Algorithm Bias and Interpretable AI Project Final Report

Jie Li, Jeffrey Zhai, Litong Liu, Virgil Chen

1. Background Information (Virgil Chen)

Artificial Intelligence (Al) and Machine Learning (ML) algorithms are nowadays among the most prospering fields of data science. These methods are considered to be highly potential for various different industries and absolutely catch the attention of actuarial scientists, who seek the application of AI and ML into the insurance industry to be incorporated with the existing methods.

However, since the training work on the algorithm can never achieve an accuracy of 100%, there is always the possibility of Bias. For example, regarding a machine learning model, which identify the risk of committing a crime by inputting a picture of a certain human being, the algorithm sometimes has bias on the gender or race of the human being in the picture of underestimating or overestimating the chance of crime commitment of this certain human being. Therefore, to avoid the possible biases existing within a model, we need to find out ways to let the "model-builders" understand more about what exactly is happening within the input, output, and algorithms.

One of the biggest challenges for AI and ML currently is the lack of transparency, and the limited interpretation of the decisions made by the "machines", especially when complex algorithms are involved. Without people fully understanding the internal logic behind all of the decisions made, the algorithms are always referred to as black boxes, devices that process outputs from inputs without revealing the internal reasons. Therefore, many of the difficulties within the algorithms stay insolvable. For example, algorithm bias is the inequality often brought

up by the algorithms when they are introduced to an under-represented or over-represented group.

As a reason, scholars developed the concept of Explainable Artificial Intelligence (XAI), where people seek to enhance the transparency of certain algorithms so that they are more understandable or interpretable from the perspectives of human beings.

To look into XAI, this project focuses on various concepts, including “Interpretability and Explainability”, “Global Model-Agnostic Methods”, “Local Model-Agnostic Methods”, etc. By learning the developing concepts and methods through reviewing the literature, our team is trying to establish a connection between XAI and the insurance industry.

2. Bias

2.1 Uncovering and Mitigating Algorithmic Bias through Learned Latent Structure (Jie Li)

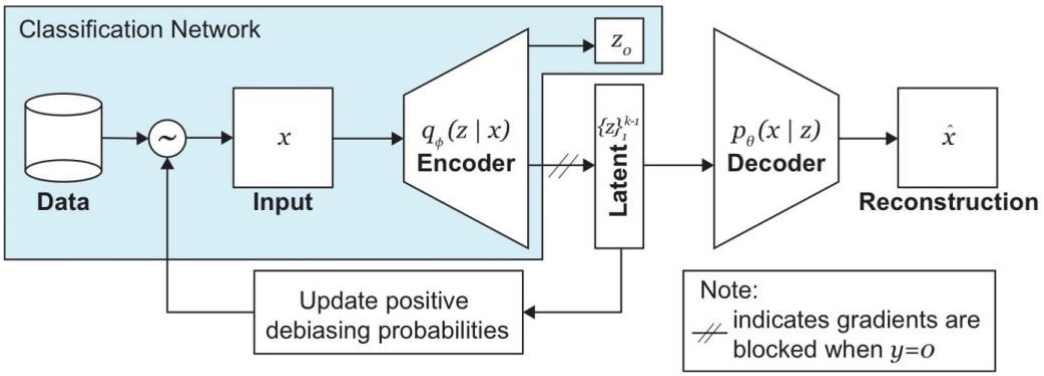
- Consider the problem of binary classification in which we are presented with a set of paired training data samples *D* train= $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ consisting of features $x \in R^m$ and labels $y \in R^d$. Our goal is to find a functional mapping $f : X \rightarrow Y$ parameterized by θ which minimizes a certain loss L(θ) over our entire training dataset. In other words, we seek to solve the following optimization problem:

$$\theta^* = arg_\theta min \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i(\theta). \quad (1)$$

- Given a new test example, (x , y), our classifier should ideally output $\hat{y} = f_\theta(x)$ where $\hat{y}$ is “close” to y, with the notion of closeness being defined from the original loss function. Now, assume that each datapoint also has an associated continuous latent vector $z \in R^k$ which captures the hidden, sensitive features of the sample (Zemel, et al.2013).
- Within a single class, the unobserved latent variables should be balanced

- We can measure the bias of the classifier by computing its accuracy across each of the sensitive categories.

(note: the overall accuracy of the classifier is the mean accuracy over all categories, the bias is the variance in these categories)



https://blog.csdn.net/smileyan9/article/details/107362252

https://baijiahao.baidu.com/s?id=1623879240712601046&wfr=spider&for=pc&searchword=DB-VAE

- We train the network end-to-end using backpropagation with a three component loss function comprised of a **supervised latent loss**, **a reconstruction loss**, and a latent loss for the **unsupervised variables**.

$$\mathcal{L}_{TOTAL} = c_1 \underbrace{\left[\sum_{i \in \{0,1\}} y_i \log\left(\frac{1}{\hat{y}_i}\right)\right]}_{\mathcal{L}_y(y,\hat{y})} + c_2 \underbrace{\left[\|x - \hat{x}\|_p\right]}_{\mathcal{L}_x(x,\hat{x})} + c_3 \underbrace{\left[\frac{1}{2}\sum_{j=0}^{k-1}(\sigma_j + \mu_j^2 - 1 - \log(\sigma_j))\right]}_{\mathcal{L}_{KL}(\mu,\sigma)} \quad (2)$$

where c1, c2, c3 are the weighting coefficients to impact the relative importance of each of the individual loss functions.

Algorithm 1 Adaptive re-sampling for automated debiasing of the DB-VAE architecture

Require: Training data $\{X, Y\}$, batch size b

Initialize weights $\{\phi, \theta\}$
for each epoch, E_t **do**
  Sample $z \sim q_\phi(z|X)$
  Update $\hat{Q}_i(z_i(x)|X)$
  $\mathcal{W}(z(x)|X) \leftarrow \prod_i \frac{1}{\hat{Q}_i(z_i(x)|X)+\alpha}$
  while $iter < \frac{n}{b}$ **do**
    Sample $\boldsymbol{x}_{batch} \sim \mathcal{W}(z(x)|X)$
    $L(\phi, \theta) \leftarrow \frac{1}{b} \sum_{i \in \boldsymbol{x}_{batch}} \mathcal{L}_i(\phi, \theta)$
    Update: $[w \leftarrow w - \eta \nabla_{\phi,\theta} \mathcal{L}(\phi, \theta)]_{w \in \{\phi,\theta\}}$
  end while
end for

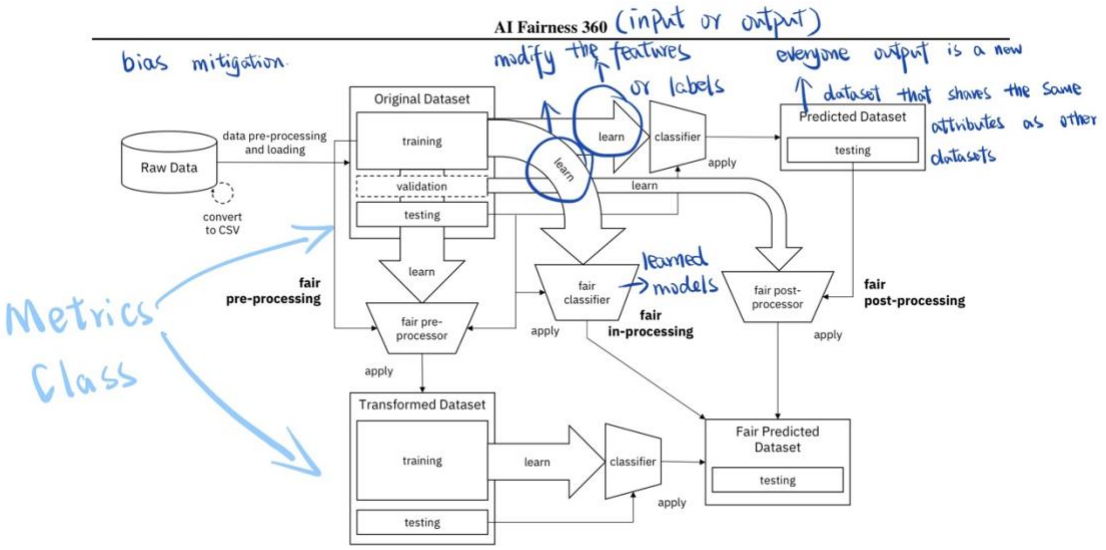
- Advantages: It's a novel, tunable debiasing algorithm to adjust the respective sampling probabilities of individual data points while training. By learning the underlying latent variables in an entirely unsupervised manner, we can scale this approach to large datasets and debias for latent features without ever hand labeling them in the training set.

2.2 AI FAIRNESS 360: AN EXTENSIBLE TOOLKIT FOR DETECTING, UNDERSTANDING, AND MITIGATING UNWANTED ALGORITHMIC BIAS

- **Introduction of AI Fairness 360: https://github.com/ibm/aif360**
  1) An extensible toolkit for detecting, understanding, and mitigating algorithmic biases.
  2) The goals are to promote a deeper understanding of fairness metrics and mitigation techniques
  3) To enable an open common platform for fairness researchers and industry practitioners to share and benchmark their algorithms
  4) To help facilitate the transition of fairness research algorithms to use in an industrial setting

AIF 360 is the first system to bring together in one open source toolkit: bias metrics, bias mitigation algorithms, bias metric explanations, and industrial usability

- **The fairness pipeline**



- The bias mitigation algorithm categories are based on the location where these algorithms can intervene in a complete machine learning pipeline. If the algorithm is allowed to modify the training data, then pre-processing can be used. If it is allowed to change the learning procedure for a machine learning model, then in-processing can be used. If the algorithm can only treat the learned model as a black box without any ability to modify the training data or learning algorithm, then only post-processing can be used.

Datasets	Adult Census Income, German Credit, COMPAS
Metrics	Disparate impact Statistical parity difference Average odds difference Equal opportunity difference
Classifiers	Logistic regression (LR), Random forest classifier (RF), Neural Network (NN)
Pre-processing Algorithms	Re-weighing (Kamiran & Calders, 2012) Optimized pre-processing (Calmon et al., 2017) Learning fair representations (Zemel et al., 2013) Disparate impact remover (Feldman et al., 2015)
In-processing Algorithms	Adversarial debiasing (Zhang et al., 2018) Prejudice remover (Kamishima et al., 2012)
Post-processing Algorithms	Equalized odds post-processing (Hardt et al., 2016) Calibrated eq. odds postprocessing (Pleiss et al., 2017) Reject option classification (Kamiran et al., 2012)

- Advantages:

  The AIF360 platform is designed to help researchers in the field of fairness investigate and compare various algorithms for detecting and mitigating bias, as well as contribute and evaluate new algorithms and datasets. It also provides resources for developers, including

education on bias-related issues, guidance on which metrics and algorithms to use, and a Python package for detecting and mitigating bias in their workflows. However, it is important to note that fairness is a complex concept that cannot be fully captured by the metrics and algorithms available in AIF360. Future research may aim to expand the toolkit to address additional aspects of justice, such as compensatory justice, and to offer a wider range of explanations. It is crucial for the research community to continue contributing to the toolkit in order to advance the goal of unbiased AI.

2.3 Debiasing word embeddings

2.3.1 What is word embedding

Word embedding is a technique used in natural language processing (NLP) to represent words in a continuous, numerical space. The goal of word embedding is to capture the meaning of words and the relationships between them in a way that can be used as input to machine learning models.

- A word embedding that represents each word (or common phrase) **w** as a d-dimensional word vector: $\vec{w} \in R^d$
- Words with similar semantic meanings tend to have vectors that are close together
- The vector differences between words in embeddings have been shown to represent relationships between words

EG: man is to king as woman is to x, denoted as **man: king :: woman: x**

$\vec{man} - \vec{woman} \approx \vec{king} - \vec{queen}$

X = Japan is returned for **Paris: France :: Tokyo :x**

- They are being studied and used in a variety of downstream applications (e.g., document ranking, sentiment analysis, and question retrieval).
- One hot encoding

If the corpus is so long, the matrix will get tedious.

$$l=\begin{matrix}1\\0\\0\\0\end{matrix} \quad o=\begin{matrix}0\\1\\0\\0\end{matrix} \quad v=\begin{matrix}0\\0\\1\\0\end{matrix} \quad e=\begin{matrix}0\\0\\0\\1\end{matrix}$$

2.3.2 The goal of debiasing

Reduce bias:

- Ensure that gender neutral words such as *nurse* are equidistant between gender pairs such as *he* and *she*
- reduce gender associations that pervade the embedding even among gender neutral words

Maintain embedding utility:

- Maintain meaningful non-gender-related associations between gender neutral words, including associations within stereotypical categories of words such as fashion-related words or words associated with football.
- Correctly maintain definitional gender associations such as between *man* and *father*

2.3.3 Methodology

- Define an embedding and some terminology:

An embedding consists of a unit vector $\vec{w} \in R^d$, with $\| \vec{w} \| = 1$

- Assume there is a set of gender neutral words N ⊂ W: *flight, attendant* or shoes
- We assume we are given a set of F-M gender pairs P ⊂ W*W, such as *she-he* or *mother-father*, whose definitions differ mainly in gender
- Similarity between words w_1 and w_2 is measured by their inner product
- The input into our analogy generator is a seed pair of words (a, b) determining a seed direction $\vec{a}$ -$\vec{b}$ corresponding to the normalized difference between the two seed words.

  Use (a, b) = (she, he)

$$\mathrm{S}_{(a,b)}(x,y) = \begin{cases} \cos\left(\vec{a}-\vec{b}, \vec{x}-\vec{y}\right) & \text{if } \|\vec{x}-\vec{y}\| \leq \delta \\ 0 & \text{otherwise} \end{cases}$$

2.3.4 Debiasing Algorithms

- Identify gender subspace: identify a direction of the embedding that captures the bias
- Define two options: Neutralize and Equalize

A: Neutralize ensures that gender-neutral words are zero in the gender subspace.

B: Equalize perfectly equalizes sets of words outside the subspace and thereby enforces the property that any neutral word is equidistant to all words in each equality set.

Eg: if ***{grandmother, grandfather}*** and **{guy, gal}** were two equality sets, then after equalization ***babysit*** would be equidistant to grandmother and grandfather and also equidistant to gal and guy, but presumably closer to the grandparents and further from the gal and guy.

1. To define the algorithm

A subspace B is defined by k orthogonal unit vectors B = $\{b_1, \ldots, b_k\} \subset R^d$.

$$V_B = \sum_{j=1}^{k} (v \cdot b_j)\, b_j$$

2. Identify gender subspace

Inputs: word sets W , defining sets $D_1, D_2, \ldots, D_n \subset$ W as well as embedding $\{\vec{w} \in R^d\}$, w ∈ W, let

$$\mu_i = \sum_{w \in D_i} \vec{w} \,/\, |D_i|$$

3. 1) Let the bias subspace B be the first k rows of SVD (C)

$$\boldsymbol{C} := \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \sum_{w \in D_i} (\vec{w} - \mu_i)^{\boldsymbol{T}} (\vec{w} - \mu_i) \,/\, |D_i|$$

2) **Hard de-biasing(neutralize and equalize)**

Additional inputs: words to neutralize N ⊆ W , family of equality sets $\varepsilon = \{E\}\_1, E\_2, \ldots, E\_m\}$ where $E\_i \subseteq$ W. For each word w ∈ N, let $\vec{w}$ be re-embedded to:

$$\vec{w} := (\vec{w} - \vec{w}_B) \,/\, \|(\vec{w} - \vec{w}_B)\|$$

For each set E ∈ $\boldsymbol{\varepsilon}$, let

$$\begin{aligned} \mu &:= \sum_{w \in E} w/|E| \\ \nu &:= \mu - \mu_B \\ \text{For each } w \in E,\ \vec{w} &:= \nu + \sqrt{1 - \|\nu\|^2} \frac{\vec{w}_B - \mu_B}{\|\vec{w}_B - \mu_B\|} \end{aligned}$$

Finally, output the subspace B and the new embedding $\{\vec{w} \in R^d\}$, w ∈ W

Additional inputs: words to neutralize N ⊆ W , family of equality sets

$\varepsilon = \{E_1, E_2, \ldots, E_m\}$ where $E_i \subseteq$ W. For each word w ∈ N, let $\vec{w}$ be re-embedded to

3. Interpretability and Explainability (Virgil Chen)

To bring the concept of interpretability into Machine Learning, scholars have developed many scopes and methods that help differentiating the different methods and to what aspect is the interpretability introduced into the model.

3.1 Definitions:

There is no mathematical definition of Interpretability.

3.1.1 Non-mathematical Definitions:

1. Interpretability is the degree to which a human can understand the cause of a decision.

2. The degree to which a human can consistently predict the model's result.

3.2 Interpretability is not necessary under:

1. When there is no significant impact or severe consequences for incorrect results.

2. When the problem is well-studied enough and validated in real applications that we trust the system's decisions, even if the system is not perfect.

3.3 Interpretability helps to optimize:

1. Fairness:

Ensure that predictions are unbiased and do not implicitly or explicitly discriminate against protected groups.

2.Privacy:

Ensure that sensitive information in the data is protected.

3.Reliability/Robustness:

Ensure that small changes in the input do not cause large changes in the prediction.

4.Trust:

It is easier for humans to trust a system that explains its decisions rather than a black box that just outputs the decision itself.

3.4 ML Interpretability Methods and Techniques:

3.4.1 Pre-Model vs. In-Model vs. Post-Model:

1. Pre-model:

Interpretability techniques are independent of the model, as they are only applicable to the data itself.

Pre-model interpretability usually happens before model selection. Pre-model interpretability is, thus, closely related to data interpretability.

2.In-model:

Interpretability concerns ML models that have inherent interpretability in it (through constraints or not), being intrinsically interpretable.

3.Post-model:

interpretability refers to improving interpretability after building a model.

3.4.2 Intrinsic vs. Post hoc:

1.Intrinsic:

interpretability is achieved through imposition of constraints on the model complexity. Used to get the answer of how the model works.

2.Post hoc:

interpretability refers to explanation methods that are applied after model training. Used to get the answer of what else can the model tell us.

3.4.3 Model-specific vs. Model-agnostic:

1.Model-specific:

interpretation methods are limited to specific model classes because each method is based on some specific model's internals. For instance, the interpretation of weights in a linear model is a model-specific interpretation.

2.Model-agnostic:

methods can be applied to any ML model (black box or not) and are applied after the model has been trained. These methods rely on analyzing pairs of feature input and output. By definition, these methods cannot have access to the model inner workings.

3.5 Results of ML Interpretability methods (Explanation Methods):

1.Feature summary:

summary statistics for each feature. This can be, e.g., a single number per feature, such as feature importance.

2.Model internals:

model internals and summary statistics, such as the weights in linear models.

3. Data Point:

There are methods that return data points (already existent or not) to make a model interpretable. These are example-based methods. This works well for images and texts but is less useful for, e.g., tabular data with hundreds of features.

4. Surrogate intrinsically interpretable model:

Another solution for interpreting black box models is to approximate them (either globally or locally) with an intrinsically interpretable model. Thus, the interpretation of the surrogate model will provide insights of the original model.

3.6 Scope of Interpretability:

3.6.1 Global Scope:

1.On a Holistic Level:

Our interpretability methods have a holistic global scope if they can answer the question of "how does a trained model make predictions?" At this scope we are trying to understand the distribution of the prediction output based on the input features.

2.On a Modular Level:

Our interpretability methods have a holistic global scope if they can answer the question of "how do parts of the model affect predictions?" Only a few models are interpretable at a parameter level. For example, for linear models, the interpretable parts are the weights; for decision trees, the interpretable parts are the splits (features and cut-off values) and leaf node predictions.

3.6.2 Local Scope:

1. For a Single Prediction:

Aiming to explain a single prediction, the general idea is to zoom in on a single instance and to try to understand how the model arrived at its prediction. This can be done by approximating a small region of interest in a black box model using a simpler interpretable model. As locally, the prediction might only depend linearly or monotonously on some features rather than having a complex dependence on them.

2. For a Group of Predictions:

In order to explain a group of predictions, there are essentially two possibilities: apply global methods and treat the group of predictions of interest as if it was the whole dataset or apply local methods on each prediction individually, aggregating and joining these explanations afterwards.

3.7 Classification of Explanation Theory:

1. Non-pragmatic theory of explanation:

The explanation should be the correct answer to the why-question. Non-pragmatic theories typically, but not always, follow a position where it is assumed there is only one true explanation. This means that the correctness of the answer has nothing to do with whether the audience can understand it or not.

2.Pragmatic Theory of explanation:

The explanation should be a good answer for anexplainer to give when answering the why-question to an audience. Pragmatic theories argue that the definition of an explanation should necessarily have a place for the listener.

3.8 Properties of Explanation Methods:

1. Expressive power:

It is the language or structure of the explanations the method is able to generate. These could be, e.g., rules, decision trees, etc.

2.Translucency:

It represents how much the explanation method relies on looking into the inner workings of the ML model, such as the model's parameters. E.g., model-agnostic methods have zero translucency.

3.Portability:

Describes the range of ML models to which the explanation method can be applied. It is inversely proportional to translucency.

4.Algorithmic Complexity:

It is related to computational complexity of the explanation method.

3.9 Properties of Explanations (Results of Explanation Methods):

1.Accuracy:

It is related to the predictive accuracy of the explanation regarding unseen data.

2.Fidelity:

It is associated with how well the explanation approximates the prediction of the black box model. Accuracy and fidelity are closely related: if the black box model has high accuracy and the explanation has high fidelity, the explanation consequently has high accuracy.

3.Consistency:

Regarding two different models that have been trained on the same task and that output similar predictions, this property is related to how different the explanations are between them. If the explanations are very similar, the explanations are highly consistent.

4.Stability:

It represents how similar the explanations are for similar instances.

3.10 Properties that make Explanations Human Friendly:

1.Contrastiveness:

Humans usually do not ask why a certain prediction was made but rather why this prediction was made instead of another prediction. People are not specifically interested in all the factors that led to the prediction but instead in the factors that need to change (in the input) so that the ML prediction/decision (output) would also change.

2.Selectivity:

People do not expect explanations that cover the actual and complete list of causes of an event. Instead, they prefer selecting one or two main causes from a variety of possible causes as the explanation.

3.Social:

The best explanation varies according to the application domain and use case.

3.11 Evaluation of Interpretability:

1.Application-grounded evaluation:

Requires conducting end-user experiments within a real application. This experiment is performed by using the explanation in a real-world application and having it tested and evaluated by the end user, who is also a domain expert. A good baseline for this is how good a human would be at explaining the same decision.

2.Human-grounded evaluation:

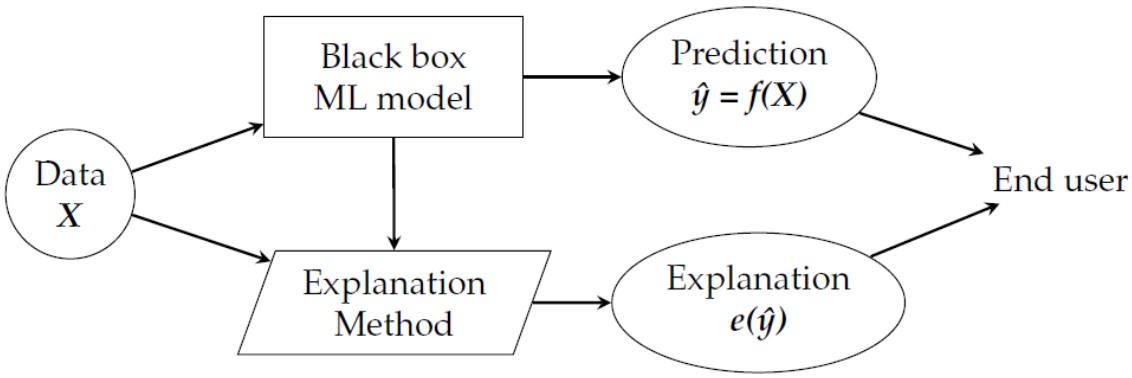
Refers to conducting simpler human–subject experiments that maintain the essence of the target application. The difference is that these experiments are not carried out with the domain experts but with laypersons. Since no domain experts are required, experiments are cheaper, and it is easier to find more testers.

3.Functionally grounded evaluation:

Requires no human experiments. In this type of evaluation, some formal definition of interpretability serves as a proxy to evaluate the explanation quality, e.g., the depth of a decision tree.

3.12

XAI Pipeline:



4. Example methods

4.1 Systematic methods

4.1.1 Explainable Machine Learning in Credit Risk Management (Litong Liu)

- Although black box AI has high predictive accuracy, it is not suitable for regulated financial services due to its lack of interpretability, which motivated the author to propose a methodology that based on combination of network analysis with Shapley values to improve the interpretation of predictive output of a machine learning model.
- **Methodology**:
    - **Statistical learning of credit risk**:

- Suppose there are n companies, use Y_i to indicate whether company has defaulted on its loan or not (Y_i=1 is company defaults,0 otherwise), and use to represent a vector of explanatory variables;
- Logistic regression model: $ln(\frac{p_i}{1-p_i}) = \alpha + \sum_{j=1}^{J} \beta_j x_{ij}$, where p_i is the probability of default for company i; α and β_j can be estimated by data, and further we can calculate and get $p_n = (1 + exp(-\alpha - \sum_{j=1}^{J} \beta_j x_{nj}))^{-1}$;

- **Machine learning of credit risk**: Use extreme gradient boost model;
- **Learning model comparison**: once a default probability estimation model is chosen, measure its predictive accuracy and compare to others to select the best one, usually we use AUROC (area under receiver operating characteristics curve) to select model;
- **Explaining model predictions**: calculate Shapley value associated with each company, which can be used as a tool to transfer predictive inferences into a linear space; after Shapley value calculated, employ similarity networks, defining a metric that provides the relative distance between companies by applying Euclidean distance between each pair (x_i, x_j) and derive the Minimal Spanning Tree (MST) representation of companies. And then use the MST to predict companies performance and model interpretability.

4.1.2 Model Agnostic Supervised Local Explanations (MAPLE) (Litong Liu)

- Common types of model explanations:
  - **Example-based**: the points in training set that most closely resemble a test point or influenced the prediction;
  - **Local explanation**: the changes in model's prediction if input change slightly;
  - **Global explanation**: the patterns that underlying model's behavior.
- Two challenges of current local interpretability methods:
  - Hard to accurately modeling/detecting global patterns;

- Hard to determine if an explanation generated at one point can be applied at a new point;

To solve the current research gap, MAPLE, a local explanation system that can detect global patterns and determine influential points, is proposed!

- **Metric for evaluation**:
  - Define local explanation at x as $exp_x()$ and prediction as $pred()$
  - Casual local explanation metric: $\mathbb{E}_{x,x':p_x}[loss(exp_x(x'), pred(x'))]$
- **MAPLE**:
  - **SILO**:
    - use random forest as a method for supervised neighborhood selection for local linear modeling:
    - Define connection function of T_K : $c_k(x, x') = 1(leaf_k(x) = leaf_k(x'))$
    - Number of training points in the same leaf node as x: $num_k(x) = \sum_{i=1}^{n} c_k(x_i, x)$
    - Weight function of random forest for the training point at x: $w(x_i, x) = \frac{1}{K}\sum_{k=1}^{K}\frac{c_k(x_i, x)}{num_k(x)}$ (which can be counted as local training distribution)
    - SILO prediction: $\hat{f}_{SILO}(x) = \hat{\beta}_x^T x$, where and
  - **DStump**:
    - Let $split_k \in \{1, ..., p\}$ be index of feature that the root node of T_K split on;
    - Suppose that split reduces the impurity of label by r_k;
    - Stump assigns feature score: $s_j = \sum_{k=1}^{K} 1\{split_k = j\} r_k$;
    - Choose subset $A_d \subset \{1, ..., p\}$ of the d highest scored features;
  - **MAPLE prediction**:
    - Let $Z_d = [X]_{:,A_d} \in \mathbb{R}^{n\times(d+1)}$, and $z_d = [x]_{A_d} \in \mathbb{R}^{d+1}$ selected from DStump;
    - MAPLE prediction: $\hat{f}_{MAPLE}(x) = \hat{\beta}_{x,d}^T z_d$, where $\hat{\beta}_{x,d} = (Z_d^T W_x Z_d)^{-1} Z_d^T W_x y$.
  - **MAPLE as an explanation system**:
    - **MAPLE use a local linear model, coefficients determine the estimated local effect of each feature**;

        - If $c_i \neq 0$: interpret the impact of feature according to the sign and magnitude of c_i;
        - Otherwise, determine whether it contains global effect or not by following procedures;
- **Detecting global patterns**:
    - For each feature: using local training distribution for the given test point to create a box-plot to visualize the distribution of each feature:
        - If box plot is substantially skewed, it is likely containing a global pattern and the test point is nearby it;
        - Else, perform a grid search across the range of the feature:
            - For each value on the grid, sample the remaining features and create a box-plot for the local training distribution across this grid:
                - If local training distributions are similar boundaries that change abruptly during the grid search, then there is likely a global pattern present in that feature;
                - If local training distributions are roughly centered around the test points during grid search and change smoothly during it, the effect of the feature does not have a significant global pattern.
    - **Influential training points**:
        - If the slope of the function does not change too rapidly, the influential training points for a prediction at x are roughly centered around x and tend to change smoothly as x changes;
        - The steeper the function is at x, the distribution of the influential points becomes more concentrated;

- If fitting a discontinuous function, the influential points may not be centered around x when x is near discontinuity and they will change abruptly as x moves past the discontinuity.

Thus, explanations at influential points can be used as exemplar explanations to explain new test points.

4.2 Post-hoc methods

4.2.1 Regularizing Black-box Models for Improved Interpretability (Litong Liu)

- Explanation-based optimization method (EXPO) can be applied to and model families and can control the quality of explanations, which satisfies the current research gap of by-design approach and post-hoc approaches of interpretable machine learning.
- Consider a supervised learning problem: $f : \mathbb{X} \to \mathbb{Y}, f \in F$
  - One can understand the behavior of f in some neighborhood, $N_x \in P[\mathbb{X}]$, where $P[\mathbb{X}]$ is the space of probability distributions over $\mathbb{X}$, by generating a local explanation;
  - Denote systems that produce local explanations (explainers) as $e : \mathbb{X} \times F \to \mathcal{E}$, where $\mathcal{E}$ is the set of possible explanations;
  - Evaluation:
    - Neighborhood-fidelity (NF) metric: $F(f, g, N_x) := \mathbb{E}_{x' \sim N_x}[(g(x') - f(x'))^2]$;
    - Stability metric: $S(f, e, N_x) := \mathbb{E}[||e(x, f) - e(x', f)||_2^2]$;
  - Post-hoc explainers: LIME
- EXPO:
  - EXPO is solving the optimization problem: $\hat{f} = argmin_{f \in F} \sum_{i=1}^{N} (L(f, x_i, y_i) + \gamma R(f, N_{x_i}^{reg}))$, where $L(f, x_i, y_i)$ is the loss function, and $R(f, N_{x_i}^{reg})$ is a regularizer that encourages to be interpretable in the neighborhood of x_i;
  - Define $R(f, N_{x_i}^{reg})$ based on neighborhood-fidelity or stability;

- EXPO approximates the $R(f, N_{x_i}^{reg})$ during the calculation by EXPO-FIDELITY and EXPO-STABILITY regularizer as the following algorithm:

Algorithm 1 Neighborhood-fidelity regularizer

input $f_\theta, x, N_x^{\text{reg}}, m$
1: Sample points: $x'_1, \ldots, x'_m \sim N_x^{\text{reg}}$
2: Compute predictions: $\hat{y}_j(\theta) = f_\theta(x'_j)$ for $j = 1, \ldots, m$
3: Produce a local linear explanation: $\beta_x(\theta) = \arg\min_\beta \sum_{j=1}^m (\hat{y}_j(\theta) - \beta^\top x'_j)^2$
output $\frac{1}{m}\sum_{j=1}^m (\hat{y}_j(\theta) - \beta_x(\theta)^\top x'_j)^2$

Algorithm 2 Neighborhood-stability regularizer

input $f_\theta, x, N_x^{\text{reg}}, m$
1: Sample points: $x'_1, \ldots, x'_m \sim N_x^{\text{reg}}$
2: Compute predictions:

$$\hat{y}_j(\theta) = f_\theta(x'_j), \text{ for } j = 1, \ldots, m$$

output $\frac{1}{m}\sum_{j=1}^m (\hat{y}_j(\theta) - f(x))^2$

4.2.2 Shapley Values (Jeffrey Zhai)

Feature importance is one of the most popular methods in explaining machine learning models. According to Bhatt et al. (2020), feature importance defines an explanation function g : f × R^d → R^d that takes a model `f` and an input `x` and returns importance score `g(f, x)` ∈ R^d for all features. According to Molnar (2022), Shapley value refers to the average expected marginal contribution of one player after all possible combinations have been considered.

- Computation: $\phi_j(val) = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \frac{|S|!(p-|S|-1)!}{p!}(val(S \cup \{j\}) - val(S))$

where p means the number of features, $val_x(S)$ means the prediction for feature values in set S that are marginalized over features that are not included in set S.

- Application:

  Shapley value, which originated from a concept in game theory, could be applied in explainable machine learning to measure the importance of a feature in the model. By looking at shapley values of each feature, we could find the contribution of each feature

to the result of the ML model. For example, ”55 percent of the decision was decided by your age, which positively correlated with the predicted outcome.”

- Advantages:

  1. The difference between the predicted and average predicted values is fairly distributed among the feature values of the instances

  2. The Shapley value allows contrastive explanations. You could compare a prediction to a subset or even a single data point.

- Limitations:

  1. The Shapley value requires a lot of computing time. In almost all cases, only the approximate solutions like Monte-Carlo sampling are feasible.

  2. Explanations created with the Shapley value method always use all the features.

  3. The Shapley value returns a simple value per feature, but no prediction model like LIME

  (Lundberg and Lee, 2017).

4.2.3 Shapley additive explanations (Jeffrey Zhai)

Shapley additive explanation is a method proposed by Lundberg and Lee (2017) to interpret individual predictions based on shapley values. SHAP comes with a number of global interpretation methods based on aggregation of Shapley values. SHAP gives the interpretation of the model as $g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i'$

where g is the explanation model, $z' \in \{0,1\}^M$ is the coalition vector, M is the number of simplified input features and φi ∈ R is the shapley value for feature i.

Unlike shapley values, SHAP values could use Kernel SHAP and Deep SHAP approximation methods. With these methods, SHAP could be computed more efficiently with higher local accuracy and consistency. The fast computation of SHAP allows it to be used in global model interpretations such as feature dependence, clustering and summary plots (Molnar, 2022). Moreover, SHAP connects LIME and shapley values, which help to unify the approaches in interpreting machine learning.

However, it is possible to create intentionally misleading interpretations with SHAP. Some biases may be hidden by SHAP which are extremely hard to detect by humans. Besides, there are still some shortcomes with KernelSHAP and TreeSHAP that need to be further improved.

- KernelSHAP:

KernelSHAP is a kernel-based estimation for an instance x the contributions of each feature value to the prediction. There are five steps to compute KernelSHAP:

1. Sample coalitions $z_k^{'} \in \{0,1\}^M$ where $k \in \{1, ..., K\}$ and 0 represents absence of feature, 1 represents presence of feature.
2. For each $z_k^{'}$, get prediction by converting $z_k^{'}$ to the original feature space and then applying model $\hat{f} : \hat{f}(h_x(z_k^{'}))$
3. Compute the weight for each $z_k^{'}$ with the SHAP kernel
4. Fit weighted linear model
5. Get shapley values ϕ_k, the coefficients from the linear model

Same as other permutation-based interpretation methods, KernelSHAP also has the limitation about too much weight on unlikely instances. To solve this problem, we need to sample from the conditional distribution to change the value function.

- TreeSHAP:

According to (Lundberg and Lee, 2017), TreeSHAP is a variant of SHAP for tree-based machine learning models such as decision trees, random forests and gradient boosted trees. Compared to traditional shapley value, TreeSHAP is a fast, model-specific alternative but may produce unintuitive feature attributions.

Instead of the marginal expectation, TreeSHAP defined the value function using the conditional expectation $E_{X_S|X_C}(f(x)|x_s)$. One problem of TreeSHAP is that it may generate non-zero estimates for features that have no influence on the prediction. On the other hand, TreeSHAP is much faster than KernelSHAP because it reduces the computational complexity from $O(TL2^M)$ to $O(TLD^2)$ where T is the number of trees, L is the maximum number of leaves in any tree, and D is the maximal depth of any tree.

4.2.4 Partial Dependency Plots (Jeffrey Zhai)

The partial dependence plot (PDP) shows the dependence between the target response and a set of input features of interest, marginalized for the values of all other input features (Hastie et al., 2001). Intuitively, we can interpret partial dependencies as a function of the expected target response and the input features of interest.

The partial dependence function for regression is defined as:

$$\hat{f}_S(x_S) = E_{X_C}[\hat{f}(x_S, X_C)] = \int \hat{f}(x_S, X_C) d\mathbb{P}(X_C)$$

where x_S are the features for which the partial dependence function should be plotted. X_C are the other features used in the machine learning model $\hat{f}$.

The partial function $\hat{f}_S$ is estimated by calculating averages in the training dataset:

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

where $x_C{}^{(i)}$ are actual feature values from the dataset for the features in which we are not interested. n is the number of observations in our dataset.

- Advantages:

  1. The computation of partial dependence plots is intuitive. people could understand the idea of PDPs quickly.

  2. PDPs perfectly represent how the feature influences the prediction on average.

  3. Partial dependence plots are easy to implement.

  4. The calculation for the partial dependence plots has a causal interpretation. We could analyze the causal relationship between the feature and the prediction (Zhao and Hastie, 2021).

- Disadvantages:

  1. The realistic maximum number of features in a partial dependence function is two.

  2. Some PD plots do not show the feature distribution. Regions with almost no data might be overinterpreted.

  3. Partial dependence plots require the assumption of independence. It is assumed that the feature(s) for which the partial dependence is computed are not correlated with other features.

4.2.5 Counterfactual explanations (Jeffrey Zhai)

Counterfactual explanations are points close to the input for which the decision of the classifier changes. (Bhatt et al., 2020). For example, ”Had your income been greater by $5000, the loan would have been granted.”

In the field of interpretable machine learning, counterfactual explanations are often used to explain the individual instances’ predictions. We call the predicted outcome of an instance an “event”. The corresponding “cause” is a specific feature value of that instance that is fed into the model and ”leads” to a certain prediction.

Counterfactuals are human-friendly explanations, because they are contrastive to the current instance and because they are selective, meaning they usually focus on a small number of feature changes.

- Advantages:

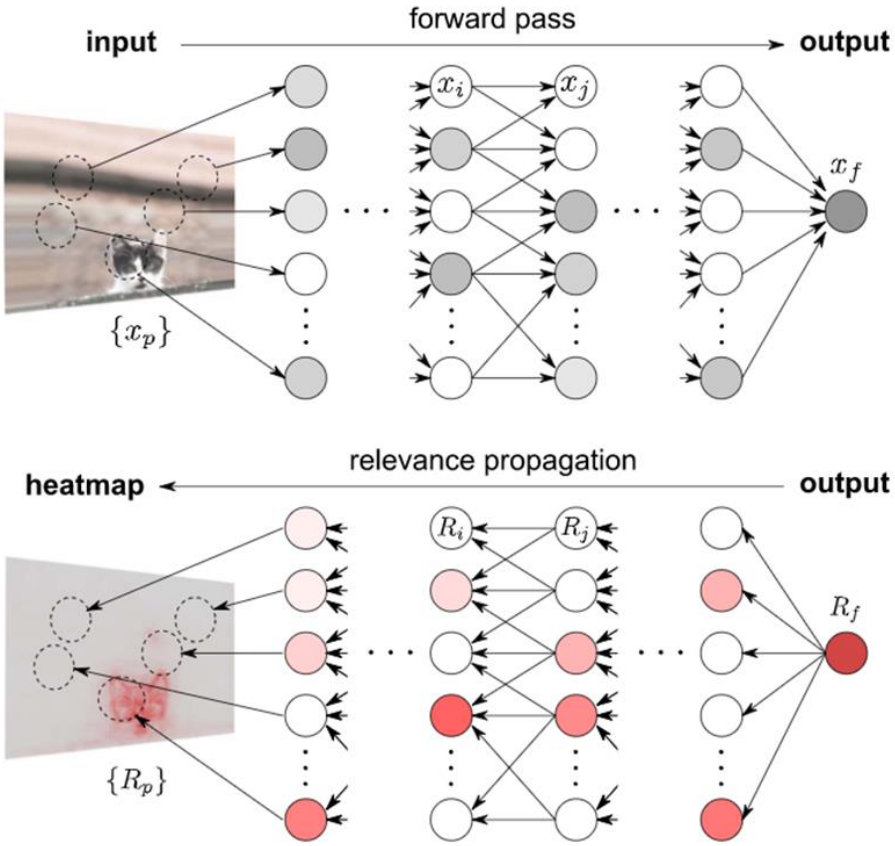
1. Works for any input data type, high flexibility.
2. Generates multiple explanations in a single run of the algorithm.
3. Does not require access to the data or the model, and the counterfactual method works also with systems that do not use machine learning.
4. Relatively easy to implement.

- Disadvantages:

1. Counterfactuality might not be feasible.
2. No guarantee of the optimality of the explanation.
3. There may be more than one counterfactual explanation present for each instance and it’s hard to compare them.

4.2.6 Deep-Taylor decomposition (Virgil Chen)

Definition: Deep Taylor decomposition efficiently utilizes the structure of the network by backpropagating the explanations from the output to the input layer. Montavon et al. (2017)



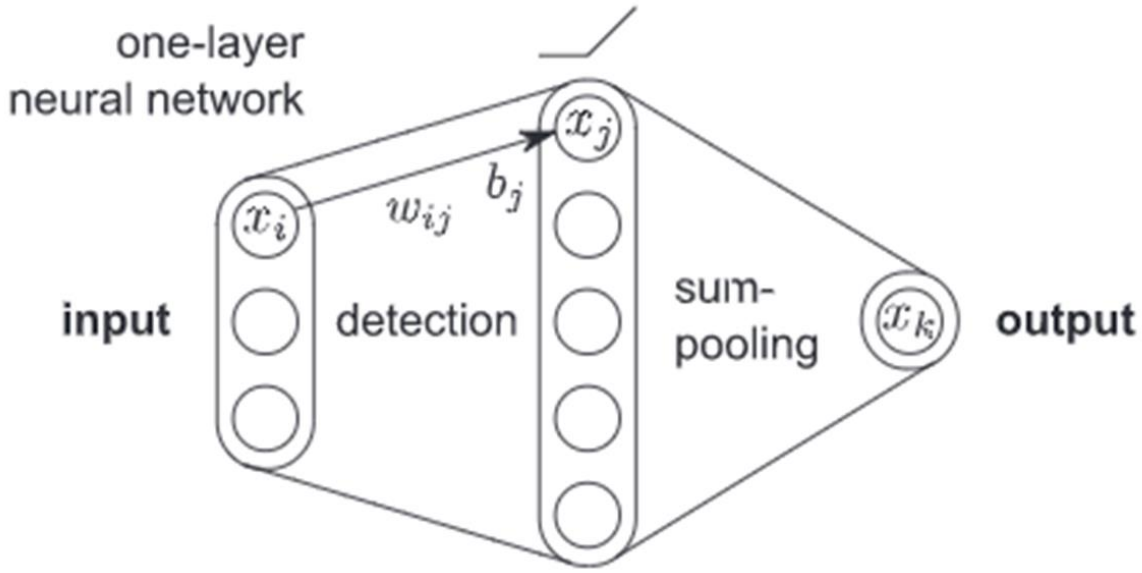
By assigning a relevance onto the output of the neural network model, the back propagation serves as the recursive calculation process to reassign the relevance onto each of the previous layers. The recursive process stops when reaching the linear layer, which is the very first layer and forming a heatmap, showing the importance of each feature.

Computation:

$$R_j = \left(\frac{j}{\partial \{x_i\}} \Big|_{\{\tilde{x}_i\}^{(j)}} \right)^T \cdot \left(\{x_i\} - \{\tilde{x}_i\}^{(j)} \right) + \varepsilon_j = \sum_i \underbrace{\frac{\partial R_j}{\partial x_i} \Big|_{\{\tilde{x}_i\}^{(j)}} \cdot \left(x_i - \tilde{x}_i^{(j)} \right)}_{R_{ij}} + \varepsilon_j$$

$$R_i = \sum_j \frac{w_{ij}^2}{\sum_{i'} w_{i'j}^2} R_i$$

Deep Taylor Pipeline:



Based on the different domain of the input, there are three different rules to be applied on, which results in some amendments based on the formula above.

Three different rules:

$Rule1 : w^2$-rule $\mathcal{X} = \mathbb{R}^d$

For all functions $g \in G$, the deep Taylor decomposition with the ω^2-rule is consistent.

$Rule1 : z^+$-rule $\mathcal{X} = \mathbb{R}^d_+$

For all functions $g \in G$ and data points $\{x_i\} \in \mathbb{R}^d_+$, the deep Taylor decomposition with the z^+-rule is consistent.

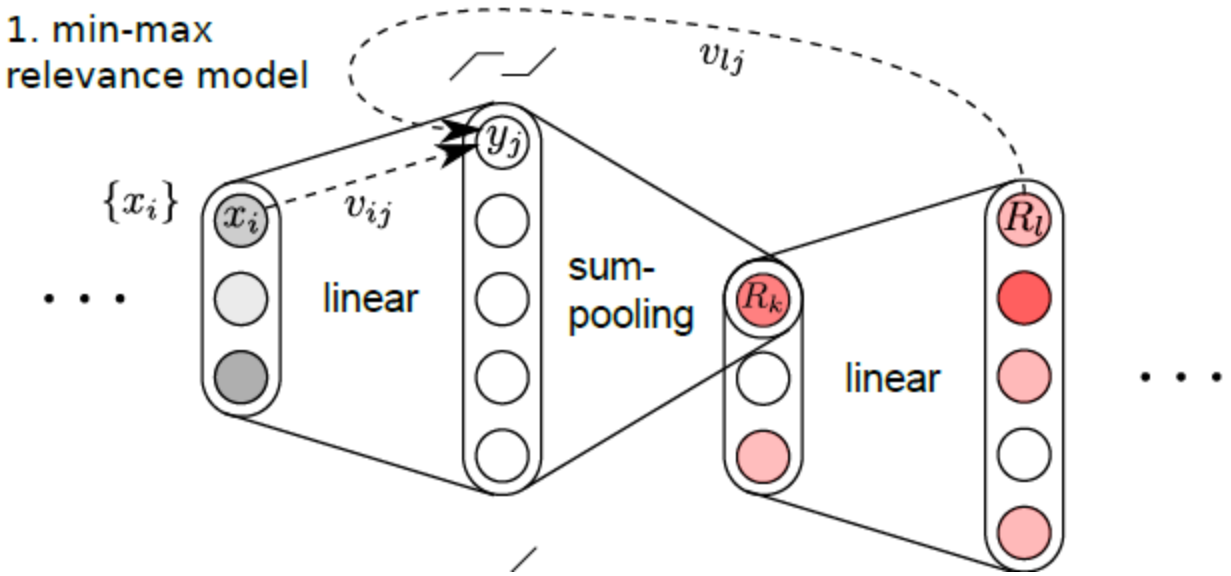
$Rule1 : z^b$-rule $\mathcal{X} = \mathcal{B}$

For all functions $g \in G$ and data points $\{x_i\} \in \mathcal{B}$, the deep Taylor decomposition with the z^b-rule is consistent.

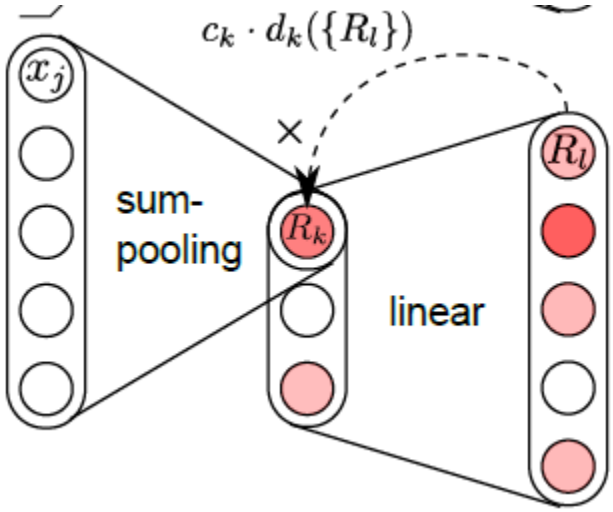
Based on the equation derived, the model can actually be refined into two models.

Two relevance models:

1. Min-Max Relevance Model



2. Training-Free Relevance Model



The simulation study shows that this method has both advantages and disadvantages.

Advantages:

1. Solved the problem of difficulty of finding a root point with Taylor Decomposition

2. Solved the problem of Gradient Shattering. Balduzzi et al. (2017)

Limitations:

1. Lack of efficiency in identifying features insignificantly different in importance.

2. Rules need to be chosen when specifying the domain of the input.

4.3 Intrinsic methods

4.3.1 Explainable Decision Forest: Transforming a Decision Forest into an Interpretable Tree (Litong Liu)

- Decision forest is favorable because of its high degree of accuracy and robustness to different sample size and feature space. However, the classification of decision forest is inefficient compared to single classifier models and it is hard to explain the rationale behind the classification of decision forest. Thus the author proposes a novel method to transform a decision forest into a decision tree, which can approximate the predictive performance of the original decision forest with high interpretability and faster predictions. It is suitable for any size forest and does not require complex hyperparameter turning.
- Given a dataset of n examples, m features and c different classes: $D = \{(x_i, y_i) : |D| = n, x_i \in R^n, y_i \in \{1, ..., c\}\}$;
    - Decision forest aggregates addictive functions and maps an m-dimensional feature vector into a c-dimensional probability vector: $\phi(x_i) = \frac{\sum_{k=1}^{|T|} t_k(x)}{|T|}, t_k \in R^c$
    - The proposed method aims to build a new tree $\hat{t}$ that: $\hat{t}(x_i) \approx \phi(x_i), \forall x_i$, which can be achieved in two steps: building a set of rule conjunctions from the given decision forest, and then organizing the set of conjunctions in a tree structure that will enable fast predictions for unseen instances.
    - To get the conjunction set of a decision forest, it requires people to do merging, which is unrealistic for arbitrary size of forest. To solve this problem, the author defines a threshold L as maximum allowed conjunctions in each iteration, and estimate the probability of a given conjunction $c_{ij} = \{r_1 \wedge ... \wedge r_n\}$ as a product of its rule's independent probabilities: $P(c_{ij}) = P(r_1)...P(r_n)$, where $P(r_n)$ is the empirical

probability of having the rule in the training set. In each iteration, only include the top L iterations in terms of conjunction probability. The time complexity for the algorithm is $O(LK)$.

- For a given rule r that split a conjunction set CS_i into two conjunctions sets CS_{i1} and CS_{i2}, define the information gain of the split as:

  $$IG(CS_i, R) = \frac{|CS_{i1}|entropy(CS_{i1}) + |CS_{i2}|entropy(CS_{i2})}{|CS_{i1}| + |CS_{i2}|} - entropy(CS_i)$$;

- The complete algorithm to generate forest based tree is as following: H

Algorithm 1: Generate a decision tree using an existing decision forest.

Input: T (A set of n decision trees $\{t_0, t_1, \ldots t_n\}$), $D_{pruning}$ (pruning dataset), $forest\_min\_size$ (minimum forest size), L (Maximum size of conjunctions at each iteration)

Output: DT (a decision tree)

$T' = \{\}$ - Pruned decision forest, $AUC_{T'}(D_{pruning}) = 0$
while $AUC_{T'}(D_{pruning})$ *was improved or* $|T'| < forest\_min\_size$ **do**
    Add to T' the decision tree t from T that maximizes $AUC_{T'}(D_{pruning})$
end
for *each decision tree* t_i *in* T' **do**
    if $i = 0$ **then**
        CS_0 =conjunction set of t_0 ($CS_i = \{(c_{i1}, \hat{y}_{c_{i1}}), (c_{i2}, \hat{y}_{c_{i2}}) \ldots (c_{iJ}, \hat{y}_{c_{iJ}})\}$ where c_{ij} is a conjunction of rules and $\hat{y}_{c_j}$ is a vector of classes probabilities
    end
    else
        $CS_i = \{\}$;
        for *leaf* c_j *in* t_i **do**
            for *conjunction* c_k *in* CS_{i-1} **do**
                if c_j *does not contradict* c_k **then**
                    add $(c_j \wedge c_k, \hat{y}_{c_j} + \hat{y}_{c_k})$ to CS_i
                end
            end
        end
        order CS_i by $P(c_{ij})$
        CS_i = top L conjunctions in CS_i
    end
end
Define DT as a single node that contains $CS_{|T'|}$
for *each node at DT that wasn't splitted or defined as a leaf* **do**
    for *each splitting candidate rule* r_{ij} *in* $\{r_{11}, r_{12} \ldots r_{iL}\}$ **do**
        Find Information Gain IG from split(CS_{node}, r_{ij})
    end
    define r' as the attribute with the highest information gain
    if $IG(r') = 0$ **then**
        define current node as a leaf
    end
    else
        split current node by r'
    end
end

4.3.2 Ensemble of Gradient Boosting Machines (Jeffrey Zhai)

Ensemble of gradient boosting machines (EGBM) is proposed by Konstantinov and Utkin (2021) that provides an improved interpretation method of traditional gradient boosting machines. It is an improved ML model based on Neural additive models. These kinds of models separate training on single features and then sum the shape functions. And the separate networks with inputs x_1, x_2, …, x_m are trained jointly using backpropagation. Gradient Boosting Machine is one of the most widely used neural additive models. And the proposed EGBM is an innovative version based on the traditional GBM which has the following algorithm:

Algorithm 1 The GBM algorithm

Require: Training set D; the number of the GBM iterations T
Ensure: Predicted function $g(\mathbf{x})$

1: Initialize the zero base model $g_0(x)$, for example, with the constant value.
2: **for** $t = 1, t \leq T$ **do**
3: Calculate the residual $q_i^{(t)}$ as the partial derivative of the expected loss function $L(y_i, g_t(\mathbf{x}_i))$ at every point of the training set, $i = 1, ..., N$, (the negative gradient)

$$q_i^{(t)} = -\left.\frac{\partial L(y_i, z)}{\partial z}\right|_{z=g_{t-1}(\mathbf{x}_i)} \tag{5}$$

4: Build a new base model (a regression tree) $h_t(\mathbf{x}_i)$ on dataset $\{(\mathbf{x}_i, q_i^{(t)})\}$
5: Find the best gradient descent step-size γ_t:

$$\gamma_t = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_t, g_{t-1}(\mathbf{x}_i) + \gamma h_t(\mathbf{x}_i)) \tag{6}$$

6: Update the whole model $g_t(\mathbf{x}) = g_{t-1}(\mathbf{x}) + \gamma_t h_t(\mathbf{x})$;
7: **end for**
8: Output

$$g_M(\mathbf{x}) = \sum_{t=1}^{T} \gamma_t h_t(\mathbf{x}) = g_{T-1}(\mathbf{x}) + \gamma_T h_T(\mathbf{x}). \tag{7}$$

For a traditional gradient boosting machine, it iteratively improves the predictions of y from x with respect to a specific loss function L. GBM adds new weak or base learners that improve upon the previous ones. In the proposed method EGBM could be seen as a weighted sum of separate GBMs, where each GBM depends on a single feature. Instead of an extremely randomized decision tree, EGBM uses partially randomized decision trees with depth 1 to avoid overfitting and reduce training time. In EGBM, each GBM computes functions $g^{(s)}(x_{i,k})$ of the kth feature in sth iteration. The process could be visualized into the following graph by Konstantinov and Utkin (2021).

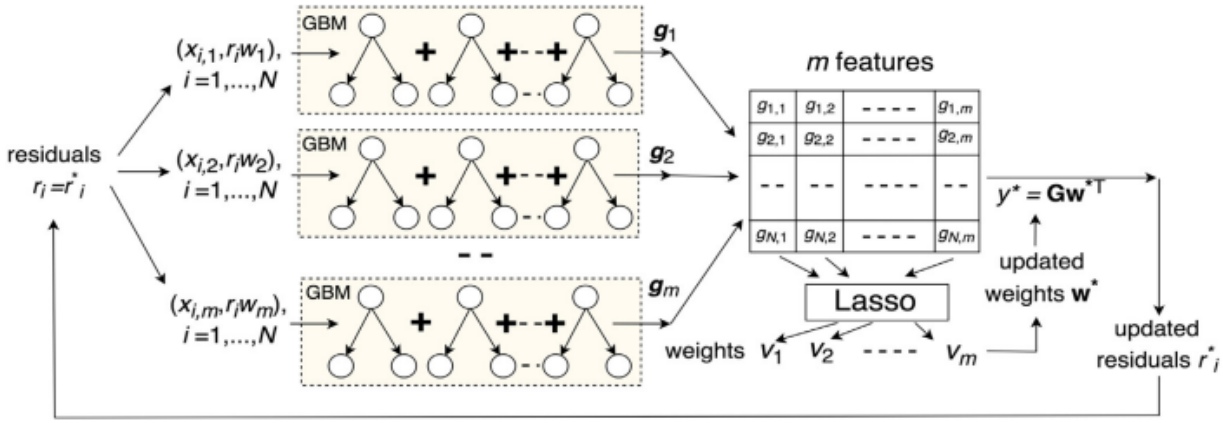


Fig. 2. A scheme of the proposed interpretation algorithm (the EGBM).

From the above graph, $r_i^{(s)}$represents the residuals. $w_k^{(s)}$ represents the weight. x_i represents the input data. After we finished running GBM on all inputs, we could get a N x m matrix of functions $g^{(s)}(x_{i,k})$ as $G^S = [g_1^{(s)}, \ldots, g_m^{(s)}]$. Then we update weights $v_k(s)$ using the Lasso method, and smooth the weight by $w_k^{(s)} = (1-\alpha)w_k^{(s-1)} + \alpha v_k^{(s)}$, k = 1, ..., m.

There are several stopping criteria for the Ensemble of GBM:

1. When the weights do not change or change is smaller than some deviation ϵ.
2. Use some predefined value T of iterations to stop EGBM.

The EGBM uses randomized decision trees of depth 1 as base models. According to the experiments completed by the author, the EGBM shows correct results with intuitive interpretations. And the EGBM is much faster than the traditional GBM with slightly higher accuracy. This method could be applied to both local and global interpretation with any models that use tabular data. Other data types such as image data need to be proven to be implemented with this EGBM. However, this method doesn't take the feature correlation into account. Thus for further direction, datasets with different feature correlation should be considered when applying EGBM.

4.3.3 Single-Index Model Tree (Jeffrey Zhai)

Single-Index Model Tree is proposed by Agus, Zebin, and Aijun (2021). It is an intrinsically interpretable model with relatively high accuracy. Current simple models such as KNN, Naive

Bayes can be interpreted easily but less accurate. Post-hoc diagnostic methods such as LIME or SHAP are just local approximations that may not be reliable. Therefore, using intrinsically interpretable models is an appealing choice.

Single-Index Models are simple prediction models using the function $y = h(w^T x) + \epsilon$, where ϵ is the zero-mean noise trem. w is the projection index. h is the univariate ridge function. When h is a linear function, SIM reduces to the linear regression model. Flexibility of h allows SIM to capture non-linear patterns. In this model, we assume datasets have homogeneous patterns. For large-scale heterogeneous datasets, we need to partition data into disjoint segments.

SIM Tree is a model-based tree using SIM as base learner. Thus it is formulated as

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} C_k(\boldsymbol{x}) h_k(\boldsymbol{w}_k^T \boldsymbol{x}),$$

The SIM tree split inputs recursively until predefined stopping criteria are satisfied. Since we have split data into homogeneous segments. Local base learners will have a higher probability to perform well. To find the optimal split variable and split points efficiently, the author proposed a fast SIM estimation methods as follows:

Algorithm 1 Fast Training Algorithm for SIMs

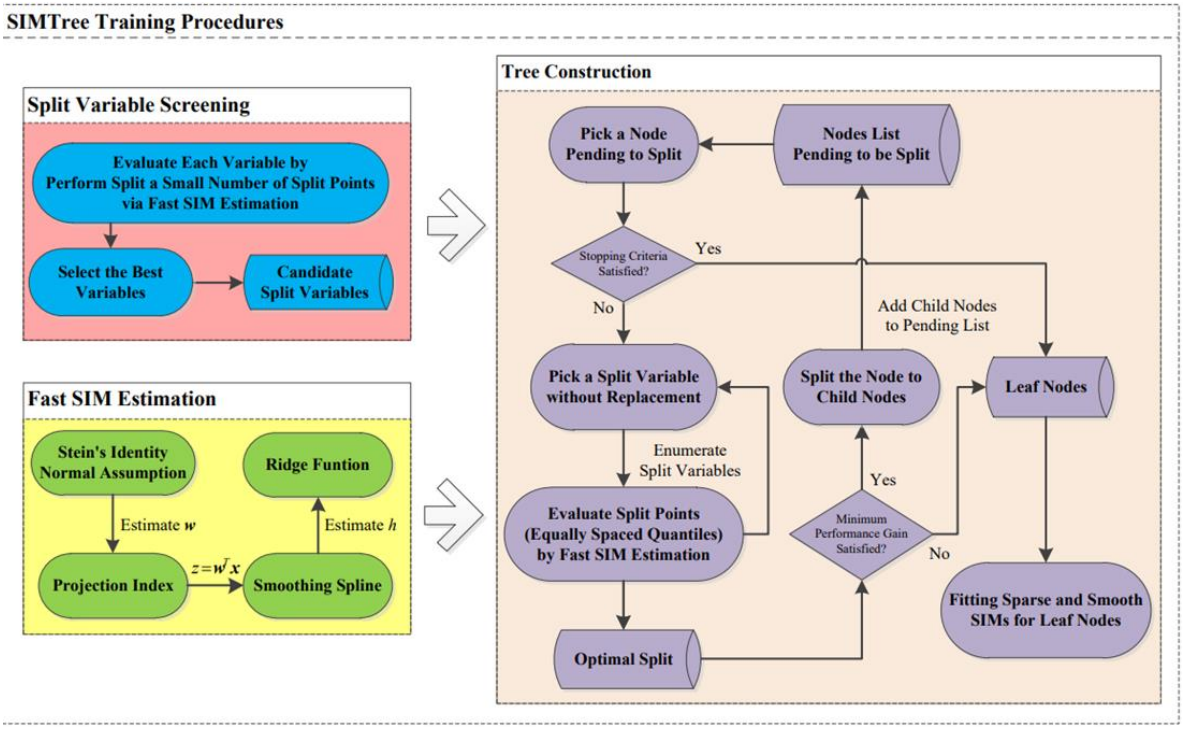
Require: $\{\boldsymbol{x}_i, y_i\}_{i\in[n]}$ (Training data), λ (Sparsity strength), γ (Smoothness strength).

if $\lambda = 0$ **then**
    Estimate the projection index $\hat{\boldsymbol{w}}$ by OLS estimator (7).
else
    Estimate $\hat{\boldsymbol{w}}$ via Lasso estimator (with λ).
end if
Calculate the projected data $z = \hat{\boldsymbol{w}}^T \boldsymbol{x}$.
Given $\{z_i, y_i\}_{i\in[n]}$, estimate $\hat{h}$ by smoothing spline (with γ).

In this algorithm, our purpose is to find the optimal h and w. The OLS estimator can be easily affected by noises. Thus, we induce sparsity strength and use Lasso estimator. In smoothing spline, h hat is expressed as a set of polynomial basis functions $\hat{h}(z) = \sum_{j=1}^{n} \hat{\beta}_j b_j(z)$.

There are several stopping criteria for SIMTree: 1. Maximum tree depth; 2. Minimum samples of leaf nodes; 3. Low performance improvement. In practice, the sparsity and smoothness strengths of each leaf node are fine-tuned by a 5-fold cross-validation grid search approach.

The overall SIMTree training procedures can be described using the following flowcharts:



The proposed SIMTree has several limitations. For some datasets, the predictive performance of SIMTree is close to GLMTree, however, its training takes slightly more time. Secondly, the splits used in SIMTree are all axis-oriented, which may be too restrictive for model expressiveness. For further direction, a possible solution is to introduce oblique splits. Thirdly, in the current implementation, a crude screening of candidate separation variables is performed to speed up the training, and a better screening strategy should be investigated in the future.

References

[1] Apley, D. W. and Zhu, J. (2016). Visualizing the effects of predictor variables in black box supervised learning models.

[2] Balduzzi, D., Frean, M., Leary, L., Lewis, J. P., Ma, K. W., and McWilliams, B. (2017). The shattered gradients problem: If resnets are the answer, then what is the question? CoRR, abs/1702.08591.

[3] Berk, R., Heidari, H., Jabbari, S., Joseph, M., Kearns, M., Morgenstern, J., Neel, S., and Roth, A. (2017). A convex framework for fair regression. arXiv preprint arXiv:1706.02409.

[4] Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M., and Eckersley, P. (2020). Explainable machine learning in deployment. Pages 648–657. Association for Computing Machinery, Inc.

[5] Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., and Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc.

[6] Bussmann, N., Giudici, P., Marinelli, D., & Papenbrock, J. (2020, September 25). Explainable Machine Learning in Credit Risk Management. *Computational Economics*. https://doi.org/10.1007/s10614-020-10042-0

[7] Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics.

[8] Cohen, W. W. (1995). Fast effective rule induction. In Machine learning proceedings 1995, pages 115–123. Elsevier.

[9] Fanaee-T, H. and Gama, J. (2014). Event labeling combining ensemble detectors and background knowledge. Progress in Artificial Intelligence, 2:113–127.

[10] Friedman, J. H. and Popescu, B. E. (2008). Predictive learning via rule ensembles. The annals of applied statistics, pages 916–954.

[11] Hastie, T., Tibshirani, R., and Friedman, J. (2001). The Elements of Statistical Learning. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

[12] Hastie, T., Tibshirani, R., and Friedman, J. (2013). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer New York.

[13] Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. Machine learning, 11(1):63–90.

[14] Konstantinov, A.V. and Utkin, L.V. (2021) “Interpretable machine learning with an ensemble of Gradient Boosting Machines,” *Knowledge-Based Systems*, 222, p. 106993. Available at: https://doi.org/10.1016/j.knosys.2021.106993.

[15] Letham, B., Rudin, C., McCormick, T. H., and Madigan, D. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. The Annals of Applied Statistics, 9(3):1350–1371.

[16] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

[17] Molnar, C. (2022). Interpretable Machine Learning. 2 edition.

[18] Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Muller, K. R. (2017). Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition, 65:211–222.

[19] Samadi, S., Tantipongpipat, U., Morgenstern, J. H., Singh, M., and Vempala, S. (2018). The Price of fair pca: One extra dimension.Advances in neural information processing systems,31.

[20] Sudjianto, A., Yang, Z. and Zhang, A. (2021) “Single-index model tree,” IEEE Transactions on Knowledge and Data Engineering, pp. 1–1. Available at: https://doi.org/10.1109/tkde.2021.3126615.

[21] Zhao, Q. and Hastie, T. (2021). Causal interpretations of black-box models. Journal of Business & Economic Statistics, 39(1):272–281.

[22] Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." *Advances in neural information processing systems* 29 (2016).

[23] Bellamy, Rachel KE, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia et al. "AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias." *IBM Journal of Research and Development* 63, no. 4/5 (2019): 4-1.

[24] Amini, Alexander, Ava P. Soleimany, Wilko Schwarting, Sangeeta N. Bhatia, and Daniela Rus. "Uncovering and mitigating algorithmic bias through learned latent structure." In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 289-295. 2019.