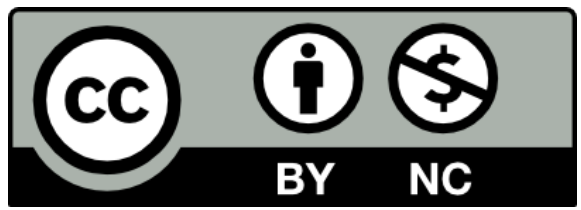
Business 4720 - Class 19

Interpretable Machine Learning – Explainable AI

Joerg Evermann

Faculty of Business Administration
Memorial University of Newfoundland
jevermann@mun.ca



MEMORIAL UNIVERSITY

This Class

What You Will Learn:

- Introduction to Interpetability and Explainability
- Model specific and Model agnostic methods
- Global explainability
- Local explainability

MEMORIAL UNIVERSITY

Based On

Molnar, Christoph: *Interpretable Machine Learning* (2023)

`https://christophm.github.io/interpretable-ml-book/`

MEMORIAL UNIVERSITY

Additional Materials

SciKit Learn

A machine learning framework for Python that also provides some interpretable ML functions.

`https://scikit-learn.org/stable/user_guide.html`

LIME

A Python package to compute Local Interpretable Model Explanations (a local model-agnostic method).

`https://github.com/marcotcr/lime`

SHAP

A Python package to compute Shapley Additive Explanations (a local model-agnostic interpretation method).

`https://shap.readthedocs.io/en/latest/`

MEMORIAL UNIVERSITY

Tools

Install required Python packages:

```
pip install statsmodels matplotlib scikit-learn \
    PyALE lime shap
```

MEMORIAL UNIVERSITY

Importance of Interpretability

Human understanding of how the AI works and arrives at its results (decisions, predictions, . . .)

- Curiosity
- Human learning
- Human sensemaking of events and phenomena
- Knowledge extraction for scientific progress
- Safety and compliance assessment
- Reliability and robustness evaluation
- Identify knowledge limits
- Auditability
- Bias detection & ensuring fairness
- Trust and acceptance
- Debugging & failure analysis
- Legal obligations (”right to explanation”)

MEMORIAL UNIVERSITY

Model Interpretability

Distinctions

- Intrinsic $\leftrightarrow$ Post-hoc
- Local $\leftrightarrow$ Global

MEMORIAL UNIVERSITY

Intrinsically Interpretable Models

Algorithm	Linear	Monotone	Interaction
Linear regression	Yes	Yes	No
Logistic regression	No	Yes	No
Decision trees	No	Some	Yes
RuleFit	Yes	No	Yes
Naive Bayes	No	Yes	No
k-NN	No	No	No

Source:

`https://christophm.github.io/interpretable-ml-book/simple.html`

MEMORIAL UNIVERSITY

Linear Regression

Using R:

```
# Load the bike rental data set
d <- read.csv('https://evermann.ca/busi4720/bike.csv')
# Perform the regression and summarize results
summary(lm(cnt~season+temp, data=d))
```

Results:

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   3151.02     169.35  18.606  < 2e-16 ***
seasonSPRING  -494.15     163.28  -3.026  0.00256 **
seasonSUMMER  -852.68     209.82  -4.064 5.35e-05 ***
seasonWINTER -1342.87     164.59  -8.159 1.49e-15 ***
temp           132.79      11.02  12.046  < 2e-16 ***
---
Residual standard error: 1433 on 726 degrees of freedom
Multiple R-squared: 0.4558, Adjusted R-squared: 0.4528
```

MEMORIAL UNIVERSITY

Linear Regression

- **Algorithmic transparency**: The ordinary least squares loss function is clear and intuitive; provides optimality guarantees
- **Coefficients** β
  - An increase of one unit of a predictor increases the prediction by β, *assuming all other predictors remain the same* (”ceteris paribus”)
  - Switching from the reference category (see ”contrasts”) to another category increases the prediction by β, *assuming all other predictors remain the same* (”ceteris paribus”)
  - **Intercept** is the predicted value when all other predictors are 0. Is this reasonable?
- R^2 is the amount of explained variance; model weights should only be interpreted when R^2 reasonable size.
- **Relative feature importance** is given by the $t = \frac{\hat{\beta}}{SE(\hat{\beta})}$ statistic.

MEMORIAL UNIVERSITY

Linear Regression

Dimension reduction to improve interpretability:

- Manual feature selection, e.g. based on effect size
- Automatic feature selection (forwards or backwards)
- Regression with PCA components
- Penalized regression with LASSO

Be aware of bias-variance trade-off with all of these.

MEMORIAL UNIVERSITY

Decision Trees

Decision Tree Types

- Regression trees
- Classification trees

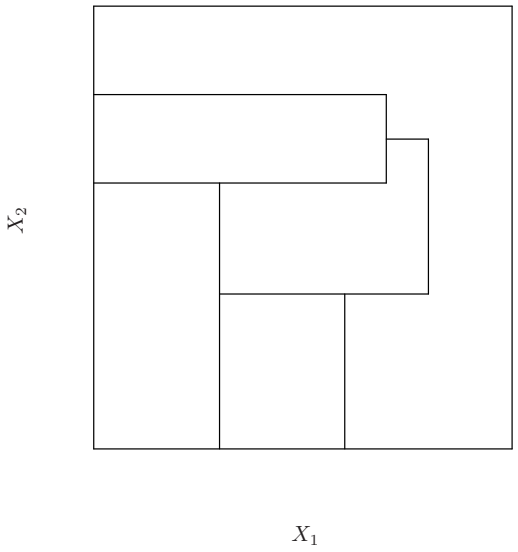
Strengths

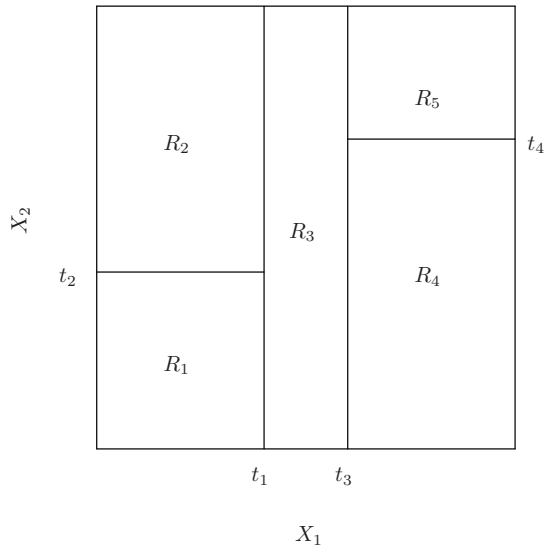
- Intrinsically interpretable and visualizable
- Individual predictions explained by path through tree
- Captures feature interactions
- No need to transform features

Weaknesses

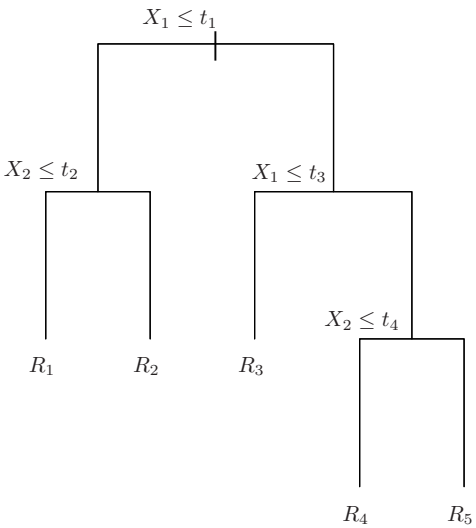
- Unstable (high variance)
- Tend to overfit
- Predictions are piecewise constant

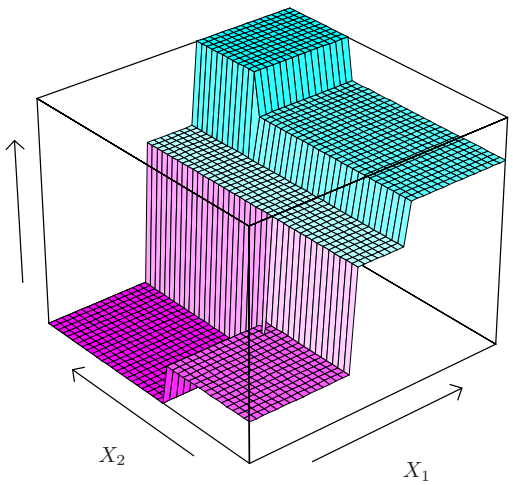
Regression Trees





Source: ISLR2
Figure 8.3





MEMORIAL UNIVERSITY

Regression Trees

1. Recursively divide the predictor space into J distinct and non-overlapping regions $R_1, R_2, \ldots, R_j$
   - For every predictor j and split point s define regions

$$R_1(j,s) = \{X | X_j < s\} \quad \text{and} \quad R_2(j,s) = \{X | X_j \geq s\}$$

   - Choose j and s to minimize variance in each region:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \bar{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \bar{y}_{R_2})^2$$

2. For every observation that falls into region R_j, prediction is the mean of the targets of training observations in R_j

MEMORIAL UNIVERSITY

Regression Trees – Python

Prepare data:

```
import matplotlib.pyplot as plt
import pandas as pd
d=pd.read_csv('https://evermann.ca/busi4720/bike.csv')
x=d[['temp', 'hum']]
y=d['cnt']
```

Fit unpruned tree:

```
from sklearn.tree import DecisionTreeRegressor
regr = DecisionTreeRegressor()
regr.fit(x, y)
```

MEMORIAL UNIVERSITY

Regression Trees – Python

Print the MSE:

```
from sklearn.metrics import mean_squared_error
mean_squared_error(regr.predict(x), y)
```

Print the tree:

```
from sklearn.tree import export_text
print (export_text(regr,feature_names=x.columns))
```

MEMORIAL UNIVERSITY

Regression Trees – Python

Early stopping can prevent overfitting and maintain interpretability.

Maximum depth:

```
regr = DecisionTreeRegressor(max_depth=3).fit(x, y)
```

Minimum samples per leaf:

```
regr = DecisionTreeRegressor(min_samples_leaf=10).fit(x, y)
```

Maximum number of leaf nodes:

```
regr = DecisionTreeRegressor(max_leaf_nodes=8).fit(x, y)
```

MEMORIAL UNIVERSITY

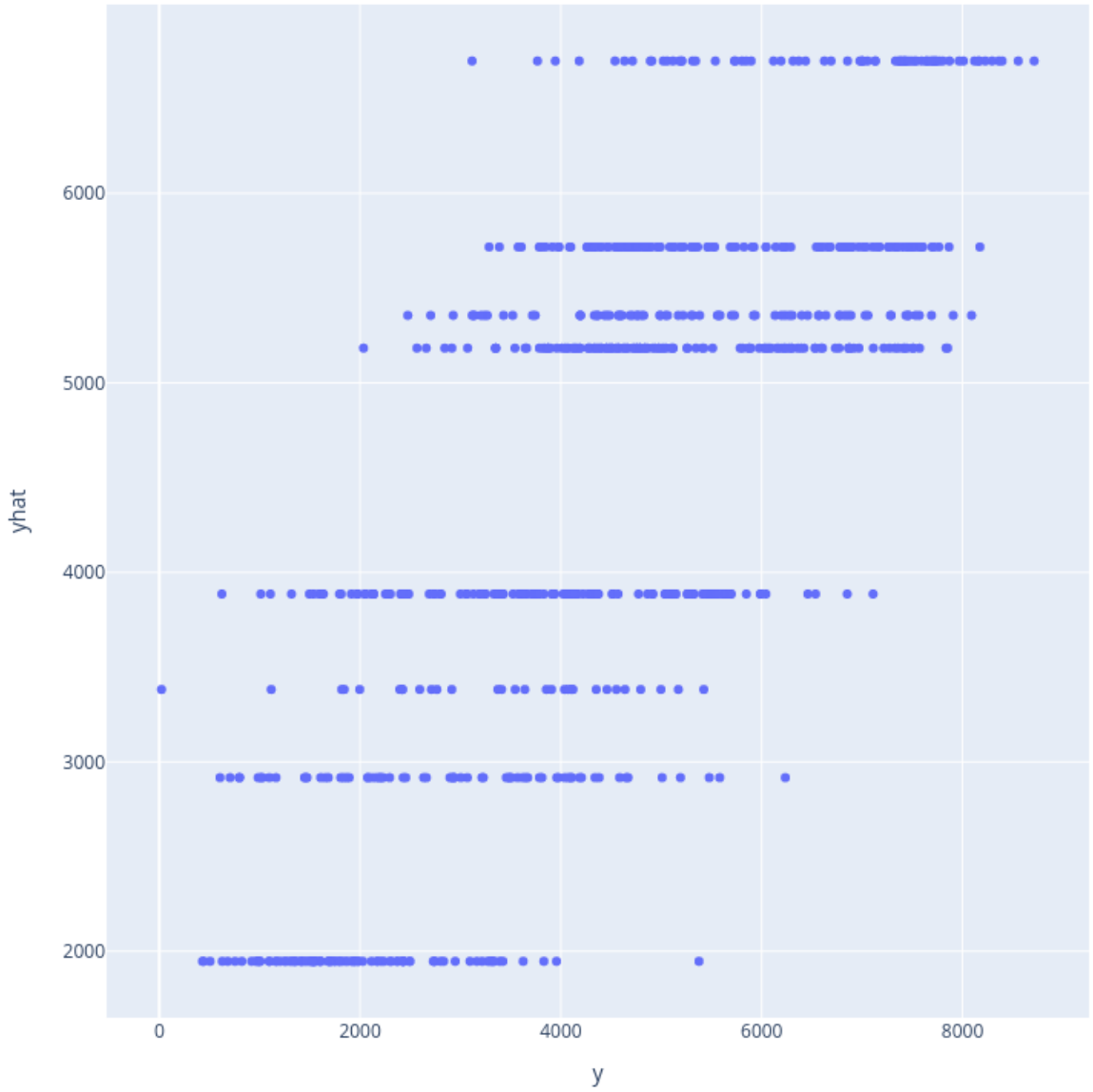
Regression Trees – Python

Plot fitted versus true values:

```
import plotly.express as px
px.scatter(
    pd.DataFrame([y, regr.predict(x)], index=['y', 'yhat']) \
    .transpose(),
    x='y', y='yhat').show()
```

MEMORIAL UNIVERSITY

Regression Trees – Python



Hands-On Exercises

Fit regression trees to the bike rental dataset on the previous slides. Calculate the MSE for each decision tree as you vary:

- **max_depth**: choose values 1, 3, 5, 7
- **min_samples_leaf**: choose values 1, 5, 10, 20
- **max_leaf_nodes**: choose values 2, 8, 16, 32

Which tree provides the best training MSE and what is that training MSE?

MEMORIAL UNIVERSITY

Classification Trees

- Proportion of observations of class k in leaf node m:

$$p_{km}$$

- Predict the most common (majority) class in a leaf node:

$$k(m) = \operatorname*{argmax}_k p_{km}$$

- Probability of choosing an item with label i in node m is

$$p_{im}$$

- Probability of misclassifying that item in node m is

$$1 - p_{im} = \sum_{k \neq i} p_{km}$$

MEMORIAL UNIVERSITY

Classification Trees [cont'd]

- Use **Gini impurity** G (node purity) for node m to determine splits:

$$G_m = \sum_{i=1}^{J} p_{im}(1 - p_{im}) = \sum_{i=1}^{J} (p_{im} - p_{im}^2) = \sum_{i-1}^{J} - \sum_{i=1}^{J} p_{im}^2$$

$$= 1 - \sum_{i=1}^{J} p_{im}^2$$

- Use **Cross-Entropy** H for node m to determine splits.

$$H_m = - \sum_{i=1}^{J} p_{im} \log p_{im}$$

MEMORIAL UNIVERSITY

Further reading:

https://scikit-learn.org/stable/modules/tree.html

https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html

https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html

Global Model Agnostic Methods

- **Partial dependence plot (PDP)**
- **Individual conditional expectation (ICE) curves**
- **Accumulated local effects plot (ALE)**
- Feature interaction
- Functional decomposition
- **Permutation feature importance**
- **Global surrogate models**
- Prototypes

MEMORIAL UNIVERSITY

Partial Dependence Plot (PDP)

Marginal effect of one (or a few) features X_S on the outcome, marginalized (i.e. sum weighted by probability) over all other (complement) features X_C.

$$\hat{f}_S(X_S) = \mathbb{E}_{X_C}\left[\hat{f}(X_S, X_C))\right] = \int \hat{f}(X_S, X_C)p(X_C)dX_c$$

Estimated from sample data as:

$$\hat{f}_S(X_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(X_S, X_C^{(i)})$$

Shows how the *average* prediction changes when the focal predictor is changed (assuming feature independence).

MEMORIAL UNIVERSITY

Partial Dependence Plot (PDP)

Read the data set:

```
import pandas as pd
d=pd.read_csv('https://evermann.ca/busi4720/bike.csv')
x=d[['temp', 'hum']]
y=d[['cnt']]
```

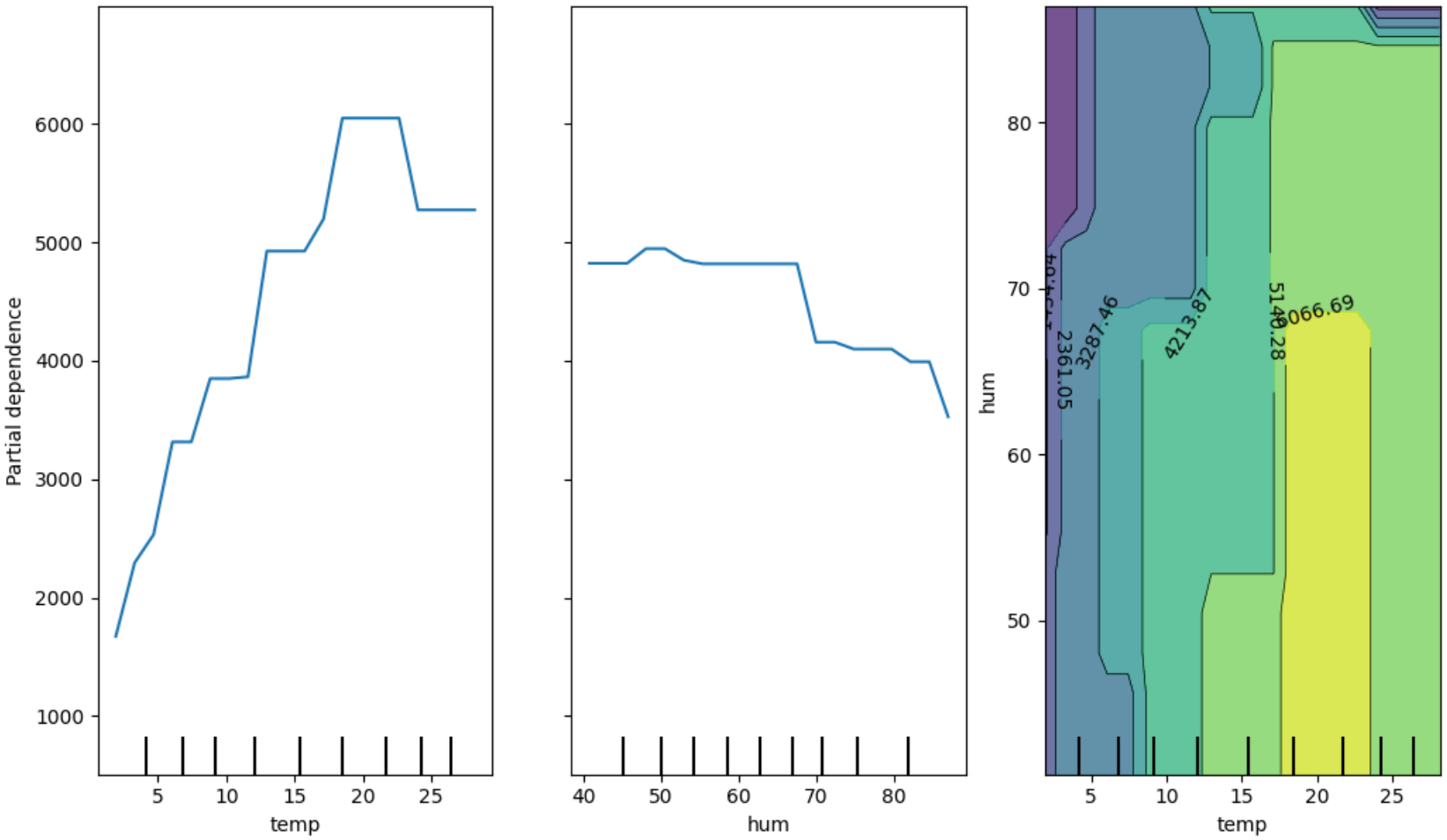
Fit a regression tree:

```
from sklearn.tree import DecisionTreeRegressor
regr = DecisionTreeRegressor(max_depth=5).fit(x, y)
```

Show the PDP:

```
import matplotlib.pyplot as plt
from sklearn.inspection import PartialDependenceDisplay
PartialDependenceDisplay \
    .from_estimator(regr, x, [0, 1, (0,1)],
        grid_resolution=20)
plt.show()
```

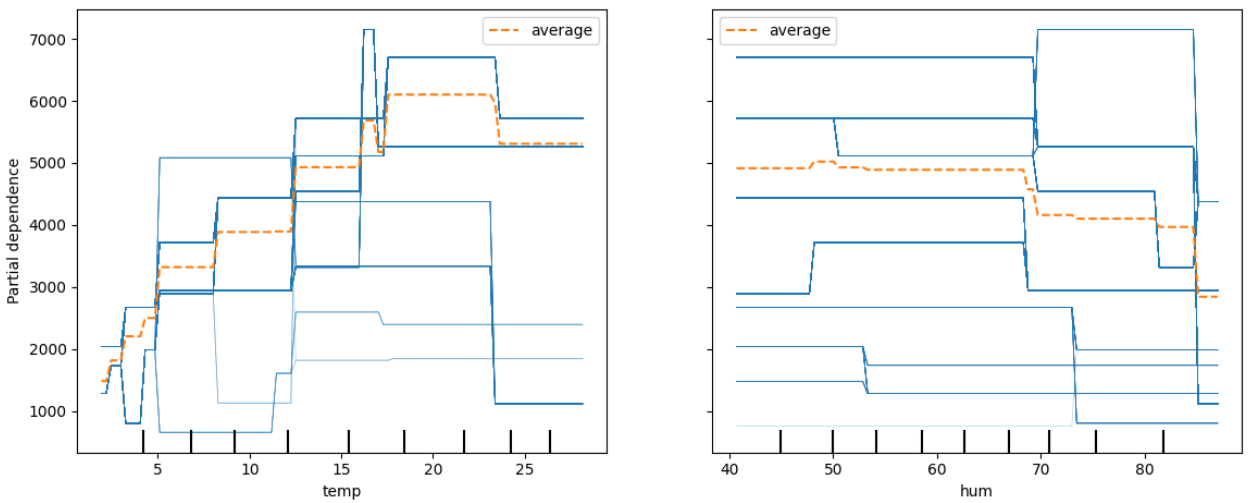
Partial Dependence Plot (PDP)



Individual Conditional Expectation (ICE) Plot

- Instead of the average effect of a feature, shows PDP for individual samples
- Identify individual **outlier** cases or **heterogeneous data**

```
PartialDependenceDisplay \
    .from_estimator(regr, x, [0, 1], kind='both')
```



(Individual samples overlaid due to piece-wise constant regression)

MEMORIAL UNIVERSITY

Accumulated Local Effects (ALE) Plot

- Effects computed for a grid of intervals (a "local window") (instead of the entire domain, as in PDP)
- Does not construct unrealistic feature combinations
- Overcomes the problem of correlated features in PDP
- Focuses on difference in predictions

$$\hat{\tilde{f}}_{j,ALE}(X) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{i:x_j^{(i)} \in N_j(k)} \left[\hat{f}(z_{k,j}, x_j^{(i)}) - \hat{f}(z_{k-1,j}, x_j^{(i)}) \right]$$

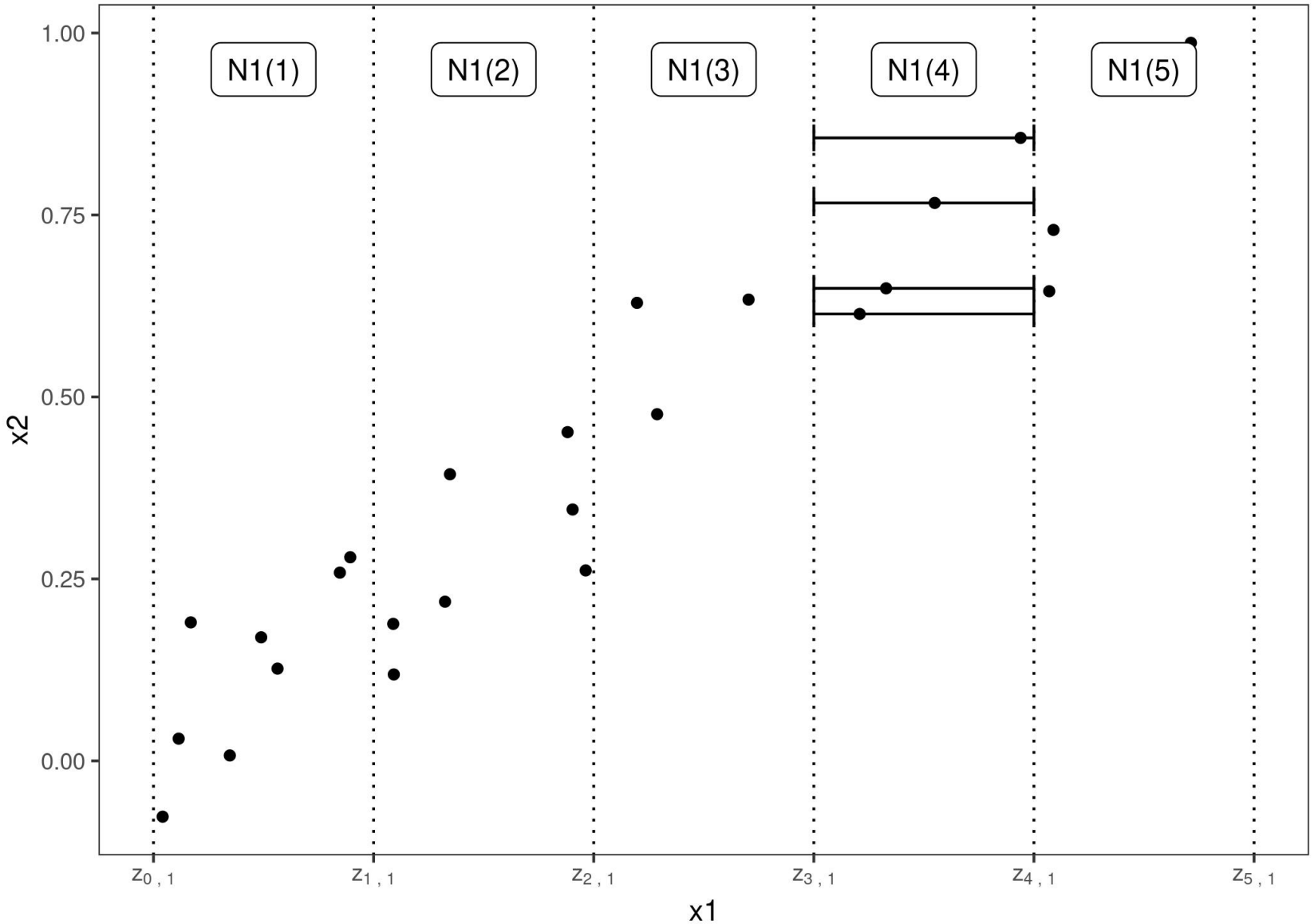
- Difference of predictions (in sq brackets) is *local* to "neighbourhood" $N_j(k)$ of feature j around observation k
- Outer sum *accumulates* the local effects

Centering the effects to mean 0:

$$\hat{f}_{j,ALE}(x) = \hat{\tilde{f}}_{j,ALE}(x) - \frac{1}{n} \sum_{i=1}^{n} \hat{\tilde{f}}_{j,ALE}(x_j^{(i)})$$

MEMORIAL UNIVERSITY

ALE Plots



Source: Molnar, Fig. 8.7

MEMORIAL UNIVERSITY

Accumulated Local Effects (ALE) Plot

Train model:

```
from sklearn.tree import DecisionTreeRegressor
regr=DecisionTreeRegressor(min_samples_leaf=10).fit(x,y)
```

Construct the ALE and plot:

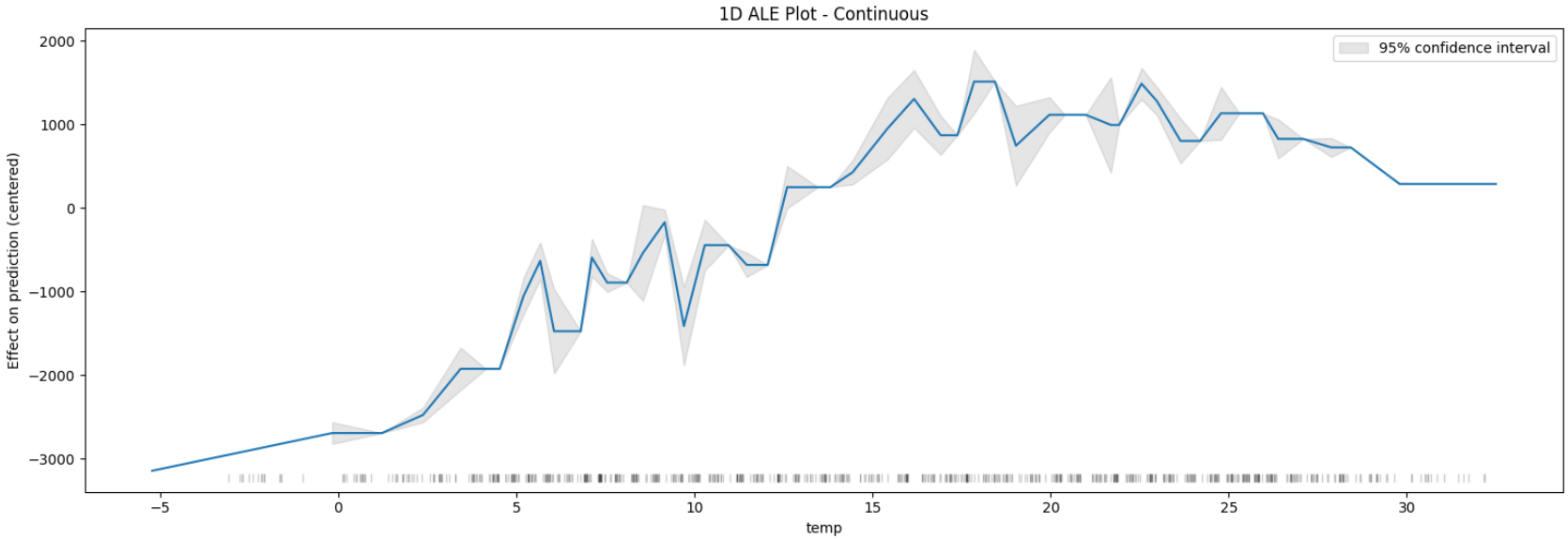
```
import matplotlib.pyplot as plt
from PyALE import ale
ale_effects = ale(X=x, model=regr, \
    feature=['temp'], grid_size=50, include_CI=True)
plt.show()
```

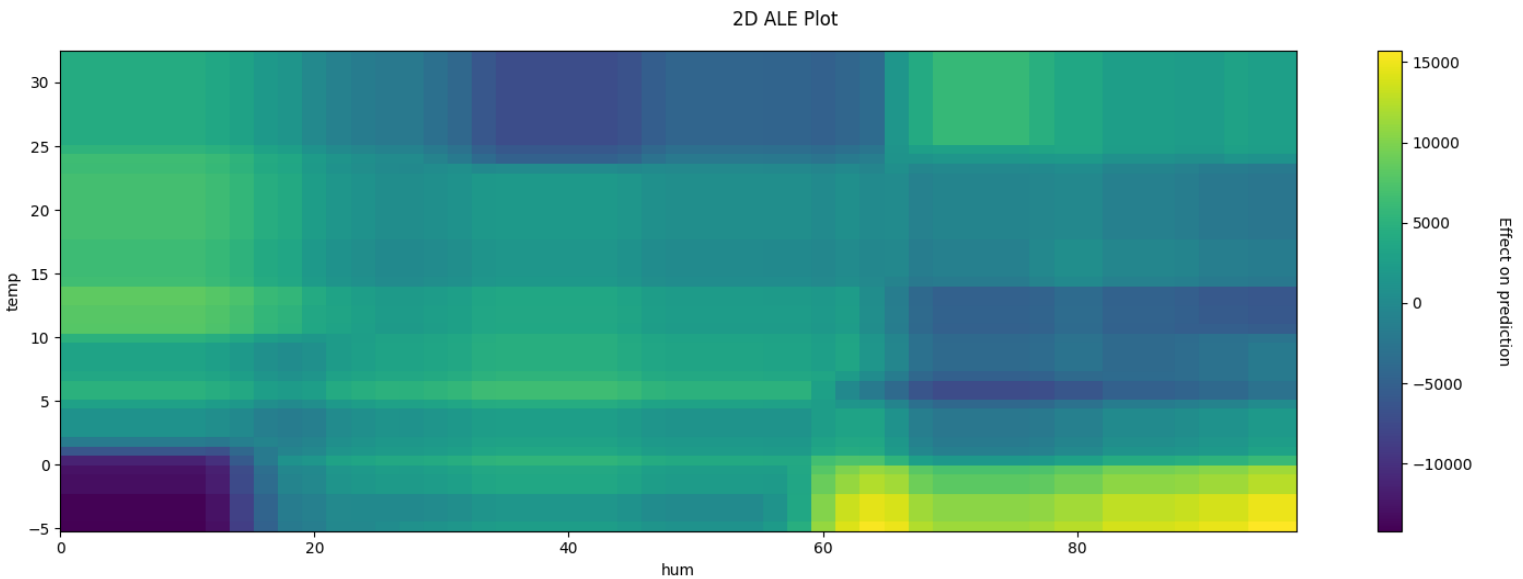
2D feature interactions:

```
ale_effects = ale(X=x, model=regr, \
    feature=['temp', 'hum'], grid_size=50)
plt.show()
```

MEMORIAL UNIVERSITY

Accumulated Local Effects (ALE) Plot





ORIAL UNIVERSITY

Permutation Feature Importance

Intuition

Calculate the increase in a model's prediction error when permuting a feature

1. Estimate model error on original data $e^{\text{orig}} = L(y, \hat{f}(X))$
2. For each feature j:
   - For each repetition k in $1 \cdots K$:
     - Generate $X^{\text{perm}}_{j,k}$ by permuting ("randomly shuffling") values of feature j
     - Estimate $e^{\text{perm}}_{j,k} = L(y, \hat{f}(X^{\text{perm}}_{j,k}))$
   - Calculate permutation feature importance as $i_j = e^{\text{orig}} - \frac{1}{K} \sum_k^K e^{\text{perm}}_{j,k}$

Calculate Permutation Feature Importance on *test* data

MEMORIAL UNIVERSITY

Permutation Feature Importance

Prepare data:

```
import pandas as pd
d=pd.read_csv('https://evermann.ca/busi4720/bike.csv')
x=pd.get_dummies(d.drop(['yr','days_since_2011'],axis=1))
y=x.pop('cnt')
```

Train model:

```
from sklearn.tree import DecisionTreeRegressor
regr=DecisionTreeRegressor(min_samples_leaf=10).fit(x,y)
```

Calculate permutation feature importance and sort them:

```
from sklearn.inspection import permutation_importance
r = permutation_importance(regr, x, y, n_repeats=30)
r_idx = r.importances_mean.argsort()
```

MEMORIAL UNIVERSITY

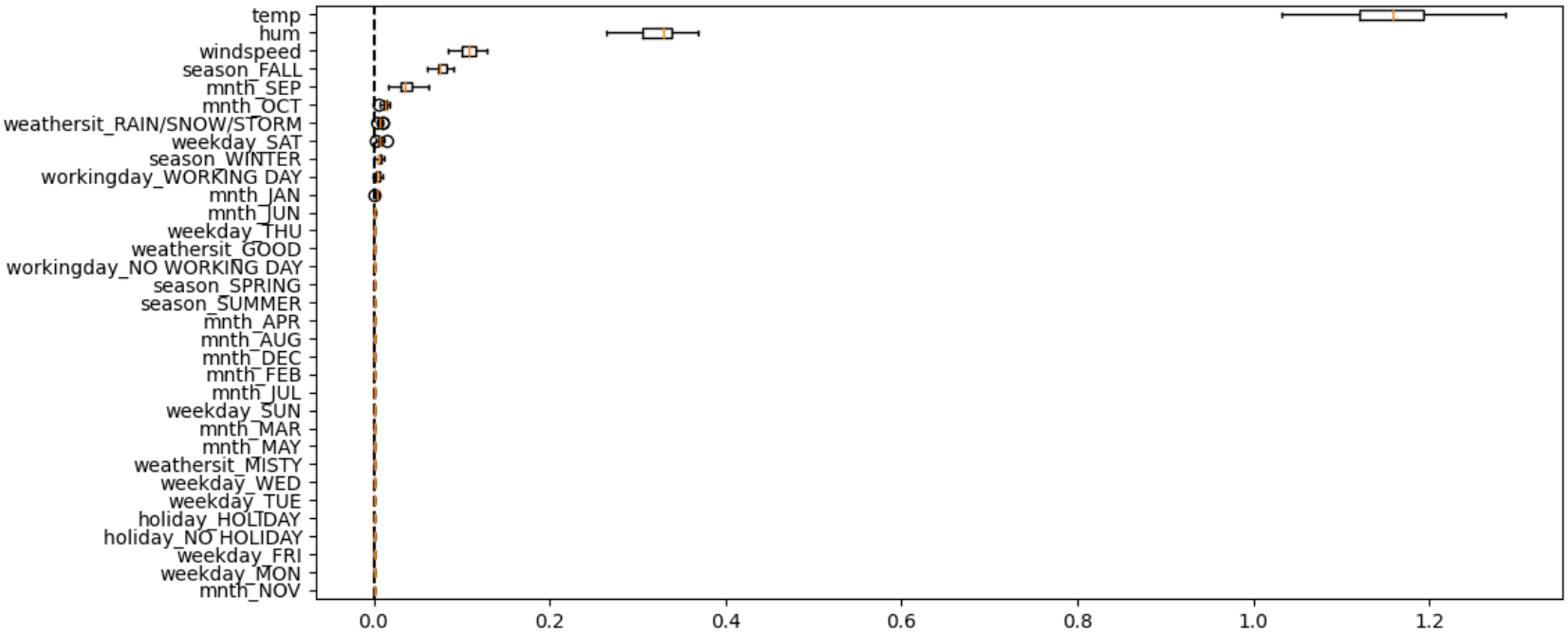
Permutation Feature Importance

Produce a nice plot of sorted feature importance:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.boxplot(
    r.importances[r_idx].T,
    vert=False,
    labels=x.columns[r_idx])
ax.axvline(x=0, color="k", linestyle="--")
plt.show()
```

MEMORIAL UNIVERSITY

Permutation Feature Importance



Uncertainty due to multiple permutations (parameter `n_repeats`)

MEMORIAL UNIVERSITY

Global Surrogate Models

Intuition

Predict the predictions of a complex ”black box” model using an intrinsically interpretable model.

Example ”black box” model:

```
from sklearn.neural_network import MLPRegressor
regr = MLPRegressor((4, 2,), max_iter=10000)
regr.fit(x, y)
preds = regr.predict(x)
```

Interpretable, linear model to explain predictions:

```
from statsmodels.api import OLS
OLS(preds, x.astype(float)).fit().summary()
```

MEMORIAL UNIVERSITY

Global Model Agnostic Methods – Summary

PDP/ICE	
Intuitive	Limited number of features
Clear interpretation	Assumes feature independence
Easy to implement	
ALE	
Unbiased for correlated features	Local interpretation only
Clear interpretation	ALE may differ from linear coefficients
Faster to compute than PDP	No ICE curves
	Unstable for large number of intervals
PFI	
Clear interpretation	Linked to model error
Concise, global measure	Requires access to true targets
Does not require retraining	May be biased for correlated features
Takes into account all interactions	
Global Surrogate Models	
Flexible	Conclusions about model, not data
Intuitive	Unclear cut-off for goodness of fit
R-squared measure for fit	

MEMORIAL UNIVERSITY

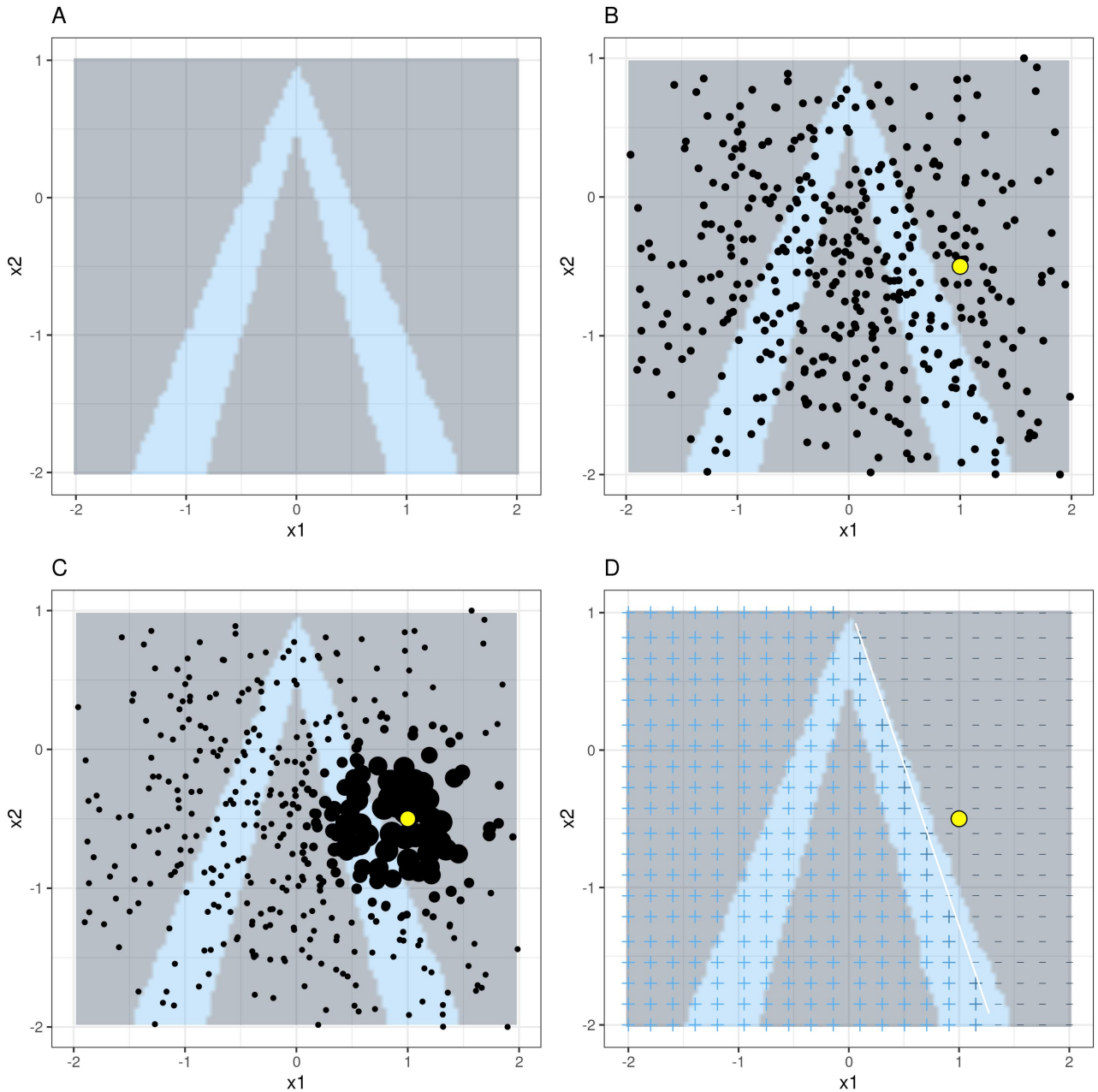
Local Interpretable Model-Agnostic Explanations

Idea

1. Choose an instance x of interest
2. Perturb data by turning on or off features i using randomized feature combination vector of $z_i \in [0, 1]$
3. Sample perturbed instances around x, weighted by kernel π_g,
4. Fit each perturbed instance using black-box model $f(z_i)$
5. Train local interpretable model on features z_i, targets $f(z_i)$ and weights $\pi_x(z_i)$

MEMORIAL UNIVERSITY

LIME – Example

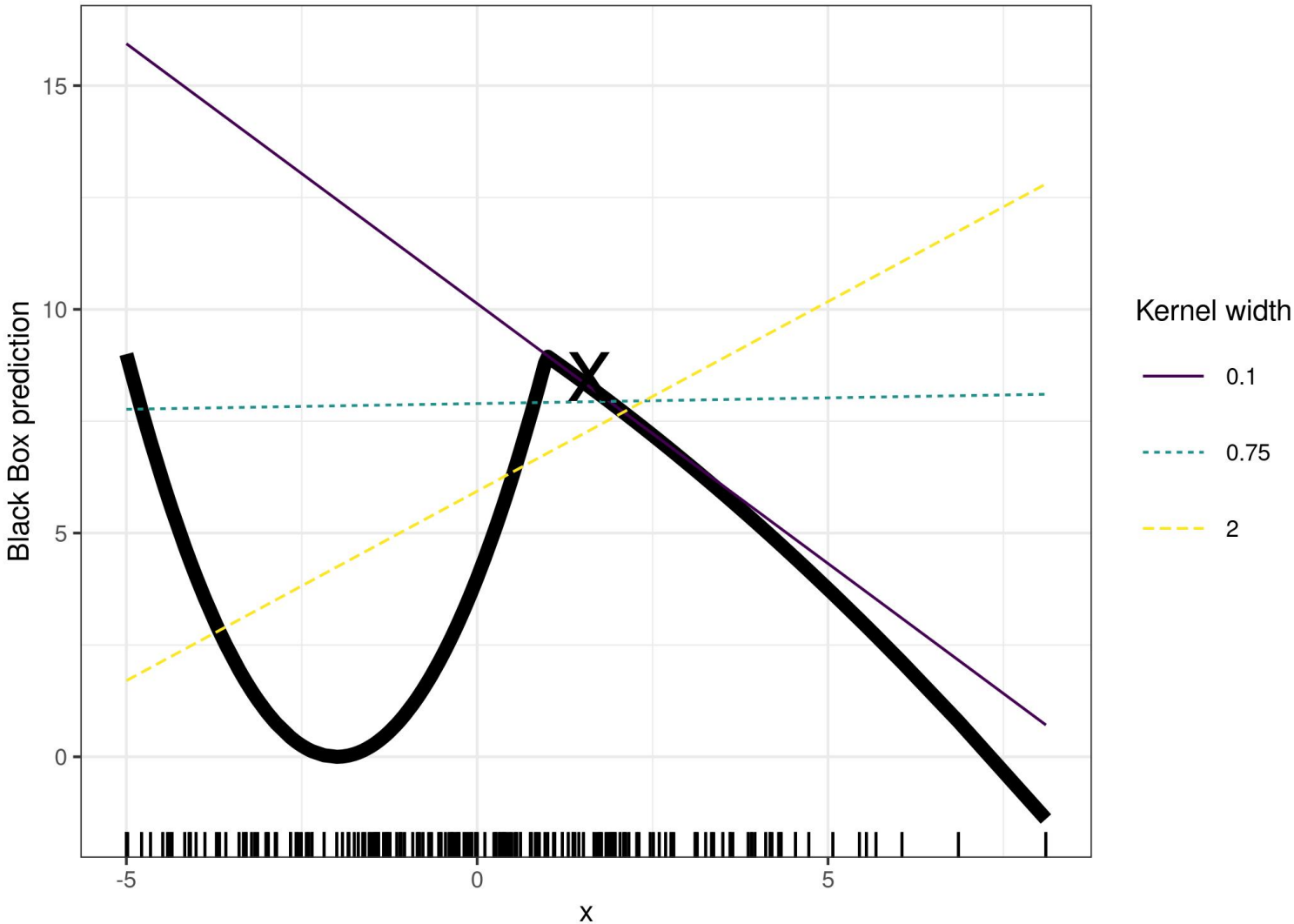


Source: Molnar
Figure 9.5

MEMORIAL UNIVERSITY

LIME – Example

- Weight function π is often an exponential smoothing kernel
- Kernel width is critical determinant of explanation



Source: Molnar Figure 9.6

LIME – Example

Using a deep decision tree as ”black box”:

```
import sklearn.tree
dt = sklearn.tree.DecisionTreeClassifier(max_depth=8)
dt.fit(x, y)
```

Create the explainer:

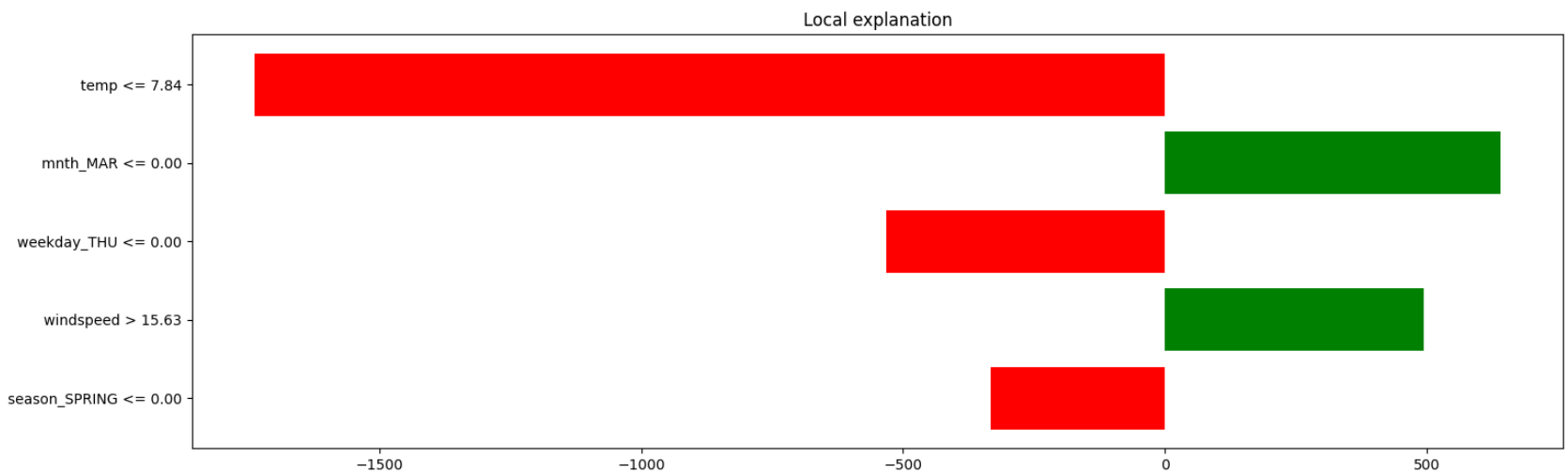
```
import lime, lime.lime_tabular
from sklearn.linear_model import Ridge

explainer = lime.lime_tabular.LimeTabularExplainer(
    x.to_numpy(),
    feature_names=x.columns,
    discretize_continuous = True,
    mode='regression',
    verbose=True)
```

MEMORIAL UNIVERSITY

LIME – Example

Explain instance number 5:

```
exp = explainer.explain_instance(
    x.to_numpy()[7],
    dt.predict,
    num_features=5,
    num_samples=1000,
    distance_metric='euclidean')
exp.as_list()
exp.as_pyplot_figure().show()
```



MEMORIAL UNIVERSITY

LIME for Images

LIME explanations for label ”bagel” and ”strawberries”:

Molnar, Figure 9.8

Python Examples:

`https://github.com/marcotcr/lime`

Paper:

`https://arxiv.org/abs/1602.04938`

Shapley Values

Motivation

How much does *feature value* x_j contribute to the overall prediction compared to the average prediction?

Game Theory

- Players cooperate in a coalition and receive a certain profit from this cooperation.
- Method for assigning payouts to players depending on their contribution to the total payout.

MEMORIAL UNIVERSITY

Shapley Values

$$\phi_i(v) = \frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \binom{n-1}{|S|}^{-1} [v(S \cup \{i\}) - v(S)]$$

- $v(S \cup \{i\}) - v(S)$: marginal contribution of player i to coalition of players S
- $\binom{n-1}{|S|}$: number of possible ways to form a coalition of size $|S|$ of the set $N \setminus \{i\}$ of $n-1$ players (set N without player i)

MEMORIAL UNIVERSITY

Shapley Value

Fairness Properties

- **Efficiency**: Contributions add up to total value
- **Symmetry**: If two players contribute equally to all possible coalitions, they have the same Shapley value
- **Dummy**: A player that does not contribute at all has a Shapley value of 0
- **Additivity**: For a game with combined payouts $v + w$, the Shapley values of players are $\phi^{(v)} + \phi^{(w)}$

MEMORIAL UNIVERSITY

Shapley Values in Interpretable ML

- Players are feature values
- Coalitions are combinations of feature values
- Presence in a coalition means we know the value
- Absence from a coalition means we don't know the value $\Rightarrow$ integrate/marginalize over all values of all features not in coalition S

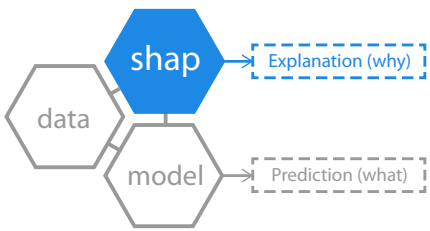
$$v_x(S) = \int \cdots \int_{\mathbb{R}} \hat{f}(x_1, \ldots, x_p) d\mathbb{P}_{x \notin S} - E_x(\hat{f}(X))$$

- Expensive to compute $\Rightarrow$ in practice, approximation by sampling and permuting values (can make for unrealistic instances when features are correlated)

MEMORIAL UNIVERSITY

Shapley Additive Explanations (SHAP)

SHAP



Paper

`https://arxiv.org/abs/1705.07874`

Documentation (Intro and Examples)

`https://shap.readthedocs.io/en/latest/index.html`

Python Code and Tutorials

`https://github.com/shap/shap`

MEMORIAL UNIVERSITY

SHAP Example

Fit a simple regression model to the California housing dataset:

```
import sklearn
import shap

X, y = shap.datasets.california(n_points=1000)
model = sklearn.linear_model.LinearRegression()
model.fit(X, y)
```

Compute the SHAP values:

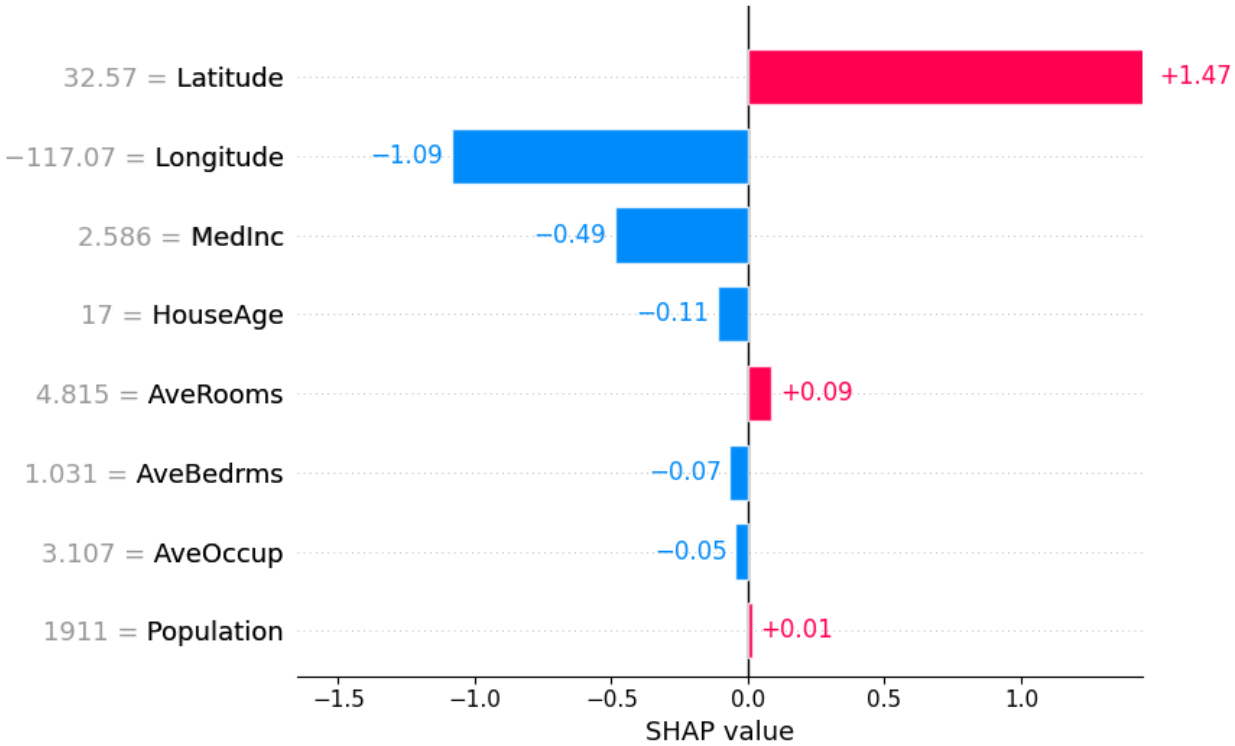
```
X100 = shap.utils.sample(X, 100)
explainer = shap.Explainer(model.predict, X100)
shap_values = explainer(X)
```

MEMORIAL UNIVERSITY

SHAP Example

The **barplot** shows the importance of feature values for an individual prediction:

```
shap.plots.bar(shap_values[20])
```

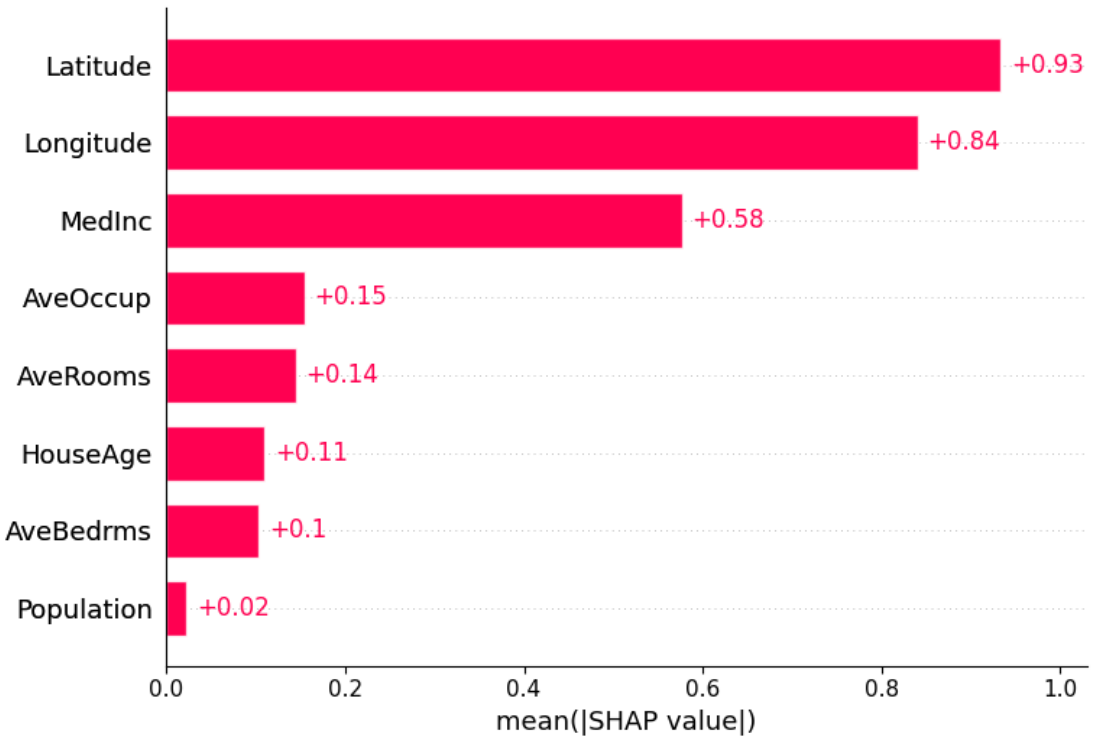


MEMORIAL UNIVERSITY

SHAP Example

The **barplot** can also show the importance of a feature by averaging over all instances (and their feature values):

```
shap.plots.bar(shap_values)
```

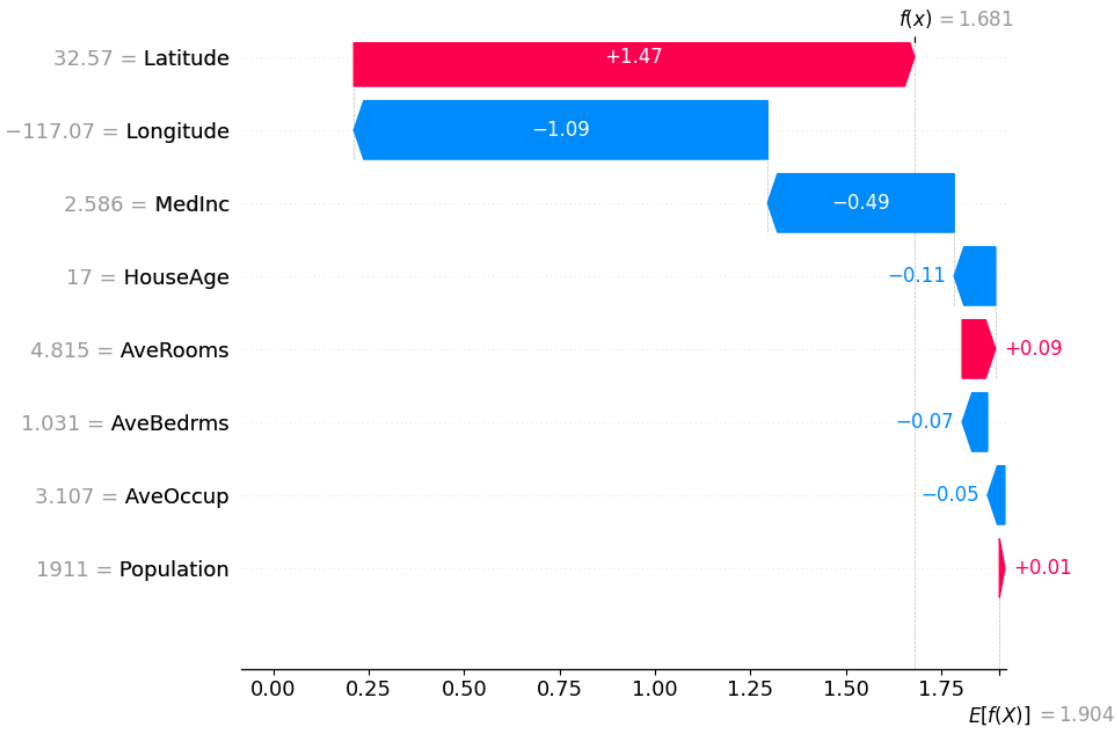


MEMORIAL UNIVERSITY

SHAP Example

Waterfall plots explain how feature values combine to produce an individual prediction:

```
sha.plots.waterfall(shap_values[20], max_display=14)
```

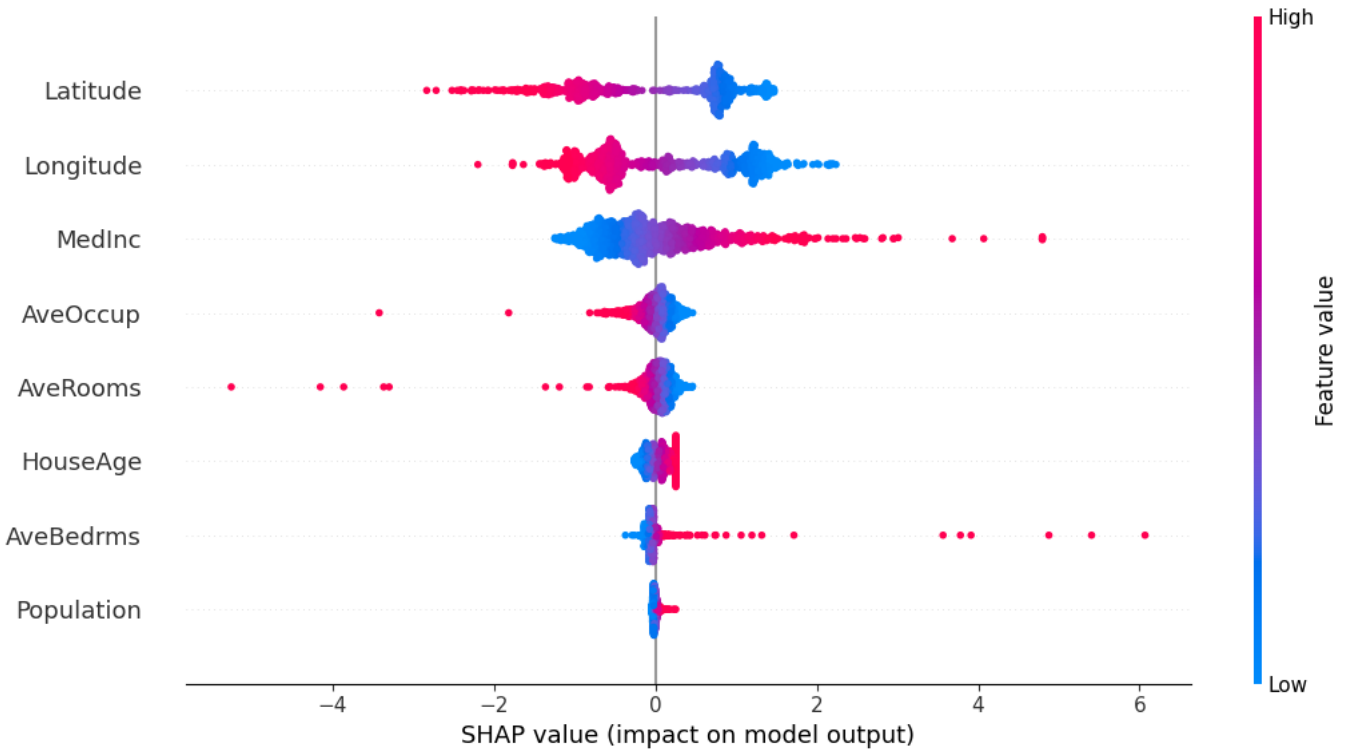


MEMORIAL UNIVERSITY

SHAP Example

Beeswarm plots explain all feature values for all instances (represented by a dot):

```
shap.plots.beeswarm(shap_values)
```

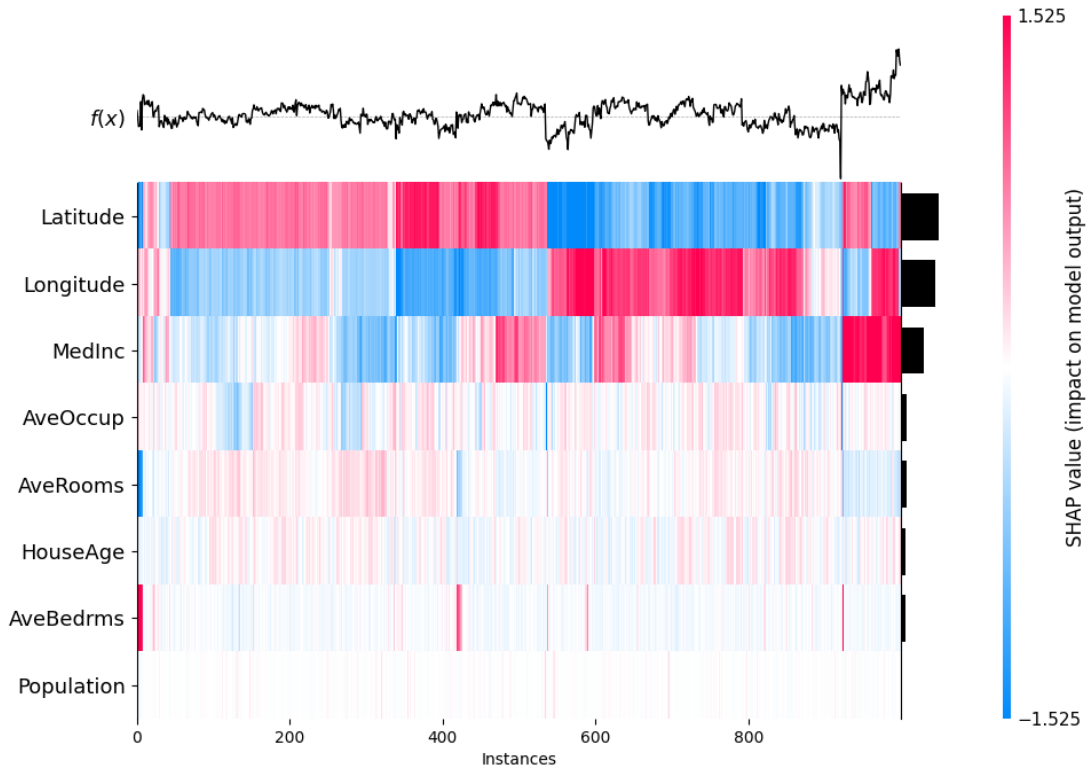


MEMORIAL UNIVERSITY

SHAP Example

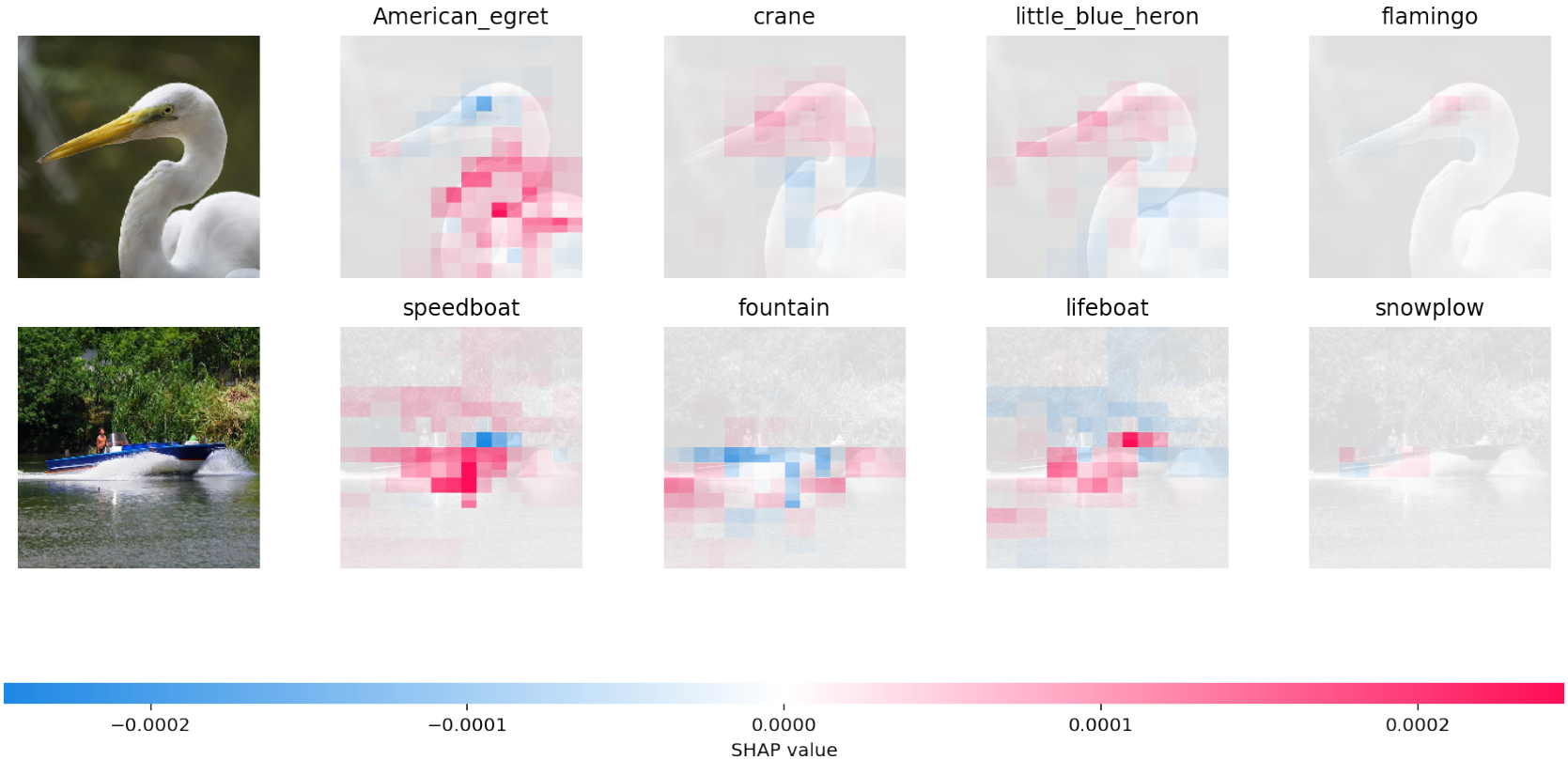
The **heatmap** shows SHAP values of feature values for all instances, and shows model prediction and global feature importance in rugs:

```
shap.plots.heatmap(shap_values)
```



SHAP for Image Classification

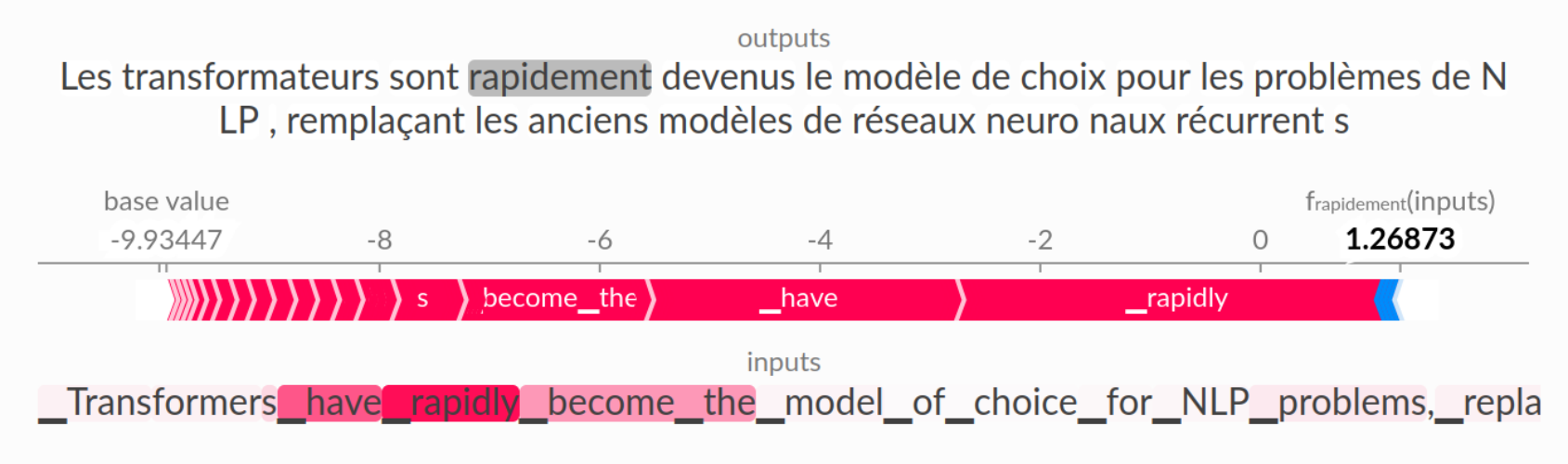
- Presence/absence of features/pixels by masking parts of an image:



Source: `https://github.com/shap` (MIT License)

MEMORIAL UNIVERSITY

SHAP for Text Classification



Source: `https://shap.readthedocs.io/en/latest/text_examples.html` (MIT License)

Fixes for Matplotlib

```
sudo apt-get install -y qt6-base-dev xcb libxcb-cursor-dev
sudo ln -sf /usr/lib/x86_64-linux-gnu/qt6/plugins/platforms/ \
    /usr/bin/
pip install pyqt6
```

MEMORIAL UNIVERSITY