss of competitive edge for companies to incorporate explainable machine learning. An alternative definition of a black box is that it is a proprietary machine-learning algorithm that is sold to a customer and serviced periodically. The idea of patenting a piece of technology still applies to machine learning algorithms. In this case, the customer only obtains the output and has no way of performing any troubleshooting tasks. To support a second algorithm that explains how the first algorithm works defeats the purpose of commercial development. Secondly, this field requires strong domain knowledge. It goes without saying that the road to the final outcome can be extremely complex. Some concepts are extremely difficult to explain even when communicating in the same language. For example, kindergarteners are unlikely to grasp astrophysics even if they are taught in their native language. Because some topics are inherently difficult or impossible to explain, it follows that there are a number of tradeoffs that one has to make in order to achieve explainability. Alternatively, there are a select few who hold the opposition to the use of Explainable Machine Learning by scientists that find black box systems to be counterproductive in high-stakes decision making. A strong voice in this field is Cynthia Rudin from Duke University, who in her paper reviews the major pitfalls of explainable machine learning [2] and advocates for Interpretable Machine Learning instead. Although the

possibilities to glean from black-box algorithms are vast, they are next to useless because the ‘whys’ of classification is the first and foremost priority. Traditional CART (classification and regression trees) machine learning algorithms are accurate enough for use and the interpretability, or human understanding of why a particular classification decision was made, is the most important piece of the algorithm. Rudin goes further to say that no problem she has ever encountered required a black-box algorithm. Even though the design stage of an interpretable machine learning algorithm is much more complicated and costly, the value of understanding the model and knowing exactly why a prediction is made is non-negotiable for high-stakes decisions.

Nevertheless, we would like to continue exploring the capacity of explainable machine learning techniques for the purpose of this project. The cons of using them are important to keep in mind as we evaluate the final performance of our models.

2.5. Our Solution

One way to explain machine learning models is to examine feature importance: for a given prediction, how important is each input feature value to that prediction value? We also call this the **prediction impact** of a feature.

For simpler models, explainability can often be provided by the model itself. This is especially true for tree-based models, where the algorithms can assess the importance of each feature, given a set of features. The model changes a feature in the tree by another random feature and calculates the impact of the removed feature

on the model quality metric. If changing the feature decreases the model quality on average, the more that feature is considered important.

For example, it is possible that the credit amount, age of the borrower be the most important features for this model. And features like purpose, housing rent are not that important. We can use feature importance for feature selection so we can prune unimportant variables and keep the model simpler but yet accurate.

Model Agnostic Explainability

Model Explainability is a hot research area these days. And the number of methods that have been created to explain models has exploded. The following diagram shows a number of methods.

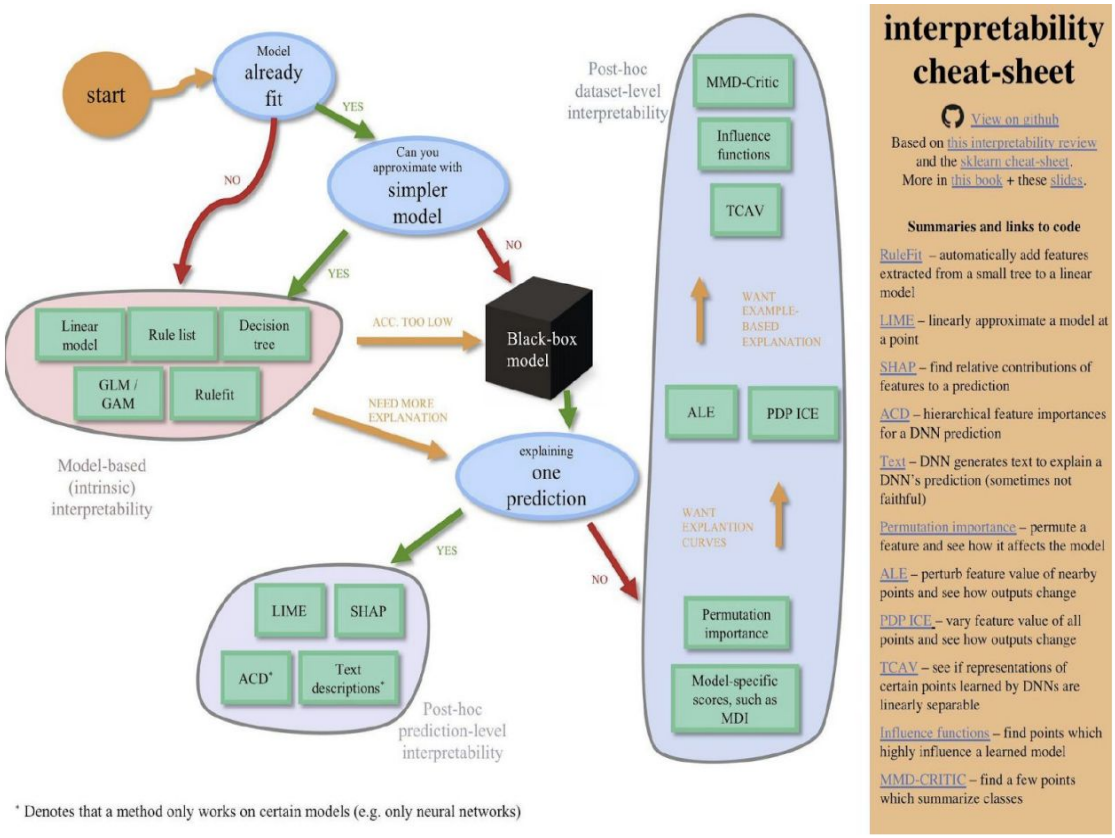


Figure 2: Model Explainability Techniques

Out of these, we will present 3 of the most important approaches.

- Partial Dependence Plots
- LIME explanations
- Shapley values

Partial Dependence Plots

A PDP is useful when we're interested in measuring the sensitivity of model output to variation of a variable. Keeping everything constant, PDP shows the relationship of the feature to the model output.

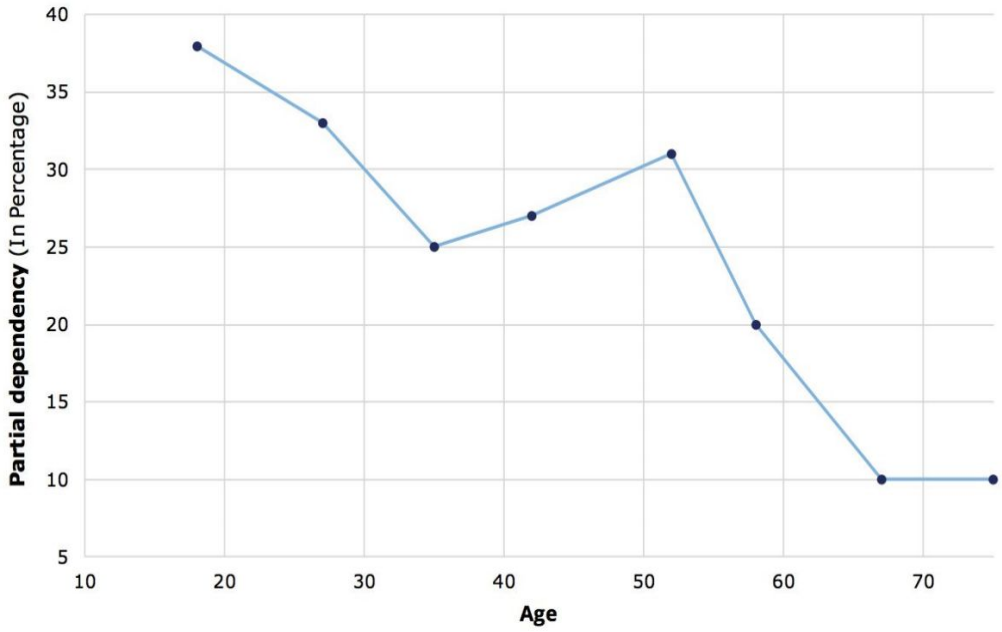


Figure 3: PDP Example showing the relationship between Age and Credit Default

For example, the PDP of age on the loan default rate can be shown like this line plot. The graph shows that the default rate decreases with the age of the loanee, which means young adults are more likely to default on the loan vs the older people. It

also shows the non-linear nature of the relationship between Age and Default Probability.

LIME

While PDP provides a global and local view of the model, it is difficult to show the impact of more than a couple of features on the prediction. Therefore we need new explanations that can work for a large number of features. LIME [4] is such a technique that can explain the model locally. The basic idea behind LIME is to create a surrogate model to explain a given data point. For example, when we apply LIME for the credit lending model and want to explain a single loan default prediction, LIME generates a new dataset permuting the features of the given data point and then builds a simpler surrogate model (linear regression, decision tree, etc) on this dataset. For example, to explain why a loanee is classified to be a default, we want to understand the most important features behind this. One of the problems with LIME is that the explanations can change based on the sampling techniques changed to construct the surrogate model. Also, different surrogate models could generate different explanations.

SHAP

To understand SHAP, we first need to understand Shapley values. The Shapley value [5], proposed by Lloyd Shapley in 1953, is a way to distribute the total gains of a collaborative game to a coalition of cooperating players. It is the only distribution with certain desirable properties (fully listed on Wikipedia), including:

- **linearity**: If two games are combined, then the total gains should correspond to the gains derived from a linear combination of the gains of each game
- **efficiency**: The sum of the values of all players equals the value of the grand coalition so that all the gain is distributed among the players.

In the case of ML, we formulate a game for the prediction at each instance. We consider the “total gains” to be the prediction value for that instance, and the “players” are the model features of that instance. The collaborative game is all of the model features cooperating to form a prediction value. In this case, the efficiency property says: the feature attributions should sum to the prediction value. The attributions can be negative or positive since a feature can lower or raise a predicted value.

SHAP [3] (SHapely Additive exPlanations) is an algorithm to build a linear explanation model of feature importance for a given prediction by computing Shapley sampling values. These approximate the effect of removing a feature under various combinations of presence or absence of the other features. The model's behavior on input with certain features absent is simulated by integration over samples from the marginal distribution of those features drawn from the training set.

For a credit lending model measuring the default probability, the objective will be to figure out which features contribute positively or negatively to predicting that a loanee will default on the loan in the future. In order to do that we calculate the marginal impact of each feature on the default probability and take an average on every possible feature combination.

$$\Phi i \;=\; \sum_{s} \frac{|S|!\;(|F|-|S|-1)!}{|F|!}\left[\, f(Xs \;\cup \{i\}) \;-\; f(Xs) \,\right]$$

Where **S** is a vector with a subset of features, **F** the full number of features, f() the output of a model, and **{i}** the added feature.

Compared to LIME, the Shapley value is the unique method that satisfies the following desirable axioms, which motivates its use in Machine Learning.

1. **Completeness**: Shapley values sum up to the difference between the target output and the output of the baseline.
2. **Symmetry**: For two features i, j, and S, where S is any subset of features not including i, j, when {S, i} = {S, j} → i and j should have the same attribution value.
3. **Dummy**: For one feature i and S, where S is any subset of participants not including i, when {S, i} = {S} → i should get zero attribution.
4. **Additivity**: For two model functions, the Shapley value of the sum of the two model functions should be the sum of the Shapley values of each of the model functions. For example, if f(x) and g(x) are two model functions, then SHAP(f(x)+g(x)) = SHAP(f(x)) + SHAP(g(x)).

SHAP gives both a global picture and an effect on a specific data point. This is great because we can study both the global importance of features as well as feature impact for a given loan application.

For example, when we take a look at a single loan application and study with SHAP, we can see that the person's profile being unskilled and non-resident, not having a lot

of money in the bank account all put the profile in a risky zone thus increasing the likelihood of default.

SHAP also provides a global feature importance plot that can tell us that young adults having smaller loan durations could be riskier and education loans are less risky compared to other loan types. The risk factors are also quantified as Shapley value impact scores.

While this is great, one of the big disadvantages of SHAP is that the computational complexity of the algorithm scales exponentially to the number of features. So this may not be suitable for inputs with a large number of features.

Our Approach

In our approach, we will build a gradient boosted tree for predicting credit risk and then use SHAP to explain the model both globally as well as locally. Boosted trees are proven to be the best algorithms in-terms of accuracy for structured datasets. Since credit risk datasets are generally structured in nature, this will be a good approach. And SHAP is an axiomatically proven explainability technique that works very well on structured data with a limited number of features.

We will combine both the algorithms to create a credit risk model that is not only accurate but also highly explainable.

2.6. Differences & Advantages

Our solution is different in the sense that most credit risk models are simpler linear models. We will use an advanced algorithm like Gradient Boosted Trees to train a credit risk model and then use an advanced explainability technique like SHAP to

explain it. Therefore we combine the best of the accuracy and explainability techniques, which would make it better than what banking institutions use today to measure credit risk for loans.

SHAP allows us to intervene with the causality of the models by asking What-If questions. Consider a credit risk model with features including the applicant's income and zip code. Say the model internally only relies on the zip code (i.e., it redlines [6] applicants). Explanations might reveal that the applicant's income, by virtue of being correlated to zip code, is as predictive of the model's output. This may mislead us to explain the model's output in terms of the applicant's income.

To learn more, we can intervene in features. One counterfactual changing zip code but not income will reveal that zip code causes the model's prediction to change. A second counterfactual that changes income but not zip code will reveal that income does not. These two together will allow us to conclude that zip code is causal to the model's prediction, and income is not.

3.Hypothesis

In this study, the first hypothesis in our case is that the explainable machine learning algorithms could be applied to explain or interpret the probability of default in the credit lending industry. We assume the LIME and SHAP explainable machine learning could be able to elaborate the prediction logic behind. In our case, in the industry, the most common algorithms for PD prediction are logistic regression and neural networks. In the paper "Why should I trust you?", the author has successfully deployed LIME and SHAP to explain the neural network model on predicting images.

Therefore, we assume it could work well in our case to predict the default rate. However, logistic regression is already an explainable machine learning algorithm. Therefore, there is no need to deploy the explainable machine learning algorithms on top of the logistic regression.

The second assumption is that the training data is not changing over time and also the training data are in the same format as testing data. If data has changed, the explainable machine learning algorithm should be able to pick up the changes due to the ability to interpret or to explain. However, this study mainly focuses on the model itself but not the data quality. Therefore, we should assume the data quality is well enough to perform the outcome we desire.

The third assumption is the model has to be local linear to be explainable. The LIME machine learning algorithm generates data around the desired points then uses those dumpy data to formulate a linear regression line in order to form an explanation of the outcome. This “technique that explains the predictions of any classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction.” [4] We assume that the dataset could be explained by a linear regression line even though in the paper of “Why should I trust you” states that in order to formulate the linear explanation regression line, there might be some features that would be eliminated. Therefore, in our case, we assume the linear regression line of our explainable machine learning model could be able to explain the prediction logic even though there might be some features eliminated in the process.

The fourth hypothesis is that interpretability is as important as accuracy for a model. This hypothesis is also our goal that interpretability is very important to the

credit lending market. As we discussed in the previous paragraph, the interpretability or the explainability for the calculation of the probability of default is very essential for the credit lending market for reasons such as fulfilling the regulation requirement, monitoring the model risk, and availability to adapting to the economical changes if necessary. These reasons require companies to change or to interpret the default rate from time to time. In order to do so, the full understanding of the models or the prediction is inevitable. Therefore, we assume this study is one of the solutions to the problem.

4.Methodology

4.1. Data Collection

For the experimentation's sake, we've used Kaggle to collect credit lending datasets. In particular, there is an open-source dataset from Lending Club that we will use for training and explaining models.

This is the dataset that has been constructed from the Lending Club dataset available on Kaggle. It contains records from over 470,000 peer-to-peer loans issued through the Lending Club platform between June 2012 and August 2015. All loans had a 3-year term, and the outcome of each loan is known (i.e. we know whether it was fully paid or charged off as a loss). The loans issued from April 2015 through August 2015 are designated as the test set.

4.2. Problem Solution

In order to solve this problem we will follow the 5 step process:

1. Clean the Kaggle dataset
2. Build features that are important
3. Train models from Logistic Regression to GBDT
4. Measure the accuracy of the models
5. Use SHAP to explain each of these models

To train the models we will use the Scikit-Learn open-source ML package to train Logistic Regression and Gradient Boosted Tree models. For the Boosted tree models, we will use LightGBM library. And we will use SHAP to help us with feature selection so we create the best possible set of features to train. After that, we will train a few different ML models and measure the AUC(Area Under the Curve), Precision and Recall scores. All the experiments will be conducted in Python and Jupyter Notebook will be used for visualization.

4.3. Output Generation

We will generate outputs in Jupyter Notebook plots to visualize global and local explanations for a given model. For example, below is a sample feature importance plot of a light GBM model created for credit risk.

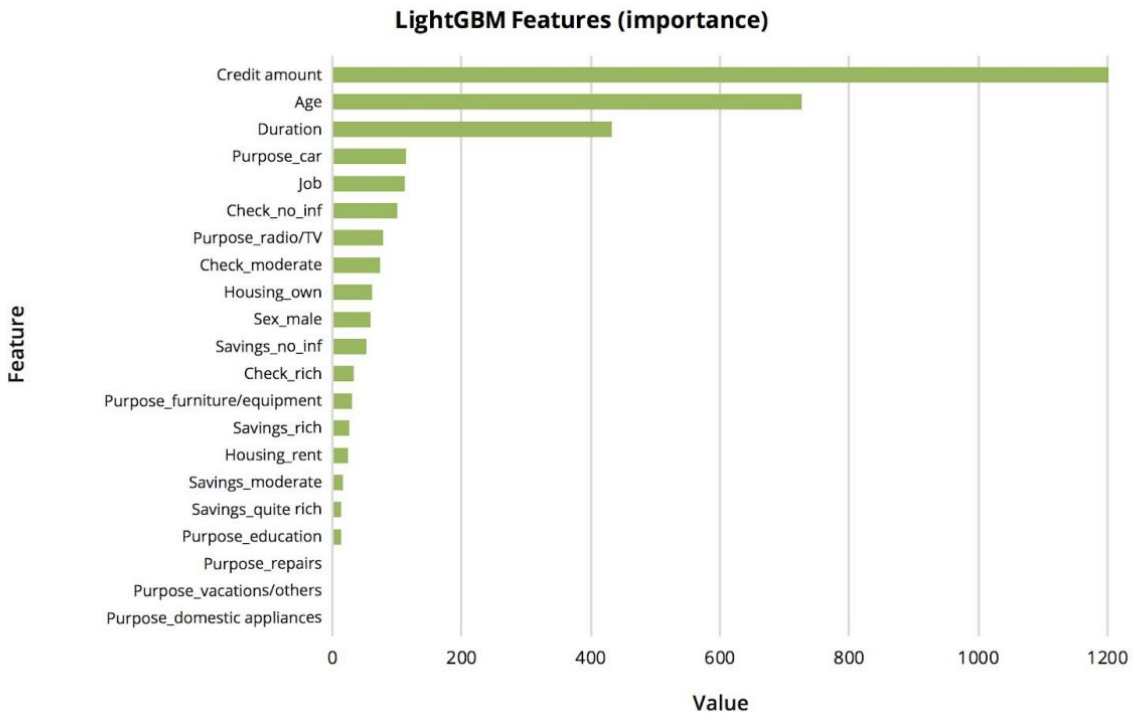


Figure 4: Example of Feature Importance for a Gradient Boosted Tree Model

Similarly, SHAP provides the ability to look into individual prediction explanations like below. For example, when we take a look at a single loan application and study with SHAP, we can see that the person's profile being unskilled and non-resident, not having a lot of money in the bank account all put the profile in a risky zone thus increasing the likelihood of default.

Figure 5: Sample SHAP explanation for a single observation

4.4. Hypothesis Testing

To test our hypothesis that it is possible to create a highly accurate model that is explainable, we will take the best models in terms of AUC score (or other such model quality metrics) and look at their explanations. We will manually inspect the explanations of the model and judge if they make sense. Since evaluating explanations is still a subjective exercise, we will wear the hats of a Credit Lending Officer and evaluate the explanation quality ourselves.

For example, we will inspect if the SHAP summary plot of a given model is picking up the most important variables as part of explanations. We can see if the top SHAP variables are highly correlated with the target or not. Intuitively it would make sense for a high impact SHAP variable to have a good correlation coefficient with the target.

5.Software Implementation

5.1. Source Code

There are three key parts to this project. We first perform exploratory data analysis in a Jupyter Notebook in Python 3. The notebook then diverges into 3 separate workbooks. The first notebook contains the logistic regression baseline model using scikit-learn. The second notebook contains the GBDT model with LIME explainable model. The third notebook contains the GBDT model with the SHAP explainable model.

Here is a snippet to train logistic regression and boosted tree models.

```
%%time
# define logistic regression
logistic_regression = sklearn.linear_model.LogisticRegression(
    C=int(1e5), solver='lbfgs', random_state=0, max_iter=500)

# define boosted trees
boosted_trees = lgb.LGBMClassifier(
    boosting_type='gbdt', class_weight=None,
    colsample_bytree=0.63157894736842102, importance_type='split',
    learning_rate=0.03, max_depth=-1, min_child_samples=668,
    min_child_weight=0.001, min_split_gain=0.0, n_estimators=681,
    n_jobs=-1, num_leaves=28, objective=None, random_state=None,
    reg_alpha=0.0, reg_lambda=0.0, silent=True,
    subsample=0.55789473684210522, subsample_for_bin=200000,
    subsample_freq=5
)

# fit the models
logistic_regression.fit(train_features_processed, train_target)
boosted_trees.fit(train_features_processed, train_target)
```

```
CPU times: user 14.2 s, sys: 298 ms, total: 14.5 s
Wall time: 2.12 s
```

And here is the code snippet to calculate SHAP values from the boosted tree model.

```
# set up the explainer, and adjust the additive attribution intercept to be
# the average over a larger set of predictions
boosted_trees_explainer = shap.KernelExplainer(lambda x: boosted_trees.predict_proba(x)[:, 1],
                                               background_data)
boosted_trees_intercept = predict(boosted_trees, test_input_sample).mean()
boosted_trees_explainer.fnull = np.array([boosted_trees_intercept])
boosted_trees_explainer.expected_value = boosted_trees_explainer.fnull

def get_shap_values(explainer, inputs, n_features=50, n_samples=int(5e3)):
    phi_ndarray = explainer.shap_values(inputs,
                                        l1_reg=f'num_features({n_features})',
                                        nsamples=n_samples)[0]
    return sorted_by_abs(pd.Series(phi_ndarray, index=inputs.columns), ascending=False)
```

```
shap_values = get_shap_values(boosted_trees_explainer, model_inputs.to_frame().T)
```

```
global_shap_values = shap.TreeExplainer(boosted_trees).shap_values(test_features)
shap.summary_plot(global_shap_values, test_features)
```

5.2. Design Document & Flowchart

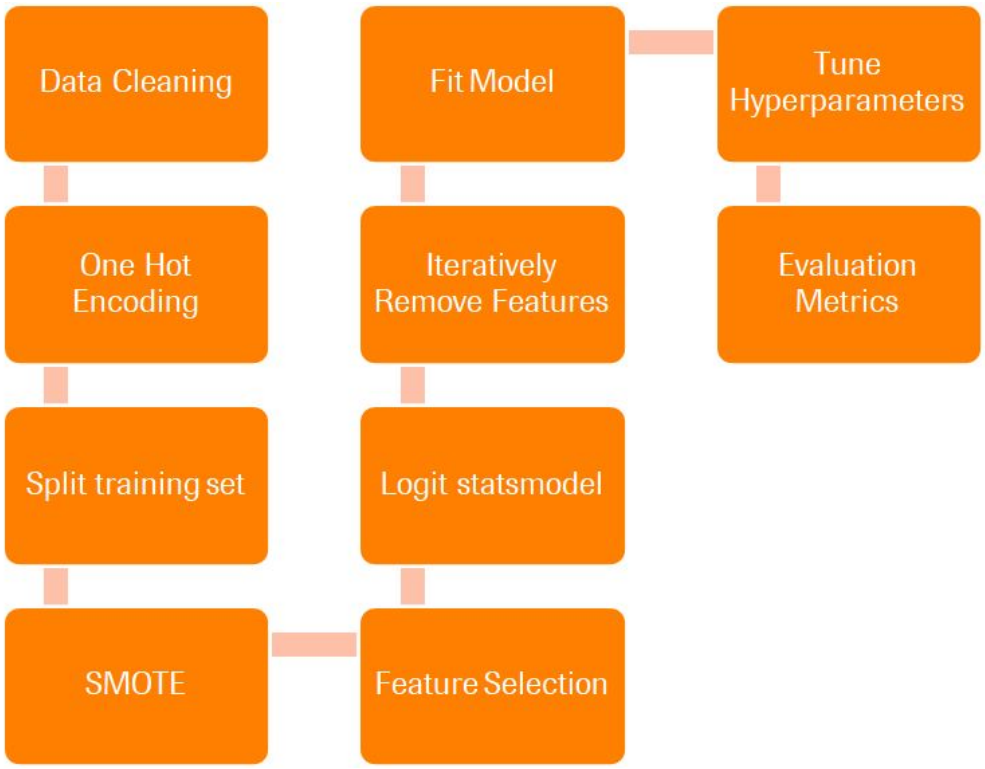


Figure 6a: Flowchart for Part 1: Logistic Regression Baseline Model

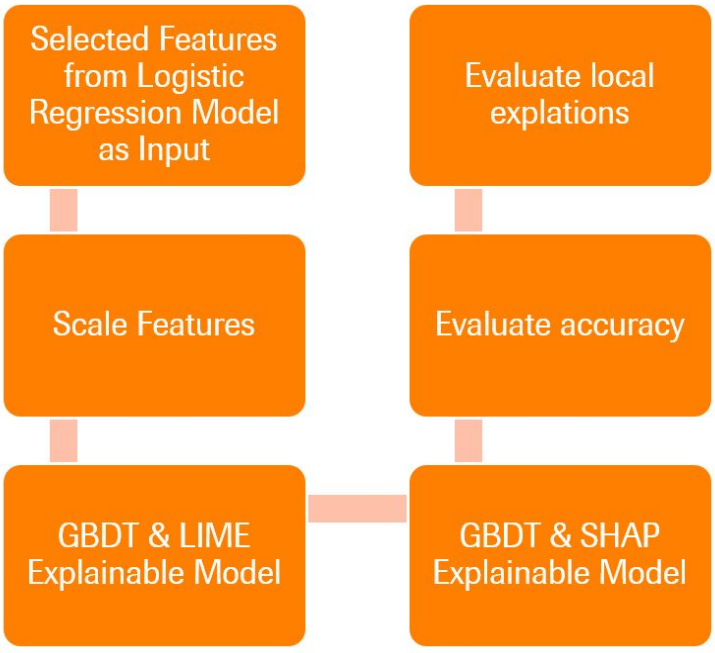


Figure 6b: Flowchart for Part 2 and 3: Problem Solution of GBDT Training and Model Evaluations

6.Data Analysis & Discussion

6.1. Output Generation & Output Analysis

Logistic Regression

Logistic Regression is a ML classification algorithm that predicts the probability of a categorical dependent variable, or the decision outcome. Due to its simplicity and explainability, it is often used as a baseline model for complex model comparisons for performance gain. The dataset contains 67 columns with both categorical and continuous independent variables and 1 column (loan_status) as the binary classification outcome. In this dataset, the dependent variable is binary--'Fully Paid' is encoded as 1 (the desired outcome), and 'Charged Off' is encoded as 0. In other words, 'Charged Off' is a failure to pay off the loan or default. We begin with a baseline logistic regression model that predicts the probability P(Y=1) as a function of X, or the independent variables.

We perform a 65/35 training/test set split. We noticed that the dataset was imbalanced in terms of classifiers. In other words, the classification categories were not equally represented. Our classes were imbalanced with a ratio of fully paid to charged off is 80:20. See Figure 7 for a representation of the entire dataset. We choose to under-sample the majority class to increase the sensitivity of the classifier to the minority class and achieve overall better classifier performance. To handle the class imbalance, we use a technique called SMOTE (Synthetic Minority Oversampling Technique) [12]. This works by creating synthetic samples from the minor class (Charged Off) by randomly choosing a k-nearest neighbor and randomly tweaking it

[12]. Implementing SMOTE allowed us to balance the proportion of outputs in the oversampled data for the training set. With SMOTE, we observed a 50:50 ratio of classification outcomes. See Figure 8 for the result of the oversampling of the minor class in the training set.

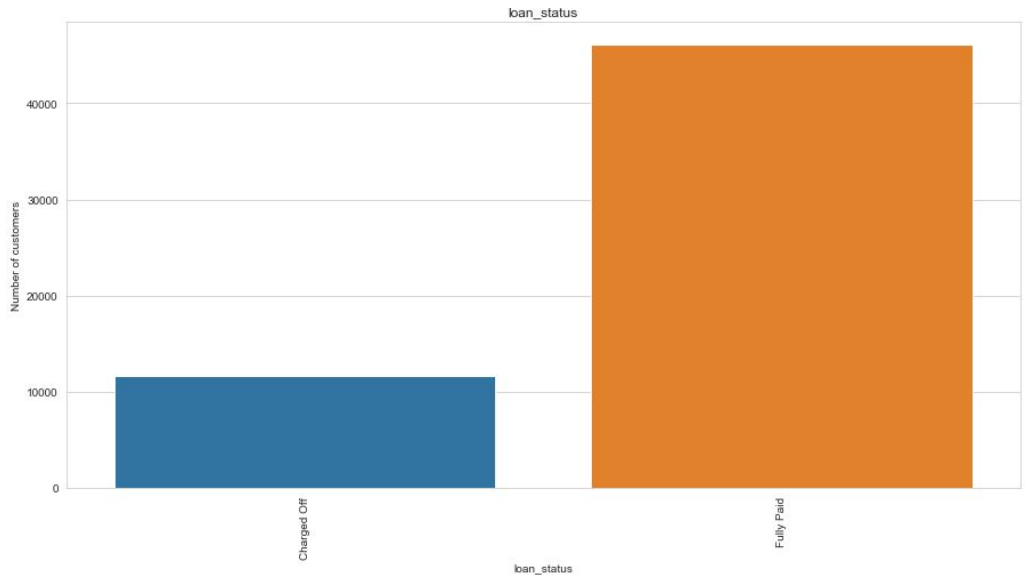


Figure 7: Class comparison for dependent variable loan_status

```
from imblearn.over_sampling import SMOTE
os = SMOTE(random_state=0)

os_data_X, os_data_y = os.fit_sample(X_train, y_train)
os_data_X = pd.DataFrame(data=os_data_X,columns=columns)
os_data_y= pd.DataFrame(data=os_data_y,columns=['loan_status'])
```

```
print("Number of records in oversampled data is ",len(os_data_X))
print("Number of 'Charged Off' in oversampled data",len(os_data_y[os_data_y[
print("Number of 'Fully Paid' in oversampled data",len(os_data_y[os_data_y['
print("Proportion of default data in oversampled data is ",len(os_data_y[os_
print("Proportion of fully paid off data in oversampled data is ",len(os_dat
```

```
Number of records in oversampled data is  60038
Number of 'Charged Off' in oversampled data 30019
Number of 'Fully Paid' in oversampled data 30019
Proportion of default data in oversampled data is  0.5
Proportion of fully paid off data in oversampled data is  0.5
```

Figure 8: SMOTE oversampling of the training set

Using SMOTE, we have a robust representation of both classes to observe feature importance for prediction.

```
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot

logreg = LogisticRegression(max_iter = 10000)

## Return logistic regression coefficient or feature importance score iteratively

logreg.fit(os_data_X, os_data_y.values.ravel())
importance = logreg.coef_[0]

for i,v in enumerate(importance):
    print('Feature %0d, Score: %.5f' % (i, v))

## Graph results
plt.figure(figsize=(15,7.5))
plt.bar([x for x in range(len(importance))], importance)
plt.grid(b=None)
plt.show()
```

```
Feature 0, Score: -0.12106
Feature 1, Score: 0.00039
Feature 2, Score: -0.00000
Feature 3, Score: -0.00043
Feature 4, Score: 0.00003
Feature 5, Score: 0.01025
Feature 6, Score: 0.01033
Feature 7, Score: 0.00004
Feature 8, Score: 0.00014
Feature 9, Score: 0.00002
Feature 10, Score: 0.00004
Feature 11, Score: -0.00092
Feature 12, Score: 0.00027
Feature 13, Score: 0.03343
Feature 14, Score: 0.08773
Feature 15, Score: -0.03303
Feature 16, Score: -0.00066
Feature 17, Score: 0.00000
Feature 18, Score: 0.00000
```

Figure 9: Logistic Regression Coefficient for each Feature, with corresponding graph

Next, we explore the feature importance in logistic regression. We use 2 methods to compare feature importance. The first is by ranking the logistic regression coefficient or feature importance score in the decision function [15]. A survey of the features shows that, of the 164 amount of features in the dataset, only a small subsample is significant for predicting the final outcome. The superimposed graph in Figure 9 visually represents the proportion of features that have significant score importance. The output suggests that of the feature list the subset containing the following 15 features:

```
## Comparison of two methods indicate that of 165 features, only a few are important. Drop the rest from the dataset.

selected_cols = ['loan_amnt', 'annual_inc', 'dti', 'delinq_2yrs','acc_open_past_24mths', 'fico_range_low',
                 'fico_range_high', 'pub_rec', 'revol_util', 'total_acc', 'total_pymnt', 'total_rec_prncp',
                 'total_rec_int', 'total_rec_late_fee', 'collections_12_mths_ex_med', 'tot_cur_bal', 'total_rev_hi_lim',
                 'bc_util', 'num_sats', 'num_tl_30dpd', 'num_tl_90g_dpd_24m', 'pct_tl_nvr_dlq', 'pub_rec_bankruptcies']
```

For comparison, we use RFE (Recursive Feature Engineering) to separately select important features. Recursive Feature Elimination from sklearn.feature_selection repeatedly constructs a model and removes the best or worst performing feature to assign weights to the remaining features in smaller and smaller sets [14]. Similar to the previous method, the RFE estimator is trained on the initial set of features through the coef_ value. RFE takes it a step further by pruning the least important features from the current set to return the user-selected number of features. Because this process takes a long time, we use the logistic regression solver ‘SAGA,’ or Stochastic Average Gradient. SAGA is a variant that optimizes the sum of a finite number of smooth convex functions [13]. SAGA is recommended for large datasets when both the number of samples and the number of features are large [13]. Therefore, we choose SAGA for feature engineering because we have a large dataset and a large number of features.

```python
print(rfe.support_)
print(rfe.ranking_)
print(rfe.n_features_)
print(rfe.estimator_)
```

```
[ True False  True False False  True  True False False False  True False
 False  True  True  True False False  True  True False False  True
  True  True False False  True False False False False False False False
 False False False False False False False False False False False False
  True False False  True  True  True  True  True False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False]
[  1   5   1   4  43   1   1  80  10  32   1   2   7   1   1   1   6  99
 115   1   1   3  15   1   1   1 103  74   1  16  21  22  14  23  27  18
  13  19  17  44  11   8  12   9 134 121  66  20   1  35  77   1   1   1
   1   1  54  56  41  36  49  45  46  47  48  50  55  62  68  71  84  91
  97 107 122 116 127 129 135 131 132 143 137 136 138 141 144 142 139 140
  25  33  58 123  86 101 111  60 133 118 108 145  98 114  81  31  82  96
 124 125  51  67 110  63  87 100 104 102  83  85  73  88  93 112 120  78
 109  70 113  94  57  72 105  90  69 119  95 130  92  42 106  76 126  79
  89 117 128  38  75  26  30  34  28  29  24  37  39  52  53  64  61  59
  65  40]
20
LogisticRegression(C=1000, max_iter=10000, solver='saga', tol=0.07)
```

Figure 10: RFE output

The default number of features RFE selects is 20. Overall, this process demonstrates that the two feature importance lists are agreeable, and that only continuous features are involved in this classification problem. We select the feature columns described above and then use the Logit Model summary feature from statsmodels to view the statistical significance of each feature shown in the summary of the results. In Figure 11, first, confirm that the optimization terminated successfully and that convergence was successful. Then, we look for the p-value of each feature to determine whether or not it should remain in the model. Based on the p-value, we iteratively remove features one by one until the p-values for each feature are low. Although in the final run, **annual_inc** and **dti** p-values are above 0.05, we permit them in the model because we are interested in keeping any important feature for the baseline. We end up with a final feature list of: **loan_amnt, annual_inc, fico_range_low, total_pymnt, total_rec_int, dti, and acc_open_past_24mnths.**

```
## Implementing the Model

logit_model=sm.Logit(y, X)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.008164
         Iterations 16
                         Results: Logit
==================================================================
Model:                  Logit              Pseudo R-squared:  0.988
Dependent Variable:     loan_status        AIC:               994.2695
Date:                   2020-05-16 18:49   BIC:               1057.2886
No. Observations:       60038              Log-Likelihood:    -490.13
Df Model:               6                  LL-Null:           -41615.
Df Residuals:           60031              LLR p-value:       0.0000
Converged:              1.0000             Scale:             1.0000
No. Iterations:         16.0000
------------------------------------------------------------------
                      Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
------------------------------------------------------------------
loan_amnt            -0.0136   0.0007 -20.7352 0.0000 -0.0148 -0.0123
annual_inc           -0.0000   0.0000  -1.7399 0.0819 -0.0000  0.0000
fico_range_low        0.0077   0.0006  13.8710 0.0000  0.0066  0.0088
total_pymnt           0.0137   0.0006  21.2559 0.0000  0.0124  0.0150
total_rec_int        -0.0142   0.0006 -21.9958 0.0000 -0.0154 -0.0129
dti                  -0.0220   0.0135  -1.6290 0.1033 -0.0484  0.0045
acc_open_past_24mths  0.1438   0.0435   3.3044 0.0010  0.0585  0.2291
==================================================================
```

Figure 11: Logit Summary of final feature selection list

Next, we fit the data to the logistic regression model. We arbitrarily choose hyperparameters that work for the initial run-through.

```
LogisticRegression(C=100000, max_iter=10000, random_state=42, solver='saga',
                   tol=0.007)
```

We predict the test set results and calculate the accuracy or AUC metric. The accuracy of this logistic regression classifier on the test set is 0.87. We then look at the confusion matrix to understand the correct and incorrect predictions. Based on this iteration, we have 18,345 correct predictions (8247 true negatives, 10098 true

positives), and 2,669 incorrect predictions (375 false negatives, 2294 false positives).

```
[[ 8247  2294]
 [  375 10098]]
```

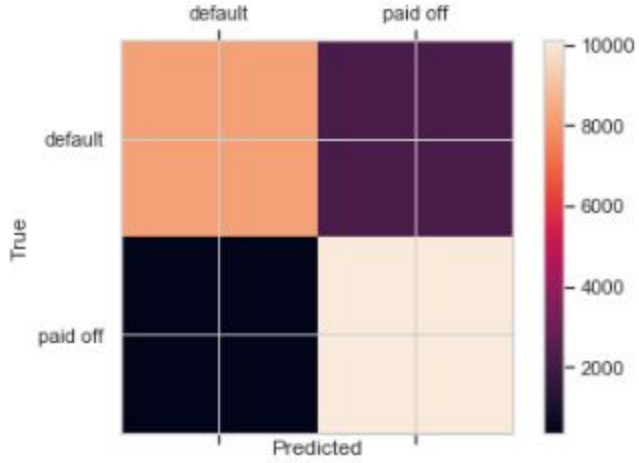


Figure 12: Confusion Matrix of Over-Sampled Training/Test Set

We then determine the precision (ratio of true positives to total positives), the recall (ratio of true positives to the total of true positives and false negatives), and the F-beta score which is the weighted mean of the two values.

```
3 from sklearn.metrics import classification_report
4 print(classification_report(y_os_test, y_os_pred))
```

	precision	recall	f1-score	support
0	0.96	0.78	0.86	10541
1	0.81	0.96	0.88	10473
accuracy			0.87	21014
macro avg	0.89	0.87	0.87	21014
weighted avg	0.89	0.87	0.87	21014

Figure 13: Classification Report of Over-Sampled Training/Test Set

From the table, we can infer that the ability of the classifier to not label a sample as paid off if it is actually defaulted is 89%. The ability of the classifier to find all paid off customers was 87%. The F-beta score is 87%.

We then create a ROC curve for the model. Because our classifier is closer to the top left corner than the ‘random’ classifier (dotted red line), we can infer that the features selected are relatively decent at predicting the tendency of a loan customer.

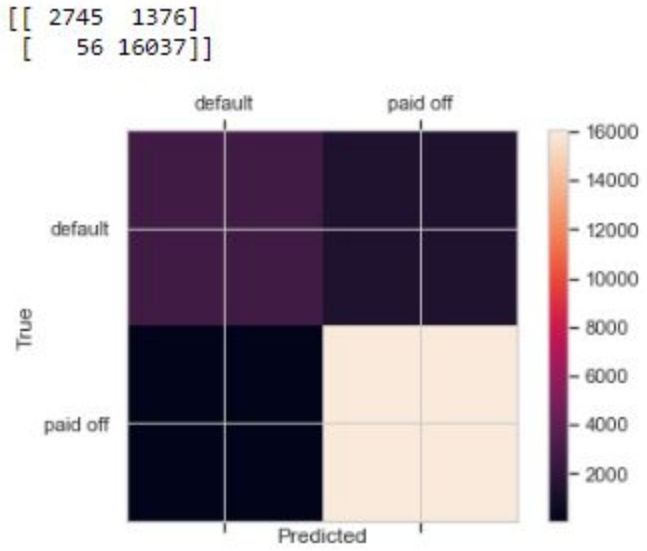
Figure 14: ROC Curve Over-Sampled Training/Test Set Logistic Regression

To improve the model, we use GridSearchCV from sklearn to determine the best hyperparameters. Originally, we used arbitrary hyperparameters that worked with the dataset. GridSearchCV uses a parameter grid with variations in terms of regularization, size of the penalty, and type of solver used [16]. We fit 5 folds for 100 candidates, totaling 500 fits to determine that the best C value to use is 78, and solver liblinear, and L1 regularization.

```
LogisticRegression(C=78, max_iter=10000, penalty='l1', random_state=42,
                   solver='liblinear', tol=0.07)
```

Repeating the model fitting steps above, we find that this greatly increases the accuracy of our model, with a new accuracy of the logistic regression classifier on the test set of 93% on the original dataset when the classes are imbalanced. The AUC

decreases to 0.83, however, the precision, recall, and f-beta scores all increase for this iteration.



	precision	recall	f1-score	support
0	0.98	0.67	0.79	4121
1	0.92	1.00	0.96	16093
accuracy			0.93	20214
macro avg	0.95	0.83	0.88	20214
weighted avg	0.93	0.93	0.92	20214

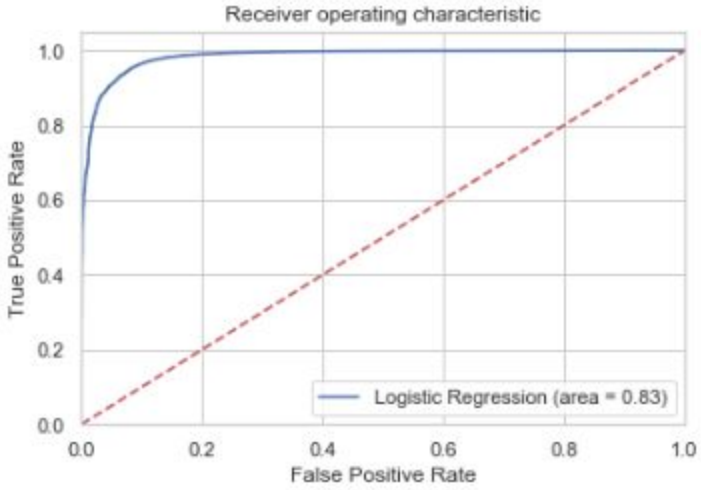


Figure 15: Confusion Matrix, Classification Report, and ROC curve of Logistic Regression Training/Test Set

The conclusion for the logistic model is that it has high accuracy for predicting default based on the loan applicant data. However, if we consider predicting risk of new applicants, there will not be any available data for the initial status of the loan. For example, total_pymnt and total_rec_int will not exist for those applicants. For the remainder of this paper, we explore the feasibility of GBDT models while excluding those two features to predict the loan status outcome, and use explainable models to evaluate its interpretability.

Gradient Boosted Decision Tree & LIME

Gradient Boost Tree is implied in this dataset to conduct the prediction. It is a common machine learning algorithm to solve regression and classification problems. It is one of the ensemble models which uses the weak prediction models to generate a strong prediction model by combining the prediction of weak prediction outcomes. The ensemble method it uses is the boosting method. Boosting method is to weight the previous misclassified instances heavier than the correct one in the next iterations. The algorithm continues until all the misclassified are correctly classified. In each iteration, the algorithm self corrects the errors by using gradient descent methodology to adjust the coefficients until the loss function is at its minimum. Due to the fact that the Gradient Boosted tree is not an explainable algorithm, it does have a feature to rank the importance of each feature, so that is why we deploy the LIME algorithm to have more explanation, especially if we want to like to have more explanation in the default case. Logistic Regression has selected the features for the Gradient Boosted

Tree model to focus in order to minimize the processing time. In our case, 'loan_amnt', 'annual_inc', 'dti', 'fico_range_low', 'acc_open_past_24mths' are selected to be our dataset to run through the prediction model. We used the GradientBoostingClassifier from sklearn.ensemble library to perform the prediction. We first standardized the dataset to fall between 0 and 1, and we also converted all categorical data into numeric data by using the LabelEncoder from sklearn.preprocessing. After the data transformation, we deploy the GradientBoostingClassifer model. In our tree, we selected the learning rate 0.1, max depth of our trees 3, and the number of estimators 100 for our trees in the algorithm.

```
clf_gbm=GradientBoostingClassifier()
clf_gbm.fit(X_train, y_train)
```

```
8]: GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                               learning_rate=0.1, loss='deviance', max_depth=3,
                               max_features=None, max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, n_estimators=100,
                               n_iter_no_change=None, presort='deprecated',
                               random_state=None, subsample=1.0, tol=0.0001,
                               validation_fraction=0.1, verbose=0,
                               warm_start=False)
```

Afterward, we predicted the test set to validate the data and calculated the accuracy by using the confusion matrix and classification report. In this model, the true negative is 4 and the true positive is 9153. In the classification report, it shows that the accuracy is 0.79.

```
[[   4 2388]
 [   6 9153]]
```

	precision	recall	f1-score	support
0	0.40	0.00	0.00	2392
1	0.79	1.00	0.88	9159
accuracy			0.79	11551
macro avg	0.60	0.50	0.44	11551
weighted avg	0.71	0.79	0.70	11551

In this GradientBoostedTree model, there is a function showing the importance of each feature. In the table below, the fico_range_low feature which is a column for the credit rating contributes 38% to the prediction and dti which stands for debt ratio contributes 25% of the prediction.

	feature_importances
fico_range_low	0.384321
dti	0.252873
annual_inc	0.173262
acc_open_past_24mths	0.095074
loan_amnt	0.094471

Table 1: Feature Importance from GBT model

In terms of accuracy, logistic regression and Gradient Boosted Tree have the same accuracy score referred to the diagram below..

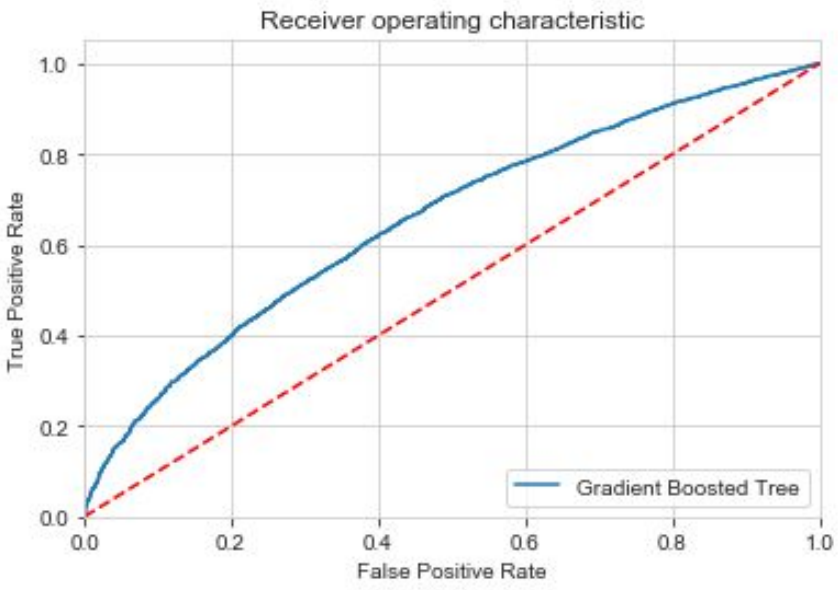


Figure 16: ROC curve

Afterward, we deploy the explainable model LIME to explain the outcome, especially the default case. We have picked one instance for an example to illustrate the explanation of the default case. We have picked the first row of the default case to implement our LIME model.

Subsequently, we use the feature of LIME explainer to look into the case.

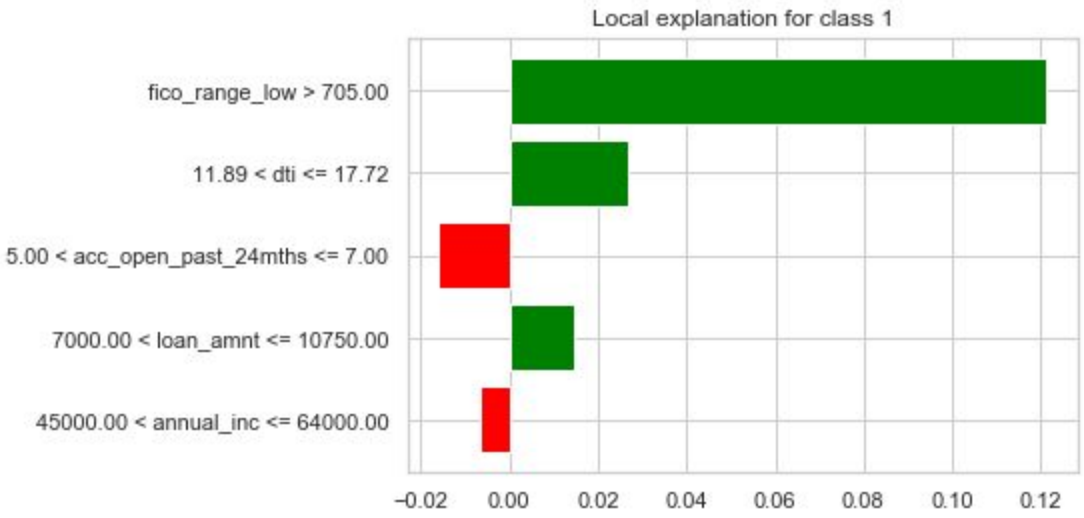


Figure 17: Feature importances from LIME

As the diagram above, it shows that fico_range_low, diti and loan_amnt are positively correlated to the default case and acc_open_past_24mths and annual_inc are negatively correlated to the case. This is coherent with our result from the feature importance function of the Gradient Boosted Tree model.

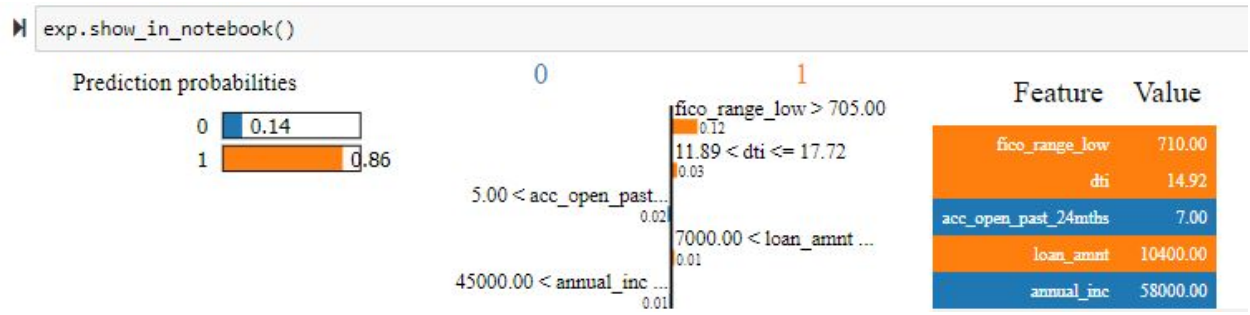


Figure 18: Explanation from LIME

In the diagram above, it shows that the LIME model elaborates the GB Tree prediction by giving the probabilities of both outcomes: default (charged off) and no default (fully paid) It also shows the coefficient or the contribution of the prediction calculation.

Gradient Boosted Decision Tree & SHAP

In this experiment, we built a Gradient Boosted Tree algorithm and tried to explain it using SHAP. For the sake of comparison, we also used Logistic Regression as a baseline.

We fit and tune two types of models: a simple linear classifier and a more complicated ensemble of tree models. For our linear classifier, we used l2-regularized logistic regression as implemented in the scikit-learn package. For our tree ensemble, we use the LightGBM implementation of gradient-boosted trees. These models will

demonstrate the tradeoff between simplicity (which leads to a degree of natural interpretability) and performance, and will also let us see the differences in how post-hoc explanation techniques work on linear and nonlinear models

We have already tuned the models using a validation split of the data. As we can see from the ROC curve and classification report. Boosted Tree has better precision of 73% compared to the precision of Logistic Regression which is 70%.

Classification Report of Logistic Regression

```
from sklearn.metrics import classification_report
print(classification_report(test_target, logistic_regression.predict(test_features_processed)))
```

```
              precision    recall  f1-score   support

           0       0.79      1.00      0.88     19754
           1       0.32      0.00      0.00      5138

    accuracy                           0.79     24892
   macro avg       0.55      0.50      0.44     24892
weighted avg       0.70      0.79      0.70     24892
```

Classification Report of Boosted Trees

```
from sklearn.metrics import classification_report
print(classification_report(test_target, boosted_trees.predict(test_features_processed)))
```

```
              precision    recall  f1-score   support

           0       0.79      1.00      0.88     19754
           1       0.47      0.01      0.02      5138

    accuracy                           0.79     24892
   macro avg       0.63      0.50      0.45     24892
weighted avg       0.73      0.79      0.71     24892
```

Both of them have similar recall and Boosted Tree has a higher overall weighted F1 score. Also, Boosted Tree has a better AUC score and ROC curve compared to the Logistic Regression.

```
*Auc of Boosted Trees* = 0.6542913068185156
*Auc of Logistic Regression* = 0.6521059361008446
```

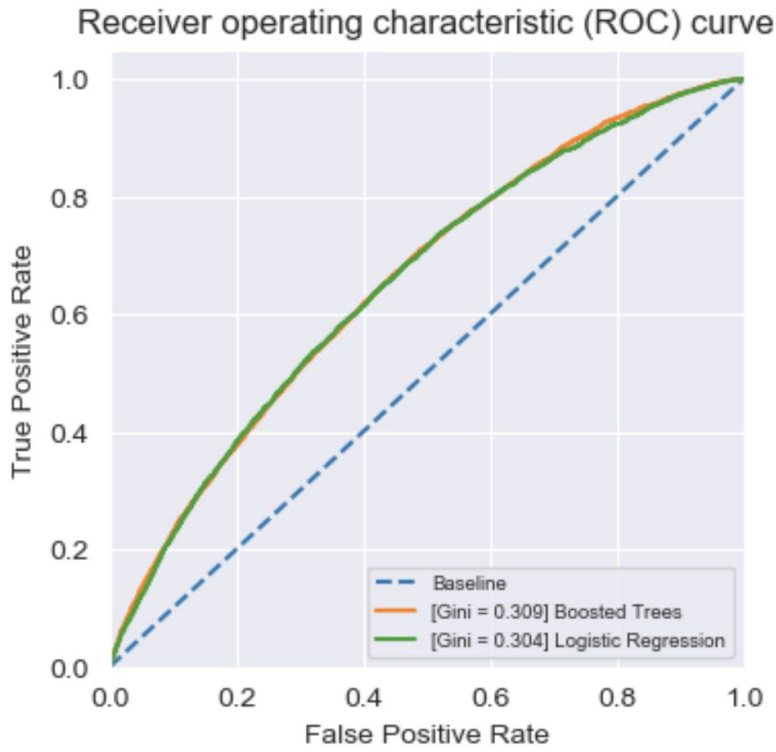


Figure 19: ROC Curve Comparison of Boosted Tree and Adjusted Logistic Regression Models

Performance and Interpretability Tradeoff

After this, we looked at the feature importance of the 2 models. We can naturally gain some level of interpretability from logistic regression, due to the fact that the model is constructed linearly (in terms of the log-odds transform of predicted probability). Since we have taken care to normalize the inputs to unit variance, we can interpret the model coefficients as both the direction and approximate magnitude of impact on final model prediction.

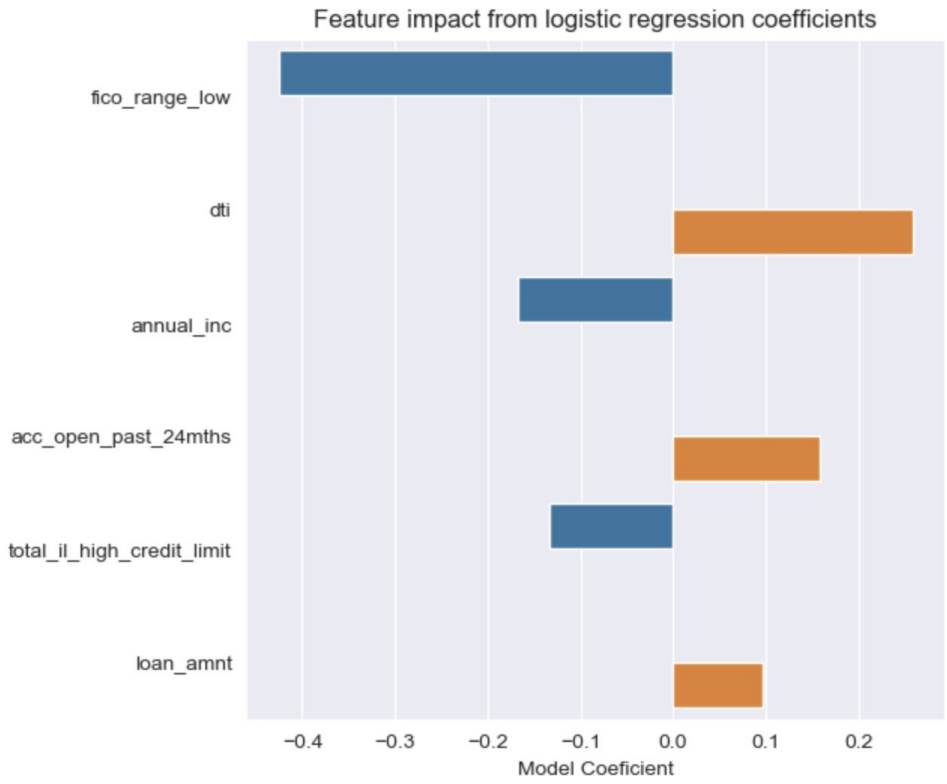


Figure 20: Feature Impact from Logistic Regression Coefficients

There is also some baked-in interpretability in tree ensemble methods, due to the fact that we can track certain measures of feature importance during training.

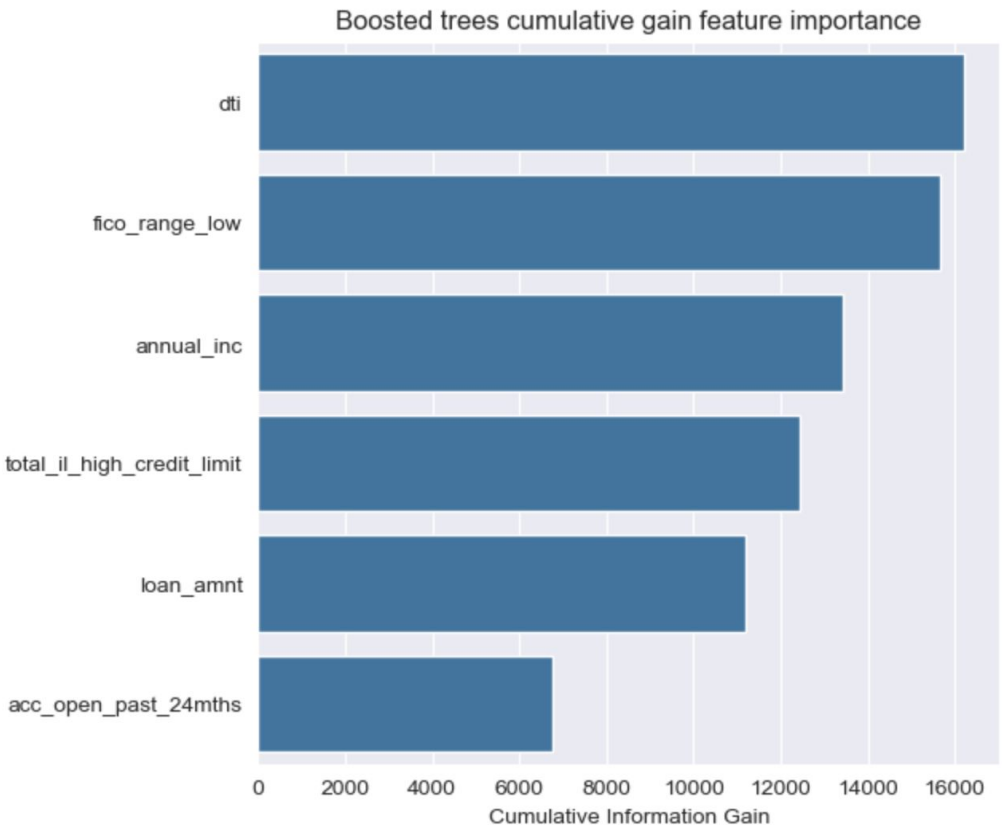


Figure 21: Boosted Trees Cumulative Gain Feature Importance

As one can see, the top features important for Boosted Tree models globally are Debt-to-Income Ratio and FICO score. And for Logistic Regression FICO score coefficient is higher than DTI.

Boosted Tree model importance does not give us directionality, whereas logistic regression gives directionality. Shap also provides a summary plot of the top features that have the most impact on the positive or negative class prediction. In this case, SHAP found FICO, Annual Income and Debt-To-Income ratio to be the top-3 factors.

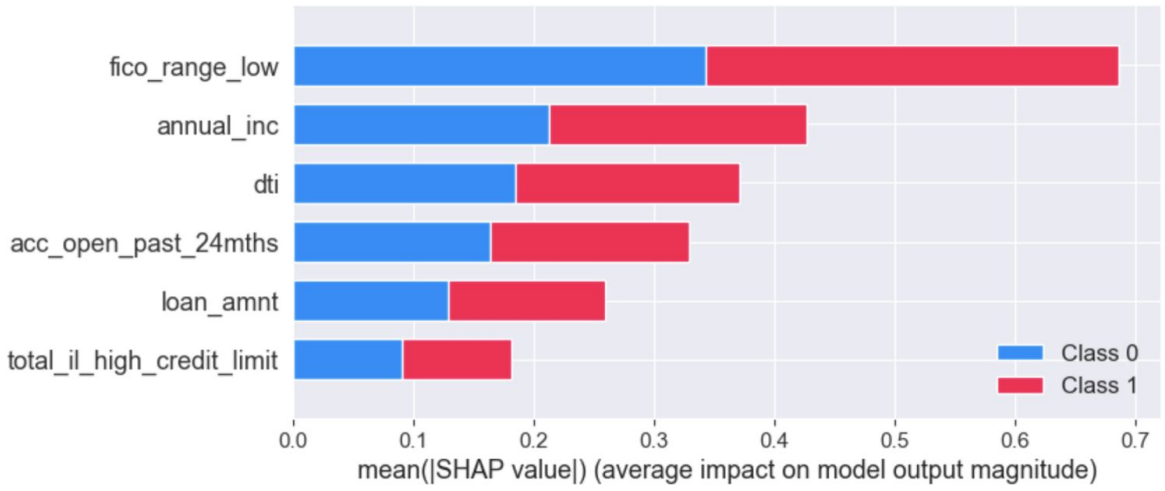


Figure 22: SHAP Evaluation of Feature Importance

6.2. Comparison of Output & Hypothesis

Logistic regression models are highly interpretable and output reliable predicted probabilities, they are vulnerable to overconfidence. Because the evaluation metrics of our final tuned model demonstrates high accuracy, we are somewhat skeptical of its predictive power as it may overstate the accuracy of its predictions. There also may be non-linearities in this problem. We create this model for the purpose of benchmarking and compare it to the 0.83 AUC metric. After an investigation into what the independent variables actually represent, we remove variable ‘total_rec_int’ as it is an outcome rather than an input into predicting default. This decreased the AUC of the logistic regression model to about 0.65.

Our hypothesis is that the explainable algorithm could be deployed to the credit lending industry, especially applying to predict the default rate, and also the LIME and SHAP could provide better explanations on top of the algorithm. For the LIME method,

we figure that LIME does not provide a more thorough or comprehensive explanation than the Gradient Boosted Tree. Gradient Boosted Tree already has a feature importance function which gives information of importance. We find it is already sufficient which indicates there is no need to deploy LIME to request more information. However, SHAP and LIME provide a good local explanation for one particular case or instance which Black box model does not offer. Normally, the Black box model such as Gradient Boosted Tree has feature importance globally but not locally. That is where explainable models could add inputs.

6.3. Explaining a Single Rejection

We pick a case from our test set where the charged_off_prediction from both models Boosted Tree and Logistic Regression are above a certain threshold (0.2). Since banks don't generally approve anything beyond 0.2, we want to explain this prediction.

	loan_amnt	annual_inc	dti	fico_range_low	acc_open_past_24mths	total_il_high_credit_limit	Boosted Trees	Logistic Regression	Loan Charged Off
22500	2.163392	0.122543	1.194965	-0.083139	-0.570206	0.473021	0.274340	0.255546	0
53563	-0.350955	-0.525909	-0.336554	-0.741500	-0.570206	-0.956372	0.359979	0.244332	0
25778	-0.099520	1.102426	0.257871	-1.070681	1.648803	3.665638	0.285913	0.206233	0
65261	-0.350955	0.209003	1.326671	-0.083139	0.380798	0.441571	0.292119	0.242850	1
15970	-0.413814	-0.597959	0.713597	0.246042	-0.253205	-0.328868	0.278401	0.213349	0
1733	-0.853825	-0.177618	0.016605	-1.070681	0.380798	-0.481974	0.256409	0.285605	0
60090	-0.696678	-0.684420	0.966519	0.246042	-0.253205	-0.327096	0.320621	0.222292	0
30843	1.258227	0.554844	0.666976	0.081451	1.014801	0.335615	0.210351	0.237508	1

Table 2a: Selected Case from Entire Dataset for Local Explanation

6.3.1 Explaining LR Rejection

As one can see in this case, the loan rejection for this user can be explained by the following:

The top-2 factors **driving his loan risk high** are:

- The debt-to-income ratio of 28.55 has a high-risk impact.
- Loan Amount of $30,000 is also having a high-risk impact.

The top-2 factors that are **reducing the loan risk** are:

- He only has 3 accounts open in the past 24 months, this is negatively affecting the charged-off probability.
- He has a pretty high credit limit of $59,722 that is also helping him.

Explaining Boosted Tree Rejection

As one can see in this case, the loan rejection for this user can be explained by the following.

Top factors	Corresponding raw input	Transformed Feature input	SHAP Feature Impact
loan_amnt	30000.00	2.163392	0.064242
acc_open_past_24mths	3.00	-0.570206	-0.042799
dti	28.55	1.194965	0.031850
annual_inc	82000.00	0.122543	0.015804
fico_range_low	690.00	-0.083139	0.002742
total_il_high_credit_limit	59722.00	0.473021	-0.001462

Table 2b: Selected Case from Entire Dataset for Local Explanation

The top-2 factors driving his loan risk high are:

- Loan Amount of $30,000 is also having a high-risk impact.
- The debt to income ratio of 28.55 is the second most important factor.

There is only 1 factor that is reducing the loan risk are:

- He only has 3 accounts open in the past 24 months, this is negatively affecting the charged-off probability.

6.4. Discussion

In this study, we want to show a black-box model with an explainable algorithm that could be utilized in the credit lending industry and it could even perform better than a machine learning model that has features of importances. We at first use the Logistic Regression Machine learning algorithm as a baseline for comparing the performance in terms of explanation. For LIME, we deploy Gradient Boosted Tree on top of it with LIME explainable algorithm. Gradient Boosted Tree model from sklearn learn has already got a feature for importance. For SHAP, it is the same situation as well. We deploy the light gradient boosted model classifier with SHAP. However, we use LIME and SHAP to explain one single instance. LIME and SHAP become a value-added to the user. Both methods are useful for explaining the instance we picked to illustrate. They provide the feature importances for that particular instance.

7. Conclusion & Recommendations

7.1. Summary & Conclusions

This project seeks to demonstrate the use of Explainable Machine Learning in a realistic setting. The goal is not to focus on how machine learning techniques can be used to predict the creditworthiness of individual credit applicants, but rather to delve into the ways in which explainability techniques can be applied to better understand how ML-based automated creditworthiness decisions are reached.

It should be noted that the authors are not domain experts in credit modeling, and the actual models discussed in the case study may not precisely reflect the best practices of the credit industry. However, the hope is that the models covered do serve as an apt case in which automated decisions have a serious impact on the lives of real people, and the predictive information and models used have a degree of sophistication and complexity equivalent to real-world scenarios.

Since the goal of this project does not lie in the exploration of data or the modeling process, we have omitted EDA and model tuning from this project. Although peer-to-peer loans are primarily evaluated by human investors, credit models serve as an important tool to help investors choose which loans to fund. Lending Club uses a creditworthiness model to assign a grade to each new application they approve, and they also release their data to allow investors to build their models. Outside of peer-to-peer lending, credit card companies use creditworthiness models every day to instantly approve or deny applications and generate credit card pre-approvals.

In both peer-to-peer lending and credit card approval, the decisions made using the output of creditworthiness models have a significant impact both on credit issuers/investors and on credit applicants. An investor must be comfortable deploying his or her capital behind the output of his or her credit model, and if the model is miscalibrated or buggy, the investor stands to sustain material losses. Similarly, applicants who are refused credit based upon the output of a model can have their lives immediately impacted by the lost opportunity for financial liquidity. Accordingly, government regulation protects credit consumers from potentially unfair credit decisions. For example, the US Equal Credit Opportunity Act (ECOA) requires

creditors to be able to give the specific reason or reasons why they rejected an application whenever they do so. If such decisions are made by a machine learning system, explainability becomes quite an important component of the automated decision-making process!

In this study, we found that Boosted Trees are more accurate than Logistic Regression. Both models support some global feature importance, we can infer the local explanation of Logistic Regression. And to explain a black-box model like Boosted Tree, we need to use SHAP or LIME to explain an individual case. SHAP has certain axiomatic guarantees for explainability and as it is based on sound game-theoretic principles called Shapley Values. It also has better support for Boosted Trees. So if we use Boosted Tree + SHAP, we can get both high accuracy and good explainability.

7.2. Recommendations for Future Studies

Having constructed different models, we found the process of predicting loan applicant probability of default to be relatively simple. However, there is room for improvement in terms of feature selection. In this paper, we determine risk objectively. This means we calculate the probability of default regardless of gender, race, age, educational level, number of dependents, and so forth as they are not included in our dataset. However, this makes the validity of the models questionable. Those factors often carry significance in the ability of one to repay a loan. Our goal is to derive trustworthy explanations for credit lending and to ignore the contribution of those factors is negligent--especially for high stakes decision making. It may be beneficial to

construct a model that includes subjective socio-demographic characteristics and compare the resultant risk determinations to observe what type of information is lost in the attempt to increase fairness.

Furthermore, the exclusion of categorical variables from the feature set in the classifiers are subject to debate. Visualizations of our categorical variables seem to suggest some predictive power. More exploration in terms of the categorical variables would be useful to possibly increase overall accuracy.

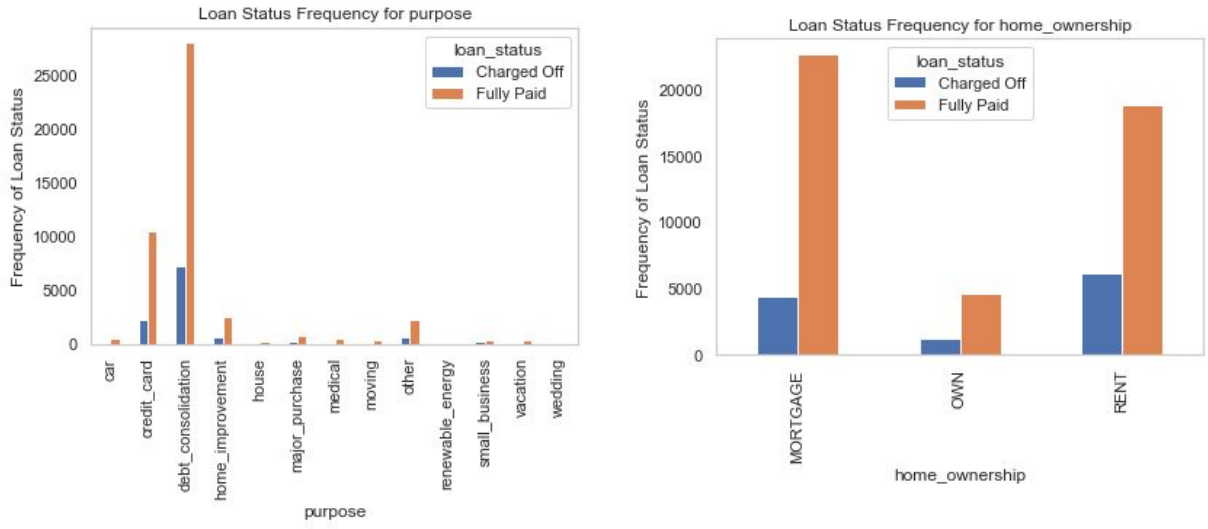


Figure 23: Loan Status Frequency for 2 Categorical Variables: Purpose and Home Ownership

Lastly, we suggest the continued effort in monitoring financial indicators. Due to yearly changes such as inflation rates and the health of the market, loan risk factors are rarely stagnant. Due to ever changing circumstances, historical data, therefore, loses its significance over time and cannot predict the future behavior of loan applicants. Accidents, health issues, or unprecedented events such as a baby or even a global pandemic severely change creditworthiness and therefore cannot be

estimated with certainty at the time of assessment. Therefore, it is no wonder that the lending club dataset continues to grow with time, and thus continues to be explored.

8. Bibliography

[1] N. Kallus and A. Zhou. The fairness of risk scores beyond classification: Bipartite ranking and the xauc metric. In Advances in Neural Information Processing Systems, 2019.

[2] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. In Nature Machine Intelligence, Vol 1, May 2019, 206-215.

[3] S. Lundberg and S. Lee. A unified approach to interpreting Model Predictions. In Neural Information Processing Systems, 2017.

[4] M.T. Riberio, S. Singh, and C. Guestrin. "Why Should I Trust You?" Explaining the predictions of any Classifier. In Knowledge Discovery in Databases, 2016.

[5] L.S. Shapley. https://en.wikipedia.org/wiki/Shapley_value

[6] Redlining. https://en.wikipedia.org/wiki/Redlining

[7] Babaev, Dmitrii, Maxim Savchenko, Alexander Tuzhilin, and Dmitrii Umerenkov. (2019, July). ET-RNN: Applying Deep Learning to Credit Loan Applications. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2183-2190).

[8] Choi, Keunho, Gunwoo Kim, and Yongmoo Suh. (2013). Classification model for detecting and managing credit loan fraud based on individual-level utility concept. ACM SIGMIS Database: the DATABASE for Advances in Information Systems, 44(3), 49-67.

[9] "Code of Federal Regulations." Consumer Financial Protection Bureau, 12 Apr. 2020, www.consumerfinance.gov/policy-compliance/rulemaking/final-rules/code-federal-regulations/ .

[10] S. Montgomery, “What's in an Algorithm? The Problem of the Black Box,” Tufts Observer, 25-Feb-2019. [Online]. Available: https://tuftsobserver.org/whats-in-an-algorithm-the-problem-of-the-black-box/. [Accessed: 13-Apr-2020].

[11] E. Choi, M. T. Bahadori, J. A. Kulas, A. Schuetz, W. F. Stewart, J. Sun. RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism, In Neural Information Processing Systems (NIPS) 2016.

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, Jan. 2002.

[13] “3.2.4.1.5. sklearn.linear_model.LogisticRegressionCV,” scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html. [Accessed: 17-May-2020].

[14] “sklearn.feature_selection.RFE,” scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html. [Accessed: 17-May-2020].

[15] “sklearn.linear_model.LogisticRegression,” scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed: 17-May-2020].

[16] “sklearn.model_selection.GridSearchCV,” scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html?highlight=gridsearchcv#sklearn.model_selection.GridSearchCV. [Accessed: 17-May-2020].