Finance and Interpretability of Machine Learning Models

by

Dorina Ababii, Tsuyoshi Iwata

A research paper submitted in the fulfillment of the requirements for the credit of Advanced International Research Projects

Dept. of Banking and Finance

Zurich University of Applied Science

2021 F.S.

Supervised by Dr. Bledar Fazlija

May 30th, 2021

ABSTRACT

Artificial intelligence is used in finance to develop accurate predictive models in numerous real-world applications. To increase accuracy, the machine learning models use complex rules that are inaccessible to the users. Therefore, the interpretability issue has become subject to not only the user but also regulatory institutions. Recent research in Interpretable Machine Learning approaches has provided various interpretation methods that can help humans optimize predictive processes and supply reasonable human-friendly interpretations. Firstly, we trained two interpretable models and one black box model using a credit-card fraud-detection dataset in this paper. Our case study showed an example that the black-box model outperforms the other two interpretable models. Therefore, the black-box model can be an attractive candidate to use in practice. Secondly, we provided a use-case of 3 interpretation methods with the trained black box and showed that interpretation methods could increase the black-box model's interpretability.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

BB – Black-box

AI – Artificial Intelligence

ML – Machine Learning

iML – interpretable Machine Learning

DL – Deep Learning

PDP – Partial Dependence Plot

1. INTRODUCTION

Many black-box (BB) models are used in finance to solve specific problems because machine learning (ML) is the prime classical method. Since the prediction-making rule is in the BB, the predictions are opaque and hard to interpret. Highly regulated financial institutions use interpretability reasons to argue why they do not adhere to the total integration of the ML models with the financial system, in contrast to other sectors (Carvalho et al., 2019). Recent regulations concerning artificial intelligence (AI) and ML in finance debate the risk of adopting AI in financial institution-related activities (EU, 2019a). Basel Committee on Banking Supervision (BCBS) says financial institutions must provide an explanation of model construction, interpret the prevision-making process and assumptions, provide a detailed description of input data and mathematical formulas employed before integrating a specific model.

According to the latest trends, financial institutions are more likely to adopt BB models in their prevision making processes, essentially driven by increasing performance in specific actions like classification. Even though the BB models provide accurate predictions, automated prevision-making models operate in their way without offering the rules-based process description. At the same time, the situation can be unprofessional because the advisor cannot explain why the client did not get the credit approval and non-transparent because the model generates the prediction based on historical data and self-defined rules (EU, 2016).

Discussions on the reasons for and results of BB model adoption in financial-related activities have focused on increasing performance, increasing margins. Still, there has been less work studying interpretable techniques' potential in enabling black-box use. In the literature on interpretable AI, these few techniques are sometimes insufficient as an end-use resort. Researchers must provide an in-depth comparative analysis to gain further understanding of interpretable techniques in the context of financial modeling. Focusing on the BB models' interpretable techniques, the financial institution can prepare the ground for practical scale ML methods in finance and fit the BB models with the current policies objective (EU, 2019b).

Our research paper seeks to define a reasonable interpretation for ML models by understanding feature discrimination for fraud detection that is interpretable for financial institutions, to deliver a good model explanation that refines confidence and understanding of fraud detection models. We aim through the advanced international research project to study interpretable machine learning (iML) methods compatible with BB models widely employed in financial modeling. To explore our goal, we will respond to subsequentially derived queries.

First, we identify popular ML models applicable in finance and interpretation methods appropriate to any ML model (section 2). Before presenting our methodology (section 4), we offer a detailed analysis of the dataset (section3). Once we train the model of run the interpretation method, we present our result in section 5.

2. LITERATURE REVIEW

With the soaring use of ML and AI to predict various types of outcomes, a need to understand the algorithm had arisen (Felzmann et al., 2020). The automated prediction-making systems' impact can influence particulars and society in general (Mittelstadt et al., 2016). Algorithms are "mathematical constructions with a finite abstract, effective, compound control structure to accomplish a given purpose under given provisions" (Hill, 2016) and can have an impact by creating ethical challenges concerning responsibility and human-machine interactions (Coeckelbergh, 2020 and Matthias, 2004). According to Jordan and Mitchell (2015), ML computer systems represent algorithms that refine self-learning through experience and data analysis without being specifically programmed.

In the last decades, these self-operating algorithms have demonstrated remarkable performance in varied duties – often outperforming humans in terms of speed – and are applicable in assessing different complex decisions (Buetti-Dinh et al., 2019). During their research, Nichols, and Schwabish (2014) determine that high costs of the workforce incite fast integration of automatic methods such as classifications.

Artificial Intelligence has allowed ML use in vast fields like healthcare, banking, and finance, insurance. As defined by J. McCarthy during the Dartmouth Conference in 1956 and developed by Turing (1950), AI is the faculty of a machine to reproduce intelligent human behavior in a specific context. In finance, precision is the key to success; therefore, financial players prefer to use complex models like BB models to increase prediction performance. Even the relevant field directly concern humans (ex. cancer prediction, fraud detection, credit default, etc.); by seeking to improve performance, ML models become heavy and highly inaccessible to human comprehension.

Like a few decades ago, the lack of interpretability of these models' input boundaries for real-world application. The increase in model complexity is, therefore, the reason why humans fear the total integration of such models in more critical decisional processes, such as credit approval, job screening, cancer detection, and sovereign cars (Tsang et al., 2021; Fong and Vedaldi, 2017). Many of these models do not allow a clear interpretation of the results due to

their complexity and make the interpretability of ML models a valuable asset. Interpretability is a primordial criterium for the adoption of ML in actual-world practice such as finance.

iML seeks, with this regard, to develop methods to increase transparency or interpretability (Lipton, 2016). In these circumstances, various interpreters for those ML have been developed in recent years. Researchers have been talking about the classification problems of those methods since the suitability of each method depends on what kind of situation it is applied to.

The world of ML models is vast, and we cannot cover every model. In this section, we are going to present the most common MLM. The taxonomy of iML algorithms depends on the desired outcome, either learning style, function, or the problem it studies. To offer a clear understanding of the models to our reader, we will present a literature review of the methods based on the model-specific and model agnostic classes.

2.1. MACHINE LEARNING MODELS AND INTERPRETABILITY

In finance, precision is the key to success; therefore, financial players prefer to use complex models like BB models to increase prediction accuracy. ML models' development demonstrated significant performance and accuracy in varied duties in the last decades – often outperforming humans in terms of speed – and solve different complex decisions (Buetti-Dinh et al., 2019) and has become heavy and highly inaccessible for human comprehension. With the increasing use of AI to predict, a need to understand the algorithm had arisen (Felzmann et al., 2020). Automated prediction-making systems can negatively influence particulars and society in general (Mittelstadt et al., 2016).

Mittelstadt et al. (2016) note that model complexity tends to challenge and shuffle humans. According to Zinke et al. (2018) and Michail et al. (1997), model complexity constantly increases concerns about personal data and put in light the question of model transparency. Transparency is a non-exhaustive criterium to distinguish if a model is interpretable. The meaning of interpretability is ambiguous. It generally describes model interpretability, multiplicity and detectability, ethics, confidence, and model fairness.

While the ML models are performant in various fields and the previsions are commonly accepted, the need to define what is a reasonable interpretation has become primordial. The common acceptance of the model prediction is not enough as ML can face errors, like incidences in finance. A single metric, such as accuracy, is an incomplete description of most real-world tasks (Doshi-Velez and Kim, 2017; Varshney & Alemzadeh, 2017).

Generally, interpretability is not required if (1) the prediction does not high-stake impact or accrue societal consequences if the results are mistaken or (2) the issue is well studied. Immediately once the financial model has a crucial impact on approval or decline of the transaction as fraudulent or changing fraud patterns over time, make interpretability primordial (Doshi-Velez & Kim, 2017).

Given societal acceptance, it has become relevant to deliver fairness and guarantee previsions perform as anticipated. We identify few criteria to assess interpretability according to Doshi-Velez & Kim (2017) and Keil et al. (2004):

- Fairness – allow the model to cope with biases and ensure a non-discriminatory prediction (ex. age, etc.)
- Privacy – guarantee that reliable information is protected.
- Reliability and robustness – make sure that changes in the input data do not considerably change the output.
- Causality – make sure that the model chooses just causal interactions.
- Trust – humans prefer a system that provides explanation rather than a black box that predicts by itself.

Concerning the commonly accepted interpretability criteria, we present a literature review of ML models in the following subsections focusing on transparency.

2.1.1. INTERPRETABLE MODEL

After studying circular paper on ML models, we identify models that explain themselves from many models by extracting a prediction rule. In big lines, developers can employ groups of algorithms to create interpretable models.

According to Simon et al. (2015), ML is a computer science category that developed from employing pattern identification and automatic learning principle in AI. Generally, it is the model that uses data sets to learn and make previsions. As presented by Mitchell (2006), a professor at Carnegie Mellon University: 'A computer program is said to learn from experience E concerning task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Transparent ML models focus on tabular data and predefines features. We can list three iML classes, including decision lists, decision sets, and linear models. Further in this section, we describe the interpretable methods for opaque ML models. (Letham et al., 2015) prove that it is possible to generate simple rules which would be commensurable with decision trees and at the same time have a sporadic classified feature preserving high model accuracy. For example, Bayesian Rule List uses permutations allowing it to deal with a large number of features. To complete the study on decision rules, (Su et al., 2016) introduce a two-level Boolean control using AND, OR, and NOT rule, which aim to interconnect features using logical allegations.

Generally easy to use in predictive analysis, linear regression models are statistical models that generate predictions based on variables like age, level of education, wage, etc. The name of linear regression suggests a linear relationship between the dependent and independent variables (Sharma et al., 2020) and (Hastie et al., 2013). The linear relationship of the interactions of the dependent variables makes the model interpretable and straightforward on a modular level. The effects are additive and offer the possibility to separate them. We can apply the model if we have either numerical, categorical, or binary features. The mode makes it easy to identify feature importance using the t-statistic test (Hastie et al., 2013; Molnar, 2020).

$$t_{\hat{\beta}_j} = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)}$$

The test indicates that the feature importance in the linear regression model increases with its weight. Even though the linear model is simple, it does not provide human-friendly interpretations. The prediction is contrastive, but the reference value is 0 or the specific category. We can extend the linear model in various ways, and this character of the linear regression model makes the interpretability poor (Molnar, 2020)

The most common transparent predictive algorithm, logistic regression, is a supervised learning technique (Hastie et al., 2013). Similar to linear regression approaches, logistic regression solves actual questions, and it applies to classification tasks. Contrary to linear regression algorithms, logistic regression can only take values from a given space: yes or no, zero or one, etc., and compliment it as the linear model is not initially designed to provide probabilities (Hastie et al., 2013; Molnar, 2020). To constraint results between 0 and 1the model uses the logit function (Molnar, 2020):

$$logistic(\eta) = \frac{1}{1 + e^{-\eta}}$$

Contrary to linear regression, the interpretation of the coefficients of the equation is less intuitive and can be derived from the transformation (Hastie et al., 2013; Molnar, 2020):

$$\log\left(\frac{P(y=)}{1-P(y=1)}\right) = e^{\beta_i}$$

The term e^{β_i} suggests that one unit change in the input feature changes the multiplicative coefficients by e^{β_i}. In the case of fraud detection, we need to predict binary categorical features (yes or no, which may be transposed in one or zero). In this case, we could interpret the logistic regression result as follow: if the reference feature changes to the other category, the estimated e^{β_i}.

Decision trees are the preferred classification algorithm. The graphical representation corresponds to a tree where the nodes compose the dataset features. The branches illustrate the rule considered (Hastie et al., 2013), and the leaf nodes represent the decision. Decision tree models, designed to complement linear regression models, can be employed if there are no linear interactions between feature and outcome or between features. In the decision trees model, feature importance metrics help the user understand how many features increase model accuracy. Among the model's advantages, we count feature interactions that can identify distinct groups, offer a natural representation of the prediction, and create human-understandable interpretations. The main disadvantage of the model is that features must be categorical most of the time (Hastie et al., 2013; Molnar, 2020).

Also employed in classification and regression tasks, support vector machines are used in n-dimensional space to divide the data set and place the new input in the correct category (Sharma et al., 2020). This model offers excellent model accuracy in forecasting static and dynamic courses (Yan and Ouyang, 2018).

Leaning on the principle of Bayes theorem, Naive Bayesian solves categorization issues. Naïve Bayesian algorithm uses probability-based classifiers and eases the sentimental analysis, classifying articles, etc. (Sharma et al., 2020).

Clustering is generally a known technique that generates cognitive maps where the inputs are receptors of the information, creating a space between inputs defining whyx_i It is different from x_j. Through pattern recognition of unlabeled data, clustering can provide prediction (Hastie et al., 2013). Other methods employed to identify cluster centers in a set of input data, K-means clustering models produce iterative distancing from the center to minimize the total variance (Hastie et al., 2013). Memory-based classifier, K-nearest neighbor models (unsupervised learning), does not ask for model fitting. It is employed to classify data according to the "majority

vote across input data (Hastie et al., 2013). These simple models solve classification problems like satellite image scenes.

Functioning on the shrinkage mechanism of the ridge regression, principal components analysis mise on the dimension reduction method and are part of projections of the input data sets by hypothesis considered uncorrelated (Hastie et al., 2013). Further studies have been made on feature extraction techniques to select statistical similarities or not by directly studying the inputs.

In the past, in the presence of a significant amount of data sets, commercials adhered to the association rule methods that imply identifying standard features across the data set in sale transactions. For example, the feature's probability density is relatively essential (Hastie et al., 2013).

Once the model becomes complex and feature space increases, the explanations became less comprehensible. To solve the default of the models presented above, (Lakkaraju et al., 2016) develop Interpretable Decision Sets to ease the description and predictability of ML approaches applying Boolean rules if-then rules. The method comes with attributing a specific feature to a class and can be interpretable for multi-class designation. As the decision list became exhaustive, the model interpretability became less accurate. (Yin & Han, 2003) develop a new technique called CMAR or Classification based on Multiple Association Rules. According to the authors, CMAR posed a higher accuracy than other classification approaches (CBA ansdC4.5). The method efficiently deals with many mined association rules and performs well concerning class prediction.

2.1.2. BLACK BOX MODEL

Famous for using a considerable amount of unsupervised data, deep learning (DL) gains territory in banking and finance-related activities (Chai et al., 2013 and J. Huang et al., 2020). DL is an input-oriented model that is close to the action of the human brain (Dayan, 2008). DL inputs create simuls, and the algorithm constructs cognitive maps based on observations (Barlow, 1989). Learning is the first observation of the input features x_i, with a probability $P_i[x]$, and some predefine know information (Dayan, 2008). Widely applied in any field, we present DL models especially applicable in banking and finance.

Reinforcement learning techniques like Q-learning apply to stock market predictions and enhance profitability by outperforming the buy and hold strategies and increasing prediction

accuracy (C. T. Chen et al., 2018; Martínez-Miranda et al., 2016; Zhang and Maringer, 2016). Jeong and Kim (2019) have proved quality learning methods increase model accuracy and break the overfitting issue caused by insufficient data. Generally, the model is helpful in the case of the learning task of action in a specific state (Hastie et al., 2013).

Further, ensemble learning ML models imply combining strengths of a collection of more straightforward methods to increase predictive abilities. Famous for breaking down the problem into smaller tasks, ensemble learning models provide the user with a base learning model accounting for the training data set and the composite predictor (Hastie et al., 2013).

Also known as bootstrapping, bagging represents an ML ensemble model. The algorithm increases the stability and accuracy of the prevision-making model or estimator using statistical classification and regression problems through an approximative illustration of the posterior mean (Hastie et al., 2013). Constructed around the averaging model approach, it was the first design for decision tree models. First, the model includes fitting the algorithm to the trained data set considering the regression problem based on the input x. Second, the model considers the mean of the overall model predictions across samples and diminishes the variance of the model. Lately, we compare the bootstrapping estimate with the model estimate. As explained by (Hastie et al., 2013) in section 8.7 of their book, the bootstrapping estimate will only differ from the original estimate if the predictions follow a nonlinear model. Essentially, in Decision Trees, the proper estimator is the prediction according to the input data. The bootstrapping estimator is the mean prediction of the input data set. Generally, the bagged decision predictor function as follows (Hu et al., 2012):

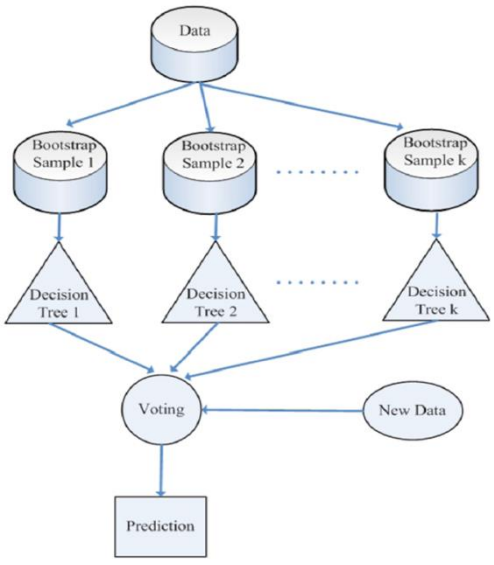

Figure 1. A general framework of the bagged decision tree predictor

Applicable to both classification and regression models, boosting models are statistical algorithms that combine different classifiers' output and generate a classification panel (Hastie et al., 2013). Based on the model 'AdaBoost. MM1' algorithm constructed by Freund and Schapire (1997) implies data modification at each boosting level attributing personalized weights to each input data. This iterative procedure contributes, therefore, to increase model prediction by continuously decreasing the error rate. According to Breiman (1998), bagging models are the best off-the-shelf classifier in the world.

Another DL models are neural network (NN) models that are applicable in diverse task solving. Barlow (1989) noted that NN models replicate the natural neurons and have comparative advantages over the linear models. NN executes tasks based on inputs, and there is no need to rule assemble the network (Hastie et al., 2013). The models are simply nonlinear statistical models with one or more hidden layers to the same authors. Usually employed in regression or classification tasks, NN works like in Figure 2. We should consider additional bias non-observed units for every unit in the hidden and output layer while working with the algorithms. According to different researchers, small numbers of layers nodes and hybrid models have better model accuracy than the initial model (Hastie et al., 2013).

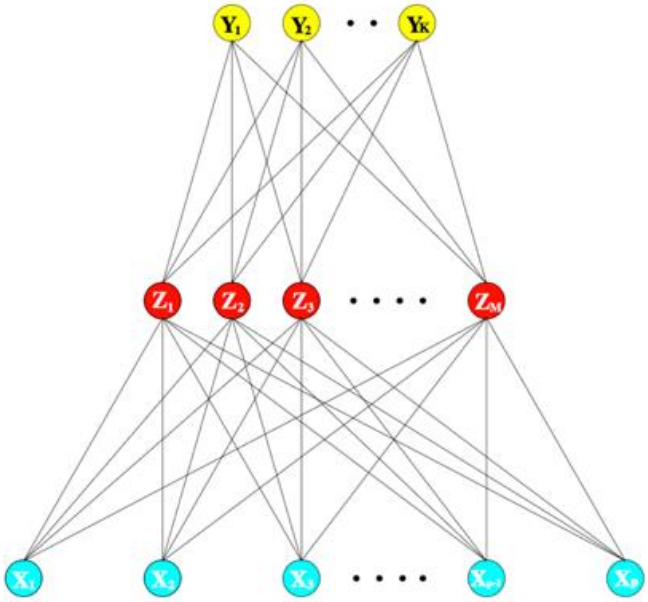

Figure 2. Schematic of a single hidden layer, fee-forward neural network

NN models contain weights – unknown parameters – that fit the model's training data set. The big challenge of the NN model is the number of parameters and optimization problems that may be unstable if the instructions are not followed properly (Hastie et al., 2013).

NN models have similar structures and great model accuracy in the financial sector and solve different tasks, including (Chai et al., 2020):

- exchange rate prediction: convolutional neural networks, feedforward neural networks, deep neural networks models.
- Stock market predictions: deep neural networks models, recurrent neural networks, convolutional neural networks, feedforward neural networks models.
- Stock trading: recurrent neural networks, convolutional neural networks models.
- Banking default risk and credit: convolutional neural networks models.
- Portfolio management: recurrent neural networks models.
- Macroeconomic prediction: deep neural networks models.
- Price prediction: artificial neural networks models.

The main issues of the NN models are starting values problem, overfitting, input scaling, the number of hidden units and layers, and multiple minima. Starting values issues connect to the fact that the nonlinear model acts as a linear model in weights close to zero. It leads to zero derivatives, creating exact symmetries, and a stational algorithm (Hastie et al., 2013). The overfitting challenges account for attributing more than one weight to the same feature. Authors of the same book specify the possibility of adding a penalty error function adding the terms to the respective weights. Since any modification of the inputs may affect the quality of the solution, scandalized approaches of imputing zero o the inputs mean and one to the standard deviations are necessary. As Hastie et al. (2013) concretize in their book, the number of hidden units influences the model flexibility. Hence it is recommended to increase the number of hidden companies by more than five up to 100. It is crucial to remark that the number of hidden layers is associated with the experience. Multiple layers influence the feature construction—numerous minima issues related to the starting configuration. Ripley (1996), average predictions across the networks could be a good approximation of the end prediction.

The specified solutions for neural network models contribute to increasing model opacity, and therefore complementary methods must be employed to make the models interpretable.

2.2. MODELS APPLIED IN THE FINANCIAL INDUSTRY

Credit scoring and credit card fraud detection are popular use-cases of ML in the financial industry.

The development of the credit scoring model is a critical task in the financial industry (Chuang & Lin, 2009; West, 2000). Linear Discriminant Analysis (LDA) is the earliest applicable model for credit scoring. Afterward, various new techniques, such as the Logistic Regression (L.R.) and the Artificial Neural Networks (ANN), have been applied (Chuang & Lin, 2009). Many works of literature have been using various machine models to identify the best-performing model for credit scoring. However, there has been no consensus of which model is the best prediction model based on its prediction accuracy (Dastile et al., 2020).

One difficulty of the credit scoring prediction is the data imbalance, which comes from the fact that default events happen only infrequently. This feature makes it difficult for models to predict outcomes due to ML mechanisms (Dastile et al., 2020). Various technical sampling methods, such as over-sampling and under-sampling, have been developed to make it easier for the model to predict (Dastile et al., 2020).

On the other hand, since the number of e-commerce transactions has been drastically increasing in the last decades, the importance of the development of high-speed and automated credit card fraud detection models also has been growing (Kokkinaki, 1997).

In comparison with the credit scoring model, the credit card fraud detection models have similar requirements as they also have to tackle the imbalance data problem (Maes et al., 1993; Phua et al., 2004) and have to deliver good performance for out-of-sample data (Maes et al., 1993).

2.2.1. DISCUSSION OF PREDICTIVE ACCURACY MEASURES

The imbalanced data problem leads to the issue of the selection of the measure of the model performance. One of the typical accuracy measures is Percentage Correctly Classified (PCC) (Dastile et al., 2020), and overall accuracy measures and calculates the proportion of the correct prediction among the whole predictions.

		Predicted	
		Positives	Negatives
Actual	Positives	TP	FN
	Negatives	FP	TN

Figure 3. A confusion matrix with a correct prediction, adapted from Dastile et al. (2020), P.9

Phua et al. (2004) discussed that due to the imbalanced data problem, the overall accuracy measure, like PCC, is not suited to evaluate the model accuracy since models can get a good PCC score easily only by predicting all transactions positive (negative). One of the alternative measures is G-Means, is (Dastile et al., 2020),

$$G - mean = \sqrt{\frac{TN}{(FP + TN)} \times \frac{FN}{(TP + FN)}}$$

And this is resistible to the imbalanced data problem (Kubat & Matwin, 1997).

The other popular candidate is Area Under ROC Curve (AUC), surrounded by the ROC curve (Figure 4). ROC is the curve made of coordinates that consist of percent true positive and percent false negative at each threshold convert predicted probabilities by model into binary classification. The components of each coordinate, Percent True Positive and Percent False Negative, in Figure 4, can be calculated by the following definitions:

$$Percent\ True\ Positive = \frac{TP}{TP + FN}$$

$$Percent\ False\ Negative = \frac{FP}{FP + TN}$$

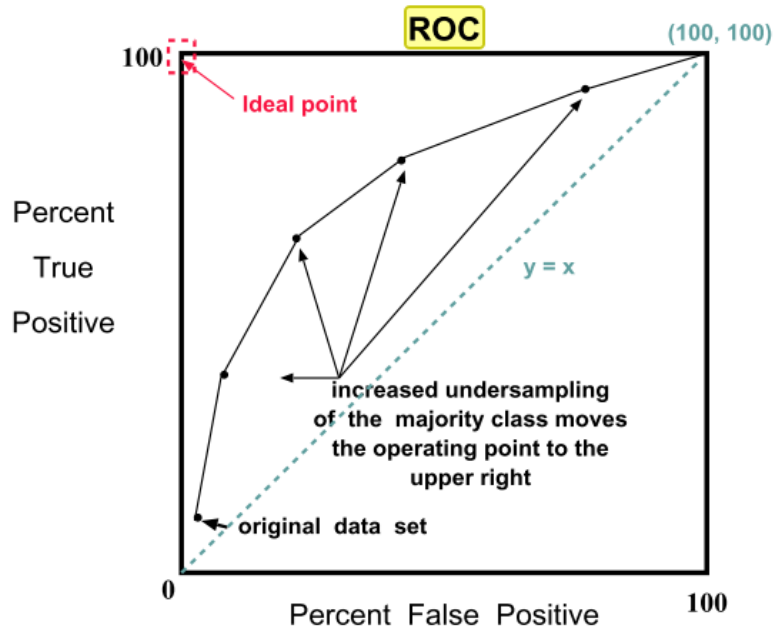

Figure 4. Illustration of ROC curve, retrieved from Dal Pozzolo et al. (2018)

This AUC measure is superior to accuracy measures, such as PCC and G-Means, in the sense of its characteristic of independence on threshold level (Ling et al., 2003).

However, all PCC, G-Means, and AUC are not very useful in comparing fraud detection models or credit scoring models since they do not consider the cost of forecasting error. For example, a false-negative error causes higher costs than a false-positive error in the credit scoring model. The false-positive errors only lead to losing an opportunity to lend money to earn a small amount of interest fee. The false-negative error leads to a considerable loss of banks that comes from the default event. PCC, G-Means, and AUC take false-positive error and false-negative error into account equally. This feature is inconvenient to compare models in the application of credit scoring and credit card fraud detection.

On the other hand, Type I Error, and Type II Error are straightforward and reasonable measures to compare models if you know error costs for models' miss specification (Dastile et al., 2020). Type I Error is the conditional possibility that the model mistakenly predicts a legit transaction as a fraud. Type II Error is the dependent possibility that the model mistakenly predicts a fraud transaction as a legit transaction. If you know the expected costs that cause by false-negative error and false-positive error, you can calculate an expected loss from each model's prediction and compare them (Hand & Till, 2001). In general, the cost from the Type II Error is higher than from the Type I Error in the credit scoring context (Chuang & Lin, 2009). In the event of credit card fraud, customers must partially or entirely pay the loss from credit card fraud. The banks must owe all the loss, including a different feature in the default modeling compared to credit score modeling. From the service providers' perspective, the event implying that the relative error cost of false-negative to one of false-positive in the credit card fraud context is lower than in the credit scoring (Maes et al., 1993).

2.2.2. DISCUSSION OF INTERPRETABILITY OF PREDICTION MODEL IN THE FINANCIAL INDUSTRY

As we already discussed in the former sections, the criterion of a good model is the model's accuracy and interpretability.

The easiest way to maximize the model's interpretability is to use interpretable models, such as logistic regression and decision trees. However, much literature reported that BB models performed well and outperformed interpretable modes in the sense of their predictive accuracy (Maes et al., 1993; Neagoe et al., 2018; Tomczak and Zieba, 2015; Trinkle and Baldwin, 2007).

Because of the superiority of BB models, the development of the methods that boost the interpretability of BB models, such as Partial Dependence plot (PDP), LIME, and SHAP, has

been actively processing in recent years and has been applied to BB models in the application in the financial industry.

Those methods for the interpretability of ML models help service providers of lending and credit card transactions detect flaws of their BB models, which are difficult to find due to the models' non-transparency and provide human-friendly explanations for their customers. And those models also help the service providers to meet the requirements from the legal side.

Questions around ethics and data privacy circulate around AI and ML concepts as the societal challenge increases with self-learning algorithms. Only two years ago, in April 2019, the European Commission group of experts on AI published the ‘Ethics Guidelines for Trustworthy AI’. In the guidelines, the responsible group develops on the essential requirements that self-learning algorithms must meet to be trustworthy, among which fundamental human rights, data privacy, diversity, and societal well-being represent the main societal concerns (EU, 2019b).

DL models has been proved to cause discrimination across minors in case of alternative sanctions, errors being caused because past oriented algorithms cannot adapt as humans unless it is changed (Angwin et al., 2016; Donnelly and Embrechts, 2010; O’Neil, 2016). Interpreting the algorithm is a way to identify discrimination and biases in the model and fit ethical concepts into the machine (O’Neil, 2016).

In the U.S., there exists the Equal Credit Opportunity Act, enacted in 1974, that prohibit to use of gender, marital status, race, whether an applicant receives welfare payment, color, religion, national origin, and age for the judgment of the credit card application (Crook, 1999).

Nevertheless, Freeman (2017) detected the existence of racism in the credit card industry. In E.U., Equal Credit Opportunity Act, enacted in 1975, requires lenders to explain the decision to give or refuse credit (Trinkle and Baldwin, 2007). The expansion of the model's interpretability contributes not only to the providers' internal audit demand but also to mitigate the limitation of the BB models' usage in practice.

Ethical and legal dimensions trigger the attention of regulatory institutions, but there is also a need for precision of what is ‘interpretability’. To understand what is a good interpretation, we first must picture the interpretability as a two-road interaction between individuals and computers that perform self-learning models. The definition implies a need to comprehend

research from data science, human science, and human-computer interaction (Abdul et al., 2018).

Interpretability suggests unclear definitions, and there is not a joint agreement. Nevertheless, when it comes to ML, interpretability is usually confounded with quasi-mathematical techniques (Lipton, 2016). Usually, close related, interpretability refers to expandability if humans can precept the method and the definition perception is related to their authors (Adadi and Berrada, 2018; Doran et al., 2018; Gilpin et al., 2018; Lords, 2019; Rudin, 2019).

According to (Bracke et al., 2019), there is a cross border between ‘explainable’ and ‘interpretability’. The explanation being able to ‘answer different kinds of question about the model’s operation depending on the stakeholder’’. ‘Interpretability’ focuses on making ‘stakeholders comprehend the main drivers of a model-driven decision’. Even though the awareness of interpretability increased, it is still unclear how the algorithms provide prediction and if interpretable methods can provide the sought information (Murdoch et al., 2019).

2.3. METHODS THAT BOOST BLACK-BOX MODELS' INTERPRETABILITY

Model agnostic methods are interpretation methods that interpret any ML model. The challenge of DL models, in general, is related to the very exhausting number of features, the complex interconnection between nodes, and unintelligible feature interpretations (Tsang et al., 2021). There are abundant research papers about interpretation methods for medicine in general. However, the majority of research does not require concern fraud detection interpretation methods. This research paper offers one contribution to apply iML in fraud detection using a reasonable model interpretation principle.

F. L. Chen and Li (2010), finance as a specific field and includes critical topics like credit scoring, which implies a massive data collection and classification effort. Using BB methods, financial institutions can optimize the prevision-making processes using intuitive experiences and quantifiable criteriums. The challenge comes from either training the financial institution to understand decision algorithms or, more scalable method, use interpretation methods to provide human-understandable interpretations.

Local model interpretation of BB models essentially concentrates the attention on the drivers that influenced a specific decision output. According to Guidotti et al. (2018), local model interpretation is generally more precise than global interpretations.
Winners of the 2016 Innovative Applications in Analytics Award, Ustun and Rudin (2016) determine that scoring approaches are less intuitive. Introducing Supersparse Linear Integer Models (SLIM), the authors define a different data contraction to boost scalability. In their research paper, Lei et al. (2016) prove that post-hoc explanations for the pre-trained model's interpretability do not respect the model's faithfulness and stability. They propose a new design of a self-learning model consisting of an end-to-end encoder and generator to create interpretable summaries and demonstrate that the new model reinforces faithfulness and model stability being a better interpreter for a complex model.

Studying the causes of a particular outcome, Štrumbelj, and Kononenko (2012) present that sensitivity analysis used in predictive methods can solve classification problems. Authors propose perturbing the inputs to acquire feature importance in interpretable processes and prove the algorithm with understandable human interpretations. The contribution is that the method performs even if the interpreted BB model is changed. Using image perturbation, Fong and Vedaldi, (2017) develop from a model initially designed to identify the parts of image classification a model agnostic method can interpret any kind of BB models.

So far, the research question of the studied papers debates on interpreting the drivers of an output. Another way of interpreting would be analyzing how the input must change so that the production changes. According to Tomei et al. (1999), using model debugging helps to go backward, changing the output. In the case of fraud detection, this implies if a particular decision-making process predicts fraud when it is just an unusual activity of the client, the financial institutions should change the rules to redefine the fraud action.

Contrary to local interpretation, global interpretable BB models understand the BB models as a whole and the logic of their behavior (Guidotti et al., 2018). Craven and Shavlik (1995) realize in their paper that the NN provides exhaustive interpretation inaccessible for human understanding. They develop a method called TREPAN to provide understandable symbolic features from trained NN. The technique acts identically as decision trees while using if-then and M-of-rules-based algorithms representing the neural network classifier with a high level of fidelity, providing accuracy and understanding.

A PDP is another interpretation method that provides the marginal effect of features on the output of the ML model. To do so, PDP calculates the averages over the training data – techniques also known as Monte Carlo simulation (Molnar, 2020):

$$\hat{f}_{x_s}(x_s) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}\,(x_S, x_C^{(i)})$$

Where S is the marginal effect of the feature and $x_C^{(i)}$ actual values of the features in the dataset, n is the number of instances, and C the data set (Molnar, 2020). The computation of the method is intuitive and easy to implement. It helps model users to check the model's trustworthiness and to have a rough explanation of how the model interprets inputs to produce its outcome. The drawback is that it plots only up to two feature importance, does not provide the user with the feature distribution, and can ignore heterogeneous effects (Molnar, 2020).

Local interpretable model-agnostic explanations (LIME) are interpretation methods that explain the BB ML model's predictions. We train LIME to produce approximative representations of the BB model. To understand what happens to the output of the BB ML model, LIME makes a new data set with permuted samples and outcomes. The method should produce accurate local predictions or, in other terms, local Fidel. Mathematically the methods account for a minimization problem (Molnar, 2020):

$$explanation\ (x) = \arg\min_{g\,\in G} L(f, g, \pi_x) + \Omega(g)$$

The interpretable model for sample x is the model g (any of the iML) that minimizes the loss L showing how close are the method results comparing to the initial model while generating a low complexity model $\Omega(g)$. The advantage of the method is that if the BB ML model changes, we can still use the LIME and consider different features compared to the original model. The biggest drawback is that the user must adjust the kernel for each application to generate logical interpretations. Therefore, we see that the interpretation method delivers varying interpretations (Molnar, 2020).

The feature interaction method is another interpretation method applicable in calculating the model's prediction error after permuting the samples (Fisher et al., 2018; Molnar, 2020). The technique produces intuitive interpretations. It is a concise interpretation of the model's behavior and is comparable across multiple problems. The disadvantage of the method is that it is a function of the errors of the BB model. The techniques work well with uncorrelated features. And can decrease the feature importance because the importance splits across them (Breiman, 2001; Molnar, 2020).

3. DATA

3.1. DATA DESCRIPTION

We used the data of the e-commerce transactions with fraud flag information and the other various types of features from Vesta Corporation, which is available on Kaggle (Kaggle, n.d.).

Vespa Corporation is a company that provides e-commerce transaction solutions, and the data they provide on Kaggle contains primarily two types of features. One is a numerical type, and the other is a categorical type. The information to identify if a transaction is fraud is provided with the value zero or one (zero means that the transaction is legit, and one means fraud). The number of the features is 434, including the fraud flag feature, and the sample size of the data is 590,540.

Figure 5. The values of each feature in a sample

feature	value	feature	value	feature	value	feature	value	feature	value	feature	value	feature	value	feature	value	feature	value	feature	value	feature	value
TransactionID	2987004	DeviceInfo	SAMSUNG SM-G892A Build/NRD90M	D10	0	V26	0	V66	0	V106	0	V146	0	V186	1	V226	0	V266	0	V306	0
id_01	0	isFraud	0	D11	0	V27	0	V67	0	V107	1	V147	0	V187	1	V227	0	V267	0	V307	0
id_02	70787	TransactionDT	86506	D12	0	V28	0	V68	0	V108	1	V148	0	V188	1	V228	1	V268	0	V308	0
id_03		TransactionAmt	50	D13	0	V29	0	V69	0	V109	1	V149	0	V189	1	V229	1	V269	0	V309	0
id_04		ProductCD	H	D14	0	V30	0	V70	0	V110	1	V150	1803	V190	1	V230	1	V270	0	V310	0
id_05		card1	4497	D15	0	V31	0	V71	0	V111	1	V151	49	V191	1	V231	0	V271	0	V311	0
id_06		card2	514	M1	0	V32	0	V72	0	V112	1	V152	64	V192	1	V232	0	V272	0	V312	0
id_07		card3	150	M2	0	V33	0	V73	0	V113	1	V153	0	V193	1	V233	0	V273	0	V313	0
id_08		card4	mastercard	M3	0	V34	0	V74	0	V114	1	V154	0	V194	1	V234	0	V274	0	V314	0
id_09		card5	102	M4	0	V35	0	V75	0	V115	1	V155	0	V195	1	V235	0	V275	0	V315	0
id_10		card6	credit	M5	0	V36	0	V76	0	V116	1	V156	0	V196	1	V236	0	V276	0	V316	0
id_11	100	addr1	420	M6	0	V37	0	V77	0	V117	1	V157	0	V197	1	V237	0	V277	0	V317	0
id_12	NotFound	addr2	87	M7	0	V38	0	V78	0	V118	1	V158	0	V198	1	V238	0	V278	0	V318	0
id_13		dist1	0	M8	0	V39	0	V79	0	V119	1	V159	15557.99	V199	1	V239	0	V279	0	V319	0
id_14	-480	dist2	0	M9	0	V40	0	V80	0	V120	1	V160	169690.8	V200	1	V240	1	V280	0	V320	0
id_15	New	P_emaildomain	gmail.com	V1	0	V41	0	V81	0	V121	1	V161	0	V201	1	V241	1	V281	0	V321	0
id_16	NotFound	R_emaildomain	0	V2	0	V42	0	V82	0	V122	1	V162	0	V202	0	V242	1	V282	1	V322	0
id_17	166	C1	1	V3	0	V43	0	V83	0	V123	1	V163	0	V203	0	V243	1	V283	1	V323	0
id_18		C2	1	V4	0	V44	0	V84	0	V124	1	V164	515	V204	0	V244	1	V284	0	V324	0
id_19	542	C3	0	V5	0	V45	0	V85	0	V125	1	V165	5155	V205	0	V245	1	V285	0	V325	0
id_20	144	C4	0	V6	0	V46	0	V86	0	V126	0	V166	2840	V206	0	V246	1	V286	0	V326	0
id_21		C5	0	V7	0	V47	0	V87	0	V127	0	V167	0	V207	0	V247	1	V287	0	V327	0
id_22		C6	1	V8	0	V48	0	V88	0	V128	0	V168	0	V208	0	V248	1	V288	0	V328	0
id_23		C7	0	V9	0	V49	0	V89	0	V129	0	V169	0	V209	0	V249	1	V289	0	V329	0
id_24		C8	1	V10	0	V50	0	V90	0	V130	0	V170	1	V210	0	V250	1	V290	1	V330	0
id_25		C9	0	V11	0	V51	0	V91	0	V131	0	V171	1	V211	0	V251	1	V291	1	V331	0
id_26		C10	1	V12	0	V52	0	V92	0	V132	0	V172	0	V212	0	V252	1	V292	1	V332	0
id_27		C11	1	V13	0	V53	0	V93	0	V133	0	V173	0	V213	0	V253	1	V293	0	V333	0
id_28	New	C12	0	V14	0	V54	0	V94	0	V134	0	V174	0	V214	0	V254	1	V294	0	V334	0
id_29	NotFound	C13	1	V15	0	V55	0	V95	0	V135	0	V175	0	V215	0	V255	1	V295	0	V335	0
id_30	Android 7.0	C14	1	V16	0	V56	0	V96	0	V136	0	V176	1	V216	0	V256	1	V296	0	V336	0
id_31	samsung browser 6.2	D1	0	V17	0	V57	0	V97	0	V137	0	V177	0	V217	0	V257	1	V297	0	V337	0
id_32	32	D2	0	V18	0	V58	0	V98	0	V138	0	V178	0	V218	0	V258	1	V298	0	V338	0
id_33	2220x1080	D3	0	V19	0	V59	0	V99	0	V139	0	V179	0	V219	0	V259	1	V299	0	V339	0
id_34	match_status:2	D4	0	V20	0	V60	0	V100	0	V140	0	V180	0	V220	0	V260	1	V300	0		
id_35	T	D5	0	V21	0	V61	0	V101	0	V141	0	V181	0	V221	1	V261	1	V301	0		
id_36	F	D6	0	V22	0	V62	0	V102	0	V142	0	V182	0	V222	1	V262	1	V302	1		
id_37	T	D7	0	V23	0	V63	0	V103	0	V143	6	V183	0	V223	0	V263	0	V303	1		
id_38	T	D8	0	V24	0	V64	0	V104	0	V144	18	V184	0	V224	0	V264	0	V304	1		
DeviceType	mobile	D9	0	V25	0	V65	0	V105	0	V145	140	V185	0	V225	0	V265	0	V305	1		

3.2. DATA PRE-PROCESSING

3.2.1. DUMMY VARIABLES

Some of the numerical features in each sample data contain no value. As we wanted to take this also like one feature of the sample, we created blank dummies of each feature and inserted 0 to each original feature value if it is empty. And we converted each categorical feature into dummy variables as well. After these data pre-processing, the number of the features increases from 434 to 3217, including the fraud flag feature.

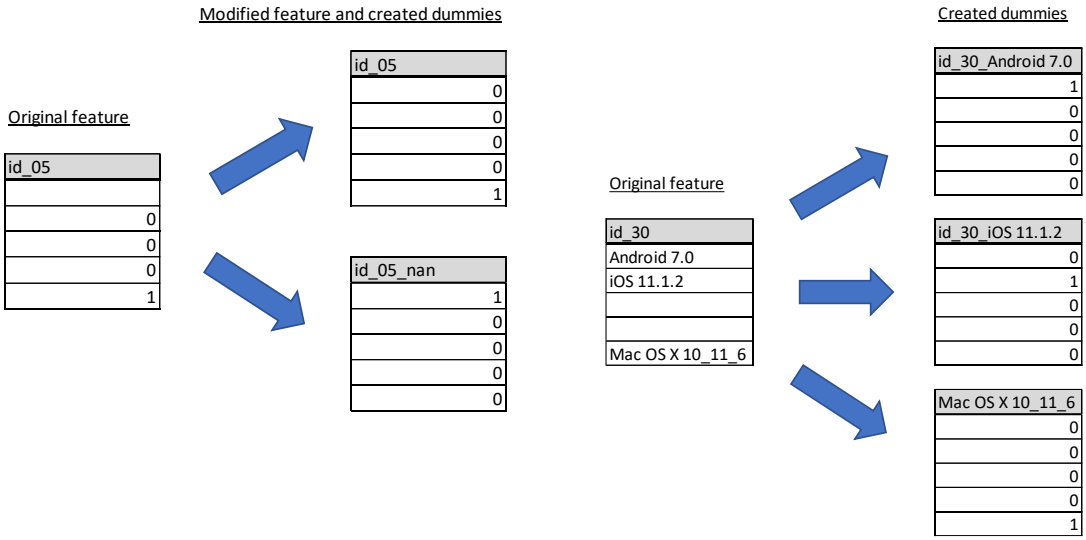

Figure 6. Image of data pre-processing: Creation of dummy variables

3.2.2. DATA SCALING

We used [0, 1] scaling for each feature. Note that, because it does not affect the distribution of the original feature values (Huang et al., 2015), this data pre-processing does not impact the optimization of the decision tree model nor logit model but neural network model.

3.2.3. DATA SPLITTING INTO A TRAINING DATASET AND TEST DATASET

When you create a prediction model, you have to prepare the dataset with which you train your model and prepare the dataset, with which you check the accuracy of the model prediction in out-sample. This checking procedure is called cross-validation. In our empirical analysis, we split the original data into a training dataset and test dataset, both of which have the same sample size. Therefore, the sample size of the training dataset and test dataset is 295,270 each.

3.2.4. OVERSAMPLING

If the sample size of one of the binary classes of the object feature is too small in the comparison of the other, it becomes difficult to get a good prediction model due to the mechanism of ML. The model tends to learn more from the class that contains many data. The issue is called the class imbalance problem.

Chawla et al. (2002) proposed oversampling techniques called Synthetic Minority Over-sampling Technique (SMOTE) to improve model accuracy. This method creates the unexisting synthetic sample from the actual samples in the data and boosts the sample size of the smaller class.

The fraud detection dataset is one of the datasets with the class imbalance problem (Fu et al., 2016; Maes et al., 1993). We applied SMOTE method to our data to boost the sample size of fraud data. The number of fraud transactions in the original training dataset is 10,288, and the proportion among the whole transactions is only about 3.5%. By applying SMOTE to our training dataset, we obtain a balanced dataset. The sample size of the fraud transaction increased up to the exact size of the legit transactions, enabling us to avoid the model overfitting to the initially larger class.

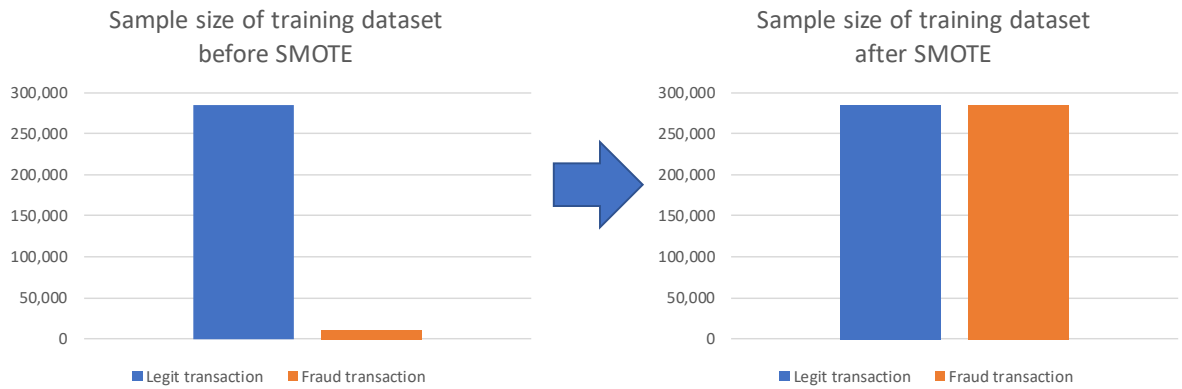

Figure 7. Image of data pre-processing: Sample size of training dataset before and after SMOTE operation

4. METHODOLOGY

This section explains the methodology of the ML models and the interpretation methods that we applied to one of the ML models, the NN model.

4.1. MACHINE LEARNING MODEL

We trained three types of machine learning models.

- Logit model
- Decision Tree
- Neural Network model

The first two models belong to the interpretable models, and the last one belongs to BB models. In the next section, we showed the prediction accuracy of each model, trained with the credit card fraud data, to show the BB model can outperform the interpretable models, giving us the incentive to use the BB model and implying the importance of the interpretation methods.

4.1.1. LOGIT MODEL

The logit model is a nonlinear parametric model used for binary classification (Molnar, 2020). The definition of the model is as follows:

$$P(y^i = 1) = \frac{1}{1 + exp(-(\beta_0 + \sum_{j=1}^{K} x_j^i \beta_j^i)}$$

Once we obtained the values of β by training this logit model with data, we can explicitly tell people how much the change of any input x_j^i affects the probability of$y^i = 1$ from the model.

4.1.2. DECISION TREE

A decision tree is a nonlinear non-parametric model for classification and regression (Molnar, 2020). We used the 'sklearn' package on python to implement this model. its mathematical description that we used for training the tree was as following (Scikit-learn, n.d.):

1. At node m, find a $\theta^* = (j, t_m)$ that satisfies the following equation:

$$\theta^* = argmin_\theta G(Q_m, \theta) = argmin_\theta \left\{ \frac{N_m^{left}}{N_m} H\left(Q_m^{left}(\theta)\right) + \frac{N_m^{right}}{N_m} H\left(Q_m^{right}(\theta)\right) \right\}$$

where

j : a feature

training vectors: $x_i \in R^n$

a label vector: $y \in R^l$

$$Q_m^{left}(\theta) = \{(x, y) | x_j \leq t_m\}$$

$$Q_m^{right}(\theta) = Q_m / Q_m^{left}(\theta)$$

Gini: $H(Q_m) = \sum_{k \in \{0,1\}} p_{mk}(1 - p_{mk})$

Predicted probability for the region: p_{mk}

2. Recurse step.1 for $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ until the m reaches the maximum allowable depth.

In the first node in Figure 8 ('V258' 0.02), a set of a feature and a figure was selected as $\theta^* = (j, t_m)$ in the above procedure since the ('V258' 0.02) was the possible argument that satisfies equation (*) in comparison with any other θ. The same process was operated on the second node and the right middle node each.

In our calculation, we set the maximum depth as three since the deep-depth decision tree interpretable (Molnar, 2020).

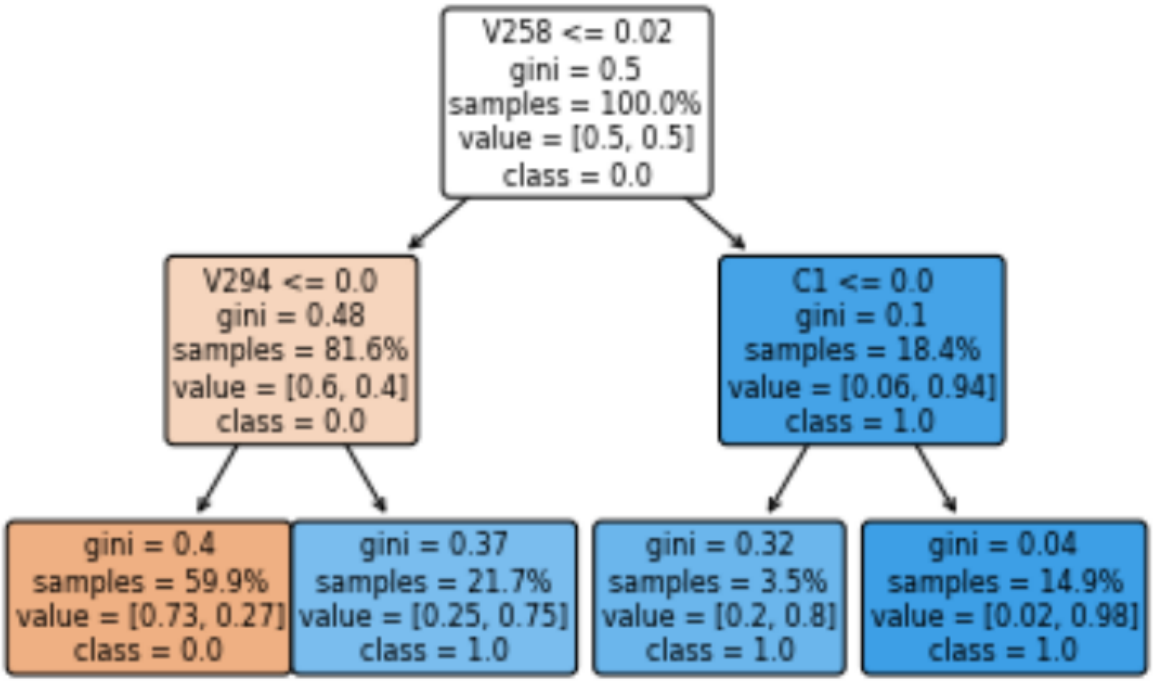

Figure 8. Image of the trained decision tree model

4.1.3. NEURAL NETWORK MODEL

The NN model is a nonlinear parametric model invented to imitate human brain structure (Rosenblatt, 1958). We used a 1D CNN model, a variant of the NN model, since our purpose to introduce this BB model is to show that the BB model can outperform simple interpretable models. Many researchers have reported its excellent performance in the last decade (Kiranyaz et al., 2021). The 2D CNN model (2d-CNN) is a famous and popular model used for 2-dimensional data classification, such as image recognition. The 1D CNN model (1d-CNN) is a model that has the same structure as 2d-CNN and applies to vector data, such as electrocardiogram data and bridge vibration data (Kiranyaz et al., 2021).

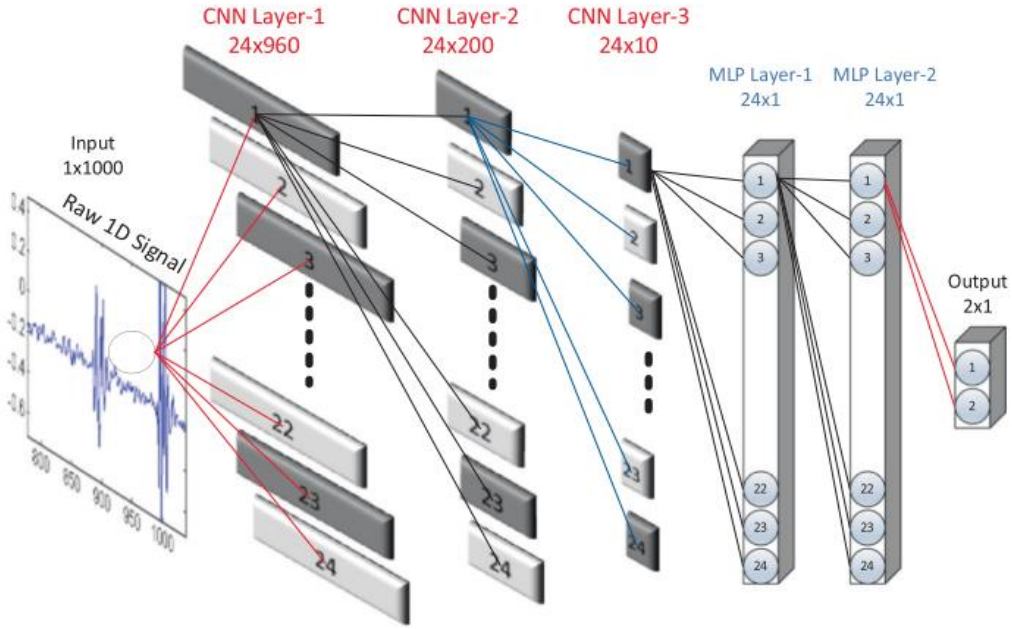

Figure 9. Image of 1D CNN model structure, retrieved from Kiranyaz et al. (2021), P.7

We used the 'TensorFlow package on python for the calculation. As shown in Figure 10, our model consists of one input layer, two 1-d CNN layers, and three dense layers. The activation function for the final layer was 'sigmoid', which is also called the logistic function.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 3178, 24)          984
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 3139, 24)          23064
_________________________________________________________________
flatten (Flatten)            (None, 75336)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              77145088
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_2 (Dense)              (None, 32)                16416
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 77,710,385
Trainable params: 77,710,385
Non-trainable params: 0
_________________________________________________________________
```

Figure 10. Summary of our NN model's structure

4.2. INTERPRETATION METHODS

We selected the following model-agnostic methods for our empirical application.

- Feature importance
- Partial dependence plot
- LIME (Local Interpretable model-agnostic Explanations)

4.2.1. FEATURE IMPORTANCE

Feature importance helps people to specify which feature contributes to the output in comparison with the others. Although the concept and the empirical calculation procedure of Feature importance were initially introduced by Breiman (2001), we constructed the Feature importance measure according to the algorism recommended by (Fisher et al., 2018) since our dataset was too big to follow the original procedure due to the computational limitation. The specific calculation procedure that we executed is as follows:

1. Estimate the original model error: $e^{original} = MSE\left(y, \hat{f}\left(X^{original}\right)\right)$
2. Split the training dataset in half and swap the value of feature s. (Generated datasets are X_1^{divide} and X_2^{divide} with its sample size = $\frac{n}{2}$ each)
3. Estimate the errors for the created datasets: $e_j^{divide} = MSE\left(y, \hat{f}\left(X_j^{divide}\right)\right)\ for\ j \in \{1, 2\}$
4. Calculate the feature importance of feature s : $FI^s = \left(\frac{e_1^{divide} + e_2^{divide}}{2}\right) \Big/ e^{original}$
5. Repeat 1 – 4 for each feature.
6. Sort features by descending FI.

where $\hat{f}$ is the trained model, $X^{original}$ is the original training dataset of the model's input and $MSE(\cdot)$ is the mean squared error function (Fisher et al., 2018; Molnar, 2020).

4.2.2. Partial dependence plot

A PDP helps people recognize how much the change of a feature input affects the output averagely and was introduced by Jerome H. Friedman (2001). The partial dependence function $\hat{f}_{x_s}(x_s)$ can be written as:

$$\hat{f}_{x_s}(x_s) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}\left(x_s, x_C^i\right)$$

where x_s is values of the features s, which you concern, and x_C^i is the values of all the other actual features C for sample i. n is the sample size of the training data (Molnar, 2020).

4.2.3. LIME (Local Interpretable model-agnostic Explanations)

LIME, introduced by Ribeiro et al. (2016), helps people understand how much each feature's values contributed to the output on individual transaction basis and model developers to detect models' possible defections. The general definition of LIME is given with the following optimization problem:

$$\underset{g \in G}{\operatorname{argmin}} L\big(f, g, \pi_x(z)\big) + \Omega(g)$$

where an explanation model of LIME is $g \in G$, where G is a class of interpretable models, such as linear models and decision trees, f is a trained original classification model, x is an instance that you want the explanation for, L is a loss function that measures the magnitude of the discrepancy of the predicted output between f and g, $\pi_x(z)$ is a distance measure between x and z, the LIME samples instances that are perturbed samples near the objective instance x, and $\Omega(\cdot)$ is a penalty function for the explanation model's complexity (Molnar, 2020; Ribeiro et al., 2016).

In our calculation we used the 'lime' package on python, and the specific definition of the optimization problem was as follows:

$$\underset{g \in G}{\operatorname{argmin}} L\big(f, g, \pi_x(z)\big) + \Omega(g)$$

where

$$g(x_i) = \sum_{j=1}^{K} x_{i,j}\, \beta_j$$

f: the trained CNN model

K: number of features

$$\pi_x(z_i) = exp\left(\frac{-\sqrt{\sum_{j=1}^{K}\big(x_{i,j} - z_{i,j}\big)^2}}{\sigma^2}\right)$$

$$\sigma = \sqrt{K}$$

$$\Omega(g) = \sqrt{\sum_{j=1}^{K} {\beta_j}^2}$$

$$L\big(f, g, \pi_x(z)\big) = \sum_{i=1}^{M} \pi_x(z_i)\big(f(z_i) - g(z_i)\big)^2$$

M: number of the perturbed samples z

5. RESULT

5.1. COMPARISON AMONG TWO INTERPRETABLE MODELS AND ONE BLACK-BOX MODEL

We applied two interpretable models and one BB model on the data to compare the prediction accuracy of each model. Therefore, we conclude that the BB models can perform better than the interpretable models in the sense of prediction accuracy. One of the interpretable models we used is the logit model, the classic statistical model for binary classification. And the other interpretable model is the decision-tree model with the shallow depth. In our analysis, the depth of the decision tree is three, as the deeper decision tree makes it complicated for humans to interpret the compensation with its prediction accuracy (Molnar, 2020). The BB model that we chose in this paper is the convolutional neural network model (CNN). It is one of the most popular BB models these days and has performed relatively better than others in the previous literature (Dastile et al., 2020).

We used two measures of the prediction accuracy for the model comparison, which are often used in many empirical papers: Percentage Correctly Classified (PCC) and G-Mean.

Logit model

training data

Act\Pred	Legit	Fraud
Legit	243753	41229
Fraud	61995	222987

test data

Act\Pred	Legit	Fraud
Legit	243119	41776
Fraud	2748	7627

Decision Tree

training data

Act\Pred	Legit	Fraud
Legit	237056	47926
Fraud	60664	224318

test data

Act\Pred	Legit	Fraud
Legit	237040	47855
Fraud	3661	6714

CNN model

training data

Act\Pred	Legit	Fraud
Legit	275845	9137
Fraud	2654	282328

test data

Act\Pred	Legit	Fraud
Legit	272974	11921
Fraud	3820	6555

Figure 11. Confusion matrix of the result from each model

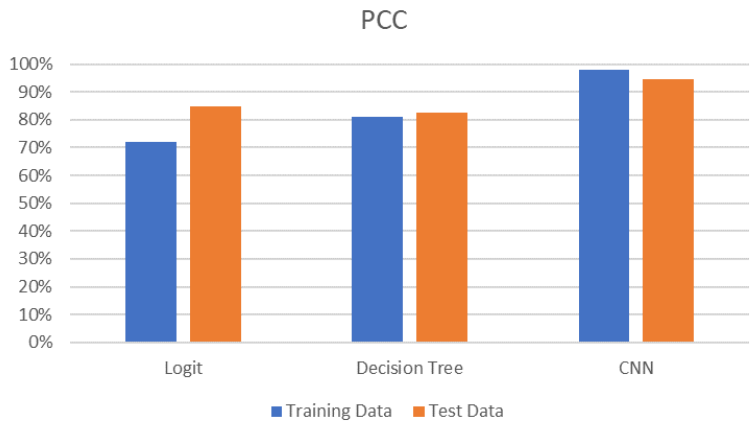

Figure 12. Comparison of PCC among two interpretable models and black-box model

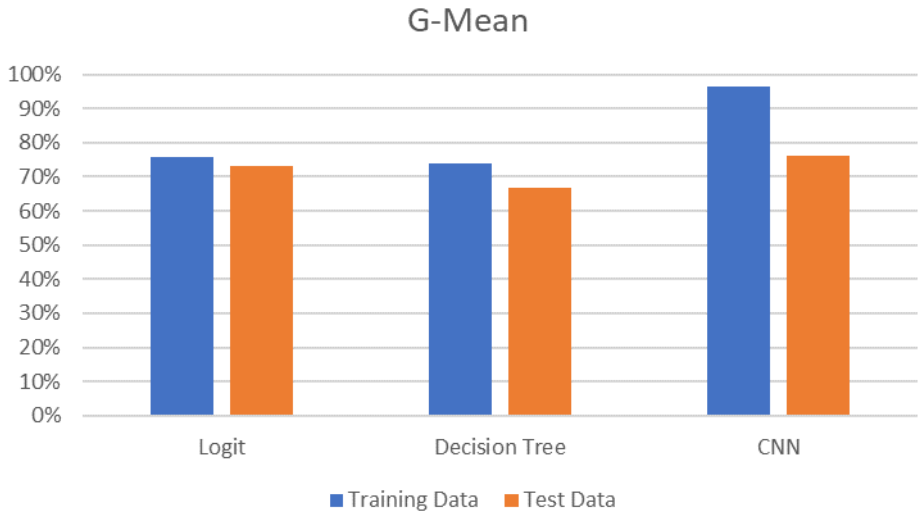

Figure 13.Comparison of G-Mean among two interpretable models and black-box model

As you see in Figure 12 and Figure 13, the CNN model predicted much more accurately than the two interpretable models both for the training dataset (in-sample) and the test dataset (out-of-sample) in the term of both PCC and G-Mean.

This result motives us to investigate methods that boost the interpretability of the BB models.

5.2. APPLICATION OF INTERPRETATION METHODS ON BLACK-BOX MODEL

5.2.1. FEATURE IMPORTANCE

As the number of features is too many, we show the feature importance only of the features that have 'C' in the first letter of their feature name ('C' features) among our data. According to Figure 14, 5, and 9, the results are outstanding compared to other 'C' features, implying that these two features' inputs relatively strongly affect the judge if the transaction is fraud.

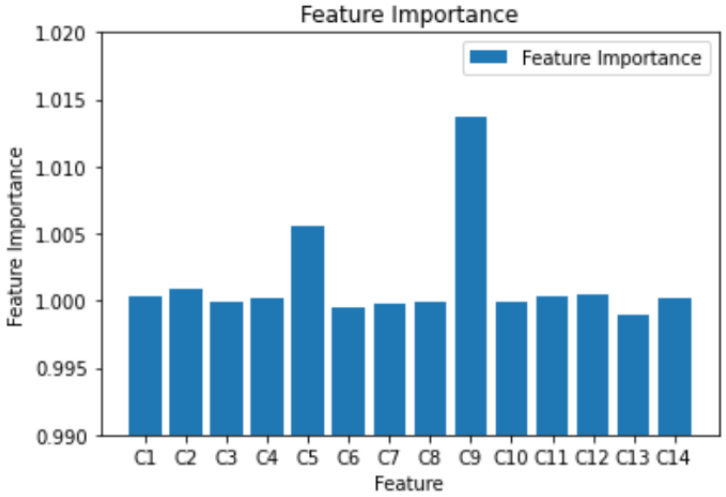

Figure 14. Feature Importance of the trained CNN

5.2.2. PARTIAL DEPENDENCE PLOT

Because of the same reason as in the previous section, we show the partial dependence plot only of the 'C' feature. Figure 15 shows that most of them show a linear relationship between the input and the output. Some of them show a nonlinear relationship between the input and the outcome. And the range of the average results varies among 'C' features. For example, the range of the average outputs of 'C1' is about 0.005, and the one of 'C9' is about 0.02, 4 times larger than 'C1'.

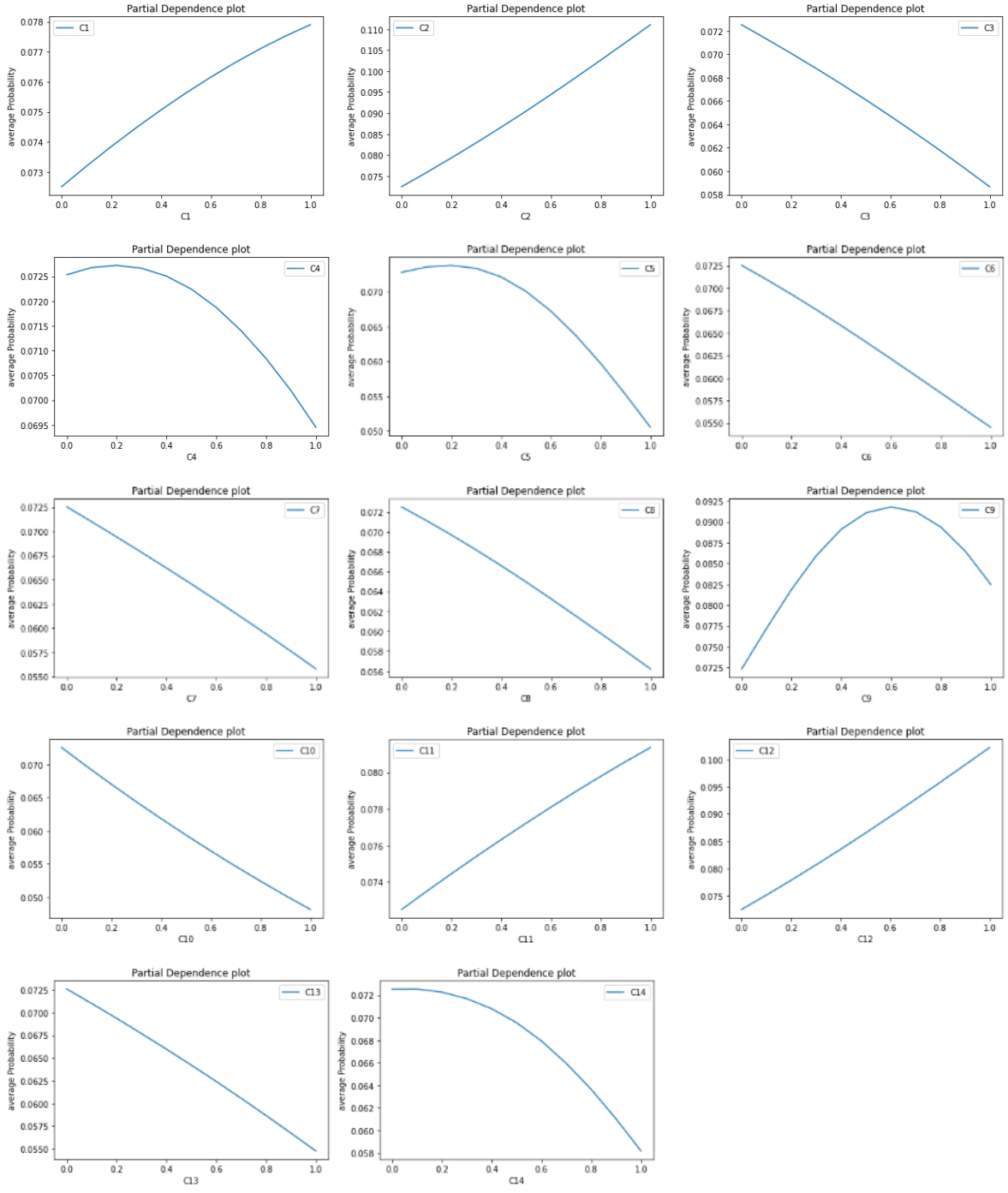

Figure 15. Graphs of Partial dependence plot

5.2.3. LIME

Figure 16 shows a LIME explanation for a sample that was predicted as a fraud transaction by the CNN model. The graph of the local explanation implies that the 'DeviceInfo_BLADE L7 Build/MRA58 K' feature affected the model's judge the most. To check if the LIME explanation is helpful to correct, we made a fake sample that has the same input figures for all the features with the sample used in Figure 16 but for the 'DeviceInfo_BLADE L7 Build/MRA58K' feature, which we changed from 1 to 0, and we inputted the fake sample into the CNN model. The calculated probability of the fraud transaction was about 87%, which was lower than the probability calculated with the original sample, 99.9%. The output got improved to some extent but not enough since it was still over 50%, and there was no space to improve from the implication of this LIME. Therefore, we rebuilt LIME with ten features, five more features than the LIME in Figure 17, to expand possible space to manipulate the output.

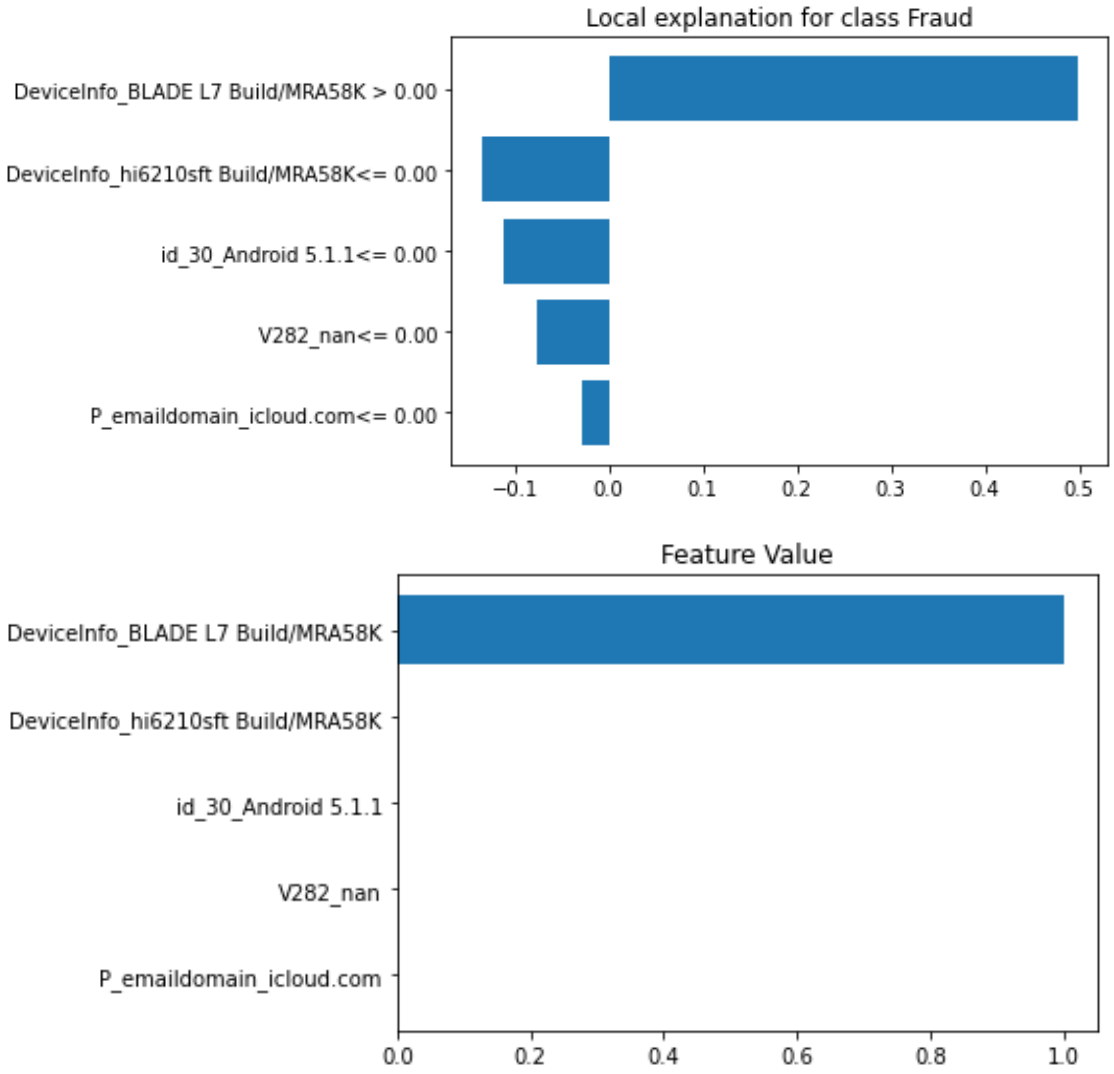

Figure 16. LIME explanation for a sample that was predicted as a fraud transaction by the CNN model

The local explanation of the LIME in Figure 17 implies that the customer of the sample transaction can improve the model's prediction by changing the sample's inputs of 'DeviceInfo_BLADE L7 Build/MRA58K' and Deviceinfo_R831L'. Thus, we made another fake sample that has the same input figures for all the features with the sample used in Figure 16 but for the 'DeviceInfo_BLADE L7 Build/MRA58K' feature, which we changed from 1 to 0, and 'Deviceinfo_R831L', which we changed from 0 to 1, and we inputted the new fake sample into the CNN model. The calculated probability of the fraud transaction was about 1%, which was way lower than 50%, and the transaction was judged as legit.

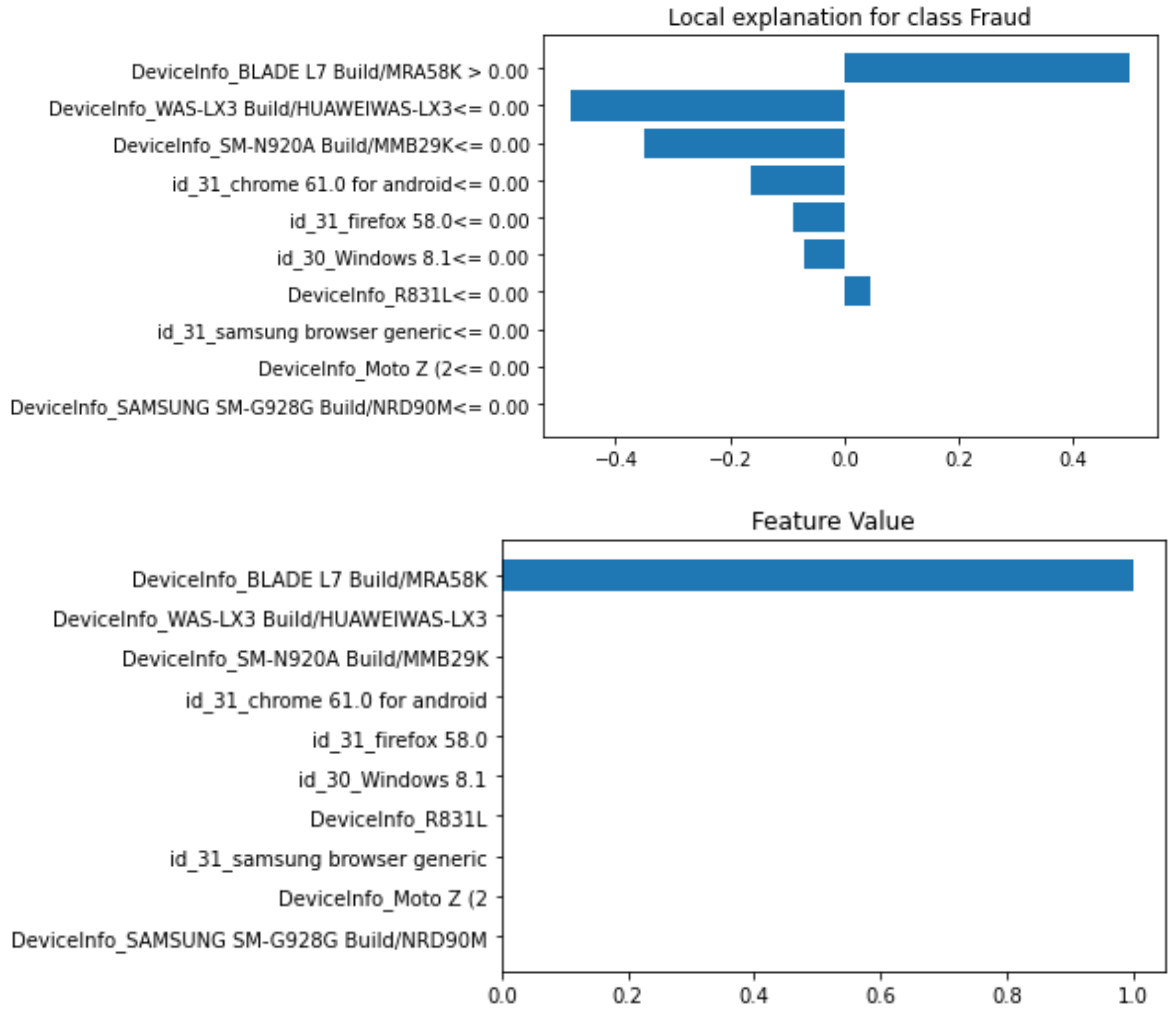

Figure 17. LIME explanation with ten features for a sample that was predicted as a fraud transaction by the CNN model

6. DISCUSSIONS

The need to understand why a specific prediction is made becomes more critical than the actual result as highly regulated, in our case, financial institutions address self-learning algorithms to create a crucial decision that affects individuals and society overall (Rudin, 2019).

Comparing the three models shows a good example that motivates credit card providers to select BB models for their credit card fraud detection. On the other hand, as discussed in the literature review section, choosing the BB model leads people to the challenge in its interpretability.

In the context of credit card fraud detection, the required interpretability differs between the credit card providers and the customers. For credit card providers, the information to judge the model validity is essential. Feature importance is helpful to recognize the order of the magnitudes of the impacts of the features on the output. This method helps the providers specify which feature the model considers essential. They can use the information to compare with their experience-based understanding, enabling them to check if the model is trustworthy.
The PDP gives different information: the relationship between a feature and the model's output. This method helps the providers recognize how much a change of input of a feature affects the outcome averagely among all samples. The providers can use the implication to judge if it matches their understanding, enabling them to assess the model's trustworthiness.
LIME gives very little information for the providers because local interpretability for one specific sample gained from LIME is not enough information to judge the whole model's trustworthiness except if they can improve the model only around the local feature input space.

On the other hand, for the customers, the information to interpret the cause of the output is essential. Feature importance is not so much helpful for the customers compared to the providers since feature importance gives only a list of the possible causal features. Unlike feature importance, the PDP gives very informative implications for the customers. The PDP enables the customers to guess how they can improve their customer profile to accept their transactions as legit one’s next time if the model detects the transactions as a fraud despite being not a fraud. And LIME is the most potent method among the methods we applied in this paper since, with LIME, the providers can provide the concrete reason for the model's prediction to their customers.

However, as you might think, you need to pay attention when you provide the information to your customers using these methods since there may be cases that the real imposters refer to deceit in the model.

7. CONCLUSIONS

In this paper, firstly, we trained two interpretable models, the Logit model and Decision Tree with its shallow depth, and one BB model, the NN model, using a credit card fraud detection dataset from Kaggle. Based on our case study, we showed an example that the BB model outperforms the other two interpretable models in the term of both PCC and G-Means. This result implies the potential advantage of applying the BB model in practice.

Secondly, we provided a use-case of 3 interpretation methods, Feature importance, Partial Dependence plot, and LIME, applying to the trained NN model, and showed that interpretation methods could increase the BB model's interpretability. We discussed the result of this use-case and found that Feature importance and PDP can be a good indicator for the model users for their business to check the model's trustworthiness. The PDP and LIME can provide good insights for the model users as the company's customers to infer the reason for the model outcome, the cause of the decline of the credit card transaction in our paper's context.

On the other hand, in this paper, we could not obtain reasonable implication that clears up the claims that come from the ethical side, like the prohibition of the usage of the personal information for the model predictions and to find suitable interpretation methods for this matter is one the of essential subjects for future research.

8. REFERENCES

Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. Proceedings of the IEEE International Conference on Computing, Networking and Informatics, ICCNI 2017, 2017-Janua, 1–9. https://doi.org/10.1109/ICCNI.2017.8123782

Abdul, A., Vermeulen, J., Wang, D., Lim, B. Y., & Kankanhalli, M. (2018). Trends and trajectories for explainable, accountable and intelligible systems: An HCI research agenda. *Conference on Human Factors in Computing Systems - Proceedings*, *2018-April*, 1–18. https://doi.org/10.1145/3173574.3174156

Adadi, A., & Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, *6*, 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). Machine Bias. ProPublica, May 23, 1–16.

Barlow, H. B. (1989). Unsupervised Learning. *Neural Computation*, *1*(3), 295–311. https://doi.org/10.1162/neco.1989.1.3.295

Bracke, P., Datta, A., Jung, C., & Sen, S. (2019). Machine Learning Explainability in Finance: An Application to Default Risk Analysis. *SSRN Electronic Journal*, *816*. https://doi.org/10.2139/ssrn.3435104

Breiman, L. (1998). Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, *26*(3), 801–849.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Buetti-Dinh, A., Galli, V., Bellenberg, S., Ilie, O., Herold, M., Christel, S., Boretska, M., Pivkin, I. V., Wilmes, P., Sand, W., Vera, M., & Dopson, M. (2019). Deep neural networks outperform human expert's capacity in characterizing bioleaching bacterial biofilm composition. *Biotechnology Reports*, *22*, e00321. https://doi.org/10.1016/j.btre.2019.e00321

Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics (Switzerland)*, *8*(8), 1–34. https://doi.org/10.3390/electronics8080832

Chai, J., Cho, S., & Huang, J. (2020). Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China*, *14*(1). https://doi.org/10.1186/s11782-020-00082-6

Chai, J., Liu, J. N. K., & Ngai, E. W. T. (2013). Application of decision-making techniques in supplier selection: A systematic review of literature. *Expert Systems with Applications*, *40*(10), 3872–3885. https://doi.org/10.1016/j.eswa.2012.12.040

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953

Chen, C. T., Chen, A. P., & Huang, S. H. (2018). Cloning strategies from trading records using an agent-based reinforcement learning algorithm. *Proceedings - 2018 IEEE International Conference on Agents, ICA 2018*, 34–37. https://doi.org/10.1109/AGENTS.2018.8460078

Chen, F. L., & Li, F. C. (2010). Combination of feature selection approaches with SVM in credit scoring. *Expert Systems with Applications*, *37*(7), 4902–4909. https://doi.org/10.1016/j.eswa.2009.12.025

Chuang, C. L., & Lin, R. H. (2009). Constructing a reassigning credit scoring model. *Expert Systems with Applications*, *36*(2 PART 1), 1685–1694. https://doi.org/10.1016/j.eswa.2007.11.067

Coeckelbergh, M. (2020). *Introduction to philosophy of education*. https://doi.org/10.1007/BF00368029

Craven, M., & Shavlik, J. (1995). Extracting tree-structured representations of trained networks. Advances in neural information processing systems, 8, 24-30.

Crook, J. (1999). Who is discouraged from applying for credit? *Economics Letters*, *65*(2), 165–172. https://doi.org/10.1016/s0165-1765(99)00151-2

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(8), 3784–3797. https://doi.org/10.1109/TNNLS.2017.2736643

Dastile, X., Celik, T., & Potsane, M. (2020). Statistical and machine learning models in credit scoring: A systematic literature survey. *Applied Soft Computing Journal*, *91*. https://doi.org/10.1016/j.asoc.2020.106263

Dayan, P. (2008). Unsupervised Learning. *Encyclopedia of Neuroscience*, 4154–4154. https://doi.org/10.1007/978-3-540-29678-2_6202

Donnelly, C., & Embrechts, P. (2010). The Devil is in the Tails: Actuarial Mathematics and the Subprime Mortgage Crisis. *ASTIN Bulletin*, *40*(1), 1–33. https://doi.org/10.2143/ast.40.1.2049222

Doran, D., Schulz, S., & Besold, T. R. (2018). What does explainable AI really mean? A new conceptualization of perspectives. *CEUR Workshop Proceedings*, *2071*.

Doshi-Velez, F., & Kim, B. (2017). *Towards A Rigorous Science of Interpretable Machine Learning*. *Ml*, 1–13. http://arxiv.org/abs/1702.08608

EU. (2016). 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 Da. *American Fuel and Petrochemical Manufacturers, AFPM - Labor Relations/Human Resources Conference 2018*, *27 April 2016*, 45–62. https://doi.org/10.1308/rcsfdj.2018.54

EU. (2019a). *30 Recommendations on regulation, innovation and finance*. *December*, 1–100.

EU. (2019b). Ethics Guidelines for Trustworthy AI. 1(8 April), 1–41. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

Felzmann, H., Fosch-Villaronga, E., Lutz, C., & Tamò-Larrieux, A. (2020). Towards Transparency by Design for Artificial Intelligence. *Science and Engineering Ethics*, *26*(6), 3333–3361. https://doi.org/10.1007/s11948-020-00276-4

Fisher, A., Rudin, C., & Dominici, F. (2018). *Model Class Reliance: Variable Importance Measures for any Machine Learning Model Class, from the "Rashomon" Perspective*.

Fong, R. C., & Vedaldi, A. (2017). Interpretable Explanations of Black Boxes by Meaningful Perturbation. *IEEE International Conference on Computer Vision (ICCV), 2017*, 3449–3457. https://doi.org/10.1109/ICCV.2017.371

Freeman, A. (2017). Racism in the Credit Card Industry. In *L. Rev* (Vol. 95).

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Fu, K., Cheng, D., Tu, Y., & Zhang, L. (2016). Credit card fraud detection using convolutional neural networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9949 LNCS*, 483–490. https://doi.org/10.1007/978-3-319-46675-0_53

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. *ArXiv*.

Guidotti, R., Monreale, A., Ruggieri, S., & Turini, F. (2018). *A Survey of Methods for Explaining Black Box Models A Survey of Methods for Explaining Black Box Models 1 INTRODUCTION*. *May*, 1–41.

Hand, D. J., & Till, R. J. (2001). A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine Learning*, *45*(2), 171–186. https://doi.org/10.1023/A:1010920819831

Hastie, T., Tibshirani, R., & Friedman, J. (2013). The Elements of Statistical Learning. Data Mining, Inference, and Prediction. In *Springer*. https://doi.org/10.1101/328864

Hill, R. K. (2016). What an Algorithm Is. *Philosophy and Technology*, *29*(1), 35–59. https://doi.org/10.1007/s13347-014-0184-5

Huang, J., Li, Y. F., & Xie, M. (2015). An empirical analysis of data preprocessing for

machine learning-based software cost estimation. Information and Software Technology, 67, 108–127. https://doi.org/10.1016/j.infsof.2015.07.004

Hu, Y. J., Ku, T. H., Jan, R. H., Wang, K., Tseng, Y. C., & Yang, S. F. (2012). Decision tree-based learning to predict patient controlled analgesia consumption and readjustment. *BMC Medical Informatics and Decision Making*, *12*(1), 131. https://doi.org/10.1186/1472-6947-12-131

Jeong, G., & Kim, H. Y. (2019). Improving financial trading decisions using deep Q-learning: Predicting the number of Shares, action Strategies, and transfer learning. *Expert Systems with Applications*, *117*, 125–138. https://doi.org/10.1016/j.eswa.2018.09.036

Jerome H. Friedman. (2001). Greedy Function Approximation: A Gradient Boosting Machine on JSTOR. *Annals of Statistics*, *29*(5), 1189–1232.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260. https://doi.org/10.1126/science.aaa8415

Kaggle. (n.d.). IEEE-CIS Fraud Detection Can you detect fraud from customer transactions? Retrieved April 17, 2021, from https://www.kaggle.com/c/ieee-fraud-detection/

Keil, F. C., Rozenblit, L., & Mills, C. M. (2004). Thinking and Seeing: Visual Metacognition in Adults and Children. In *Thinking and Seeing: Visual metacognition in adults and children*.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, *151*. https://doi.org/10.1016/j.ymssp.2020.107398

Kokkinaki, A. I. (1997). On atypical database transactions: Identification of probable frauds using machine learning for user profiling. *Proceedings of the IEEE Knowledge & Data Engineering Exchange Workshop, KDEX*, 107–113. https://doi.org/10.1109/kdex.1997.629848

Kubat, M., & Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. *Icml*, *97*, 179–186.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 1675–1684. https://doi.org/10.1145/2939672.2939874

Lei, T., Barzilay, R., & Jaakkola, T. (2016). Rationalizing neural predictions. *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 107–117. https://doi.org/10.18653/v1/d16-1011

Letham, B., Rudin, C., McCormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, *9*(3), 1350–1371. https://doi.org/10.1214/15-AOAS848

Ling, C. X., Huang, J., & Zhang, H. (2003). AUC: A better measure than accuracy in comparing learning algorithms. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2671*, 329–341. https://doi.org/10.1007/3-540-44886-1_25

Lipton, Z. C. (2016). The mythos of model interpretability. *Communications of the ACM*. https://doi.org/10.1145/3233231

Lords, U. G. H. of. (2019). *AI in the UK: Ready, Willing and Able? 2017*.

Maes, S., Maes, S., Tuyls, K., Vanschoenwinkel, B., & Manderick, B. (1993). Credit Card Fraud Detection Using Bayesian and Neural Networks. *IN: MACIUNAS RJ, EDITOR. INTERACTIVE IMAGE-GUIDED NEUROSURGERY. AMERICAN ASSOCIATION NEUROLOGICAL SURGEONS*, 261--270.

Martínez-Miranda, E., McBurney, P., & Howard, M. J. W. (2016). Learning unfair trading: A market manipulation analysis from the reinforcement learning perspective. *Proceedings of the 2016 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS 2016*, 103–109. https://doi.org/10.1109/EAIS.2016.7502499

Matthias, A. (2004). The responsibility gap: Ascribing responsibility for the actions of learning automata. *Ethics and Information Technology*, *6*(3), 175–183. https://doi.org/10.1007/s10676-004-3422-1

Michail, Z., Joseph, P. Z., & Ronald, E. M. (1997). *From Instability to Intelligence - Complexity and Predictability in Nonlinear Dynamics*.

Mitchell, T. M. (2006). The Discipline of Machine Learning. *Machine Learning*, *17*(July), 1–7. http://www-cgi.cs.cmu.edu/~tom/pubs/MachineLearningTR.pdf

Mittelstadt, B. D., Allo, P., Taddeo, M., Wachter, S., & Floridi, L. (2016). The ethics of algorithms: Mapping the debate. *Big Data and Society*, *3*(2), 1–21. https://doi.org/10.1177/2053951716679679

Molnar, C. (2020). *Interpretable machine learning*. https://christophm.github.io/interpretable-ml-book/

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences of the United States of America*, *116*(44). https://doi.org/10.1073/pnas.1900654116

Neagoe, V.-E., Ciotec, A.-D., & Cucu, G.-S. (2018). *Deep Convolutional Neural Networks Versus Multilayer Perceptron for Financial Prediction*. 201–206. https://doi.org/10.1109/iccomm.2018.8484751

Nichols, A., & Schwabish, J. (2014). Effects of a Higher Minimum Wage in the District of Columbia. Urban Institute, 1, 1–16.

O'Neil, C. (2016). *More Advance Praise for Stroke*. Crown/Archetype.

Phua, C., Alahakoon, D., & Lee, V. (2004). Minority report in fraud detection. *ACM SIGKDD Explorations Newsletter*, *6*(1), 50–59. https://doi.org/10.1145/1007730.1007738

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 1135–1144. https://doi.org/10.1145/2939672.2939778

Rosenblatt, F. (1958). THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN 1. In *Psychological Review* (Vol. 65, Issue 6).

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x

Scikit-learn. (n.d.). *1.10. Decision Trees — scikit-learn 0.24.2 documentation*. Retrieved May 7, 2021, from https://scikit-learn.org/stable/modules/tree.html#tree

Sharma, R., Sharma, K., & Apyrva, K. (2020). Study of Supervised Learning and Unsupervised Learning. *International Journal for Research in Applied Science and Engineering Technology*, *8*(6), 588–593. https://doi.org/10.22214/ijraset.2020.6095

Simon, A., Deo, M. S., Venkatesan, S., & Babu, D. R. R. (2015). An Overview of Machine Learning. *International Journal of Electrical Sciences & Engineering (IJESE)*, *1*(January), 22–24. https://doi.org/10.1007/978-1-4842-3916-2_1

Štrumbelj, E., & Kononenko, I. (2012). *Explaining prediction models and individual predictions with feature contributions*. 647–665. https://doi.org/10.1007/s10115-013-0679-x

Su, G., Wei, D., Varshney, K. R., & Malioutov, D. M. (2016). *Interpretable Two-level Boolean Rule Learning for Classification*. *August*. http://arxiv.org/abs/1606.05798

Tomczak, J. M., & Zie˛ba, M. (2015). Classification Restricted Boltzmann Machine for comprehensible credit scoring model. *Expert Systems with Applications*, *42*(4), 1789–1796. https://doi.org/10.1016/j.eswa.2014.10.016

Tomei, L., Roussel, A., Incitti, I., Vitale, R. L., Mathieu, M., & Francesco, R. De. (1999). Crystal structure of the RNA-dependent RNA polymerase of hepatitis C virus. PNAS, 96(23), 1–6.

Trinkle, B. S., & Baldwin, A. A. (2007). Interpretable credit model development via artificial neural networks. *Intelligent Systems in Accounting, Finance & Management*, *15*(3–4), 123–147. https://doi.org/10.1002/isaf.289

Tsang, M., Enouen, J., & Liu, Y. (2021). *Interpretable Artificial Intelligence through the Lens of Feature Interaction*. 1–16. http://arxiv.org/abs/2103.03103

Turing, A. M. (1950). COMPUTING MACHINERY AND INTELLIGENCE. *A QUARTERLY*

REVIEW OF PSYCHOLOGY AND PHILOSOPHY, 433–460.

Ustun, B., & Rudin, C. (2016). Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, *102*(3), 349–391. https://doi.org/10.1007/s10994-015-5528-6

Varshney, K. R., & Alemzadeh, H. (2017). On the Safety of Machine Learning: Cyber-Physical Systems, Decision Sciences, and Data Products. *Big Data*, *5*(3), 246–255. https://doi.org/10.1089/big.2016.0051

West, D. (2000). Neural network credit scoring models. *Computers and Operations Research*, *27*(11–12), 1131–1152. https://doi.org/10.1016/S0305-0548(99)00149-5

Yan, H., & Ouyang, H. (2018). Financial Time Series Prediction Based on Deep Learning. *Wireless Personal Communications*, *102*(2), 683–700. https://doi.org/10.1007/s11277-017-5086-2

Yin, X., & Han, J. (2003). *CPAR: Classification based on Predictive Association Rules*. 331–335. https://doi.org/10.1137/1.9781611972733.40

Zhang, J., & Maringer, D. (2016). Using a Genetic Algorithm to Improve Recurrent Reinforcement Learning for Equity Trading. *Computational Economics*, *47*(4), 551–567. https://doi.org/10.1007/s10614-015-9490-y

Zinke, C., Anke, J., Meyer, K., & Schmidt, J. (2018). Modeling, analysis and control of personal data to ensure data privacy – A use case driven approach. *Advances in Intelligent Systems and Computing*, *593*(October 2017), 87–96. https://doi.org/10.1007/978-3-319-60585-2_10

APPENDIX

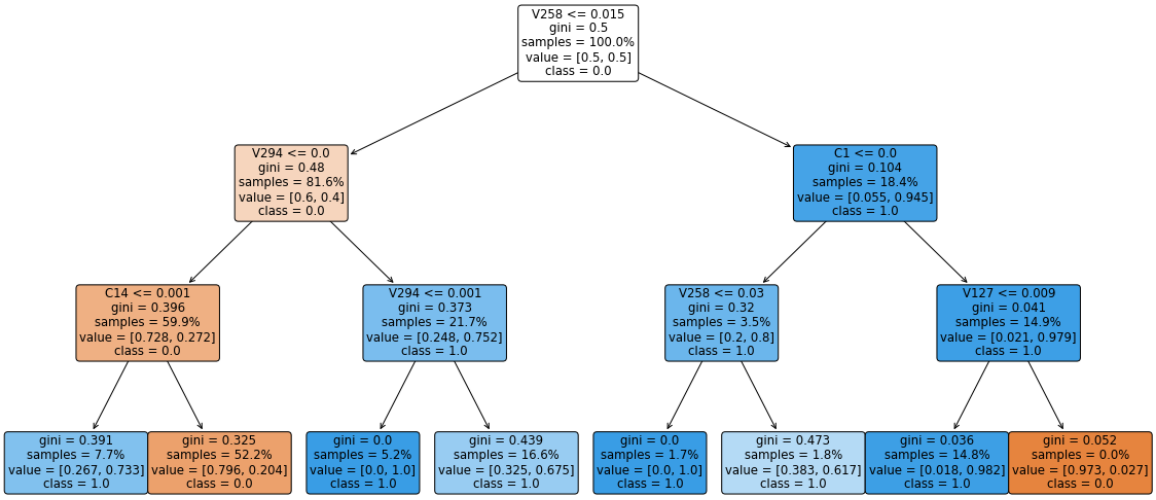

Figure 18. (Appendix 1). The trained decision tree model with depth = 3

```
# -*- coding: utf-8 -*-
import pandas as pd
import os
dir_data='***'
os.chdir(dir_data)

# ----------------------------------------------
# --------merge and save original data in pickle
# ----------------------------------------------
df=pd.read_csv('data/ieee-fraud-detection/train_identity.csv')
tmp=pd.read_csv('data/ieee-fraud-detection/train_transaction.csv')
df=pd.merge(df,tmp,on='TransactionID',how='outer')
df.to_pickle('input/fraud-detection_rawdata.pkl')
del df
del tmp

# ----------------------------------------------
# --------make data set for DL
# ----------------------------------------------
```

```
import numpy as np
df=pd.read_pickle('input/fraud-detection_rawdata.pkl')

# ----------- create nan_feature from each interval & ratio variables and insert 0 to each cells with nan
# specify the columns
col_Float_with_nan=[]
for col in df.columns:
    if 'nan' in np.unique(df[col].astype(str)) and df[[col]].dtypes[0]!=object:
        col_Float_with_nan.append(col)

i=1
total=len(col_Float_with_nan)
for col in col_Float_with_nan:
    # create and combine nan feature dummy
    dummy=[1 if df[col][i].astype(str)=='nan' else 0 for i in df.index]
    df[col + '_nan']=dummy
    # replace nan with 0
    df[col]=[0 if df[col][i].astype(str)=='nan' else df[col][i] for i in df.index]
    print(str(i)+"/"+str(total))
    i+=1



# ----------- convert nominal variable into dummy variables
col_notFloat=df.columns[df.dtypes==object]



for col in col_notFloat:
    dummy=pd.get_dummies(df[[col]])
    if 'nan' in np.unique(df[col].astype(str)):
        df=pd.concat([df,dummy],axis=1)
    else:
        dummy=dummy.iloc[:,:-1]
        df=pd.concat([df,dummy],axis=1)

df=df.drop(columns=col_notFloat)
```

```
df.to_pickle('input/fraud-detection.pkl')
del df,col,dummy,col_notFloat

# ----------------------------------------------
# --------data Scaling
# ----------------------------------------------
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import numpy as np

df=pd.read_pickle('input/fraud-detection.pkl')
tmp1=df.iloc[:,:round(df.shape[1]/2)]
tmp2=df.iloc[:,round(df.shape[1]/2):]
del df
for col in tmp1.columns:
    tmp1[col] = MinMaxScaler().fit_transform(tmp1[col].values.reshape(-1,1))
for col in tmp2.columns:
    tmp2[col] = MinMaxScaler().fit_transform(tmp2[col].values.reshape(-1,1))

df=pd.concat([tmp1,tmp2],axis=1)
del tmp1, tmp2

# push down data size by converting dtype from float64 to float16
# and save this file
df.astype(np.float16).to_pickle('input/fraud-detection_transformed.pkl')
del df,col

#-----------------------------------
#---------make train & test data with SMOTEing train data
#-----------------------------------
from sklearn.model_selection import train_test_split # splitting the data
df=pd.read_pickle('input/fraud-detection_transformed.pkl')

y_col='isFraud'
y_var=df[y_col]
X_var=df.loc[:, df.columns != y_col]
```

```
del df,y_col

# split data into training data set and test data set
X_train, X_test, y_train, y_test = train_test_split(X_var, y_var, test_size = 0.5, random_state = 0)
del y_var, X_var

X_train.to_pickle('input/fraud-detection_transformed_X_train.pkl')
X_test.to_pickle('input/fraud-detection_transformed_X_test.pkl')
y_train.to_pickle('input/fraud-detection_transformed_y_train.pkl')
y_test.to_pickle('input/fraud-detection_transformed_y_test.pkl')
del X_train,y_train,X_test,y_test



# use smote to increase fraud case sample size
import numpy as np
from imblearn.over_sampling import SMOTE
X_train=pd.read_pickle('input/fraud-detection_transformed_X_train.pkl')
y_train=pd.read_pickle('input/fraud-detection_transformed_y_train.pkl')
X_train_smote, y_train_smote = SMOTE().fit_resample(X_train, y_train)
del X_train,y_train
X_train_smote.to_pickle('input/fraud-detection_transformed_X_train_smote.pkl')
y_train_smote.to_pickle('input/fraud-detection_transformed_y_train_smote.pkl')
del X_train_smote,y_train_smote
```

Figure 19. (Appendix 2). Python code for Data Processing

```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 10 15:45:25 2021

@author: n2ngo
"""
import pandas as pd
import os
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import plot_confusion_matrix
```

```
from termcolor import colored as cl # text customization
from sklearn.tree import DecisionTreeClassifier as dtc # tree algorithm
from sklearn.metrics import accuracy_score # model precision
from sklearn.tree import plot_tree # tree diagram

dir_data='***'
os.chdir(dir_data)

# ----------------------------------------------
# --------Calc with logit model
# ----------------------------------------------

X_train=pd.read_pickle('input/fraud-detection_transformed_X_train_smote.pkl')
y_train=pd.read_pickle('input/fraud-detection_transformed_y_train_smote.pkl')

model = LogisticRegression(solver='liblinear', random_state=0)
model.fit(X_train, y_train) # matrix is too big to calculate
plot_confusion_matrix(model,X_train, y_train)
del X_train,y_train



X_test=pd.read_pickle('input/fraud-detection_transformed_X_test.pkl')
y_test=pd.read_pickle('input/fraud-detection_transformed_y_test.pkl')
plot_confusion_matrix(model,X_test, y_test)

del y_test, X_test,  model

# ----------------------------------------------
# --------Calc with Desicion tree model
# ----------------------------------------------
depth_DT=3
X_train=pd.read_pickle('input/fraud-detection_transformed_X_train_smote.pkl')
y_train=pd.read_pickle('input/fraud-detection_transformed_y_train_smote.pkl')



model = dtc(criterion = 'gini', max_depth = depth_DT)
```

```
model.fit(X_train, y_train)
# model.to_pickle('output/DecisionTree_'+str(depth_DT)+'.model')
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(model,X_train, y_train)
del y_train, X_train,depth_DT

# calc prediction accuracy for test data
X_test=pd.read_pickle('input/fraud-detection_transformed_X_test.pkl')
y_test=pd.read_pickle('input/fraud-detection_transformed_y_test.pkl')

pred_model = model.predict(X_test)
print(cl('Accuracy of the model is {:.0%}'.format(accuracy_score(y_test, pred_model)), attrs
= ['bold']))

# draw predicted decision tree
feature_names = X_test.columns
target_names = pd.unique(y_test).astype(str)
plot_tree(model,
        feature_names = feature_names,
        class_names = target_names,
        filled = True,
        rounded = True,
        proportion=True)

# make Confusion Matrix
# from sklearn.metrics import confusion_matrix
# confusion_matrix(y_test,pred_model,normalize=None)
plot_confusion_matrix(model,X_test, y_test)

del y_test, X_test, pred_model , feature_names,target_names, model
```

Figure 20. (Appendix 3). Python code for Logit model and Decision Tree

```
# -*- coding: utf-8 -*-
import os
from matplotlib import pyplot as plt
import seaborn as sns
```

```
import pandas as pd
import tensorflow.keras as keras
import tensorflow as tf

dir_data='***'
os.chdir(dir_data)



def myConfusionMatrixPlot(y_test,y_pred,norm_flg):
    y_testdf=pd.DataFrame(y_test.iloc[:,0])
    y_testdf.reset_index(inplace=True)
    y_preddf=pd.DataFrame(y_pred.iloc[:,0])
    y_preddf.reset_index(inplace=True)
    tmp=pd.DataFrame(y_preddf.iloc[:,1])
    tmp['true']=y_testdf.iloc[:,1]
    mat=pd.crosstab(tmp.iloc[:,1],tmp.iloc[:,0],normalize=norm_flg)
    fig=plt.figure()
    fig.add_subplot(111)
    plt.xlabel("predicted label")
    plt.ylabel("true label")
    if norm_flg :
        plt.title("Confusion Matrix (%)")
        sns.heatmap(round(mat*100,3),annot=True,square=True,cbar=False)
    else:
        plt.title("Confusion Matrix")
        sns.heatmap(mat,annot=True,square=True,cbar=False, fmt='g')
    fig.show()

def return_y_pred(X,model):
    pred_model_p=model.predict(X)
    try:
        y_pred=[1 if pred_model_p[i]>0.5 else 0 for i in range(len(pred_model_p))]
    except:
        y_pred=[1 if pd.DataFrame(pred_model_p)[0][i]>0.5 else 0 for i in
range(len(pd.DataFrame(pred_model_p)[0]))]
    y_pred=pd.DataFrame(y_pred,columns=['y_pred'])
```

```
    return y_pred

#-----------------------------------
#---------CNN
#-----------------------------------

X_train=pd.read_pickle('input/fraud-detection_transformed_X_train_smote.pkl')
# convert the shape to make df fit for Conv1D func
X_train=X_train.values.reshape(X_train.shape[0],X_train.shape[1],1)
y_train=pd.read_pickle('input/fraud-detection_transformed_y_train_smote.pkl')



# 1 dimentional convolutional neural network model
model = tf.keras.models.Sequential()
model.add(keras.layers.Conv1D(filters=24,                                    kernel_size=40,
input_shape=(X_train.shape[1:])))
model.add(keras.layers.Conv1D(filters=24, kernel_size=40))
# model.add(keras.layers.Conv1D(filters=24, kernel_size=100))
# model.add(keras.layers.Conv1D(filters=24, kernel_size=200))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(1024, activation=tf.nn.relu))
model.add(keras.layers.Dense(512, activation=tf.nn.relu))
model.add(keras.layers.Dense(32, activation=tf.nn.relu))
model.add(keras.layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['acc'])
model.summary()

model.fit(X_train,    y_train,epochs=10,verbose=1,    batch_size=512,use_multiprocessing=
True)
model.save(dir_data+'/output/CNN_smote')



model = keras.models.load_model(dir_data+'/output/CNN_smote')
myConfusionMatrixPlot(pd.DataFrame(y_train),return_y_pred(X_train,model),False)
del y_train, X_train
```

```
# make Confusion Matrix for test set
X_test=pd.read_pickle('input/fraud-detection_transformed_X_test.pkl')
X_test=X_test.values.reshape(X_test.shape[0],X_test.shape[1],1)
y_test=pd.read_pickle('input/fraud-detection_transformed_y_test.pkl')
myConfusionMatrixPlot(pd.DataFrame(y_test),return_y_pred(X_test,model),False)

y_est_p = model.predict(X_test)
y_est=[1 if y_est_p[i]>0.5 else 0 for i in range(len(y_est_p))]

df=pd.DataFrame({'Pred_Prob':[y_est_p[i][0] for i in range(len(y_est_p))],
                 'Pred_y':y_est,
                 'Act_y':y_test.to_list()})

df.to_csv(dir_data+'/output/CNN_smote/prediction.csv',index=False)
del y_test, X_test

model.save(dir_data+'/output/CNN_smote')
del model
```

Figure 21. (Appendix 4). Python code for Neural Network model

```
# -*- coding: utf-8 -*-
import os
import pandas as pd
import keras
import numpy as np
import matplotlib.pyplot as plt

dir_data='***'
os.chdir(dir_data)

model = keras.models.load_model(dir_data+'/output/CNN_smote')

FI=[]
col_n=34 # from C1
```

```
y_train=pd.read_pickle('input/fraud-detection_transformed_y_train.pkl').astype('float16')
X_train=pd.read_pickle('input/fraud-detection_transformed_X_train.pkl').astype('float16')
X_train=X_train.values.reshape(X_train.shape[0],X_train.shape[1],1)
y_est_o=model.predict(X_train)
y_est_o=[y_est_o[i][0] for i in range(len(y_est_o))]
error_o=np.std(y_est_o-y_train)
del X_train

for k in range(0,14):
    y_train=pd.read_pickle('input/fraud-detection_transformed_y_train.pkl').astype('float16')
    X_train_per=pd.read_pickle('input/fraud-
detection_transformed_X_train.pkl').astype('float16')
    col=X_train_per.columns[col_n]
    cut_point=int(X_train_per.shape[0]/2)
    X_train_a=X_train_per.iloc[:cut_point,:]
    X_train_b=X_train_per.iloc[cut_point:,:]

    print(str(k)+':'+col)

    tmp_a=X_train_a[col].values
    tmp_b=X_train_b[col].values
    X_train_a.loc[:,col]=[tmp_b[i] for i in range(len(tmp_b))]
    X_train_b.loc[:,col]=[tmp_a[i] for i in range(len(tmp_a))]
    del X_train_per,tmp_a,tmp_b

    X_train_a=X_train_a.astype('float16')
    X_train_b=X_train_b.astype('float16')
    X_train_a=X_train_a.values.reshape(X_train_a.shape[0],X_train_a.shape[1],1)
    X_train_b=X_train_b.values.reshape(X_train_b.shape[0],X_train_b.shape[1],1)
    y_est_p_a=model.predict(X_train_a)
    y_est_p_a=[y_est_p_a[i][0] for i in range(len(y_est_p_a))]
    error_p_a=np.std(y_est_p_a-y_train[:cut_point])
    y_est_p_b=model.predict(X_train_b)
    y_est_p_b=[y_est_p_b[i][0] for i in range(len(y_est_p_b))]
    error_p_b=np.std(y_est_p_b-y_train[cut_point:])
    error_p=(error_p_a+error_p_b)/2
```

```
    FI.append(error_p/error_o)
    del y_est_p_a,y_est_p_b, y_train
    del error_p, X_train_a,X_train_b
    col_n+=1

X_train=pd.read_pickle('input/fraud-detection_transformed_X_train.pkl').astype('float16')
col_n=34
plt.bar(X_train.columns[col_n:(col_n+14)], FI, label='Feature Importance')
plt.legend()
plt.xlabel('Feature')
plt.ylabel('Feature Importance')
plt.ylim(0.99, 1.02)
plt.title('Feature Importance')
plt.show()
del X_train
```

Figure 22. (Appendix 5). Python code for application of Interpretation methods 1: Feature Importance applied to CNN model

```
# -*- coding: utf-8 -*-
import os
import pandas as pd
import keras
import numpy as np
import matplotlib.pyplot as plt
import gc

dir_data='***'
os.chdir(dir_data)

model = keras.models.load_model(dir_data+'/output/CNN_smote')

def my_PDp(col_n,model):
    PD=[]
    for i in range(11):
        print(str(i)+'/10')
```

```
        X_train=pd.read_pickle('input/fraud-
detection_transformed_X_train.pkl').astype('float16')
        col=X_train.columns[col_n]
        tmp=X_train
        tmp[col]=np.zeros(tmp.shape[0])+i/10
        tmp=tmp.astype('float16')
        tmp=tmp.values.reshape(tmp.shape[0],tmp.shape[1],1)
        del X_train
        y_est=model.predict(tmp)
        PD.append(np.average(y_est))
        del tmp,y_est
        print(str('PD is:'+str(PD[i])))

    plt.plot([j/10 for j in range(11)], PD, label=col)
    plt.legend()
    plt.xlabel(col)
    plt.ylabel('average Probability')
    plt.title('Partial Dependence plot')
    plt.show()

    gc.collect()

for col_n in range(36,34+14):
    my_PDp(col_n,model)
```

Figure 23. (Appendix 6). Python code for application of Interpretation methods 2: Partial Dependence plot applied to CNN model

```
# -*- coding: utf-8 -*-
import lime
import lime.lime_tabular
import os
import keras
import pandas as pd
import numpy as np
# import gc
```

```
dir_data='***'
os.chdir(dir_data)

model = keras.models.load_model(dir_data+'/output/CNN_smote')
X_train=pd.read_pickle('input/fraud-detection_transformed_X_train.pkl').astype('float16')
feature_nm=np.array(X_train.columns)
categ_features =[]
X_train=X_train.values.reshape(X_train.shape[0],X_train.shape[1],1)
explainer = lime.lime_tabular.RecurrentTabularExplainer(X_train,
                                        feature_names=feature_nm,
                                        class_names=['Fraud'],
                                        categorical_features=categ_features,
                                        verbose=True)


# make function
class myLIME:
    def contribution(exp):
        res=exp.as_list(0)
        coef=exp.as_map()
        x=[]
        y=[]
        for j in range(len(coef[0])):
            res[j]=list(res[j])
            x.append(feature_nm[coef[0][j][0]] + res[j][0][-7:])
            y.append(res[j][1])
        import matplotlib.pyplot as plt
        fig, ax = plt.subplots()
        ax.barh(x,y)
        ax.set_title('Local explanation for class Fraud')
        ax.invert_yaxis()
        plt.show()
    def featurevalue(exp):
        coef=exp.as_map()
        x=[]
        y=[]
```

```
        for j in range(len(coef[0])):
            x.append(feature_nm[coef[0][j][0]])
            y.append(X_train[i][coef[0][j][0]][0])
        import matplotlib.pyplot as plt
        fig, ax = plt.subplots()
        ax.barh(x,y)
        ax.set_title('Feature Value')
        ax.invert_yaxis()
        plt.show()

i = 3
exp = explainer.explain_instance(X_train[i], model.predict_proba,labels=(0, ), num_features=10)

myLIME.contribution(exp)
myLIME.featurevalue(exp)

X_train=pd.read_pickle('input/fraud-detection_transformed_X_train.pkl').astype('float16')
X_train=X_train.values.reshape(X_train.shape[0],X_train.shape[1],1)
xcol=pd.read_pickle('input/fraud-detection_transformed_X_train.pkl').astype('float16').columns
model.predict(X_train[i:i+1])

# make fake sample as a counter sample
counter_sample=X_train[i:i+1]
counter_sample[0][xcol=="DeviceInfo_BLADE L7 Build/MRA58K"]=0
counter_sample[0][xcol=="DeviceInfo_R831L"]=1
model.predict(counter_sample)
```

Figure 24. (Appendix 7). Python code for application of Interpretation methods 3: LIME applied to CNN model