Machine learning

Interpretability, explainability and actionability

Isak Samsten

2022

Interpretability

Machine learning and model interpretation

- Machine learning algorithms are about finding *correlation* in data
- In contrast, interpretable machine learning is about understanding *causality*
- Applying domain knowledge requires understanding of the model and how to interpret its output

Usefulness of model interpretability

- Often, we need to understand individual predictions a model is making

For instance, a model might:

- Recommend a treatment for a patient or estimate disease to be likely. However, the *doctor* needs to understand the reasoning behind the prediction to **trust** it.

Usefulness of model interpretability

- Often, we need to understand individual predictions a model is making

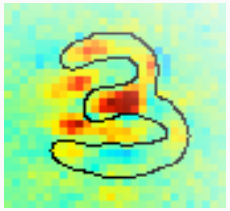
For instance, a model might:

- Recommend a treatment for a patient or estimate disease to be likely. However, the *doctor* needs to understand the reasoning behind the prediction to **trust** it.
- Classify a user as a scammer, but the user disputes it. The *fraud analyst* needs to understand why the model made the classification to **justify** it.

Usefulness of interpretability

- We need to understand differences on a datasets level
  - Why does this product release receive worse feedback than previous updates
  - Why are the grain yields in one region higher in another?
- Model debugging. We might want to understand why a model that worked previously are not working when applied to new data.

Dimensions of interpretability



Prediction-level interpretability

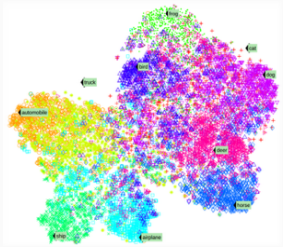
Explain why **x** is predicted as *y* by f(**x**)

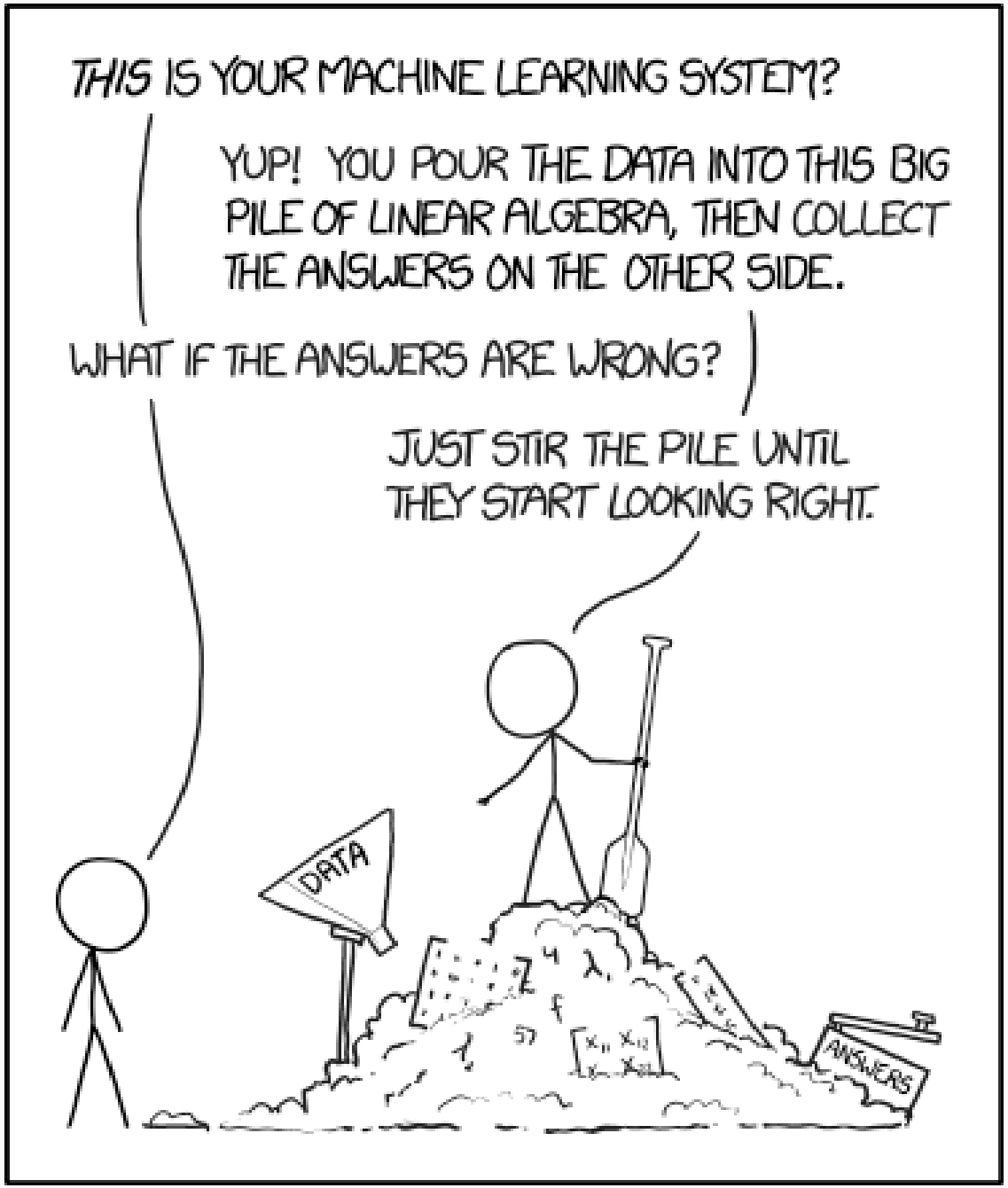
Model-level interpretability

What does a pattern belonging to our model look like?

Data-level interpretability

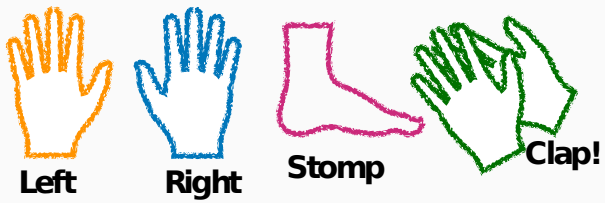
What is the most important dimension of our data?




THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

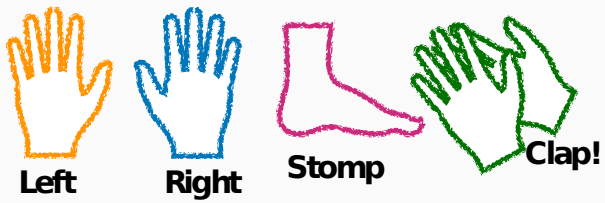
What about decision trees and rules?

Experiment:

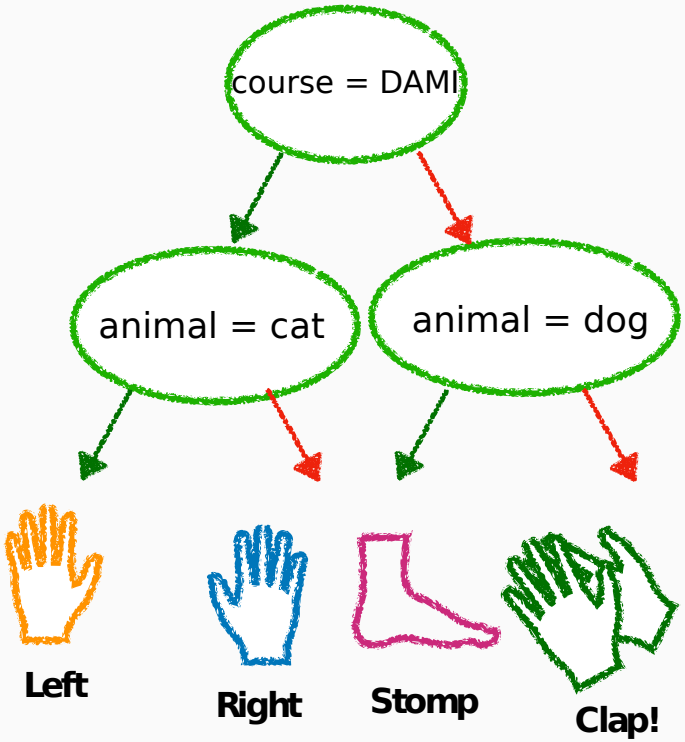


What about decision trees and rules?

Experiment:

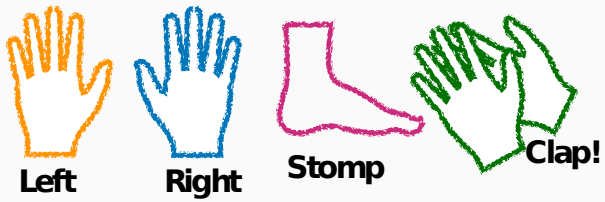


Given, the following decision tree:

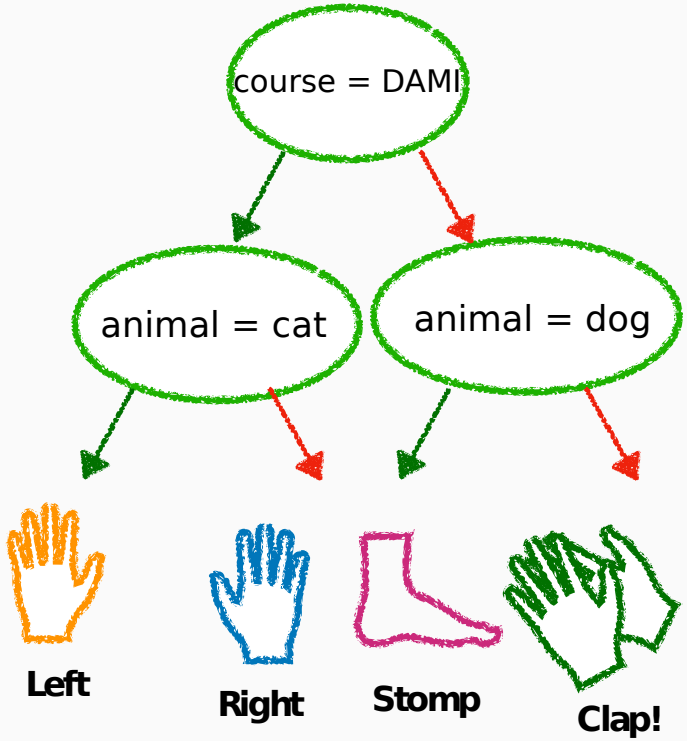


What about decision trees and rules?

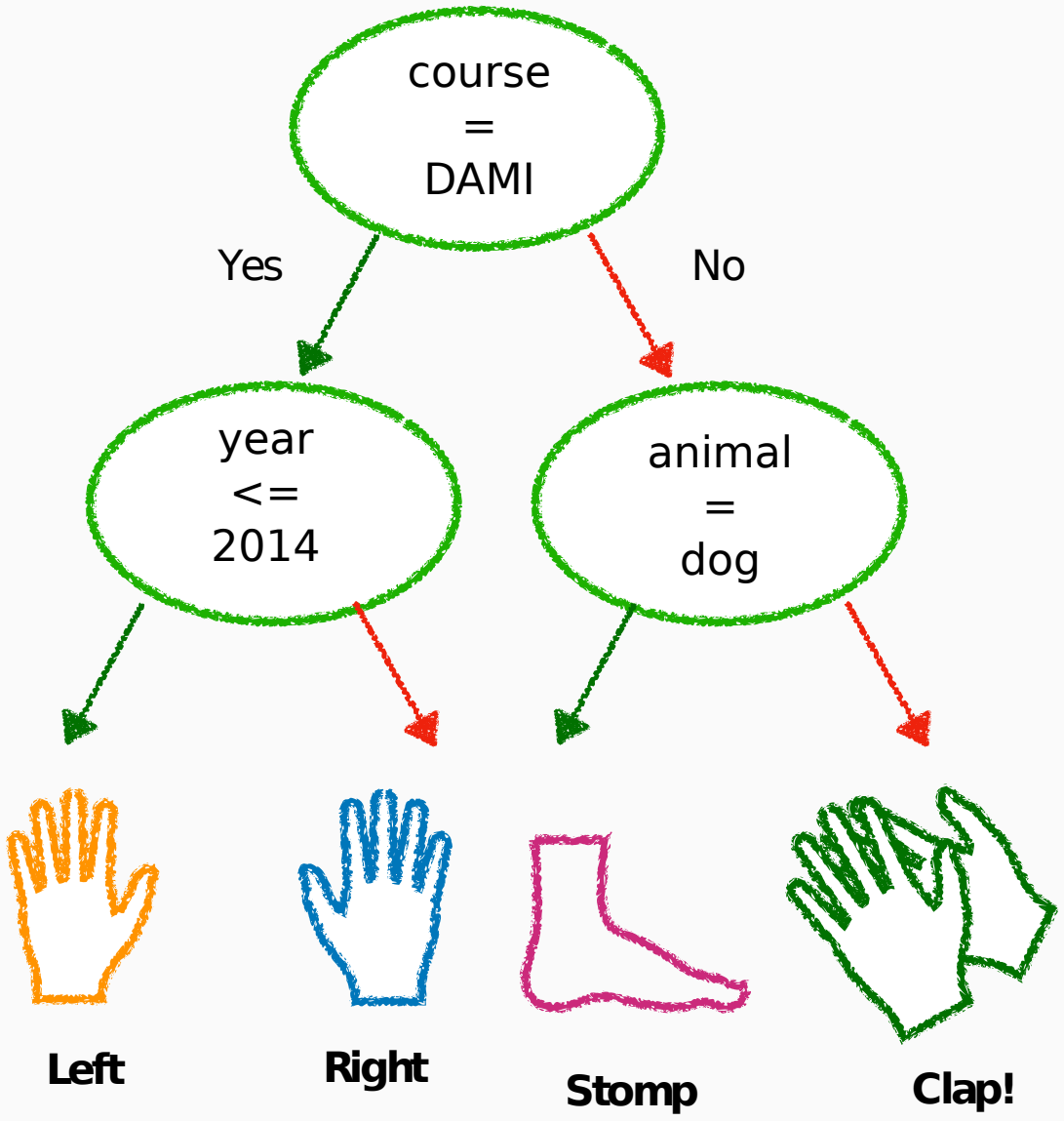
Experiment:



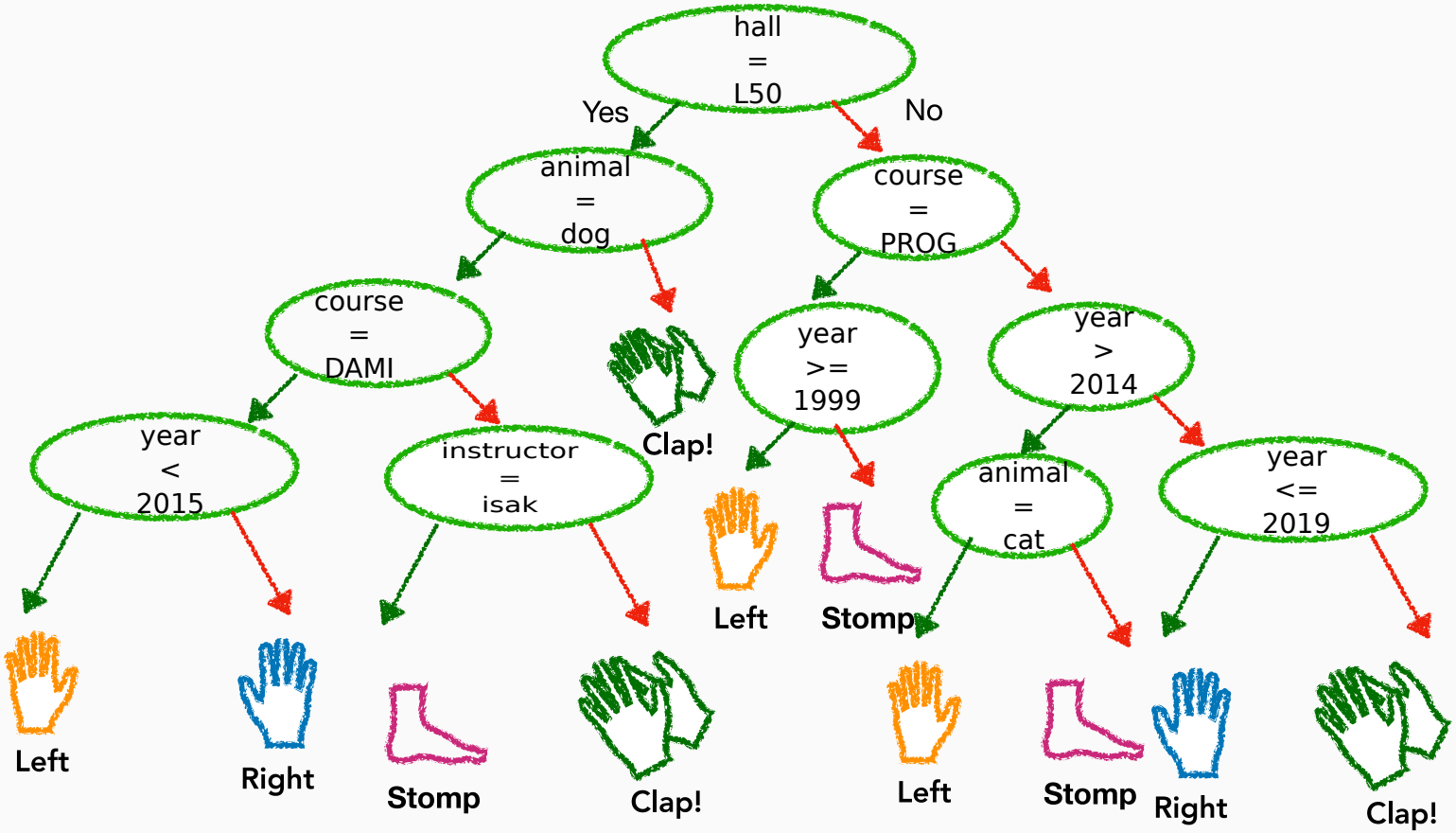
Given, the following decision tree: **predict:** [*DAMI*, *dog*]



Input: [DAMI, 2017, Isak, cat]



Input: [DAMI, 2014, isak, cat, L70]



Decision trees

- Can you explain the overall logic of the system?
- Can you guess which attributes was most important?

Dimensions of interpretability

Data-level interpretability

- Visualization
- Exploratory data analysis

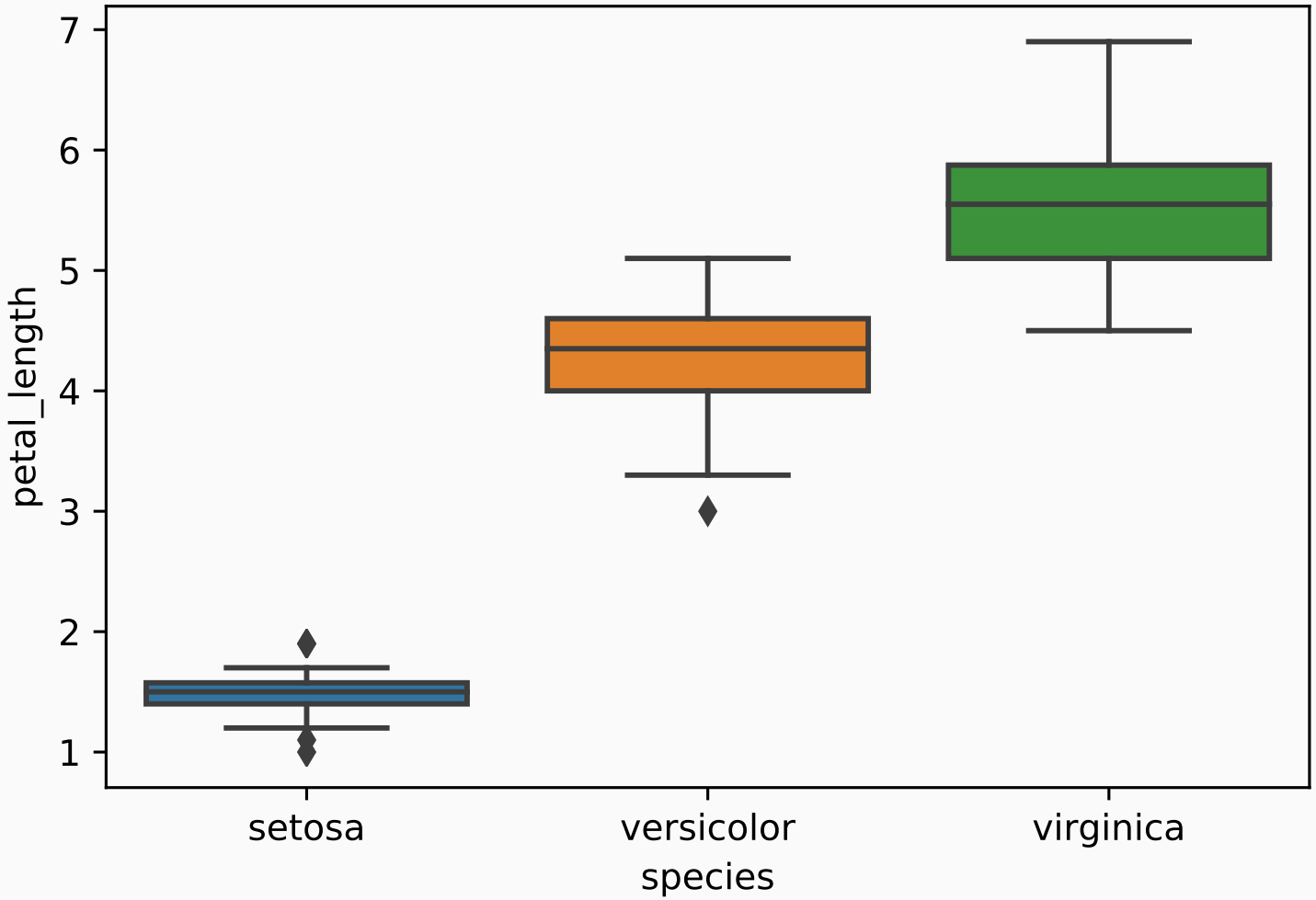
Visualization

- Univariate visualization
- Bivariate visualization
- Multivariate visualization

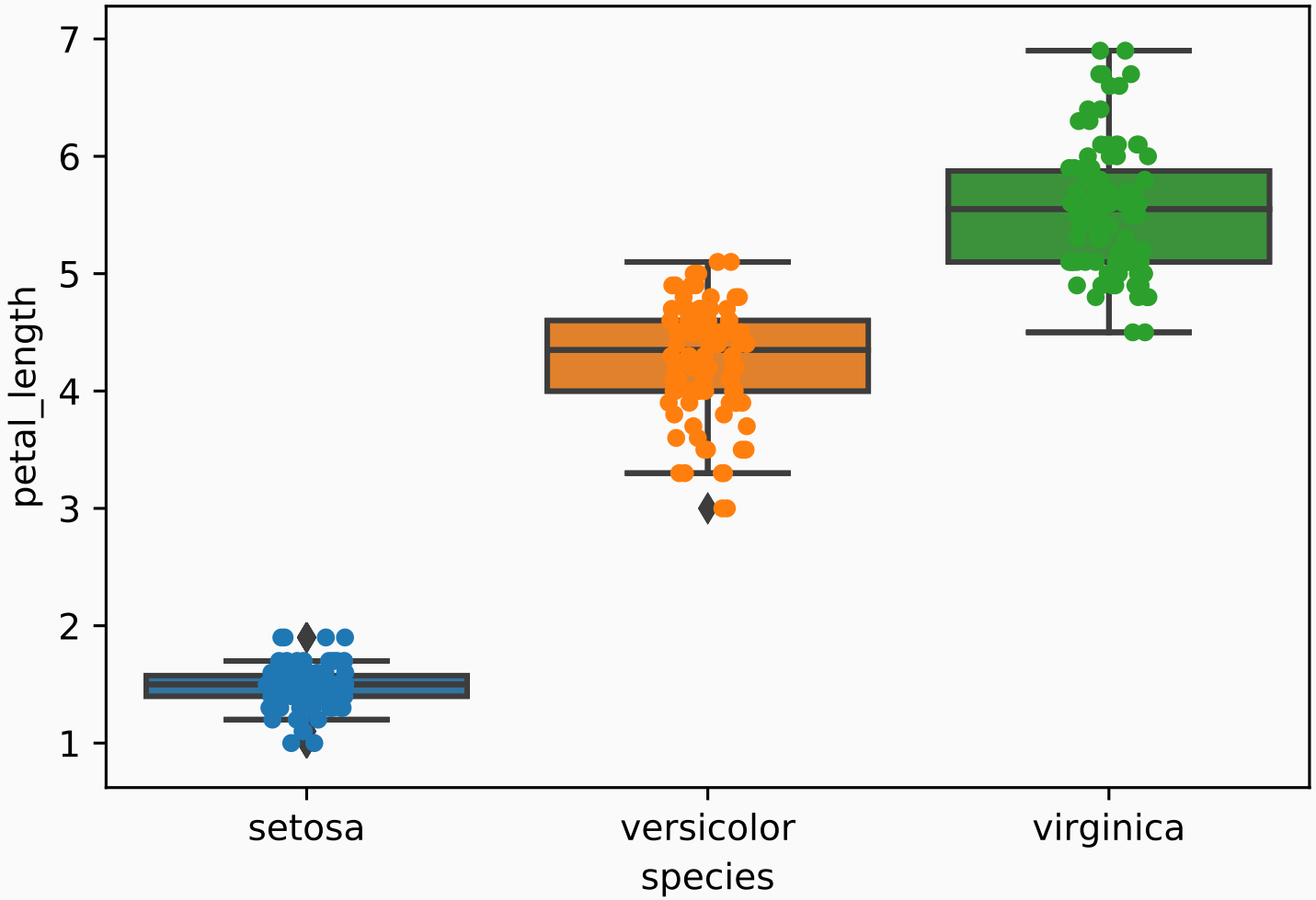
Example data:

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa

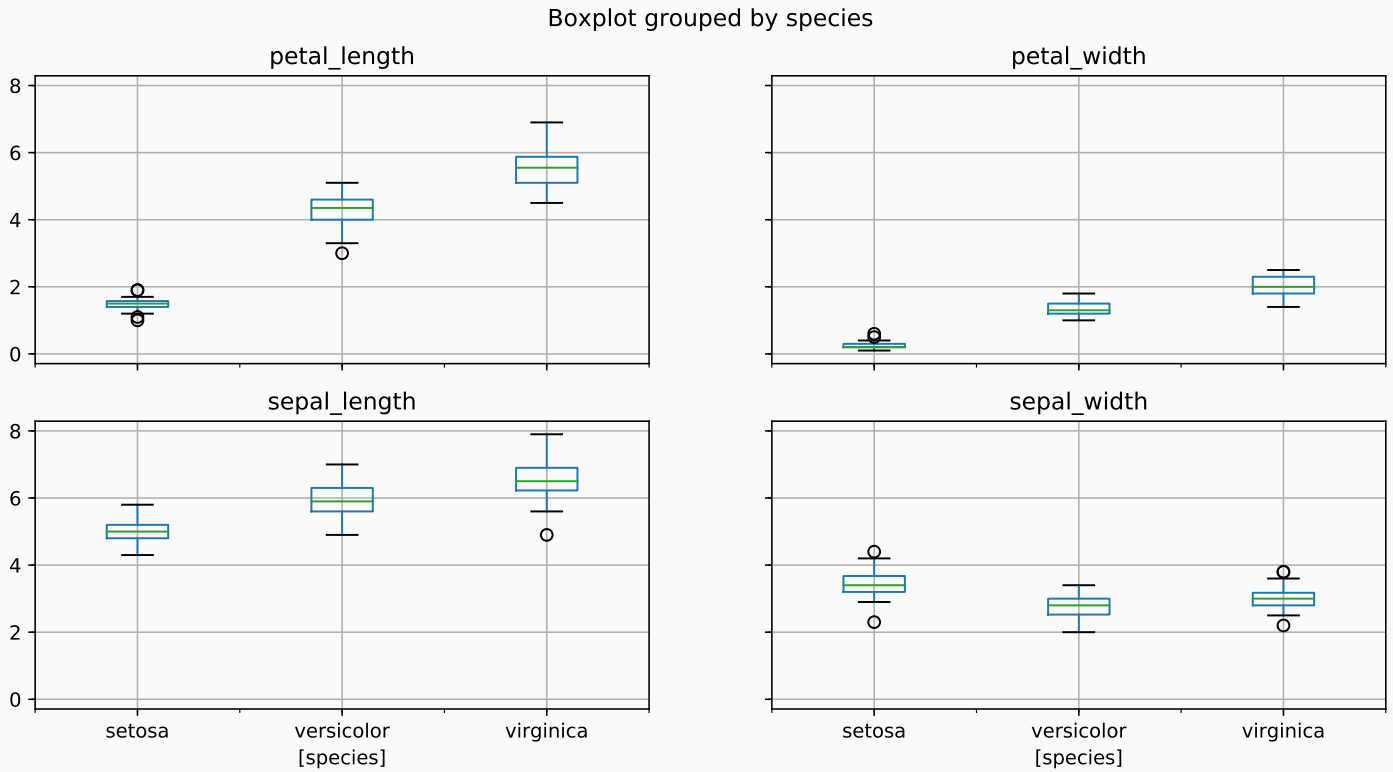
Univariate visualization — box plot



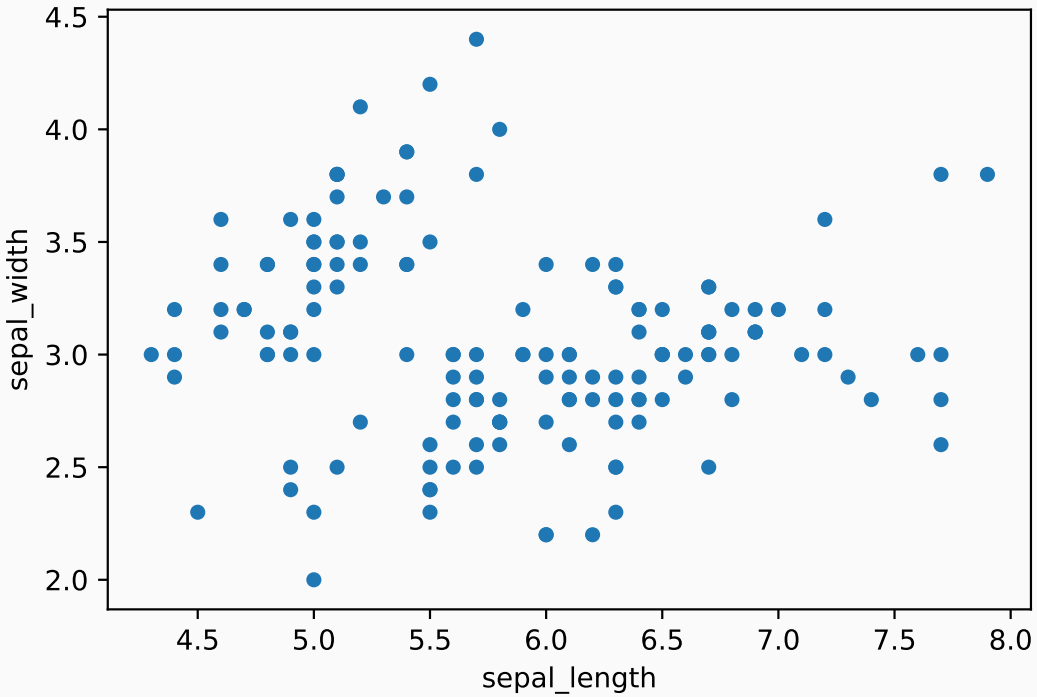
Univariate visualization — box plot



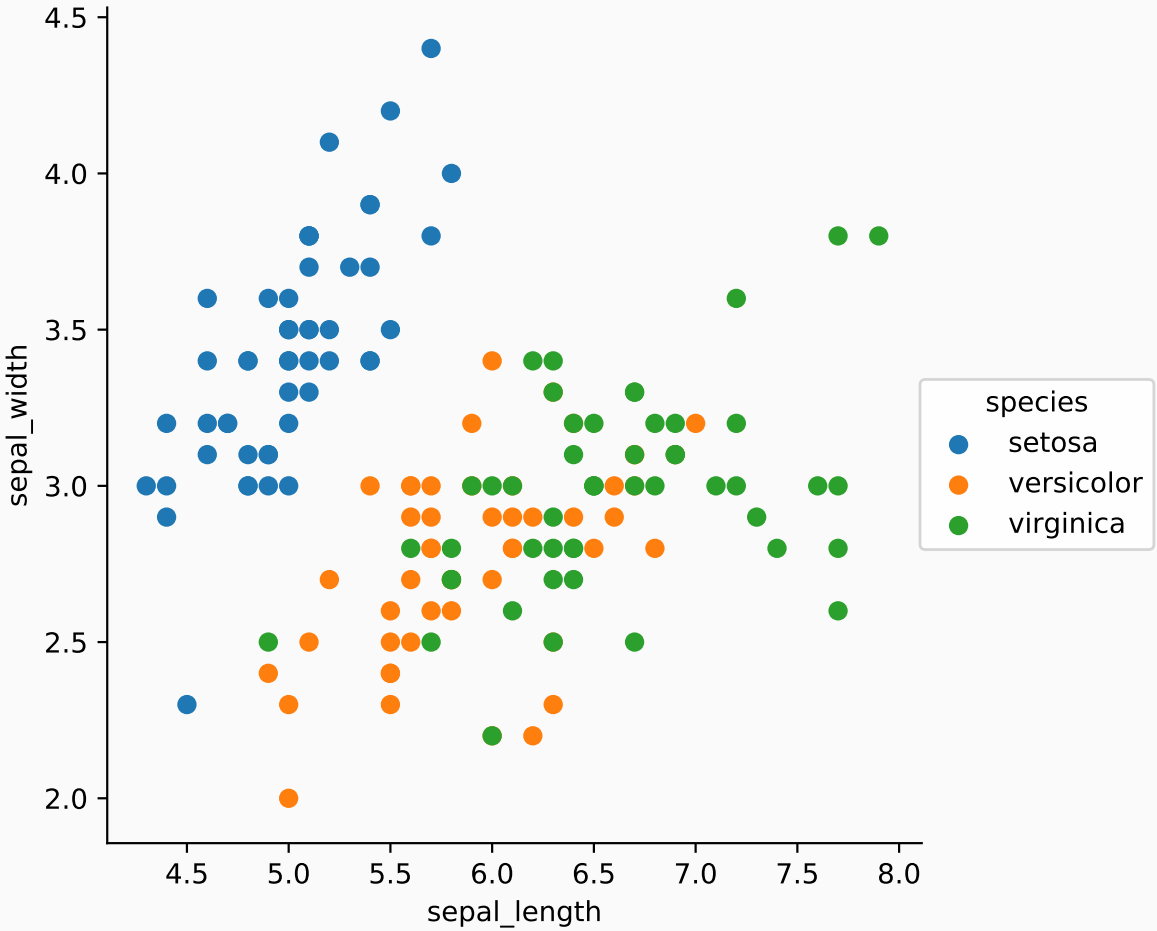
Univariate visualization — multi-box plot



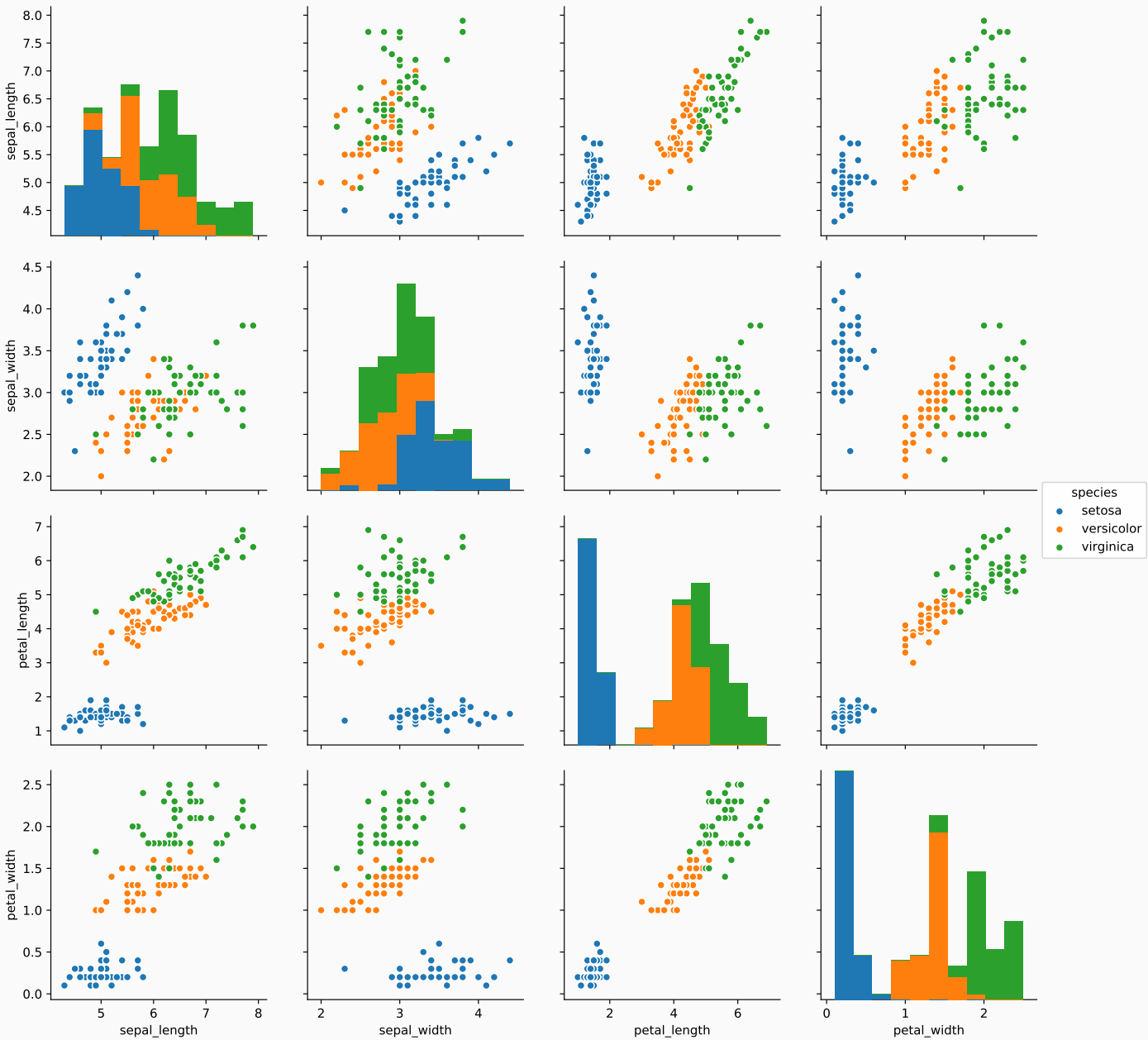
Bivariate visualization



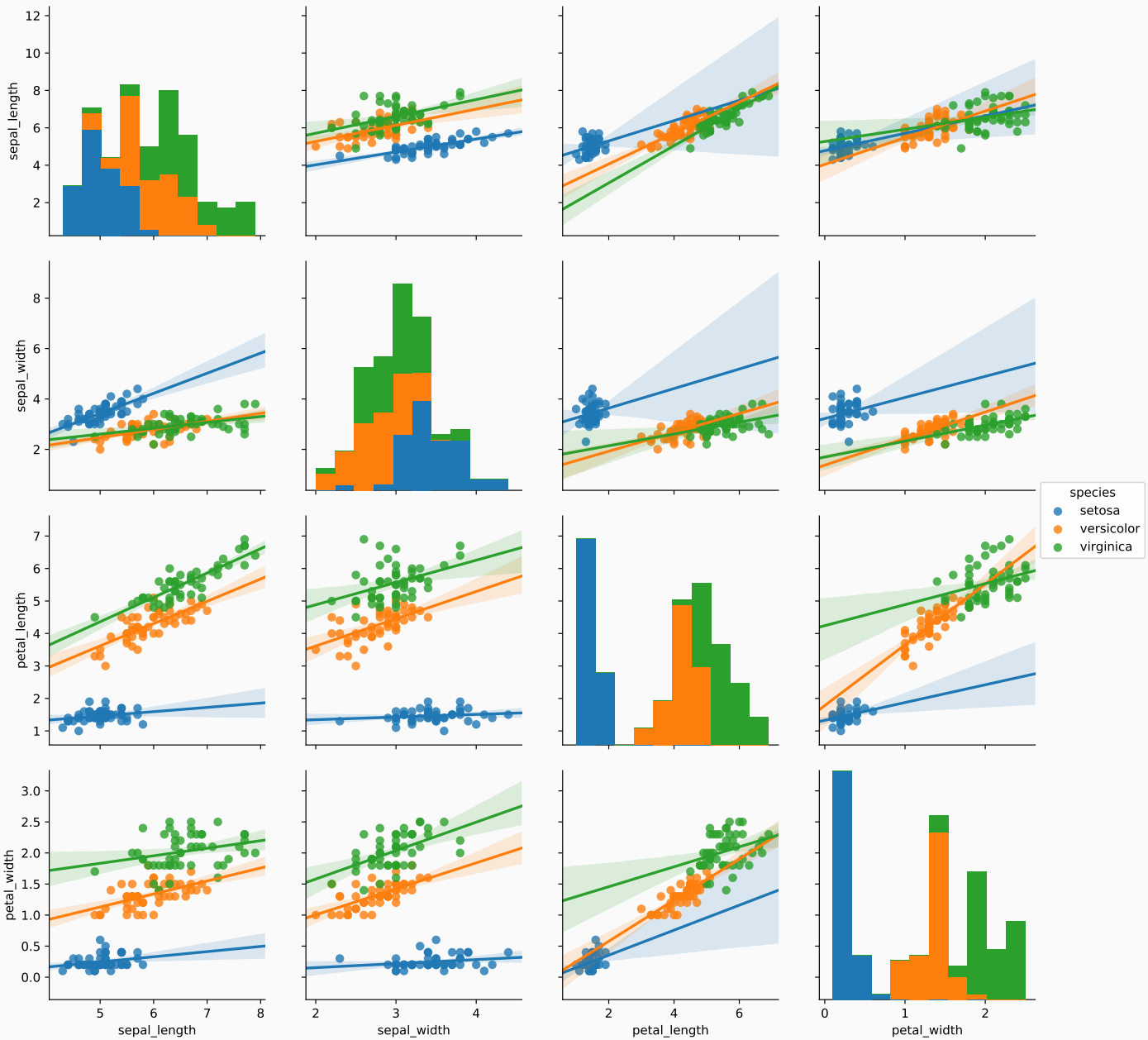
Bivariate visualization



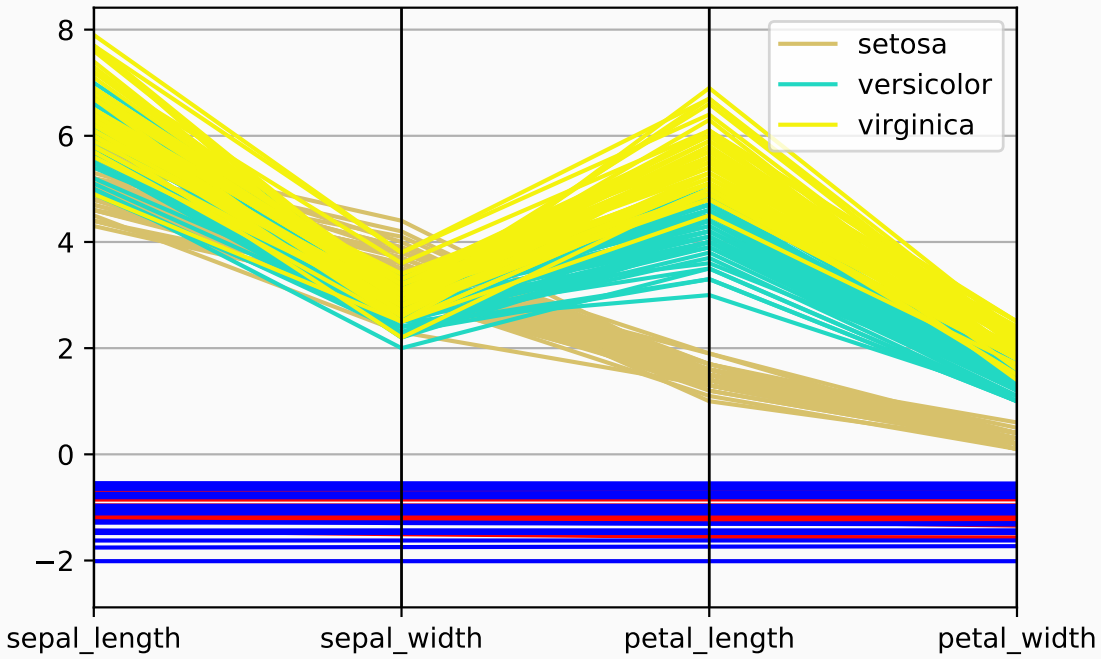
Multivaraite visualization



Multivaraite visualization



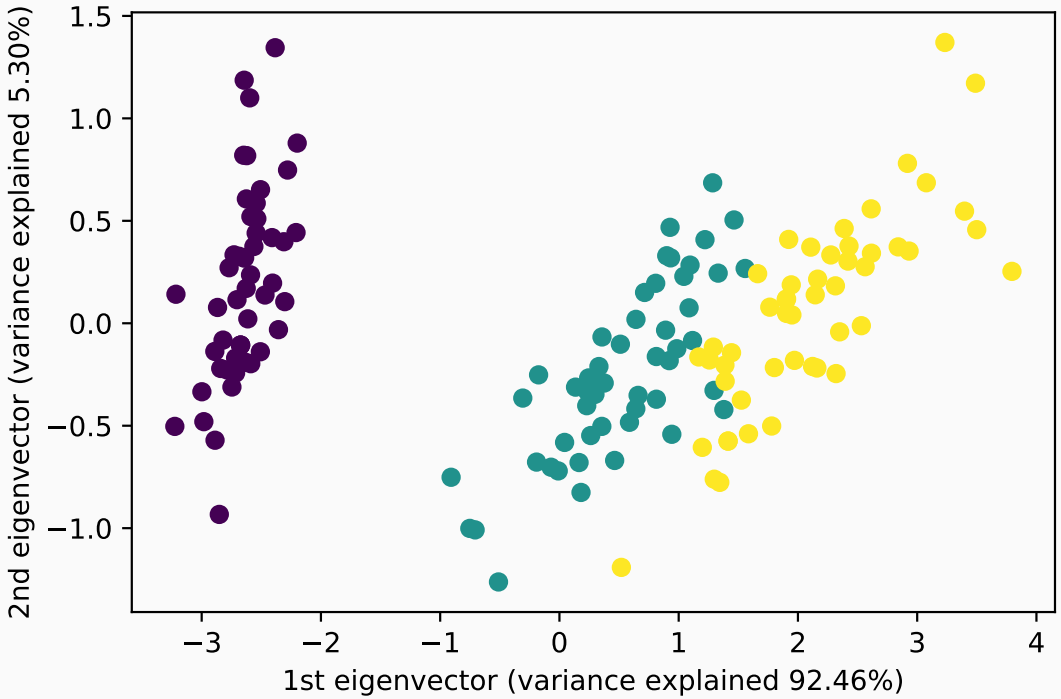
Multivariate visualization — parallel coordinate plot



Multivariate visualization

- **What if we have to many dimensions**
- Say, more than 5 or 6?
- For example, principal component analysis

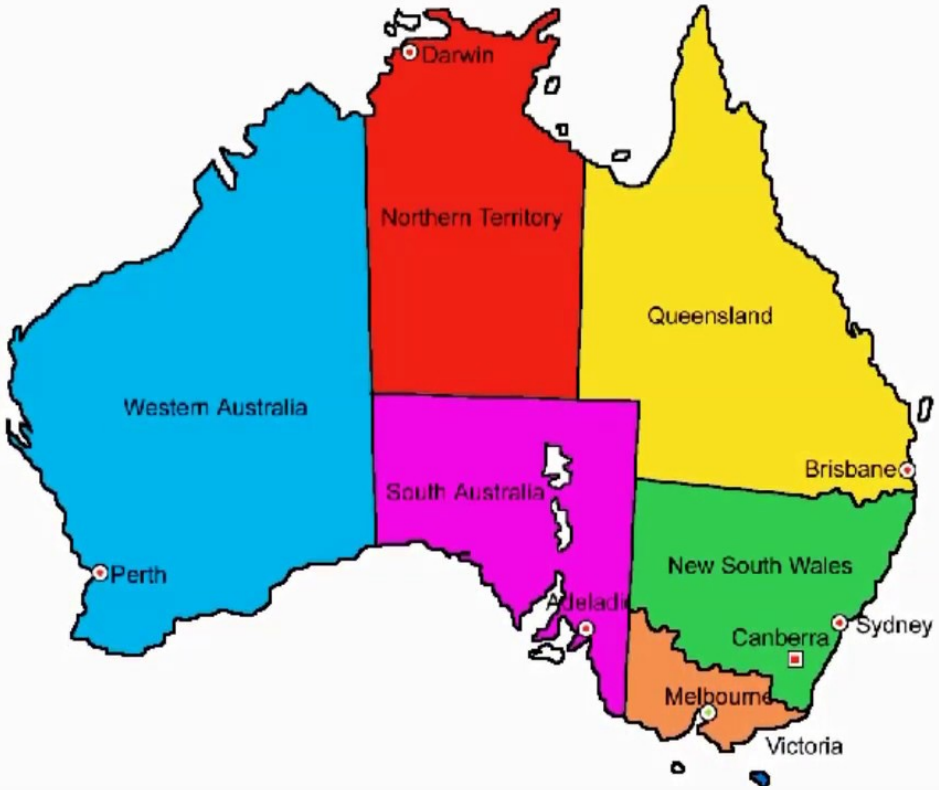
Principal component analysis



- Project the data to a lower dimensionality such that as much as possible of the variance is preserved.

What if we only know the distance between objects?

Distance between cities:



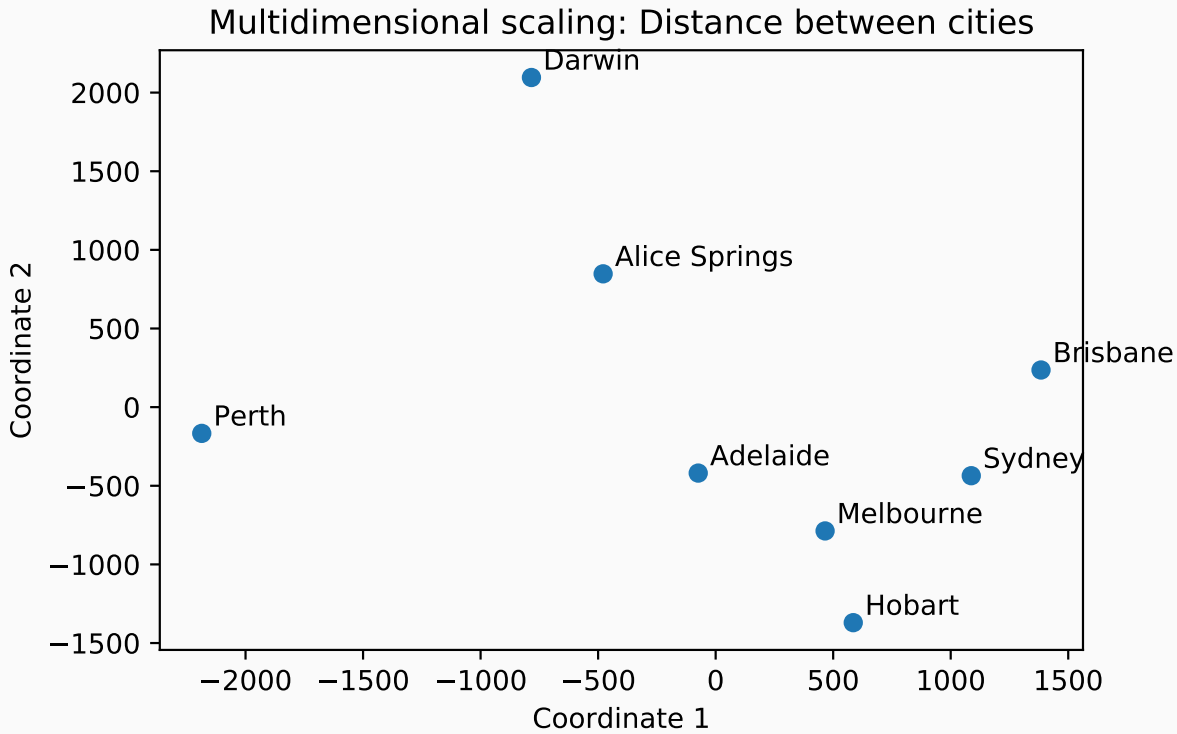
What if we only know the distance between objects?

Distance between cities:

	A	AS	B	D	H	M	P	S
A	0	1328	1600	2616	1161	653	2130	1161
AS	1328	0	1962	1289	2463	1889	1991	2026
B	1600	1962	0	2846	1788	1374	3604	732
D	2616	1289	2846	0	3734	3146	2652	3146
H	1161	2463	1788	3734	0	598	3008	1057
M	653	1889	1374	3146	598	0	2720	713
P	2130	1991	3604	2652	3008	2720	0	3288
S	1161	2026	732	3146	1057	713	3288	0

What if we only know the distance between objects?

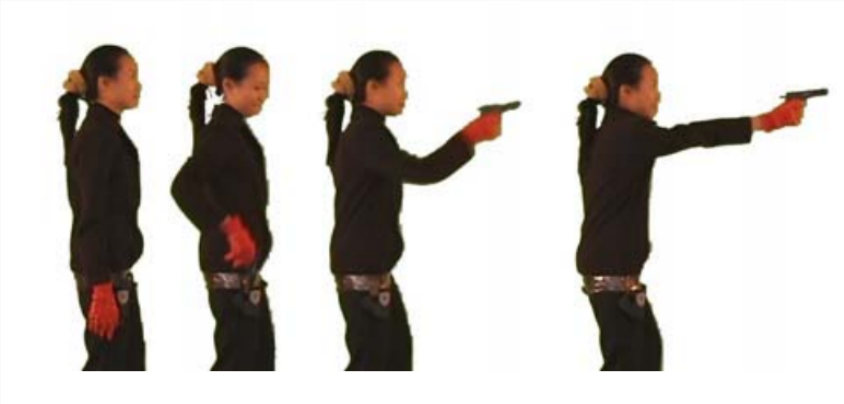
Distance between cities:

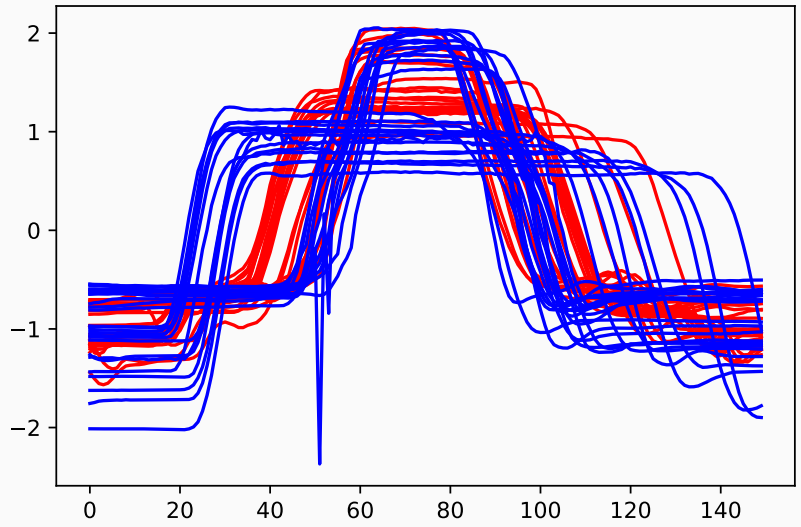


- Multidimensional scaling
- Given a **distance matrix**, construct points in a n-dimensional space, *such that their distances are preserved*

Another example: Time series

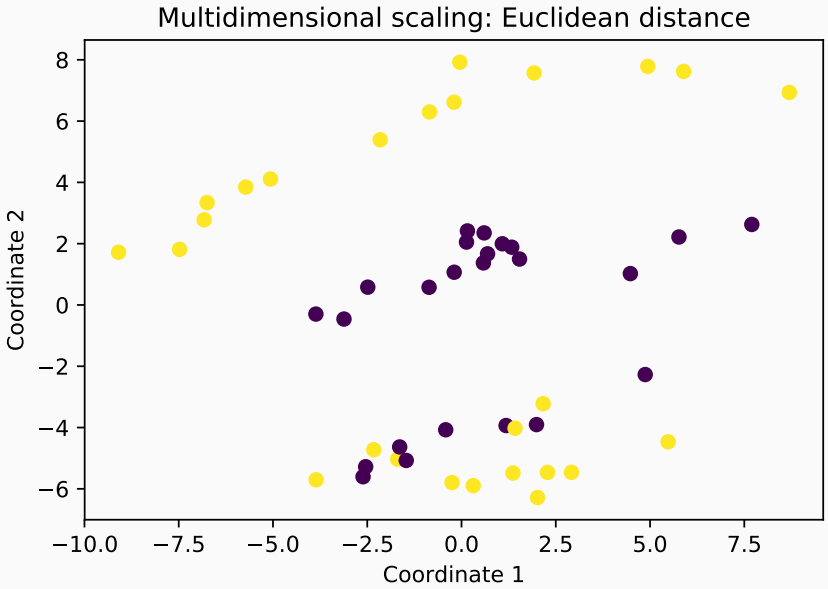
- One measurement that evolves over time
- We could plot them, similar to the parallel coordinate plot





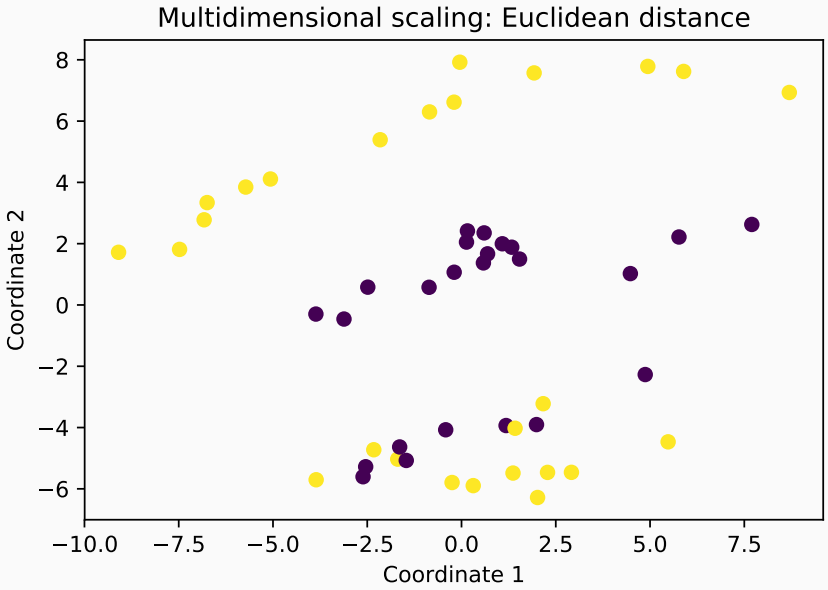
What do we know about the data?

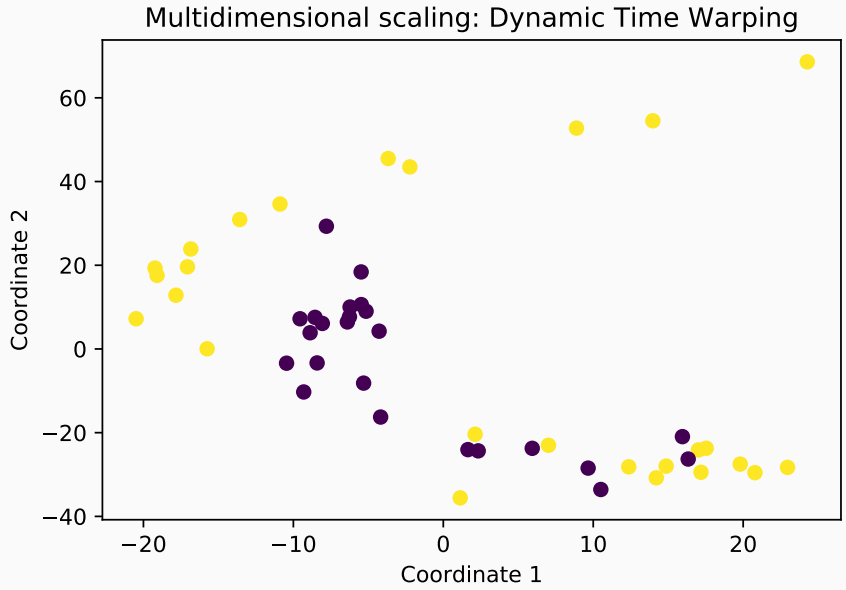
- We can compute the pairwise distance between points!
- For instance, using the (left) Euclidean distance or (right) dynamic time warping



What do we know about the data?

- We can compute the pairwise distance between points!
- For instance, using the (left) Euclidean distance or (right) dynamic time warping





Model-level interpretability

- Traditionally three types of mainstream models: *decision trees*, *rule sets or decision lists* and *linear models*

Model-level interpretability

- Traditionally three types of mainstream models: *decision trees*, *rule sets or decision lists* and *linear models*
- Linear models:

Model-level interpretability

- Traditionally three types of mainstream models: *decision trees*, *rule sets or decision lists* and *linear models*
- Linear models:
  - $y = w_0 + w_1 x_1 + \ldots + w_m x_m$

Model-level interpretability

- Traditionally three types of mainstream models: *decision trees*, *rule sets or decision lists* and *linear models*
- Linear models:
  - $y = w_0 + w_1 x_1 + \ldots + w_m x_m$
  - $\textit{heartdisease} = 0.4 + 0.18 \times \text{smoke} + 0.04 \times \text{age} + 0.84 \times \text{famhist} \ldots$

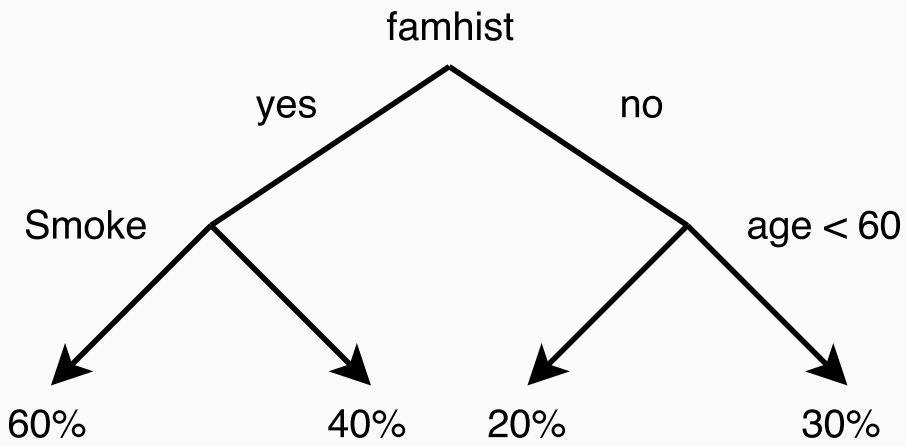
Model-level interpretability

- Traditionally three types of mainstream models: *decision trees*, *rule sets or decision lists* and *linear models*
- Linear models:
  - $y = w_0 + w_1 x_1 + \ldots + w_m x_m$
  - $heartdisease = 0.4 + 0.18 \times \text{smoke} + 0.04 \times \text{age} + 0.84 \times \text{famhist} \ldots$
  - The coefficients in a logistic regression model can be interpreted as the effect of a unit of change in the x_j variable on the predicted probability with the other variables in the model held constant

Model-level interpretability

- Traditionally three types of mainstream models: *decision trees*, *rule sets or decision lists* and *linear models*
- Linear models:
  - $y = w_0 + w_1 x_1 + \ldots + w_m x_m$
  - $\textit{heartdisease} = 0.4 + 0.18 \times \text{smoke} + 0.04 \times \text{age} + 0.84 \times \text{famhist} \ldots$
  - The coefficients in a logistic regression model can be interpreted as the effect of a unit of change in the x_j variable on the predicted probability with the other variables in the model held constant
  - For example: given the above model, the effect of 1-unit increase in smoking $e^{0.18} = 1.197$, increases the risk (odds) of heart disease by $\approx 20\%$

Decision tree



Permutation importance

- We measure the importance of a feature by calculating the increase in the model's prediction error after permuting the feature
- A feature is "important" if shuffling its values increases the model error, because in this case the model relied on the feature for the prediction.
- A feature is "unimportant" if shuffling its values leaves the model error unchanged, because in this case the model ignored the feature for the prediction.

Algorithm

- Estimate original model error $e_{orig} = \mathcal{L}(y, \hat{f}(X))$

Algorithm

- Estimate original model error $e_{orig} = \mathcal{L}(y, \hat{f}(X))$
- For each attribute $j \in \{1, \ldots, p\}$

Algorithm

- Estimate original model error $e_{orig} = \mathcal{L}(y, \hat{f}(X))$
- For each attribute $j \in \{1, \ldots, p\}$
    - Generate feature matrix X_{perm} by randomly permuting values of that feature. This breaks the association between the feature and the true outcome in y.

Algorithm

- Estimate original model error $e_{orig} = \mathcal{L}(y, \hat{f}(X))$
- For each attribute $j \in \{1, \ldots, p\}$
  - Generate feature matrix X_{perm} by randomly permuting values of that feature. This breaks the association between the feature and the true outcome in y.
  - Estimate error $e_{perm} = \mathcal{L}(y, \hat{f}(X_{perm}))$ based on the predictions of the permuted data.

Algorithm

- Estimate original model error $e_{orig} = \mathcal{L}(y, \hat{f}(X))$
- For each attribute $j \in \{1, \ldots, p\}$
  - Generate feature matrix X_{perm} by randomly permuting values of that feature. This breaks the association between the feature and the true outcome in y.
  - Estimate error $e_{perm} = \mathcal{L}(y, \hat{f}(X_{perm}))$ based on the predictions of the permuted data.
  - Calculate feature importance as $I_j = \frac{e_{perm}}{e_{orig}}$ or $I_j = e_{orig} - e_{perm}$.

Algorithm

- Estimate original model error $e_{orig} = \mathcal{L}(y, \hat{f}(X))$
- For each attribute $j \in \{1, \ldots, p\}$
  - Generate feature matrix X_{perm} by randomly permuting values of that feature. This breaks the association between the feature and the true outcome in y.
  - Estimate error $e_{perm} = \mathcal{L}(y, \hat{f}(X_{perm}))$ based on the predictions of the permuted data.
  - Calculate feature importance as $I_j = \frac{e_{perm}}{e_{orig}}$ or $I_j = e_{orig} - e_{perm}$.
- Sort features by descending I

Drawbacks of permutation importance

- Or attributes that are only important in combination with another attribute
- How about attributes that are masked by other attributes?

[1] A peek into the black box: exploring classifiers by randomization

Drawbacks of permutation importance

- Or attributes that are only important in combination with another attribute
- How about attributes that are masked by other attributes?

- Identify interacting attributes are hard
- One solution proposed by Henelius et. al. (2014)[1], is "an efficient iterative algorithm to find the attributes and dependencies used by any classifier when making predictions."

    - How does different randomization strategies affect the error a classifier make.
    - Intuition: if an attribute B is randomized conditional to an attribute A, will the model make the same errors?

[1] A peek into the black box: exploring classifiers by randomization

Feature importance on training or testing data?

- Argument for test data:
  - Permutation importance rely on performance estimates
  - Training performance estimates are garbage
  - Training importances are garbage
  - Gives us an estimate on how important the feature is for the models performance
- Argument for training
  - We want to know what dependencies in the data the model rely on

Surrogate models

- A global surrogate model is an interpretable model that is trained to approximate the predictions of a black box model
- We can draw conclusions about the black box model by interpreting the surrogate model.

There is actually not much theory needed to understand surrogate models. We want to approximate our black box prediction function f as closely as possible with the surrogate model prediction function g, under the constraint that g is interpretable

Algorithm

Training a surrogate model is a model-agnostic method, since it does not require any information about the inner workings of the black box model, only access to data and the prediction function is necessary.

- Select a dataset X. This can be the same dataset that was used for training the black box model or a new dataset from the same distribution.
- Get the predictions of the black box model f over X.
- Select an interpretable model type
- Train the interpretable model on the dataset X and f-predictions. That is, $g = G(X, f(X))$
- Measure how well the surrogate model replicates the predictions of the black box model.

How to measure the surrogates replication?

We can use the R-square measure:

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{i=1}^{n} \left(g(x_i) - f(x_i)\right)^2}{\sum_{i=1}^{n} \left(g(x_i) - \mu(f(X))\right)^2} \tag{1}$$

- SSE: Sum of square error
- SST: Sum of square total

If R-squared is close to 1 (= low SSE), then the interpretable model approximates the behavior of the black box model very well. If the interpretable model is very close, you might want to replace the complex model with the interpretable model. If the R-squared is close to 0 (= high SSE), then the interpretable model fails to explain the black box model.

Prediction-level interpretability

- Many of the state of the art machine learning models are functionally black boxes, as it is nearly impossible to get a feeling for its inner workings.

Prediction-level interpretability

- Many of the state of the art machine learning models are functionally black boxes, as it is nearly impossible to get a feeling for its inner workings.
- This brings us to a question of trust:

Prediction-level interpretability

- Many of the state of the art machine learning models are functionally black boxes, as it is nearly impossible to get a feeling for its inner workings.
- This brings us to a question of trust:
  - do I trust that a certain prediction from the model is correct?

Prediction-level interpretability

- Many of the state of the art machine learning models are functionally black boxes, as it is nearly impossible to get a feeling for its inner workings.
- This brings us to a question of trust:
  - do I trust that a certain prediction from the model is correct?
  - or do I even trust that the model is making reasonable predictions in general?

Prediction-level interpretability

- Many of the state of the art machine learning models are functionally black boxes, as it is nearly impossible to get a feeling for its inner workings.
- This brings us to a question of trust:
  - do I trust that a certain prediction from the model is correct?
  - or do I even trust that the model is making reasonable predictions in general?
- It seems intuitive that explaining the rationale behind individual predictions would make us better positioned to trust or mistrust the prediction, or the classifier as a whole.

Prediction-level interpretability

- Many of the state of the art machine learning models are functionally black boxes, as it is nearly impossible to get a feeling for its inner workings.
- This brings us to a question of trust:
  - do I trust that a certain prediction from the model is correct?
  - or do I even trust that the model is making reasonable predictions in general?
- It seems intuitive that explaining the rationale behind individual predictions would make us better positioned to trust or mistrust the prediction, or the classifier as a whole.
- Even if we can't necessarily understand how the model behaves on all cases, it may be possible to understand how it behaves in particular cases.

Interpretability vs. explainability

- Explainabilitity refers to a models ability to explain its predictions (in a language that users can understand)
- Actionability refers to a models ability to recommend actions to a user, such that a desired outcome can be achieved or an undesired outcome be avoided.
- Both actionability and explainability requires models with high accuracy
    - Bad model = poor explanation
    - Bad model = poor recommendation

Local interpretable model-agnostic explanations

Local Interpretable Model Agnostic Explanations (LIME)[2], a technique to explain the predictions of any machine learning classifier.

- **Local** refers to local fidelity — i.e., the explanation should really reflect the behavior of the classifier *around* the instance being predicted

[2] "Why Should I Trust You?": Explaining the Predictions of Any Classifier

Local interpretable model-agnostic explanations

Local Interpretable Model Agnostic Explanations (LIME)[2], a technique to explain the predictions of any machine learning classifier.

- **Local** refers to local fidelity — i.e., the explanation should really reflect the behavior of the classifier *around* the instance being predicted
- **Interpretable** explanation in terms of interpretable representations, so human can make sense of it

[2] "Why Should I Trust You?": Explaining the Predictions of Any Classifier

Local interpretable model-agnostic explanations

Local Interpretable Model Agnostic Explanations (LIME)[2], a technique to explain the predictions of any machine learning classifier.

- **Local** refers to local fidelity — i.e., the explanation should really reflect the behavior of the classifier *around* the instance being predicted
- **Interpretable** explanation in terms of interpretable representations, so human can make sense of it
- It is able to explain any model without needing to "peak" into it, so it is **model-agnostic** (in fact, it works with any model able to output prediction probabilities)

[2] "Why Should I Trust You?": Explaining the Predictions of Any Classifier

What is an explanation?

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception network, highlighting positive pixels. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$)**, "Acoustic guitar"** ($p = 0.24$) **and "Labrador"** ($p = 0.21$)

What constitues a good explanation?

- Interpretable: humans can understand
- Faithful: describes the inner working of the model
- According to philosophy: when someone no longer need to ask *why*

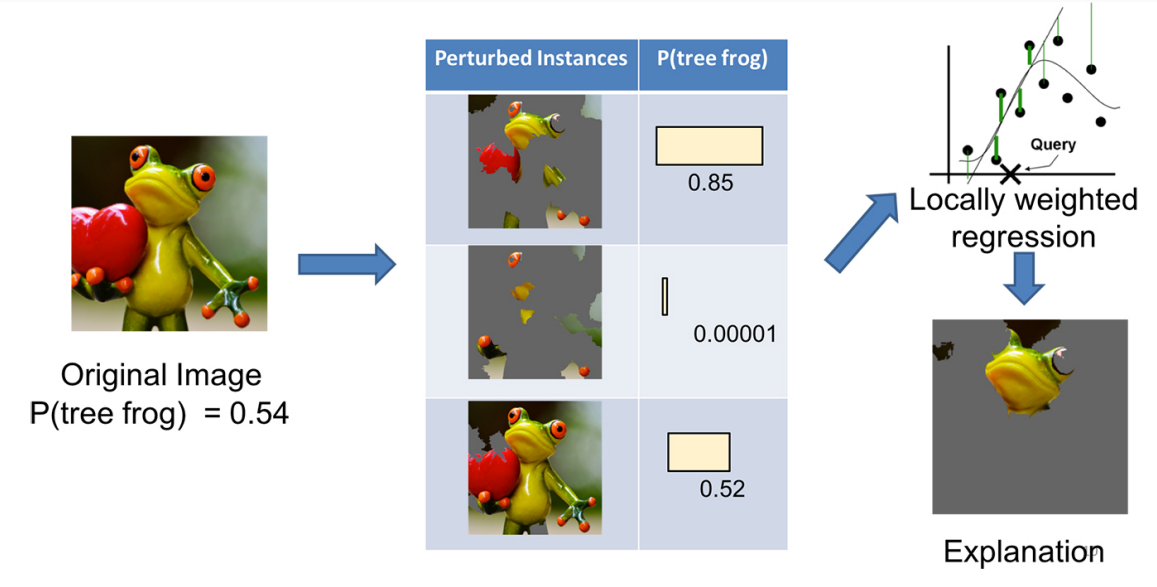
Example: image classification

Original Image

Interpretable Components

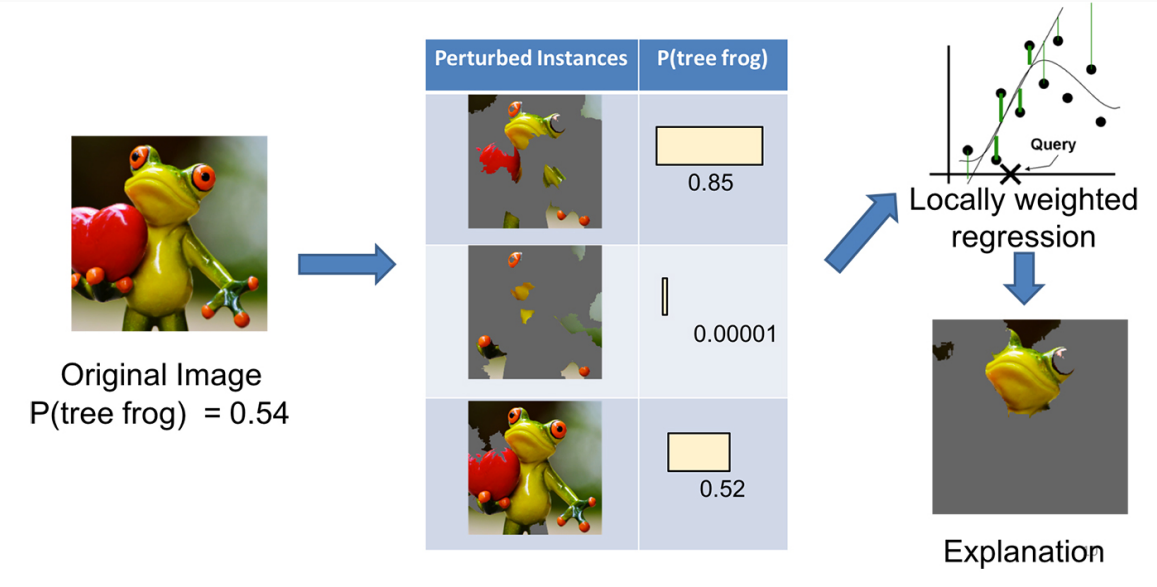
- Imagine that we want to explain a classifier that predicts how likely it is for an image to contain a frog.
- Take the image on the left and divide it into "interpretable" components (contiguous superpixels)

Example: image classification



- Generate a data set of perturbed instances by turning some of the interpretable components "off"
- For each perturbed instance, use the model to compute the probability that a the image contains a frog

Example: image classification



- Finally, train a linear model on this data set, which is locally weighted, i.e., care more about making mistakes in perturbed instances that are more similar to the original image.
- ... and present the superpixels with highest positive weights as an explanations (graying out everything else)

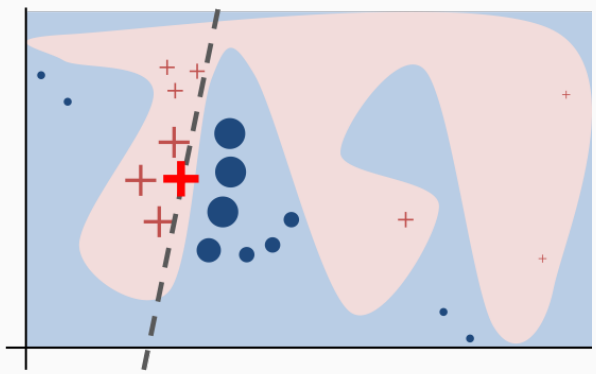
Example: image classification

P() = 0.54

P() = 0.07

P() = 0.05

LIME visualized

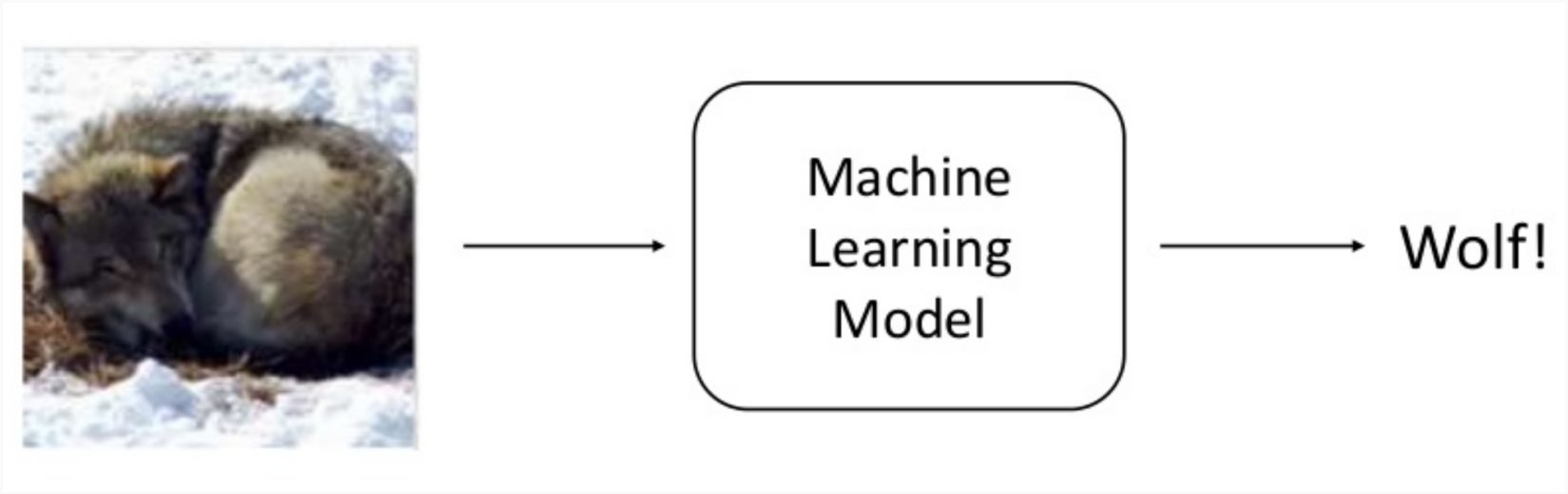


$P(y)$	x^1	...	x^k
0.3	0	...	1
0.5	0	...	0
0.9	1	...	0

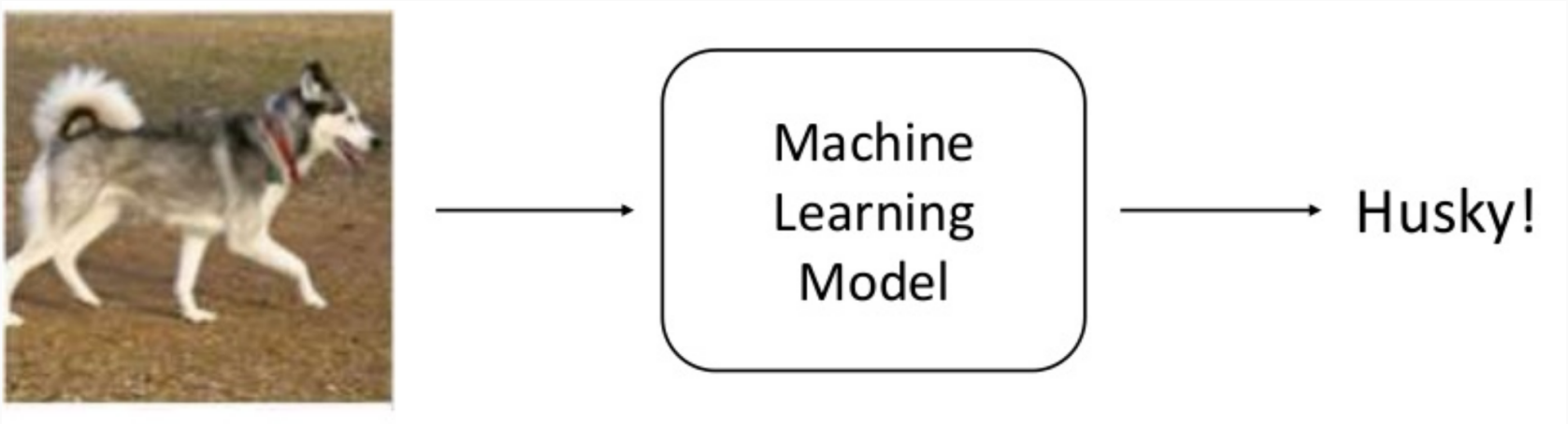
The bright red cross is the instance being explained (let's call it **x**):

- generate multiple instances of **x** with perturbations
- weight the examples according to proximity to **x**
- predict the output probability for label y for the
- the resulting matrix corresponds to what perturbations are "active" and the output is the probability of predicting y
- learn a linear model that approximates the model well in the vicinity of **x**, using the new representation

Human-level evaluation

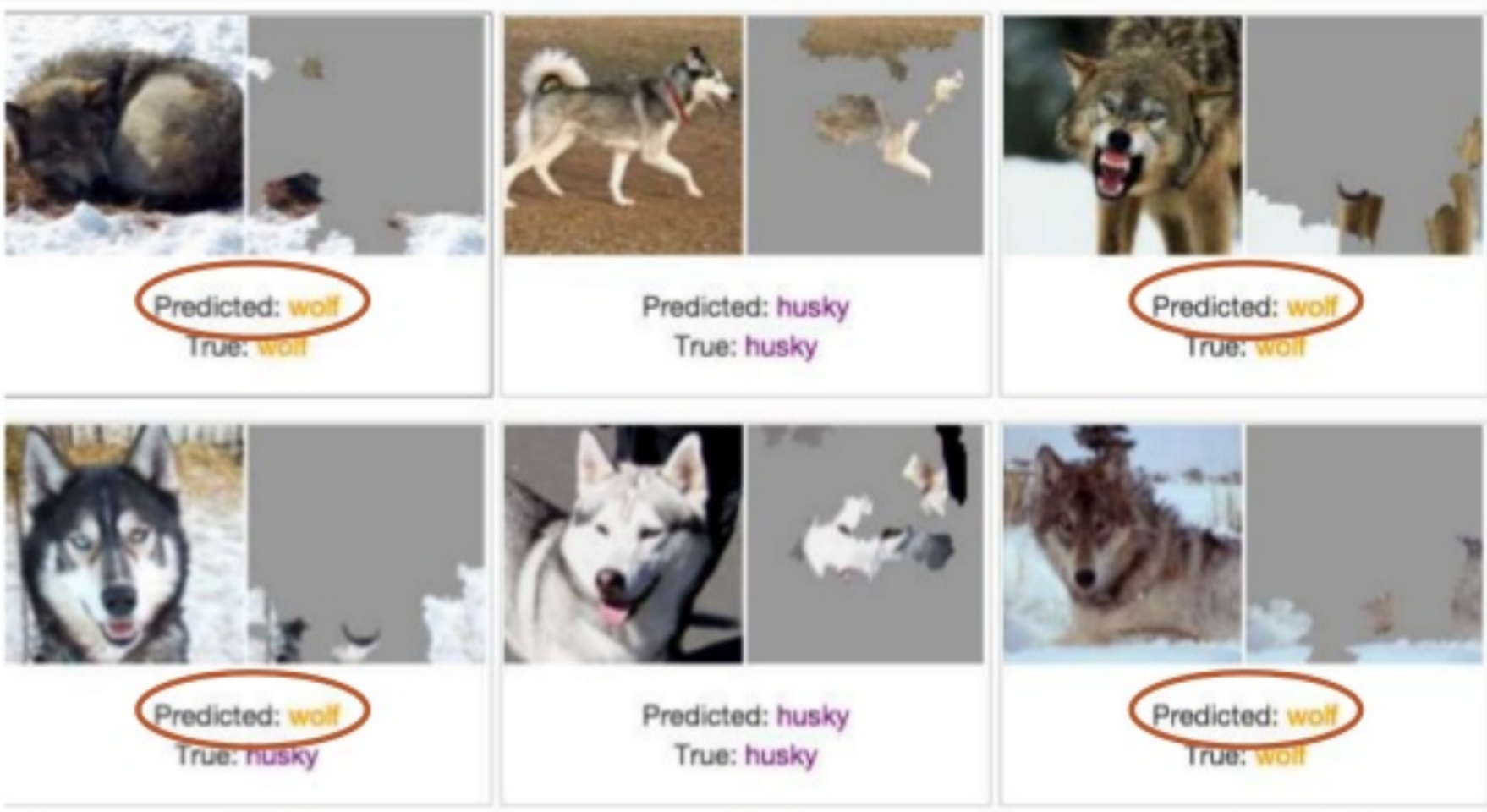


Human-level evaluation



Human-level evaluation

Human-level evaluation



Human-level evaluation

- **Before** 60% did not trust the model
- **After** 90% did not trust the model

Human-level evaluation

- **Before** 60% did not trust the model
- **After** 90% did not trust the model

however:

- **Before** 40% found the model to provide insight
- **After** 92% found the model to provide insight

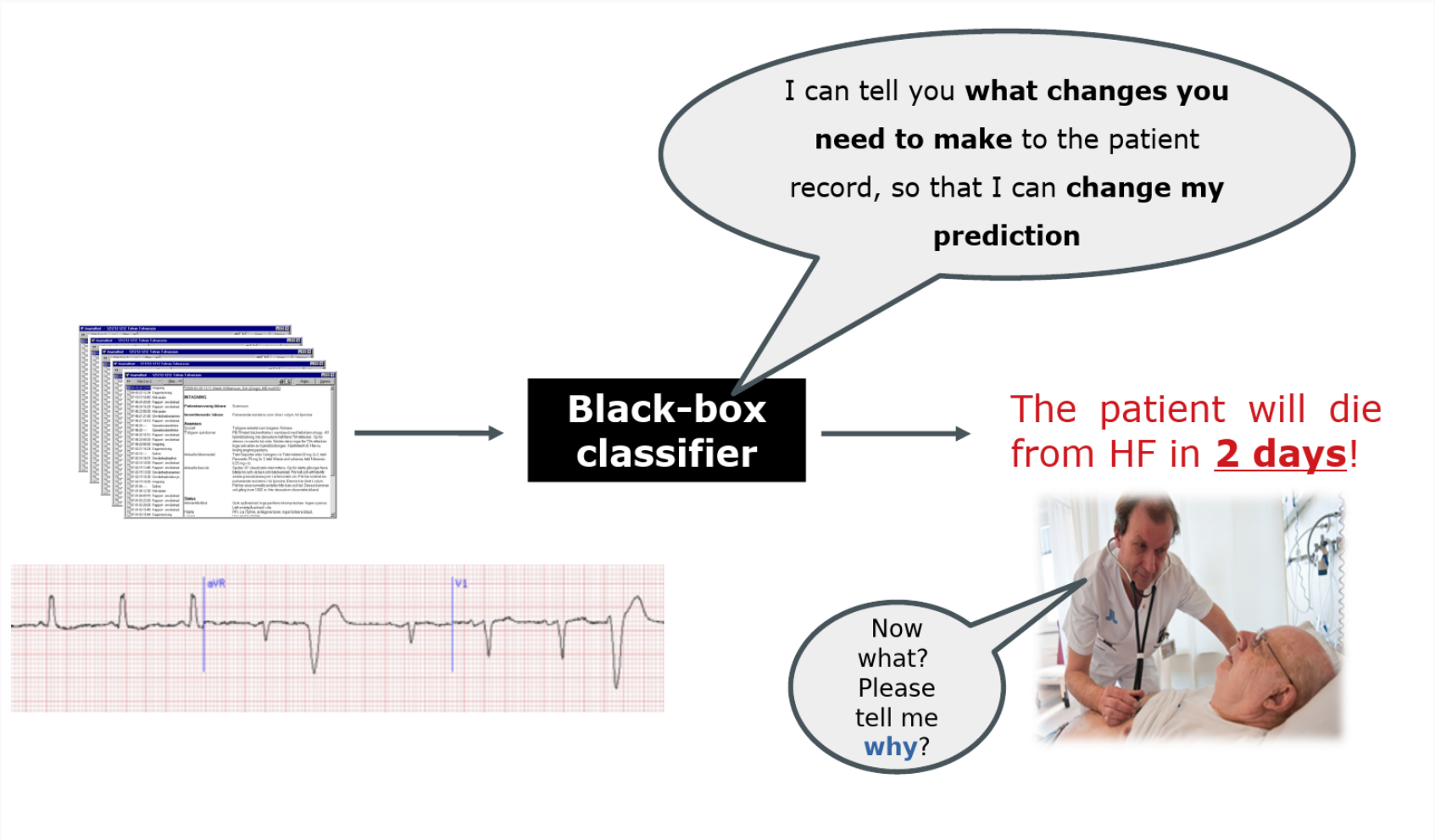
Actionability and counterfactual explanations

Actionability and attribute modifications

- In many applications, practitioners are happy with black-box models which sacrifice interpretability for predictive performance
- However, under certain circumstances practitioners are interested in how to **transform a predicted negative instance to a positive instance**[3]
- Gain to gain **actionable** insights on how to change an undesired outcome.

[3] Tolomei, Gabriele, et al. *Interpretable predictions of tree-based ensembles via actionable feature tweaking.*, 2017

Explainable and actionable



Why actionability?

- Given a model for predicting whether or not a patient is likely to develop a disease:
- understand what actions should be taken to transition a negative outcome prediction to a positive outcome
- For example, from the undesired outcome **patient is likely to develop X** to desired outcome **patient is unlikely not develop X**
- Also, what *changes should I perform in order to avoid the undesired outcome!*

Counterfactual explanations: problem formulation

- Given a n-dimensional vector space $\mathcal{X} \in \mathbf{R}^n$ with instances $x \in \mathcal{X}$ labeled as either $\{+, -\} \in \mathcal{Y}$.
- Given an unknown target function $f : \mathcal{X} \to \mathcal{Y}$ and a already trained approximation $\hat{f}$, where $\hat{f} \sim f$ (i.e., a good predictor).
- Specifically, here $\hat{f}$ is an ensemble of (binary) decision trees: $\hat{f} = \phi(\hat{h}_1, \ldots, \hat{h}_K)$, where each $\hat{h}$ is a base estimator (tree) and ϕ is an ensembling operator (e.g., majority voting).

Problem forumlation cont.

- The goal of finding a transition function is to (efficiently) identify a *transformation* that transitions a negatively predicted instance into positively predicted instance.
- More formally, the task is to transition the original instance x into a new instance x', $x \rightarrow x'$, such that $\hat{f}(x) = -$ and $\hat{f}(x') = +$.
- Typically a more well-formed version of this problem is investigated where the goal is to choose the transformed instance, such that a cost function, δ, is minimized.
- Formalized as:

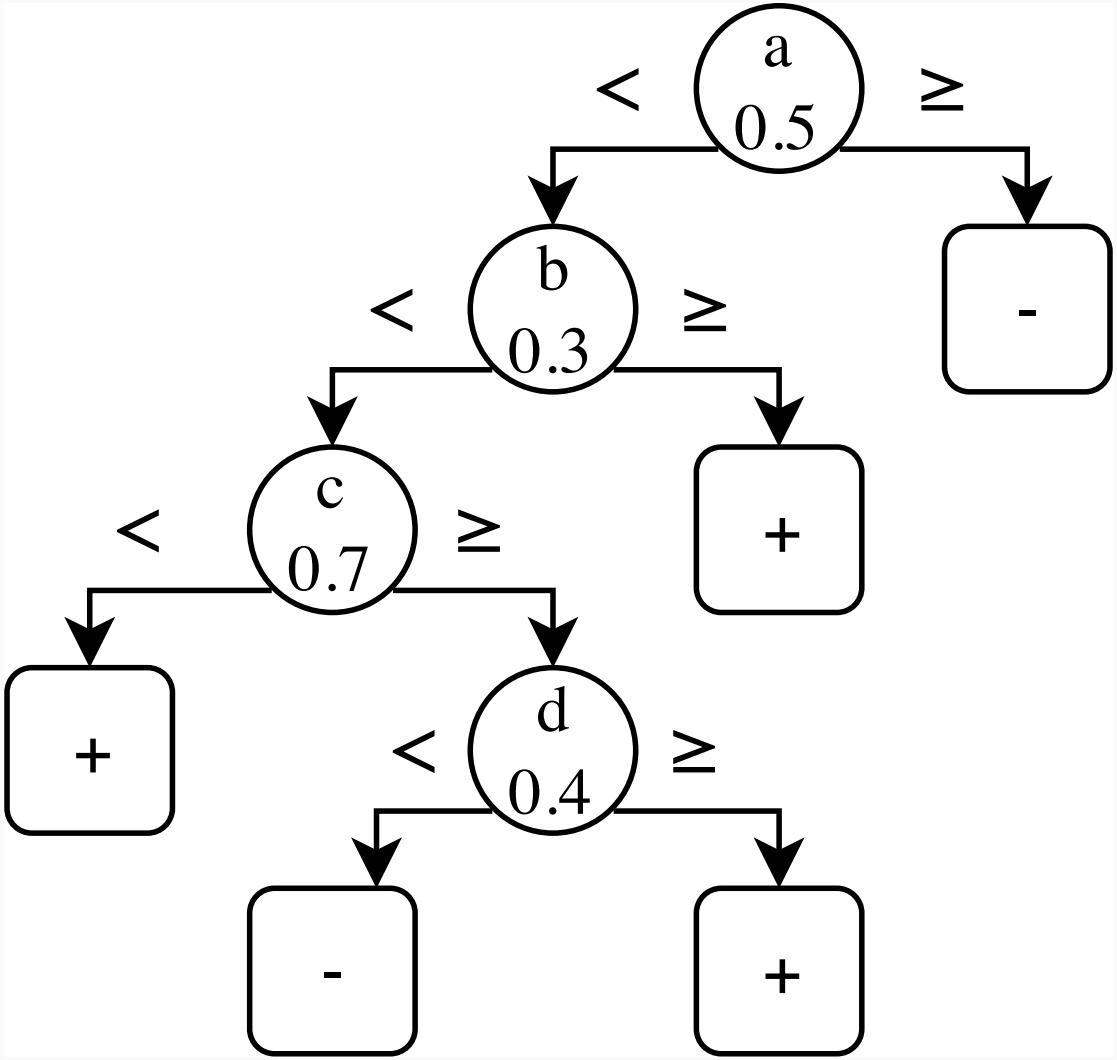
$$x' = \arg\min_{x*} \left\{ \delta(x, x*) | \hat{f}(x) = - \wedge \hat{f}(x*) = + \right\} \qquad (2)$$

Counterfactual explanations for decision tree ensembles

- Tolomei et.al., considered the problem of identifying a transition function using decision tree ensembles
- Infinite number of possible transformations
- Use the information in the individual decision trees to *try* and change the ensemble decision
- Finding the optimal transition for the entire ensemble is **NP**-hard[4]
- A transformation must change a majority of the ensemble members

[4]Karlsson, Isak, et al. *Explainable time series tweaking via irreversible and reversible temporal transformations*, 2018

Decision tree paths



Decision tree paths

The root-to-leaf path of a decision tree can be interpreted as a cascade of *if-then-else* statements:

- $a \geq 0.5 \rightarrow -$
- $a < 0.5 \wedge b < 0.3 \wedge c \geq 0.7 \wedge d < 0.4 \rightarrow -$
- $a < 0.5 \wedge b \geq 0.3 \rightarrow +$
- $a < 0.5 \wedge b < 0.3 \wedge c < 0.7 \rightarrow +$
- $a < 0.5 \wedge b < 0.3 \wedge c < 0.7 \wedge d \geq 0.4 \rightarrow +$

Decision tree ensemble transitions

The basic model is as follows:

1. Given an instance x
2. For every tree in the ensemble where both the forest and the tree agree on a negative classification, i.e., $\hat{f}(x) = - \wedge \hat{f}(x) = \hat{h}_k(x)$
3. Extract all positive prediction paths, p_k^+, from the k:th tree
4. For each positive prediction path $p_{k,j}^+ \in p_k^+$, construct a positive instance according to:

$$x_j^{'\epsilon} = \begin{cases} \theta_i - \epsilon & \text{if the i:th condition is } x_i < \theta_i \\ \theta_i + \epsilon & \text{if the i:th condition is } x_i \geq \theta_i \end{cases} \tag{3}$$

and θ_i is the z-standardized feature split value (e.g., a in the previous slide)

Example

- Given path, $a < 0.5 \wedge b < 0.3 \wedge c < 0.7 \wedge d \geq 0.4$ and $\epsilon = 0.1$
- and a negative prediction x (i.e., $\hat{f}(x) = -$):
  - $x[a] = 0.7$
  - $x[b] = 0.2$
  - $x[c] = 0.9$
  - $x[d] = 0.1$
- Then, according to the path, x' transformed into:
  - $x[a] = 0.5 - 0.1$, since $x[a] > 0.5$
  - $x[b] = 0.3 + 0.1$, since $x[b] < 0.3$
  - $x[c] = 0.7 - 0.1$, since $x[c] > 0.7$
  - $x[d] = 0.4 + 0.1$, since $x[d] < 0.4$
- New example: $x' = [0.4, 0.4, 0.6, 0.5]$

Decision tree ensemble transitions

5 If the newly created instance $x_j'^{\epsilon}$ is considered positive by **the full ensemble**, that is $\hat{f}(x) = +$

5 Check whether or not the transformation has the lowest cost observed so far, i.e., $\delta(x, x_j'^{\epsilon}) < \delta_{min}$.

5 If the cost is lowest, set the current transformation as the best.

Tolomei et. al., suggests two alternative implementations of δ:

- The euclidean distance between the original and transformed instance
- The number of attributes values that have been changed

Evaluation

The method is evaluated on an ad-quality prediction task: How can a **bad landing page** be converted into a **good landing page**?

- Prediction task at Yahoo
- Some suggestions are useful others not
- Some suggestions are not possible
- Some landing pages cannot be transitioned

Other transition approaches

- Instead of only utilizing the prediction model...

Other transition approaches

- Instead of only utilizing the prediction model. . .
- why don't we utilize the training example that built the model?

1-nearest-neighbor transitions

- For a 1-nearest neighbor under any distance measure d the least costly transition from x to x' is given by:

$$\underset{\{x' | (\hat{y}, x') \in \mathcal{X}, \hat{y} = y'\}}{\arg\min} \quad d(x, x'). \tag{4}$$

- This transition function is guaranteed to find the transition among the examples in the training set that minimizes the transformation cost
- Karlsson et.al (2018, 2019)[5]

[5] *Locally and globally explainable time series tweaking* and *Explainable time series tweaking via irreversible and reversible temporal transformations*

k-nearest-neighbor transitions

- How can we generalize the former algorithm to arbitrary k?
- One idea, explored by Karlsson et al. (2019), is to use clustering

k-nearest-neighbor transitions

Input : A desired number of k nearest neighbours, a training set $\mathcal{X}$ with corresponding labels $\mathcal{Y}$, an example x to be transformed and a desired class y'

Output: A transformed example x'

1 $C \leftarrow \lfloor \frac{|\mathcal{X}|}{k} \rfloor$

2 Apply k-means clustering with C clusters to $\mathcal{X}$, resulting in a set of centroids $\mathcal{K}$

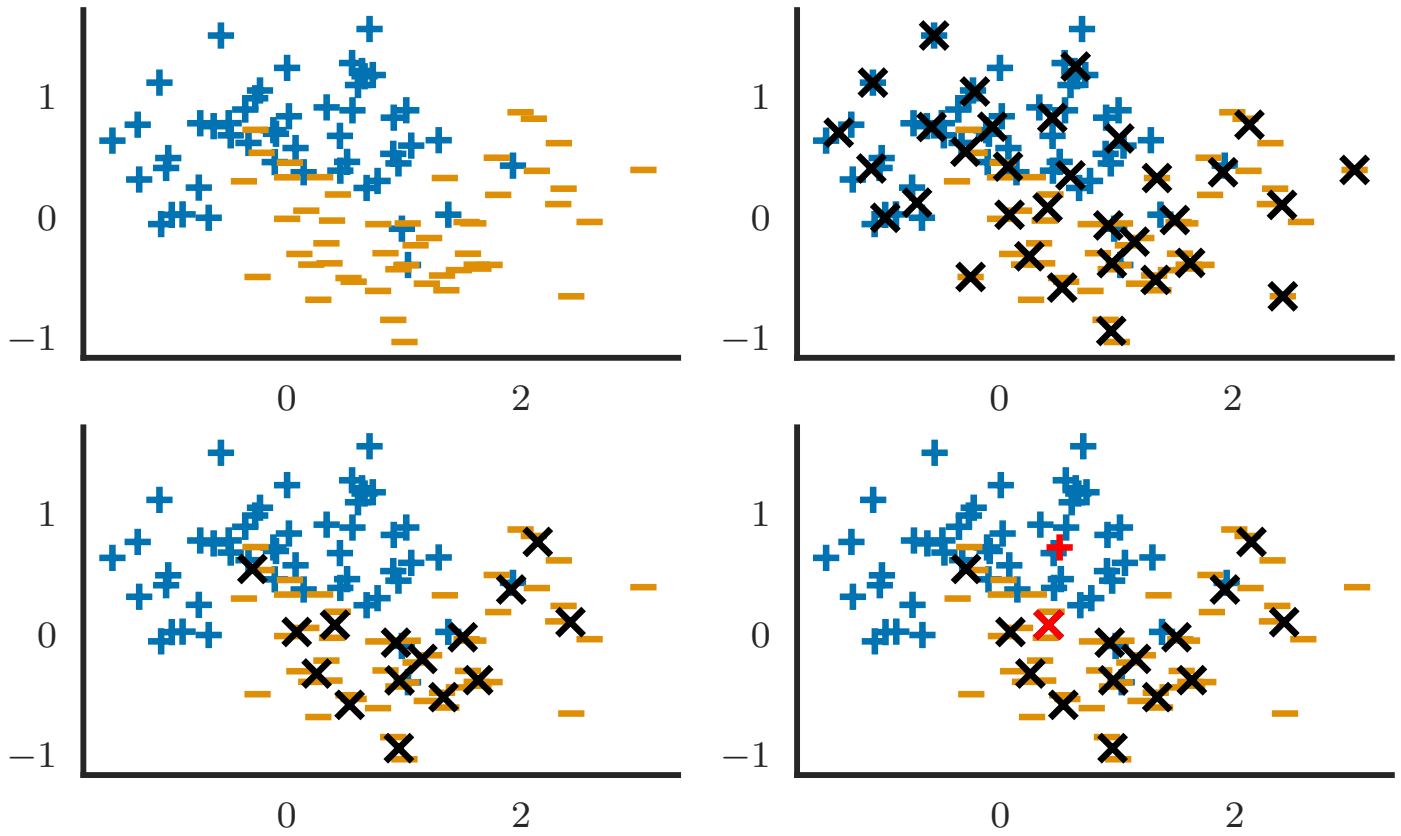
3 Select the centroids in $\mathcal{K}$ such that a majority of the examples closest to $K \in \mathcal{K}$ are labeled as y' and the number of time series labeled as $y' > \frac{k}{|\mathcal{C}|} + 1$, i.e.,

$$\mathcal{K}' \leftarrow \left\{ K \in \mathcal{K} \mid \text{majority}(K, \mathcal{X}, \mathcal{Y}) = y' \wedge \text{count}(K, \mathcal{X}, \mathcal{Y}, y') > \frac{k}{\mathcal{C}} + 1 \right\}$$

4 $x' \leftarrow \arg\min_{K \in \mathcal{K}'} d(K, x)$

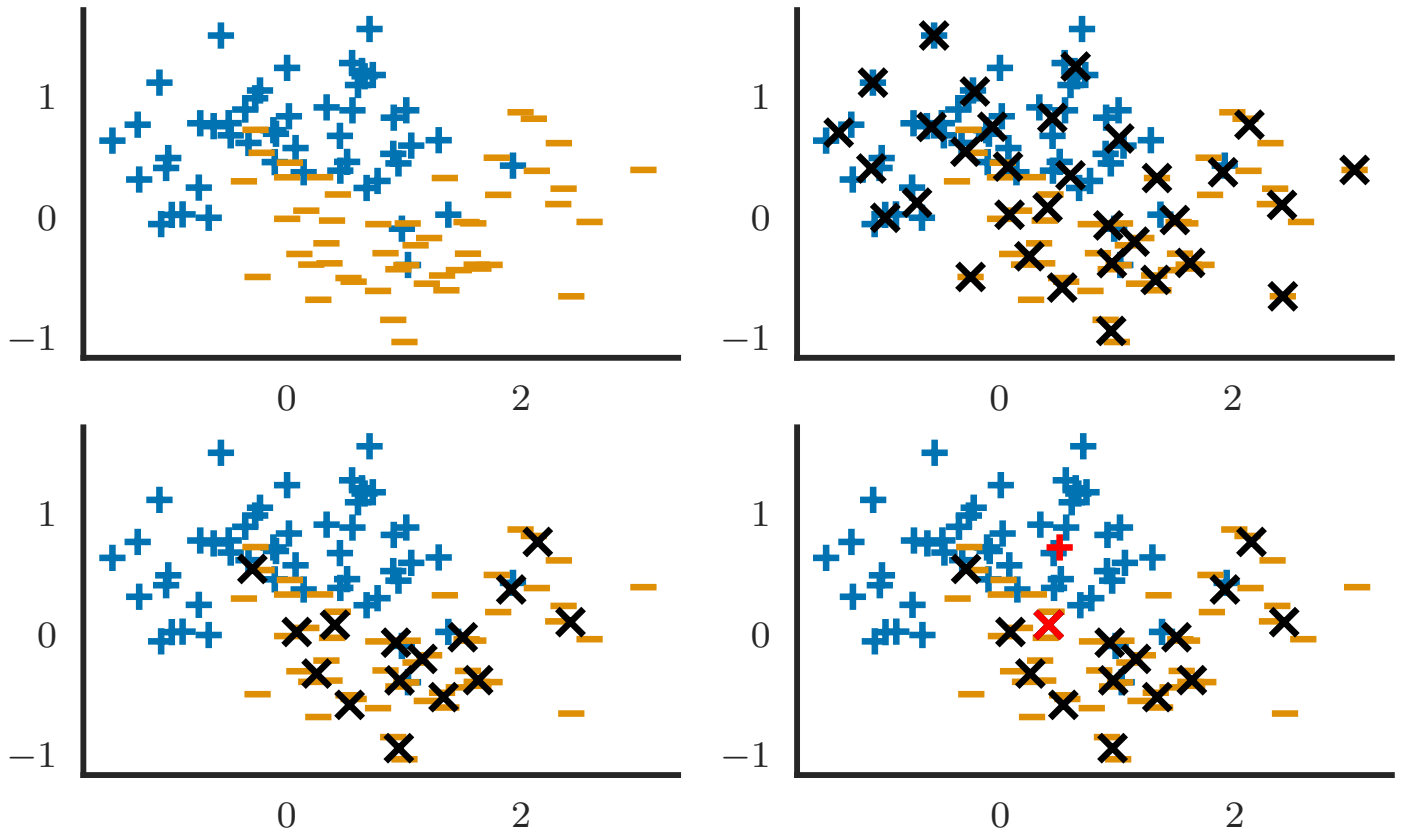
5 **return** x'

k-nearest-neighbor transitions



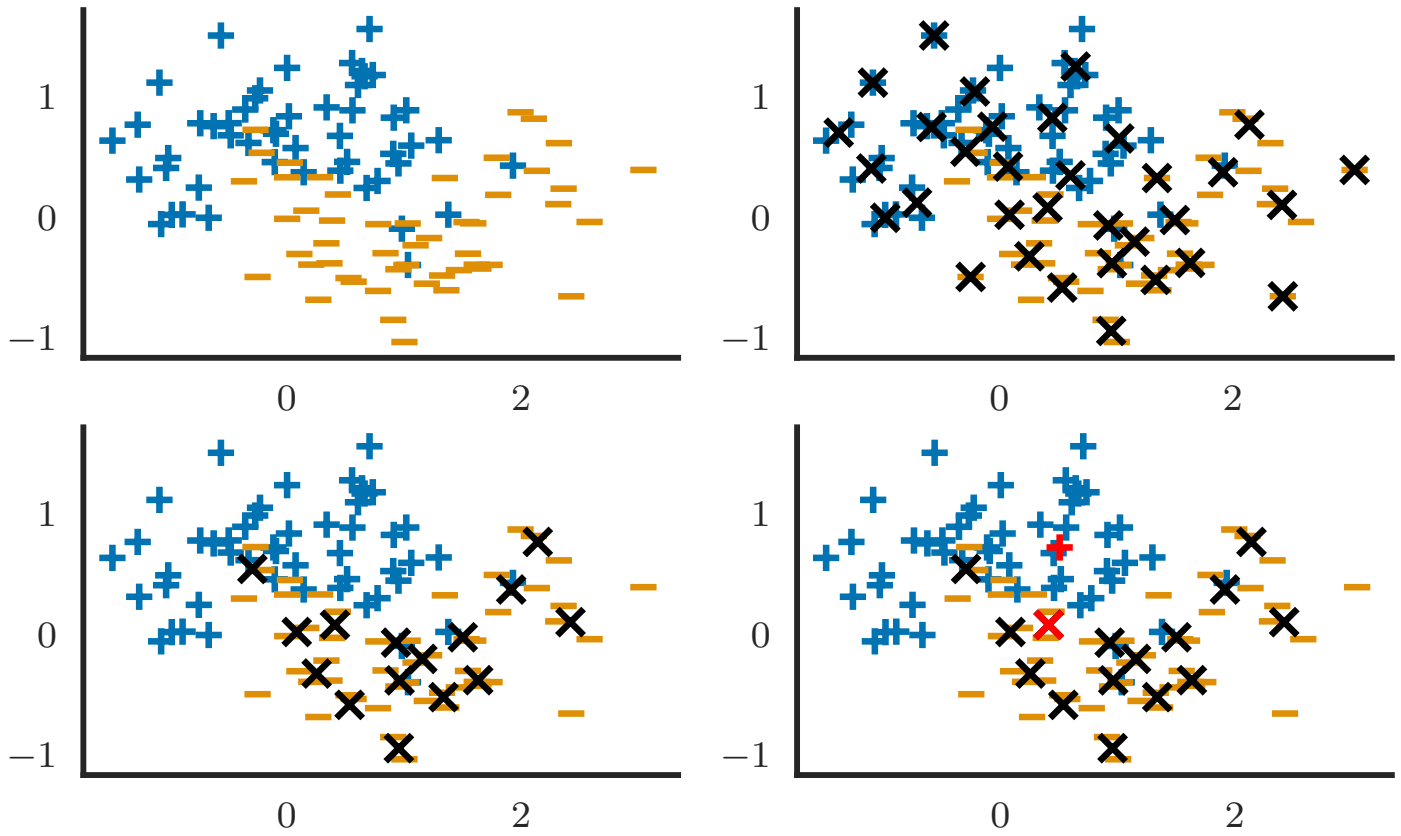
- Given $k = 3$, find $\lfloor \frac{100}{3} = 33 \rfloor$ centroids (top-right)

k-nearest-neighbor transitions



- Given $k = 3$, find $\lfloor \frac{100}{3} = 33 \rfloor$ centroids (top-right)
- Select the transition guaranteed centroids (bottom-left)

k-nearest-neighbor transitions



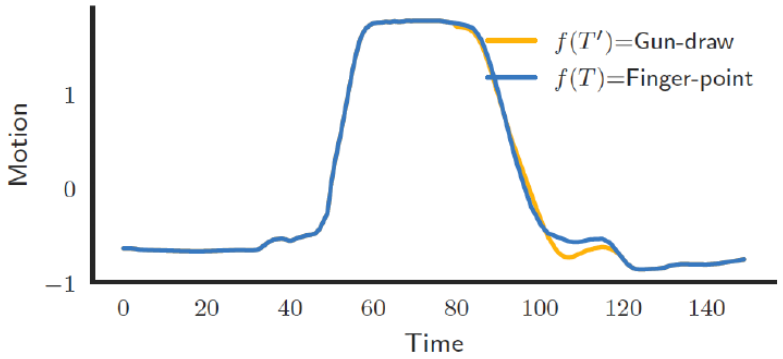
- Given $k = 3$, find $\lfloor \frac{100}{3} = 33 \rfloor$ centroids (top-right)
- Select the transition guaranteed centroids (bottom-left)
- Transform to the closest centroids (bottom-right)

Now what about other types of data?

- What if we have other types of data?
- For instance, time series where the individual attributes correspond to the values at particular times
- The values are thus sequentially dependent
- Traditional transition functions are not well-behaved, nor are traditional machine learning algorithms applicable
- Though, given a well-defined distance measure the k-nearest neighbor approach works!

Time series transition using decision tree ensembles

What is the minimum number of changes to apply to a time series ***T*** so that a given opaque classifier changes its prediction?



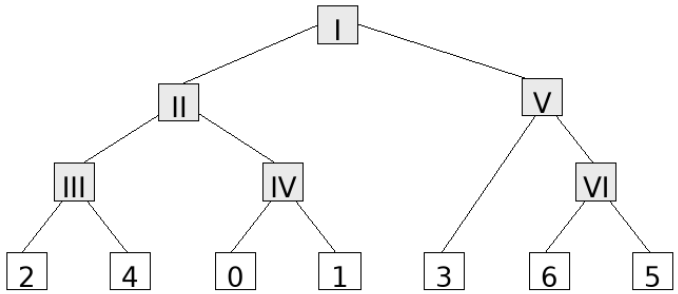
$$\mathcal{T} \rightarrow \mathcal{T}^1 \rightarrow \mathcal{T}^2 \rightarrow \ldots \rightarrow \mathcal{T}'$$

- **Reversible tweaking:** each subsequent transformation **can** override a previous one
- **Irreversible tweaking:** each subsequent transformation **cannot** override a previous one

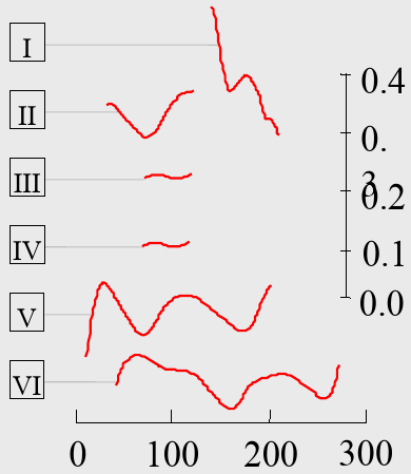
Decision tree ensembles for time series?

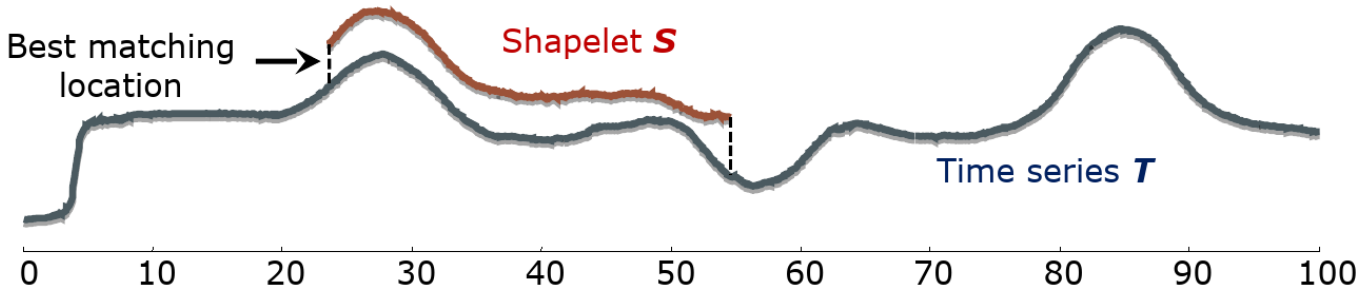
Shapelets: class-distinctive time series subsequences capturing local trends in time series

Shapelet Tree



Shapelet Dictionary

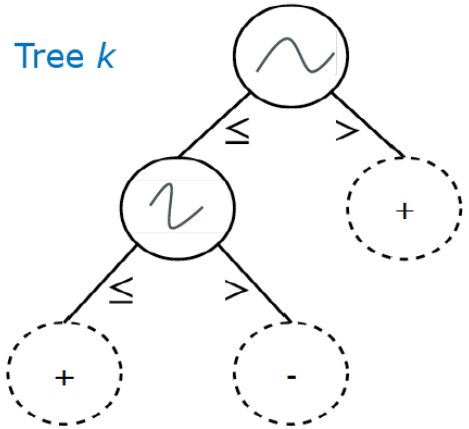


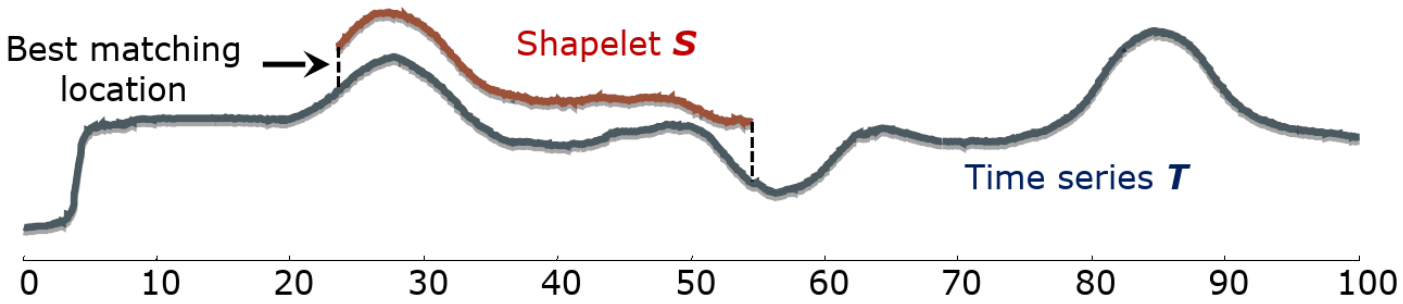


Root-to-leaf paths for shapelet trees

- Each tree defines several root-leaf paths
- The *j*th path of *the k*th *tree* is defined as follows:

$$p_{k,j} = \{(x_1 \lesseqgtr \theta_1), (x_2 \lesseqgtr \theta_2), \ldots, (x_n \lesseqgtr \theta_n)\}$$





Shapelet tree transitions

- Focus on the trees that predict '*-1*'
- For each tree, explore the positive paths, i.e., those that predict '*+1*'
- Try to force those trees to predict '*+1*' by *tweaking* parts of of ***T***

Given a non-leaf node (S^j_k, θ^j_k)

- **Case (a) increase distance**
  - if S^j_k exists in ***T***, that is: $d_s(\mathcal{S}^j_k, \mathcal{T}) \leq \theta^j_k)$
  - *and* the current node condition demands otherwise
  - ✓ We increase the distance of **all matching instances** of S^j_k, so that they all fall **above the distance threshold** θ^j_k

Shapelet tree transitions

- Focus on the trees that predict '*-1*'
- For each tree, explore the positive paths, i.e., those that predict '*+1*'
- Try to force those trees to predict '*+1*' by *tweaking* parts of of ***T***

Given a non-leaf node (S^j_k, θ^j_k)

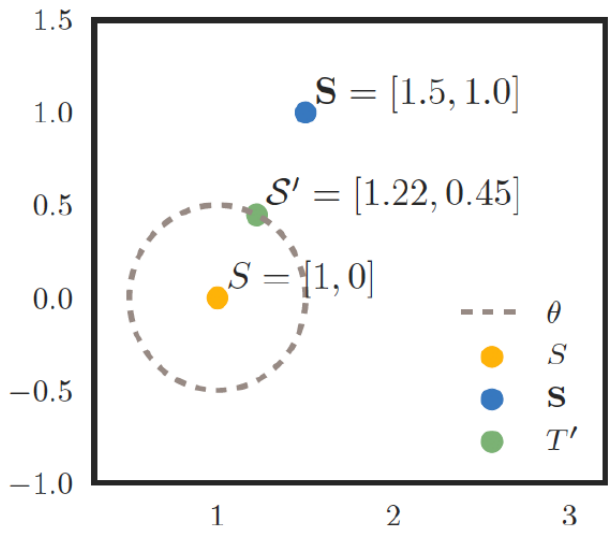
- **Case (b) decrease distance**
  - if S^j_k does not exist in ***T***, that is $d_s(\mathcal{S}^j_k, \mathcal{T}) > \theta^j_k)$
  - and the current node condition demands otherwise
  - ✓ We decrease the distance of **the best matching instance** of S^j_k, so that it falls **below the distance threshold** θ^j_k

How can a time series be moved?

- Consider S as an *m*-dimensional point
- Define an *m*-sphere with S as its center and radius θ

- The transformed time series counterpart is given by the following equation:

$$\tau_{\mathcal{S}}(\mathbf{S}, p_{ik}^{j}, \epsilon) = \mathcal{S}_k^j + \frac{\mathcal{S}_k^j - \mathbf{S}}{\|\mathcal{S}_k^j - \mathbf{S}\|_2}(\theta_k^j + (\epsilon\delta_{ik}^j))$$



Shapelet forest transitions transition

Algorithm 1: Reversible time series tweaking algorithm (τ_{RT})

input : A shapelet forest $\mathcal{R}$, a time series $\mathcal{T}$ and a desired class y' and transformation strength ϵ
output: A transformed time series $\mathcal{T}'$

$\mathcal{T}' \leftarrow \mathcal{T}.copy$
$c_{min} \leftarrow \infty$
for $j \leftarrow 1$ **to** $|\mathcal{R}|$, $k \leftarrow 1$ **to** $|F_j|$ **do**
  for $i \leftarrow 1$ **to** u **do**
    if $y^k = y' \wedge \phi(\mathcal{T}', p^j_{ik})$ *is* false **then**
      $\mathbf{T} \leftarrow \mathcal{T}'.copy$
      if $d_s(\mathcal{S}^j_k, \mathcal{T}) \leq \theta^j_k$ **then**
        while $d_s(\mathcal{S}^j_k, \mathcal{T}) \leq \theta^j_k$ **do**
          $idx \leftarrow$ start index of subsequence with lowest distance, $d_s(\mathcal{S}^j_k, \mathbf{T})$
          $\mathcal{S}' \leftarrow \tau_{\mathcal{S}}(\mathbf{T}[idx : idx + |\mathcal{S}^j_k|], p^j_{ik}, \epsilon)$
          Assign $\mathcal{S}'$ **to** $\mathbf{T}[idx : idx + |\mathcal{S}^j_k|]$
      else
        $idx \leftarrow$ start index of subsequence with lowest distance, $d_s(\mathcal{S}^j_k, \mathbf{T})$
        $\mathcal{S}' \leftarrow \tau_{\mathcal{S}}(\mathbf{T}[idx : idx + |\mathcal{S}^j_k|], p^j_{ik}, \epsilon)$
        Assign $\mathcal{S}'$ **to** $\mathcal{T}[idx : idx + |\mathcal{S}^j_k|]$
  if $c(\mathbf{T}, \mathcal{T}) < c_{min} \wedge f(\mathbf{T}, \mathcal{R}) = y'$ **then**
    $\mathcal{T}' \leftarrow \mathbf{T}$
    $c_{min} \leftarrow d(\mathcal{T}', \mathcal{T})$
return $\mathcal{T}'$

If the *class* of the path is the same as the desired class **and** case (a) or (b) does not hold

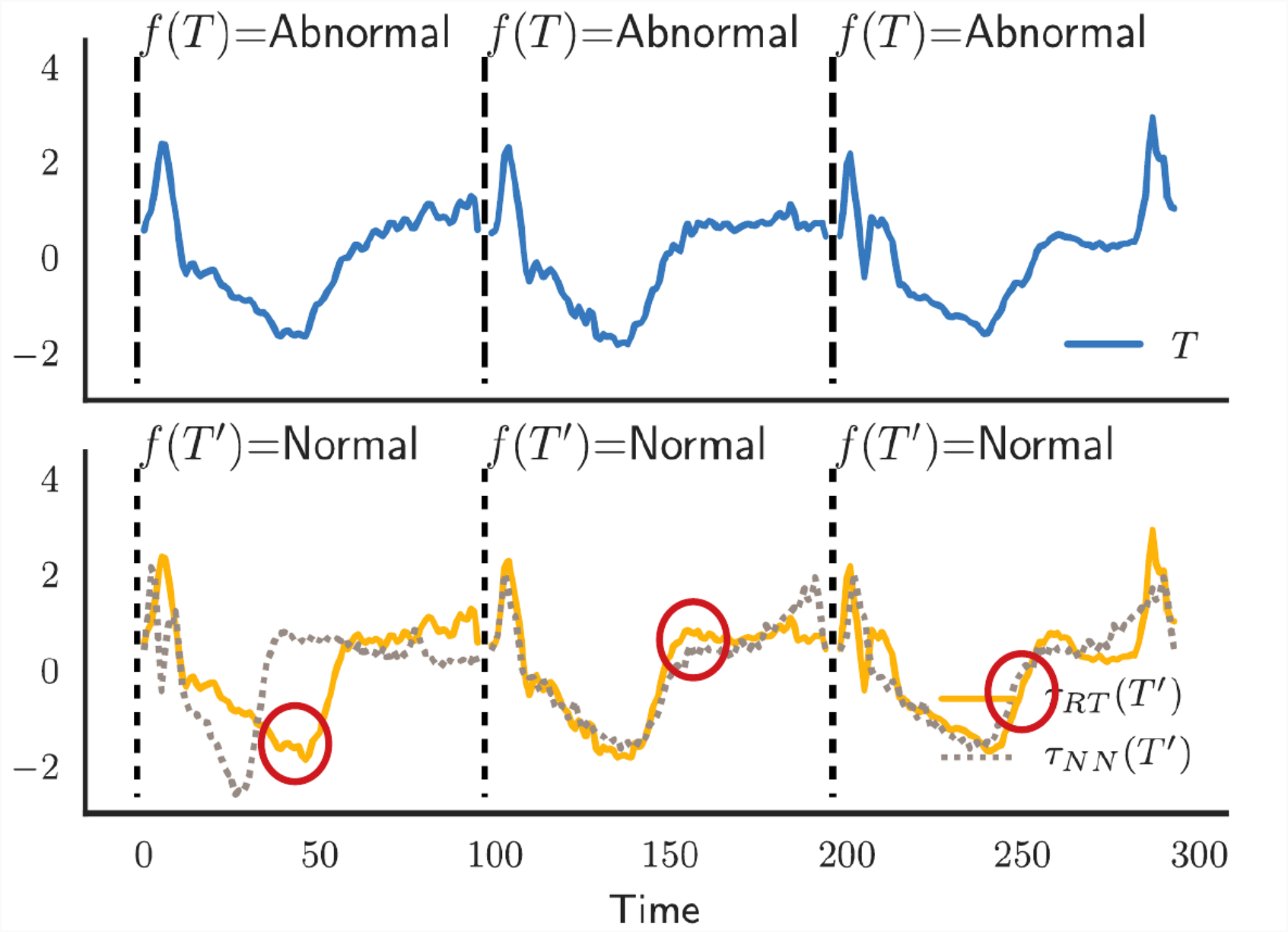
For case (a), increase **all** matching positions

Update the subsequences at the matching position

For case (b), decrease the **best** matching position

Compute the cost of transformation from **T** to T (Euclidean distance)

Example


$f(T)$=Abnormal
$f(T)$=Abnormal
$f(T)$=Abnormal
T
$f(T')$=Normal
$f(T')$=Normal
$f(T')$=Normal
$\tau_{RT}(T')$
$\tau_{NN}(T')$
4
2
0
−2
0
50
100
150
200
250
300
Time

Evaluating interpretability

Evaluation

Function based

- How important are the attributes?
- How does attribute importance differ between method X and Y?
- In synthetic data with attribute Z, method X detects the importance

Cognition based

- What factors should change to change the outcome?
- What are the discriminative attributes?

Application based

- How much did we improve patient outcome?
- Do doctors find the explanations useful?

Involving humans

- Hard to compare results between different methods

Involving humans

- Hard to compare results between different methods
- Costly in terms of resources

Involving humans

- Hard to compare results between different methods
- Costly in terms of resources
- Uncomfortable for computer scientists :-)

Involving humans

- Hard to compare results between different methods
- Costly in terms of resources
- Uncomfortable for computer scientists :-)
- But, it's real evaluation