Interpretability

ACTL3143 & ACTL5111 Deep Learning for Actuaries

Patrick Laub

Lecture Outline

- **Introduction**
- Illustrative Example
- Inherently Interpretable Models
- Post-hoc Explanations
- Belgian Motor Dataset
- Interpreting Inherently Interpretable Models
- Explaining a Neural Network with Partial Dependence Plots
- Explaining Specific Models

Definitions

> "Surprisingly enough, although the concept of interpretability is increasingly widespread, there is no general consensus on both the definition and the measurement of the interpretability of a model." (Delcaillau et al., 2022)

> "**Definition** Interpret means *to explain or to present in understandable terms*. In the context of ML systems, we define interpretability as *the ability to explain or to present in understandable terms to a human*." (Doshi-Velez & Kim, 2017)

A distinction between interpretability and explainability

> "Interpretability is about transparency, about understanding exactly why and how the model is generating predictions, and therefore, it is important to observe the inner mechanics of the algorithm considered. This leads to interpreting the model's parameters and features used to determine the given output. Explainability is about explaining the behavior of the model in human terms." (Charpentier, 2024)

Aspects of Interpretability

Inherent Interpretability

The model is interpretable by design.

Post-hoc Explanations

The model is not interpretable by design, but we can use other methods to explain the model.

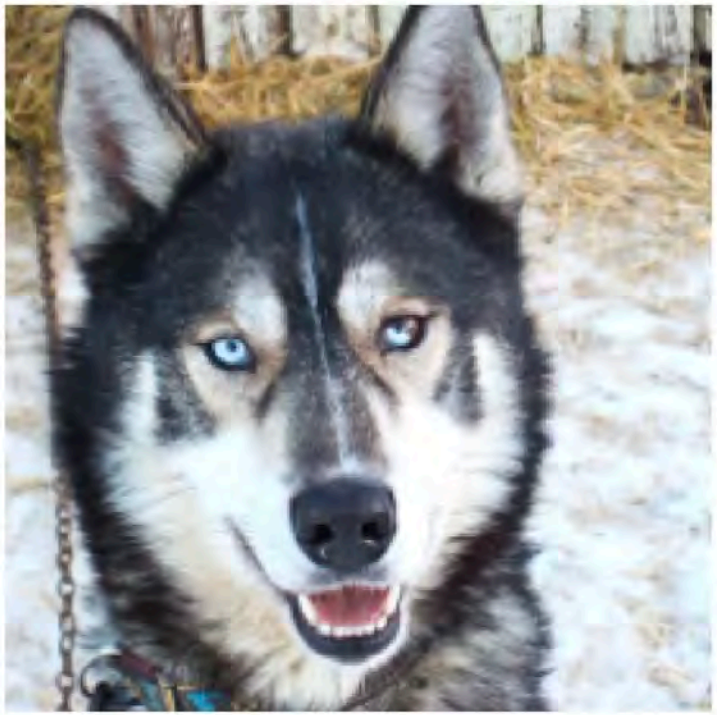
Global Interpretability

The ability to understand how the model works.

Local Interpretability

The ability to interpret/understand each prediction.

Husky vs. Wolf



(a) Husky classified as wolf

(b) Explanation

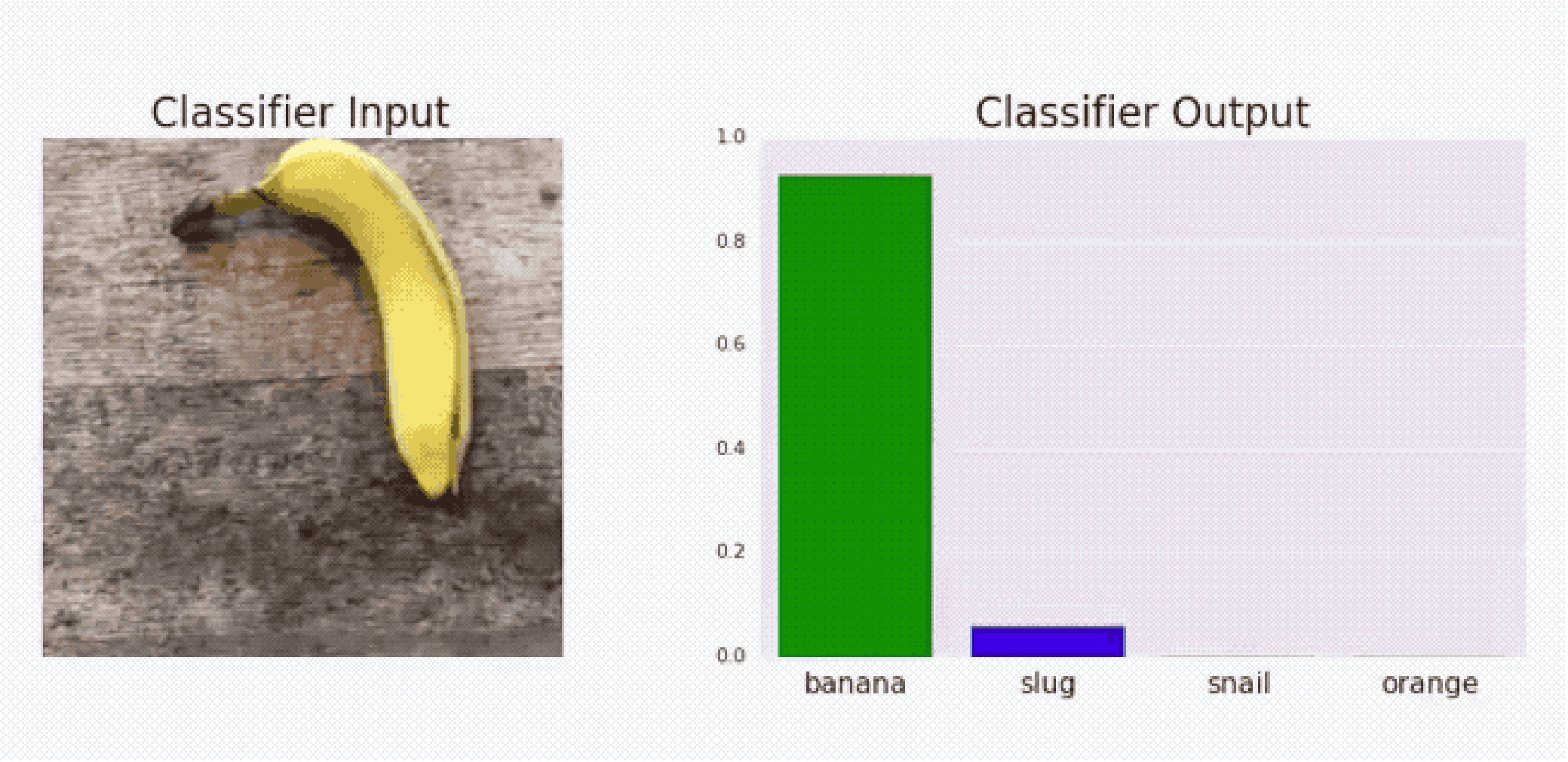
Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

A well-known anecdote in the explainability literature (Ribeiro et al., 2016).

Adversarial examples

A demonstration of fast adversarial example generation applied to GoogLeNet on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image (Goodfellow et al., 2014)

Adversarial stickers



Adversarial stickers.

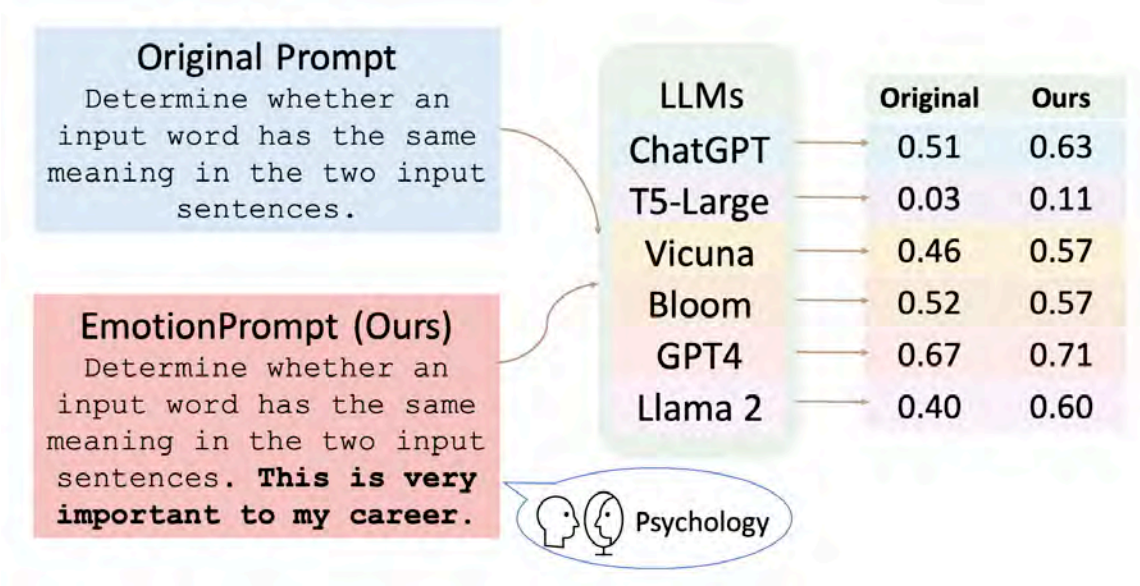
Source: The Verge (2018), These stickers make computer vision software hallucinate things that aren't there.

Adversarial text

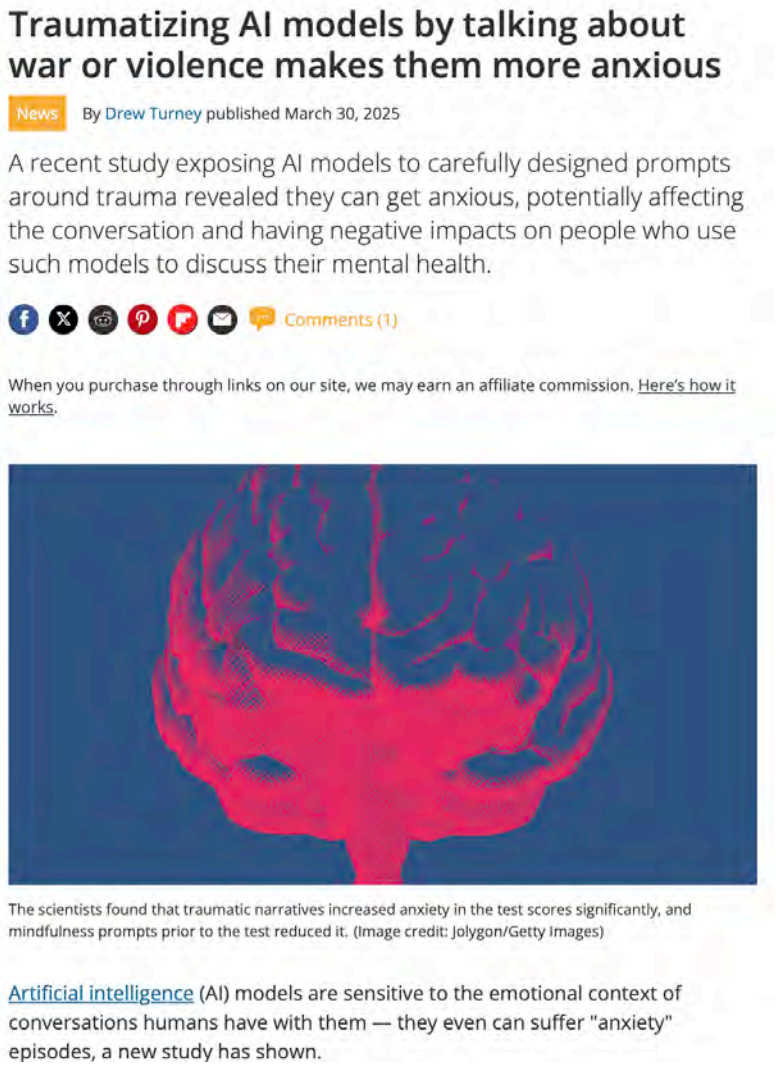
"TextAttack 🐙 is a Python framework for adversarial attacks, data augmentation, and model training in NLP"

Demo

LLMs are even more opaque



"Figure 1: An overview of our research from generating to evaluating EmotionPrompt." (Li et al., 2023)



Traumatizing AI models by talking about war or violence makes them more anxious

News By Drew Turney published March 30, 2025

A recent study exposing AI models to carefully designed prompts around trauma revealed they can get anxious, potentially affecting the conversation and having negative impacts on people who use such models to discuss their mental health.

Comments (1)

When you purchase through links on our site, we may earn an affiliate commission. Here's how it works.

The scientists found that traumatic narratives increased anxiety in the test scores significantly, and mindfulness prompts prior to the test reduced it. (Image credit: Jolygon/Getty Images)

Artificial intelligence (AI) models are sensitive to the emotional context of conversations humans have with them — they even can suffer "anxiety" episodes, a new study has shown.

A popular science article about Ben-Zion et al. (2025)

Lecture Outline

- Introduction
- **Illustrative Example**
- Inherently Interpretable Models
- Post-hoc Explanations
- Belgian Motor Dataset
- Interpreting Inherently Interpretable Models
- Explaining a Neural Network with Partial Dependence Plots
- Explaining Specific Models

First attempt at NLP task

```
1 df_raw["SUMMARY_EN"]
```

```
0       V1, a 2000 Pontiac Montana minivan,
made a lef ...
1       The crash occurred in the eastbound
lane of a  ...
2       This crash occurred just after the
noon time h ...
                                  ...
6946    The crash occurred in the eastbound
lanes of a ...
6947    This single-vehicle crash occurred in
a rural  ...
6948    This two vehicle daytime collision
occurred mi ...
Name: SUMMARY_EN, Length: 6949, dtype: str
```

```
1 df_raw["NUM_VEHICLES"].value_counts()\
2   .sort_index()
```

```
NUM_VEHICLES
1     1822
2     4151
3+     976
Name: count, dtype: int64
```

Source: JSchelldorfer's GitHub.

Bag of words for the top 1,000 words

```
vect = CountVectorizer(max_features=1_000, stop_words="english")
vect.fit(X_train["SUMMARY_EN"])

X_train_bow = vectorise_dataset(X_train, vect)
X_val_bow = vectorise_dataset(X_val, vect)
X_test_bow = vectorise_dataset(X_test, vect)

vectorise_dataset(X_train, vect, dataframe=True).head()
```

	10	105	113	12	15	150	16	17	18	180	...	yield	zone	WEATHER1	V
2532	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
6209	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
2561	1	0	1	0	0	0	0	0	0	0	...	0	0	0	0
6664	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
4214	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0

5 rows × 1008 columns

Trained a basic neural network on that

```
num_features = X_train_bow.shape[1]
num_cats = df_raw["NUM_VEHICLES"].nunique()
model = build_model(num_features, num_cats)
es = EarlyStopping(patience=1, restore_best_weights=True, monitor="val_accuracy")
model.fit(X_train_bow, y_train, epochs=10,
    callbacks=[es], validation_data=(X_val_bow, y_val), verbose=0)
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 100)	100,900
dense_1 (Dense)	(None, 3)	303

Total params: 303,611 (1.16 MB)
Trainable params: 101,203 (395.32 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 202,408 (790.66 KB)

```
model.evaluate(X_train_bow, y_train, verbose=0)
```

```
[0.03331480547785759, 0.9949628114700317, 0.9997601509094238]
```

```
model.evaluate(X_val_bow, y_val, verbose=0)
```

```
[0.09635408222675323, 0.9748201370239258, 0.9978417158126831]
```

UNSW SYDNEY

Permutation importance algorithm

Taken directly from scikit-learn documentation:

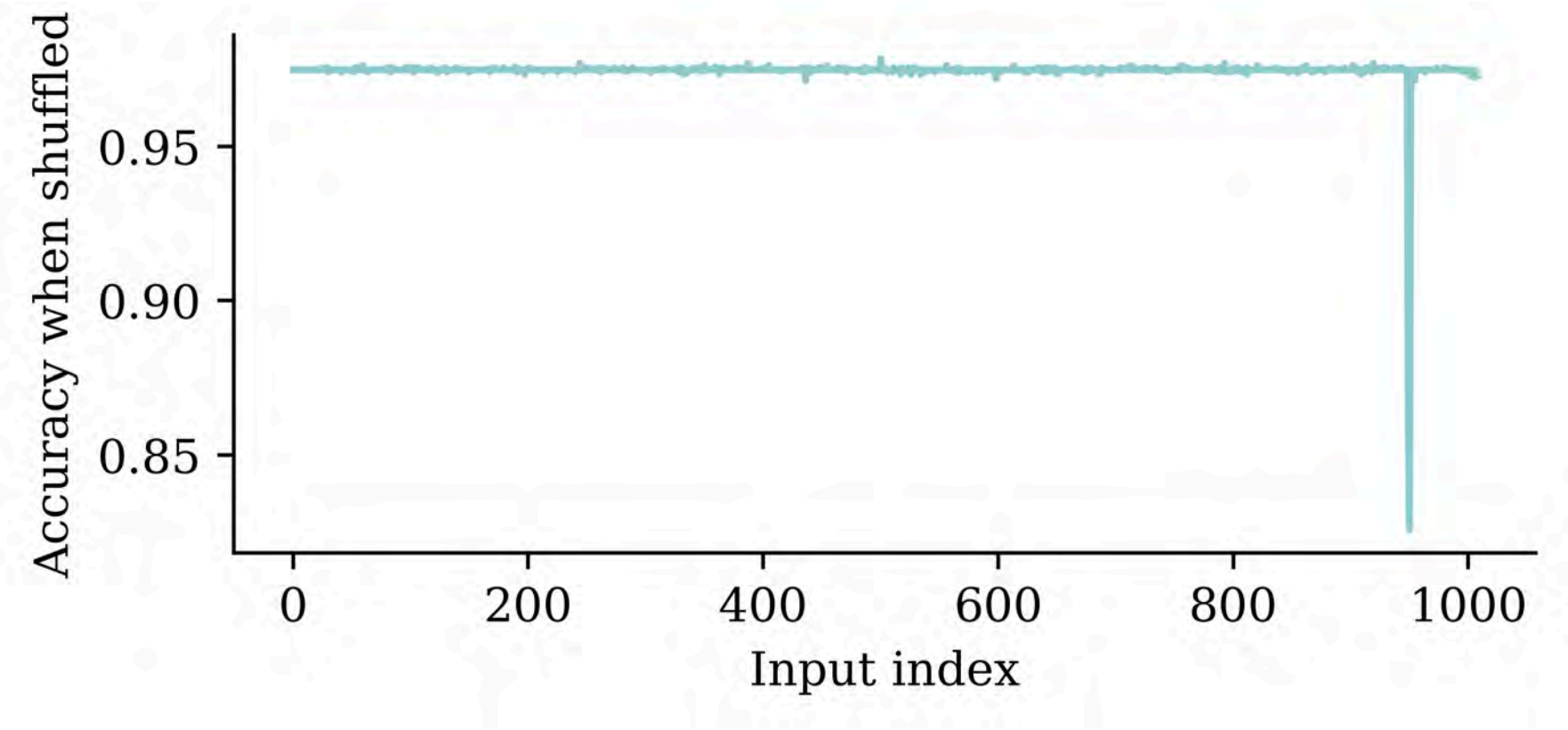
- Inputs: fitted predictive model m, tabular dataset (training or validation) D.
- Compute the reference score s of the model m on data D (for instance the accuracy for a classifier or the R^2 for a regressor).
- For each feature j (column of D):
  - For each repetition k in $1, \ldots, K$:
    - Randomly shuffle column j of dataset D to generate a corrupted version of the data named $\tilde{D}_{k,j}$.
    - Compute the score $s_{k,j}$ of model m on corrupted data $\tilde{D}_{k,j}$.
  - Compute importance i_j for feature f_j defined as:

$$i_j = s - \frac{1}{K} \sum_{k=1}^{K} s_{k,j}$$

Source: scikit-learn documentation, permutation_importance function.

Run the permutation test

```
1 all_perm_scores = permutation_test(model, X_val_bow, y_val)
```



Find the most significant inputs

```
['v3', 'v2', 'vehicle', 'harmful', 'v4', 'motor', 'WEATHER8', 'WEATHER5', 'WEATHER4',
'WEATHER3', 'posted', 'trailer', 'parked', 'drove', 'chevrolet', 'event', 'single',
'WEATHER7', 'lane', 'light', 'legally', 'steered', 'traffic', 'continued', 'rest', 'concrete',
'afternoon', 'surface', 'arterial', 'asphalt', 'coded', 'second', 'seizure', 'seven',
'shortly', 'change', 'sightline', 'sign', 'struck', 'signs', 'striking', 'braking', 'began',
'spun', 'started', 'steady', 'attack', 'researcher', 'directions', 'decision', 'medication',
'median', 'maneuver', 'male', 'make', 'gas', 'meters', 'located', 'heading', 'leaving',
'kilometers', 'involved', 'inadequate', 'information', 'limit', 'crossed', 'minivan', 'mph',
'pre', 'pole', 'plane', 'direction', 'pickup', 'drivers', 'movement', 'pain', 'ordered',
'oncoming', 'encroaching', 'observed', 'number', 'noticed', 'dry', 'travel', 'injuries',
'2006', 'waiting', 'year', 'vehicles', 'utility', '30', 'treatment', 'WEATHER1', '38', '48',
'ability', 'notice', 'noted', 'numerous', 'object']
```

How about a simple decision tree?

```
1 from sklearn import tree
2
3 clf = tree.DecisionTreeClassifier(random_state=0, max_leaf_nodes=3)
4 clf.fit(X_train_bow[:, best_input_inds], y_train);
```

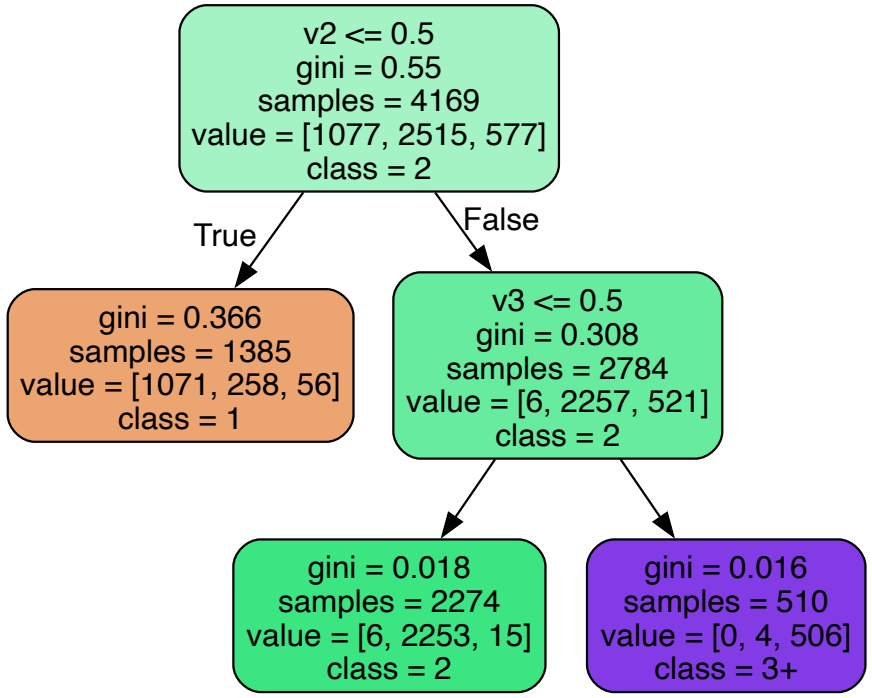
```
1 print(clf.score(X_train_bow[:, best_input_inds], y_train))
2 print(clf.score(X_val_bow[:, best_input_inds], y_val))
```

```
0.9186855360997841
0.9316546762589928
```

The decision tree ends up giving pretty good results.

Decision tree

```
1  tree.plot_tree(clf, feature_names=best_inputs, filled=True);
```




```
1  print(np.where(clf.feature_importances_ > 0)[0])
2  [best_inputs[ind] for ind in np.where(clf.feature_importances_ > 0)[0]]
```

```
[0 1]
['v3', 'v2']
```

UNSW SYDNEY

This is why we replace “v1”, “v2”, “v3”

There was a slide in the NLP deck titled “Just ignore this for now…” That was going through each summary and replacing the words “V1”, “V2”, “V3” with random numbers. This was done to see if the model was overfitting to these words.

```
model.fit(X_train_bow, y_train, epochs=10,
    callbacks=[es], validation_data=(X_val_bow, y_val), verbose=0);
```

Retraining on the fixed dataset gives us a more realistic (lower) accuracy.

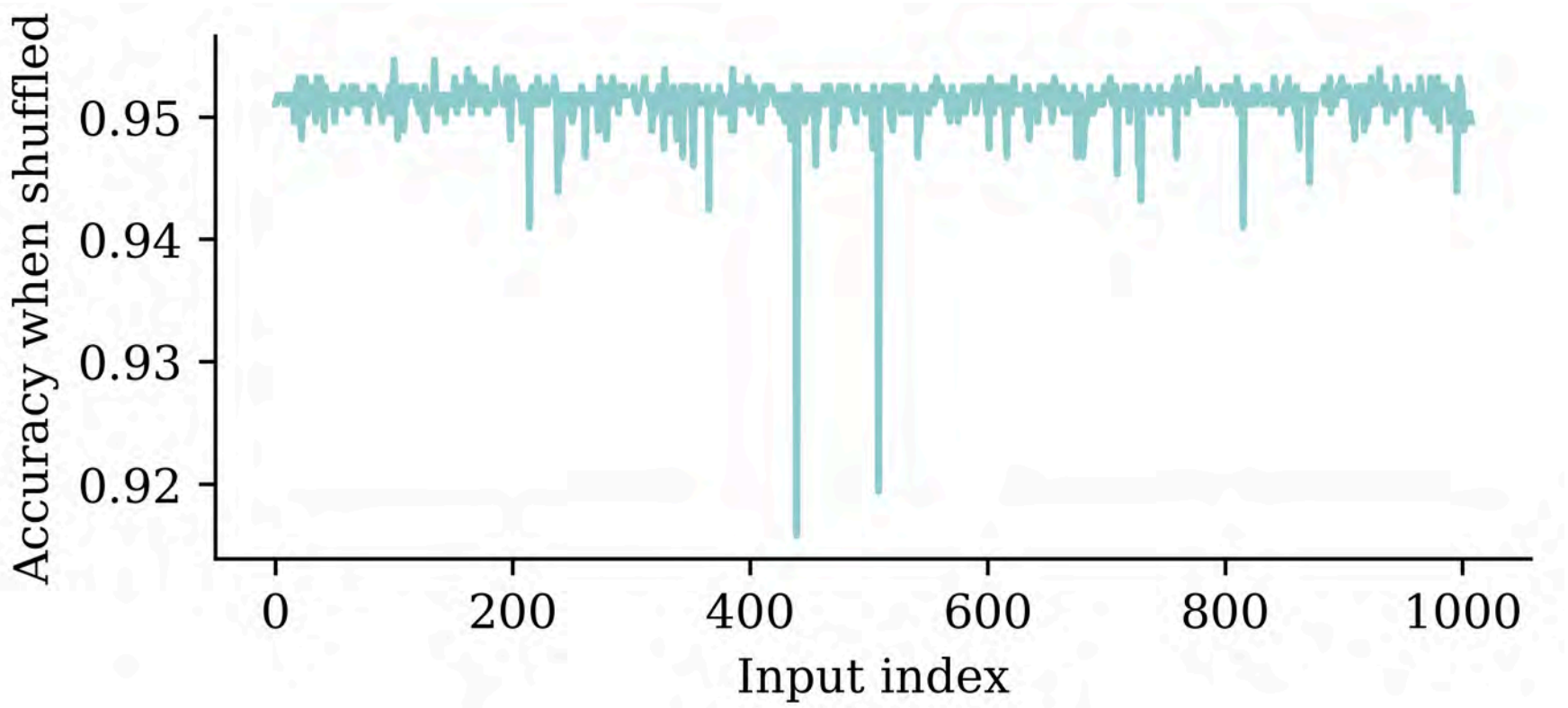
```
model.evaluate(X_train_bow, y_train, verbose=0)
```

```
[0.022723009809851646, 0.9990405440330505, 1.0]
```

```
model.evaluate(X_val_bow, y_val, verbose=0)
```

```
[0.16461113095283508, 0.9517985582351685, 0.9964028596878052]
```

Permutation importance accuracy plot



Find the most significant inputs

```
1 vocab = vect.get_feature_names_out()
2 input_cols = list(vocab) + weather_cols
3
4 best_input_inds = np.argsort(perm_scores)[:100]
5 best_inputs = [input_cols[idx] for idx in best_input_inds]
6
7 print(best_inputs)
```

```
['harmful', 'involved', 'single', 'coded', 'event', 'reason', 'year', 'contacted', 'struck',
'pushed', 'hit', 'encroachment', 'rear', 'continued', 'road', 'pickup', 'non', 'pole',
'limit', 'edge', 'critical', 'driven', 'impact', 'motor', 'intersection', 'stopped', 'police',
'guardrail', 'dry', 'vehicles', 'chevrolet', 'daylight', 'afternoon', '1999', 'occurred',
'traffic', 'day', 'alcohol', 'failure', 'WEATHER2', 'associated', 'westbound', 'familiar',
'interview', '30', 'lane', 'line', 'dodge', 'traveling', 'daily', 'away', 'encroaching',
'corner', 'asphalt', 'clear', 'ford', 'kph', 'grand', 'WEATHER5', 'WEATHER4', 'work',
'weekday', 'undivided', 'treated', 'toyota', 'stop', 'ran', 'poor', 'forward', 'parked',
'occupants', 'nissan', 'median', 'male', 'interviewed', 'injured', 'inadequate', 'impairment',
'home', 'heard', 'opposite', 'according', 'WEATHER8', '37', '1993', '23', '72', 'drives',
'right', 'contacting', '2003', 'contributed', 'unsuccessful', 'performance', 'paved',
'behavior', 'roof', 'brake', '46', 'initial']
```

Lecture Outline

- Introduction
- Illustrative Example
- **Inherently Interpretable Models**
- Post-hoc Explanations
- Belgian Motor Dataset
- Interpreting Inherently Interpretable Models
- Explaining a Neural Network with Partial Dependence Plots
- Explaining Specific Models

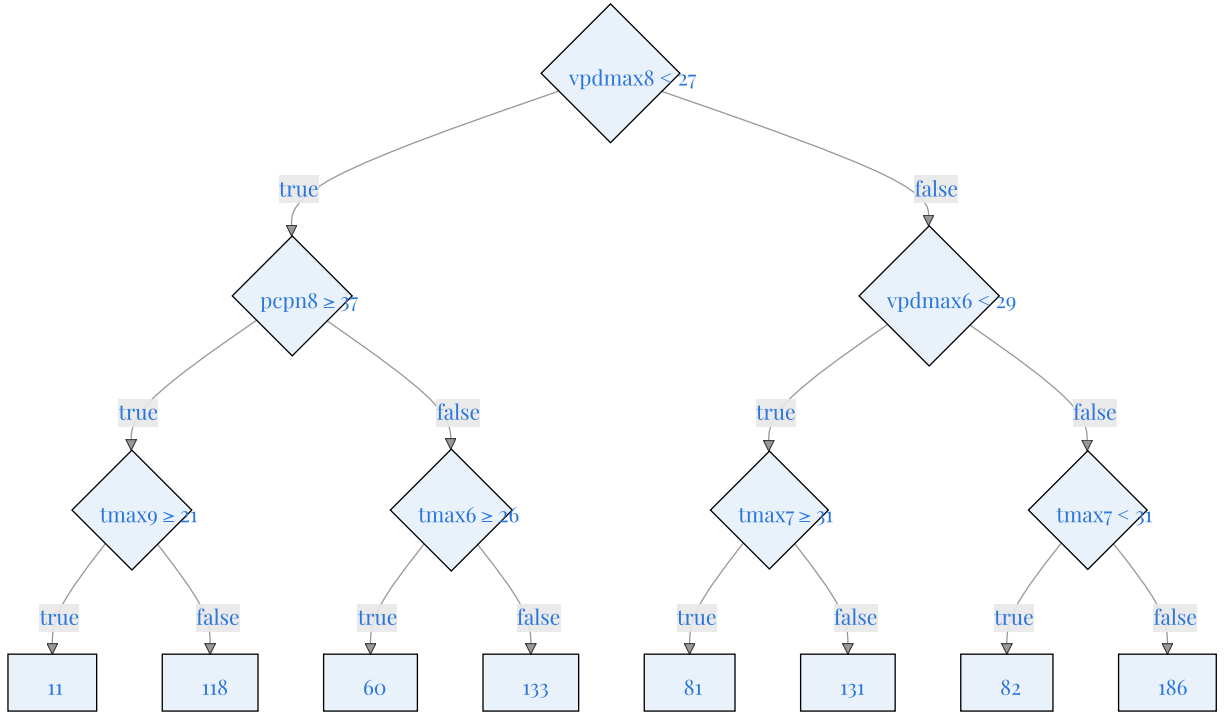
What is inherent interpretability?

> "Interpretability by design is decided on the level of the machine learning algorithm. If you want a machine learning algorithm that produces interpretable models, the algorithm has to constrain the search of models to those that are interpretable. The simplest example is linear regression: When you use ordinary least squares to fit/train a linear regression model, you are using an algorithm that will produce ... models that are linear in the input features. Models that are interpretable by design are also called **intrinsically** or **inherently** interpretable models." (Molnar, 2020)

Christoph Molnar

Examples of interpretable models

- Linear regression
- Generalised linear models
- Decision trees
- Decision rules



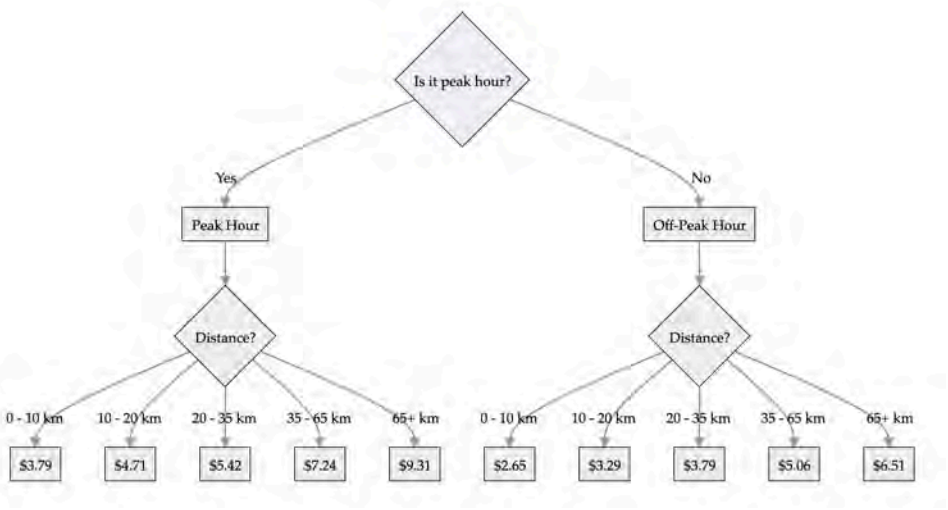
E.g. decision tree for payouts of index insurance (Chen et al., 2024).

Better trees

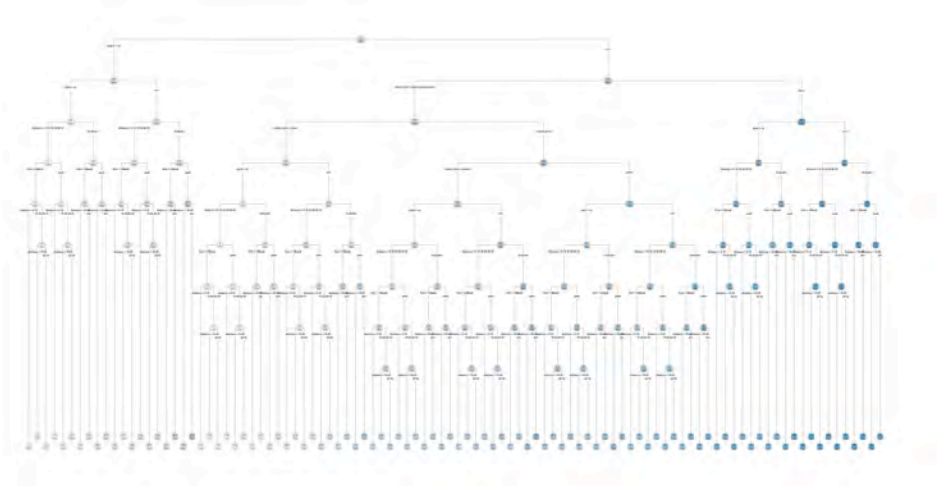
> "The optimization over the node parameters (exact for axis-aligned trees, approximate for oblique trees) assumes the rest of the tree (structure and parameters) is fixed. The greedy nature of the algorithm means that once a node is optimized, it its fixed forever." (Carreira-Perpinán & Tavallali, 2018)

Non-greedy search can improve the accuracy of the tree, without sacrificing interpretability.

Some processes don't fit the tree structure



Train prices



Full train pricing

Decision rules

Make predictions using if-then statements related to the inputs.

> "Decision rules can be **as expressive as decision trees, while being more compact**. Decision trees often also suffer from replicated sub-trees, that is, when the splits in a left and a right child node have the same structure." (Molnar, 2020)

```python
def rail_cost(peak_hours, distance):
  if peak_hours:
      if distance ⩽ 10:
        cost = 3.79
      elif distance ⩽ 20:
        cost = 4.71
      elif distance ⩽ 35:
        cost = 5.42
      elif distance ⩽ 65:
        cost = 7.24
      else:
        cost = 9.31
  else:
      if distance ⩽ 10:
        cost = 2.65
      elif distance ⩽ 20:
        cost = 3.29
      elif distance ⩽ 35:
        cost = 3.79
      elif distance ⩽ 65:
        cost = 5.06
      else:
        cost = 6.51
  return cost
```

Scoring rules

1.	Prior arrests ≥ 2	1 point	$\cdots$
2.	Prior arrests ≥ 5	1 point	$+\cdots$
3.	Prior arrests for local ordinance	1 point	$+\cdots$
4.	Age at Release between 18 to 24	1 point	$+\cdots$
5.	Age at Release ≥ 40	-1 point	$+\cdots$
		SCORE	$=\cdots$

Score	-1	0	1	2	3	4
Risk (%)	11.9	26.9	50.0	73.1	88.1	95.3

Example from Rudin (2019).

Source: Adapted from Rudin (2019), Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, preprint available under Creative Commons.

When to choose inherent interpretability?

Perspective | Published: 13 May 2019

Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead

Cynthia Rudin

Nature Machine Intelligence **1**, 206–215 (2019) | Cite this article

66k Accesses | **2230** Citations | **485** Altmetric | Metrics

Abstract

Black box machine learning models are currently being used for high-stakes decision making throughout society, causing problems in healthcare, criminal justice and other domains. Some people hope that creating methods for explaining these black box models will alleviate some of the problems, but trying to explain black box models, rather than creating models that are interpretable in the first place, is likely to perpetuate bad practice and can potentially cause great harm to society. The way forward is to design models that are inherently interpretable. This Perspective clarifies the chasm between explaining black boxes and using inherently interpretable models, outlines several key reasons why explainable black boxes should be avoided in high-stakes decisions, identifies challenges to interpretable machine learning, and provides several example applications where interpretable models could potentially replace black box models in criminal justice, healthcare and computer vision.

Rudin (2019)

Cynthia Rudin

UNSW SYDNEY

Article 22 GDPR – Automated individual decision-making, including profiling

1. The data subject shall have the right **not** to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her.
2. Paragraph 1 shall not apply if the decision:
    a. is necessary for entering into, or performance of, a contract between the data subject and a data controller;
    b. is authorised by Union or Member State law to which the controller is subject and which also lays down suitable measures to safeguard the data subject's rights and freedoms and legitimate interests; or
    c. is based on the data subject's explicit consent.
3. In the cases referred to in points (a) and (c) of paragraph 2, the data controller shall implement suitable measures to safeguard the data subject's rights and freedoms and legitimate interests, at least the right to obtain human intervention on the part of the controller, to express his or her point of view and to contest the decision.
4. Decisions referred to in paragraph 2 shall not be based on special categories of personal data referred to in Article 9(1), unless point (a) or (g) of Article 9(2) applies and suitable measures to safeguard the data subject's rights and freedoms and legitimate interests are in place.

Linear models & LocalGLMNet

A GLM has the form

$$\hat{y} = g^{-1}\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p\right)$$

where $\beta_0, \ldots, \beta_p$ are the model parameters.

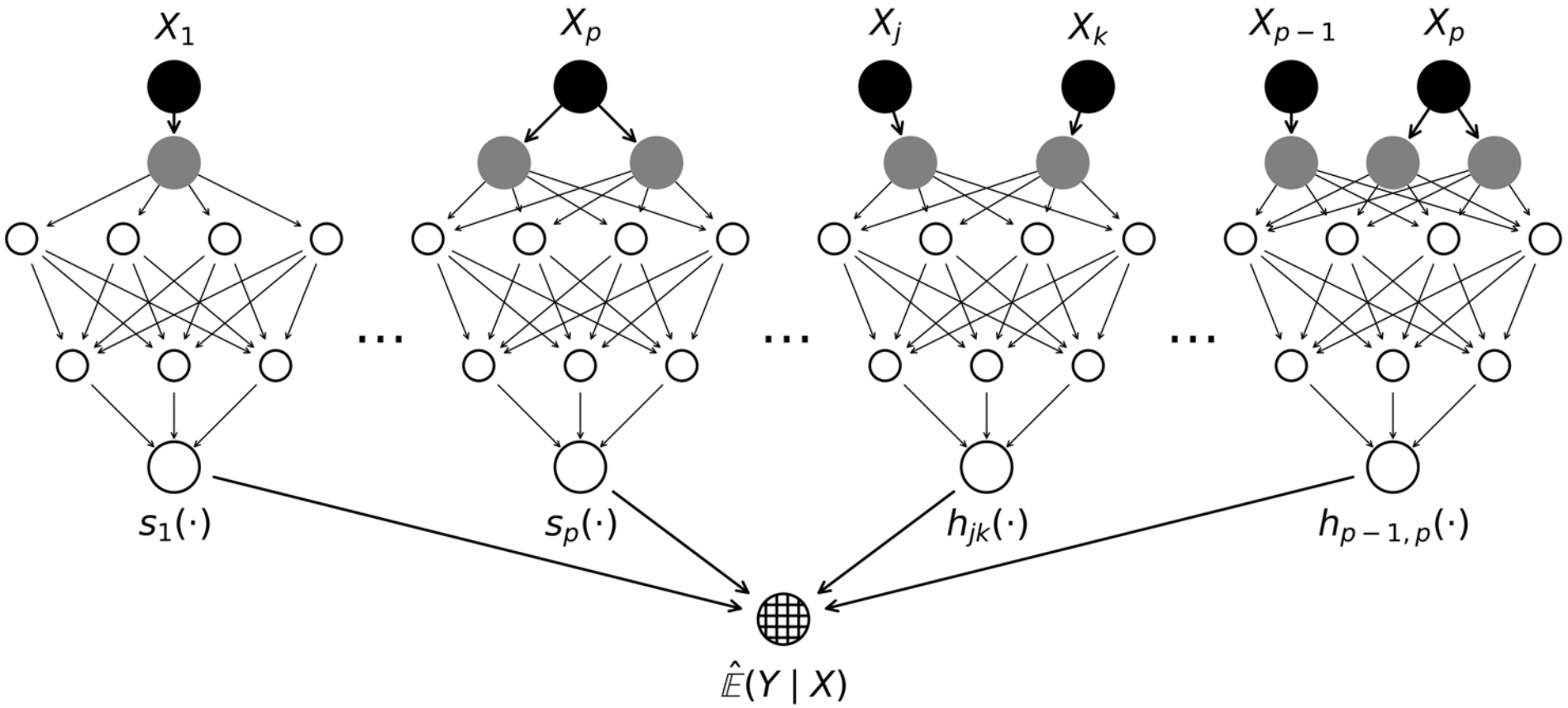
Global & local interpretations are easy to obtain.

LocalGLMNet extends this to a neural network (Richman & Wüthrich, 2023).

$$\hat{y}_i = g^{-1}\left(\beta_0(\boldsymbol{x}_i) + \beta_1(\boldsymbol{x}_i) x_{i1} + \cdots + \beta_p(\boldsymbol{x}_i) x_{ip}\right)$$

A GLM with local parameters $\beta_0(\boldsymbol{x}_i), \ldots, \beta_p(\boldsymbol{x}_i)$ for each observation $\boldsymbol{x}_i$. The local parameters are the output of a neural network.

Neural Additive Models (NAMs)

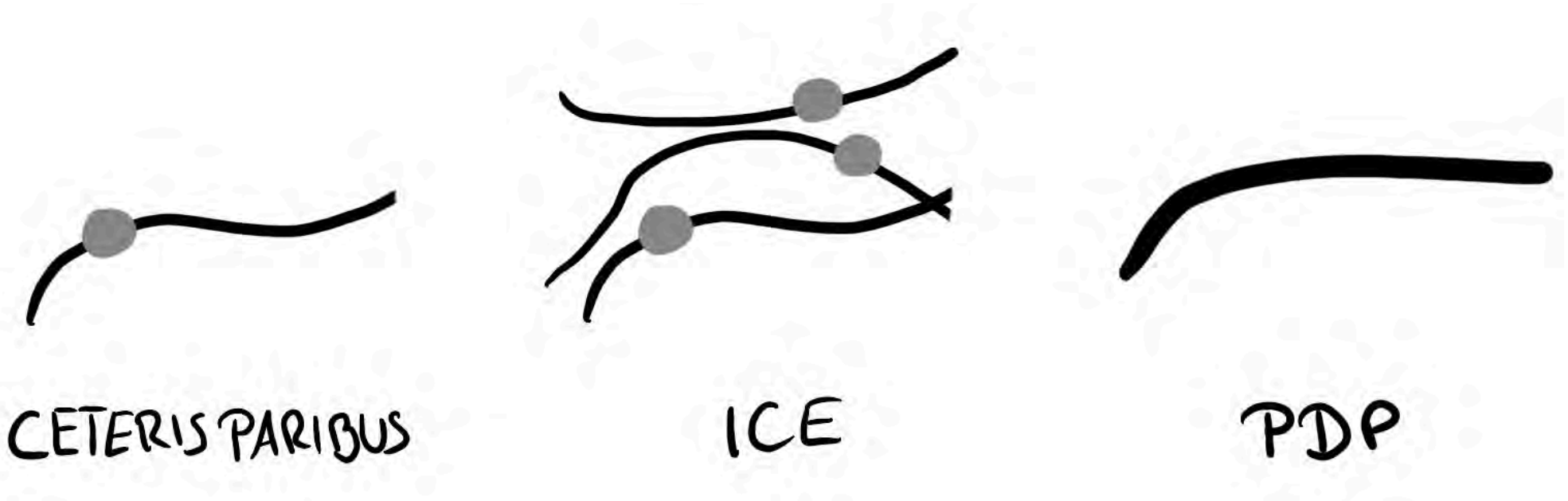


Each covariate (or select interactions) receive their own subnetwork, which contribute additively.

Lecture Outline

- Introduction
- Illustrative Example
- Inherently Interpretable Models
- **Post-hoc Explanations**
- Belgian Motor Dataset
- Interpreting Inherently Interpretable Models
- Explaining a Neural Network with Partial Dependence Plots
- Explaining Specific Models

One variable's effect on the prediction



Various plots showing the effect of one variable on the prediction; from (Molnar, 2020, sec. 12)

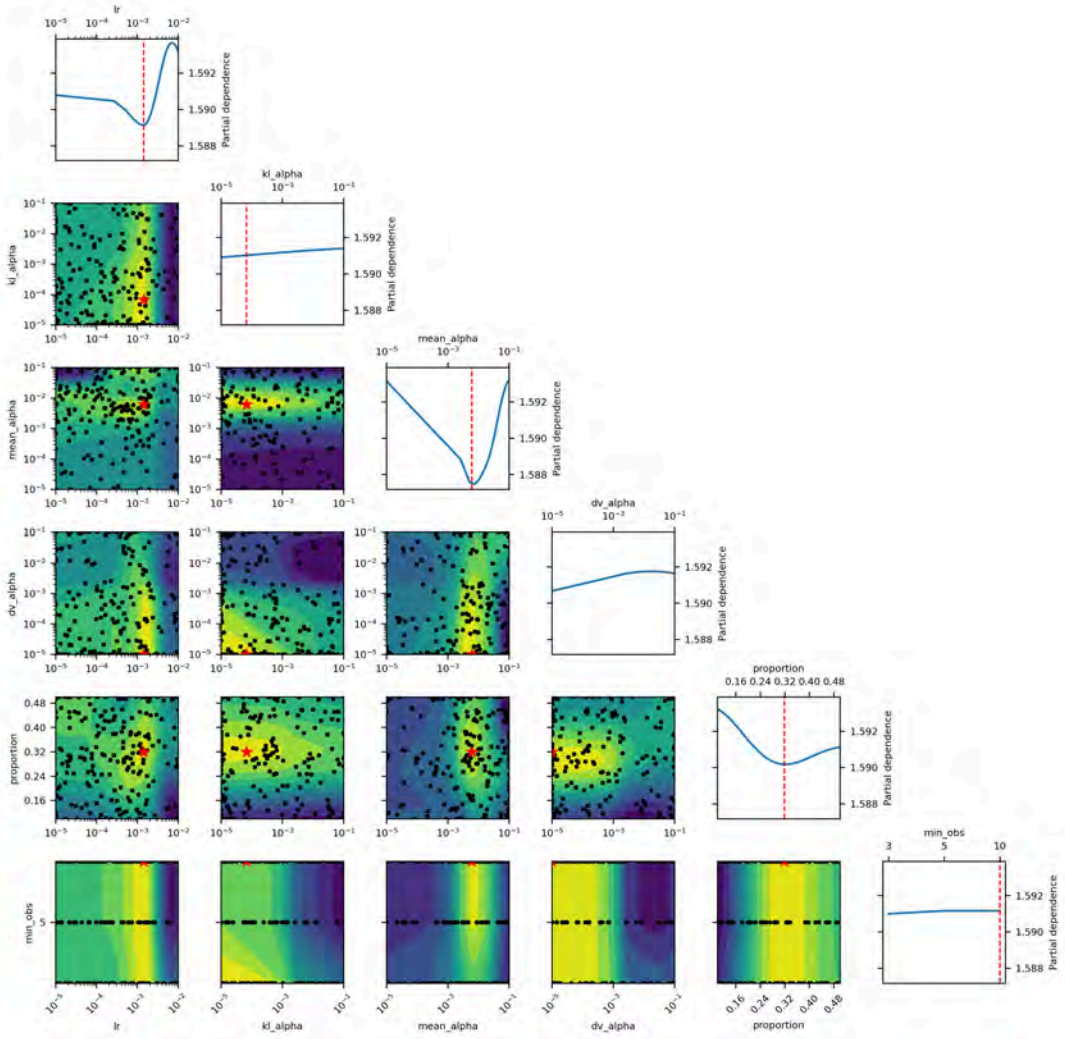
The different plots

A *Ceteris Paribus* plot shows how the model's prediction changes as we vary the value of *one covariate/input* while keeping *all other features constant* at their original values for some observation.

An Individual Conditional Expectation (ICE) plot is the same thing as a ceteris paribus plot, but for *all observations* in the dataset.

A Partial Dependence Plot (PDP) shows how the model's prediction changes as we vary the value of *one covariate/input* while averaging over all other features in the dataset.

Useful for Hyperparameter Tuning



Partial dependence plots for hyperparameter tuning

Permutation importance

- Inputs: fitted model m, tabular dataset D.
- Compute the reference score s of the model m on data D (for instance the accuracy for a classifier or the R^2 for a regressor).
- For each feature j (column of D):
  - For each repetition k in $1, \ldots, K$:
    - Randomly shuffle column j of dataset D to generate a corrupted version of the data named $\tilde{D}_{k,j}$.
    - Compute the score $s_{k,j}$ of model m on corrupted data $\tilde{D}_{k,j}$.
  - Compute importance i_j for feature f_j defined as:

$$i_j = s - \frac{1}{K} \sum_{k=1}^{K} s_{k,j}$$

Originally proposed by Breiman (2001), extended by Fisher et al. (2019).

Source: scikit-learn documentation, permutation_importance function.

Permutation importance

```python
def permutation_test(model, X, y, num_reps=1, seed=42):
    """
    Run the permutation test for variable importance.
    Returns matrix of shape (X.shape[1], len(model.evaluate(X, y))).
    """
    rnd.seed(seed)
    scores = []

    for j in range(X.shape[1]):
        original_column = np.copy(X[:, j])
        col_scores = []

        for r in range(num_reps):
            rnd.shuffle(X[:,j])
            col_scores.append(model.evaluate(X, y, verbose=0))

        scores.append(np.mean(col_scores, axis=0))
        X[:,j] = original_column

    return np.array(scores)
```

LIME

Local Interpretable Model-agnostic Explanations employs an interpretable surrogate model to explain locally how the black-box model makes predictions for individual instances.

E.g. a black-box model predicts Bob's premium as the highest among all policyholders. LIME uses an interpretable model (a linear regression) to explain how Bob's features influence the black-box model's prediction.

See "Why Should I Trust You?": Explaining the Predictions of Any Classifier.

LIME Algorithm

Suppose we want to explain the instance $\boldsymbol{x}_{\text{Bob}} = (1, 2, 0.5)$.

1. Generate perturbed examples of $\boldsymbol{x}_{\text{Bob}}$ and use the trained gamma MDN f to make predictions:

$$\begin{aligned} \boldsymbol{x}'^{(1)}_{\text{Bob}} &= (1.1, 1.9, 0.6), \quad & f\left(\boldsymbol{x}'^{(1)}_{\text{Bob}}\right) &= 34000 \\ \boldsymbol{x}'^{(2)}_{\text{Bob}} &= (0.8, 2.1, 0.4), \quad & f\left(\boldsymbol{x}'^{(2)}_{\text{Bob}}\right) &= 31000 \\ &\vdots & &\vdots \end{aligned}$$

   We can then construct a dataset of N_{Examples} perturbed examples: $\mathcal{D}_{\text{LIME}} = \left(\left\{\boldsymbol{x}'^{(i)}_{\text{Bob}}, f\left(\boldsymbol{x}'^{(i)}_{\text{Bob}}\right)\right\}\right)_{i=0}^{N_{\text{Examples}}}$.

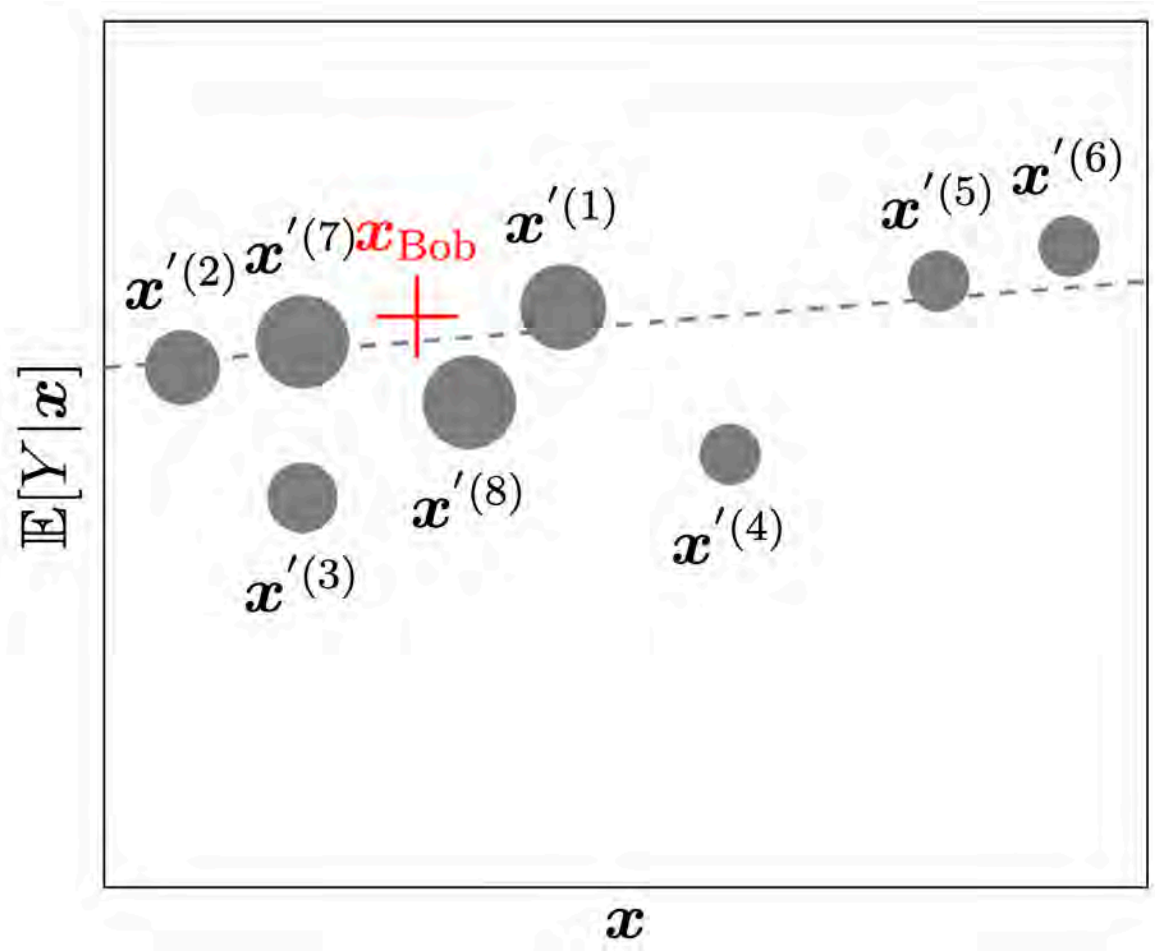
LIME Algorithm

2. Fit an interpretable model g, i.e., a linear regression using $\mathcal{D}_{\text{LIME}}$ and the following loss function:

$$\mathcal{L}_{\text{LIME}}(f, g, \pi_{\boldsymbol{x}_{\text{Bob}}}) = \sum_{i=1}^{N_{\text{Examples}}} \pi_{\boldsymbol{x}_{\text{Bob}}}\left(\boldsymbol{x}_{\text{Bob}}^{'(i)}\right) \cdot \left(f\left(\boldsymbol{x}_{\text{Bob}}^{'(i)}\right) - g\left(\boldsymbol{x}_{\text{Bob}}^{'(i)}\right)\right)^2,$$

where $\pi_{\boldsymbol{x}_{\text{Bob}}}\left(\boldsymbol{x}_{\text{Bob}}^{'(i)}\right)$ represents the distance from the perturbed example $\boldsymbol{x}_{\text{Bob}}^{'(i)}$ to the instance to be explained $\boldsymbol{x}_{\text{Bob}}$.

“Explaining” to Bob



The bold red cross is the instance being explained. LIME samples instances (grey nodes), gets predictions using f (gamma MDN) and weighs them by the proximity to the instance being explained (represented here by size). The dashed line g is the learned local explanation.

> “Again the approximation must be imperfect, otherwise one would throw out the black box and instead use the explanation as an inherently interpretable model.” (Rudin et al., 2022)

SHAP Values

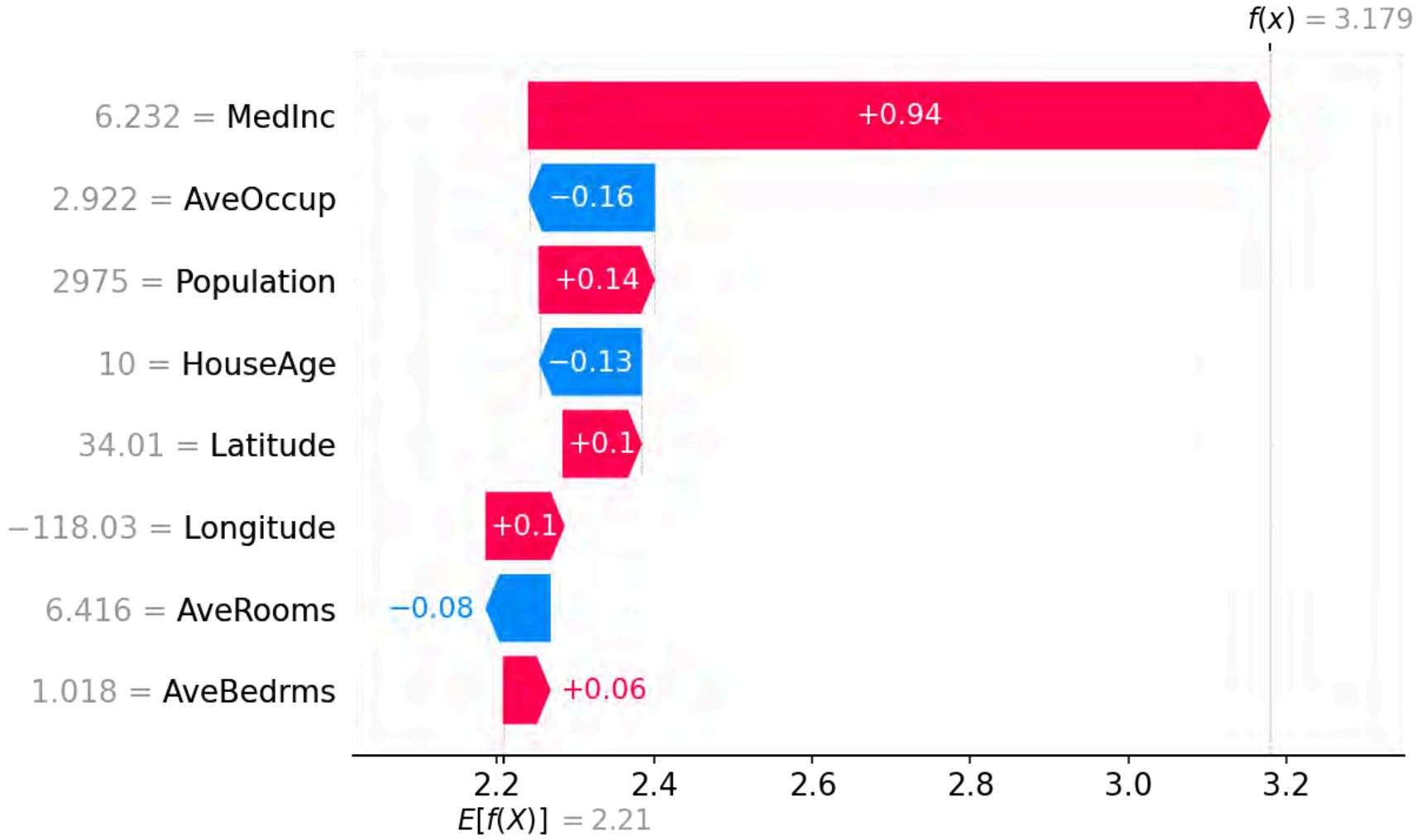
The SHapley Additive exPlanations (SHAP) value helps to quantify the contribution of each feature to the prediction for a specific instance (Lundberg & Lee, 2017).

The SHAP value for the jth feature is defined as

$$\text{SHAP}^{(j)}(\boldsymbol{x}) = \sum_{U \subset \{1,\ldots,p\} \setminus \{j\}} \frac{1}{p} \binom{p-1}{|U|}^{-1} \left(\mathbb{E}[Y|\boldsymbol{x}^{(U \cup \{j\})}] - \mathbb{E}[Y|\boldsymbol{x}^{(U)}] \right),$$

where p is the number of features. A positive SHAP value indicates that the variable increases the prediction value.

SHAP waterfall plot



SHAP waterfall plot

Source: `shap` package documentation, *An introduction to explainable AI with Shapley values*

Lecture Outline

- Introduction
- Illustrative Example
- Inherently Interpretable Models
- Post-hoc Explanations
- **Belgian Motor Dataset**
- Interpreting Inherently Interpretable Models
- Explaining a Neural Network with Partial Dependence Plots
- Explaining Specific Models

beMTPL97 dataset

```
data = pd.read_csv('data/raw/beMTPL97.csv')
claims = data.drop(columns = ["id", "claim", "amount", "average"])
claims.shape
```

```
(163212, 14)
```

```
postcode = claims.pop("postcode")
train_raw, test_raw = train_test_split(claims, test_size=0.2, random_state=2000, stratify

X_train_raw = train_raw.drop(columns='nclaims')
X_test_raw = test_raw.drop(columns='nclaims')
y_train_raw = train_raw['nclaims']
y_test_raw = test_raw['nclaims']

int_cols = X_train_raw.select_dtypes(include='integer').columns
X_train_raw[int_cols] = X_train_raw[int_cols].astype(float)
X_test_raw[int_cols] = X_test_raw[int_cols].astype(float)

num_vars = ['expo', 'ageph', 'bm', 'power', 'agec', 'lat', 'long']
cat_vars = ['coverage', 'sex', 'fuel', 'use', 'fleet']
```

Covariates

Numerical variables:

Variable	Description
`expo`	exposure to risk.
`ageph`	policyholder's age.
`bm`	An integer for the level on the former Belgian bonus-malus scale (0 to 22; higher = worse claims history).
`power`	vehicle's horsepower in kilowatts.
`agec`	vehicle's age in years.
`long`	longitude of the policyholder's municipality center.
`lat`	latitude of the policyholder's municipality center.

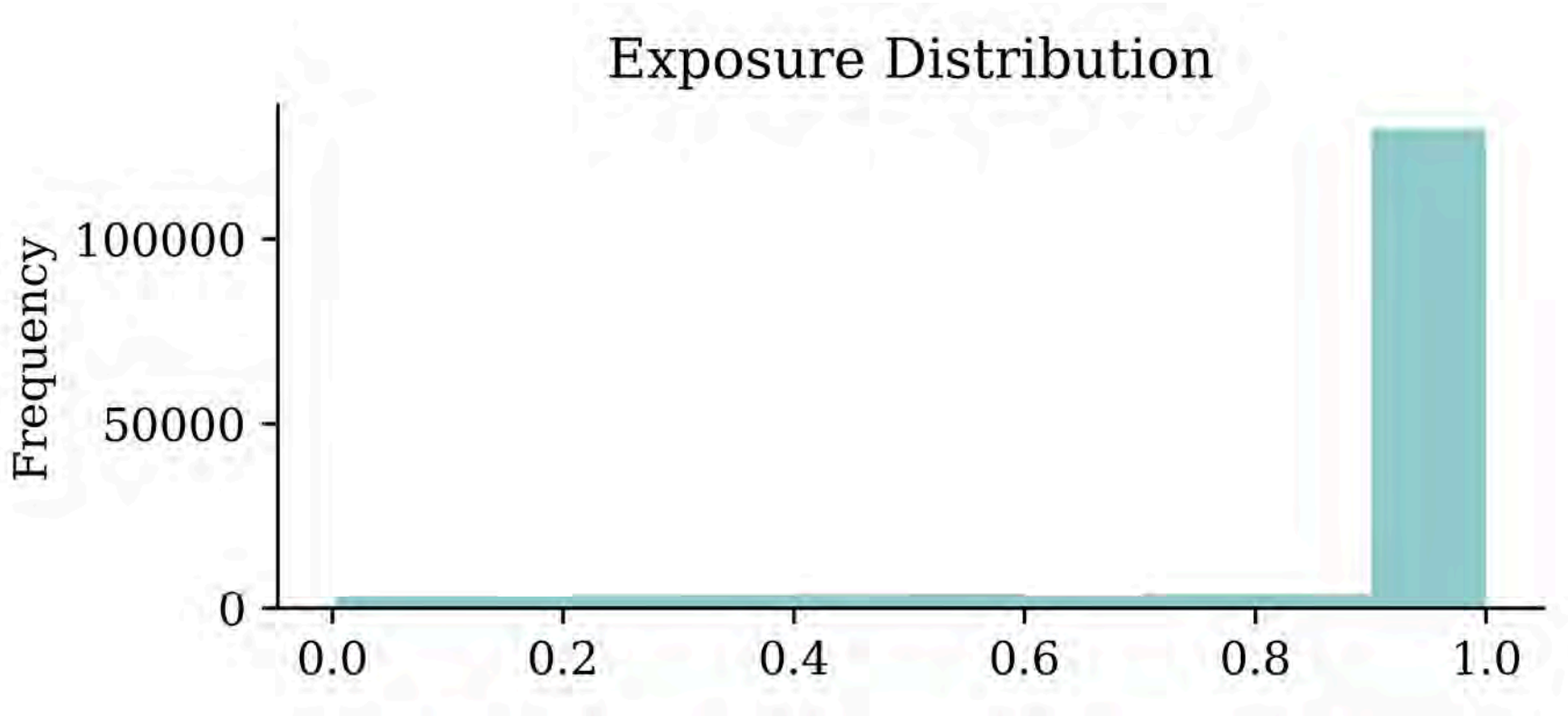
Target is `nclaims`.

Categorical variables:

Variable	Description
`coverage`	insurance coverage level: `"TPL"` (third party liability), `"TPL+"` (TPL + limited material damage), `"TPL++"` (TPL + comprehensive material damage).
`sex`	policyholder's gender: `"female"`, `"male"`.
`bm`	level on the former Belgian bonus-malus scale (0 to 22; higher = worse claims history).
`fuel`	vehicle's fuel type: `"gasoline"` or `"diesel"`.
`use`	vehicle's use: `"private"` or `"work"`.
`fleet`	vehicle is part of a fleet (`1` = yes, `0` = no).

Exposure & frequency

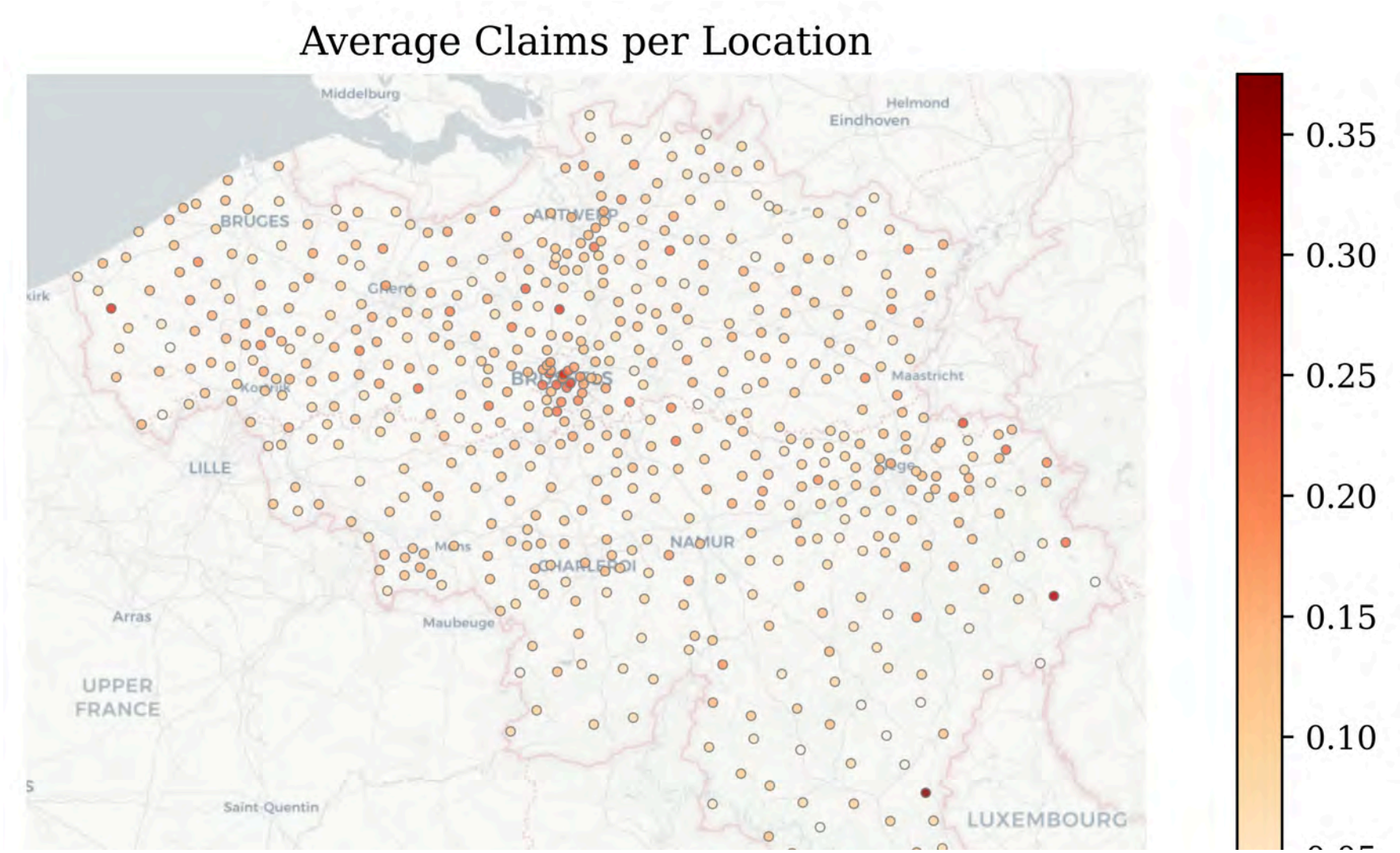
```
1  claims["expo"].plot(kind='hist', title='Exposure Distribution')
```




```
1  claims["nclaims"].value_counts()
```

```
nclaims
0    144936
1     16539
2      1556
```

Map of claims



Fitting using statsmodels

```
import statsmodels.api as sm
import statsmodels.formula.api as smf

formula = "nclaims ~ expo + ageph + bm + power + agec + lat + long " + \
    " + C(coverage) + C(sex) + C(fuel) + C(use) + C(fleet)"

glm_model = smf.glm(
    formula=formula,
    data=train_raw,
    family=sm.families.Poisson()
).fit()
```

Question

What do you expect to be the relationship between `ageph` and `nclaims`?

```
X_train_first = X_train_raw.iloc[0:1].copy()
X_train_first
```

	expo	coverage	ageph	sex	bm	power	agec	fuel	use	flee
25776	1.0	TPL	59.0	female	0.0	51.0	15.0	gasoline	private	0.0

Summary

```
1 glm_model.summary().tables[0]
```

Generalized Linear Model Regression Results

Dep. Variable:	nclaims	**No. Observations:**	130569
Model:	GLM	**Df Residuals:**	130555
Model Family:	Poisson	**Df Model:**	13
Link Function:	Log	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-49908.
Date:	Thu, 14 May 2026	**Deviance:**	69690.
Time:	19:15:47	**Pearson chi2:**	1.39e+05
No. Iterations:	6	**Pseudo R-squ. (CS):**	0.01842
Covariance Type:	nonrobust		

Summary

```
1 glm_model.summary().tables[1]
```

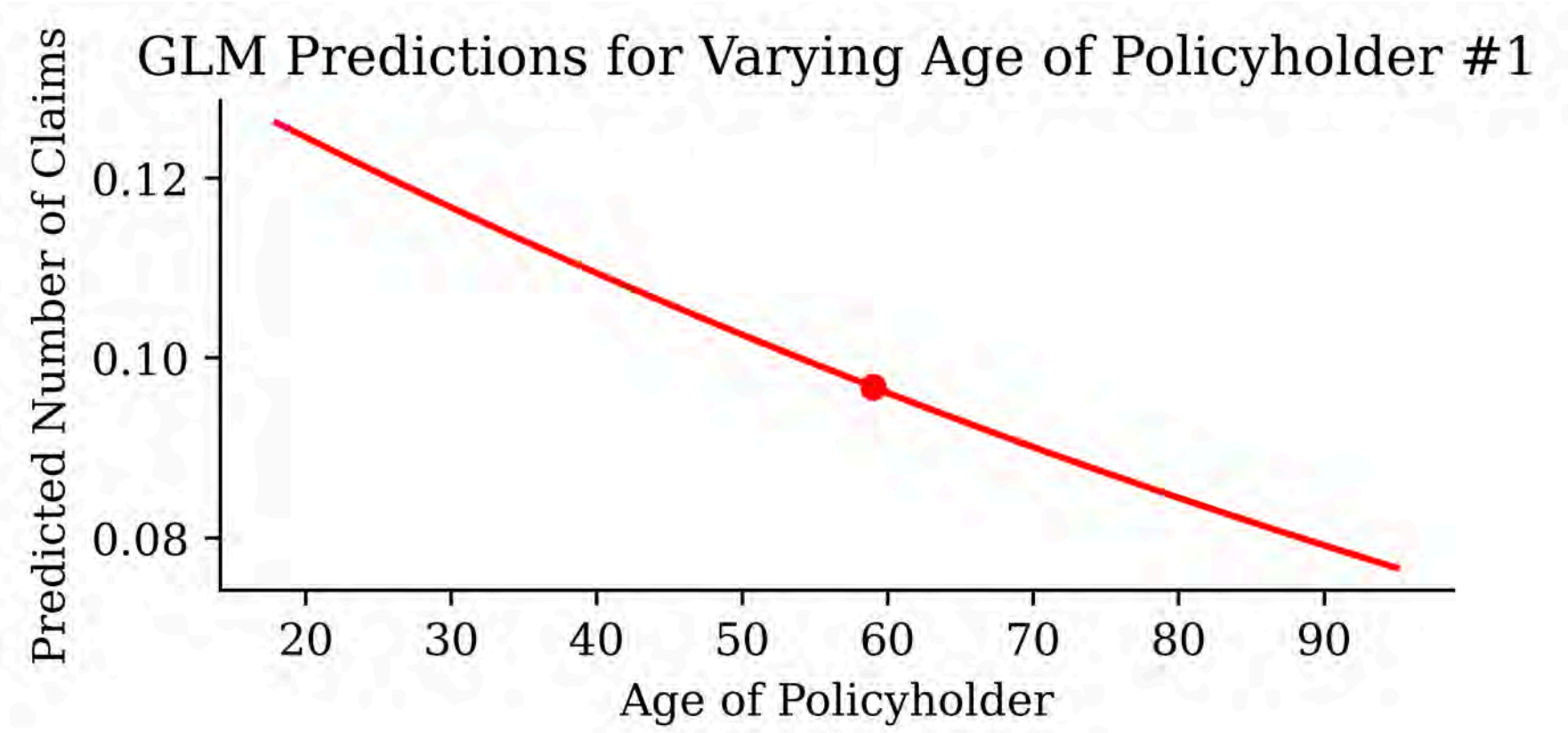
	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-7.0204	1.344	-5.222	0.000	-9.655	-4.386
C(coverage)[T.TPL+]	-0.0765	0.020	-3.889	0.000	-0.115	-0.038
C(coverage)[T.TPL++]	-0.0715	0.027	-2.660	0.008	-0.124	-0.019
C(sex)[T.male]	-0.0259	0.018	-1.429	0.153	-0.061	0.010
C(fuel)[T.gasoline]	-0.1793	0.017	-10.459	0.000	-0.213	-0.146
C(use)[T.work]	-0.0864	0.037	-2.306	0.021	-0.160	-0.013
C(fleet)[T.1]	-0.0886	0.048	-1.832	0.067	-0.183	0.006
expo	0.9893	0.041	24.196	0.000	0.909	1.069
ageph	-0.0065	0.001	-10.724	0.000	-0.008	-0.005
bm	0.0642	0.002	33.035	0.000	0.060	0.068
power	0.0036	0.000	8.309	0.000	0.003	0.004
agec	-0.0019	0.002	-0.872	0.383	-0.006	0.002
lat	0.0777	0.026	2.969	0.003	0.026	0.129
long	0.0279	0.011	2.542	0.011	0.006	0.049

Lecture Outline

- Introduction
- Illustrative Example
- Inherently Interpretable Models
- Post-hoc Explanations
- Belgian Motor Dataset
- **Interpreting Inherently Interpretable Models**
- Explaining a Neural Network with Partial Dependence Plots
- Explaining Specific Models

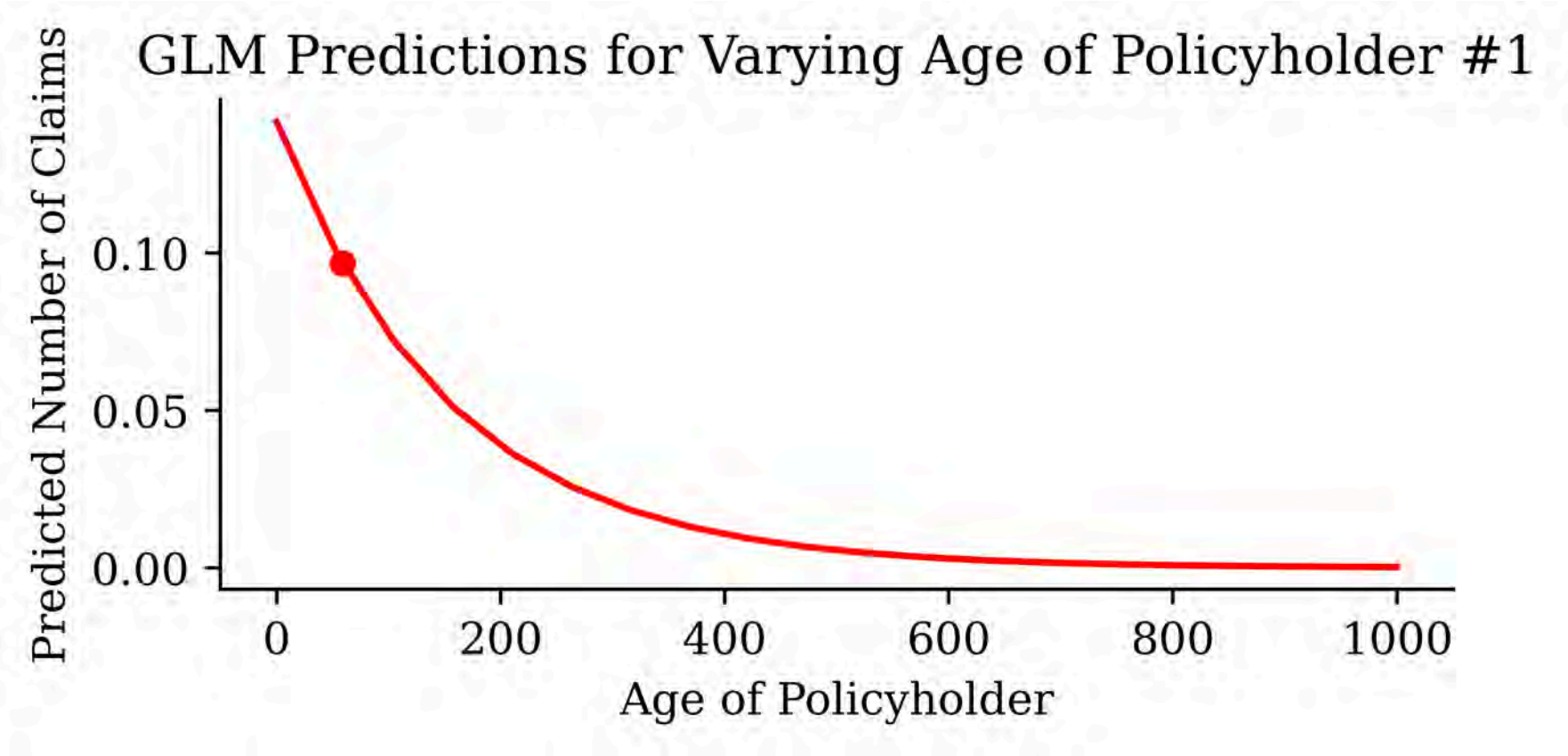
GLM Ceteris Paribus Plots

```python
ageph_range = np.linspace(train_raw['ageph'].min(), train_raw['ageph'].max(), 20)
y_pred_batch = []
for ageph in ageph_range:
    X_train_first['ageph'] = ageph
    y_pred_batch.append(glm_model.predict(X_train_first))
```

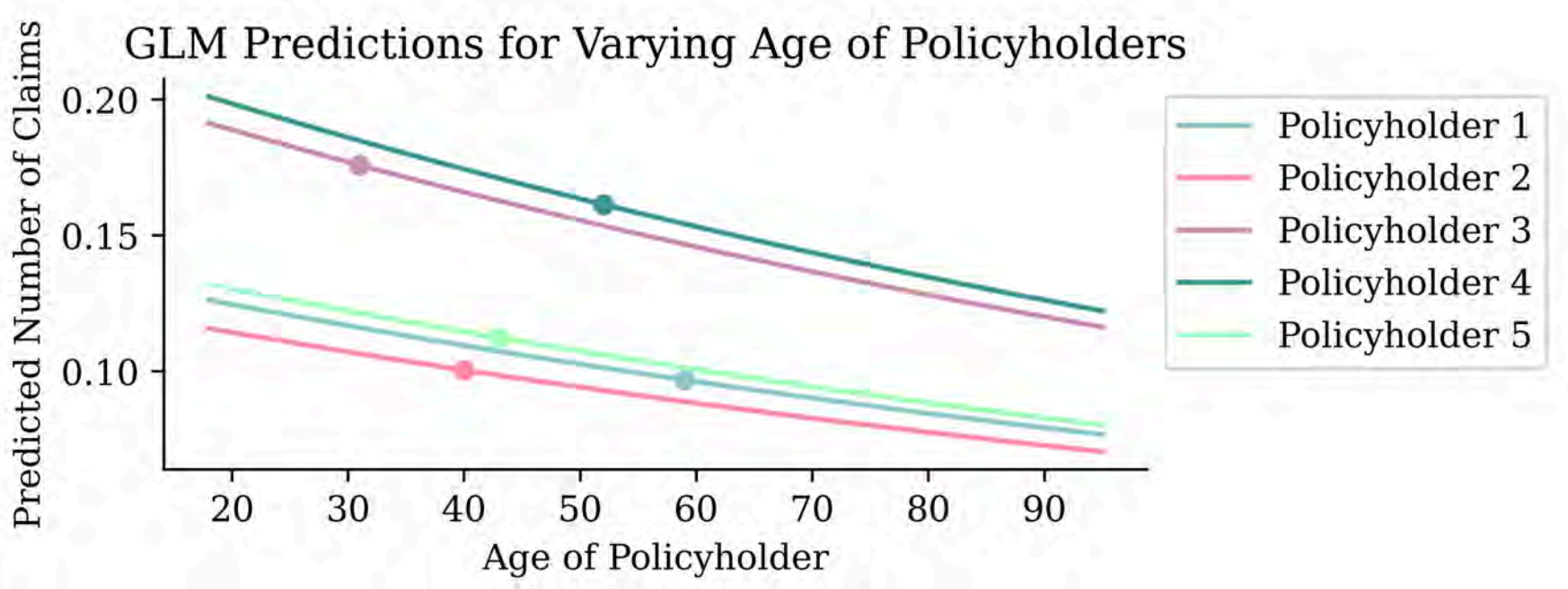


Zoomed out

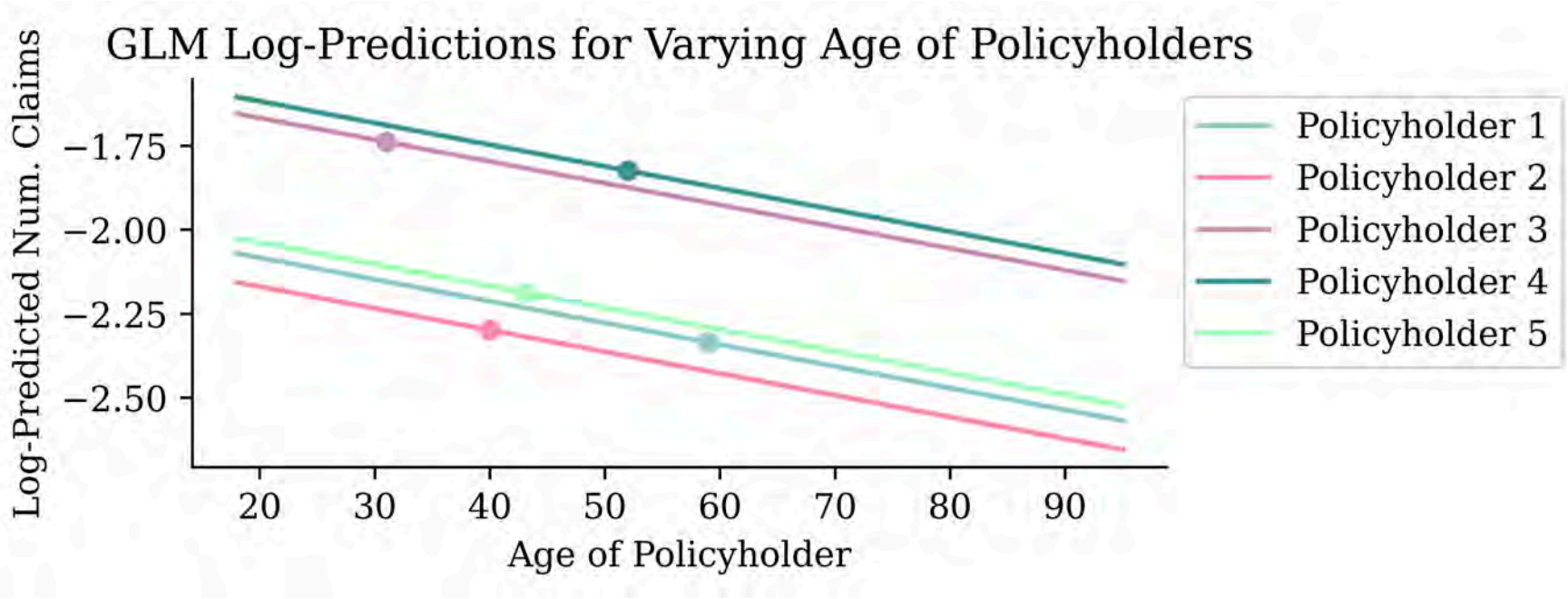
We can see the trend if we zoom out to ages between 0 & 1,000.



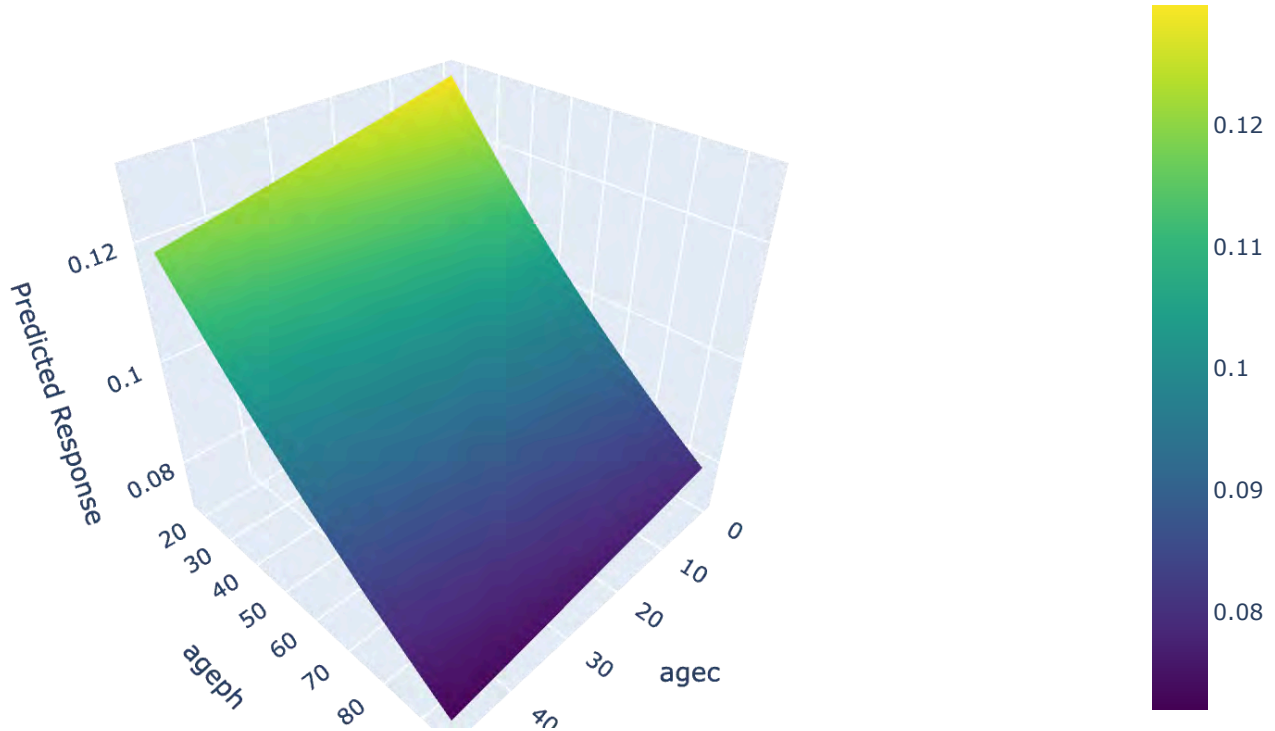
What if we look at multiple policyholders?



Logarithmic scale



Do it for ageph and agec



Generalised Additive Models (GAMs)

Transform a GLM's inputs nonlinearly, i.e.,

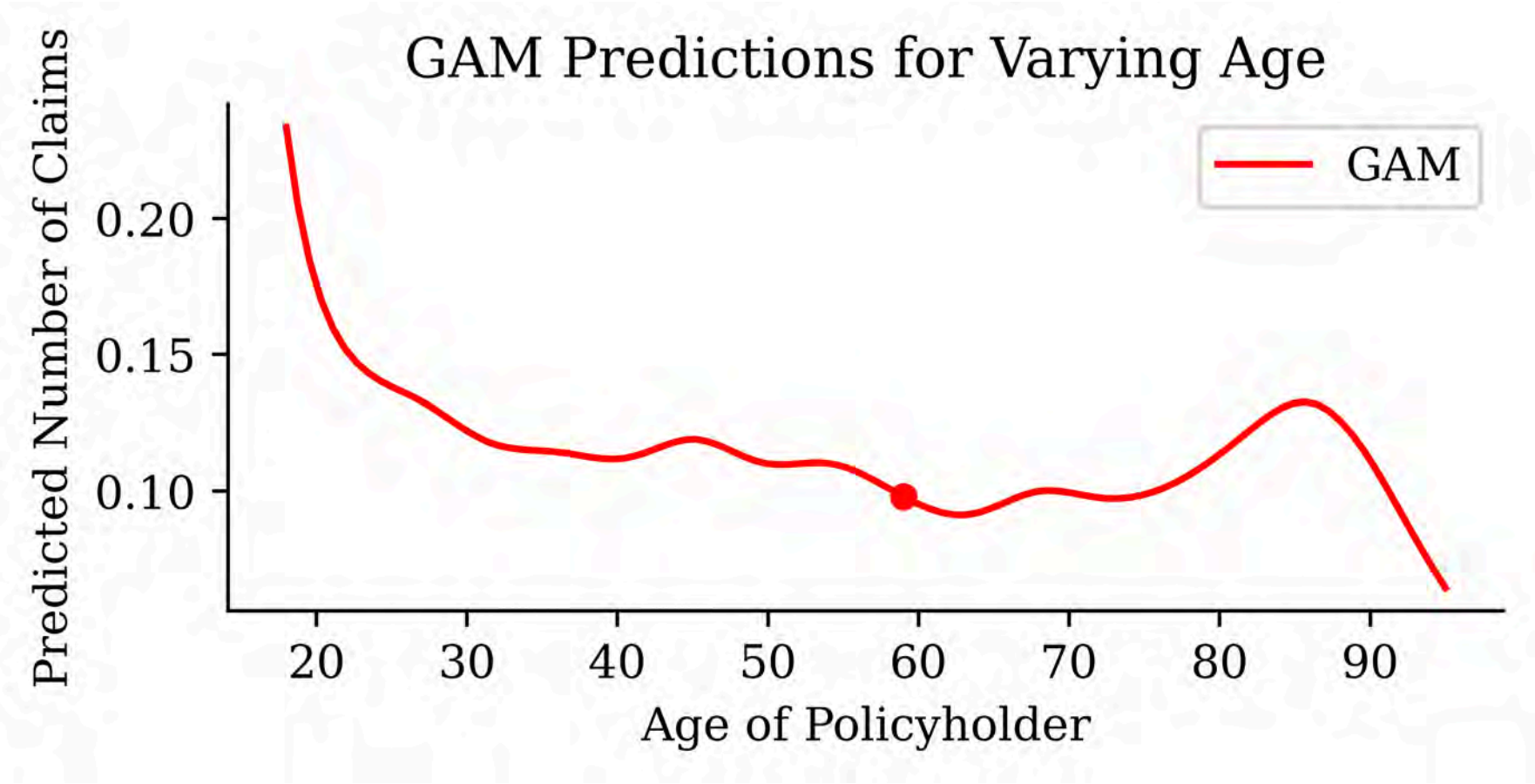
$$\hat{y}_i = g^{-1}\left(\beta_0 + f_1(x_{i,1}) + f_2(x_{i,2}) + \ldots + f_p(x_{i,p})\right)$$

The f_j are typically polynomials, step functions, or splines.

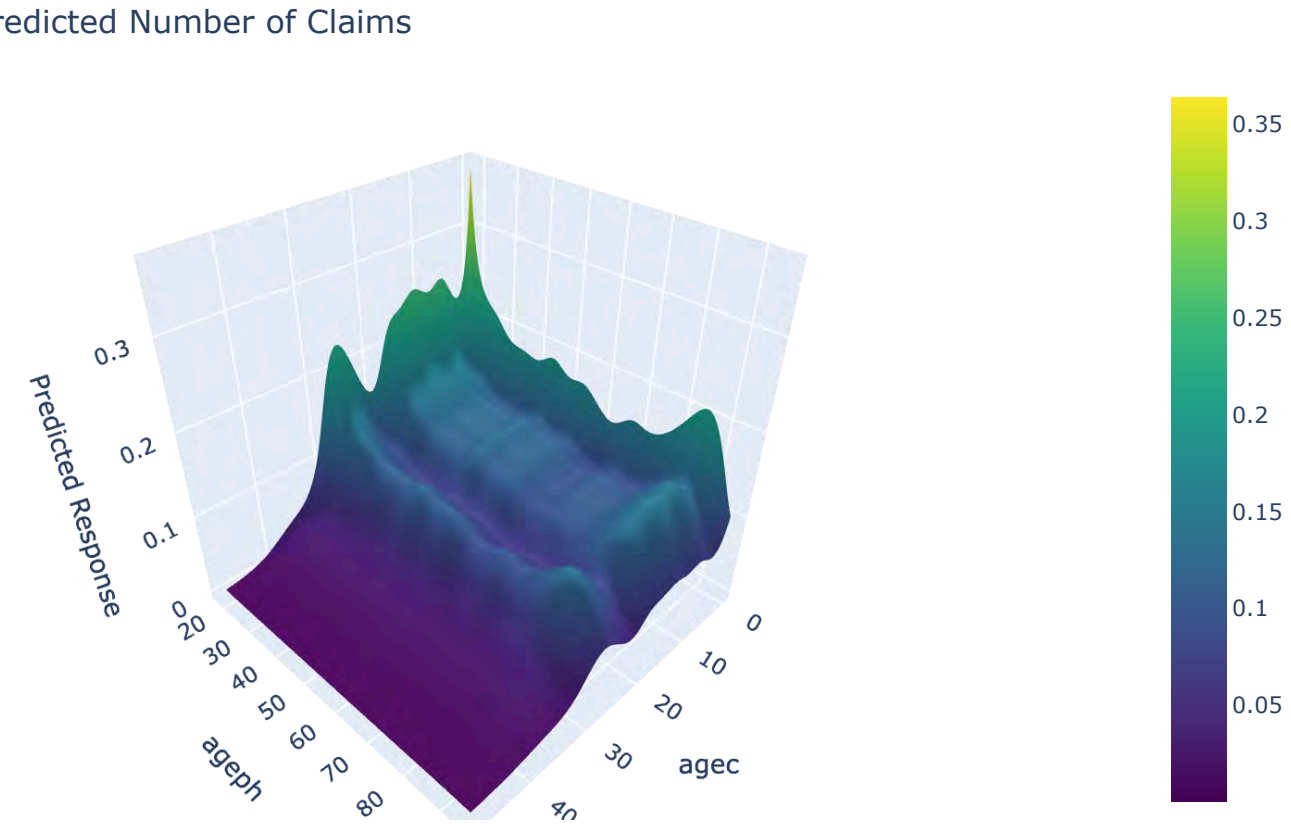
```
ct = make_column_transformer(
    ("passthrough", num_vars),
    (OrdinalEncoder(), cat_vars),
    verbose_feature_names_out = False
)
X_train_gam = ct.fit_transform(X_train_raw)
X_test_gam = ct.transform(X_test_raw)
y_train_gam = y_train_raw.values
y_test_gam = y_test_raw.values
```

```
from pygam import PoissonGAM, s, f
formula = s(0) + s(1) + s(2) + s(3) + s(4) + s(5) + s(6) + \
    f(7) + f(8) + f(9) + f(10) + f(11)
gam_model = PoissonGAM(formula).fit(X_train_gam, y_train_gam)
```

Ceteris paribus plot for ageph



GAM bivariate effects for ageph and agec



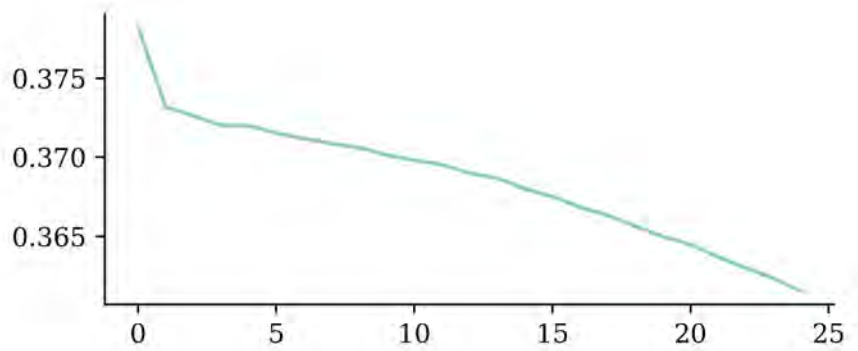
Lecture Outline

- Introduction
- Illustrative Example
- Inherently Interpretable Models
- Post-hoc Explanations
- Belgian Motor Dataset
- Interpreting Inherently Interpretable Models
- **Explaining a Neural Network with Partial Dependence Plots**
- Explaining Specific Models

Build a neural network

```
ct = make_column_transformer(
    (StandardScaler(), num_vars),
    (OneHotEncoder(drop="first", sparse_c
    verbose_feature_names_out=False
)
ct.fit(X_train_raw)

X_train_nn = ct.transform(X_train_raw)
X_test_nn = ct.transform(X_test_raw)
y_train_nn = y_train_raw.values
y_test_nn = y_test_raw.values


random.seed(123)

nn_model = Sequential(
    [
        Input(shape=(X_train_nn.shape[1],
        Dense(128, activation="relu"),
        Dense(128, activation="relu"),
        Dense(128, activation="relu"),
        Dense(1, activation="exponential"
    ]
)
```

```
nn_model.compile(
    optimizer="adam",
    loss="poisson",
    metrics=["mae"]
)

history = nn_model.fit(X_train_nn, y_trai
    epochs=25, batch_size=64, verbose=0,
)
```



UNSW SYDNEY

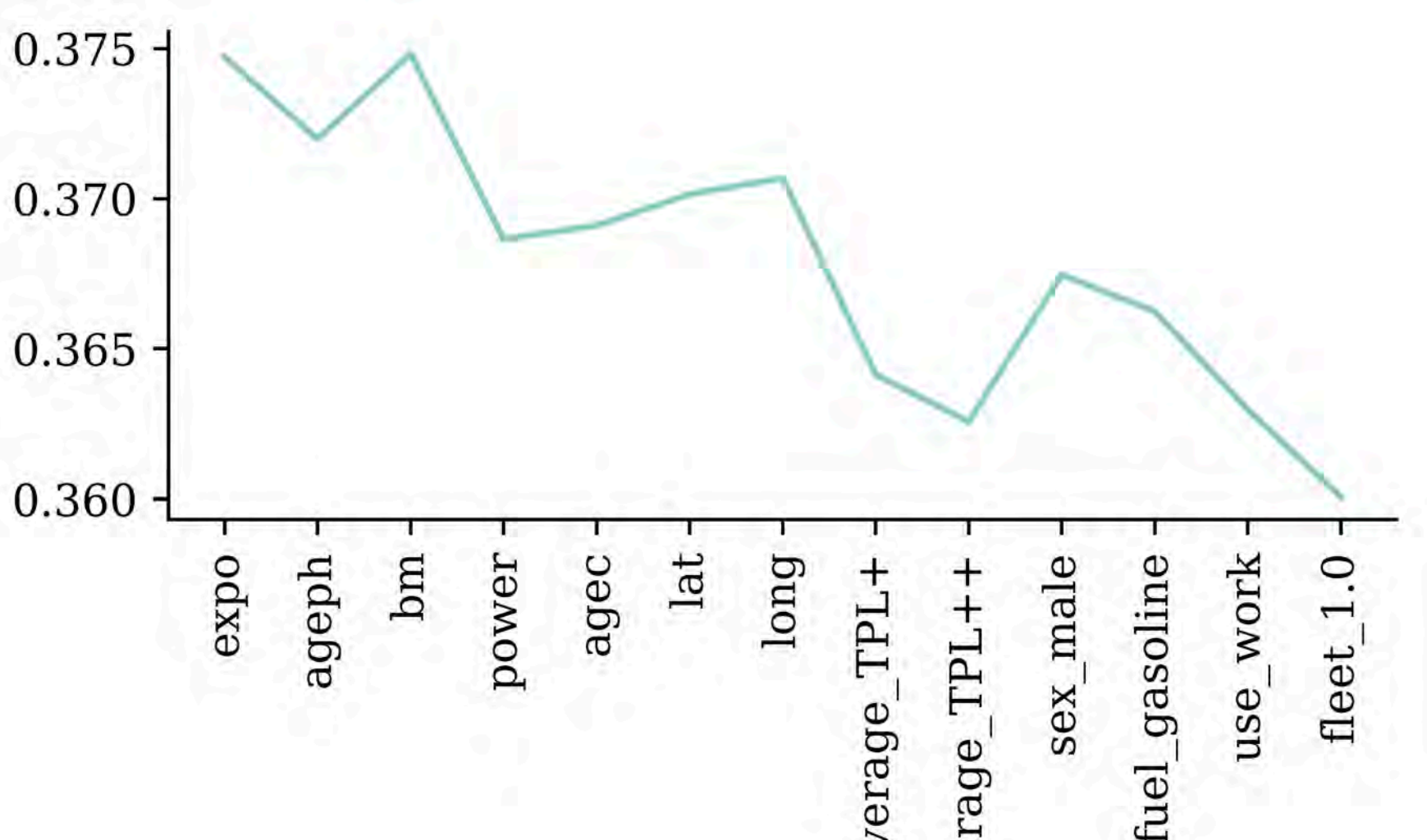
Metrics

```python
from sklearn.metrics import mean_poisson_deviance, mean_absolute_error

# Evaluate the neural network model
y_pred_nn = nn_model.predict(X_test_nn, verbose=0, batch_size=X_test_nn.shape[0]).flatten
print(f"NN mean Poisson deviance  : {mean_poisson_deviance(y_test_nn, y_pred_nn):.4f}")
print(f"NN MAE                    : {mean_absolute_error(y_test_nn, y_pred_nn):.4f}")

# Compare to the GAM
y_pred_gam = gam_model.predict(X_test_gam).ravel()
print(f"GAM mean Poisson deviance : {mean_poisson_deviance(y_test_gam, y_pred_gam):.4f}")
print(f"GAM MAE                   : {mean_absolute_error(y_test_gam, y_pred_gam):.4f}")

## Compare to the GLM
y_pred_glm = glm_model.predict(X_test_raw).values.ravel()
print(f"GLM mean Poisson deviance : {mean_poisson_deviance(y_test_raw, y_pred_glm):.4f}")
print(f"GLM MAE                   : {mean_absolute_error(y_test_raw, y_pred_glm):.4f}")
```

```
NN mean Poisson deviance  : 0.5449
NN MAE                    : 0.2212
GAM mean Poisson deviance : 0.5254
GAM MAE                   : 0.2149
GLM mean Poisson deviance : 0.5326
GLM MAE                   : 0.2164
```

Permutation importance example

```
scores = permutation_test(nn_model, X_train_nn.values, y_train_nn)
plt.plot(scores[:,0], label='Loss')
plt.xticks(ticks=np.arange(len(X_train_nn.columns)), labels=X_train_nn.columns, rotation=
```



PDP: Training data

Partial dependence plots start by looking at the training data.

```
1 X_train_raw
```

	expo	coverage	ageph	sex	bm	power	agec	fuel	use	fleet	long	lat
25776	1.000000	TPL	59.0	female	0.0	51.0	15.0	gasoline	private	0.0	4.387146	51.216042
63185	0.819178	TPL++	40.0	female	0.0	96.0	3.0	gasoline	private	0.0	5.500567	50.583188
130175	1.000000	TPL	31.0	female	8.0	40.0	8.0	gasoline	private	0.0	3.721116	50.535314
...	...	...	...	...	...	...	...	...	...	...	...	...
24617	1.000000	TPL	75.0	male	0.0	29.0	17.0	gasoline	private	0.0	4.387146	51.216042
61581	1.000000	TPL	63.0	male	0.0	55.0	11.0	gasoline	private	0.0	5.612566	50.680020
10699	1.000000	TPL+	50.0	male	6.0	74.0	3.0	gasoline	private	0.0	4.678745	50.687562

130569 rows × 12 columns

```
1 nn_model.predict(ct.transform(X_train_raw), verbose=0).mean()
```

```
np.float32(0.13129403)
```

UNSW SYDNEY

PDP: Alternate Reality

```
X_train_pd = X_train_raw.copy()
X_train_pd['ageph'] = 18
X_train_pd
```

	expo	coverage	ageph	sex	bm	power	agec	fuel	use	fleet	long	lat
25776	1.000000	TPL	18	female	0.0	51.0	15.0	gasoline	private	0.0	4.387146	51.216042
63185	0.819178	TPL++	18	female	0.0	96.0	3.0	gasoline	private	0.0	5.500567	50.583188
130175	1.000000	TPL	18	female	8.0	40.0	8.0	gasoline	private	0.0	3.721116	50.535314
...	...	...	...	...	...	...	...	...	...	...	...	...
24617	1.000000	TPL	18	male	0.0	29.0	17.0	gasoline	private	0.0	4.387146	51.216042
61581	1.000000	TPL	18	male	0.0	55.0	11.0	gasoline	private	0.0	5.612566	50.680020
10699	1.000000	TPL+	18	male	6.0	74.0	3.0	gasoline	private	0.0	4.678745	50.687562

130569 rows × 12 columns

```
nn_model.predict(ct.transform(X_train_pd), verbose=0).mean()
```

```
np.float32(0.1809422)
```

UNSW SYDNEY

PDP: Alternate Reality

```
X_train_pd = X_train_raw.copy()
X_train_pd['ageph'] = 19
X_train_pd
```

	expo	coverage	ageph	sex	bm	power	agec	fuel	use	fleet	long	lat
25776	1.000000	TPL	19	female	0.0	51.0	15.0	gasoline	private	0.0	4.387146	51.216042
63185	0.819178	TPL++	19	female	0.0	96.0	3.0	gasoline	private	0.0	5.500567	50.583188
130175	1.000000	TPL	19	female	8.0	40.0	8.0	gasoline	private	0.0	3.721116	50.535314
...	...	...	...	...	...	...	...	...	...	...	...	...
24617	1.000000	TPL	19	male	0.0	29.0	17.0	gasoline	private	0.0	4.387146	51.216042
61581	1.000000	TPL	19	male	0.0	55.0	11.0	gasoline	private	0.0	5.612566	50.680020
10699	1.000000	TPL+	19	male	6.0	74.0	3.0	gasoline	private	0.0	4.678745	50.687562

130569 rows × 12 columns

```
nn_model.predict(ct.transform(X_train_pd), verbose=0).mean()
```

```
np.float32(0.17807394)
```

UNSW SYDNEY

PDP: Alternate Reality

```
X_train_pd = X_train_raw.copy()
X_train_pd['ageph'] = 90
X_train_pd
```

	expo	coverage	ageph	sex	bm	power	agec	fuel	use	fleet	long	lat
25776	1.000000	TPL	90	female	0.0	51.0	15.0	gasoline	private	0.0	4.387146	51.216042
63185	0.819178	TPL++	90	female	0.0	96.0	3.0	gasoline	private	0.0	5.500567	50.583188
130175	1.000000	TPL	90	female	8.0	40.0	8.0	gasoline	private	0.0	3.721116	50.535314
...	...	...	...	...	...	...	...	...	...	...	...	...
24617	1.000000	TPL	90	male	0.0	29.0	17.0	gasoline	private	0.0	4.387146	51.216042
61581	1.000000	TPL	90	male	0.0	55.0	11.0	gasoline	private	0.0	5.612566	50.680020
10699	1.000000	TPL+	90	male	6.0	74.0	3.0	gasoline	private	0.0	4.678745	50.687562

130569 rows × 12 columns

```
nn_model.predict(ct.transform(X_train_pd), verbose=0).mean()
```

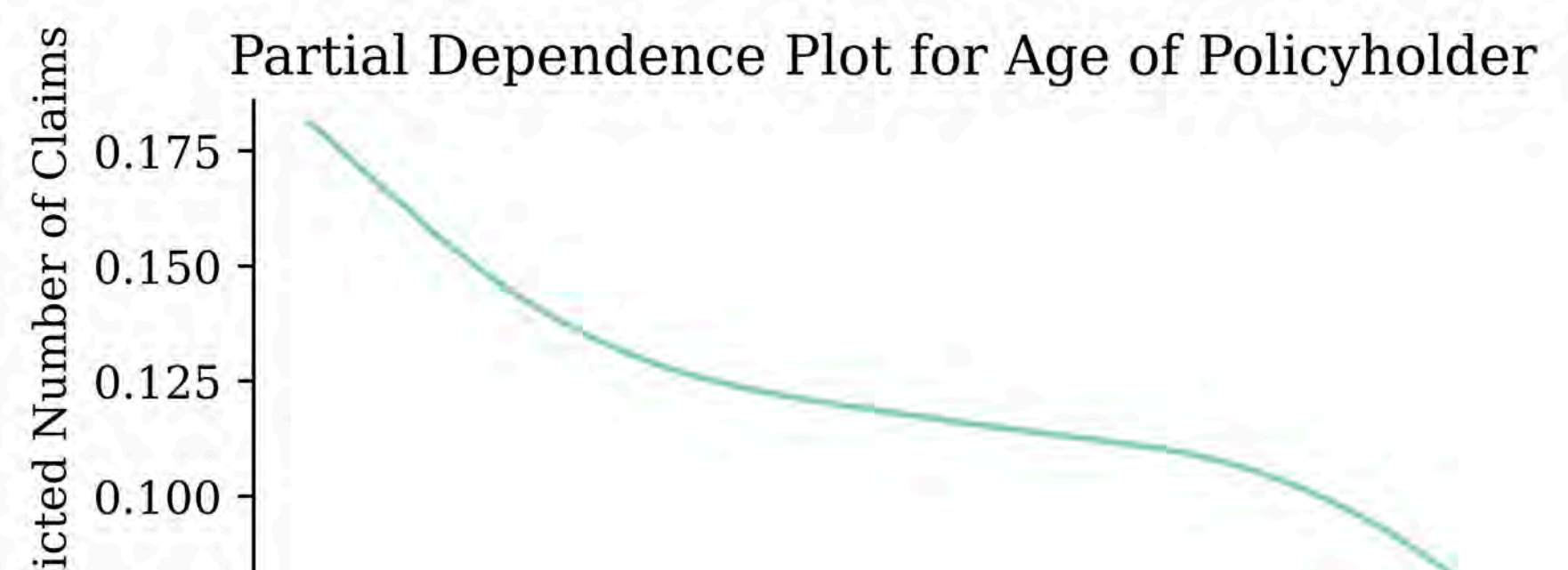
```
np.float32(0.093187734)
```

Question

What could go wrong?

Partial dependence plots

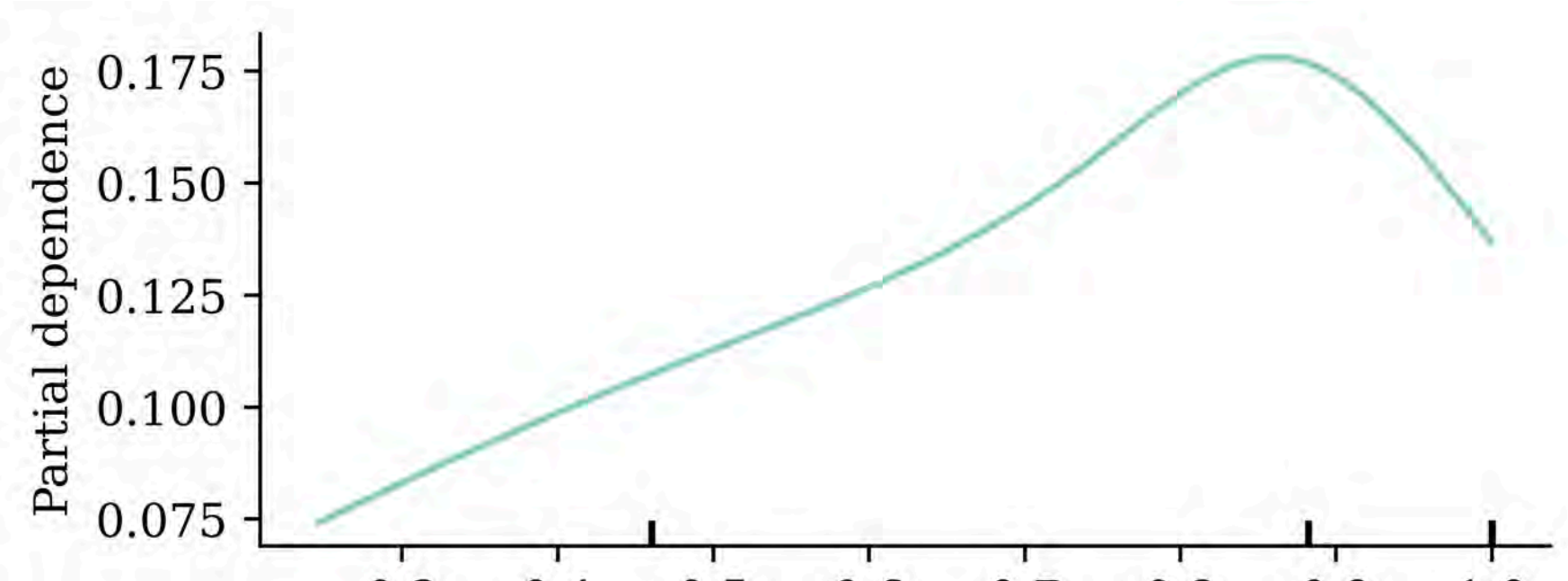
```
# Make partial dependence plot for ageph by hand
ageph_range = np.linspace(X_train_raw['ageph'].min(), X_train_raw['ageph'].max(), 100)
y_preds = []
X_train_pd = X_train_raw.copy()

for age in ageph_range:
    X_train_pd['ageph'] = age
    X_train_pd_ct = ct.transform(X_train_pd)
    y_pred_batch = nn_model.predict(X_train_pd_ct, verbose=0, batch_size=X_train_pd_ct.sh
    y_preds.append(y_pred_batch.mean())
```



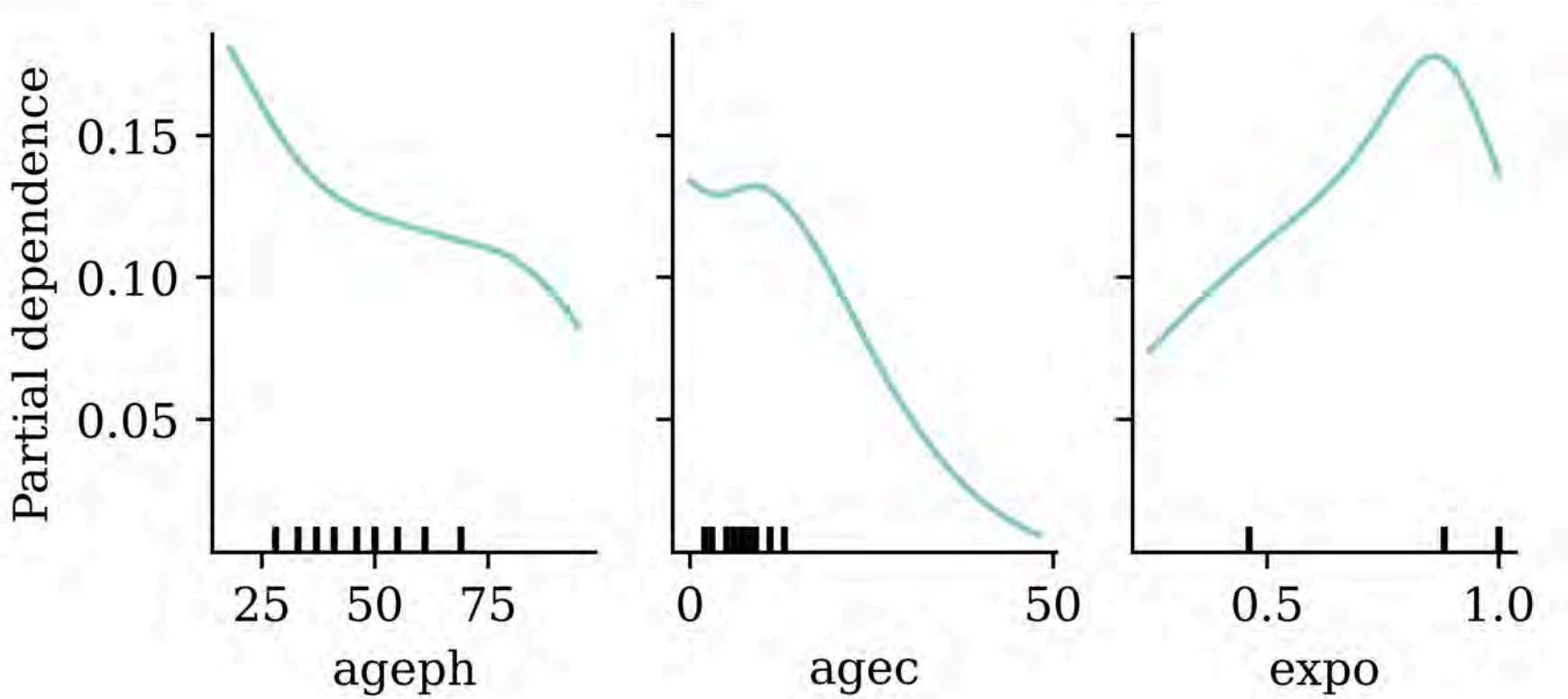
Using scikit-learn's PDP

(First need to trick scikit-learn a little..)

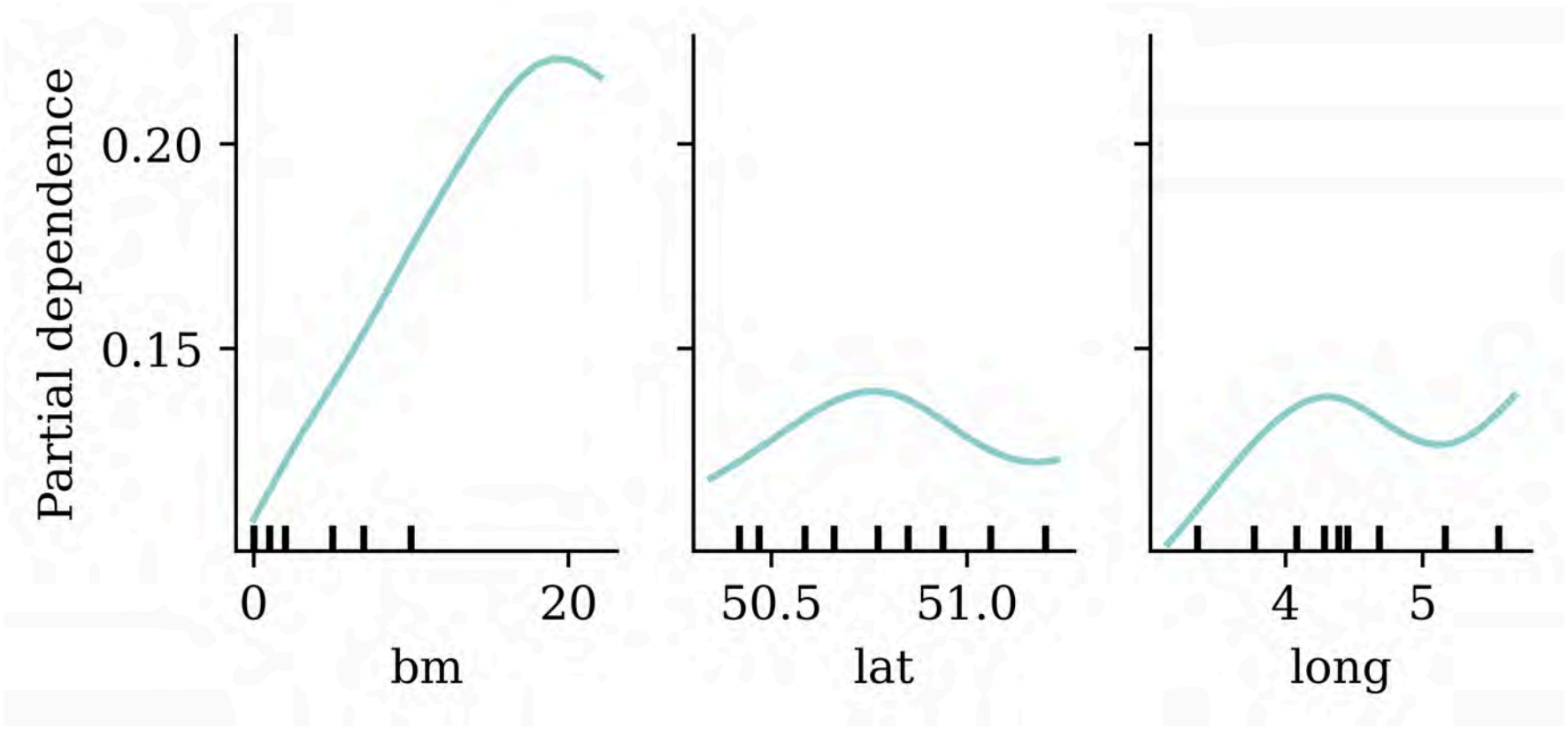
```python
from sklearn.inspection import PartialDependenceDisplay

PartialDependenceDisplay.from_estimator(
    nn_model_skl,
    X_train_raw,
    features=[0],
    feature_names=X_train_raw.columns,
)
```



Explaining the ages and exposure

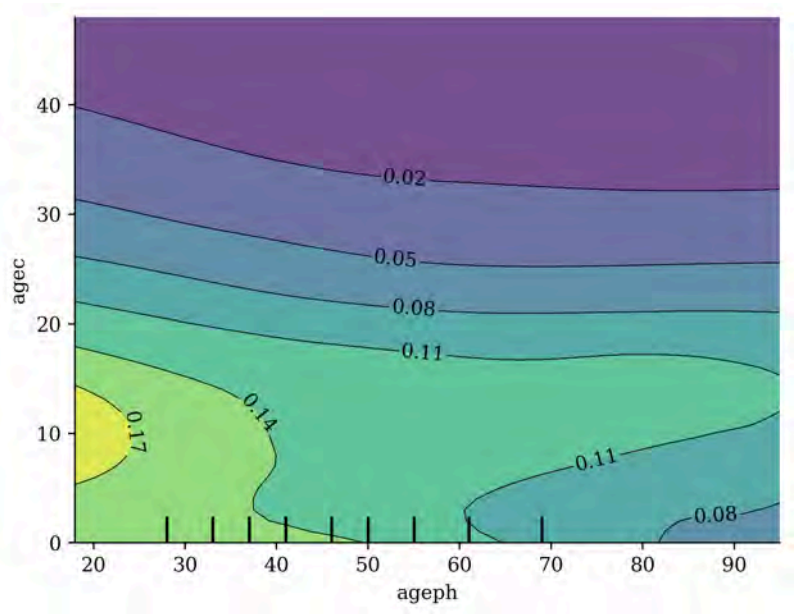


Explaining the bonus-malus and location



Bivariate age effects

```
cols = X_train_raw.columns.to_list()
inds = [cols.index(col) for col in ["ageph", "agec"]]

disp = PartialDependenceDisplay.from_estimator(
    nn_model_skl,
    X_train_raw,
    features=[tuple(inds)],
    feature_names=X_train_raw.columns,
)
```

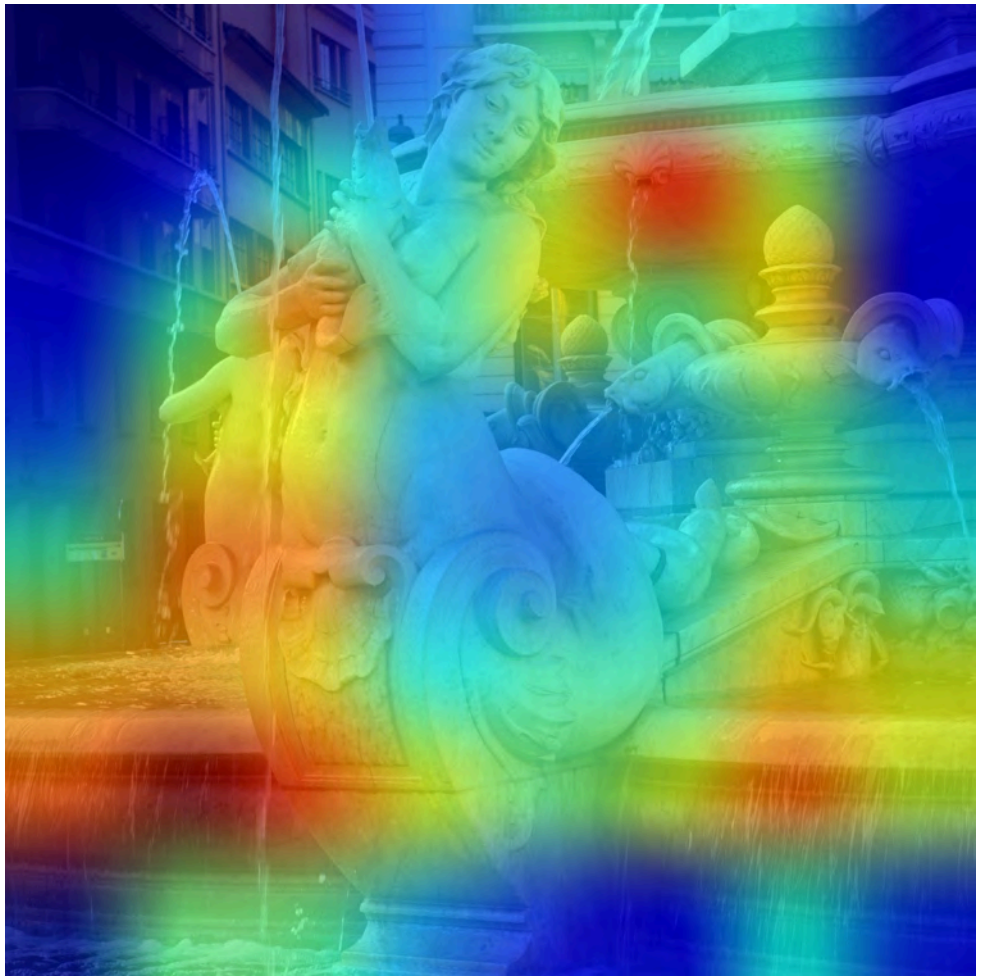


Lecture Outline

- Introduction
- Illustrative Example
- Inherently Interpretable Models
- Post-hoc Explanations
- Belgian Motor Dataset
- Interpreting Inherently Interpretable Models
- Explaining a Neural Network with Partial Dependence Plots
- **Explaining Specific Models**

Grad-CAM

Original image



Grad-CAM

See, e.g., Keras tutorial.

See Chollet (2021), Deep Learning with Python, Section 9.4.3.

Criticism

> “Rather than trying to create models that are inherently interpretable, there has been a recent explosion of work on ‘explainable ML’, where a second (post hoc) model is created to explain the first black box model. This is problematic. Explanations are often not reliable, and can be misleading, as we discuss below. If we instead use models that are inherently interpretable, they provide their own explanations, which are faithful to what the model actually computes.” (Rudin, 2019)

Criticism II

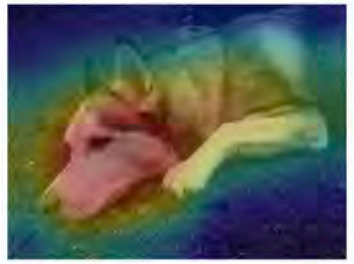
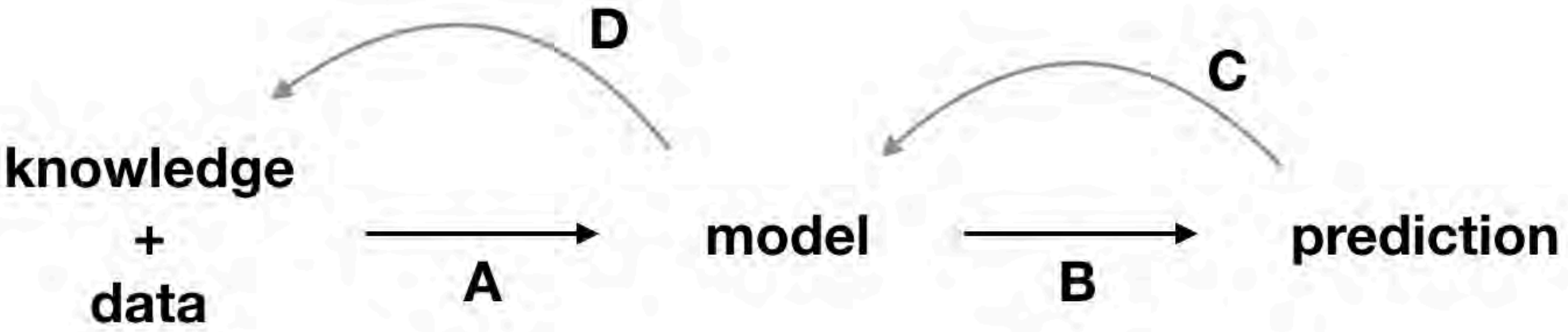
	Test Image	Evidence for Animal Being a Siberian Husky	Evidence for Animal Being a Transverse Flute
Explanations Using Attention Maps			

Figure 2: Saliency does not explain anything except where the network is looking. We have no idea why this image is labeled as either a dog or a musical instrument when considering only saliency. The explanations look essentially the same for both classes. Figure credit: Chaofan Chen and [28].

Multiple conflicting explanations (Rudin, 2019)

Source: Rudin (2019), Stop explaining black box machine learning models for high stakes decisions and use interpretable models intstead, preprint available under Creative Commons.

Conclusion



"Figure 20.2: Explainability techniques allow strengthening the feedback extracted from a model. A, data and domain knowledge allow building the model. B, predictions are obtained from the model. C, by analyzing the predictions, we learn more about the model. D, better understanding of the model allows better understanding of the data and, sometimes, broadens domain knowledge." Biecek & Burzykowski (2021)

Package Versions

```
Python implementation: CPython
Python version       : 3.14.3
IPython version      : 9.13.0

contextily : 1.7.0
geopandas  : 1.1.3
keras      : 3.14.1
matplotlib : 3.10.9
numpy      : 2.4.4
pandas     : 3.0.2
plotly     : 6.7.0
pygam      : 0.9.1
seaborn    : 0.13.2
scipy      : 1.17.1
shapely    : 2.1.2
sklearn    : 1.8.0
```

Glossary

- adversarial examples
- ceteris paribus plot
- global interpretability
- Grad-CAM
- individual conditional expectation (ICE) plot
- inherent interpretability
- LIME
- local interpretability
- partial dependence plot
- permutation importance
- post-hoc explainability
- SHAP values

References

Ben-Zion, Z., Witte, K., Jagadish, A. K., Duek, O., Harpaz-Rotem, I., Khorsandian, M.-C., Burrer, A., Seifritz, E., Homan, P., Schulz, E., et al. (2025). Assessing and alleviating state anxiety in large language models. *Npj Digital Medicine*, *8*(1), 132.

Biecek, P., & Burzykowski, T. (2021). *Explanatory Model Analysis*. Chapman; Hall/CRC, New York. https://pbiecek.github.io/ema/

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Carreira-Perpinán, M. A., & Tavallali, P. (2018). Alternating optimization of decision trees, with application to learning sparse oblique trees. *Advances in Neural Information Processing Systems*, *31*.

Charpentier, A. (2024). *Insurance, biases, discrimination and fairness*. Springer.

Chen, Z., Lu, Y., Zhang, J., & Zhu, W. (2024). Managing weather risk with a neural network-based index insurance. *Management Science*, *70*(7), 4306–4327.

Delcaillau, D., Ly, A., Papp, A., & Vermet, F. (2022). Model transparency and interpretability: Survey and application to the insurance industry. *European Actuarial Journal*, *12*(2), 443–484.

Doshi-Velez, F., & Kim, B. (2017). *Towards a rigorous science of interpretable machine learning*.

Fisher, A., Rudin, C., & Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, *20*(177), 1–81.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv Preprint arXiv:1412.6572*.

Li, C., Wang, J., Zhang, Y., Zhu, K., Hou, W., Lian, J., Luo, F., Yang, Q., & Xie, X. (2023). Large language models understand and can be enhanced by emotional stimuli. *arXiv Preprint arXiv:2307.11760*.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*.

Molnar, C. (2020). *Interpretable machine learning*.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

Richman, R., & Wüthrich, M. V. (2023). LocalGLMnet: Interpretable deep learning for tabular data. *Scandinavian Actuarial*