# Foundations of Explainable AI and Applications to AutoML

Marius Lindauer

Leibniz Universität Hannover

LUH|AI

Forschungszentrum · Research Center L3S

XAISS'22

# Acknowledgments

Several groups, incl.

- Bernd Bischl, Giuseppe Casalicchio et al. at LMU (Munich)
- Marvin Wright at Univ. Bremen
- Avishek Anand at TU Delft
- Marius Lindauer at LUH (Hannover)

are jointly working on an upcoming MOOC on iML. The material is partially taken from that and should allow you to flawlessly dive deeper into it in the future.

# Agenda

Goals for today's lecture:

1. Global effects via Partial Dependence Plots (PDPs)
2. Contribution of a feature to a prediction via SHAP
3. Local explanation via LIME
4. Explaining the effects of hyperparameters (AutoML) via iML

# Model-Agnostic vs. Model-Specific Methods

Leibniz Universität Hannover

Model-agnostic methods can be applied to all kinds of predictive models to explain them

Model-specific methods can only be applied to one model-class by making use of how the model is represented or constructed

$\rightsquigarrow$ I will focus on model-agnostic methods today

# Advantages of Model-Agnostic Methods [Ribeiro et al. 2016]

Model flexibility: Method can be applied to any predictive model, e.g., RFs or DNNs

Explanation flexibility: Different explanations can be applied (based on the needs of the user)

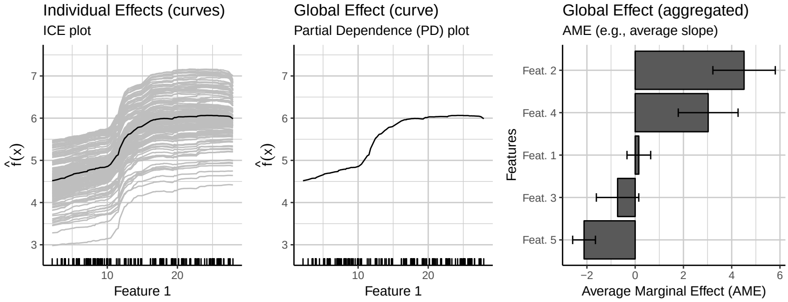
Representation Flexibility: Different representations of features (e.g., tabular, images or text) should be applicable.

# PDP: Partial Dependence Plots

# Global Feature Effects

Global Feature Effects visualize or quantify the (average) relationship between the features and the model predictions.

- Methods: PD Plots, ICE curves, ALE plots




$$\text{Individual (curves)} \xrightarrow{\text{aggregate}} \text{Global (curve)} \xrightarrow{\text{aggregate}} \text{Global (number)}$$

# Individual Conditional Expectation (ICE) [Goldstein et al. 2014]

- each observation $x^{(i)}$ can be partitioned into $x_S^{(i)}$ and $x_C^{(i)}$
  - $x_S^{(i)}$ containing the features of interest
  - $x_C^{(i)}$ the remaining features

# Individual Conditional Expectation (ICE) [Goldstein et al. 2014]

- each observation $x^{(i)}$ can be partitioned into $x_S^{(i)}$ and $x_C^{(i)}$
  - $x_S^{(i)}$ containing the features of interest
  - $x_C^{(i)}$ the remaining features

ICE curves visualize how the model prediction of individual observations $x^{(i)}$ change by varying the feature values in $x_S$ while keeping all other features in $x_C^{(i)}$ fixed:

$$\hat{f}_S^{(i)}(x_S) = \hat{f}(x_S, x_C^{(i)})$$

# Individual Conditional Expectation (ICE) [Goldstein et al. 2014]

- each observation $x^{(i)}$ can be partitioned into $x_S^{(i)}$ and $x_C^{(i)}$
  - $x_S^{(i)}$ containing the features of interest
  - $x_C^{(i)}$ the remaining features

ICE curves visualize how the model prediction of individual observations $x^{(i)}$ change by varying the feature values in $x_S$ while keeping all other features in $x_C^{(i)}$ fixed:

$$\hat{f}_S^{(i)}(x_S) = \hat{f}(x_S, x_C^{(i)})$$

In practice, $x_S$ consists of one or two features.

# Individual Conditional Expectation (ICE) [Goldstein et al. 2014]

- each observation $x^{(i)}$ can be partitioned into $x_S^{(i)}$ and $x_C^{(i)}$
  - $x_S^{(i)}$ containing the features of interest
  - $x_C^{(i)}$ the remaining features

ICE curves visualize how the model prediction of individual observations $x^{(i)}$ change by varying the feature values in $x_S$ while keeping all other features in $x_C^{(i)}$ fixed:

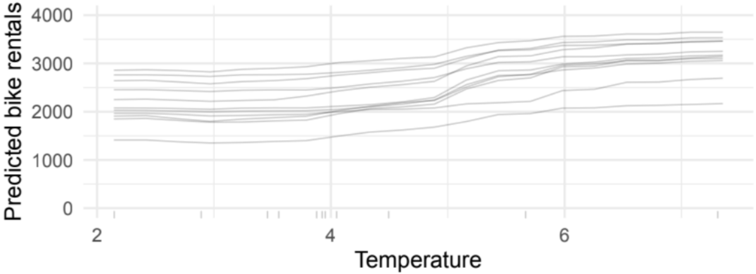
$$\hat{f}_S^{(i)}(x_S) = \hat{f}(x_S, x_C^{(i)})$$

In practice, $x_S$ consists of one or two features.

Note: $\hat{f}$ indicates we do this on our predictive model by simply querying it at different $x_S$.

# ICE Curves






- Each line displays the change in prediction for a single observation due to varying the feature temperature.

# Individual Conditional Expectation (ICE)

| $x_1$ | $x_2$ | $x_3$ |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | 9 |

**Sampling:** Choose grid points along $x_1$ that will be used to intervene the data (here: all unique values $1, 2$ and $3$).

# Individual Conditional Expectation (ICE)

| $x_1$ | $x_2$ | $x_3$ |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | 9 |

⇒

| $i$ | $x_S$ | $x_2$ | $x_3$ |
|---|---|---|---|
| 1 | 1 | 4 | 7 |
| 2 | 1 | 5 | 8 |
| 3 | 1 | 6 | 9 |

| $i$ | $x_S$ | $x_2$ | $x_3$ |
|---|---|---|---|
| 1 | 2 | 4 | 7 |
| 2 | 2 | 5 | 8 |
| 3 | 2 | 6 | 9 |

| $i$ | $x_S$ | $x_2$ | $x_3$ |
|---|---|---|---|
| 1 | 3 | 4 | 7 |
| 2 | 3 | 5 | 8 |
| 3 | 3 | 6 | 9 |

**Intervention:** Replace all observed values in $x_1$ for each observation with the previously sampled grid points.

# Individual Conditional Expectation (ICE)

| $x_1$ | $x_2$ | $x_3$ |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | 9 |

⇨

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 1 | 4 | 7 | 0.4 |
| 2 | 1 | 5 | 8 | 0.6 |
| 3 | 1 | 6 | 9 | 0.1 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 0.6 |
| 2 | 2 | 5 | 8 | 0.8 |
| 3 | 2 | 6 | 9 | 0.5 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 3 | 4 | 7 | 0.7 |
| 2 | 3 | 5 | 8 | 0.9 |
| 3 | 3 | 6 | 9 | 0.6 |

**Prediction:** Make predictions and plot $\hat{f}_1^{(i)}(x_1)$ vs. $x_1$, where

$$\hat{f}_1^{(i)}(x_1) = \hat{f}(x_1, x_{2,3}^{(i)}).$$

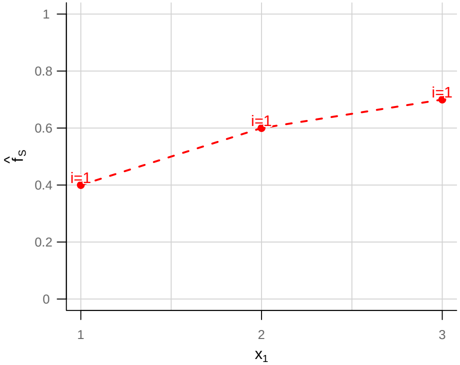
# Individual Conditional Expectation (ICE)



| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 1 | 4 | 7 | 0.4 |
| 2 | 1 | 5 | 8 | 0.6 |
| 3 | 1 | 6 | 9 | 0.1 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 0.6 |
| 2 | 2 | 5 | 8 | 0.8 |
| 3 | 2 | 6 | 9 | 0.5 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 3 | 4 | 7 | 0.7 |
| 2 | 3 | 5 | 8 | 0.9 |
| 3 | 3 | 6 | 9 | 0.6 |



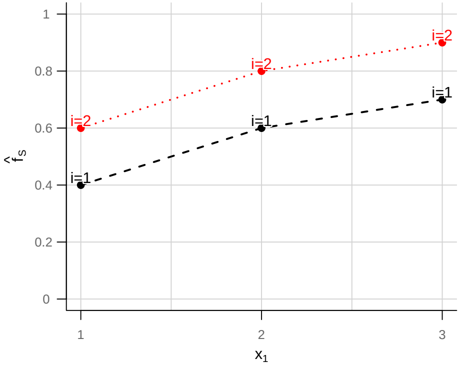

**Visualization:** ICE curve for observation $i = 1$ connects all predictions at the corresponding grid points associated to the $i$-th observation.

# Individual Conditional Expectation (ICE)

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 1 | 4 | 7 | 0.4 |
| 2 | 1 | 5 | 8 | 0.6 |
| 3 | 1 | 6 | 9 | 0.1 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 0.6 |
| 2 | 2 | 5 | 8 | 0.8 |
| 3 | 2 | 6 | 9 | 0.5 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 3 | 4 | 7 | 0.7 |
| 2 | 3 | 5 | 8 | 0.9 |
| 3 | 3 | 6 | 9 | 0.6 |



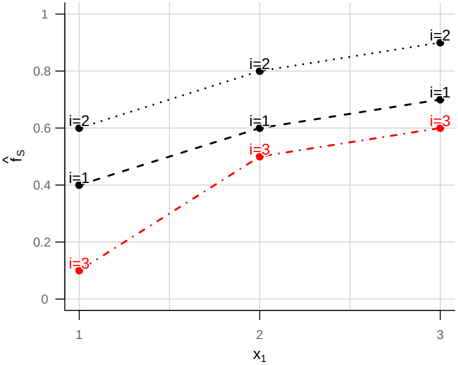

**Visualization:** ICE curve for observation $i = 2$ connects all predictions at the corresponding grid points associated to the $i$-th observation.

# Individual Conditional Expectation (ICE)

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 1 | 4 | 7 | 0.4 |
| 2 | 1 | 5 | 8 | 0.6 |
| 3 | 1 | 6 | 9 | 0.1 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 0.6 |
| 2 | 2 | 5 | 8 | 0.8 |
| 3 | 2 | 6 | 9 | 0.5 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 3 | 4 | 7 | 0.7 |
| 2 | 3 | 5 | 8 | 0.9 |
| 3 | 3 | 6 | 9 | 0.6 |




**Visualization:** ICE curve for observation $i = 3$ connects all predictions at the corresponding grid points associated to the $i$-th observation.

# Partial Dependence [Friedman 2001] [Scholbeck et al. 2019]

The partial dependence (PD) plot is the expectation of the ICE curves w.r.t. the marginal distribution of complementary features $x_C$:

$$\mathbb{E}_{x_C}\left(\hat{f}(x_S, x_C)\right) = \int_{-\infty}^{\infty} \hat{f}(x_S, x_C)\, dP(x_C)$$

For a single $x_S$, it is estimated by the point-wise average of the ICE curves:

$$\mathrm{PD}_S := \hat{f}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)})$$

# Partial Dependence: Example

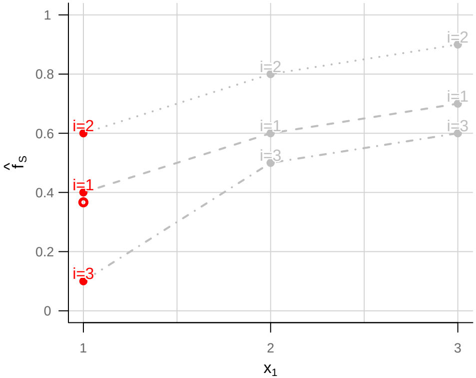


| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 1 | 4 | 7 | 0.4 |
| 2 | 1 | 5 | 8 | 0.6 |
| 3 | 1 | 6 | 9 | 0.1 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 0.6 |
| 2 | 2 | 5 | 8 | 0.8 |
| 3 | 2 | 6 | 9 | 0.5 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 3 | 4 | 7 | 0.7 |
| 2 | 3 | 5 | 8 | 0.9 |
| 3 | 3 | 6 | 9 | 0.6 |

| $\frac{1}{3}\sum_{i=1}^{3}\hat{f}$ |
|---|
| ⅓ (0.4 + 0.6 + 0.1) |
| ⅓ (0.6 + 0.8 + 0.5) |
| ⅓ (0.7 + 0.9 + 0.6) |




**Aggregation:** Estimate partial dependence by the point-wise average of the ICE curves at $\boxed{x_S = x_1 = 1}$:

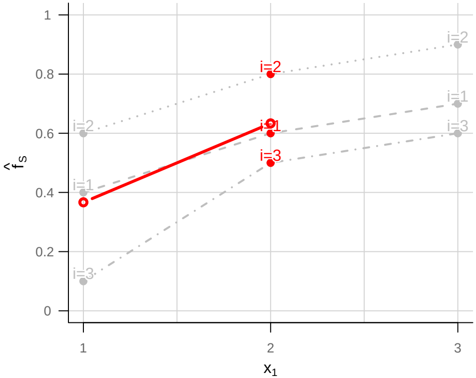
$$PD_1 = \hat{f}_1(x_1) = \frac{1}{n}\sum_{i=1}^{n}\hat{f}(x_1, x_{2,3}^{(i)})$$

# Partial Dependence: Example

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 1 | 4 | 7 | 0.4 |
| 2 | 1 | 5 | 8 | 0.6 |
| 3 | 1 | 6 | 9 | 0.1 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ | $\frac{1}{3}\sum_{i=1}^{3}\hat{f}$ |
|---|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 0.6 | ⅓ (0.4 + 0.6 + 0.1) |
| 2 | 2 | 5 | 8 | 0.8 | ⅓ (0.6 + 0.8 + 0.5) |
| 3 | 2 | 6 | 9 | 0.5 | ⅓ (0.7 + 0.9 + 0.6) |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 3 | 4 | 7 | 0.7 |
| 2 | 3 | 5 | 8 | 0.9 |
| 3 | 3 | 6 | 9 | 0.6 |




**Aggregation:** Estimate partial dependence by the point-wise average of the ICE curves at $\boxed{x_S = x_1 = 2}$:

$$PD_1 = \hat{f}_1(x_1) = \frac{1}{n}\sum_{i=1}^{n}\hat{f}(x_1, x_{2,3}^{(i)})$$

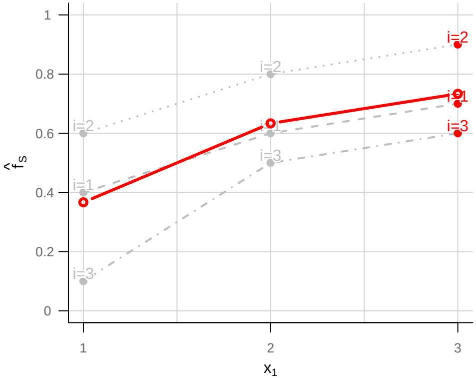
# Partial Dependence: Example

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 1 | 4 | 7 | 0.4 |
| 2 | 1 | 5 | 8 | 0.6 |
| 3 | 1 | 6 | 9 | 0.1 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 0.6 |
| 2 | 2 | 5 | 8 | 0.8 |
| 3 | 2 | 6 | 9 | 0.5 |

| $\frac{1}{3}\sum_{i=1}^{3}\hat{f}$ |
|---|
| ⅓ (0.4 + 0.6 + 0.1) |
| ⅓ (0.6 + 0.8 + 0.5) |
| ⅓ (0.7 + 0.9 + 0.6) |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 3 | 4 | 7 | 0.7 |
| 2 | 3 | 5 | 8 | 0.9 |
| 3 | 3 | 6 | 9 | 0.6 |




**Aggregation:** Estimate partial dependence by the point-wise average of the ICE curves at $\boxed{x_S = x_1 = 3}$:

$$PD_1 = \hat{f}_1(x_1) = \frac{1}{n}\sum_{i=1}^{n}\hat{f}(x_1, x_{2,3}^{(i)})$$

# Partial Dependence: Example

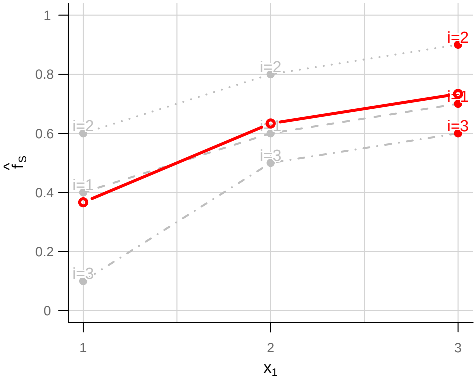


| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 1 | 4 | 7 | 0.4 |
| 2 | 1 | 5 | 8 | 0.6 |
| 3 | 1 | 6 | 9 | 0.1 |

| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 0.6 |
| 2 | 2 | 5 | 8 | 0.8 |
| 3 | 2 | 6 | 9 | 0.5 |

| $\frac{1}{3}\sum_{i=1}^{3}\hat{f}$ |
|---|
| ⅓ (0.4 + 0.6 + 0.1) |
| ⅓ (0.6 + 0.8 + 0.5) |
| ⅓ (0.7 + 0.9 + 0.6) |

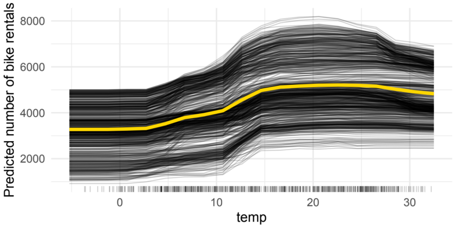
| $i$ | $x_S$ | $x_2$ | $x_3$ | $\hat{f}$ |
|---|---|---|---|---|
| 1 | 3 | 4 | 7 | 0.7 |
| 2 | 3 | 5 | 8 | 0.9 |
| 3 | 3 | 6 | 9 | 0.6 |




**Aggregation:** Estimate partial dependence by the point-wise average of the ICE curves at $\boxed{x_S = x_1 = 3}$:

$$PD_1 = \hat{f}_1(x_1) = \frac{1}{n}\sum_{i=1}^{n}\hat{f}(x_1, x_{2,3}^{(i)})$$

# Interpretation: PD and ICE




- **ICE curve:** Visualize how the prediction of an **individual observation** changes if the feature value is changed.
  $\Rightarrow$ ICE is a local interpretation method (black curves).
- **PD plot:** Visualizes the **average effect of a feature**, i.e., how the expected model prediction changes if the feature value is changed.
  $\Rightarrow$ PD plot is a global interpretation method (yellow curve).

# LIME: Local Interpretable Model-agnostic Explanations

# Motivation of Local Explanations

Explaining the **local** behavior of a model.

- Some local methods provide insight into the driving factors for a **particular prediction**.
- Others, help to understand the model's decision making in a **local environment** of the input space.

# Motivation of Local Explanations

Explaining the **local** behavior of a model.

- Some local methods provide insight into the driving factors for a **particular prediction**.
- Others, help to understand the model's decision making in a **local environment** of the input space.
- Local Methods can address questions such as:
  - **Why** did the model decide $y$ for input $x$?
  - **How** decides the model for cases similar to $x$?
  - **What** would the ML model have decided if $x$ differed in $\mathcal{X}$?
  - **Where** does the model fail?

# Social Motivation for Laypersons

- Explanations for laypersons must be tailored for the **explainee** (i.e. the person receiving the explanation).

# Social Motivation for Laypersons

- Explanations for laypersons must be tailored for the **explainee** (i.e. the person receiving the explanation).
- Thus, such explanations should be case specific, easy for humans to understand, and faithful to the explained mechanism.

# Social Motivation for Laypersons

- Explanations for laypersons must be tailored for the **explainee** (i.e. the person receiving the explanation).
- Thus, such explanations should be case specific, easy for humans to understand, and faithful to the explained mechanism.
- In particular if algorithms make decisions in **socially/safety critical domains**, end users have justified interest in receiving explanations.

# Social Motivation for Laypersons

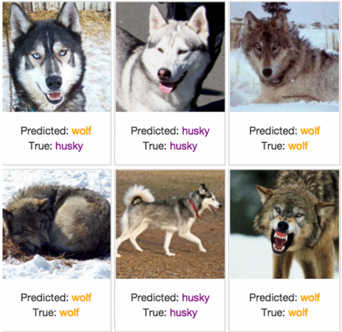
- Explanations for laypersons must be tailored for the **explainee** (i.e. the person receiving the explanation).
- Thus, such explanations should be case specific, easy for humans to understand, and faithful to the explained mechanism.
- In particular if algorithms make decisions in **socially/safety critical domains**, end users have justified interest in receiving explanations.
- Local Explanations can not only increase **user trust**, but also help to detect **critical local biases** in algorithmic decision making.

# Social Motivation for Laypersons

- Explanations for laypersons must be tailored for the **explainee** (i.e. the person receiving the explanation).
- Thus, such explanations should be case specific, easy for humans to understand, and faithful to the explained mechanism.
- In particular if algorithms make decisions in **socially/safety critical domains**, end users have justified interest in receiving explanations.
- Local Explanations can not only increase **user trust**, but also help to detect **critical local biases** in algorithmic decision making.
- European citizens have the legally binding **right to explanation** as given in the General Data Protection Regulation (GDPR; DSGVO in Germany).

# Example: Husky or Wolf?

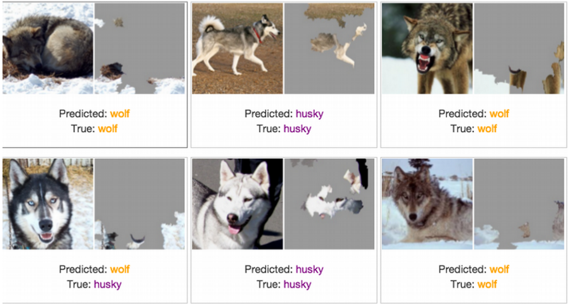
- Model to predict if an image shows a wolf or a husky
- Below the predictions on six test images
- Do you trust our predictor?




Source: [Sameer Singh. 2018]

# Example: Husky or Wolf?

- We can use local explanations (in this case LIME) to highlight the parts of an image which led to the prediction.
- We can see that our predictor is actually a snow detector.




Source: [Sameer Singh. 2018]

# Example: Stop or Right-of-Way?

Assume you work in a car company and are about to use an image classifier for autonomous driving. Then, you show your model the following image (an adversarial example). The classifier is $99\%$ sure it describes a right-of-way sign.


Source: [Eykholt et. al. 2018]

Would you entrust other peoples lives into the hands of this software?

# LIME

- Local Interpretable Model-agnostic Explanations (LIME) assume that even if a machine learning model is very complex, the local prediction can be described with a simpler model.

# LIME

- Local Interpretable Model-agnostic Explanations (LIME) assume that even if a machine learning model is very complex, the local prediction can be described with a simpler model.
- Therefore, LIME explains **individual** predictions of **any** black-box model by approximating the model **locally** with an interpretable model.

# LIME

- Local Interpretable Model-agnostic Explanations (LIME) assume that even if a machine learning model is very complex, the local prediction can be described with a simpler model.
- Therefore, LIME explains **individual** predictions of **any** black-box model by approximating the model **locally** with an interpretable model.
- These approximations are called local surrogate models.
  Typically, they are linear models or trees.

# LIME

- Local Interpretable Model-agnostic Explanations (LIME) assume that even if a machine learning model is very complex, the local prediction can be described with a simpler model.
- Therefore, LIME explains **individual** predictions of **any** black-box model by approximating the model **locally** with an interpretable model.
- These approximations are called local surrogate models.
  Typically, they are linear models or trees.
- They should answer why a machine learning model predicted $y$ for input $x$.

# LIME



- Local Interpretable Model-agnostic Explanations (LIME) assume that even if a machine learning model is very complex, the local prediction can be described with a simpler model.
- Therefore, LIME explains **individual** predictions of **any** black-box model by approximating the model **locally** with an interpretable model.
- These approximations are called local surrogate models.
  Typically, they are linear models or trees.
- They should answer why a machine learning model predicted $y$ for input $x$.
- Since they can be applied to any black-box model they are model-agnostic.

# Formal Definition

LIME provides a local explanation for a black-box model $\hat{f}$ in form of a model $g \in \mathcal{G}$ with $\mathcal{G}$ as the class of potential (interpretable) models. This model $g$ should have two characteristics:

1. It should be **interpretable**, i.e. provide qualitative understanding between the input variables and the response that are easy to understand.
2. It should be **locally faithful**, i.e. it should behave similarly to $\hat{f}$ in the vicinity of the instance being predicted. This characteristic is also called local fidelity.

Formally, we want to receive a model $g$ with minimal complexity and maximal local-fidelity.

# Model Complexity

We can measure the complexity of a model $g$ using $J(g)$.

**Example: Linear model**
Let $\mathcal{G} = \{g : \mathcal{X} \to \mathbb{R} \mid g(x) = s(w^\top x)\}$ be the class of linear models with $s(\cdot)$ being either the identity function for linear regression or the logistic sigmoid function for logistic regression.
Then, $J(g) = \sum_{j=1}^{p} \mathrm{I}_{\{w_j \neq 0\}}$ could be the $\mathrm{L}_0$ loss, i.e. the number of non-zero coefficients.

**Example: Tree**
Let $\mathcal{G} = \left\{g : \mathcal{X} \to \mathbb{R} \mid g(x) = \sum_{m=1}^{M} c_m \mathrm{I}_{\{x \in Q_m\}}\right\}$ be the class of trees (i.e., class of additive model over the leaf-rectangles) then $J(g)$ could measure the number of terminal/leaf nodes.

# Local Model Fidelity

- A model $g$ is locally faithful to $\hat{f}$ w.r.t. a point $x$
  if for points $z \in \mathcal{Z} \subseteq \mathbb{R}^p$ close to $x$, the predictions of $g(z)$ are close to $\hat{f}(z)$.

# Local Model Fidelity

- A model $g$ is locally faithful to $\hat{f}$ w.r.t. a point $x$
  if for points $z \in \mathcal{Z} \subseteq \mathbb{R}^p$ close to $x$, the predictions of $g(z)$ are close to $\hat{f}(z)$.
- In an optimization task: the closer $z$ is to $x$, the closer $g(z)$ should be to $\hat{f}(z)$.

# Local Model Fidelity

- A model $g$ is locally faithful to $\hat{f}$ w.r.t. a point $x$ if for points $z \in \mathcal{Z} \subseteq \mathbb{R}^p$ close to $x$, the predictions of $g(z)$ are close to $\hat{f}(z)$.
- In an optimization task: the closer $z$ is to $x$, the closer $g(z)$ should be to $\hat{f}(z)$.
- For this definition we need two measures:
  1. A proximity measure $n(z)$ between $z$ and $x$, e.g. the exponential kernel

$$n(z) = exp(-d(x, z)^2/\sigma^2)$$

     with $\sigma$ as the kernel width. $d$ could be for example the Euclidean distance for numeric features.

# Local Model Fidelity

- A model $g$ is locally faithful to $\hat{f}$ w.r.t. a point $x$ if for points $z \in \mathcal{Z} \subseteq \mathbb{R}^p$ close to $x$, the predictions of $g(z)$ are close to $\hat{f}(z)$.
- In an optimization task: the closer $z$ is to $x$, the closer $g(z)$ should be to $\hat{f}(z)$.
- For this definition we need two measures:
  1. A proximity measure $n(z)$ between $z$ and $x$, e.g. the exponential kernel

     $$n(z) = exp(-d(x,z)^2/\sigma^2)$$

     with $\sigma$ as the kernel width. $d$ could be for example the Euclidean distance for numeric features.
  2. A distance measure or loss function $L(\hat{f}(z), g(z))$ to assess how close the predictions of $\hat{f}(z)$ and $g(z)$ are, e.g. the L2 loss/squared error

     $$L(\hat{f}(z), g(z)) = (g(z) - \hat{f}(z))^2.$$

# Local Model Fidelity

- A model $g$ is locally faithful to $\hat{f}$ w.r.t. a point $x$ if for points $z \in \mathcal{Z} \subseteq \mathbb{R}^p$ close to $x$, the predictions of $g(z)$ are close to $\hat{f}(z)$.
- In an optimization task: the closer $z$ is to $x$, the closer $g(z)$ should be to $\hat{f}(z)$.
- For this definition we need two measures:
  1. A proximity measure $n(z)$ between $z$ and $x$, e.g. the exponential kernel

     $$n(z) = exp(-d(x,z)^2/\sigma^2)$$

     with $\sigma$ as the kernel width. $d$ could be for example the Euclidean distance for numeric features.
  2. A distance measure or loss function $L(\hat{f}(z), g(z))$ to assess how close the predictions of $\hat{f}(z)$ and $g(z)$ are, e.g. the L2 loss/squared error

     $$L(\hat{f}(z), g(z)) = (g(z) - \hat{f}(z))^2.$$

- Given points $z$, we can measure local fidelity of $g$ with respect to $\hat{f}$ in terms of a weighted loss

$$L(\hat{f}, g, n) = \sum_{z \in \mathcal{Z}} n(z) L(\hat{f}(z), g(z)) \tag{1}$$

# Minimization task



- Formally, an explanation produced by LIME is obtained by the following:

$$\arg\min_{g \in \mathcal{G}} L(\hat{f}, g, n) + J(g)$$

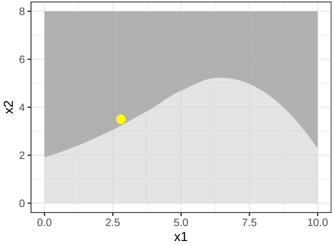
- In practice, LIME only optimizes $L(\hat{f}, g, n)$ (model-fidelity).
- The model complexity ($J(g)$) is determined by users beforehand by restricting the class $\mathcal{G}$. For example, users could only consider sparse linear models.
- Since we want a **model-agnostic** explainer, we need to optimize $L(\hat{f}, g, n)$ without making any assumptions about $\hat{f}$.
- Therefore, we learn $g$ only approximately with the following algorithm.

# LIME Algorithm [Ribeiro et al. 2016]

For the algorithm, we need a pre-trained model $\hat{f}$, $x$ whose prediction we want to explain and model class $\mathcal{G}$.

We illustrate the steps of the algorithm with a classification example:

- The light/dark background represents the prediction surface of a classifier $\hat{f} : \mathbb{R}^2 \to \{0, 1\}$.
- The yellow point displays $x$ we are interested in.
- $\mathcal{G}$ is restricted to the class of logistic regression models ($\to$ classification).



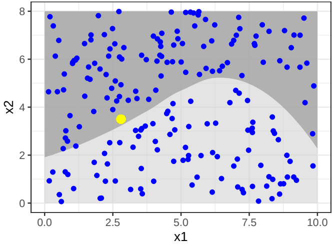
# LIME Algorithm [Ribeiro et al. 2016]

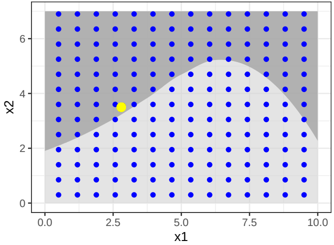


1. Independently sample new points $z \in \mathcal{Z}$.
2. Retrieve predictions $\hat{f}(z)$ for obtained points $z$.

Strategies for sampling:

- Uniformly sample new points from the feasible feature range.
- Use the training data set with or without perturbations.
- Draw samples from the estimated univariate distribution of each feature.
- Create an equidistant grid over the supported feature range.



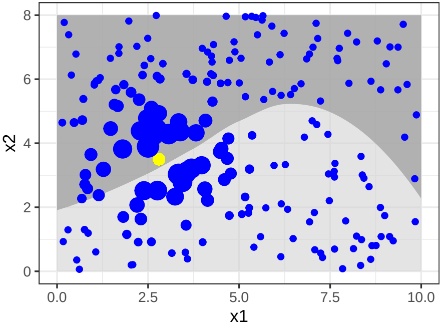




3. Weight $z \in \mathcal{Z}$ by their proximity $\pi(z)$.

# LIME Algorithm [Ribeiro et al. 2016]

In this example, we use the exponential kernel defined on the Euclidean distance $d$

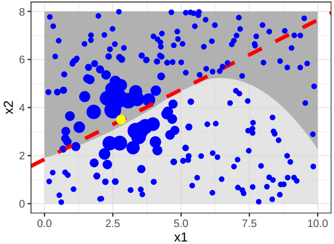
$$n(z) = exp(-d(x,z)^2/\sigma^2).$$



# LIME Algorithm [Ribeiro et al. 2016]

4. Train an interpretable model $g$ on weighted data points $z \in \mathcal{Z}$. The obtained predictions $\hat{f}(z)$ is the target of this model.
5. Return the interpretable model $g$ as the explainer.

Popular interpretable models are linear models, LASSO, classification/regression trees, decision rules.
In our example, we fit a logistic regression model.
Consequently, $L(\hat{f}(z), g(z))$ is the Bernoulli loss in Eq. (1).



# SHAP: Local Feature Importance via Shapley Values

# Example

- Exemplary model for predicting apartment price
- Inputs
  - Size ($m^2$)
  - location (e.g., floor)
  - close by (e.g., train station)
  - pets allowed?
- Average apartment price is 310k

# Example

- Exemplary model for predicting apartment price
- Inputs
  - Size ($m^2$)
  - location (e.g., floor)
  - close by (e.g., train station)
  - pets allowed?
- Average apartment price is 310k

- Query: $50m^2$, 2nd floor, park close by, pets not allowed
- Prediction: 300k

$\rightsquigarrow$ Why?

# Example (cont'd)

- Query: 50m$^2$, 2nd floor, park close by, pets not allowed
- Prediction: 300k

$\rightsquigarrow$ Why?

- Possible approaches: linear models or LIME
- New question: Under all possible subsets of features, how much would the set of features $S$ contribute to the prediction?

# Shapley Values



- Origin in game theory [Shapley 1953]
  - How much does a player contributes to the total payout (i.e., gain) under different coalitions with other players?

# Shapley Values

- Origin in game theory [Shapley 1953]
  - How much does a player contributes to the total payout (i.e., gain) under different coalitions with other players?
- Relation to iML:
  - Game: Prediction task
  - Players: Features values of the instance
  - Gain: Difference between actual prediction and average prediction $\rightsquigarrow$ -10k in our example

# Shapley Values

- Origin in game theory [Shapley 1953]
  - How much does a player contributes to the total payout (i.e., gain) under different coalitions with other players?
- Relation to iML:
  - Game: Prediction task
  - Players: Features values of the instance
  - Gain: Difference between actual prediction and average prediction $\rightsquigarrow$ -10k in our example
- The answer could be:
  - park close by $\rightsquigarrow$ 30k
  - 50m$^2$ $\rightsquigarrow$ 10k
  - 2nd floor $\rightsquigarrow$ 0k
  - pets banned $\rightsquigarrow$ -50k

# Shapley Values

- Origin in game theory [Shapley 1953]
  - How much does a player contributes to the total payout (i.e., gain) under different coalitions with other players?
- Relation to iML:
  - Game: Prediction task
  - Players: Features values of the instance
  - Gain: Difference between actual prediction and average prediction $\rightsquigarrow$ -10k in our example
- The answer could be:
  - park close by $\rightsquigarrow$ 30k
  - 50m$^2$ $\rightsquigarrow$ 10k
  - 2nd floor $\rightsquigarrow$ 0k
  - pets banned $\rightsquigarrow$ -50k

$\rightsquigarrow$ Shapley value: Average marginal contribution of a feature value across all possible coalitions.

# Shapley Values: In a Nutshell

- For all possible subsets of features (i.e., coalition):
  1. add feature value under consideration to the coalition once and once a random value
  2. randomly change features not being in the coalition for both samples
  3. predict difference between both samples

  $\rightsquigarrow$ marginal contribution of feature to coalition

# Shapley Values: Continued example

- Possible coalition for example if we consider pet:
  - $\emptyset$ (no features)
  - size
  - location
  - close-by
  - size + location
  - size + close-by
  - location + close-by
  - size + location + close-by

# Shapley Values: Continued example

- Exemplary coalition: size
- Considered feature value: pets banned
- considered, randomly sampled feature vectors:
  1. size:50, pets:banned, location: 3rd, close-by:train-station
     size:50, pets:allowed, location: 3rd, close-by:train-station
  2. size:50, pets:banned, location: 1st close-by:park
     size:50, pets:allowed, location: 1st, close-by:park
  3. ...

# Cooperative Games

- In cooperative games, a set of players $P$ with $P = \{1, \ldots, p\}$ forms a coalition $S \subseteq P$.

# Cooperative Games



- In cooperative games, a set of players $P$ with $P = \{1, \dots, p\}$ forms a coalition $S \subseteq P$.
- A value function $v(S) : 2^{|P|} \mapsto \mathbb{R}$ describes the payout (or gain) achieved by any coalition $\forall S \subseteq P$. The value of the empty coalition must be zero: $v(\emptyset) = 0$.

# Cooperative Games

- In cooperative games, a set of players $P$ with $P = \{1, \dots, p\}$ forms a coalition $S \subseteq P$.
- A value function $v(S) : 2^{|P|} \mapsto \mathbb{R}$ describes the payout (or gain) achieved by any coalition $\forall S \subseteq P$. The value of the empty coalition must be zero: $v(\emptyset) = 0$.
- As some players contribute more than others, we are interested to fairly divide the total payout $v(P)$ among the players.

# Cooperative Games



- In cooperative games, a set of players $P$ with $P = \{1, \dots, p\}$ forms a coalition $S \subseteq P$.
- A value function $v(S) : 2^{|P|} \mapsto \mathbb{R}$ describes the payout (or gain) achieved by any coalition $\forall S \subseteq P$. The value of the empty coalition must be zero: $v(\emptyset) = 0$.
- As some players contribute more than others, we are interested to fairly divide the total payout $v(P)$ among the players.
- We call the payout per player $\phi_j$, $j \in P$.

# Cooperative Games

- In cooperative games, a set of players $P$ with $P = \{1, \ldots, p\}$ forms a coalition $S \subseteq P$.
- A value function $v(S) : 2^{|P|} \mapsto \mathbb{R}$ describes the payout (or gain) achieved by any coalition $\forall S \subseteq P$. The value of the empty coalition must be zero: $v(\emptyset) = 0$.
- As some players contribute more than others, we are interested to fairly divide the total payout $v(P)$ among the players.
- We call the payout per player $\phi_j$, $j \in P$.

$\rightsquigarrow$ What would be properties of a fair distribution of the payout?

# Axioms of Fair Payouts

One possibility to define **fair** payouts are the following axioms for a given value function $v$:

- **Efficiency**: Player contributions add up to the total payout of the game: $\sum_{j=1}^{p} \phi_j = v(P)$

# Axioms of Fair Payouts

One possibility to define **fair** payouts are the following axioms for a given value function $v$:

- **Efficiency**: Player contributions add up to the total payout of the game: $\sum_{j=1}^{p} \phi_j = v(P)$
- **Symmetry**: Players $j \in P$ and $k \in P$ who contribute the same to any coalition get the same payout:
  If $v(S \cup \{j\}) = v(S \cup \{k\})$ for all $S \subseteq P - \{j, k\}$, then $\phi_j = \phi_k$

# Axioms of Fair Payouts

One possibility to define **fair** payouts are the following axioms for a given value function $v$:

- **Efficiency**: Player contributions add up to the total payout of the game: $\sum_{j=1}^{p} \phi_j = v(P)$
- **Symmetry**: Players $j \in P$ and $k \in P$ who contribute the same to any coalition get the same payout:
  If $v(S \cup \{j\}) = v(S \cup \{k\})$ for all $S \subseteq P - \{j, k\}$, then $\phi_j = \phi_k$
- **Dummy / Null Player**: The payout is zero for players who don't contribute to the value of any coalition:
  If $v(S \cup \{j\}) = v(S) \quad \forall \quad S \subseteq P - \{j\}$, then $\phi_j = 0$

# Axions of Fair Payouts

One possibility to define **fair** payouts are the following axioms for a given value function $v$:

- **Efficiency**: Player contributions add up to the total payout of the game: $\sum_{j=1}^{p} \phi_j = v(P)$
- **Symmetry**: Players $j \in P$ and $k \in P$ who contribute the same to any coalition get the same payout:
  If $v(S \cup \{j\}) = v(S \cup \{k\})$ for all $S \subseteq P - \{j, k\}$, then $\phi_j = \phi_k$
- **Dummy / Null Player**: The payout is zero for players who don't contribute to the value of any coalition:
  If $v(S \cup \{j\}) = v(S) \quad \forall \quad S \subseteq P - \{j\}$, then $\phi_j = 0$
- **Additivity**: For a game $v$ with combined payouts $v(S) = v_1(S) + v_2(S)$, the payout is the sum of payouts: $\phi_{j,v} = \phi_{j,v_1} + \phi_{j,v_2}$

# Axioms of Fair Payouts



One possibility to define **fair** payouts are the following axioms for a given value function $v$:

- **Efficiency**: Player contributions add up to the total payout of the game: $\sum_{j=1}^{p} \phi_j = v(P)$
- **Symmetry**: Players $j \in P$ and $k \in P$ who contribute the same to any coalition get the same payout:
  If $v(S \cup \{j\}) = v(S \cup \{k\})$ for all $S \subseteq P - \{j, k\}$, then $\phi_j = \phi_k$
- **Dummy / Null Player**: The payout is zero for players who don't contribute to the value of any coalition:
  If $v(S \cup \{j\}) = v(S) \quad \forall \quad S \subseteq P - \{j\}$, then $\phi_j = 0$
- **Additivity**: For a game $v$ with combined payouts $v(S) = v_1(S) + v_2(S)$, the payout is the sum of payouts: $\phi_{j,v} = \phi_{j,v_1} + \phi_{j,v_2}$

Is there an attribution formula that adheres to all these axioms?

# Shapley Values

Shapley values solve the attribution problem and provide a unique solution given axioms of efficiency, symmetry, dummy and additivity.

- Shapley values were proposed by Lloyd Shapley in 1951 for cooperative games (game theory).
- The **Shapley value** assigns a value to each player according to the marginal contribution of each player in all possible coalitions.

# Shapley Values

Shapley values solve the attribution problem and provide a unique solution given axioms of efficiency, symmetry, dummy and additivity.

- Shapley values were proposed by Lloyd Shapley in 1951 for cooperative games (game theory).
- The **Shapley value** assigns a value to each player according to the marginal contribution of each player in all possible coalitions.
- $\phi_j = \sum_{S \subseteq P - \{j\}} \frac{|S|!(|P|-|S|-1)!}{|P|!} (v(S \cup \{j\}) - v(S))$
- $v(S \cup \{j\}) - v(S)$ is the marginal contribution of player $j$ to coalition $S$.

# Shapley Values

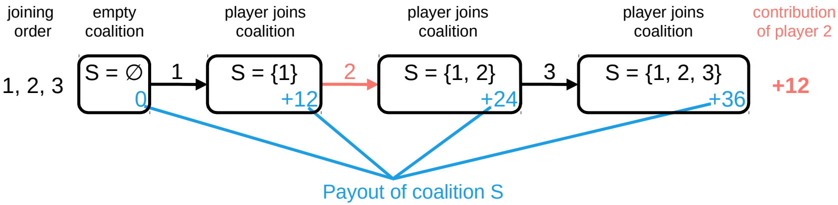
Shapley values solve the attribution problem and provide a unique solution given axioms of efficiency, symmetry, dummy and additivity.

- Shapley values were proposed by Lloyd Shapley in 1951 for cooperative games (game theory).
- The **Shapley value** assigns a value to each player according to the marginal contribution of each player in all possible coalitions.
- $\phi_j = \sum_{S \subseteq P - \{j\}} \frac{|S|!(|P|-|S|-1)!}{|P|!}(v(S \cup \{j\}) - v(S))$
- $v(S \cup \{j\}) - v(S)$ is the marginal contribution of player $j$ to coalition $S$.
- To compute the Shapley payout for a player, we average, for all possible coalitions, how much the player would increase the value of the coalition (=marginal contribution).
- Shapley values are the *only* solution for the attribution with the specified axioms.

Shapley, Lloyd S. (August 21, 1951). "Notes on the n-Person Game – II: The Value of an n-Person Game" (PDF). Santa Monica, Calif.: RAND Corporation.

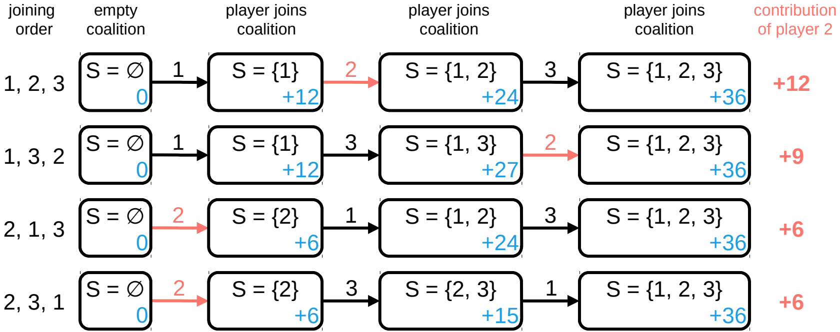
# Shapley Values - Illustration

The Shapley value of a player $j = 2$ is the marginal contribution to the value function when Player $2$ enters an arbitrary coalition.
Here, Player $2$ enters the coalition after Player $1$, resulting in a value change of $v(\{1,2\}) - v(\{1\}) = 24 - 12 = 12$. Overall, the coalition has a value of $v(\{1,2,3\}) = 36$.



# Shapley Values - Illustration

The Shapley value of a player $j = 2$ is the marginal contribution to the value function when Player $2$ enters an arbitrary coalition.
We produce all possible orders of player coalitions and measure the value change if Player $2$ enters the coalition.



# Applications of Shapley Value

- Game theory
- Economics (e.g., cost allocation) [Moulin. 1992]
- Marketing (e.g., social network analysis to discover influencers) [Naraynam et al. 2010]
- ...
- Machine learning
  - Feature selection: Attribute loss reduction to features. [Cohen et al. 2005]
  - Quantify data value: Attribute loss reduction to data points. [Ghorbani et al. 2019]
  - Algorithm Selection: Attribute loss reduction to algorithms of portfolios. [Fréchette et al. 2016]
  - Explain individual predictions $\rightsquigarrow$ local explanation.

# Shapley Values

We can use Shapley values to explain individual predictions $\hat{f}(x^{(i)})$ of a machine learning model $\hat{f}$:

- Players $\hat{=}$ feature values of i-the observation $x_j^{(i)}, j \in \mathcal{P}$.
- Features cooperate to produce a prediction $\hat{f}(x_1^{(i)}, x_2^{(i)}, \dots, x_p^{(i)})$.

# Shapley Values

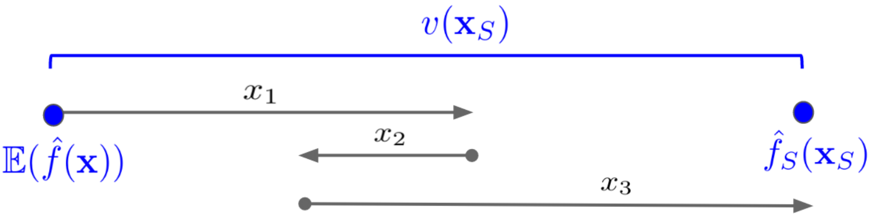
We can use Shapley values to explain individual predictions $\hat{f}(x^{(i)})$ of a machine learning model $\hat{f}$:

- Players $\hat{=}$ feature values of i-the observation $x_j^{(i)}, j \in \mathcal{P}$.
- Features cooperate to produce a prediction $\hat{f}(x_1^{(i)}, x_2^{(i)}, \dots, x_p^{(i)})$.
- The value function / payout of coalition $S$ for observation $x^{(i)}$ is

$$v(S) = \hat{f}_S(x_S^{(i)}) - \mathbb{E}[\hat{f}(x^{(i)})]$$

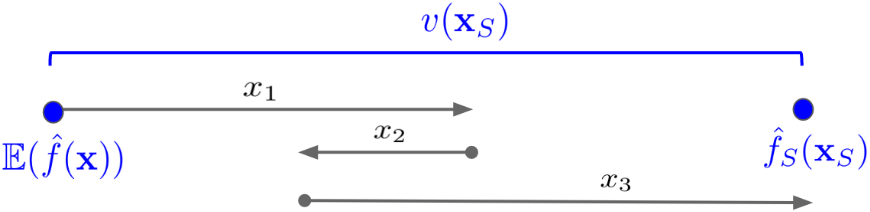
  where $x_S^{(i)} = \{x_j^{(i)}\}_{j \in S}$ and $\hat{f}_S : \mathcal{X}_S \mapsto \mathcal{Y}$.
- The marginal prediction $\hat{f}_S$ is defined as $\hat{f}_S(x_S^{(i)}) := \int_{X_C} \hat{f}(x_S, X_C) dP_{X_C}$
  - Similar as in PDPs.
- Subtraction of $\mathbb{E}[\hat{f}(x^{(i)})]$ to achieve $v(\emptyset) = 0$.
- By using the marginal prediction, we have defined what it means for features to be missing for the prediction: We remove it by integrating over its distribution.

# Shapley Values



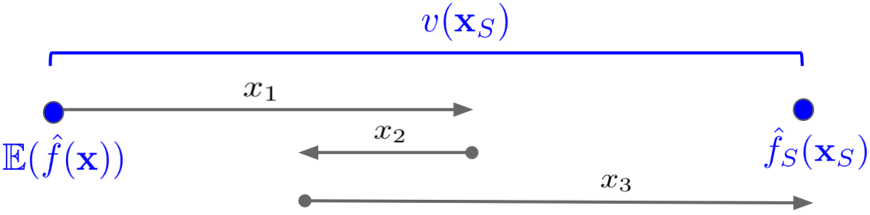

- Shapley values tell us what the payout of each feature is
  - how each feature contributes to the overall prediction of a specific observation.

# Shapley Values




- Shapley values tell us what the payout of each feature is
  - how each feature contributes to the overall prediction of a specific observation.
- The Shapley value is the average marginal contribution of a feature towards the prediction across all possible feature coalitions.

# Shapley Values




- Shapley values tell us what the payout of each feature is
  - how each feature contributes to the overall prediction of a specific observation.
- The Shapley value is the average marginal contribution of a feature towards the prediction across all possible feature coalitions.
- The sum of Shapley values over all features yields the difference between the average prediction of all data points (baseline) and the selected individual prediction.

# Shapley Value - Definition [Strumbelj et al. 2014]

Using the order definition, the Shapley value for feature $j$ and a given data point $x^{(i)}$ can be computed as:

$$\phi_j^{(i)} = \frac{1}{p!} \sum_{S \cup \{j\}} \underbrace{\hat{f}_{S\cup\{j\}}(x_{S\cup\{j\}}) - \hat{f}_S(x_S)}_{\text{marginal contribution of feature } j}$$

- The term $\mathbb{E}[\hat{f}(x)]$ drops due to the subtraction of value functions.
- Interpretation of Shapley value $\phi_j^{(i)}$ for feature $j$ and observation $x^{(i)}$: The feature value $x_j^{(i)}$ contributed $\phi_j^{(i)}$ towards the prediction $\hat{f}(x)$ compared to the average prediction for the dataset.

# Shapley Value - Definition [Strumbelj et al. 2014]

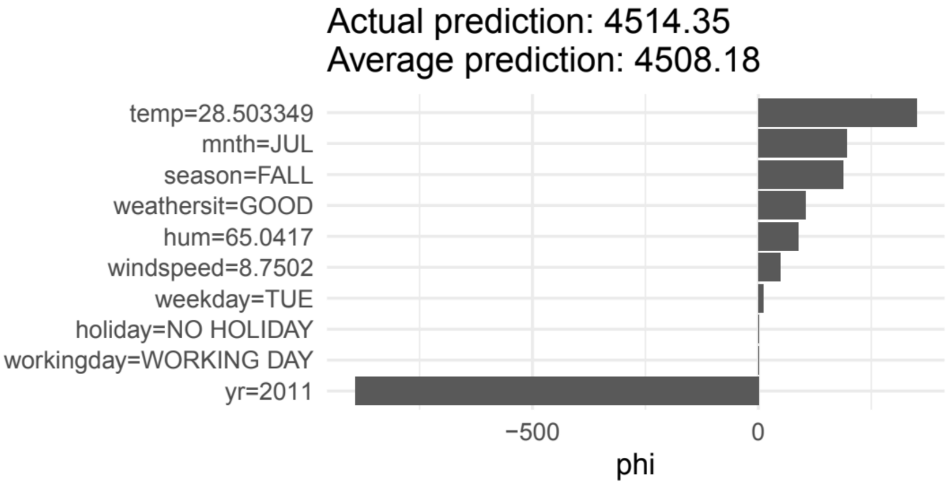
Using the order definition, the Shapley value for feature $j$ and a given data point $x^{(i)}$ can be computed as:

$$\phi_j^{(i)} = \frac{1}{p!} \sum_{S \cup \{j\}} \underbrace{\hat{f}_{S\cup\{j\}}(x_{S\cup\{j\}}) - \hat{f}_S(x_S)}_{\text{marginal contribution of feature } j}$$

- The term $\mathbb{E}[\hat{f}(x)]$ drops due to the subtraction of value functions.
- Interpretation of Shapley value $\phi_j^{(i)}$ for feature $j$ and observation $x^{(i)}$: The feature value $x_j^{(i)}$ contributed $\phi_j^{(i)}$ towards the prediction $\hat{f}(x)$ compared to the average prediction for the dataset.
- Note: Marginal contributions and Shapley values can be negative.

Shapley, Lloyd S. 1953. "A Value for N-Person Games."

# Bike Sharing Dataset



# Explainable AutoML

# Problem of Hyperparameters

- ML algorithms have important hyperparameters
  - learning rate, regularization strength, complexity of the model, ...
- Depending on your hyperparameter configuration, your model might learn only noise, will learn a constant model or maybe even achieve state-of-the-art performance

# Problem of Hyperparameters

- ML algorithms have important hyperparameters
  - learning rate, regularization strength, complexity of the model, ...
- Depending on your hyperparameter configuration, your model might learn only noise, will learn a constant model or maybe even achieve state-of-the-art performance
- Manual hyperparameter optimization?
  - tedious, error-prone, time-consuming
  - Requires years of expertise

# Problem of Hyperparameters

- ML algorithms have important hyperparameters
  - learning rate, regularization strength, complexity of the model, ...
- Depending on your hyperparameter configuration, your model might learn only noise, will learn a constant model or maybe even achieve state-of-the-art performance
- Manual hyperparameter optimization?
  - tedious, error-prone, time-consuming
  - Requires years of expertise
- Simply choosing default settings?
  - Developers of algorithms often tuned the hyperparameters on a limited set of datasets

  $\rightsquigarrow$ might work well on your dataset or might fail

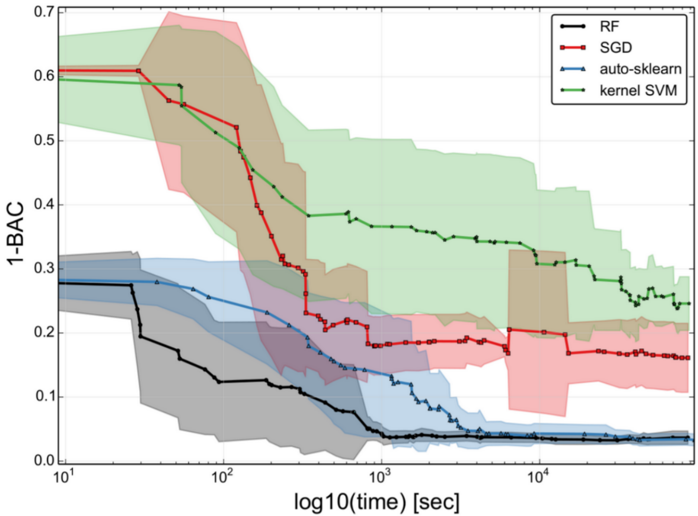
# Problem of Hyperparameters

- ML algorithms have important hyperparameters
  - learning rate, regularization strength, complexity of the model, ...
- Depending on your hyperparameter configuration, your model might learn only noise, will learn a constant model or maybe even achieve state-of-the-art performance
- Manual hyperparameter optimization?
  - tedious, error-prone, time-consuming
  - Requires years of expertise
- Simply choosing default settings?
  - Developers of algorithms often tuned the hyperparameters on a limited set of datasets
  - $\rightsquigarrow$ might work well on your dataset or might fail

$\rightsquigarrow$ Can we automate this hyperparameter optimization (HPO)?

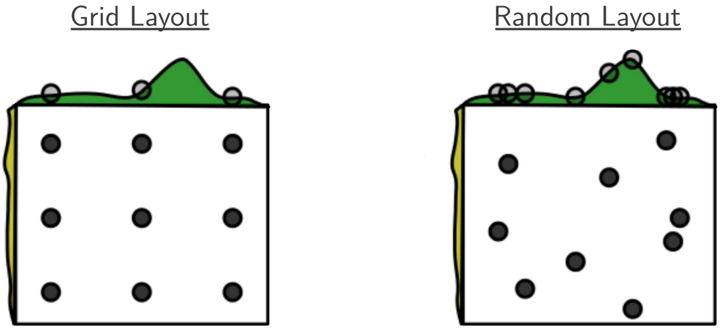
# Effect of Hyperparameter Optimization





# Grid Search vs. Random Search

- Let's tune 2 hyperparameters
- Plotting the samples of hyperparameter configurations for grid search and random search
- On each axis, we see the effect of each hyperparameter on the validation loss




[Bergstra and Bengio 2012]

# Global Black-Box Optimization

- Consider the global optimization problem of finding:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} f(\boldsymbol{\lambda})$$

# Global Black-Box Optimization

- Consider the global optimization problem of finding:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} f(\boldsymbol{\lambda})$$

- In the most general form, function $f$ is a blackbox function:

$$\boldsymbol{\lambda} \longrightarrow \blacksquare \longrightarrow f(\boldsymbol{\lambda})$$

  - Only mode of interaction with $f$: querying $f$'s value at a given $\boldsymbol{\lambda}$
  - Function $f$ may not be available in closed form, not convex, not differentiable, noisy, etc.

# Global Black-Box Optimization

- Consider the global optimization problem of finding:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} f(\boldsymbol{\lambda})$$

- In the most general form, function $f$ is a blackbox function:

$$\boldsymbol{\lambda} \longrightarrow \blacksquare \longrightarrow f(\boldsymbol{\lambda})$$

  - Only mode of interaction with $f$: querying $f$'s value at a given $\boldsymbol{\lambda}$
  - Function $f$ may not be available in closed form, not convex, not differentiable, noisy, etc.

- We'll discuss a Bayesian approach for solving such blackbox optimization problems

# Global Black-Box Optimization

- Consider the global optimization problem of finding:

$$\boldsymbol{\lambda}^* \in \underset{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}}{\arg\min}\, f(\boldsymbol{\lambda})$$

- In the most general form, function $f$ is a blackbox function:

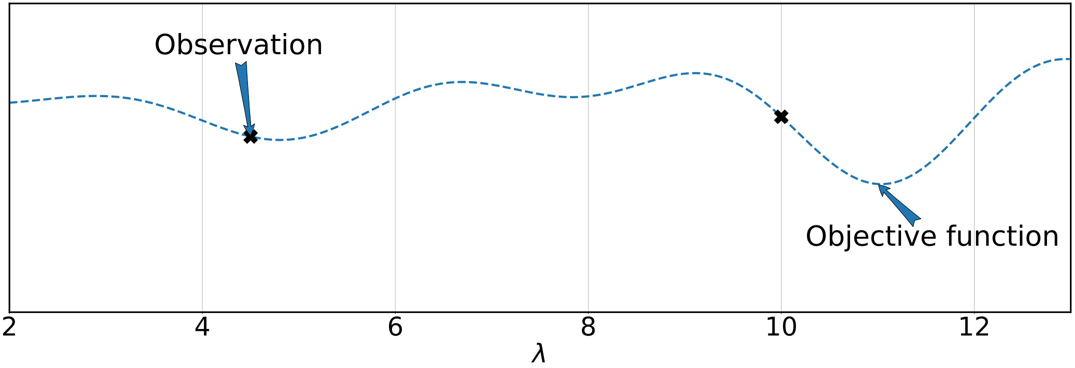
$$\boldsymbol{\lambda} \longrightarrow \blacksquare \longrightarrow f(\boldsymbol{\lambda})$$

  - Only mode of interaction with $f$: querying $f$'s value at a given $\boldsymbol{\lambda}$
  - Function $f$ may not be available in closed form, not convex, not differentiable, noisy, etc.

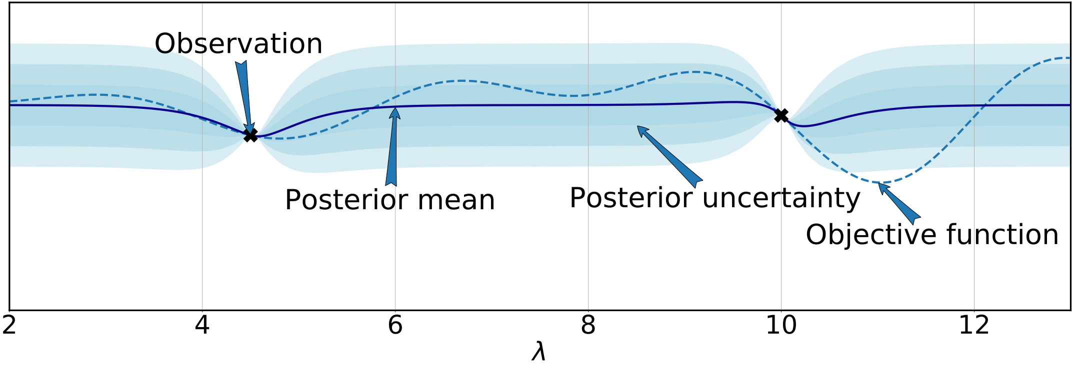
- We'll discuss a Bayesian approach for solving such blackbox optimization problems

- Blackbox optimization can be used for hyperparameter optimization (HPO)
  - Define $f(\boldsymbol{\lambda}) := \mathcal{L}(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$

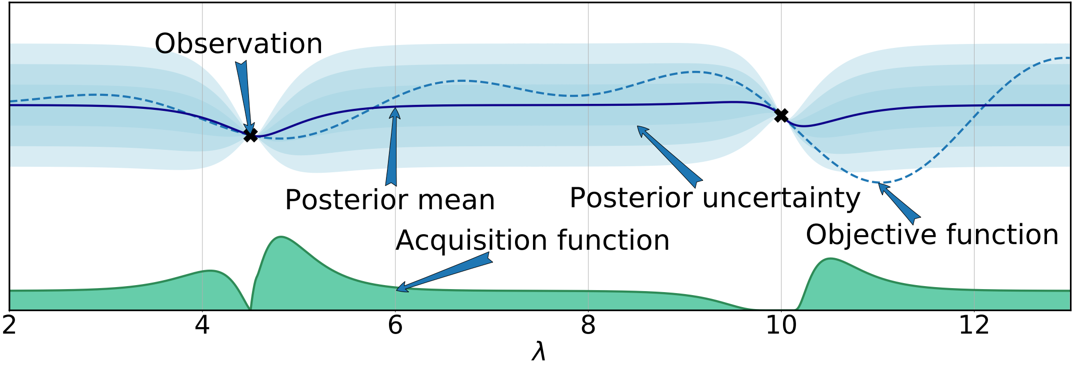
# Bayesian Optimization of a Blackbox Function in a Nutshell



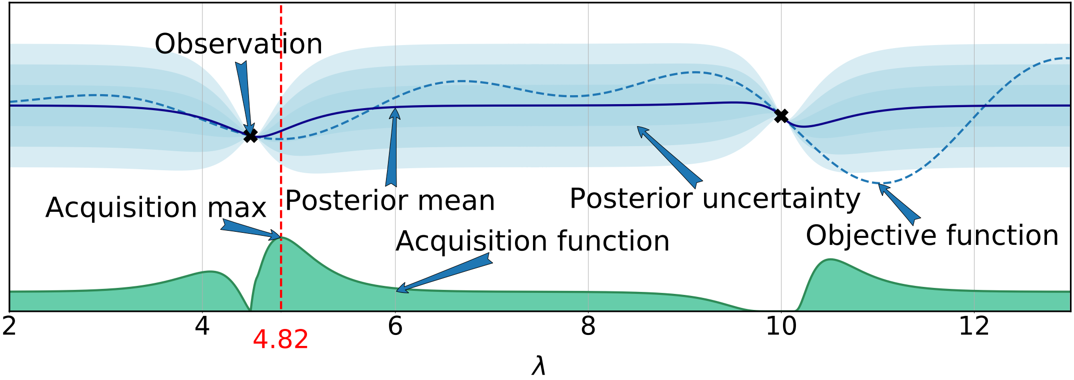
# Bayesian Optimization of a Blackbox Function in a Nutshell



# Bayesian Optimization of a Blackbox Function in a Nutshell



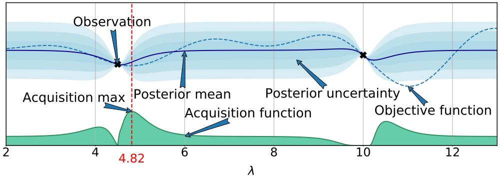
# Bayesian Optimization of a Blackbox Function in a Nutshell



# Bayesian Optimization of a Blackbox Function in a Nutshell

General approach

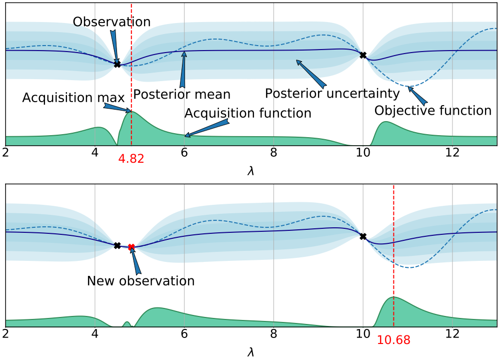
- Fit a probabilistic model to the collected function samples $\langle \boldsymbol{\lambda}, c(\boldsymbol{\lambda}) \rangle$
- Use the model to guide optimization, trading off exploration *vs* exploitation



# Bayesian Optimization of a Blackbox Function in a Nutshell

General approach

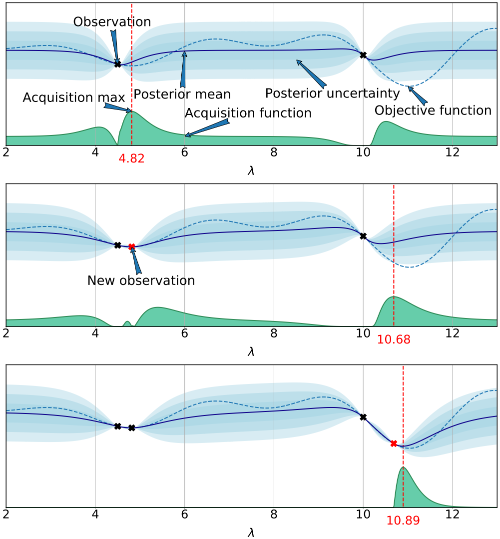
- Fit a probabilistic model to the collected function samples $\langle \boldsymbol{\lambda}, c(\boldsymbol{\lambda}) \rangle$
- Use the model to guide optimization, trading off exploration *vs* exploitation



# Bayesian Optimization of a Blackbox Function in a Nutshell

General approach

- Fit a probabilistic model to the collected function samples $\langle \boldsymbol{\lambda}, c(\boldsymbol{\lambda}) \rangle$
- Use the model to guide optimization, trading off exploration *vs* exploitation



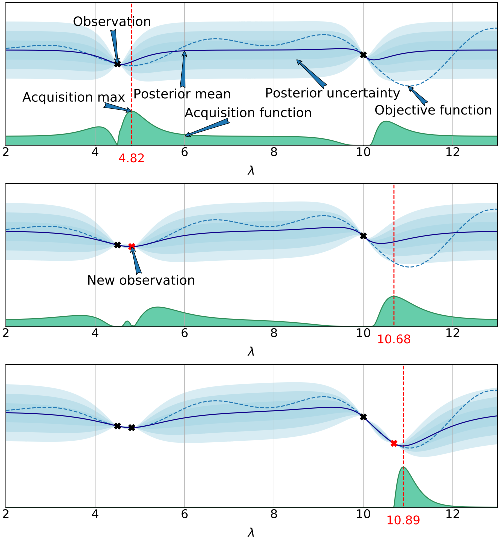
# Bayesian Optimization of a Blackbox Function in a Nutshell

General approach

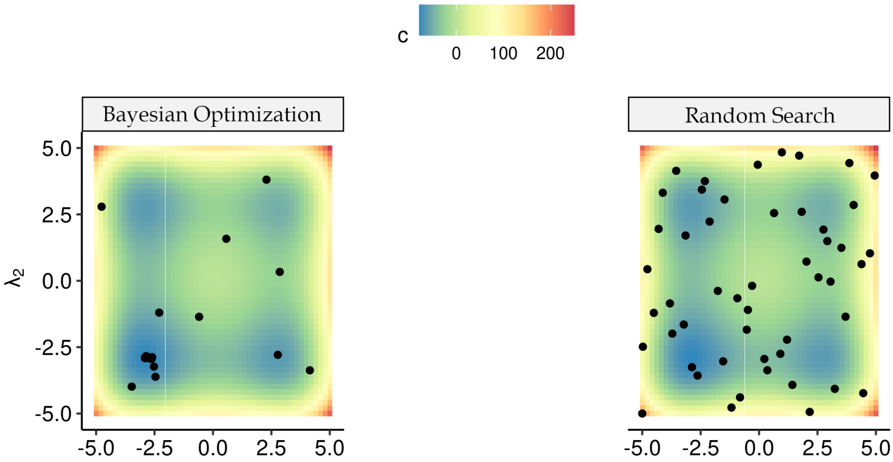
- Fit a probabilistic model to the collected function samples $\langle \boldsymbol{\lambda}, c(\boldsymbol{\lambda}) \rangle$
- Use the model to guide optimization, trading off exploration *vs* exploitation

Popular approach in the statistics literature since [Mockus. 1975]

- Efficient in #function evaluations
- Works when objective is nonconvex, noisy, has unknown derivatives, etc.
- Convergence results [Srinivas et al. 2009]; [Bull. 2011] [de Freitas et al. 2012]; [Kawaguchi et al. 2015]

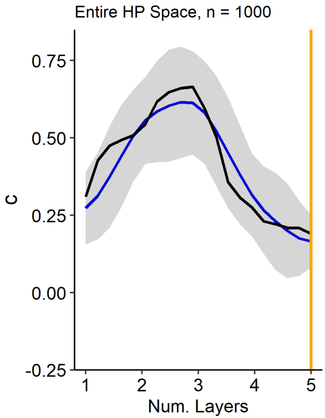


# Bayesian Optimization vs Random Search



# xAutoML via Effects of Hyperparameters





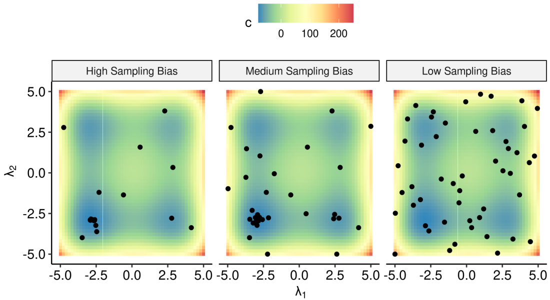

- How does changing one of the hyperparameters affect the performance ($c$) of the ML model?

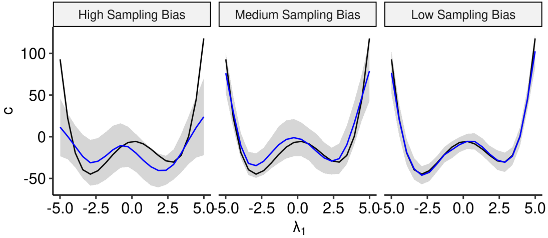
← PDPs on the model of Bayesian Optimization can be used for these questions

- Can be extended to two hyperparameters and their interaction effects
- other iML techniques (such as LIME and SHAP) can also be aplied

⇝ improves understanding and trust into AutoML

# BUT: Sampling Distribution of Data




- High sampling bias ⇝ Good optimization performance ⇝ Poor PDP estimates
- Low sampling bias ⇝ Good PDP estimates ⇝ Poor optimization performance

# Solution Approaches

- [Moosbauer et al. 2021] identified regions with confident PDPs
- [Moosbauer et al. 2022] adapts the exploration in Bayesian Optimization to allow for globally well-estimated PDPs

# Summary

# Summary

1. Partial Dependence Plots (PDPs) show you the effect of hyperparameter(s)
2. LIME provide a local explanation of a single prediction
3. SHAP returns the feature importance of a prediction

# Summary

1. Partial Dependence Plots (PDPs) show you the effect of hyperparameter(s)
2. LIME provide a local explanation of a single prediction
3. SHAP returns the feature importance of a prediction

4. iML can be applied to AutoML and HPO
5. But data distribution has to be taken into account