

# Responsible Machine Learning with Insurance Applications

Christian Lorentzen & Michael Mayer

Autumn 2023

# Table of Contents

XAI: Introduction

XAI: Explaining Models

XAI: Improving Explainability

# Introduction

*How to explain and interpret a given model, even if it seems a black box?*

Answering this question is a key aspect of responsible ML

1. Information for stakeholders
2. Detect problems in modeling process

XAI: e**X**plainable **A**rtificial **I**ntelligence

Collection of methods to explain and interpret models

# Scope and Taxonomy

## Scope

XAI methods for structured data and **bold** aspects below

## Taxonomy of explainability

- ▶ **Global** vs. local: Describe model as a whole or around an observation.
- ▶ Model-specific vs. **model-agnostic**: Some methods are tailored to specific model classes (linear regression, tree-based), others work for all types of models.
- ▶ Intrinsic versus **post-hoc**: Simple models like a linear regression can be interpreted intrinsically, while complex models require post-hoc analysis of fitted model.

## Notes

- ▶ Model-agnostic methods are always post-hoc
- ▶ Model-agnostic methods can also be applied to intrinsically interpretable models
- ▶ Won't make difference between “explainable”, “interpretable”, “intelligible”

# XAI Outline

## 1. Introduction

- ▶ Notation
- ▶ Non-life insurance pricing
- ▶ Main example

## 2. Explaining Models

- ▶ Important post-hoc interpretation methods
- ▶ SHAP
- ▶ Improve GLM with the help of ML and XAI

## 3. Improving Explainability

Improve intrinsic explainability of complex models by simplifying their structure

# Notation

## Basic modeling situation

$$T(Y \mid \mathbf{x}) \approx m(\mathbf{x})$$

- ▶ Distributional property  $T(Y \mid \mathbf{X} = \mathbf{x}) = T(Y \mid \mathbf{x})$  of response  $Y$
- ▶ Model  $m : \mathbf{x} \in \mathbb{R}^p \mapsto \mathbb{R}$  of feature vector  $\mathbf{X} = (X^{(1)}, \dots, X^{(p)})$  with realization  $\mathbf{x} = (x^{(1)}, \dots, x^{(p)})$
- ▶  $m$  estimated by  $\hat{m}$  from training data by minimizing objective criterion  $\sum_{i=1}^n w_i L(\hat{y}_i, y_i) / \sum_{i=1}^n w_i + \lambda \Omega(m)$
- ▶  $L$ : loss/scoring function, ideally strictly consistent for  $T$ ;  $\lambda \Omega(m)$ : optional penalty
- ▶  $\mathbf{w} = (w_1, \dots, w_n)^T$ : vector of (optional) case weights
- ▶  $\mathbf{y} = (y_1, \dots, y_n)^T$ : observed values of  $Y$
- ▶  $\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_n)^T$ : predicted/fitted values  $\hat{y}_i = \hat{m}(\mathbf{x}_i)$
- ▶  $\mathbf{x}_1, \dots, \mathbf{x}_n$ :  $n$  feature vectors;  $x_i^{(j)}$ : value of  $j$ -th feature of  $i$ -th observation

# Examples of Models

- ▶ Linear regression
- ▶ Generalized linear models (GLM)
- ▶ Generalized additive models (GAM)
- ▶ Gradient boosted trees

Will peek into them as a quick refresher and to get used to notation

# Linear Regression

- ▶ Model equation postulates

$$\mathbb{E}(Y \mid \mathbf{x}) = m(\mathbf{x}) = \beta_o + \beta_1 x^{(1)} + \dots + \beta_p x^{(p)}$$

- ▶  $(\beta_o, \beta_1, \dots, \beta_p) \in \mathbb{R}^{p+1}$ : parameter vector to be estimated
- ▶ Objective: Minimize sum of squared errors

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2$$

by linear least-squares

- ▶ Non-linear effects and interactions have to be added manually
- ▶ Penalized regression?
- ▶ Important extension: the generalized linear model (GLM)



# Generalized Linear Model (GLM)

- ▶ Model equation postulates

$$\mathbb{E}(Y \mid \mathbf{x}) = m(\mathbf{x}) = g^{-1}(\eta(\mathbf{x})) = g^{-1}(\beta_o + \beta_1 x^{(1)} + \dots + \beta_p x^{(p)})$$

- ▶  $g^{-1}$ : inverse link,  $g$ : link function,  $\eta$ : linear predictor
- ▶ Parameters  $\beta_j$  estimated by minimizing the (possibly weighted) *average deviance*

$$\bar{S}(\hat{m}, D_{\text{train}}) = \sum_{i=1}^n w_i S(\hat{y}_i, y_i) / \sum_{i=1}^n w_i$$

over training data  $D_{\text{train}} = \{(y_i, w_i, \mathbf{x}_i), i = 1, \dots, n\}$

## (Unit) deviance

- ▶ Distribution-specific measure: Poisson, Gamma, Bernoulli, normal, ...
- ▶ In our examples, we will often work with Poisson deviance  
 $S(\hat{y}_i, y_i) = 2(y_i \log(y_i/\hat{y}_i) - (y_i - \hat{y}_i))$

# Generalized Additive Model (GAM)

- ▶ Extension of the GLM
- ▶ Model equation assumes

$$\mathbb{E}(Y \mid \mathbf{x}) = m(\mathbf{x}) = g^{-1}(\beta_o + f_1(x^{(1)}) + \cdots + f_p(x^{(p)}))$$

- ▶  $f_j$ : Sufficiently nice functions (some may be fully parameteric)
- ▶ Estimated to minimize average deviance, e.g. using backfitting
- ▶ Unlike a GLM, automatically accounts for non-linear effects
- ▶ Like a GLM, a GAM can also include interaction effects

# Gradient Boosted Trees

- ▶ Typical black box  $m$
- ▶ Sum of decision trees
- ▶ In contrast to GAM, automatically picks up interactions
- ▶ Can optimize same objective criterion as GLMs/GAMs
- ▶ Using a different model structure and a different optimization technique
- ▶ Important implementations: LightGBM, XGBoost

Later, we will also work with neural nets

# Non-Life Insurance Pricing

Main task: Predict *pure premium* of insurance policy

- ▶ Financial loss per year or per some other relevant exposure measure
- ▶ Used by company to optimize tariffs and to estimate expected future profit
- ▶ Predictions of statistical models fitted on historic data

## Discussion

Why is it important to have good tariff?

# Characterization of Insurance Policy

- ▶  $w > 0$ : The exposure. Other quantities will refer to this
- ▶  $N$ : Number of claims
- ▶  $C$ : Total claim amount
- ▶  $C/w$ : Pure premium
- ▶  $Y = N/w$ : Claims frequency
- ▶  $Z = C/N$ : Severity = avg cost per claim
- ▶  $\mathbf{X}$ : One or more risk characteristics

Example (fictive motor third-part liability (MTPL) policies)

id	$w$	$N$	$C$	$C/w$	$Y$	$Z$	Driver's age	Horse power
1	1	0	0	0	0	-	28	80
2	0.5	2	5000	10000	4	2500	20	250
2	0.5	1	1000	2000	2	1000	21	250

## Remark

Due to additivity of  $w$ ,  $N$ , and  $C$ , these quantities can also be defined for multiple policies together, e.g., for the entire portfolio

## Classic Pricing Models

- ▶ Instead of creating a model for  $\mathbb{E}(C/w \mid \mathbf{x})$ , decompose pure premium

$$C/w = (C/w) \cdot (N/N) = (N/w) \cdot (C/N) = YZ$$

into product of frequency  $Y$  and severity  $Z$

- ▶ Frequency model:  $\mathbb{E}(Y \mid \mathbf{x}) \approx m_Y(\mathbf{x})$   
→ Poisson GLM with log link and case weights  $w$
- ▶ Severity model:  $\mathbb{E}(Z \mid \mathbf{x}) \approx m_Z(\mathbf{x})$   
→ Gamma GLM with log link and case weights  $N$ , using only rows with  $N > 0$
- ▶ Assuming conditional independence of  $Y$  and  $Z$ , pure premium model is then  $\mathbb{E}(C/w \mid \mathbf{x}) \approx m_Y(\mathbf{x})m_Z(\mathbf{x})$

## Alternative to GLMs

- ▶ Replace GLMs by GAMs or modern ML techniques
- ▶ Use same losses (deviance), weights, links

## More on Non-Life Insurance Pricing

- ▶ The severity model can use different features than the frequency model
- ▶ The Gamma model with link is slightly biased → can be fixed by applying empirical multiplicative correction factor

$$c = \sum_{i=1}^n y_i / \sum_{i=1}^n \hat{y}_i$$

calculated on the training data

- ▶ Why using a log link for the Gamma model?
- ▶ As an alternative to model claims frequency using case weights  $w$ , directly model claim counts  $N$  without weights but using an offset of  $\log(w)$   
→ Same effects, but different intercept, response

# Main Example

## Example (French motor third-part liability (MTPL) dataset)

1. Understand data
2. Descriptive analysis
3. Build claim frequency models

## The models

- ▶ Poisson GLM
- ▶ Poisson GAM
- ▶ Poisson gradient boosted trees

## Notes

- ▶ Grouped train/test split
- ▶ Model interpretation



# Table of Contents

XAI: Introduction

XAI: Explaining Models

XAI: Improving Explainability

# Introduction

Basic workflow to inspect and explain supervised learning model  $m$

1. Study model performance
2. Study feature importance
3. Study feature effects, ideally also interactions

## Focus

Global, model-agnostic, post-hoc explainability

## Analysis result

- ▶ Information gain
- ▶ Reveal problems in model/data
- ▶ Increase confidence in model and modeler

## Main references

- ▶ Online book of Christoph Molnar
- ▶ Tutorial (Mayer & Lorentzen 2020)

# Chapter Outline

1. Software
2. Performance
3. Excursion: grouped data
4. Variable importance
5. Effects
6. Global surrogate models
7. Improve linear models by XAI
8. SHAP

# Software for Post-Hoc Interpretation

## R

- ▶ DALEX
- ▶ iml
- ▶ flashlight
- ▶ SHAP: shapviz, fastshap, kernelshap
- ▶ ...

## Programming workflow

1. Build model
2. Create explainer object
3. Calculate and visualize results

## Python

- ▶ sklearn.inspect
- ▶ DALEX
- ▶ SHAP: shap
- ▶ ...

## Example

# Performance

- ▶ Study one or more relevant performance measures
- ▶ Often: Average loss or function of it
- ▶ Gives valuable information
- ▶ Training versus test performance? → assess overfitting/optimism
- ▶ Absolute and relative measures

## Also helps to detect problems

Is performance **much lower** than expected?

- ▶ Preprocessing errors
- ▶ Missing key feature
- ▶ Convergence problem

Is it **much better**?

- ▶ Data partitions not independent?
- ▶ Leakage from response to feature?

## Example: Claims Frequency Models

- ▶ Calculate weighted average Poisson deviance on test data:

$$\bar{S}(\hat{m}, D_{\text{test}}) = \sum_{i=1}^n w_i S(\hat{y}_i, y_i) / \sum_{i=1}^n w_i$$

with  $S(\hat{y}_i, y_i) = 2(y_i \log(y_i/\hat{y}_i) - (y_i - \hat{y}_i))$

- ▶ Relative deviance improvement (one of many “pseudo R-squared”)

$$1 - \frac{\bar{S}(\hat{m}, D_{\text{test}})}{\bar{S}(\hat{m}_{\text{trivial}}, D_{\text{test}})},$$

where  $\hat{m}_{\text{trivial}}(\mathbf{x}) = \sum_{D_{\text{train}}} w_i y_i / \sum_{D_{\text{train}}} w_i$  is the intercept-only model with constant predictions ideally calculated on the training data

- ▶ Repeat on training data (why?)

## Excursion: Grouped Data

- ▶ Flawed validation strategy → biased performance assessment
- ▶ Detailed knowledge of data and model required → difficult to detect

### Typical reason for flawed validation: Grouped data

- ▶ Pricing data
- ▶ Reserving: Models for ultimate claim amount
- ▶ Customer analytics: Browser behaviour of online visitors
- ▶ Banking: Financial transactions of clients

### Grouped splitting

- ▶ Instead of random sampling of *rows*, we sample *groups*
- ▶ All rows of a group go in same data partition
- ▶ If ignored: Overfitting is being rewarded

# Example and Simulation

## Example (French MTPL)

- ▶ Model performance of GLM and boosted trees model without grouped splitting?
- ▶ Impact on model tuning?

## Example (Simulation)

- ▶ Random data
- ▶ Linear regression and random forest
- ▶ 80%/20% split
- ▶ Increasing proportion of duplicated rows
- ▶ Random split versus grouped split



## More on Grouped Data

- ▶ We used grouping structure to create clean data splits
- ▶ Sometimes, one wants to also make use of within-group info in model
- ▶ Panel data, time-series

### Examples

- ▶ Insurance of large vehicle fleets: credibility factors
- ▶ Banking: Financial transactions of client

Tricky to have clean validation strategy and to apply model correctly

# Variable Importance

1. Information: Most/least important features?
2. Challenge correctness of model
  - ▶ Results as expected or not?
  - ▶ Seemingly unimportant feature is top predictor → leakage?
  - ▶ Key features not among important features → preprocessing problem, not sufficient understanding of data or modeling situation?

## Model-specific variable importance measures

- ▶ Linear model: normalized coefficients, test statistics etc.
- ▶ Tree-based models: Split gain or split count

## Model-agnostic measures

- ▶ Permutation importance (Breiman 2001 for random forests)
- ▶ SHAP feature importance

## Permutation Importance

Permutation importance of  $j$ -th feature  $X^{(j)}$ , data  $D$ , and performance measure  $\hat{S}$ :

$$\text{PVI}(j, D) = \hat{S}(\hat{m}, D^{(j)}) - \hat{S}(\hat{m}, D)$$

- ▶  $D^{(j)}$  is version of  $D$  with randomly permuted values in  $j$ -th feature column
- ▶ Read: How much  $\hat{S}$  worsens after shuffling column  $j$ ?  
The larger, the more important. If 0, feature is unimportant

Algorithm to calculate  $\text{PVI}(j, D)$  for all features

Algorithm 1: Permutation importance
scoreOriginal $\leftarrow$ performance on data <b>for</b> $x$ <i>in</i> variables <b>do</b> dataShuffled $\leftarrow$ data with permuted column $x$ scoreShuffled $\leftarrow$ performance on dataShuffled importance[ $x$ ] $\leftarrow$ scoreShuffled - scoreOriginal <b>end</b> <b>output</b> : importance

Source: Mayer and Lorentzen, 2020

# Remarks and Example

## Remarks

- ▶ Computationally cheap  $\rightarrow$  repeat  $m$  times
- ▶ Model is never refitted
- ▶ There is no formal definition of variable importance  $\rightarrow$  inconsistency across methods
- ▶ Different definitions of permutation importance
- ▶ Strongly dependent features  
 $\rightarrow$  decorrelate or analyze together
- ▶ Training or test data?

## Example (French MTPL)

- ▶ PVI using exposure-weighted average Poisson deviance
- ▶ Hold-out data
- ▶ Compare with tree-split gain

# Effects

Study and understand feature effects is of key importance

- ▶ How does  $m(\mathbf{x})$  change with  $j$ -th feature?
- ▶ Often Ceteris Paribus: other components in  $\mathbf{x}$  fixed

Advantage of intrinsically interpretable models

- ▶ (Ceteris Paribus) effect of feature  $X^{(j)}$  in a linear regression

$$\mathbb{E}(Y \mid \mathbf{x}) \approx m(\mathbf{x}) = \beta_o + \beta_1 x^{(1)} + \dots + \beta_p x^{(p)}$$

- ▶ In an additive model

$$\mathbb{E}(Y \mid \mathbf{x}) \approx m(\mathbf{x}) = \beta_o + f_1(x^{(1)}) + \dots + f_p(x^{(p)})$$

- ▶ In a black box model?

# Methods

1. Individual conditional expectation (ICE)
2. Partial dependence
3. Classic diagnostic plots
4. Interactions?
5. Later: SHAP dependence plots

# Individual Conditional Expectation (ICE)

## Basic thinking

- ▶ If  $m$  is additive in feature  $X^{(j)}$ , the Ceteris Paribus effect of  $X^{(j)}$  is the same for all observations  $\rightarrow$  complete description of effect / full transparency
- ▶ If complex interactions involved  $\rightarrow$  approximate description only

## Idea (Goldstein et al., 2015)

- ▶ Study Ceteris Paribus effect of  $X^{(j)}$  for one observation
- ▶ *ICE function* for feature  $X^{(j)}$  of model  $m$  and observation  $\mathbf{x} \in \mathbb{R}^p$

$$\text{ICE}_j : v \in \mathbb{R} \mapsto m(v, \mathbf{x}_{\setminus j})$$

- ▶  $\mathbf{x}_{\setminus j}$  denotes all but the  $j$ -th component of  $\mathbf{x}$ , which is replaced by  $v$
- ▶ *ICE curve* represents graph  $(v, \text{ICE}_j(v))$  for grid of values  $v \in \mathbb{R}$

# Simple Algorithm

Algorithm to calculate  $ICE_j(v)$

**Algorithm 2:** ICE for variable  $x$  and one observation

```
obs  $\leftarrow$  data row  
for  $v$  in grid of values do  
  | obs[ $x$ ]  $\leftarrow v$   
  | ice[ $v$ ]  $\leftarrow$  prediction for obs  
end  
output : ice
```

Source: Mayer and Lorentzen, 2020



# ICE Plot: Visualize ICE curves of multiple observations

## Example

### Notes

- ▶ Curves with different shapes indicate interaction effects
- ▶ Parallel curves  $\rightarrow$  additivity in  $X^{(j)}$
- ▶ Centered ICE plots
- ▶ Usually on link scale (why?)
- ▶ ICE plots of higher dimension
- ▶ Training versus test data

### Pros and Cons

- + Simple to compute
- + Easy to interpret (Ceteris Paribus)
- + Gives impression about interactions
- Suboptimal when Ceteris Paribus unnatural
- Model applied to rare/impossible  $\mathbf{x}$
- Does not show what variables are interacting

# Partial Dependence Plot PDP (Friedman 2001)

- ▶ Average of many ICE curves
- ▶ Ceteris Paribus effect of  $X^{(j)}$  averaged over all interaction effects

## Definition

- ▶ (Empirical) partial dependence function of feature  $j$

$$\text{PD}_j(v) = \frac{1}{n} \sum_{i=1}^n \hat{m}(v, \mathbf{x}_{i,\setminus j})$$

- ▶  $\mathbf{x}_{i,\setminus j}$  feature vector of  $i$ -th observation without  $j$ -th component
- ▶ Estimate for  $\mathbb{E}_{\mathbf{X}_{\setminus j}} m(v, \mathbf{X}_{\setminus j})$  (expectation runs over joint marginal distr. of  $\mathbf{X}_{\setminus j}$ )
- ▶ PDP equals graph  $(v, \text{PD}_j(v))$  for grid of values  $v \in \mathbb{R}$
- ▶ Sum runs over reference data (=?)

# Algorithm

Calculate  $PD_j(v)$  on grid of values for  $v$

**Algorithm 3:** Partial dependence profile for variable  $x$

```
 $n \leftarrow$  number of observations  
 $m \leftarrow$  grid size for variable  $x$   
iceProfiles  $\leftarrow$  matrix with  $n$  rows and  $m$  columns  
for  $i$  in 1 to  $n$  do  
  | iceProfiles[ $i$ ,:]  $\leftarrow$  ice curve for  $i$ th obs and variable  $x$   
end  
pd  $\leftarrow$  column means of iceProfiles  
output : pd
```

Source: Mayer and Lorentzen, 2020

Example

# More on Partial Dependence

## Remarks

- ▶ 2-dimensional PDP of  $X^{(j)}$  and  $X^{(k)}$ :

$$PD_{jk}(v_j, v_k) = \frac{1}{n} \sum_{i=1}^n \hat{m}(v_j, v_k, \mathbf{x}_{i, \setminus \{j, k\}})$$

- ▶ Accumulated local effects (ALE)

## Pros and Cons

- + Simple to compute
- + Easy to interpret (Ceteris Paribus)
- Suboptimal when Ceteris Paribus unnatural
- Model applied to rare/impossible  $\mathbf{x}$
- No information about interactions

# Classic Diagnostic Plots

## Related plots

1. Response versus covariate  
→ Descriptive marginal effects
2. Predicted versus covariate  
→ Modeled marginal effects
3. Residual versus covariate:  
→ Bias assessment

## Remarks

- ▶ Small and large datasets
- ▶ Binning of feature values
- ▶ Training versus test data?
- ▶ Relation to PDP?
- ▶ Pros and Cons?

## Example

## Interaction Effects

- ▶ Including interactions: Linear models versus black box models
- ▶ ICE plot for  $X^{(j)}$  gives impression of total interaction effects associated with  $X^{(j)}$

Pairwise interaction strength: Friedman's  $H$

$$H_{jk}^2 = \frac{\sum_{i=1}^n \left[ \text{PD}_{jk}(x_i^{(j)}, x_i^{(k)}) - \text{PD}_j(x_i^{(j)}) - \text{PD}_k(x_i^{(k)}) \right]^2}{\sum_{i=1}^n \left[ \text{PD}_{jk}(x_i^{(j)}, x_i^{(k)}) \right]^2}$$

- ▶ Sums run over reference data
- ▶ Partial dependence functions are mean centered
- ▶ Interpretation of  $H^2$ ? When close to 0 or 1?
- ▶  $H$  versus  $H^2$

# Absolute Interaction Strength

Friedman's  $H$  is a relative measure  $\rightarrow$  absolute measure?

$$\tilde{H}_{jk} = \sqrt{\frac{1}{n} \sum_{i=1}^n \left[ \text{PD}_{jk}(x_i^{(j)}, x_i^{(k)}) - \text{PD}_j(x_i^{(j)}) - \text{PD}_k(x_i^{(k)}) \right]^2}$$

## Remarks on $H$ and $\tilde{H}$

- ▶ Find out *how* features are interacting?  
 $\rightarrow$  Conditional PDP/ICE, 2D PDP, SHAP
- ▶ Computational burden?
- ▶ Usually, one works on link scale

## Example

# Global Surrogate Models

## Idea

- ▶ Fit intrinsically interpretable model  $m_I$  to predictions of  $\hat{m}$
- ▶ Usually a small decision tree
- ▶  $\hat{m}_I$  is called (global) surrogate model for  $\hat{m}$
- ▶ Objective function and R-squared of  $\hat{m}_I$ ?

## Remarks

- ▶ Training or test data?
- ▶ Variable importances of  $\hat{m}_I$ ?

## Example



# Improve Linear Models by XAI

## Workflow

- ▶ Build strong GLM by the help of ML and XAI
- ▶ Why not directly use ML model?

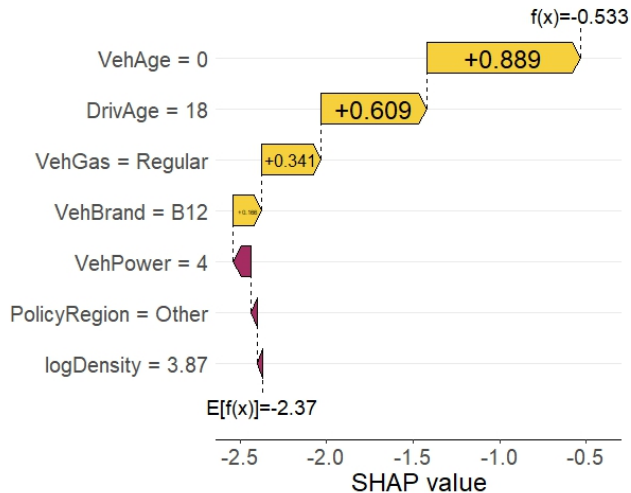
## Compare XAI aspects

1. Performance: Difference between GLM and ML model large or small?
2. Variable importance: Similar features important?
3. Main effects: Similar or not? → Change representation in GLM
4. Interaction effects: Add strong meaningful interactions to GLM

## Example

# SHAP: SHapley Additive exPlanations

- ▶ Local explanations
- ▶ Basic idea of SHAP?
- ▶ LIME: Local Interpretable Model-agnostic Explanations



# Shapley Values

- ▶  $\mathcal{M}$ : Set of  $p = |\mathcal{M}|$  players
- ▶ Playing cooperative game with numeric payoff
- ▶ Contribution of subset  $\mathcal{L} \subseteq \mathcal{M}$  of players measured by function  $v : \mathcal{L} \mapsto \mathbb{R}$

How to distribute payoff **fairly** among players?

Answer by Lloyd Shapley (1953): Player  $j$  should receive “Shapley value”

$$\phi_j(v) = \phi_j = \sum_{\mathcal{L} \subseteq \mathcal{M} \setminus \{j\}} \underbrace{\frac{|\mathcal{L}|!(p - |\mathcal{L}| - 1)!}{p!}}_{\text{Shapley weight}} \underbrace{(v(\mathcal{L} \cup \{j\}) - v(\mathcal{L}))}_{\text{Contribution of player } j}.$$

- ▶ Weighted average contribution over all coalitions  $\mathcal{L} \subseteq \mathcal{M} \setminus \{j\}$
- ▶ Average contribution over all  $p!$  permutations  $(\underbrace{\dots}_{\mathcal{L}}, j, \underbrace{\dots}_{\text{not playing}})$

# What means Fair?

Shapley values are only way to distribute total winnings fairly in this sense:

## Efficiency

$v(\mathcal{M}) = \phi_o + \sum_{j=1}^p \phi_j$ , where  $\phi_o = v(\emptyset)$  denotes non-distributed payoff

## Symmetry

If  $v(\mathcal{L} \cup \{i\}) = v(\mathcal{L} \cup \{j\})$  for every  $\mathcal{L} \subseteq \mathcal{M} \setminus \{i, j\}$ , then  $\phi_i = \phi_j$

## Dummy player

If  $v(\mathcal{L} \cup \{j\}) = v(\mathcal{L})$  for all coalitions  $\mathcal{L} \subseteq \mathcal{M} \setminus \{j\}$ , then  $\phi_j = 0$

## Linearity

Consider two cooperative games with gain functions  $v$  and  $w$ . Then,  
 $\phi_j(v + w) = \phi_j(v) + \phi_j(w)$  and  $\phi_j(\alpha v) = \alpha \phi_j(v)$  for all  $1 \leq j \leq p$  and  $\alpha \in \mathbb{R}$

# Shapley Values in Statistics and ML

## Early idea

Lipovetsky and Conklin (2001): Fair decomposition of *R-squared* in linear regression

Nowadays: Štrumbelj and Kononenko (2010, 2014), Lundberg and Lee (2017)

- ▶ Decompose *predictions* fairly into  $m(\mathbf{x}) = \phi_o + \sum_{j=1}^p \phi_j$ ,  $\phi_o = \mathbb{E}(m(\mathbf{X}))$
- ▶ Fair only if  $\phi_j$  are Shapley values
- ▶ Natural contribution function:  $v(\mathcal{L}) = m(\mathbf{x}_{\mathcal{L}})$ ,  $\mathbf{x}_{\mathcal{L}}$  are components in  $\mathcal{L} \subseteq \mathcal{M}$
- ▶ But: Features cannot be turned off in  $m \rightarrow$  use statistics to estimate it

## Situations where no estimation is required

- ▶  $p = 1$ : Then  $\phi_1 = m(\mathbf{x}) - \phi_o$
- ▶ Linear regression without correlations:

$$m(\mathbf{x}) = \underbrace{\beta_0}_{\phi_o} + \underbrace{\beta_1 x^{(1)}}_{\phi_1} + \dots + \underbrace{\beta_p x^{(p)}}_{\phi_p}$$

# How to Estimate Contribution Function

## Controversy in estimating $m(\mathbf{x}_{\mathcal{L}})$

- ▶ Statistically natural: Conditional expectation  $\mathbb{E}(m(\mathbf{X}) \mid \mathbf{x}_{\mathcal{L}})$
- ▶ Causal inference prefers marginal expectations:  $\mathbb{E}_{\mathbf{x}_{\mathcal{M} \setminus \mathcal{L}}}(m(\mathbf{X}))$

## Algorithms (from slow to fast)

1. **Monte Carlo sampling:** For each  $j$  and many  $\mathcal{L}$ , contributions  $v(\mathcal{L} \cup \{j\}) - v(\mathcal{L})$  are evaluated using marginal expectations and then plugged into Shapley's Eq.
2. **Kernel SHAP:** For many  $\mathcal{L}$ , evaluate  $v(\mathcal{L})$  using marginal expectations. Then use weighted regression to get all Shapley values without plugging into Shapley's Eq.
3. **TreeSHAP:** Uses properties of trees to directly calculate  $v(\mathcal{L})$  for all  $\mathcal{L} \subseteq \mathcal{M}$  and then plugging into Shapley's Eq.

## Example

# From Local to Global Explanations

## Decompose $n$ predictions, not just one

- ▶  $X$ :  $(n \times p)$  feature matrix with elements  $x_{ij}$ ,  $1 \leq i \leq n$ ,  $1 \leq j \leq p$
- ▶  $\Phi$ :  $(n \times p)$  matrix of SHAP values with elements  $\phi_{ij}$
- ▶  $\phi_o = \frac{1}{n} \sum_{i=1}^n \hat{m}(\mathbf{x}_i)$
- ▶  $\hat{m}(\mathbf{x}_i) = \phi_o + \sum_{j=1}^p \phi_{ij}$  for  $n$  feature vectors  $\mathbf{x}_i$

## Strategy to understand model as a whole

- ▶ SHAP feature importance:  $l_j = \frac{1}{n} \sum_{i=1}^n |\phi_{ij}|$
- ▶ SHAP dependence plots:  $\{(x_{ij}, \phi_{ij}), 1 \leq i \leq n\}$
- ▶ Interactions: Use  $x_{ik}$ ,  $k \neq j$  to add color to SHAP dependence plot (alternative?)

## Examples

Each aspect separately and full analysis

# SHAP Analysis to Improve Linear Model

## Revisit strategy

- ▶ A lot of info on ML black box can be generated very quickly
- ▶ Use to build strong GLM

## Example

## Remember

SHAP has a solid theoretical foundation. In practice, some of it is lost because statistics is not mathematics.



# Table of Contents

XAI: Introduction

XAI: Explaining Models

XAI: Improving Explainability

# Introduction

“The best explanation of a simple model is the model itself” (Lundberg and Lee, 2017)

## Two main strategies in XAI

- ▶ Model is black box → interpret it post-hoc
- ▶ Make model less opaque / improve intrinsic explainability

## Basic hierarchy of intrinsic explainability

1. Linear additive models (like GLMs)
2. Additive models (like GAMs)
3. Black box models (like boosted trees or neural nets)

## Note

- ▶ To link or not?
- ▶ Single decision tree

# Boundaries are Blurred

## Examples

- ▶ GLMs can have non-linear effects
- ▶ GLMs and GAMs can have interactions
- ▶ A complex GLM can be (almost) as black box as a boosted trees model
- ▶ Boosted trees models can have all or some features additive
- ▶ Neural nets can have all or some features additive

## Partly additive models = Additive with (possibly complex) interactions

- ▶ Additive time effects:  $m(\mathbf{x}) = f(\text{Time}) + f'(\text{other features})$
- ▶ Additive gender effects:  $m(\mathbf{x}) = f(\text{Gender}) + f'(\text{other features})$
- ▶ Additive model with non-additive location effects:  
$$m(\mathbf{x}) = f_1(x^{(1)}) + \dots + f_{p_1}(x^{(p_1)}) + f(\text{location features})$$

# Chapter Outline

## Structuring boosted trees

- ▶ Additive models
- ▶ Partly additive models
- ▶ Monotonicity

## Structuring neural nets

- ▶ Additive models
- ▶ Partly additive models

(Tune boosted trees models)

# Structuring Boosted Trees

## Interpreting boosted trees models

- ▶ Single decision trees  $m_k$  are simple to interpret
- ▶ Boosted trees  $m$  are sums of  $K$  decision trees

$$m(\mathbf{x}) = m_1(\mathbf{x}) + \cdots + m_K(\mathbf{x})$$

- ▶ Interpretation only post-hoc

Let's add structure to boosted trees!

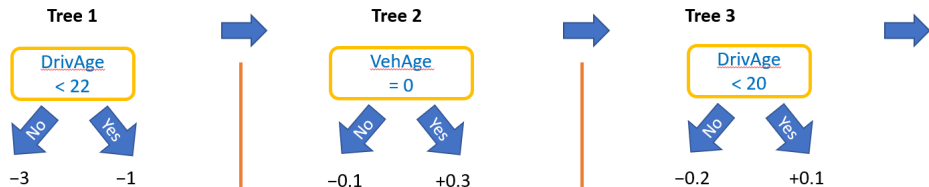
# Additive Boosted Trees

## What is a tree stump?

- ▶ Decision tree  $m_k$  with only one split (assume on  $j$ -th feature)
- ▶  $m_k(\mathbf{x}) = v_1 + (v_2 - v_1)\mathbb{I}(x^{(j)} \leq s)$ ; ( $v_1, v_2, s = ?$ )

## Boosted tree stumps are additive models

- ▶  $f_j$ : sum of those  $m_k$  that split on  $j$ -th feature
- ▶  $m(\mathbf{x}) = \beta_o + f_1(x^{(1)}) + \dots + f_p(x^{(p)})$



## More on Boosted Tree Stumps

- ▶ Additivity → full description of feature effects via ICE/PDP
- ▶ SHAP dependence plot? (Mayer 2022)
- ▶ Discussion: Pros and cons versus classic GAM?
- ▶ References: Lou et al. (2012), Nori et al. (2019)

### Example

# Partly Additive Boosted Trees

- ▶ Grow trees of depth 2  $\rightarrow$  pairwise interactions
- ▶ Partly additive model via *interaction constraints* (Lee et al. 2015)

## Interaction constraints

- ▶  $IC = \{F_1, \dots, F_K\}$
- ▶ Each  $F_k$  is feature subset allowed to interact

## How do they work?

- ▶ Consider a decision tree
- ▶ Each branch will use features only from one  $F_k \in IC$ .
- ▶ Rule: Each split considers features only from those  $F_k$  that contain all previous split variables of the branch. First split?
- ▶ Translates to tree and to sum of trees.



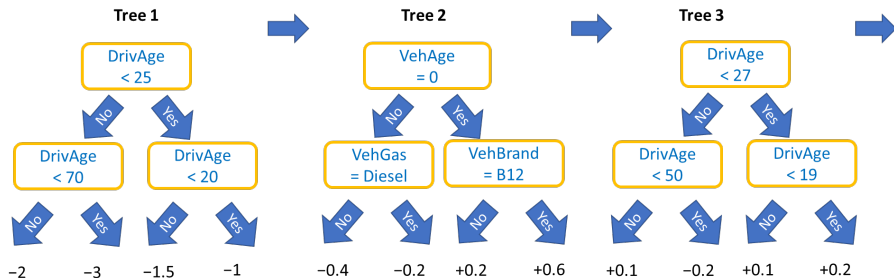
# Partly Additive Models via Interaction Constraints

How to set  $IC$  so that model is additive in  $j$ -th feature?

- ▶  $F_1 = \{X^{(j)}\}$
- ▶  $X^{(j)} \notin F_k$ , for  $k > 1$

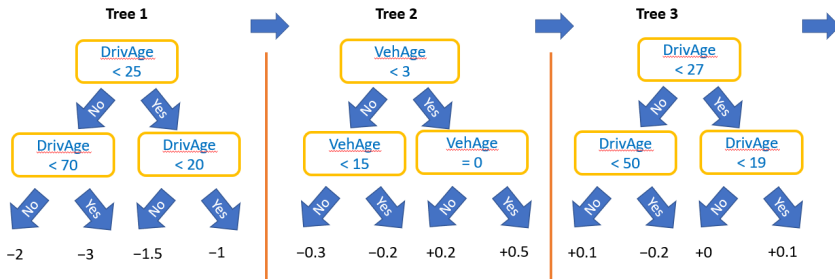
## Example

$IC = \{\{\text{DrivAge}\}, \{\log\text{Density}\}, \{\text{PolicyRegion}\}, \{\text{VehAge}, \text{Brand}, \text{Gas}, \text{Power}\}\}$



## More on Interaction Constraints

- ▶ If all elements in  $IC$  are disjoint, each tree uses features from only one  $F_k$
- ▶  $IC = \{\{X^{(1)}\}, \dots, \{X^{(p)}\}\}$  gives additive model:



- ▶ Difference to boosted tree stumps?

# Monotonic Constraints

- ▶ Monotonicity of  $m(\mathbf{x})$  in  $j$ -th feature is another aspect of interpretability
- ▶ Violated natural monotonicity can have dramatic impact on trustworthiness
- ▶ Example: car *collision* model
- ▶ Quite simple to implement for decision trees!
- ▶ Translates to tree ensembles

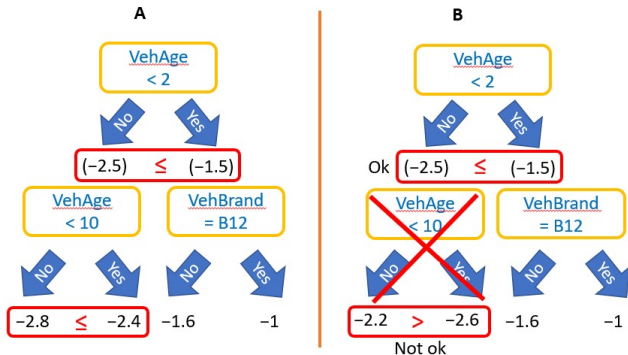


Figure: Monotone decreasing predictions in vehicle age

## More on Monotonic Constraints

- ▶ Be careful when imposing monotonicity (why?)
- ▶ Can help to reduce wiggleness of effect
- ▶ Monotonicity for other model classes like GLMs, GAMs, neural nets?
- ▶ Monotonicity and outlying feature values

### Example

# Structuring Neural Nets

## Swiss army knife of ML: Neural nets can

- ▶ mimic GLMs and GAMs,
- ▶ learn interactions and non-linear effects,
- ▶ fit data larger than RAM (e.g. images, videos),
- ▶ use multidimensional input and output,
- ▶ use input and output of mixed dimensionality,
- ▶ fit models with millions of parameters,
- ▶ perform non-linear dimension reduction,
- ▶ ...

## How to create

1. linear,
  2. complex,
  3. additive, and
  4. partly additive
- neural nets?

## Some Notes on Neural Nets

- ▶ Why haven't we worked with neural nets so far?
- ▶ Neural nets versus boosted trees?
- ▶ TensorFlow (with Keras), PyTorch
- ▶ Keras: sequential versus functional API
- ▶ Keras in R

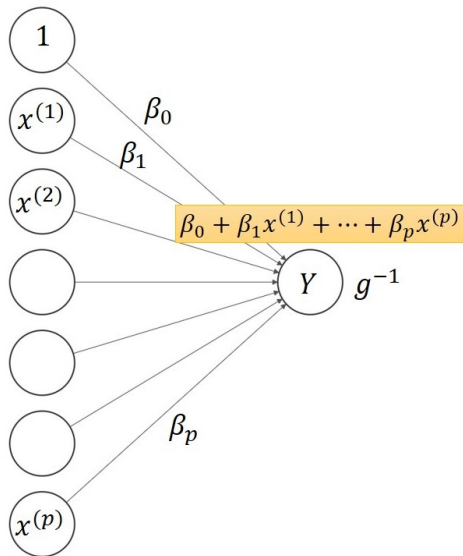
# A Simple Neural Net: GLM

## Some slang

- ▶ Input and output layer?
- ▶ Nodes and node values?
- ▶ Fully connected / dense layer
- ▶ Exponential activation function

## Example

- ▶ Scaling
- ▶ Callbacks
- ▶ Integer encoding



# Mini-Batch Gradient Descent with Backpropagation

Notation: Neural net  $m_\beta$ ; its total loss on data  $D$  and loss function  $L$ :

$$Q(m_\beta, D) = \sum_{(y_i, \mathbf{x}_i) \in D} L(m_\beta(\mathbf{x}_i), y_i)$$

1. Init: Randomly initialize parameter vector  $\beta$  by  $\hat{\beta}$
2. Forward: Calculate  $Q(m_{\hat{\beta}}, D_{\text{batch}})$  on **batch**
3. Backprop: Modify  $\hat{\beta}$  to improve  $Q(m_{\hat{\beta}}, D_{\text{batch}})$ 
  - 3.1 Calculate partial derivatives  $\nabla \hat{\beta} = \frac{\partial Q(m_\beta, D_{\text{batch}})}{\partial \beta} \big|_{\beta=\hat{\beta}}$  using backprop (=?)
  - 3.2 Gradient descent: Move slightly into right direction:  $\hat{\beta} \leftarrow \hat{\beta} - \lambda \cdot \nabla \hat{\beta}$
4. Repeat Steps 2 and 3 until one **epoch** is over
5. Repeat Step 4 until some stopping criterion triggers



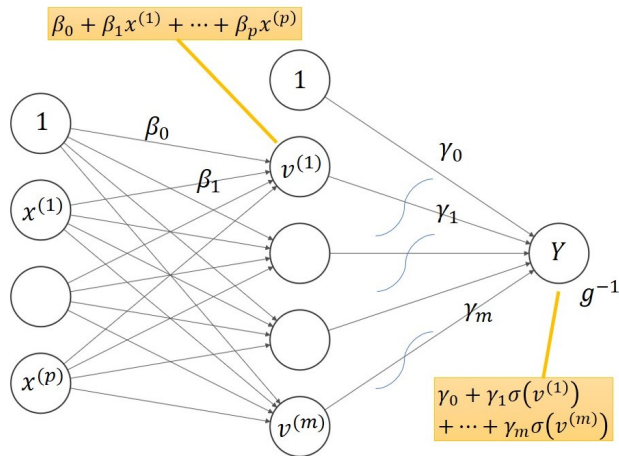
# More Complex Models

## Some additional slang

- ▶ Hidden layers
- ▶ Representational learning
- ▶ Activation functions: two purposes
- ▶ How to choose architecture?
- ▶ How to choose number of parameters/weights?

## Example

- ▶ Three hidden layers
- ▶ 561 parameters

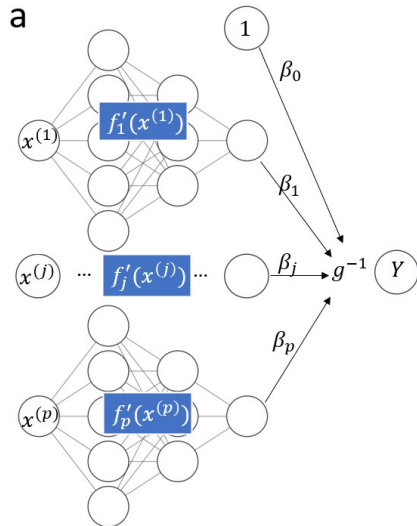


# Additive Neural Nets (Agarwal et al. 2020)

- ▶ Represent each feature by single-output net
- ▶ Directly connected to output layer
- ▶ Linear components?
- ▶ Unordered categorical features?

## Example

- ▶ 'VehBrand', 'PolicyRegion': 1-D embedding
- ▶ 'VehGas' and 'logDensity': Scaled and represented by linear function
- ▶ Rest: Scaled and represented by small net each
- ▶ Similar structure as our original GAM



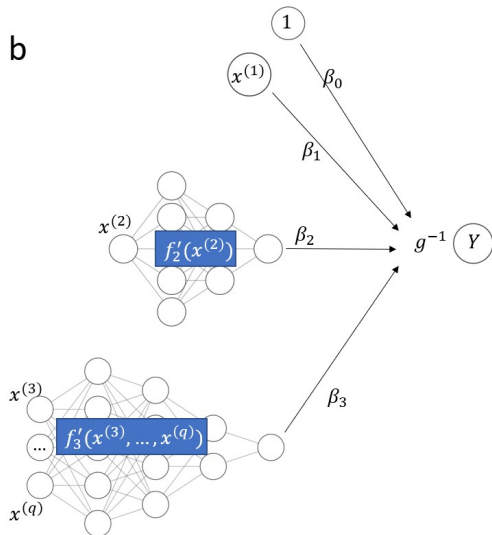
Source: Fig 1a in  
<https://www.mdpi.com/1911-8074/15/5/193>

# Partly Additive Models

- ▶ Pairwise interactions?
- ▶ How constructed?

## Example: Additive driver effects?

- ▶ 'logDensity': Scaled and represented as linear function
- ▶ 'DrivAge': Scaled and represented by a small sub-network
- ▶ 'PolicyRegion': 1-D embedding
- ▶ Vehicle features: sub-network with different inputs and five outputs



Source: Fig 1b in <https://www.mdpi.com/1911-8074/15/5/193>

# Excursion: Tuning Boosted Trees

## Model tuning in general

- ▶ How to choose hyperparameters of ML models?
- ▶ Each model class (GLMs, GAMs, random forests, boosted trees, neural nets, ...) has specialities that should be respected
- ▶ Examples?

## Why focussing on boosted trees?

- ▶ Usually among best performing models for tabular data
- ▶ Boosting + SHAP → strong GLMs
- ▶ It takes some practice

## Aspects

1. Objective and evaluation metric
2. Number of boosting rounds
3. Learning rate
4. Further parameters

# Objective and Metric

## Ideal choice of loss function

- ▶ Meaningful for task
- ▶ Strictly consistent for target functional  $T$

## Translation to objective and metric

- ▶ Objective: average loss on training data (plus regularization) used for model training
- ▶ Evaluation metric: average (cross-)validation loss used for model comparison and selection

## Example

# Number of Boosting Rounds

Very important to select reasonable number of boosting rounds

- ▶ Boosting round = tree
- ▶ Too few rounds  $\rightarrow$  underfitting
- ▶ Too many rounds  $\rightarrow$  overfitting
- ▶ Heavily depends on choice of other parameters, thus difficult to choose

“Early stopping” as standard solution

- ▶ How does it work?
- ▶ Why is it so convenient?

# Learning Rate

- ▶ Weight of each tree in final model
- ▶ Often between wide range of 1 and 0.005
- ▶ Good value heavily depends on number of boosting rounds
- ▶ Trick: select it so that early stopping ends after 100 – 1000 trees (why?)
- ▶ Halving the number of trees means doubling the learning rate for comparable performance

# Regularization Parameters

## Additional parameters to select

- ▶ number of leaf nodes
- ▶ tree depth
- ▶ loss penalties
- ▶ different types of subsampling rates
- ▶ ...

## Choose them by (cross-)validation

- ▶ One by one
- ▶ Grid-search
- ▶ Random search

## Note

- ▶ Early-stopping often compensates for suboptimal choice of other parameters
- ▶ Very different parameter combinations may lead to similar performance



# Overall Strategy

## Three steps

1. Choose strictly consistent and meaningful loss for functional  $T$   
→ objective and evaluation metric
2. Choose learning rate to get 100 – 1000 trees with early stopping
3. Select remaining parameters manually or by random search via (cross-)validation

## Simplification

When to skip expensive Step 3?

## Example

- ▶ French MTPL
- ▶ Speciality: grouped partitions