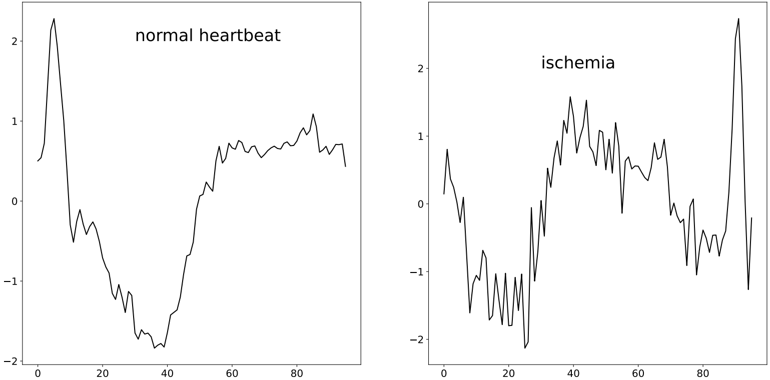
6.4. LIMESegment

Time series classification

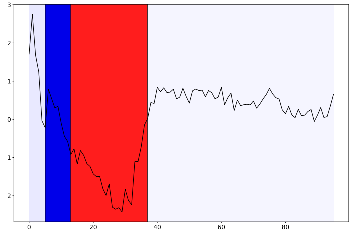
- **Time series:** ordered sequence of T observations
- **Example:**[44] ECG from one heartbeat, detect ischemia or not



[44] Olszewski, *Generalized feature extraction for structural pattern recognition in time-series data*, Carnegie Mellon, 2001

LIMESegment

- **Idea:**[45] adapt the LIME framework to time series
- similar high-level operation (differences in **bold**):
  1. **create interpretable features**
  2. **sample** n perturbed samples $x_1, \ldots, x_n$ from ξ
  3. **weight** the x_is
  4. train a local surrogate model
- **Output:** highlight important parts of the time-series



[45] Sivill, Flach, *LIMESegment: Meaningful, Realistic Time Series Explanations*, AISTATS, 2022

Step 1: interpretable features

- **Interpretable features:** homogeneous segments in the time series
- standard problem (usually called *change-point detection*[46])
- proposed methodology: `NNSegment`
- **Reminder:** empirical mean: let $A \in \mathbb{R}^\ell$,

$$\overline{A} := \frac{1}{\ell} \sum_{i=1}^{\ell} A_i \, .$$

- **Reminder:** empirical covariance / variance:

$$\widehat{\mathrm{Cor}}(A,B) := \frac{1}{\ell - 1} \sum_{i=1}^{\ell - 1} (A_i - \overline{A})(B_i - \overline{B}) \, , \qquad \widehat{\mathrm{Var}}(A) := \frac{1}{\ell - 1} \sum_{i=1}^{\ell} (A_i - \overline{A})^2 \, .$$

[46] Truong, Oudre, Vayatis, *Selective review of offline change-point detection methods*, Signal Processing, 2020

Step 1: interpretable features

- let w_s be a fixed window size, define

$$x_{a:b} := (x_a, x_{a+1}, \ldots, x_b)^\top .$$

- for a given *window size* w_s, define $w_i := x_{i:(i+w_s)}$
- **Definition:** normalized cross-correlation (*a.k.a.* sample correlation):

$$\forall s_1, s_2 \in [T - w_s], \qquad \psi(w_{s_1}, w_{s_2}) := \frac{\widehat{\mathrm{Cor}}(w_{s_1}, w_{s_2})}{\sqrt{\widehat{\mathrm{Var}}(w_{s_1})\widehat{\mathrm{Var}}(w_{s_2})}} .$$

- **Intuition:** higher is better (= more similar)
- **Examples:**
  - if $w_{s_1} = w_{s_2}$, then $\widehat{\mathrm{Cor}}(w_{s_1}, w_{s_2}) = 1$
  - if w_{s_1} and w_{s_2} are “independent,” then $\widehat{\mathrm{Cor}}(w_{s_1}, w_{s_2}) = 0$

Step 1: interpretable features

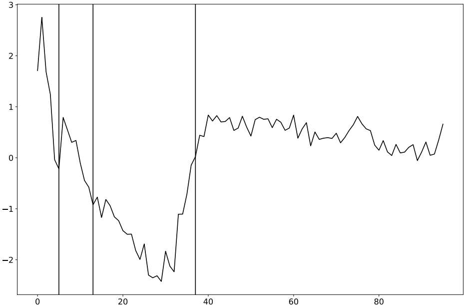
- back to `NNSegment`:
  1. compute all pairwise correlations between segments $\psi(s_1, s_2)$
  2. connect each segment ot its *nearest neighbor*
  3. group adjacent segments together (nearest neighbor $=$ next segment)
- **Further refinement:** look at difference in signal to noise ratio

$$\rho(w_i, w_j) := \left| \frac{\mu(w_i)}{\sigma(w_i)} - \frac{\mu(w_j)}{\sigma(w_j)} \right| ,$$

  and then:
  - if $\rho(w_i, w_{i-w_s}) > \rho(w_i, w_{i+w_s})$, group i with $i + w_s$
  - if $\rho(w_i, w_{i-w_s}) < \rho(w_i, w_{i+w_s})$, group i with $i - w_s$
- stop doing this when we have reached the user-specified number of segments T

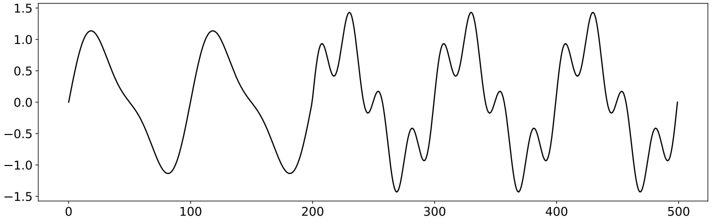
Step 1: interpretable features

- **Output:** segmented signal
- **Example:** here we obtain 4 segments, that is, 3 breakpoints



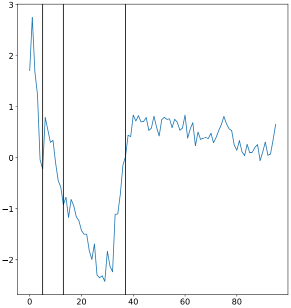
Step 2: perturbed examples

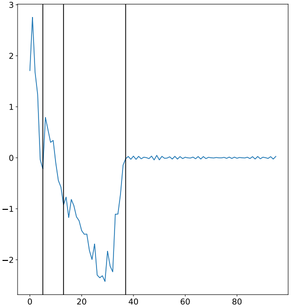
- **Idea:** identify background signal in the spectral domain
- **Discrete Short Time Frequency Transform (STFT):** $\rightarrow$ time-frequency domain
- **Example:** (local) spectrogram of superposition of sine waves



Step 2: perturbed samples

- identify a persistent frequency, map it back via inverse STFT
- **Example:** perturbing the last segment of the signal





Step 3: weights

- similar idea: exponential weights depending on a distance
- **Issue:** Euclidean distance between the z_i does not reflect distance between signals
- **Dynamic time warping (DTW):**[47] distance between signals taking alignment into account
- formally,

$$\mathrm{DTW}(x, x')^2 := \min_{\pi \in P(x,x')} \sum_{(i,i') \in \pi} d(x_i, x'_{i'}) ,$$

  where π is an *admissible path*
- namely:
  - $\pi_1 = (1, 1)$ (beginning of signals matched together);
  - $\pi_K = (S, T)$ (end of signals matched together);
  - writing π_k as (i_k, i'_k), both i and i' are non-decreasing.

[47] Bellman, Kalaba, *On adaptive control processes*, IRE Transactions on Automatic Control, 1959

Summary

- **Final steps:** surrogate model as before (ridge), coefficients given as importance
- **Main message:** a lot depends on the data-type and the kind of perturbation we want
- results depends a lot on the segmentation / sampling scheme
- no existing theoretical analysis
- many other methods[48]

[48] see Theissler et al., *Explainable AI for Time Series Classification: A review, taxonomy and research directions*, for an overview

6.5. Anchors

Notation and first definitions

- **Back to text:** ξ = document to explain = ordered sequence of tokens $(\xi_1, \ldots, \xi_T)$, f = classifier

Definition: we define an *anchor* A as an ordered subset of the words of ξ. We let $\mathcal{A}$ be the set of all possible *non-empty* anchors.

- two key definitions:
  1. *precision* = probability of same classification knowing that the document contains A
  2. *coverage* = how many documents in the dataset contain A
- one-sentence summary: find anchor with prescribed precision and maximal coverage

```
The selection on the menu is great, and so is the food! The service is not bad, prices are fine.
```

$\Longrightarrow$

$$\mathrm{Prec}(A) = 0.97$$
$$\mathrm{Cov}(A) = 0.12$$

How precision is computed

- **Formal definition:**

$$\mathrm{Prec}(A) := \mathbb{P}_A\left(f(X) = f(\xi)\right) ,$$

where X is a random perturbation of ξ containing all words in A

- **Question:** what is the distribution of "X given A" in this definition?
  - **default implementation:** i.i.d. Bernoulli for each word not in A to decide removal, replace by `UNK` token if removed (more on that later)
  - **generative model:** for instance, using BERT[49] to generate the missing words,...
  - **deterministic replacements:** get word embedding and replace by word having similar embeddings,[50]...

[49] Devlin, Chang, Lee, Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, Proc. ACL, 2019

[50] Ribeiro, Singh, Guestrin, *"Why should I trust you?" Explaining the prediction of any classifier*, ACM SIGKDD, 2016

Sampling mechanism

The selection on the menu is great, and so is the food! The service is not bad, prices are fine.

the selection on the menu is great and so is the food the service is not bad prices are fine

the selection on the menu is great and so is the food the service is not bad prices are fine

the selection on the menu is great and so is the food the service is not bad prices are fine

the selection UNKthe menu is great and so is the food the UNK is not bad prices UNK fine

Sampling mechanism

The selection on the menu is great, and so is the food! The service is not bad, prices are fine.

the selection on the menu is great and so is the food the service is not bad prices are fine

the selection on the menu is great and so is the food the service is not bad prices are fine

the selection on the menu is great and so is the food the service is not bad prices are fine

the selection on the menu is great and so is UNK food the UNK is not bad prices are fine

Sampling mechanism

The selection on the menu is great, and so is the food! The service is not bad, prices are fine.

the selection on the menu is great and so is the food the service is not bad prices are fine

the selection on the menu is great and so is the food the service is not bad prices are fine

the selection on the menu is great and so is the food the service is not bad prices are fine

UNK selection on the menu UNKgreat and UNKUNKthe UNK the UNK UNKnot bad UNK are fine

Estimating $\mathrm{Prec}(A)$

- wlog, one can assume that $f(\xi) = 1$
- thus
$$\mathrm{Prec}(A) := \mathbb{P}_A\left(f(X) = 1\right) .$$
- **Remark:** of course, impossible to compute in practice (too costly with `UNK` replacement, worse with BERT)
- **Solution:** Monte-Carlo estimate:
$$\widehat{\mathrm{Prec}}_n(A) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{f(X_i)=1} ,$$
where X_i i.i.d. draw from X
- in practice, $n = 10$

Coverage

- **Formal definition:** let $\mathcal{C}$ be a given set of documents. For any anchor A, we define

$$\mathrm{Cov}(A) := \left|\{\delta \in \mathcal{C} \quad \text{s.t.} \quad \forall w \in A, w \in \delta\}\right| \, .$$

- **Remark:** in practice, shorter anchors have higher coverage
- **Why?** think one common word: contain in many documents
- in the other direction, whole sentence $\rightarrow$ only contained in one document
- since $\mathrm{Cov}(A)$ costly to compute, Anchors minimizes $|A|$ instead of maximizing $\mathrm{Cov}(A)$

Summary

- let $\varepsilon > 0$ be some tolerance threshold (by default, $\varepsilon = 0.05$)
- **What is described originally:**

$$\operatorname*{Maximize}_{A\in\mathcal{A}} \mathrm{Cov}(A) \quad \text{subject to} \quad \mathrm{Prec}(A) \geq 1 - \varepsilon\,.$$

- **What the actual goal is:**

$$\operatorname*{Minimize}_{A\in\mathcal{A}} |A| \quad \text{subject to} \quad \widehat{\mathrm{Prec}}_n(A) \geq 1 - \varepsilon\,. \tag{$\star$}$$

- **Additional caveat:** if ξ has length b, $|\mathcal{A}| = 2^b$...
- **What is done in practice:** use KL-UCB[51] to approximately solve $(\star)$

[51] Kaufmann and Kalyanakrishnan, *Information complexity in bandit subset selection*, COLT, 2013

Visualization

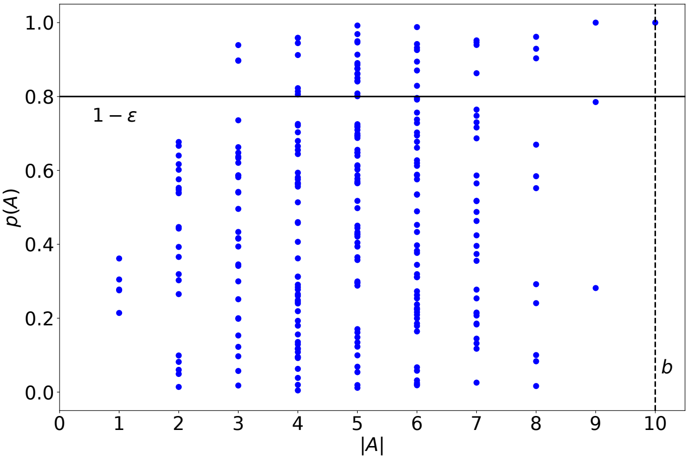


Figure: all anchors for a given example / classifier represented in the $|A|$ / $p(A) = \mathrm{Prec}(A)$ space

Visualization

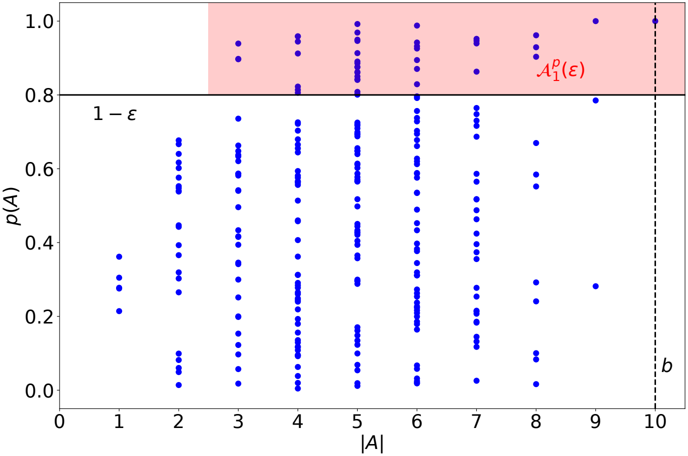


Figure: selecting $\mathcal{A}_1^p(\varepsilon)$, set of all anchors with evaluation higher than $1-\varepsilon$

Visualization

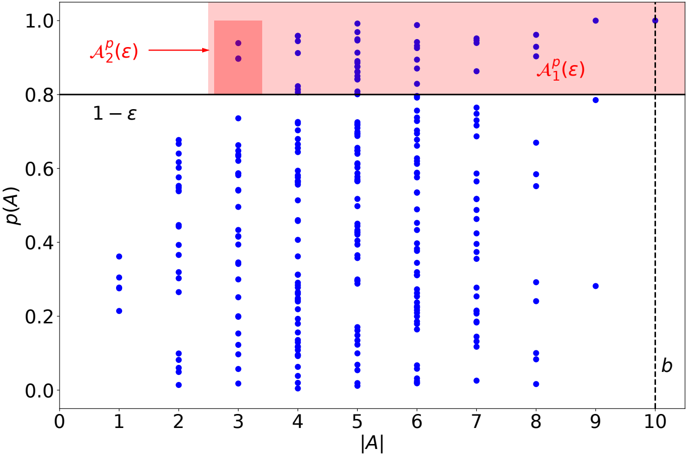


Figure: selecting $\mathcal{A}_2^p(\varepsilon)$, anchors with $p(A) \geq 1-\varepsilon$ and minimal length

Visualization

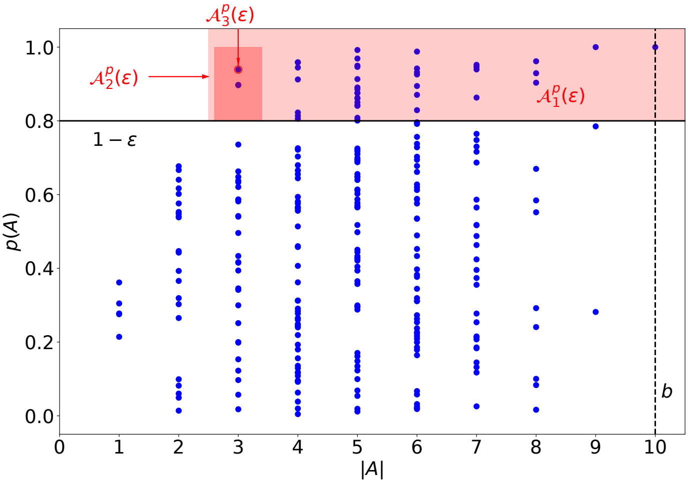


Figure: selecting $\mathcal{A}_3^p(\varepsilon)$, anchors with $p(A) \geq 1 - \varepsilon$, minimal length, and maximal $p(A)$

Summary

- rule selection via random perturbation
- interpretable features = subset of the words
- post-hoc, local method, with a global flavor
- very costly to run
- some theoretical limited theoretical analysis (indicator and linear models)[52]

[52] Lopardo, Precioso, Garreau, *A sea of words: an in-depth analysis of Anchors for text data*, AISTATS, 2023

6.6. A game-theoretical perspective: Shapley values

Shapley values

- **Setting:** D-player game[53]
- characteristic function $v : 2^D \to \mathbb{R}$, gives the *value* of a coalition S
- total sum of gains the members of S can obtain by cooperation
- **Idea:** distribute fairly the total gains to the players, assuming that they all contribute

Definition: Shapley value of player j:

$$\phi_j(v) = \sum_{S \subseteq [D] \setminus \{j\}} \frac{|S|!(D - |S| - 1)!}{D!} \left(v(S \cup \{j\}) - v(S) \right) .$$

- **Intuition:** if player j plays much better than the others, then $v(S \cup \{j\})$ consistently higher than $v(S)$, and $\phi_j(v) \gg 0$

[53] Shapley, *A value for n-person game*, Contributions to the theory of games, 1953

Properties

- Shapley values have nice theoretical properties:
  - *efficiency:* sum of Shapley values = gain of the whole coalition:
$$\sum_j \phi_j(v) = v(\{1, \dots, D\})\,.$$
  - *symmetry:* players with the same skills are rewarded equally:
$$\forall S \subseteq \{1, \dots, D\},\ v(S \cup \{j\}) = v(S \cup \{k\}) \quad \Rightarrow \quad \phi_j(v) = \phi_k(v)\,.$$
  - *linearity:* v and w two characteristic functions, then
$$\forall j \in \{1, \dots, D\}, \quad \phi_j(v + w) = \phi_j(v) + \phi_j(w)\,.$$
  - *null player:* a player that does not bring anything is not rewarded:
$$\forall j \in \{1, \dots, D\}, \quad v(S \cup \{j\}) = v(S) \quad \Rightarrow \quad \phi_j(v) = 0\,.$$

Shapley values, ctd.

- other nice properties:
  - *anonymity*
  - *standalone test*
  - ...
- more interestingly:

Theorem:[54] Shapley values are the only payment rule satisfying efficiency, symmetry, linearity, and null player.

- **Question:** connection with interpretability?
- we can see f as the reward and a subset of features as the player

[54] *ibid*

Shapley regression values

- **Example:** linear model
- for each subset of features $S \subseteq [D]$, retrain a model f_S only using the features in S

Definition:[55] the *Shapley regression value* associated to feature j is given by

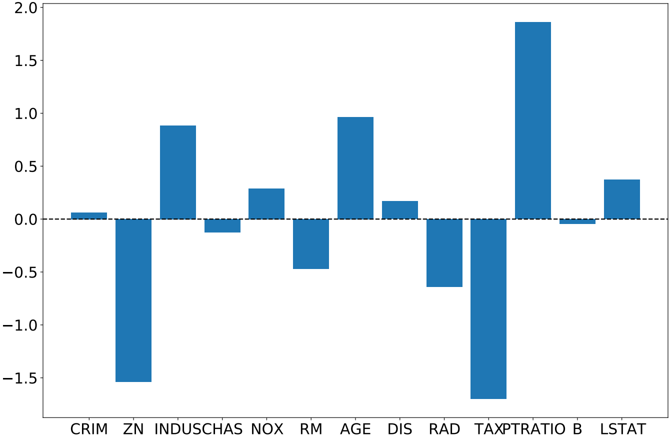
$$\phi_j := \sum_{S \subseteq [D] \setminus \{j\}} \frac{|S|!(D - |S| - 1)!}{D!} \left(f_{S \cup \{j\}}(\xi_{S \cup \{j\}}) - f_S(\xi_S) \right) ,$$

where ξ_S is the restriction of ξ to S features.

[55] Lipovetsky and Conklin, *Analysis of regression in game theory approach*, Applied Stochastic Models in business and industry, 2001

Shapley regression values

- **Example:** output for linear regressor on Boston housing dataset



Shapley sampling values

- there are two main problems with this approach:
  - *computational cost* $= \mathcal{O}\left(2^D\right)$
  - *retraining* the model each time
- a first solution: *Shapley sampling values*[56]
  - subsample in the sum over all subsets
  - instead of retraining the model, mimic the removal a variables by randomly sampling over the training set
- in other words, replace $f_S(\xi_S)$ by

$$\mathbb{E}\left[f(x) \mid x_S = \xi_S\right] .$$

- f can now be any model, provided that we can query efficiently

[56] Štrumbelj and Kononenko, *Eplaining models and individual predictions with feature contributions*, Knowledge and information systems, 2014

Kernel SHAP

- still very costly to test *all the coalitions*
- **Idea:** linear regression on the presence / absence of features
- as before, define interpretable features $z \in \{0,1\}^d$, with $d \leq D$
- $h_\xi : \{0,1\}^d \to \mathbb{R}^D$ mapping function such that $h_\xi(\mathbf{1}) = \xi$

Definition (kernel SHAP)[57]: define ϕ as the minimizer of

$$\sum_{z\in\{0,1\}^d} \frac{d-1}{\binom{d}{|z|}\cdot |z| \cdot (d-|z|)} \left(f(h_\xi^{-1}(z)) - \phi^\top z\right)^2 .$$

[57] Lundberg and Lee, *A Unified Approach to Interpreting Model Predictions*, NeurIPS, 2017

Kernel SHAP

- can be seen weighted linear regression
- computational cost: $\mathcal{O}\left(2^d + d^3\right)$
- **Remark:** not practical if $d \gg 1$
- in that case, subsample: $z_1, \ldots, z_n$ i.i.d. Bernoulli $\in \{0,1\}^d$ and minimize for $\phi \in \mathbb{R}^d$

$$\sum_{i=1}^{n} \pi_i \cdot \left(f(h_\xi^{-1}(z_i)) - \phi^\top z_i\right)^2 ,$$

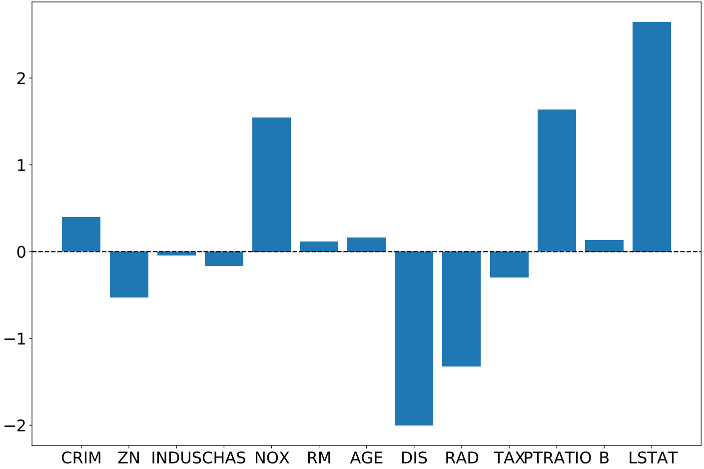
  with

$$\pi_i := \frac{d-1}{\binom{d}{|z_i|} \cdot |z_i| \cdot (d - |z_i|)} .$$

- **Remark:** very similar to LIME

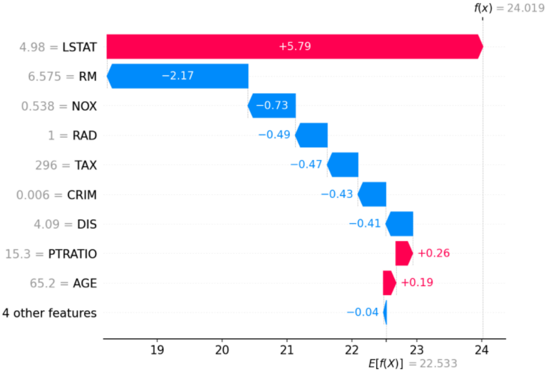
SHAP, tabular example

- **Example:** interpreting a linear model on the Boston dataset:



SHAP, tabular example

- we can also use the `shap` Python package
- really nice visualizations:



Kernel SHAP properties

- assume f is linear, that is,

$$f(x) := \sum_{j=1}^{d} \lambda_j x_j + b\,.$$

Corollary:[58] If f is linear, then $\phi_0 = b$ and

$$\phi_j = \lambda_j(\xi_j - \overline{x}_j)\,,$$

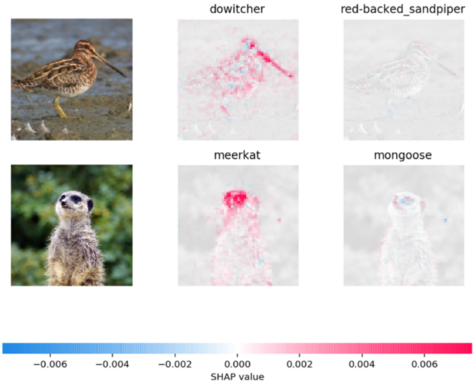
where $\overline{x}_j$ is the mean of feature j on the dataset.

- we recover the coefficients of the linear model multiplied by the (normalized) input

[58] ibid

Extensions

- Kernel SHAP is not restricted to tabular data
- **Example:** explaining the predictions of VGG16 for two classes



Summary

Advantages:

- Kernel SHAP can be used on any model
- can take advantage of specific architectures:
  - *TreeSHAP*[59] (tree-based predictors)
  - *DeepSHAP* (DeepLIFT[60] + Shapley values)

Inconvenients:

- costly to run[61]
- not easy to read if many features

[59] Lundberg et al., *Consistent individualized feature attribution for tree ensembles*, arxiv, 2018
[60] Shrikumar et al., *Learning important features through propagating activation differences*, ICML, 2017
[61] improving the efficiency is work in progress, *e.g.*, Covert and Lee, *Improving KernelSHAP: Practical Shapley Value Estimation via Linear Regression*, AISTATS, 2021