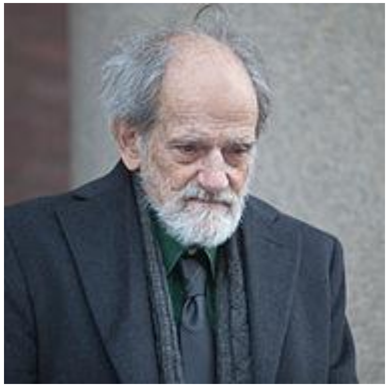
Shapley Value Explanations, Comparison with LIME & Our work

Martin Jullum

Shapley values



Lloyd S. Shapley
Nobel Price Winner in
Economics, 2012.

- Originating from cooperative game theory Shapley (1953)
- Used to distribute payments to players based on their contribution
- Shapley value for a player = the “fair” payment that player should get
- Has an explicit mathematical formula
- Several nice optimality properties in terms of fairness

Shapley values for prediction explanation

- Idea for use in prediction explanation
  - Players = variables/features $(x_1, \dots, x_p)$
  - Payment = prediction $f(x^*)$
- Shapley value for feature j = ϕ_j:

$$\phi_j = \sum_{S \subseteq M \setminus \{j\}} w(S) \left(v(S \cup \{j\}) - v(S) \right), \qquad w(S) = \frac{|S|!\,(|M| - |S| - 1)!}{|M|!}$$

  - Contribution function $v(S) \approx$ prediction “knowing only the features in S”
  - $M = \{1, \dots, p\}$
  - S is a subset of M

Shapley formula with 3 features

- The Shapley formula from the previous slide

$$\phi_j = \sum_{S \subseteq M \setminus \{j\}} w(S) \left(v(S \cup \{j\}) - v(S) \right)$$

$$\begin{aligned}
\phi_1 &= \tfrac{1}{3}\left(v(\{1{,}2{,}3\}) - v(\{2{,}3\})\right) + \tfrac{1}{6}\left(v(\{1{,}2\}) - v(\{2\})\right) + \tfrac{1}{6}\left(v(\{1{,}3\}) - v(\{3\})\right) + \tfrac{1}{3}\left(v(\{1\}) - v(\emptyset)\right), \\
\phi_2 &= \tfrac{1}{3}\left(v(\{1{,}2{,}3\}) - v(\{1{,}3\})\right) + \tfrac{1}{6}\left(v(\{1{,}2\}) - v(\{1\})\right) + \tfrac{1}{6}\left(v(\{2{,}3\}) - v(\{3\})\right) + \tfrac{1}{3}\left(v(\{2\}) - v(\emptyset)\right), \\
\phi_3 &= \tfrac{1}{3}\left(v(\{1{,}2{,}3\}) - v(\{1{,}2\})\right) + \tfrac{1}{6}\left(v(\{1{,}3\}) - v(\{1\})\right) + \tfrac{1}{6}\left(v(\{2{,}3\}) - v(\{2\})\right) + \tfrac{1}{3}\left(v(\{3\}) - v(\emptyset)\right).
\end{aligned}$$

SHAP

- Lundberg & Lee (2017): Shapley value explanation using $v(S) = E[f(x)|x_S = x_S^*]$
- $E[f(x)|x_S = x_S^*]$ is unknown, so has to be approximated

$$E[f(x)|x_S = x_S^*] = E[f(x_{\bar{S}}, x_S)|x_S = x_S^*] = \int f(x_{\bar{S}}, x_S^*) p(x_{\bar{S}}|x_S = x_S^*) \mathrm{d}x_{\bar{S}}$$

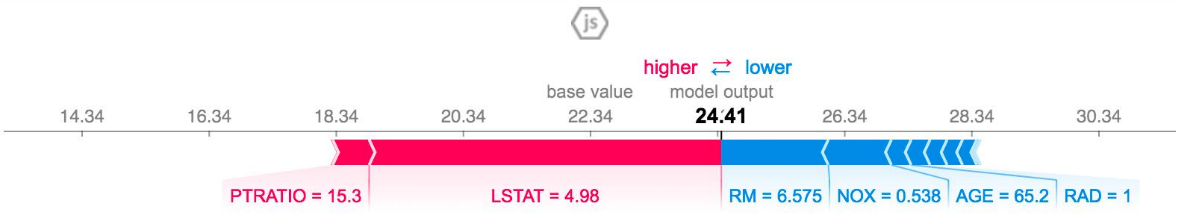
- SHAP assumes feature independence in this stage
  - Replaces $p(x_{\bar{S}}|x_S = x_S^*)$ by $p(x_{\bar{S}})$
- Approximates the integral by Monte Carlo sampling
  - $v_{SHAP}(S) = \frac{1}{K}\sum_{k=1}^{K} f(x_{\bar{S}}^{(k)}, x_S^*)$, where $x_{\bar{S}}^{(k)}$ is a sample from the training data, sampled **independently** of x_S^*
- Strumbelj & Kononenko (2014) doing a simular thing

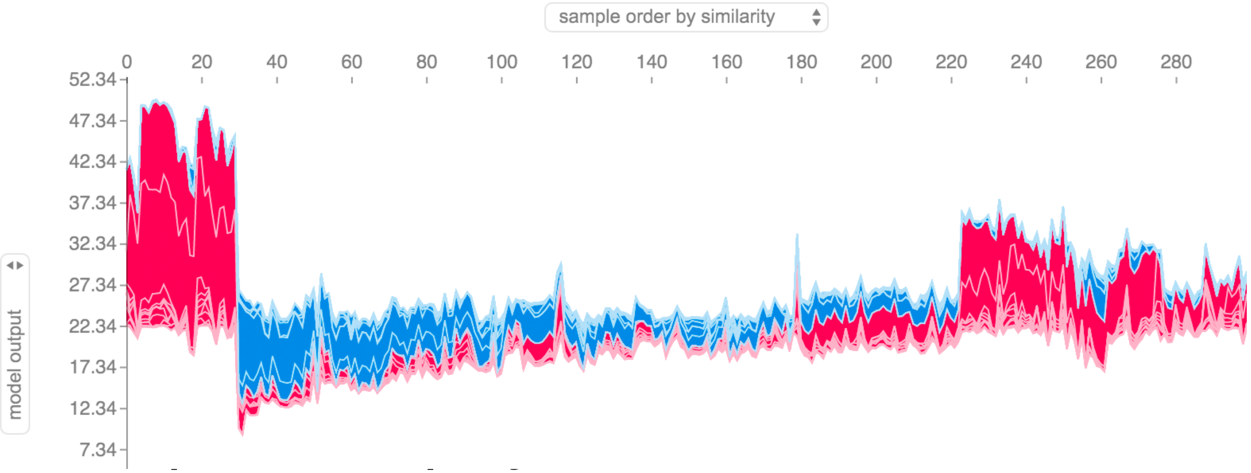
Kernel SHAP

- Computing ϕ_j requires approximation of 2^{p+1} different $v(S)$
  - Computationally too heavy with many features (large p)
- The majority of the Shapley weights $w(S)$ are usually very small compared to the largest ones.
- Kernel SHAP (Lundberg & Lee, 2017)
  - Limit the computational problem by sampling a finite set S sets with probabilities proportional to $w(S)$, and only perform computation for those S
  - Computes all ϕ_j simultaneously by rephrasing it as the solution to a weighted least squares problem

Available software

- Python library for (kernel) SHAP by Scott Lundberg: https://github.com/slundberg/shap

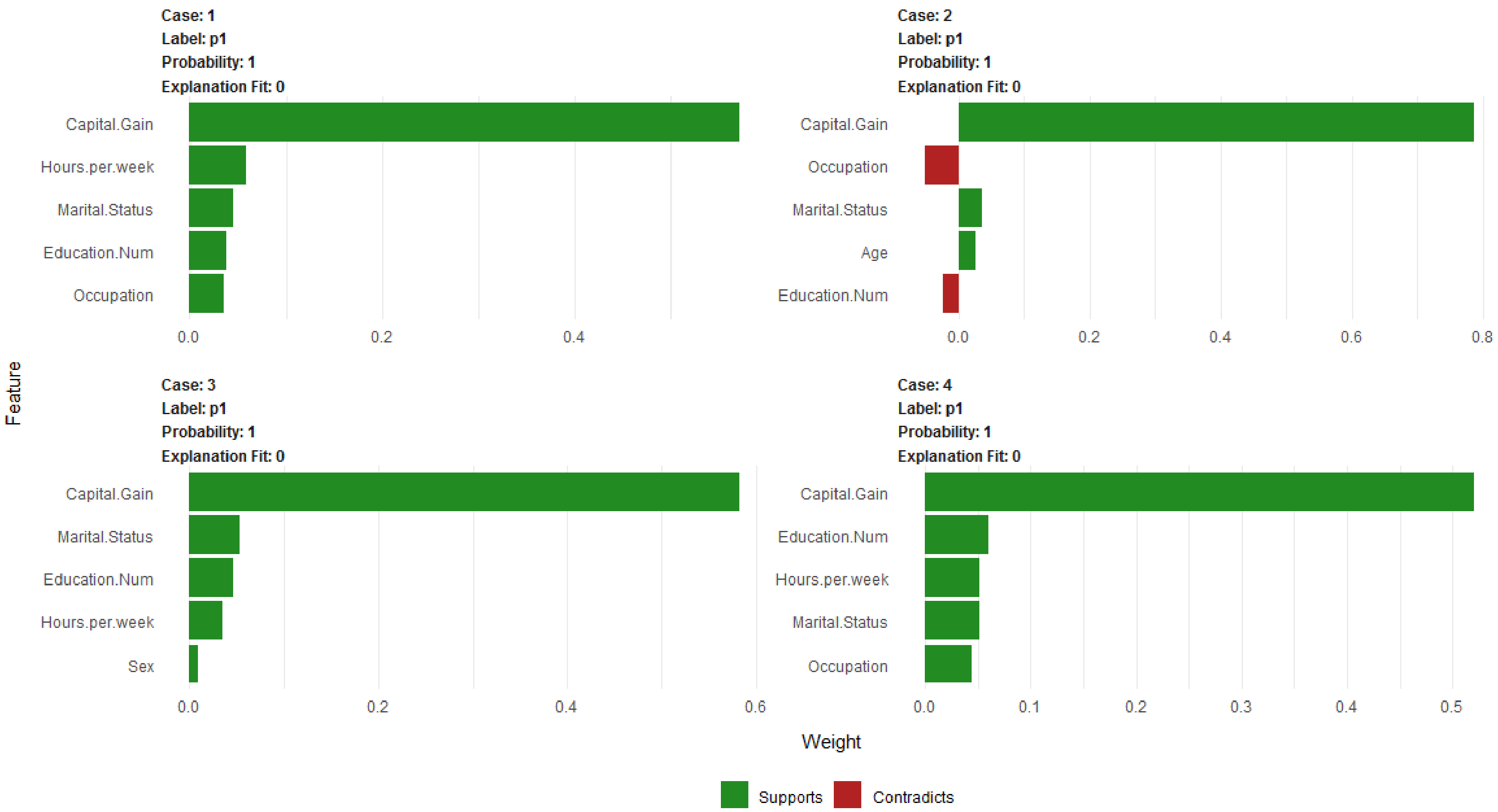




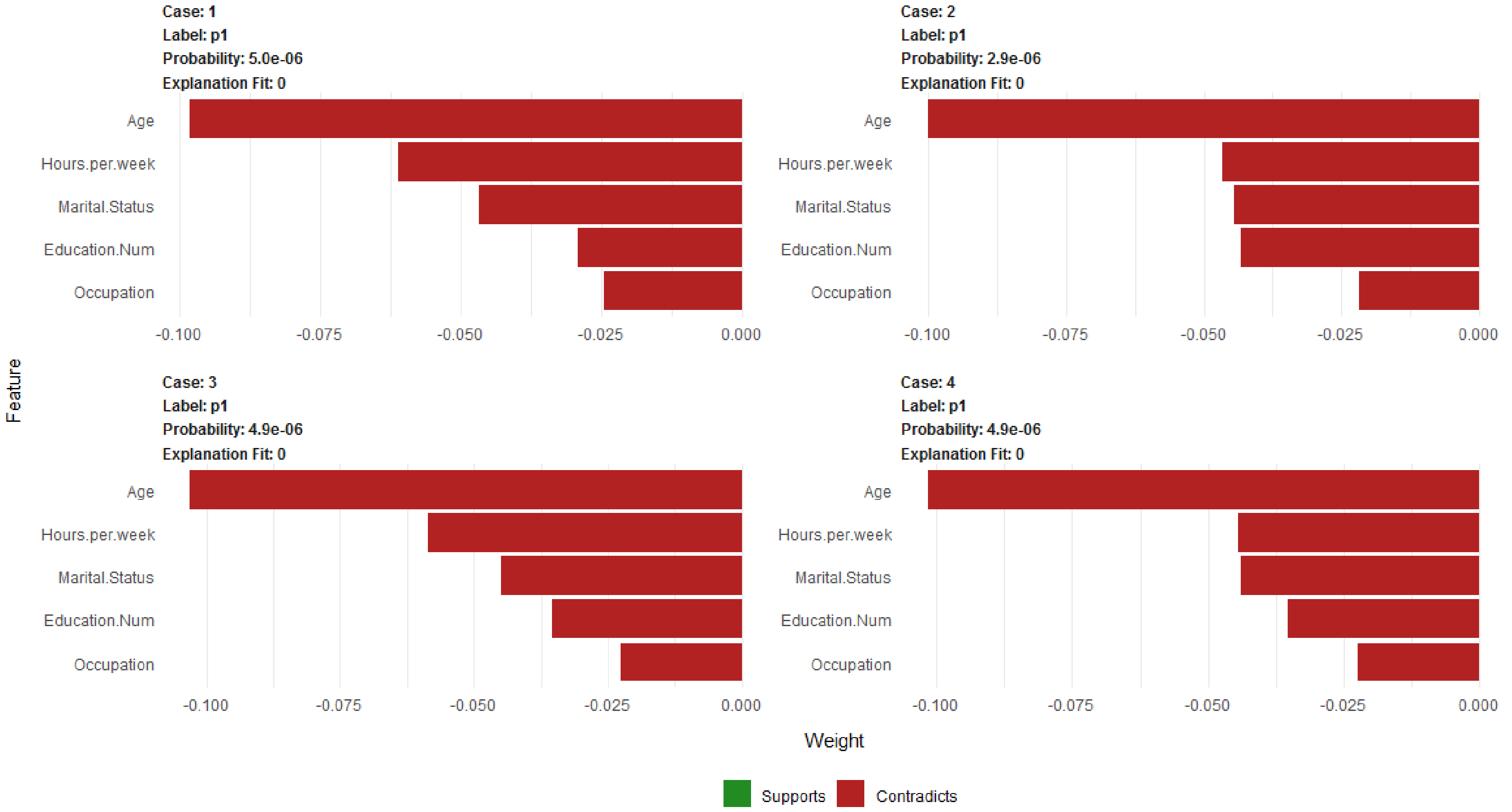
- M ectly in X

- R- ml) and *Ex* pr ex.html) ca

Kjersti’s example using SHAP – high probability



Kjersti's example using SHAP – low probability



Comparing LIME and Shapley/SHAP I

Explains two different things

LIME

- Individual explanation with **local** reference level
- $\phi_j \approx$ How does the prediction change if you change x_j from any other category/bin of x_j to that of x_j^*
- "*How can I increase/reduce my prediction?*"
- $\phi_1 + \phi_2 + \cdots + \phi_p \approx f(x^*) - \phi_0$

Individuals similar to you

Shapley/SHAP

- Individual explanation with **global** reference level
- $\phi_j \approx$ How does the prediction change from not knowing x_j^*
- "*How is the prediction influenced by the observing different features?*"
- $\phi_1 + \phi_2 + \cdots + \phi_p = f(x^*) - \phi_0$

All individuals

Comparing LIME and Shapley/SHAP I

Explains two different things

LIME

- Individual explanation with **local** reference level
- $\phi_j \approx$ How does the prediction change if you change x_j from any other category/level of x_j to that of x_j^*
- “*How can I increase/reduce my prediction?*”
- $\phi_1 + \phi_2 + \cdots + \phi_p \approx f(x^*) - \phi_0$

?

?

Individuals similar to you

Shapley/SHAP

- Individual explanation with **global** reference level
- $\phi_j \approx$ How does the prediction change from not knowing x_j^*
- “*How is the prediction influenced by the observing different features?*”
- $\phi_1 + \phi_2 + \cdots + \phi_p = f(x^*) - \phi_0$

All individuals

Comparing LIME and Shapley/SHAP II

LIME

- Conceptually easy
- Easy-to-use software
- No theoretical foundation or optimality results
- Assumes feature independence when sampling for local fitting
- “Chooses” some features that get non-zero ϕs
- Not necessarily continuous ϕ_j

Shapley/SHAP

- Harder to understand how it works
- Some software exists
- Complete theoretical framework with nice properties
- Assumes feature independence when approximating $v(S)$
- All contributing x_j get a non-zero ϕ_j
- Continuous ϕ_j

Comparing LIME and Shapley/SHAP II

LIME

- Conceptually easy
- Easy-to-use software
- No theoretical foundation or optimality results
- Assumes feature independence when sampling for local fitting
- “Chooses” some features that get non-zero ϕs
- Not necessarily continuous ϕ_j

Shapley/SHAP

- Harder to understand how it works
- Some software exists
- Complete theoretical framework with nice properties
- Assumes feature independence when approximating $v(S)$
- All contributing x_j get a non-zero ϕ_j
- Continuous ϕ_j

Problematic in case of (strong) feature dependence

Our research within Big Insight

- We prefer the Shapley framework
- The (only?) problem with SHAP is the assumption of features independence when approximating
  $$v(S) = E[f(x)|x_S = x_S^*] = \int f(x_{\bar{S}}, x_S^*) p(x_{\bar{S}}|x_S = x_S^*) \mathrm{d}x_{\bar{S}}$$
- Our novel idea: “Repair” (Kernel) SHAP by approximating $v(S)$ properly
  - Estimate the conditional distribution $p(x_{\bar{S}}|x_S = x_S^*)$ instead of inserting the empirical distribution of $p(x_{\bar{S}})$
  - Approximate the integral by Monte Carlo sampling similar to before
    - $v_{COND.SHAP}(S) = \frac{1}{K}\sum_{k=1}^{K} f(x_{\bar{S}}^{(k)}, x_S^*)$, where $x_{\bar{S}}^{(k)}$ is a sample from an approximation to $p(x_{\bar{S}}|x_S = x_S^*)$

Approximating the conditional distribution

- (At least) three alternatives:
  1. Assume a parametric multivariate distribution with known conditionals, e.g.
     - Gaussian distribution
     - Generalised Hyperbolic Distribution
  2. Use a copula with a dependence distribution with know conditionals
  3. Use a nonparametric **conditional** empirical distribution

- Obviously computationally more heavy than using the empirical distribution of $p(x_{\bar{S}})$ directly

Concluding remarks

- Still needs to set some parameters
  - Number of Monte Carlo samples (K): We typically use 10^3 to 10^4
  - Bandwidth parameter for the conditional empirical approach: We have used AICc (Hurvich et al., 2007) for selection
- Experiments with different methods:
  - Performance depends on data distribution and prediction model
  - Empirical approach preferable for $|S| \leq 3$, otherwise copula method is preferable
  - Hard to estimate conditional distributions, but our methods are always* better than assuming independence
  - TreeSHAP in XGBoost/LightGBM/CatBoost is very inaccurate
- We are currently writing a paper
- Will also publish an R-package

Copula method

Procedure to sample from $p(x_{\bar{S}}|x_S = x_S^*)$ assuming a Gaussian copula

1. For every feature: Transform the training observations to standard normal $z_j = \Phi^{-1}(\hat{F}_j\left(x_j\right))$
2. Fit a Gaussian distribution p_G to the transformed training data $(z_1, \dots, z_p)$
3. Sample $(z_{\bar{S}}^{(1)}, \dots, z_{\bar{S}}^{(K)})$ from $p_G(z_{\bar{S}}|z_S = z_S^*)$
4. For every feature in S: Convert the samples back to the original marginal: $x_{\bar{S},j}^{(k)} = \hat{F}_j^{-1}(\Phi\left(z_{\bar{S},j}^{(k)}\right))$

Conditional empirical distribution approach

- Compute the Mahalanobis distance $D_S(x, x^*)$ between x^* and all observations x in the training set, **using only the elements in S**
- Compute the weight for each observation $w_S(x) = \exp(D_S(x, x^*)^2/(2\sigma))$
- Define the conditional empirical distribution of $x_{\bar{S}}$ given $x_S = x_S^*$ as that having point mass of size $w_S(x)$ at $x_{\bar{S}}$
- Order the weights from large to small $w_S^{(1)}, \dots, w_S^{(n)}$, and use K largest weights instead of Monte Carlo sampling

$$v(S) = \frac{\sum_{k=1}^{K} w_S^{(k)}(x) f(x_{\bar{S}}, x_S^*)}{\sum_{k=1}^{K} w_S^{(k)}(x)}$$