# Algorithm Bias and XAI Notes

April 29, 2023

**Abstract**

# 1 Introduction

# 2 Bias

Jie Li

## 2.1 Definition of Bias

**Definition:** Fairness is the absence of any prejudice or favoritism toward an individual or group based on their inherent or acquired characteristics.

## 2.2 Classical example of Bias

**Correctional Offender Management Proiling for Alternative Sanctions:** Judges use COMPAS to decide whether to release an offender or to keep him or her in prison.

## 2.3 Assessment tools

**Aequitas:**

1. let users test models with regard to several bias and fairness metrics for different population subgroups.
2. Aequitas produces reports from the obtained data that helps data scientists, machine learning researchers, and policymakers to make conscious decisions and avoid harm and damage toward certain populations.

**AI Fairness 360:**

1. This is another toolkit developed by IBM to help moving fairness research algorithms into an industrial setting and to create a benchmark for fairness algorithms to get evaluated and an environment for fairness researchers to share their ideas.

## 2.4 Types of bias

Bias can exist in many shapes and forms, some of which can lead to unfairness in different learning tasks.

**Data to algorithm:**

1. **Measure Bias:** Measurement, or reporting, bias arises from how we choose, utilize, and measure particular feature.(Harini Suresh and John V.,2019)

2. **Omitted Variable Bias:** Omitted variable bias occurs when one or more important variables are left out of the model.

3. **Representation Bias:** Representation bias arises from how we sample from a population during data collection process.(Harini Suresh and John V.,2019) Non-representative samples lack the diversity of the population, with missing subgroups and other anomalies.

4. **Aggregation Bias:** Aggregation Bias arises when false conclusions are derived about individuals from observing the entire population.(Harini Suresh and John V.,2019)

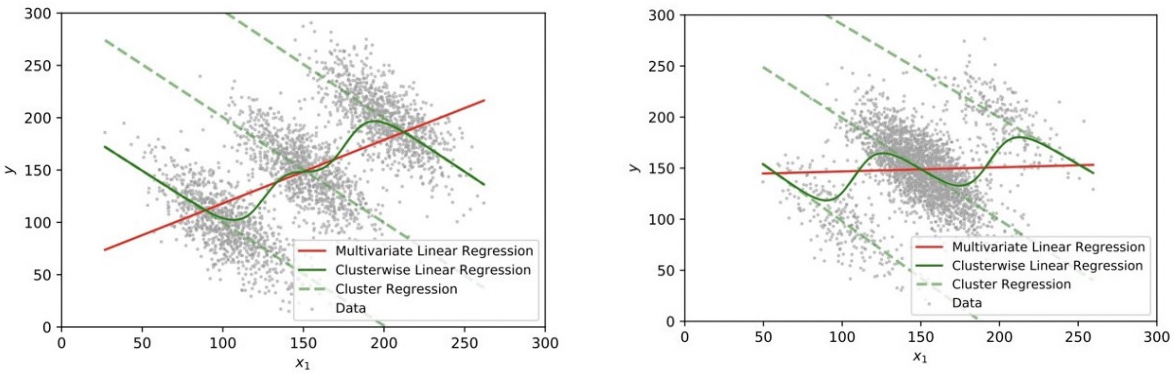



Figure 1: Aggregation bias

The red line shows the regression for the entire population, while dashed green lines are regressions for each subgroup, and the solid green line is the unbiased regression. (a) When all subgroups are of equal size, then regression shows a positive relationship between the outcome and the independent variable. (b) Regression shows almost no relationship in less balanced data. The relationships between variables within each subgroup, however, remain the same. (Credit: Nazanin Alipourfard.)

5. **Sampling Bias:** sampling bias is similar to representation bias, and it arises due to non-random sampling of subgroups. (Mehrabi, Ninareh,2021)

6. **Linking Bias:** Linking bias arises when network attributes obtained from user connections, activities, or interactions differ and misrepresent the true behavior of the users

**Algorithm to User:** Algorithms modulate user behavior. Any biases in algorithms might introduce biases in user behavior.

1. **Algorithmic Bias:** Algorithms bias is when the bias is not present in the input data and is added purely by the algorithm.(Ricardo Baeza-Yates. 2018)

2. **User interaction bias:** User interaction bias is a type that can not only be observant on the Web but also get triggered from two sources-the user interface and through the user itself by imposing his/her self-selected biased behavior and interaction.(Ricardo Baeza-Yates. 2018)

3. **Emergent bias:** Emergent bias occurs as a result of use and interaction with real users. This bias arises as a result of change in population, cultural values, or societal knowledge usually some time after the completion of design. (Batya Friedam and Helen Nissenbaum. 1996)

4. **Evaluation bias:** Evaluation bias happens during model evaluation

## 2.5 Bias loop:

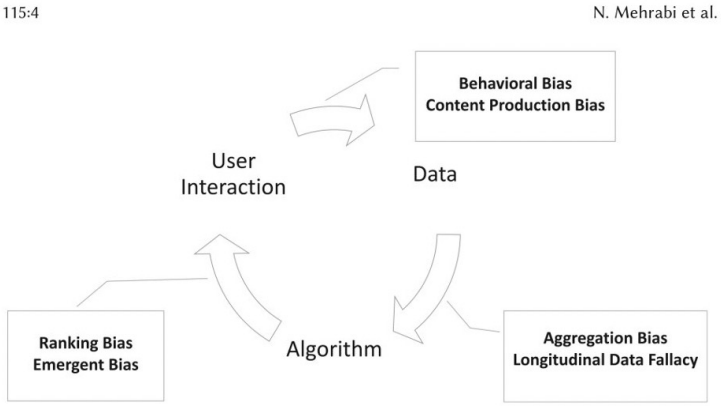



Figure 2: Bias loop

1. Most AI systems and algorithms are data driven and require data to be trained. Thus, data is tightly coupled to the functionality of these algorithms and systems.

2. If underlying training data contains biases, the algorithms trained on them will learn these biases and reflect them into their predictions. As a result, existing biases in data can affect the algorithms using the data, producing biased outcomes.

3. Algorithms can amplify and perpetuate existing biases in the data.

4. The outcomes of these biased algorithms can then be fed into real-world systems and affect users' decisions, which will result in more biased data for training future algorithms.

## 2.6 Methods for fair machine learning

**Pre-processing:** Pre-processing techniques try to transform the data so the underlying discrimination is removed. If the algorithm is allowed to modify the training data, then pre-processing can be used.

**In-processing:** In-processing techniques try to modify and change state-of-the-art learning algorithms to remove discrimination during the model training process. If we can not change the dataset, then we can use in-processing techniques to debias.

**Post-processing:** Post-processing is performed after training by accessing a holdout set that was not involved during the training of the model. 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 Bolukbasi et al. (2016).

| Algorithm | Pre-processing | In-processing | Post-processing |
|---|---|---|---|
| Community detection | ✓ | | |
| Word embedding | ✓ | | |
| Optimized pre-processing | ✓ | | |
| Data pre-processing | ✓ | | |
| Classification | | ✓ | |
| Regression | | ✓ | |
| Classification | | ✓ | |
| Classification | | ✓ | |
| Adversarial learning | | ✓ | |
| Classification | | | ✓ |
| Word embedding | | | ✓ |
| Classification | | | ✓ |

Figure 3: Methods for fair machine learning

### 2.6.1 Fair Regression

Reference Berk et al. (2017) proposes a fair regression method along with evaluating it with a measure introduced as the "price of fairness"(POF) to measure accuracy-fairness tradeoffs.

**Individual Fairness:** The definition for individual fairness, as stated in reference Berk et al. (2017), "for every cross pair$(x, y) \in S_1, (x', y') \in S_2$, a model $w$ is penalized for how differently it treats $x$ and $x'$ where $S_1$ and $S_2$ are different groups from the sampled population." Formally, this is operationalized as:

$$f_1(w, S) = \frac{1}{n_1 n_2} \sum_{\substack{(x_i, y_i) \in S_1 \\ (x_j, y_j) \in S_2}} d\left(y_i, y_j\right) \left(w \cdot x_i - w \cdot x_j\right)^2$$

**Group Fairness:** "On average, the two groups' instances should have similar labels( weihjted by the nearness of the labels of the instances)" Berk et al. (2017):

$$f_2(w,S) = \left( \frac{1}{n_1 n_2} \sum_{\substack{(x_i,y_i)\in S_1 \\ (x_j,y_j)\in S_2}} d\left(y_i,y_j\right)\left(w\cdot x_i - w\cdot x_j\right)^2 \right)$$

**Hybrid Fairness:** "Hybrid fairness requires both positive and both negatively labeled cross pairs to be treated similarly in an average over the two groups: Berk et al. (2017):

$$f_3(w,S) = \left( \sum_{\substack{(x_i,y_i)\in S_1 \\ (x_j,y_j)\in S_2 \\ y_i=y_j=1}} \frac{d\left(y_i,y_j\right)\left(w\cdot x_i - w\cdot x_j\right)}{n_{1,1}n_{2,1}} \right)^2 + \left( \sum_{\substack{(x_i,y_i)\in S_1 \\ (x_j,y_j)\in S_2 \\ y_i=y_j=-1}} \frac{d\left(y_i,y_j\right)\left(w\cdot x_i - w\cdot x_j\right)}{n_{1,-1}n_{2,-1}} \right)^2$$

### 2.6.2 Structured prediction

**RBA(reducing bias amplification:** RBA is a technique for debiasing models by calibrating prediction in structured prediction. The idea behind RBA is to ensure that the model predictions follow the same distribution in the training data.

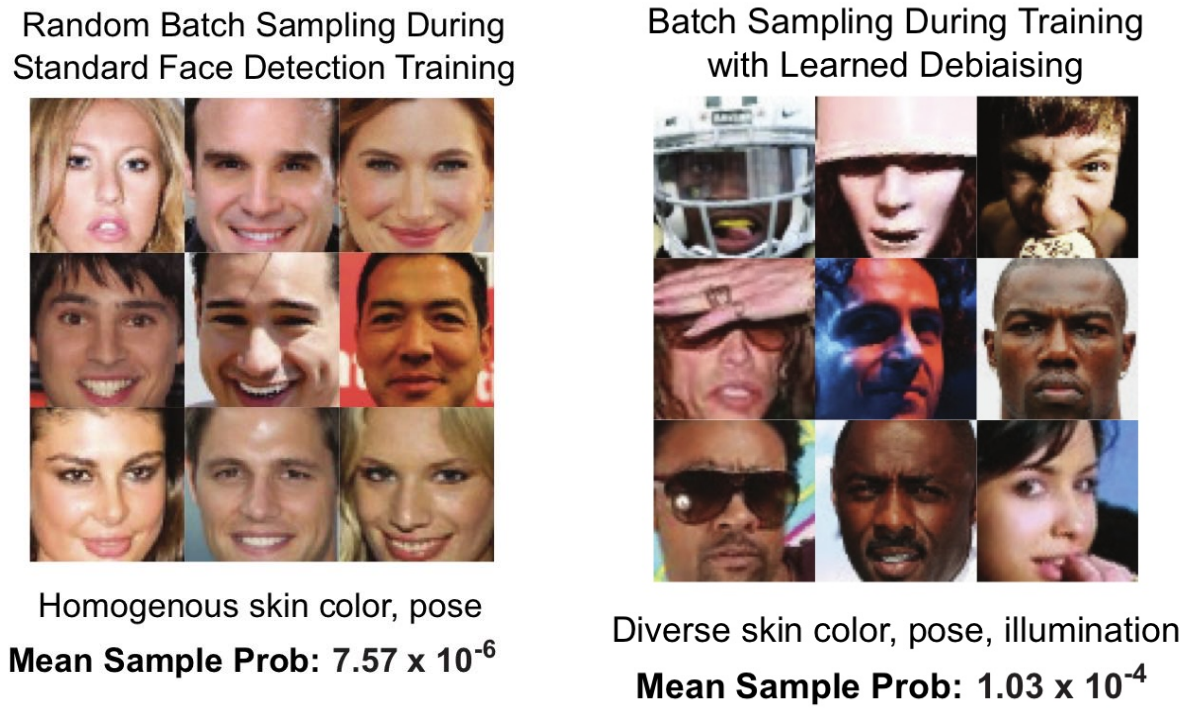
### 2.6.3 Fair PCA

In reference Samadi et al. (2018) authors show that vanilla PCA can exaggerate the error in reconstruction in one group of people over a diferent group of equal size, so they propose a fair method to create representations with similar richness for diferent populations—not to make them indistinguishable or to hide dependence on a sensitive or protected attribute. They show that vanilla PCA on the labeled faces in the wild (LFW) dataset has a lower reconstruction error rate for men than for women faces, even if the sampling is done with an equal weight for both genders. They intend to introduce a dimensionality reduction technique that maintains similar idelity for diferent groups and populations in the dataset. Therefore, they introduce Fair PCA and deine a fair dimensionality reduction algorithm. Their deinition of Fair PCA (as an optimization function) is as follows, in which $A$ and $B$ denote two subgroups, $U_A$ and $U_B$ denote matrices whose rows correspond to rows of $U$ that contain members of subgroups A and B given m data points in $R^n$:

$$\min_{U\in R^{m\times n},\mathrm{rank}(U)\leq d} \max\left\{ \frac{1}{|A|}\,\mathrm{loss}\left(A,U_A\right), \frac{1}{|B|}\,\mathrm{loss}\left(B,U_B\right) \right\}$$

## 2.7 Uncovering and Mitigating Algorithmic Bias through Learned Latent Structure

### 2.7.1 Facial Recognition




(a) (b)

Figure 4: Facial Recognition

Figure (4): Batches sampled for training without (left) and with (right) learned debiasing. The proposed algorithm iden- tiies, in an unsupervised manner, under-represented parts of training data and subsequently increases their respective sampling probability. The resulting batch (right) from the CelebA dataset shows increased diversity in features such as skin color, illumination, and occlusions.

### 2.7.2 VAE (Variational autoencoder)

**VAE(Variational autoencoder):** VAE, which is built on top of a variational autoencoder(VAE), is capable of identifying under-represented examples in the training dataset and subsequently increasing the probability at which the learning algorithm samples these data points.

### 2.7.3 Methodology

Consider the problem of binary classiication 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 $f : X \to Y$ parameterized by  which minimizes a certain loss $L(\theta)$ over our entire training dataset. In other words, we seek to solve the following optimization problem:

# 3 Explanation and Interpretation (Carvalho et al. (2019))

Virgil Chen

## 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 makes 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 an explainer 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 are the explanations between them. If the explanations are very similar, the explanations are highly consistent.

4. **Stability:** It represents how similar are the explanations 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:

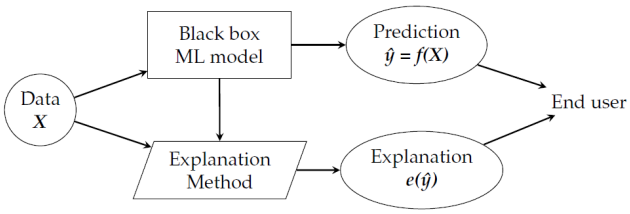



Figure 5: XAI Pipeline

# 4 Back-propagation Method

Virgil Chen

## 4.1 Deep-Taylor decomposition

**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)

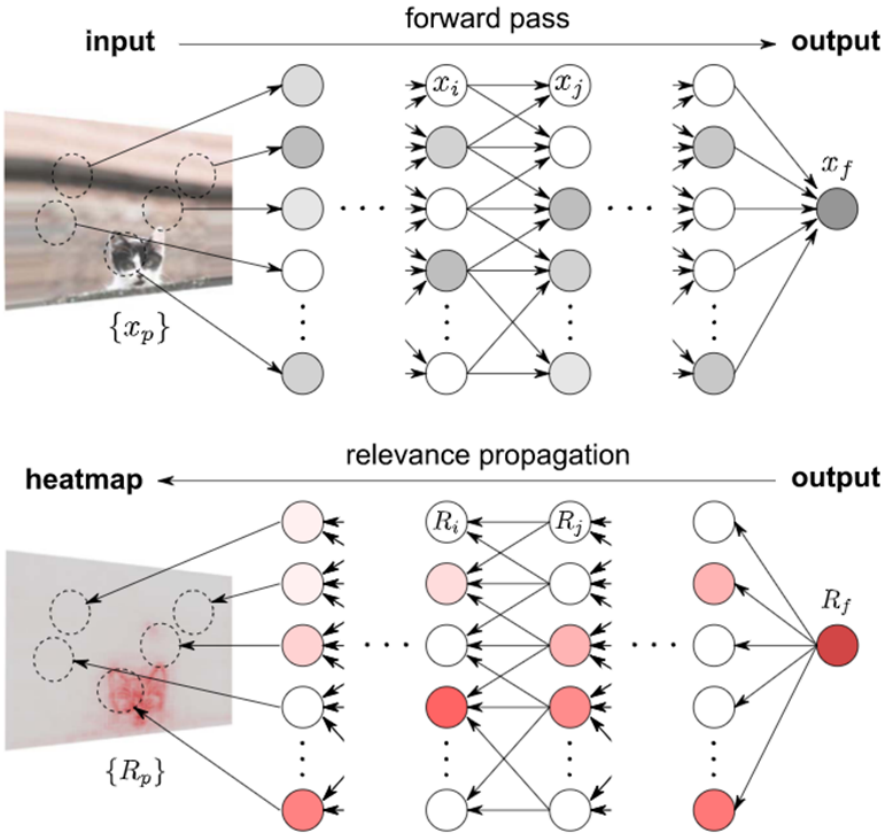



Figure 6: Back Propagation

**Computation:**

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

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

**Deep Taylor Pipeline:**

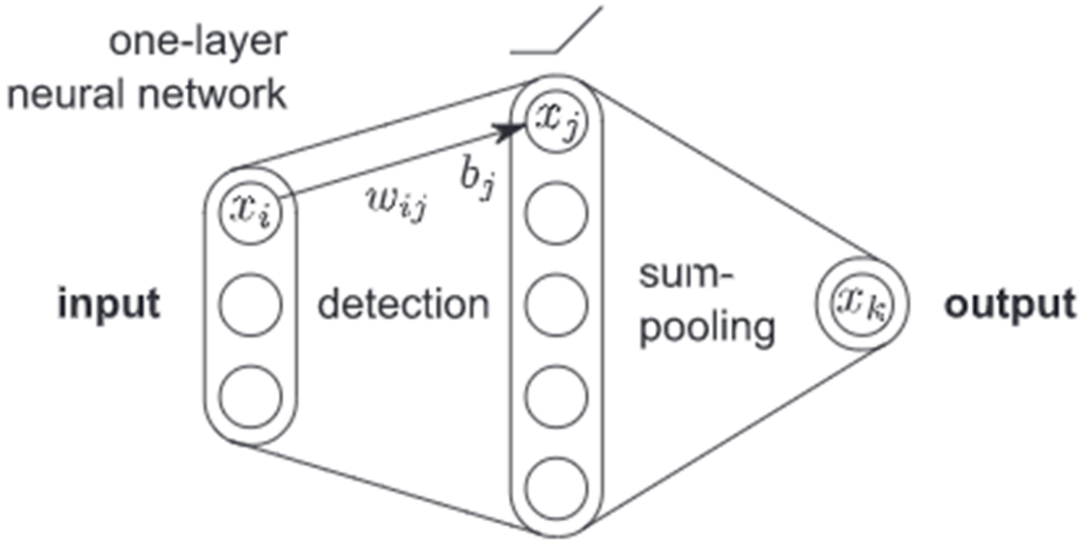



Figure 7: Deep Taylor Pipeline

**Three different rules:**

1. $Rule1 : w^2$-rule $\mathcal{X} = \mathbb{R}^d$
   For all functions $g \in G$, the deep Taylor decomposition with the $\omega^2$-rule is consistent.

2. $Rule1 : z^+$-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.

3. $Rule1 : z^b$-rule $\mathcal{X} = \mathcal{B}$
   For all function $g \in G$ and data points $\{x_i\} \in \mathcal{B}$, the deep Taylor decomposition with the $z^b$-rule is consistent.

**Two relevance models:**

1. Min-Max Relevance Model

2. Training-Free Relevance Model

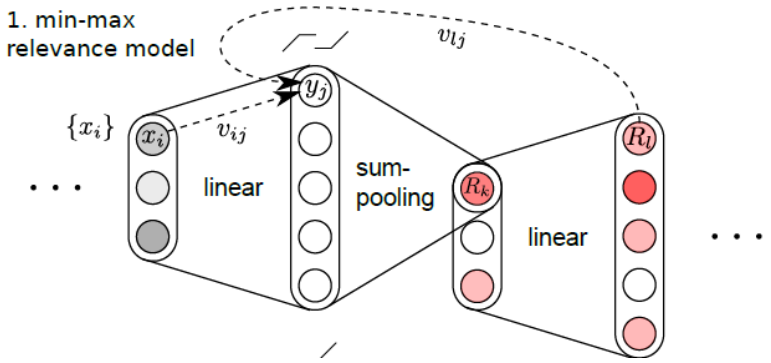



Figure 8: Min-Max Relevance Model

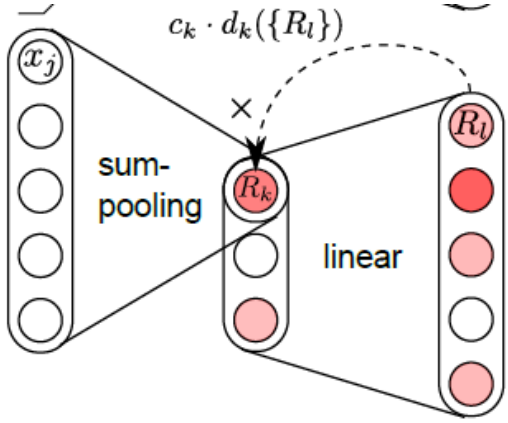



Figure 9: Training-Free Relevance Model

**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 from when specifying the domain of the input.

# 5 Model-Agnostic Methods

Jeffrey Zhai

## 5.1 Local Model-Agnostic Methods

### 5.1.1 Shapley Values

**Definition:** According to Molnar (2022), Shapley value refers to the average expected marginal contribution of one player after all possible combinations have been considered.

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

**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 features, we could find the contribution of each feature to the result of 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).

### 5.1.2 Shapley additive explanations

**Definition:** Shapley additive explanations (SHAP) is introduced by (Lundberg and Lee, 2017) to explain individual predictions. Comparing to method using shapley values, SHAP provides a kernel-based estimation method KernelSHAP for shapley values. Besides, the authors also proposed TreeSHAP, an efficient estimation approach for tree-based models.

**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 $\phi_k$, 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:** Acoording 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. Comparing 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 $E_{X_S|X_C}(\hat{f}(x)|x_S)$. One problem of TreeSHAP is that it may generate non-zero estimate 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.

### 5.1.3 Counterfactual Explanations

**Definition:** 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 interpretable machine learning, counterfactual explanations can be used to explain the predictions of individual instances. An "event" is the predicted outcome of an instance, and a "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.

**Criteria of Counterfactual Explanations:**

1. A counterfactual instance produces the predefined prediction as closely as possible.
2. A counterfactual should be as similar as possible to the instance regarding feature values.
3. A counterfactual should generate multiple diverse counterfactual explanations.
4. A counterfactual instance should have feature values that are likely.

### 5.1.4 Local interpretable model-agnostic explanations (LIME)

**Definition:** Local interpretable model-agnostic explanations is a widely used interpretable method that are used to explain individual predictions of black box machine learning models (Montavon et al., 2017). The authors proposed this method as a concrete implementation of local surrogate models.

The nature of LIME is the variations of the dataset in the machine learning model. For each new dadtaset, LIME generates a perturbed sample and the corresponding predictions from the black box model. On this specific dataset, LIME trains any selected interpretable model (linear regression model, decision tree, etc.) to get a result that is weighted by the proximity of the sampled observations to the observations of our interest.

In mathematical formula, LIME can be expressed as the following equation:

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

From the formula, the explanation model for an instance x is the ML model g that minimizes a loss function L. $\Omega(g)$ is the model complexity as a penalty term. G is the family of every possible explanations. $\pi_x$ is the proximity that measures how large the neighborhood around the instance x.

In practice, LIME consists of the following steps:

1. Find the instance of interest, which is the one you want to be explained.
2. Perturb the dataset and generate the predictions for these new dataset.
3. Weigth the new dataset according to the proximity defined above.
4. Train an weighted, interpretable model on the dataset with the variations.
5. Interpret the results of the prediction of the local model.

## 5.2 Global Model-Agnostic Methods

### 5.2.1 Partial Dependence Plot

**Definition:** 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. Heterogeneous effects might be hidden because PD plots only show the average marginal effects.

### 5.2.2 Accumulated Local Effects Plot

**Definition:** Accumulated local effects describe how features affect the predictions of a machine learning model on average (Apley and Zhu, 2016). The ALE plot is a faster and unbiased alternative to the partial dependency plot (PDP).

If features of a machine learning model are correlated, the partial dependence plot cannot be trusted. In ALE plots, we calculate the differences in predictions instead of averages (based on the conditional distribution of the features).

Partial dependence plots average the predictions over the marginal distribution. But ALE plots average the changes in the predictions and accumulate them over the grid.

**Advantages:**

1. ALE plots are unbiased, which means they still work when features are correlated.
2. ALE plots are faster to compute than PDPs and scale with O(n), since the largest possible number of intervals is the number of instances with one interval per instance.
3. The interpretation of ALE plots is clear and easy to understand.
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. An interpretation of the effect across intervals is not permissible if the features are strongly correlated.
2. ALE effects may differ from the coefficients specified in a linear regression model when features interact and are correlated.
3. The implementation of ALE plots is much more complex and less intuitive compared to partial dependence plots.

# 6 Interpretable Methods

Litong Liu

According to Molnar (2022), One way to achieve interpretability by using subset of interpretable models. In this section, we introduce the common types of interpretable models, with a focus on how to interpret them. The following table summarize the interpretable model types and corresponding properties, including linearity, monotonicity, interactions, and the task types that the model can handle. (Monotonicity can guarantee the relationship between a feature and the target outcome always goes in the same direction over the entire range of the feature. ) Hastie et al. (2013)

| Algorithm | Linear | Monotone | Interaction | Task |
|---|---|---|---|---|
| Linear regression | Yes | Yes | No | regr |
| Logistic regression | No | Yes | No | class |
| Decision trees | No | Some | Yes | class,regr |
| RuleFit | Yes | No | Yes | class,regr |
| Naive Bayes | No | Yes | No | class |
| k-nearest neighbors | No | No | No | class,regr |

Figure 10: Overview of interpretable models

## 6.1 Linear Regression

**Definition:** Linear regression models are the models that predict the target as a weighted sum of feature inputs, making the learned relationship easy to interpret. The following is the mathematic representation for linear regression model:

$$y = \beta_0 + \beta_1 x_+ ... + \beta_p x_p + \epsilon \tag{1}$$

The predicted outcome of an instance is a weighted sum of its p features, where coefficients ($\beta_j$) represent the learned feature weights. The first term $\beta_0$ is called the intercept, which represents the bias term. $\epsilon$ is the error of the regression, i.e. the difference between the prediction and actual outcome. Usually, we assume $\epsilon$ follows a Gaussian distribution, meaning that we can make both positive and negative errors, and most of them have small values, only a few have large values.

**Estimate $\beta$:** There are multiple ways to get the coefficients, the commonly used one is the ordinary least squares method:

$$\hat{\beta} = argmin_{\beta_0,...,\beta_p} \sum_{i=1}^{n} (y^{(i)} - (\beta_0 + \sum_{j=1}^{p} \beta_j x_j^{(i)}))^2 \tag{2}$$

**Interpretation:** The interpretation of a weight in the linear regression model depends on the type of the corresponding feature.

1. Numerical feature: Increasing the numerical feature by one unit changes the estimated outcome by its weight, i.e. an increase of feature $x_k$ by one unit increases the prediction for $y$ by $\beta_k$ units when all other feature values remain fixed.

2. Binary feature (A feature that takes one of two possible values of each instance): Changing the feature form the reference category to other category changes the estimated outcome by the feature's weight, i.e. changing feature $x_k$ from the reference category to the other category increases the prediction for $y$ by $\beta_k$ when all other features remain fixed.

3. Categorical feature with multiple categories (A feature with a fixed number of possible values): To deal with many categories is the one-hot-encoding, meaning that each category has its own binary column. For a categorical feature with L categories, only need L-1 columns, and then the interpretation for each feature is same as the interpretation for binary features.

4. Intercept $\beta_0$: For an instance with all numerical feature values at zero and the categorical feature values at the reference categories, the model prediction is the intercept weight $\beta_0$.

Another important measurement for interpreting linear models is the R-squared measurement, which can tell how much of the total variance of target outcome can be explained by the model. The higher R-squared, the better model explains the data.

$$R^2 = 1 - SSE/SST \tag{3}$$

SSE (the squared sum of error terms) :

$$SSE = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 \tag{4}$$

SST (the squared sum of data variance) :

$$SST = \sum_{i=1}^{n} (y^{(i)} - \bar{y})^2 \tag{5}$$

SSE tells how much variance remains after fitting the linear model, while SST is the total variance of the outcome. Sometimes, it happens that $R^2 < 0$, which implies that a model does not capture the trend of data and fits to the data worse than using the mean of target as the prediction.

However, by the definition of $R^2$, it is obvious that $R^2$ increases with the number of features in the model. Thus, it is better to use the adjusted R-squared:

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{6}$$

where p is the number of features and n is the total number of instances.

**Feature Importance:** The importance of a feature in a linear regression model can be measured by the absolute value of its t-statistic (SE represents the standard error):

$$t_{\hat{\beta}_j} = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)} \tag{7}$$

The importance of a feature increases with increasing weight. The more variance the estimated weight has, the less importance the feature is (correspondingly, t-statistic smaller).

## 6.2 Logistic Regression

**Theory:** Logistic function is defined as:

$$logistic(\alpha) = \frac{1}{1 + exp(-\alpha)} \tag{8}$$

For classification, use logistic function to transform linear regression to a probability between 0 and 1:

$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + ... + \beta_p x_p^{(i)}))} \tag{9}$$

**Interpretation:**

$$log(\frac{P(y=1)}{P(y=0)}) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p \tag{10}$$

Define odds as:

$$odds := \frac{P(y=1)}{1 - P(y=1)} = exp(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p) \tag{11}$$

When we increase one feature values by 1:

$$\frac{odds_{x_j+1}}{odds_{x_j}} = exp(\beta_j) \tag{12}$$

which means one unit change in $x_j$ increases the log odds ratio by the value of corresponding weight. For different feature types:

1. Numerical feature: Increasing the numerical feature by one unit changes the odd by a factor of $exp(\beta_j)$.

2. Binary feature (A feature that takes one of two possible values of each instance): Changing the feature form the reference category to other category changes the odd by a factor of $exp(\beta_j)$.

3. Categorical feature with multiple categories (A feature with a fixed number of possible values): To deal with many categories is the one-hot-encoding, meaning that each category has its own binary column. For a categorical feature with L categories, only need L-1 columns, and then the interpretation for each feature is same as the interpretation for binary features.

4. Intercept $\beta_0$: For an instance with all numerical feature values at zero and the categorical feature values at the reference categories, the estimated odds are $exp(\beta_0)$.

## 6.3 GLM, GAM and more

### 6.3.1 Non-Gaussian Outcomes - GLMs

The linear regression model assumes that the outcome given the input features follows a Gaussian distribution, which excludes many cases: the outcome can be a category, a count, or the time to the occurrence of an event. To solve this problem, we need to extend the linear regression model to Generalized Linear Models, or GLMs for short. The key concept for any GLM is: keep the weighted sum of the features, but allow non-Gaussian outcome distributions and connect the expected mean of this distribution and the weighted sum through a possibly nonlinear function.

The GLM mathematically links the weighted sum of the features with the mean value of the assumed distribution using the link function g, which can be chosen flexibly depending on the type of outcome:

$$g(E_Y(y|x)) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p \tag{13}$$

GLMs consist of three components: the link function $g$, the weighted sum (linear predictor) $X^T\beta$, and a probability distribution from the exponential family that defines $E_Y$. The GLM framework makes it possible to choose the link function independently of the distribution.

### 6.3.2 Interactions

The linear regression model assumes that the effect of one feature is the same regardless of the values of the other features (= no interactions), but it is often not true in reality. To include the interactions, we add a column to the feature matrix that represents the interaction between features and fit the model on this modified feature matrix as usual. For categorical features, we

create a new feature column that has value 1 if both features have a certain category, otherwise 0; and for two numerical features, the interaction column is constructed by simply multiply both numerical features. What is more, there are some approaches to automatically detect and add interaction terms, like RuleFit algorithm.

### 6.3.3 Nonlinear Effects - GAMs

Due to the complexity of real life data, linearity assumption in linear models may not hold. For example, increasing temperature from 10 Celsius to 15 may have positive effect on the number of bicycle rentals, while an increment from 35 Celsius to 40 may have opposite effect, and the linear model fails to work under this case.

To model the nonlinear relationships, one can use the following techniques:

- Simple transformation of features (e.g. logarithm)
- Categorization of the feature
- Generalized Additive Models (GAMs)

To visualize each of these methods, take the bicycle rental, from Fanaee-T and Gama (2014), as an example:

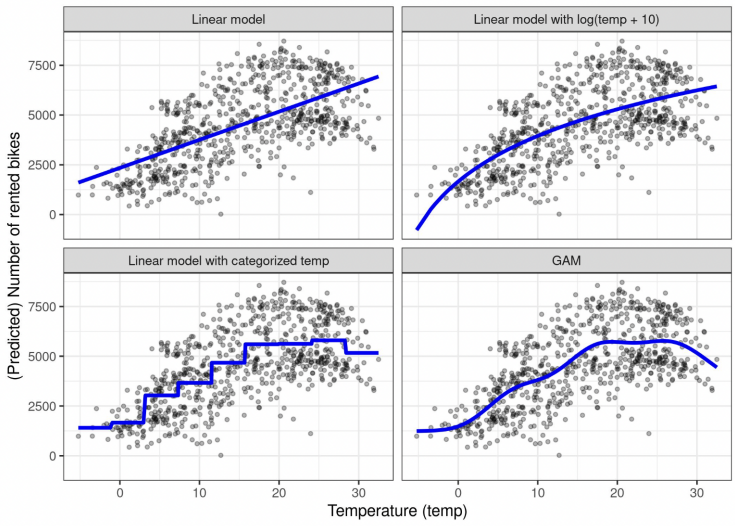



Figure 11: Visualization of bicycle rental example by nonlinear techniques

**Feature Transformation**: To do feature transformation, one should replace the column of the potential feature in data with a function of the feature, like logarithm or square root, and then fit the linear model as usual. The interpretation of the feature changes according to the selected transformation.

**Feature Categorization**: Another technique to achieve nonlinearity is to discretize the feature: turning it into a categorical feature. For discrete features, the linear model would estimate a step function. However, the feature categorization approach has several limitations, which is more likely to overfit and it is not clear how to discretize a feature meaningfully.

**Generalized Additive Models (GAMs)**: GAMs relax the restriction that relationship needs to be simple weighted sum, instead it assumes the outcome can be modeled by a sum of arbitrary functions of each feature. Mathematically speaking:

$$g(E_Y(y|x)) = \beta_0 + f_1(x_1) + ... + f_p(x_p) \tag{14}$$

One can use spline functions to learn the nonlinear functions in above equation. Splines are functions that are constructed from simple functions to approximate complex functions.

## 6.4 Decision Trees

When linear and logistic regression models fail, one can consider to use the decision tree. Tree based models split the data multiple times according to certain cutoff values in the features. Through splitting, different subsets of dataset are created, with each instance belonging to one subset. The final subset that only contains one node is called leaf node. To predict the outcome in each leaf node, the average outcome of the training data in the node is used. Trees can be used for both classification and regression. There are different algorithms that can grow a tree, here we focus on the most popular one: the classification and regression trees (CART) algorithm. By CART, the prediction outcome is:

$$\hat{y} = \hat{f}(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\} \tag{15}$$

where $I\{x \in R_m\}$ is the identity function that returns 1 if $x$ is in the subset $R_m$ and 0 otherwise. If instance falls into a leaf node $R_l$, the predicted outcome is $\hat{y} = c_l$, where $c_l$ is the average of all training instances in leaf node $R_l$.

More specifically, CART takes a feature and determines which cut-off point minimizes the variance of y for a regression task or Gini index of the class distribution of y for classification task. The best cut-off point makes the two resulting subsets as different as possible.

### 6.4.1 Interpretation

Starting form root node, go to the next nodes and the edges tell which subset it is. Once you arrive the leaf node, the node tells how you get the predicted outcome. All edges (checking conditions) are connected by "AND".

**Feature Importance**: the overall importance of a feature in a decision tree can be computed by go through all the splits for which feature was used and measure how much it has reduced the variance or Gini index compared to the parent node.

**Tree decomposition**: Individual predictions of a decision tree can be explained by decomposing the decision path into one component per feature. We can track a decision through the tree and explain a prediction by the contributions added at each decision node.

The root node in a decision tree is our starting point. If we were to use the root node to make predictions, it would predict the mean of the outcome of the training data. With the next split, we either subtract or add a term to this sum, depending on the next node in the path. To get to the final prediction, we have to follow the path of the data instance that we want to explain and keep adding to the formula.

$$\hat{f}(x) = \bar{y} + \sum_{d=1}^{D} split.contrib(d, x) = \bar{y} + \sum_{j=1}^{p} feat.contrib(j, x) \tag{16}$$

The prediction of an individual instance is the mean of the target outcome plus the sum of all contributions of the D splits that occur between the root node and the terminal node where the instance ends up.

**Advantages**: tree structure is ideal for capturing interactions between features in the data, and data ends up in distinct groups that are often easier to understand than points on a multi-dimensional hyperplane as in linear regression. Tree structure also has a natural visualization. **Disadvantages**: However, tree structures also have some disadvantages. Trees are inefficient when dealing with linear relationships, and it usually lack smoothness. Moreover, tree structures are not stable and tend to have overfitting problem.

## 6.5 Decision Rules

A decision rule is a simple IF-THEN statement consisting of a condition (also called antecedent) and a prediction. A single decision rule or a combination of several rules can be used to make predictions. Due to their standard structures, decision rules are one of the most interpretable prediction models.

A decision rule uses at least one feature=value statement in the condition, with no upper limit on how many more can be added with an 'AND'. An exception is the default rule that has no explicit IF-part and that applies when no other rule applies, but more about this later.

The usefulness of a decision rule can be summarized into two numbers: Support and accuracy.

- **Support**: The percentage of instances to which the condition of a rule applies is called the support.
- **Accuracy**: The accuracy of a rule is a measure of how accurate the rule is in predicting the correct class for the instances to which the condition of the rule applies.

Usually, there is a trade-off between accuracy and support: By adding more features to the condition, we can achieve higher accuracy, but lose support.

When running into problems of rules overlapping or no rule applies, one may want to consider to build a decision list (ordered) or a decision set (unordered).

**Decision list**: introduce an order to the decision rules. If the condition of the first rule is true for an instance, we use the prediction of first rule. If not, we go to the next rule and check if it applies and so on. Decision list solve the problem of overlapping rules by only returning the prediction of first rule in the list that applies.

**Decision set**: a decision set resembles a democracy of the rules, except that some rules might have a higher voting power. In a set, the rules are either mutually exclusive, or there is a strategy for resolving conflicts, such as majority voting, which may be weighted by the individual rule accuracies or other quality measures. Interpretability suffers potentially when several rules apply.

The next question is how to learn rules from data. Here, we list three of those methods that represent different approaches.

- **OneR** learns rules from a single feature. OneR is characterized by its simplicity, interpretability and its use as a benchmark.
- **Sequential covering** is a general procedure that iteratively learns rules and removes the data points that are covered by the new rule. This procedure is used by many rule learning algorithms.
- **Bayesian rule lists** combine pre-mined frequent patterns into a decision list using Bayesian statistics. Using pre-mined patterns is a common approach used by many rule learning algorithms.

**Algorithm of learn rules from s single feature (OneR)**, Holte (1993):

1. Discretize the continuous features by choosing appropriate intervals.
2. For each feature:
   - Create a cross table between the feature values and the (categorical) outcome.
   - For each value of the feature, create a rule which predicts the most frequent class of the instances that have this particular feature value (can be read from the cross table).
   - Calculate the total error of the rules for the feature.
3. Select the feature with the smallest total error.

OneR selects the one that carries the most information about the outcome of interest and creates decision rules from this feature. OneR always covers all instances of the dataset, since it uses all levels of the selected feature. Missing values can be either treated as an additional feature value or be imputed beforehand.

**Algorithm of Sequential Covering**:

Sequential covering is a general procedure that repeatedly learns a single rule to create a decision list (or set) that covers the entire dataset rule by rule. First, find a good rule that

applies to some of the data points. Remove all covered data points. A data point is covered when conditions apply, no matter whether the points are classified correctly or not. Repeat the rule-learning and removal of covered points until no points left or other stopping criteria is met. Algorithm of sequential covering is cited from Cohen (1995)

The algorithm is listed in the following:

- Start with an empty list of rules (rlist).
- Learn a rule r.
- While the list of rules is below a certain quality threshold (or positive examples are not yet covered):
  - Add rule r to rlist.
  - Remove all data points covered by rule r.
  - Learn another rule on the remaining data.
- Return the decision list.

**Algorithm of Bayesian Rule Lists**:

A Bayesian Rule List, proposed by Letham et al. (2015), can be learned by two steps:

- Pre-mine frequent patterns from the data that can be used as conditions for the decision rules.
- Learn a decision list from a selection of the pre-mined rules.

To draw an initial decision list:

- Pre-mine patterns with FP-Growth.
- Sample the list length parameter m from a truncated Poisson distribution.
- For the default rule: Sample the Dirichlet-Multinomial distribution parameter $\theta_0$ of the target value (i.e. the rule that applies when nothing else applies).
- For decision list rule j=1,...,m, do:
  - Sample the rule length parameter l (number of conditions) for rule j.
  - Sample a condition of length $l_j$ from the pre-mined conditions.
  - Sample the Dirichlet-Multinomial distribution parameter for the THEN-part (i.e. for the distribution of the target outcome given the rule)
- For each observation in the dataset:
  - Find the rule from the decision list that applies first (top to bottom).
  - Draw the predicted outcome from the probability distribution (Binomial) suggested by the rule that applies.

After getting the initial list, new decision lists are sampled by starting from the initial list and then randomly either moving a rule to a different position in the list or adding a rule to the current decision list from the pre-mined conditions or removing a rule from the decision list. Which of the rules is switched, added or deleted is chosen at random. At each step, the algorithm evaluates the posteriori probability of the decision list (mixture of accuracy and shortness).

## 6.6 RuleFit

To build a model that is simple and interpretable as linear model, but also integrates feature interactions, one can employ ReluFit. Like mentioned in Friedman and Popescu (2008), RuleFit learns a sparse linear model with the original features and also a number of new features that are decision rules. These new features capture interactions between the original features. RuleFit automatically generates these features from decision trees. Each path through a tree can be transformed into a decision rule by combining the split decisions into a rule. The node predictions are discarded and only the splits are used in the decision rules.

**Algorithm**: RuleFit can be divided into two components: the first component creates "rules" from decision trees and the second component fits a linear model with the original features and the new rules as input (hence the name "RuleFit").

**Step 1: Rule generation**

Rules are constructed by decomposing decision trees: Any path to a node in a tree can be converted to a decision rule. The trees used for the rules are fitted to predict the target outcome. (Therefore the splits and resulting rules are optimized to predict the outcome you are interested in.) Any tree ensemble algorithm can be used to generate the trees for RuleFit. A tree ensemble can be described with this general formula:

$$\hat{f}(x) = a_0 = \sum_{m=1}^{M} a_m \hat{f}_m(x) \tag{17}$$

where M is the number of trees and $\hat{f}_m(x)$ is the prediction function of the m-th tree.

The created rules all have the forms of:

$$r_m(x) = \prod_{j \in T_m} I(x_j \in s_{jm}) \tag{18}$$

where $T_m$ is the set of features used in the m-th tree. I is the indicator function that is 1 when feature $x_j$ is in the specified subset of values s for the j-th feature (as specified by the tree splits) and 0 otherwise. For numerical features, $s_{jm}$ is an interval in the value range of the feature.

In other words, RuleFit generates a new set of features from original features. These features are binary and can represent quite complex interactions of your original features. Rules are chosen to maximize the prediction task. Take them as new features based on the original features.

**Step 2: Sparse linear model**

After step 1, many rules are generated. Every rule and every original feature becomes a feature in the linear model and gets a weight estimate. The original raw features are added because trees fail at representing simple linear relationships between y and x. With the features, RuleFit generates a sparse linear model with Lasso with following structure:

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{k=1}^{K} \hat{\alpha}_k r_k(x) + \sum_{j=1}^{p} \hat{\beta}_j x_j \tag{19}$$

The result is a linear model that has linear effects for all of the original features and for the rules. The interpretation is the same as for linear models, the only difference is that some features are now binary rules.

It can still have an additional optional step for feature importance. Feature importance is measured with the standardized predictor: $I_j = |\hat{\beta}_j| std(x_j)$, where $\beta_j$ is the weight from Lasso model and *std* is the standard deviation over the data. For decision rules, the importance is: $I_k = |\hat{\alpha}_k| \sqrt{s_k(1 - s_k)}$, where $\hat{\alpha}_k$ is the associated Lasso weight of the decision rule and $s_k$ is the support of the feature in the data.

**Interpretation**: Since RuleFit estimates a linear model in the end, the interpretation is the same as for "normal" linear models. The only difference is that the model has new features derived from decision rules. Decision rules are binary features: A value of 1 means that all conditions of the rule are met, otherwise the value is 0. For linear terms in RuleFit, the interpretation is the same as in linear regression models: If the feature increases by one unit, the predicted outcome changes by the corresponding feature weight.

# References


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

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.

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*.

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.

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.

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

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

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

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

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.

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.

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

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.

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.

Molnar, C. (2022). *Interpretable Machine Learning*. 2 edition.

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

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.

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