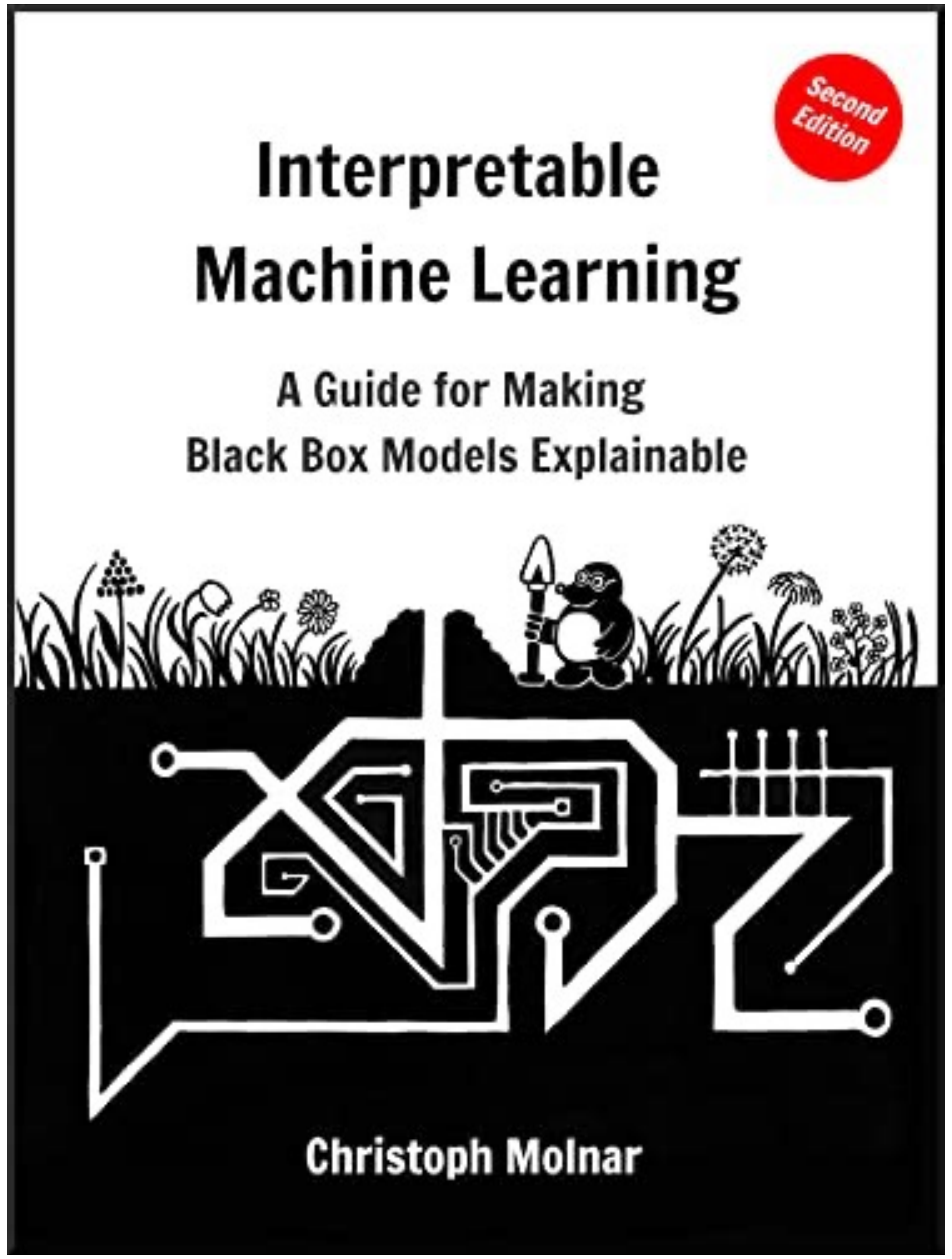
Statistical and Sequential Learning for Time Series Forecasting

Interpretable Machine Learning

Margaux Brégère

References



Second Edition

Interpretable Machine Learning

A Guide for Making Black Box Models Explainable

Christoph Molnar

- Apley and Zhu. Visualizing the effects of predictor variables in black box supervised learning models, 2020

- Fisher et al. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously, 2019

- Friedman and Popescu. Predictive learning via rule ensembles, 2008

- Kim, Khanna, and Koyejo. Examples are not enough, learn to criticize! criticism for interpretability, 2016

- Koh and Liang. Understanding black-box predictions via influence functions, 2017

- Ribeiro et al."Why should i trust you?" Explaining the predictions of any classifier, 2016

- Shapley et al. A value for n-person games, 1953

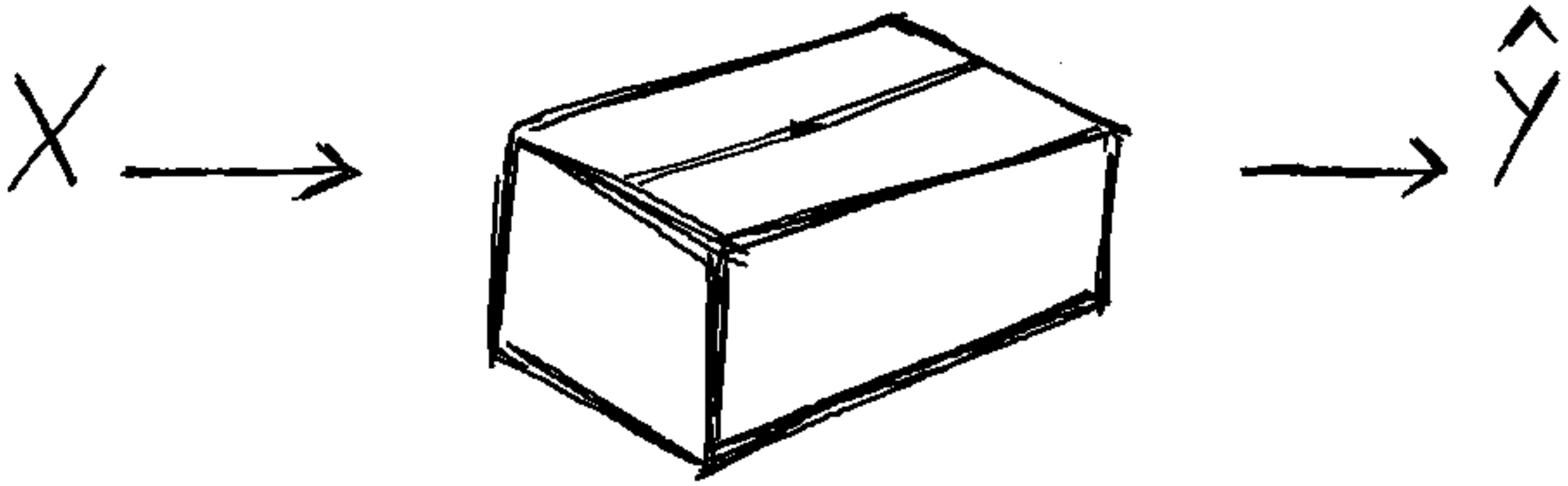
- Štrumbelj and Kononenko. Explaining prediction models and individual predictions with feature contributions, 2014

- Szegedy et al. Intriguing properties of neural networks, 2013

- Wachter et al. Counterfactual explanations without opening, 2017

Introduction

Post hoc (model analysis after training) interpretability methods can be



model-specific (limited to specific model classes) or model-agnostic

local (explain an individual prediction) or global (explain the entire model behaviour)

and may output

- Feature summary statistic or visualisation
- Model internals (learned weights)
- Data point (counterfactual explanations)
- Intrinsically interpretable model

Interpretable models

- Linear models
- Generalised additive models
- Decision trees

Model-agnostic methods

- Partial dependance plot
- Accumulated local effects
- Feature importance
- Surrogate model
- Shapley values

Example-based methods

- Counterfactual explanations
- Adversarial exemples
- Prototypes and criticisms

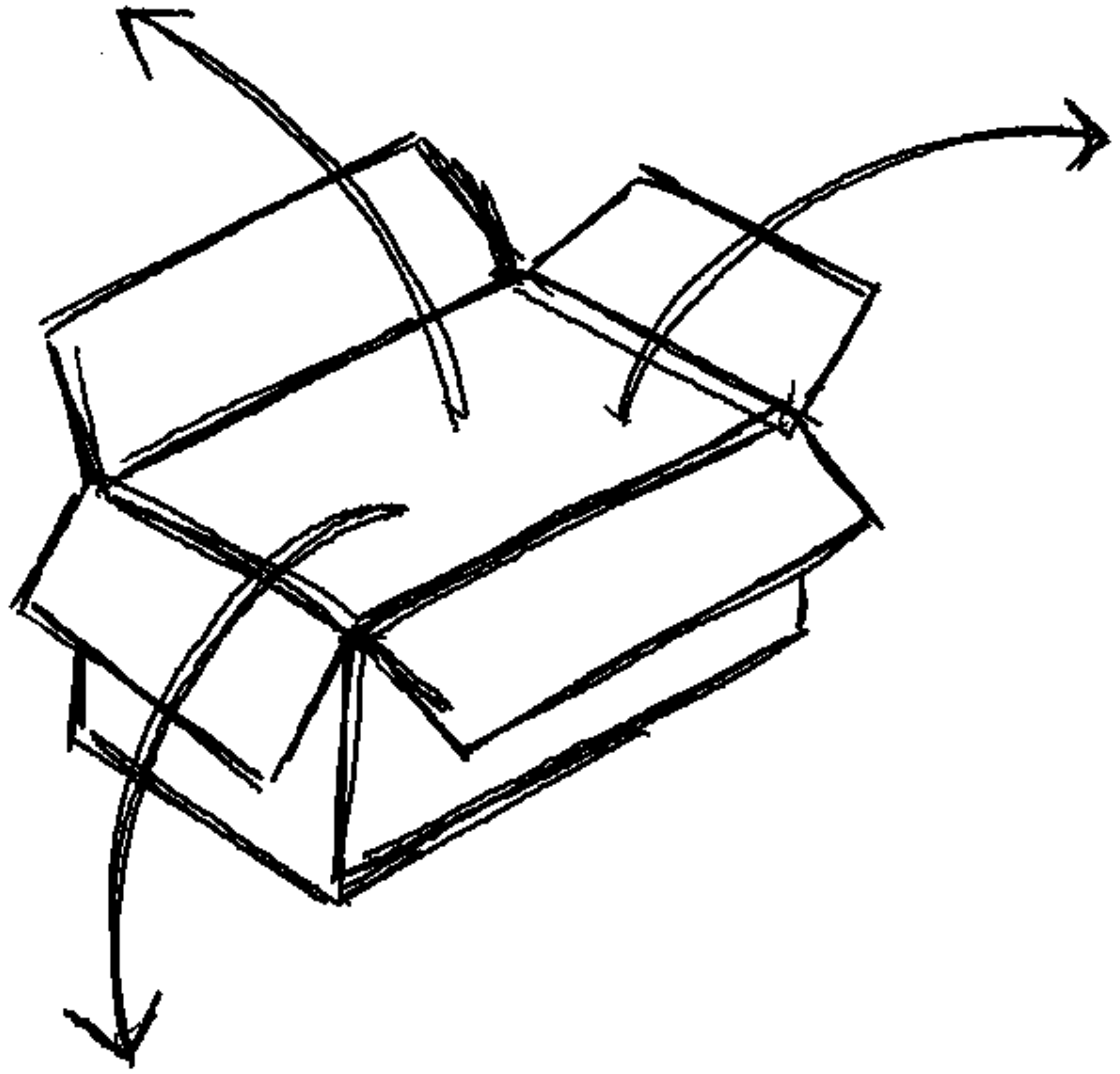
Interpretable models

Linear Models $\mathbb{E}[Y] = X^{\mathrm{T}}\beta$

$$\hat{Y} = X^{\mathrm{T}}\hat{\beta} \;\rightarrow\; \hat{\beta}_1, \ldots, \hat{\beta}_p$$

Generalised Additive Models $\mathbb{E}\left[g(Y)\right] = \sum_k f_k(X)$

$$\hat{Y} = g^{-1}\left(\sum_k \hat{f}_k(X)\right) \;\rightarrow\; \hat{f}_k\,(X)$$



Classification And Regression Trees

$$\left(\text{variable}_k, \text{threshold}_k\right)_{k \in \text{splits}}$$

Use case: daily french power consumption

Model	RMSE		MAPE	
	Train	Test	Train	Test
Benchmark (empirical mean)	10 852 MW	10 669 MW	17.4%	18.9%
Linear regression	5 091 MW	5 718 MW	8.10%	9.97%
Generalized additive model	3 498 MW	4 097 MW	4.85%	6.32%
CART	3 784 MW	4 505 MW	5.16%	6.70%

Variable to forecast:

- Daily french power consumption (MW)
- Train:

  Jan. 1st 2016 - Jul. 23rd 2021
- Test:

  Jul. 24th 2021 - May 31st 2023

Explanatory variables:

- Temperature
- Day (Mon., Tue., … Sun.)

Linear regression

Assumptions: $\forall i = 1,\ldots n, Y_i = X_i\beta + \varepsilon_i\text{, with}\,\varepsilon_i \sim \mathcal{N}(0,\sigma^2)\ \mathrm{i.i.d}$

Model: $\hat{\beta} = \left(X^{\mathrm{T}}X\right)^{-1}X^{\mathrm{T}}Y$ (ordinary least squares)

Prediction: $\hat{Y}_{\mathrm{new}} = \hat{\beta}X_{\mathrm{new}}$

Weights interpretation: features do not have to be strongly correlated!

- Numerical feature: increasing the feature by one unit changes the prediction by its weight
- Binary features: changing the feature from the reference category to the other category changes the prediction by the feature’s weight

  → One-hot-encoding for many categories
- Intercept: if all feature values are zero and all categorical feature values at the references, prediction = intercept weight. This is meaningful when the features have been standardised (= prediction of an input where all features are at their mean value)

Linear regression

R-squared reflects how much of the total variance of the target outcome is explained by the model

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{n} \left(Y_i - \bar{Y}\right)^2} \quad \text{with} \quad \bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

$R^2 = 0$ with the model does not explain the data at all and $R^2 = 1$ if it explains all of the variance

R-squared $\nearrow$ if $p \nearrow$ even if the new variables do not contain any information

→ use the adjusted R-squared:

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

It is not meaningful to interpret a model with very low (adjusted) R-squared.

Linear regression

Feature importance: absolute value of its t-statistic (testing $\beta_j = 0$ against $\beta_j \neq 0$), which is the estimated weight scaled with its standard error:

$$\frac{\hat{\beta}_j}{\sqrt{\mathrm{Var}(\hat{\beta}_j)}} \approx \frac{\hat{\beta}_j}{\sigma\sqrt{\left(X^{\mathrm{T}}X\right)^{-1}_{jj}}}$$

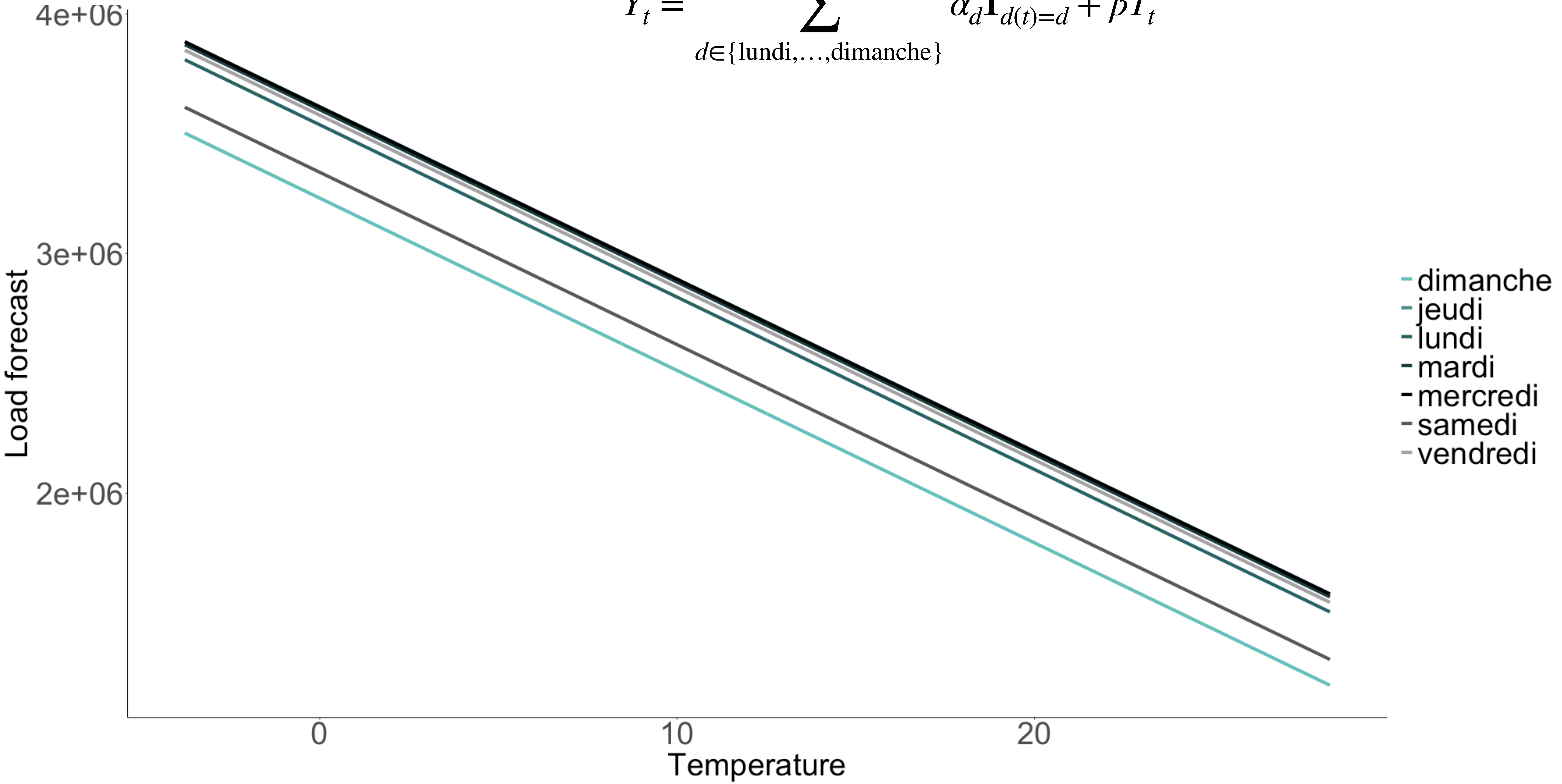
- Importance of feature j $\nearrow$ if $\hat{\beta}_j \nearrow$
- Importance of feature j $\searrow$ if $\mathrm{Var}(\hat{\beta}_j) \nearrow$ (we are less certain about the weight value)

Feature selection:

→ LASSO (least absolute shrinkage and selection operator) introduces

Example

$$\hat{Y}_t = \sum_{d \in \{\text{lundi}, \ldots, \text{dimanche}\}} \alpha_d \mathbf{1}_{d(t)=d} + \beta T_t$$



Pros and cons

Linear regression

- models the predictions as a weighted sum of features
- ensures the feature number remains small with LASSO
- is used in many domains
- comes with solid statistical theory

but it

- can only represent linear relationships (each non-linearity or interaction has to be hand-crafted and explicitly given to the model as an input feature)
- has generally poor predictive performance (oversimplify model)
- can be counter-intuitive when features are correlated (weighted features can then offset each other; with completely correlated features there is no longer any uniqueness of solution)

Generalised additive model

Assumption: $\mathbb{E}\left[g(Y)\right] = \sum_k f_k(X)$

Model: $\forall k, f_k$ is approximated using splines

Prediction: $\hat{Y}_{\text{new}} = g^{-1}\left(\sum_k \hat{f}_k(X_{\text{new}}) \right)$

→ functions $\hat{f}_k$ can be plotted

→ predictions are a sum of weighted transformed features

It

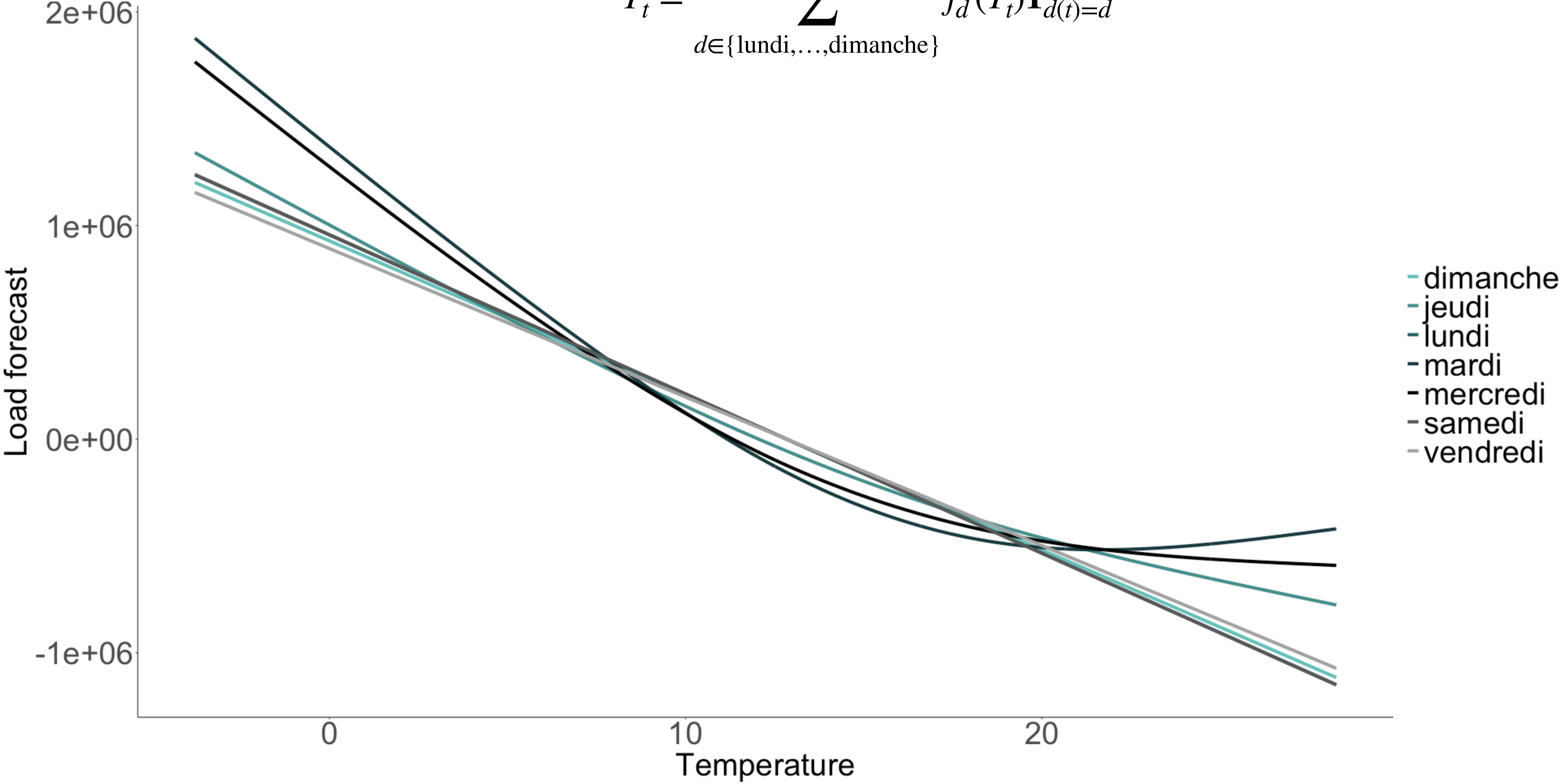
- can represent non-linear relationships
- can model bivariates relationships

but

- the more the linear model is modified, the less it can be interpreted

Example

$$\hat{Y}_t = \sum_{d \in \{\text{lundi},\ldots,\text{dimanche}\}} f_d(T_t)\mathbf{1}_{d(t)=d}$$



Decision Tree

Tree based models split the data multiple times according to threshold values of the features

They are built recursively using a segmentation criterion C (Variance criterion or Gini impurity):

For a node $\mathcal{N}$, feature $j^\star$ and the associated threshold value $s^\star$ to cut $\mathcal{N}$ are chosen according:

$$(j^\star, s^\star) \in \arg\min_{j,s} \mathrm{C}(\mathcal{N}^+) + \mathrm{C}(\mathcal{N}^-) \text{ , where } \mathcal{N}^- = \{i \,|\, X_{i,j} < s\} \text{ and } \mathcal{N}^+ = \{i \,|\, X_{i,j} \geq s\}$$

With $\mathcal{L}$ the leaves of the tree, the prediction is

Regression (mean of the leaf): $\hat{Y}_{\text{new}} = \sum_{\ell \in \mathcal{L}} \frac{\mathbf{1}_{X_{\text{new}} \in \ell}}{|\,\ell\,|} \sum_{i \in \ell} Y_i$

Classification (vote in the leaf): $\hat{Y}_{\text{new}} \in \arg\max_{m} \sum_{\ell \in \mathcal{L}} \mathbf{1}_{X_{\text{new}} \in \ell} \sum_{i \in \ell} \mathbf{1}_{Y_i = m}$

Decision Tree - Global interpretation

Contribution of a split k which splits $\mathcal{N}_k$ into $\mathcal{N}_k^-$ and $\mathcal{N}_k^+$ measures how much it has reduced the criterion compared to the parent node:

$$\imath(k) = C(\mathcal{N}_k) - \left(\mathrm{C}(\mathcal{N}_k^+) + \mathrm{C}(\mathcal{N}_k^-)\right)$$

Importance of feature j is the scaled sum of the contributions of the splits considering j:

With K_j the set of splits related to feature j:

$$I(j) = \frac{\sum_{k \in K_j} \imath(k)}{\sum_{j'} \sum_{k' \in K_{j'}} \imath(k')}$$

Decision Tree - Local interpretation

$\hat{Y}_{\text{new}}$ can be explained by decomposing the decision path into one component per feature

Data point X_{new} starts at the root $\mathcal{N}_0$ and goes thought $\mathcal{N}_1, \ldots, \mathcal{N}_{D-1}$ until the leaf $\mathcal{N}_D$

With $\bar{Y}_{\mathcal{N}} = \dfrac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} Y_i$ (regression) or $\bar{Y}_{\mathcal{N}} \in \arg\max_m \sum_{i \in \mathcal{N}} \mathbf{1}_{Y_i = m}$ (classification), $\hat{Y}_{\text{new}}$ decomposes:

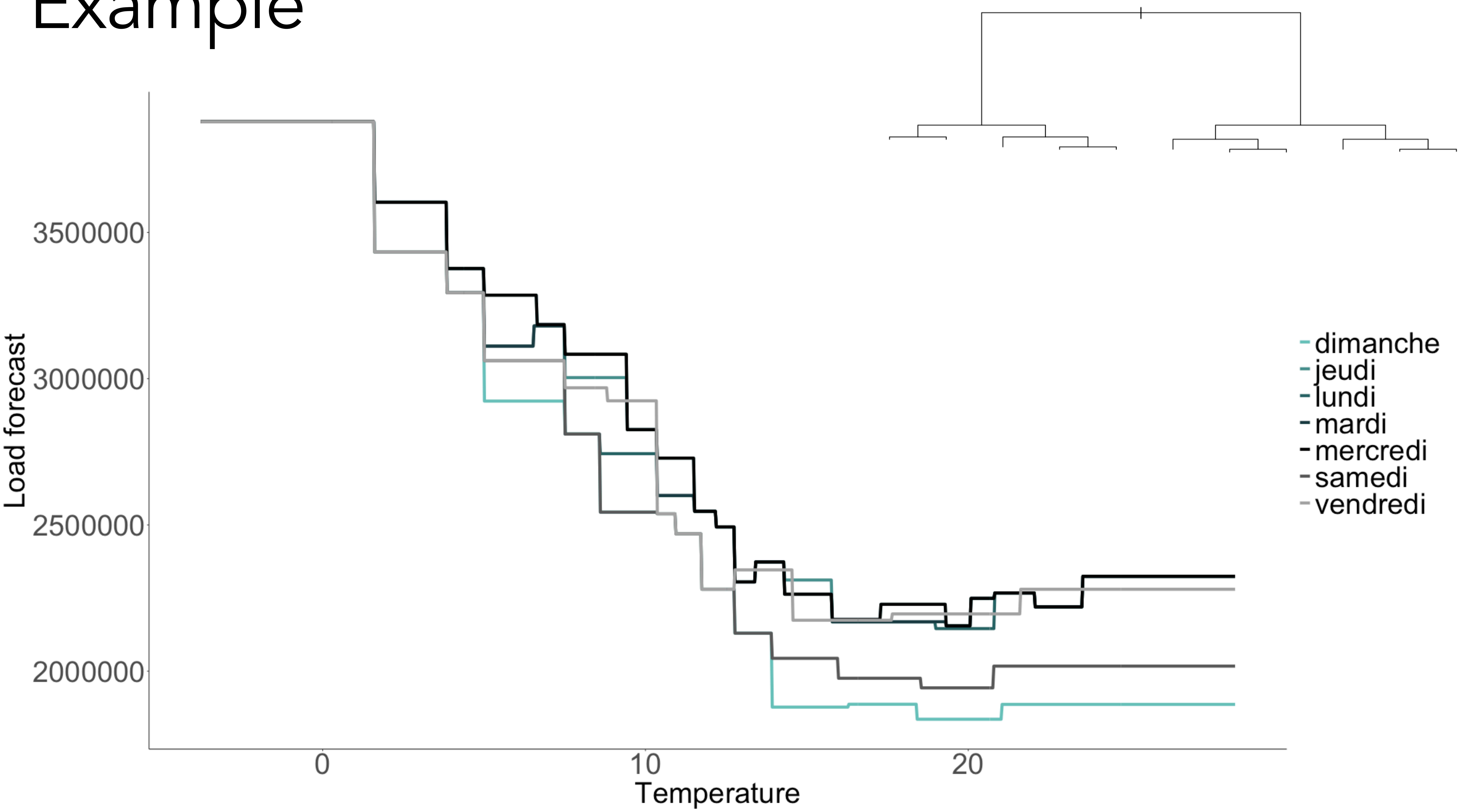
$$\hat{Y}_{\text{new}} = \bar{Y}_{\mathcal{N}_D} = \bar{Y}_{\mathcal{N}_0} + \sum_{d=0}^{D-1} \left(\bar{Y}_{\mathcal{N}_{d+1}} - \bar{Y}_{\mathcal{N}_d}\right) = \bar{Y}_{\mathcal{N}_0} + \sum_j \Delta_j(X_{\text{new}})$$

where $\Delta_j(X_{\text{new}})$ is the sum of the terms related to the feature j:

$$\Delta_j(X_{\text{new}}) = \sum_{d=1\ldots,D \,|\, j(d)=j} \left(\bar{Y}_{\mathcal{N}_{d+1}} - \bar{Y}_{\mathcal{N}_d}\right)$$

where $j(d)$ is the feature used to split $\mathcal{N}_d$

Example



Load forecast
3500000
3000000
2500000
2000000
0
10
20
Temperature
dimanche
jeudi
lundi
mardi
mercredi
samedi
vendredi

Pros and cons

Decision tree

- is ideal for capturing interactions between features in the data
- as a natural visualisation, with its nodes and edges
- invites to think about predicted values for individual input as counterfactual: if a feature had been greater / smaller than the split point, the prediction would have been y_1 instead of y_2 etc.
- does not require feature transformation

but it

- is interpretable as long as it remains short
- fails to deal with linear or ``smooth'' relationships in this case (it needs many splits)
- is quite unstable (a few changes in the training data set may lead to a completely different tree)

Model-Agnostic methods

Partial dependence plot

Partial dependence plot shows the marginal effect some features have on the prediction

For a set of features J the partial dependence function is the $|J|$-dimensional function:

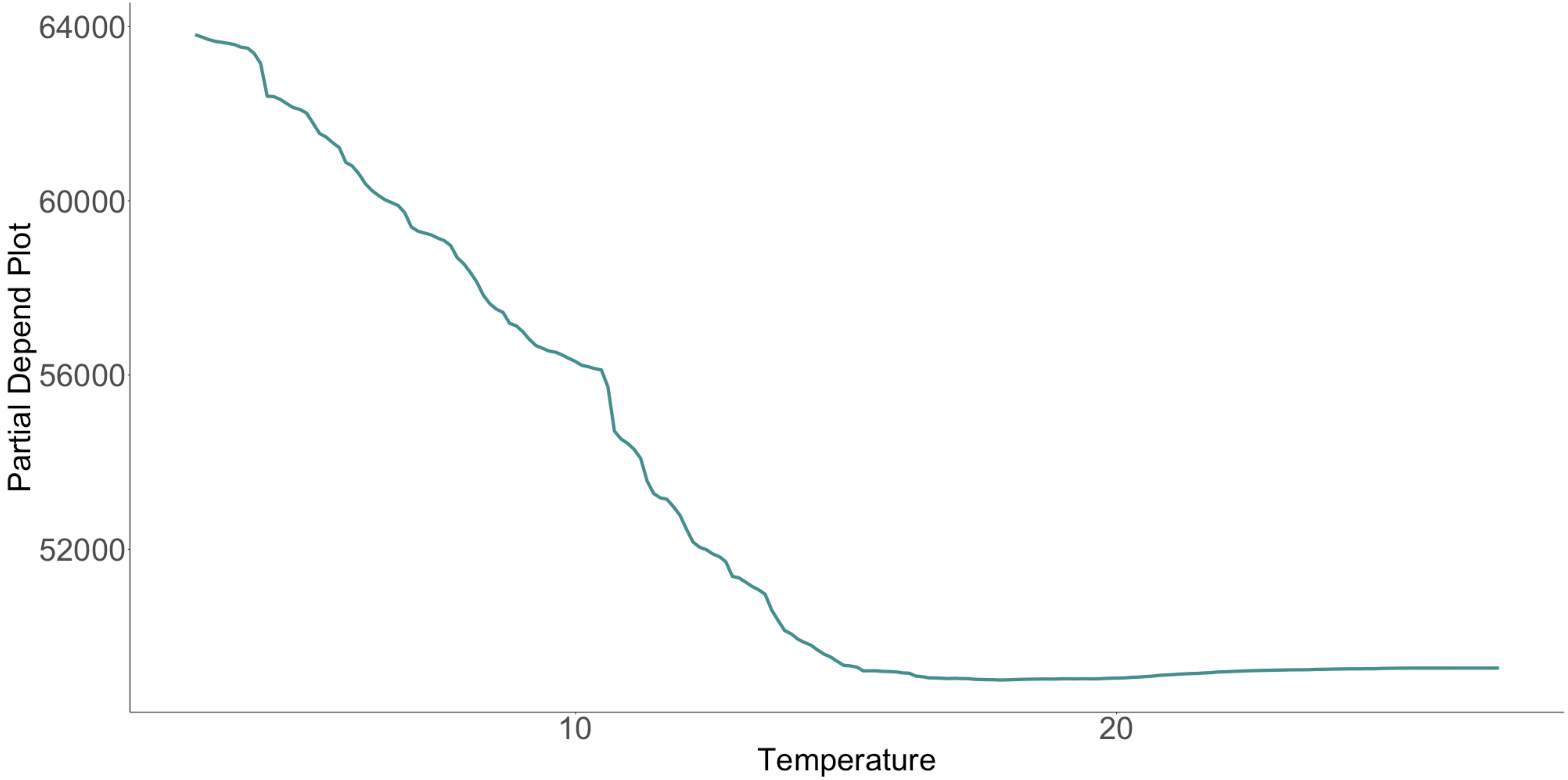
$$\mathrm{PD}_J(x) = \mathbb{E}_{X_{-J}}\left[\hat{f}(x, X_{-J})\right] = \int \hat{f}(x, x_{-J}) \mathrm{d}\mathbb{P}(x_{-J})$$

It is estimated by calculating empirical means in the training data - Monte Carlo approach:

$$\mathrm{PD}_J(x) \approx \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x, X_{i,-J})$$

- Underlying assumption: features in J are not correlated with features in $-J$ (otherwise calculations include data points that are very unlikely or even impossible)
- Heterogeneous effects might be hidden: PD only shows the average marginal effects (if for a feature half data points have a positive association with the prediction and the other half has a negative association, then the PD curve could be a horizontal line)

Example - Partial dependence plot



Individual Conditional Expectation (ICE)

ICE plot visualizes the dependence of the prediction on a feature j for each data point separately, resulting in n lines, compared to one line overall in partial dependence plots:

For each $i = 1, \ldots, n$ (or a sub-sample),

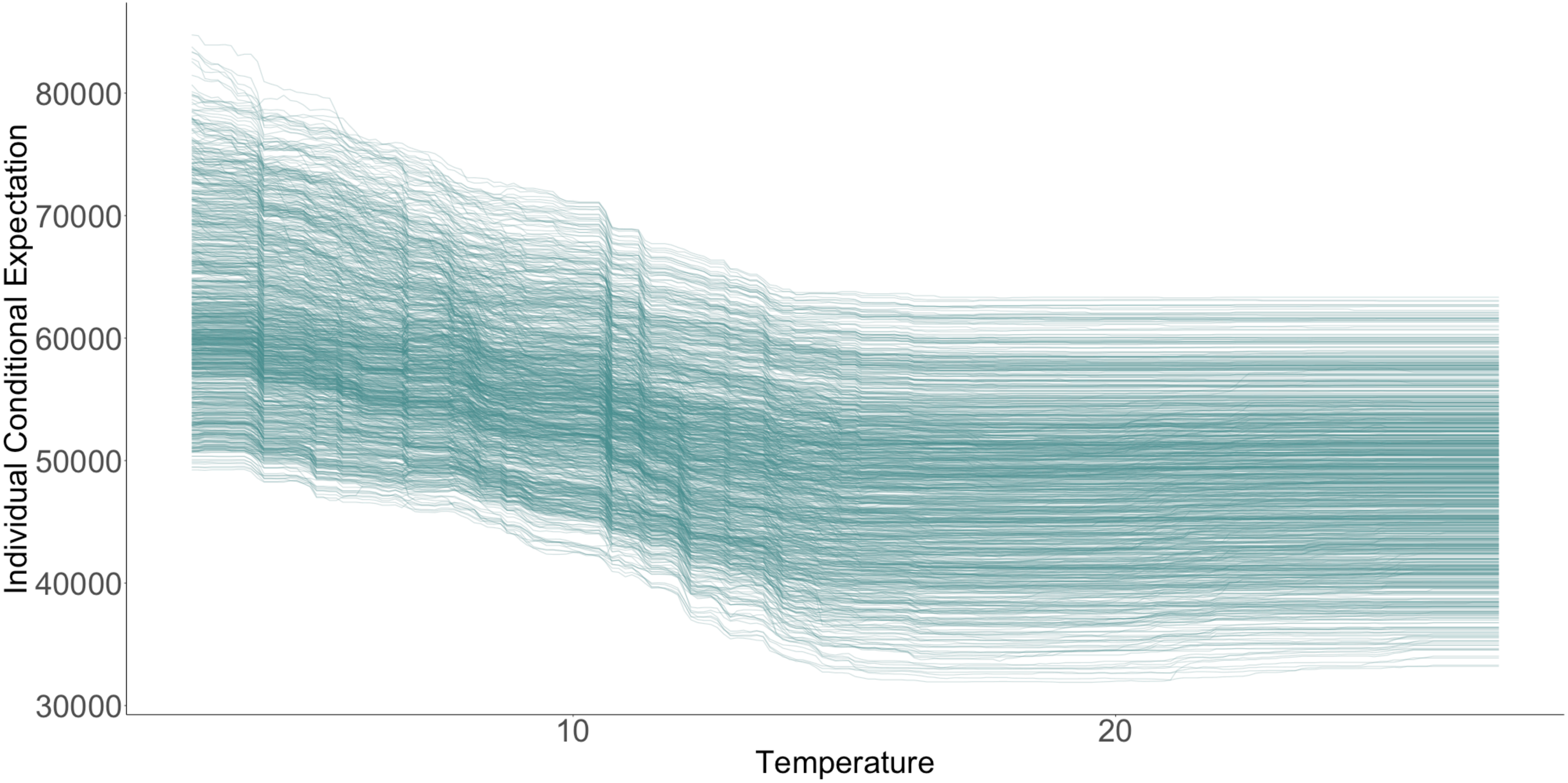
$x \mapsto \hat{f}(x, x_{i,-j})$ is plotted

PDP is the average of the lines of an ICE plot.

Centered ICE

It may be hard to tell whether the ICE curves differ between inputs because they start at different predictions, the solution is to center the curves at a certain point and display only the differences: $x \mapsto \hat{f}(x, x_{i,-j}) - \hat{f}(x_{\text{anchor},j}, x_{i,-j})$

Example - Individual Conditional Expectation



Accumulated Local Effects (ALE)

If features are correlated, PDPs are biased and cannot be trusted (computation uses artificial data points that are unlikely in reality)

With $N_j(x)$ a neighbourhood of x , the marginal plot function

$$\mathrm{M}_j(x) = \mathbb{E}_{X_{-J}|X_J=x}\left[\hat{f}(X_J, X_{-J}) \mid X_J\right] = \int_{x_{-J}} \hat{f}(x, x_{-J}) \mathrm{d}\mathbb{P}(x_{-J} \mid x_J = x) \approx \frac{1}{|N_j(x)|} \sum_{i \in N_j(x)} \hat{f}(x, x_{i,-j}),$$

avoids averaging predictions of unlikely data inputs, but mixes the effect of feature j with the effects of all correlated features

Accumulated Local Effects plots solve this problem by calculating – also based on the conditional distribution of the features – changes (defined as the gradient) in predictions instead of averages

Accumulated Local Effects (ALE)

Gradients are accumulated locally by integrating over the range of feature j up to x

Instead of directly averaging the predictions, ALE method calculates the prediction differences conditionally to feature j and integrates the derivative over features j

Derivation and integration usually cancel each other out but here the derivative isolates the effect of the feature of interest and blocks the effect of correlated features

By subtracting a constant, the ALE plot is centred so the average effect over the data is zero

$$\mathrm{ALE}_j(x) = \int_{x_j^0}^{x} \mathbb{E}_{X_{-j}|X_j}\left[\frac{\partial \hat{f}}{\partial x_j}(X_j, X_{-j}) \mid X_j = s\right] \mathrm{d}s - \mathrm{constant}$$

where x_j^0 is some value chosen near the lower bound of the effective support of feature j (not crucial, since it only affects the vertical translation of the ALE and the constant in will be chosen to vertically center the plot)

Accumulated Local Effects (ALE)

To estimate $\mathrm{ALE}_j(x)$, let consider a sufficiently fine partition $\{N_j(k) =]z_{k-1}, z_k] \mid k = 1,2,\ldots,K\}$ of the sample range of $\{x_{j,i}\}_{i=1\ldots,n}$ (take the k/K empirical quantiles of $\{x_{j,i}\}_{i=1\ldots,n}$ for z_k)

The non-centered ALE is approximated with

$$
\begin{aligned}
\widetilde{\mathrm{ALE}}_j(x) &\approx \int_{x_j^0}^{x} \frac{1}{|N_j(s)|} \sum_{i\in N_j(s)} \frac{\partial \hat{f}}{\partial x_j}(s, X_{-j,i})\mathrm{d}s \\
&\approx \sum_{k=1}^{k(x)} \left(s_k - s_{k-1}\right) \times \frac{1}{|N_j(s)|} \sum_{i\in N_j(s_k)} \frac{\hat{f}(s_k, X_{i,-j}) - \hat{f}(s_{k-1}, X_{i,-j})}{s_k - s_{k-1}} \\
&\approx \sum_{k=1}^{k(x)} \frac{1}{|N_j(s)|} \sum_{i\in N_j(s_k)} \hat{f}\left(s_k, X_{i,-j}\right) - \hat{f}\left(s_{k-1}, X_{i,-j}\right),
\end{aligned}
$$

where $k(x)$ is the index of the interval into which x falls, i.e., $x \in]s_{k-1}, s_k]$

Three levels approximation:

- on the expectation (as for marginal plots)
- on the integral (sum on the $k(x)$ intervals)
- on the derivative (finite differences)

Accumulated Local Effects (ALE)

Finally, the ALE is centered so that the mean effect is zero:

$$\mathrm{ALE}_j(x) = \widetilde{\mathrm{ALE}}_j(x) - \sum_{i=1}^{n} \widetilde{\mathrm{ALE}}_j(X_{j,i})$$

$\mathrm{ALE}_j(x)$ shows how the model predictions change in a small "window" of feature j around x for data points in that window, it can be interpreted as the main effect of feature j at x compared to the average prediction of the data

For example, an ALE estimate of 2 at $X_j = 3$ means that when feature j has value 3, then the prediction is lower by 2 compared to the average prediction

Because of quantiles, intervals can have very different lengths (weird ALE_j if j is very skewed)

ALE for the interaction of two features and more

ALE plots can also show the interaction effect of two features j_1 and j_2. The calculation are the same (rectangular cells instead of intervals because effects are accumulated in two dimensions):

$$\mathrm{ALE}_{j_1,j_2}(x_1,x_2) = \int_{x_{j_1}^0}^{x_1}\int_{x_{j_2}^0}^{x_2} \mathbb{E}_{X_{-j_1,j_2}|X_{j_1},X_{j_2}}\left[\frac{\partial^2 \hat{f}}{\partial x_{j_1}\partial x_{j_2}}(X_{j_1},X_{j_2},X_{-j_1,j_2}) \mid X_{j_1}=s_1, X_{j_2}=s_2\right] \mathrm{d}s_1 \mathrm{d}s_2 - \text{constant}$$

ALE for two features estimate the second-order effect, which does not include the main effects of the features; it is the additional interaction effect of the features (if two features do not interact, the 2D ALE should be close to zero)

PDPs always show the total effect, while ALE plots show the first - or second-order effect.

The formula generalises to any number of features, and Apley and Zhu. (2020) shows that:

$$\hat{f}(x) = \sum_{J\subseteq\{1,\ldots,p\}} \mathrm{ALE}_J(x_J)$$

ALE for categorical features

ALE needs the feature values to have an order, because the method accumulates effects in a certain direction

For a categorical features; an order has to be defined

It influences the calculation and interpretation of ALEs

→ Order the categories according to their similarity based on the other features (sum over the distances of each feature)

Pros and cons

ALE plots are

- unbiased (still work when features are correlated)
- faster to compute than PDPs
- easy to interpret, centered at zero and 2D ALE plot only shows the interaction

but they

- may be shaky with a high number of intervals (↘ number of intervals makes the estimates more stable but smooths the true complexity of the model)
- have a much more complex and less intuitive implementation compared to partial dependence plots

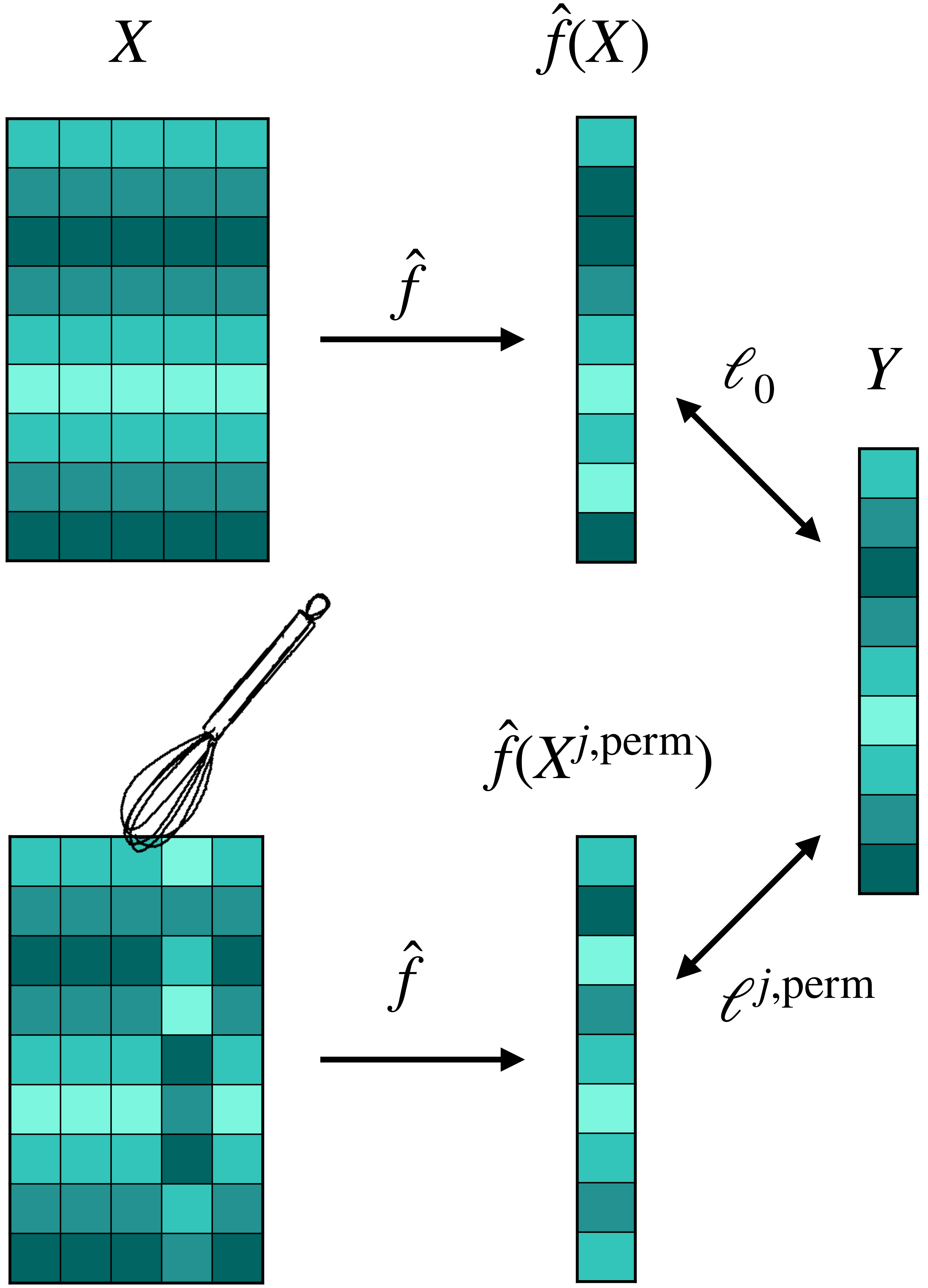
Feature importance

The importance of a feature is the increase in the model's prediction error after permuting the feature

Fisher et al. (2019) proposes the permutation feature importance algorithm:

```
Estimate the original model error $\ell_0 = \ell\left(Y, \hat{f}(X)\right)$
For $j = 1\ldots, d$:
    • Generate feature matrix $X^{j,\text{perm}}$ by permuting $j$ in $X$
    • Estimate the error $\ell^{j,\text{perm}} = \ell\left(Y, \hat{f}(X^{j,\text{perm}})\right)$
    • Calculate the feature importance
```

$$\mathrm{FI}^j = \frac{\ell^{j,\text{schuffle}}}{\ell_0} \text{ or } \mathrm{FI}^j = \ell^{j,\text{perm}} - \ell_0$$



Feature importance - Training or testing data?

It depends on the expected values for the importance of a feature that has no relationship with Y but for which the model has found one (i.e. over-fitting on this feature)

Should it be zero because the feature does not contribute to improved performance on unseen test data?

→ testing data is good choice

Should the importance reflect how much the model depends on the feature, regardless whether the learned relationship generalize to unseen data?

→ training data is good choice

Example - Feature importance

Feature	Importance on training set	Importance on testing set
Temperature	2.1%	0.6%
Nebulosity	0.3%	0.0%
Humidity	0.8%	0.2%
Position in the year	2.2%	0.5%
Month	0.9%	0.1%
Half-hour	7.3%	4.6%
Day of the week	3.8%	2.4%
Holiday / Working dat	0.2%	0.1%
Type of Holiday	0.1%	0.0%
Holidays	0.4%	0.1%
Smoothed temperature short	9.5%	7.4%
Smoothed nebulosity	1.1%	0.1%
Smoothed humidity	3.1%	0.9%
Smoothed temperature long	2.0%	0.2%

Pros and cons

Feature importance

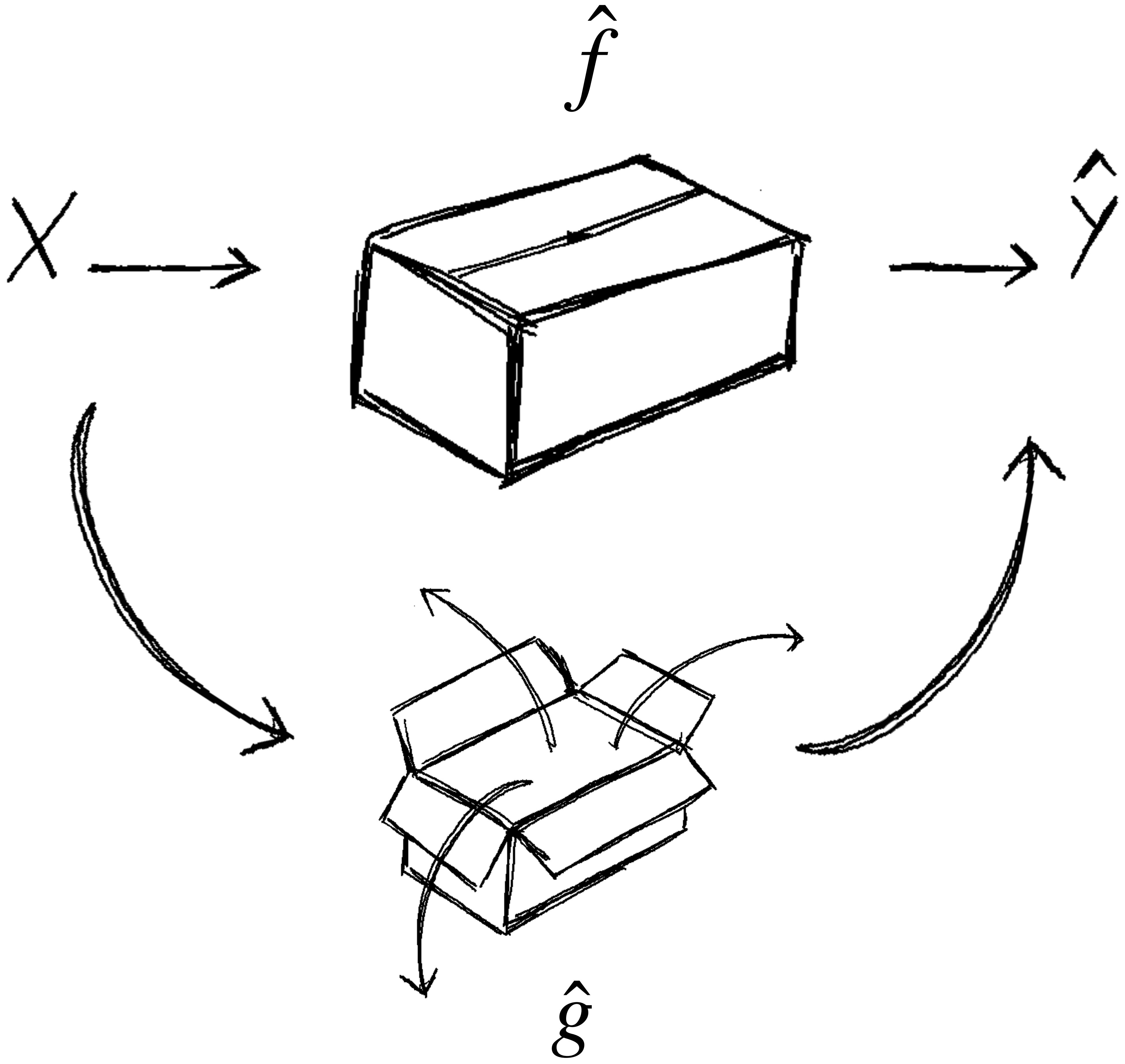
- provides a highly compressed, global insight into the model's behaviour
- automatically takes into account all interactions with other features (the importance of the interaction between two features is included in the importance measurements of both features...)
- does not require retraining the model (some other methods suggest deleting a feature, retraining the model and then comparing the model error)

but it

- requires access to the true outcome Y
- depends on the permutation of the feature vector and might vary greatly (repeating the permutation and averaging the importance measures over repetitions stabilises the measure)
- is biased by unrealistic data inputs when features are correlated
- decreases when a correlated feature is added

Surrogate model

A global surrogate model is an interpretable model (linear model, decision tree, for e.g.) $\hat{g}$ that is trained (using the data set than for the original model training or a new one) to approximate as accurately as possible the predictions of the black box model $\hat{f}$



Pros and cons

Global Surrogate

- is flexible
- is very intuitive and straightforward
- comes with a measure of how good it approximates the black box model (R-squared, for e.g.)

but it

- draws conclusions about the model and not about the data, since it never sees the real outcomes Y
- can very close to the original model for one subset of the data-set, but widely divergent for another subset
- comes with all the advantages and disadvantages of the chosen interpretable model

Shapley values

A prediction is explained by assuming that each feature value of the input is a "player" in a collaborative game where the prediction is the payoff

Shapley values fairly distribute the "payoff"/prediction among the players/features (from coalition game theory, Shapley 1953)

Example - linear model: the contribution of feature j in the prediction $\hat{f}(x)$ is

$$\phi_{j,\hat{f}}(x) = \hat{\beta}_j x_j - \hat{\beta}_j \mathbb{E}(X_j)$$

and the prediction is the mean predicted value plus the sum of the contributions (which can be negative):

$$\hat{f}(x) = \mathbb{E}\left(\hat{f}(X)\right) + \sum_j \phi_{j,\hat{f}}(x)$$

Shapley values in game theory

In a cooperative game, with $v(S)$ the payoff of a coalition of players $S \subseteq \{1,2,\ldots,p\}$, the Shapley value of player j is

$$\begin{aligned}\phi_{j,v} &= \frac{1}{\text{number of players}} \sum \text{coalition including } j \; \frac{\text{contribution of } j \text{ to coalition}}{\text{number of coalition including } j} \\ &= \frac{1}{p} \sum_{S \subseteq \{1,2,\ldots,p\} \backslash \{j\}} \frac{1}{\binom{p-1}{|S|}} \Big(v(S \cup \{j\}) - v(S) \Big)\end{aligned}$$

The Shapley value is the only attribution method that satisfies the properties of a fair payout:

- Efficiency: dividing by the number of players ensures that $v\big(\{1,2,\ldots,p\}\big) = \displaystyle\sum_{j=1}^{p} \phi_{j,v}$
- Symmetry: if $\forall S \backslash \{i,j\}\$, \$v(S \cup \{i\}) = v(S \cup \{j\})$ then $\phi_{i,v} = \phi_{j,v}$
- Dummy: if $\forall S \backslash \{i\}$, $v(S \cup \{i\}) = v(S)$ then $\phi_{i,v} = 0$
- Additivity: for a game combined payouts $v + v'$, the respective Shapley values are $\phi_{i,v} + \phi_{i,v'}$

Shapley values for Machine Learning

Players = features $x_1, \ldots, x_p$ and payoff = prediction $\hat{f}(x)$

For a set of features S taking values x_S, as it is generally not possible to compute predictions without the features $\bar{S} = \{1,2,\ldots,p\} \backslash S$, they are replaced in the calculation of $v(x_S)$

$v(x_S)$ is defined as the prediction for feature values x_S in that are marginalised over features in $\bar{S}$:

$$v(x_S) = \mathbb{E}_{X_{\bar{S}}}\left[\hat{f}(x_S, X_{\bar{S}})\right] = \int \hat{f}(x_S, X_{\bar{S}}) \mathrm{d}\mathbb{P}_{X_{\bar{S}}}$$

For more than a few features in J, the exact solution becomes problematic as the number of possible coalitions exponentially increases

→ Approximation with Monte-Carlo sampling (Štrumbelj and Kononenko, 2014) of the Shapley value of a feature j

Shapley values approximation

For $m = 1,\ldots,M$:

- Draw a random input z from the data
- Pick a random subset of feature indices $J \subseteq \{1,\ldots,p\}\backslash\{j\}$
- Construct the two inputs:

  $x_{+j}^{m} = (x_J, z_{-J}, x_j)$

  $x_{-j}^{m} = (x_J, z_{-J}, z_j)$

- Compute the contribution (corresponds to “$v(S \cup \{i\}) - v(S)$” in the game theory formulation of Shapley values):

  $\phi_j^m(x) = \hat{f}(x_{+j}^m) - \hat{f}(x_{-j}^m)$

Compute Shapley value as the average:

$$\phi_j(x) = \frac{1}{M}\sum_{m=1}^{M} \phi_j^m(x)$$

Pros and cons

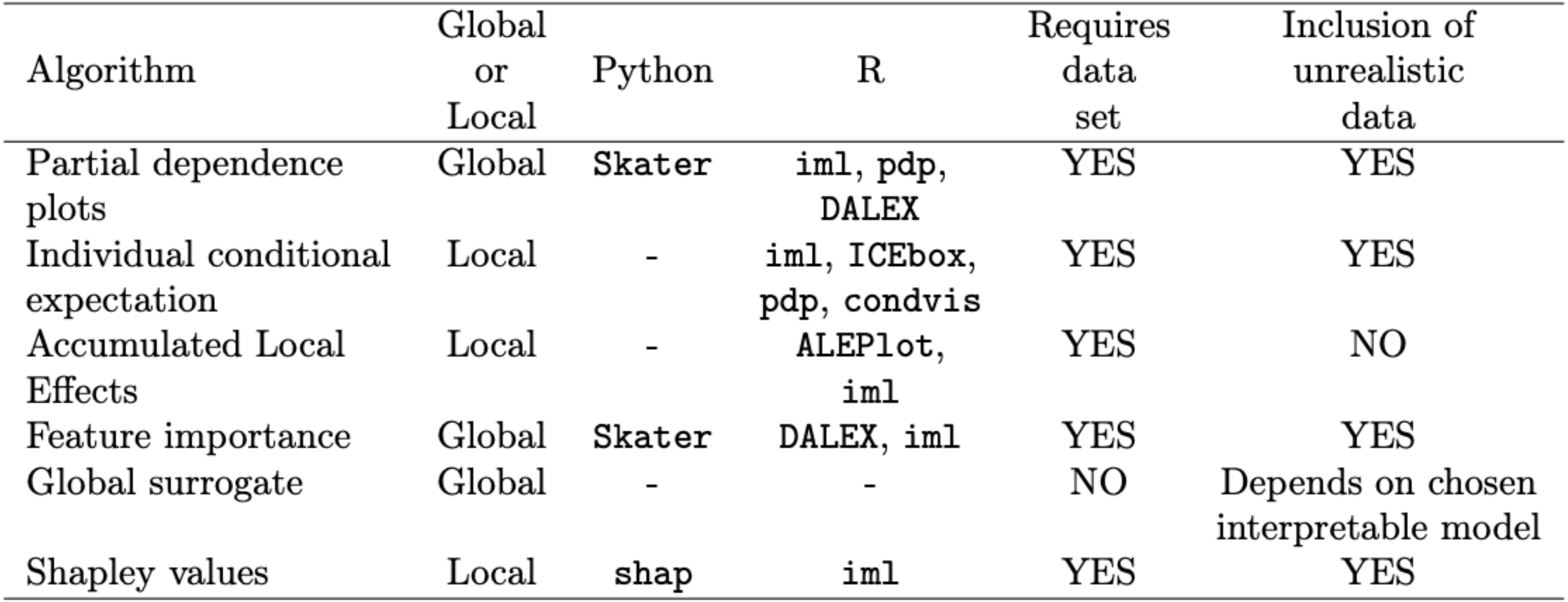
Shapley values

- are efficient: the difference between the prediction and the average prediction is fairly distributed among the feature values of the input and give a clear interpretation
- allows contrastive explanations: instead of comparing a prediction to the average prediction of the entire data-set, it is possible to compare it to a subset or even to a single data point
- are the only explanation method with a solid theory

but they

- require a lot of computing time - only approximate solution are feasible
- return a simple value per feature, so it cannot be used to make statements about changes in prediction for changes in the input
- require access to the data
- suffer from inclusion of unrealistic data points when features are strongly correlated

Implementation



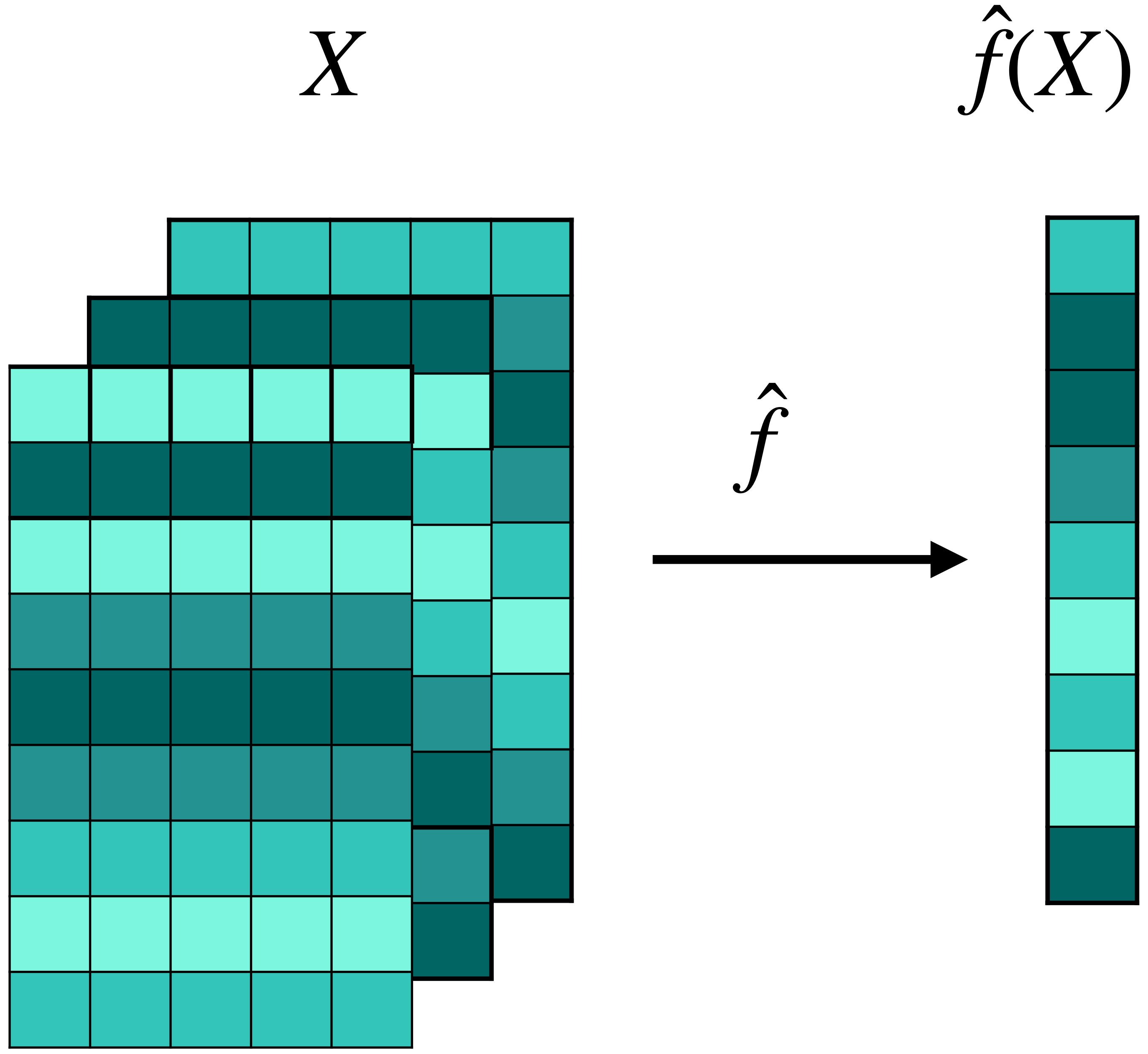
Algorithm	Global or Local	Python	R	Requires data set	Inclusion of unrealistic data
Partial dependence plots	Global	`Skater`	`iml`, `pdp`, `DALEX`	YES	YES
Individual conditional expectation	Local	-	`iml`, `ICEbox`, `pdp`, `condvis`	YES	YES
Accumulated Local Effects	Local	-	`ALEPlot`, `iml`	YES	NO
Feature importance	Global	`Skater`	`DALEX`, `iml`	YES	YES
Global surrogate	Global	-	-	NO	Depends on chosen interpretable model
Shapley values	Local	`shap`	`iml`	YES	YES

Example-based methods

Example-based explanation methods select particular inputs of the data-set

- to explain the behaviour of machine learning models
- to explain the underlying data distribution

They explain a model by selecting inputs of the data-set and not by creating summaries of features (model-agnostic methods)



Counterfactual explanations

How an input has to change to significantly change its prediction?

Counterfactual explanation = smallest change in the feature values that changes the prediction to a predefined output - flip in predicted class (classification) or reaching some threshold (regression)

- Counterfactual X' should be as similar as possible to X

  → requires a distance measure d between two inputs (Manhattan distance, e.g.)
- Counterfactual X' should have feature values that are likely
- Prediction has to change significantly

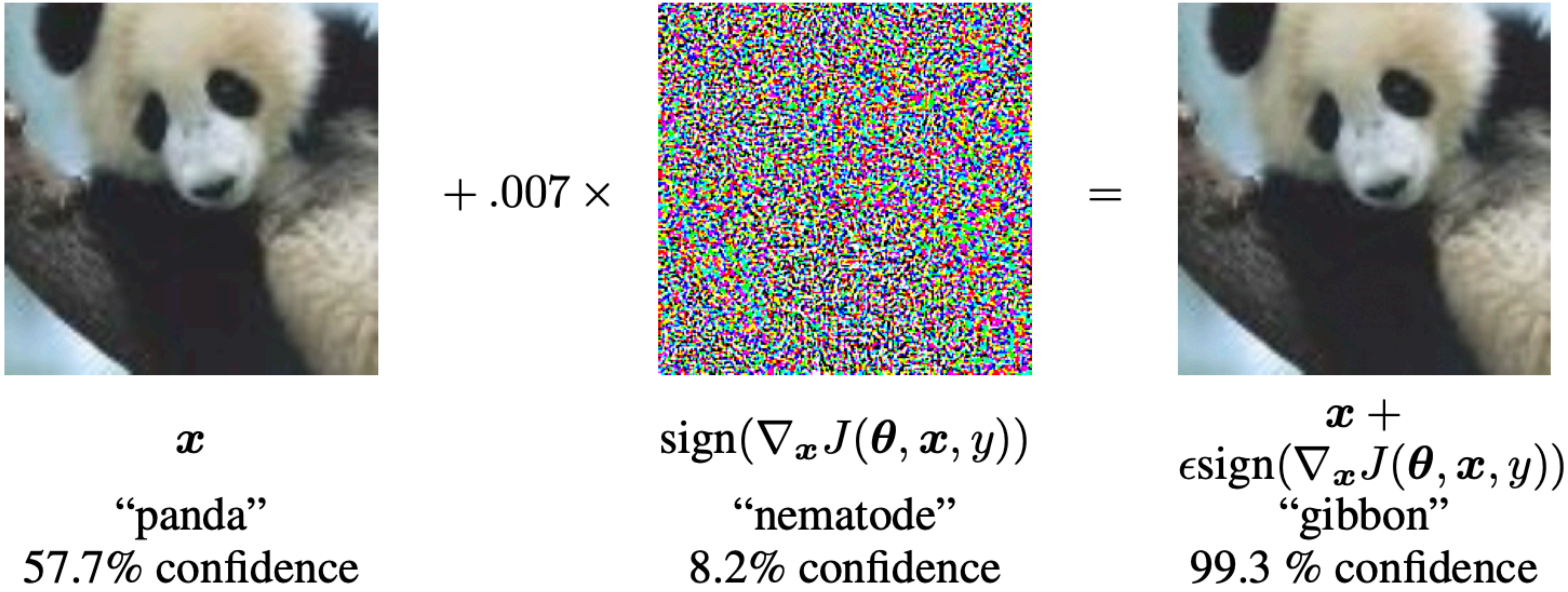
  → requires a desired outcome Y'

Wachter et al. (2017) suggests to minimise

$$L(X, X', Y', \lambda) = \lambda\left(\hat{f}(X') - Y'\right)^2 + d(X, X')$$

Adversarial examples

Adversarial examples are counterfactual examples used to fool machine learning model


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Goodfellow, et al. Explaining and harnessing adversarial examples (ICLR 2015)

Prototypes and Criticisms (Kim et al., 2016)

Prototypes: data input that is representative of all the dat

Maximum mean discrepancy method finds m prototypes $z_1, z_2, \ldots, z_m$ by minimising

$$\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} k(z_i, z_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(z_i, x_j) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x_j),$$

where k is a kernel function that measures the similarity of two inputs

Criticisms: data input that is not well represented by the set of prototype (extremes)

With,

$$\text{witness}(x) = \frac{1}{n}\sum_{i=1}^{n} k(x, x_i) - \frac{1}{m}\sum_{i=1}^{m} k(x, z_i)$$

criticisms are the extreme values of the witness function in both negative and positive directions

Prototypes and Criticisms

$$\text{witness}(x) = \frac{1}{n}\sum_{i=1}^{n} k(x, x_i) - \frac{1}{m}\sum_{i=1}^{m} k(x, z_i)$$

$\text{witness}(x) > 0 \rightarrow$ the prototype distribution overestimates the data distribution (a prototype has been selected but there are only few data points nearby)

$\text{witness}(x) < 0 \rightarrow$ the prototype distribution underestimates the data distribution (there are many data points around $x$but any prototypes nearby)

Prototypes and Criticisms

Prototypes and Criticisms are useful to

- Understand the data distribution

- Build a nearest prototype interpretable model:

$$\hat{f}(x) = Y_{j^\star} \text{ with } j^\star \in \arg\max_{j=1\ldots,m} k(x, z_j)$$

- Make a black box model interpretable by analysing the predictions of the prototypes and criticisms: in which cases was the algorithm wrong?

That’s all folks!