



Machine Learning: Model Interpretability and Risk Assessment

Aijun Zhang, Ph.D.

Corporate Model Risk, Wells Fargo

Sharing with Chinese American Data Science & Engineering Association (CADSEA) | Sep 10, 2023

Disclaimer: This material represents the views of the presenter and does not necessarily reflect those of Wells Fargo.

Biographical Sketch

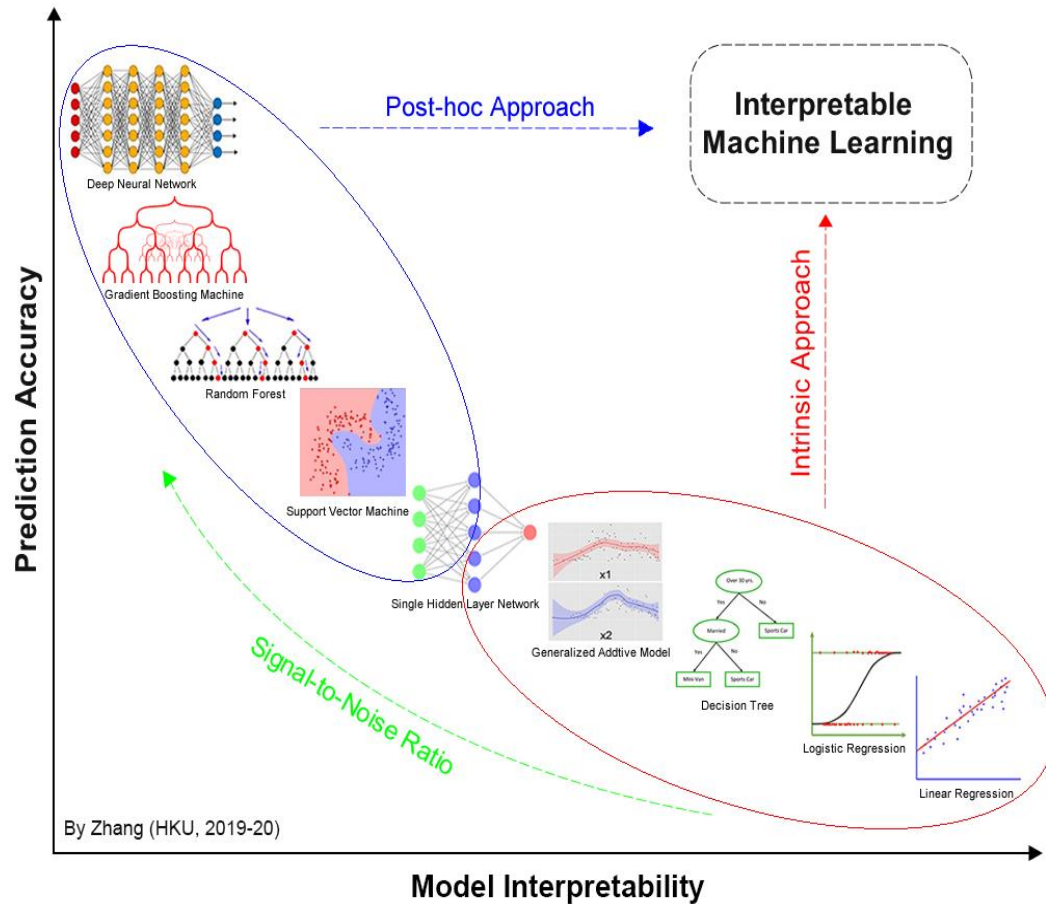


- Aijun Zhang is a senior vice president, quantitative analytics manager with Wells Fargo. He leads a machine learning & validation engineering team at Corporate Model Risk, responsible for a PiML toolbox of interpretable machine learning and a validation-on-demand platform for model validation. Aijun holds PhD degree in Statistics from University of Michigan at Ann Arbor, and he has over 10 years of experience working in financial risk management. Prior to joining Wells Fargo, Aijun was a tenure-track assistant professor at Department of Statistics and Actuarial Science, University of Hong Kong. He has published nearly 40 papers in professional conferences and journals, with topics in interpretable machine learning, data science and statistics.

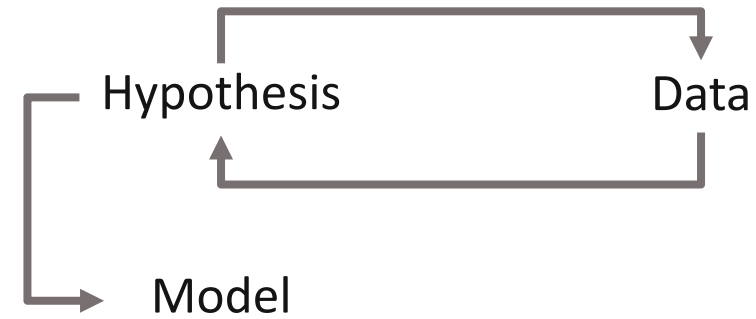
Outline

- **Introduction**
 - Interpretable machine learning
 - PiML toolbox
- **Machine Learning Interpretability**
 - Post-hoc explainability pitfalls
 - Inherently interpretable models
- **Model Diagnostics**
 - Outcome testing
 - Model comparison
- **PiML User Guide and Examples**

Interpretable Machine Learning



Breiman (2001). Statistical modeling: The two cultures. *Statistical Science*.
Gunning (2017). Explainable Artificial Intelligence (XAI). *US DARPA Report*.

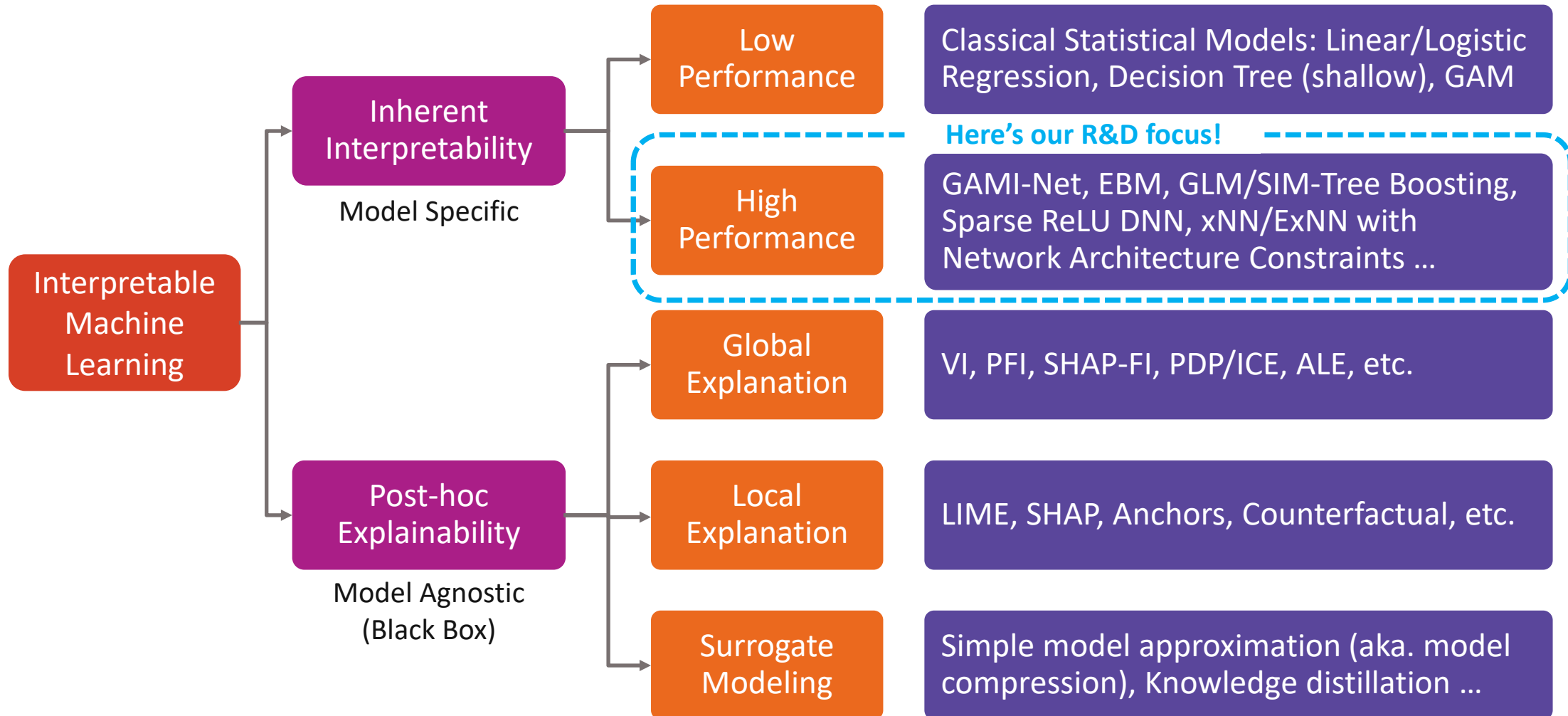


Last 20 years: modeling culture shift
from data hypothesis to algorithmic
prediction.

Models are increasingly black box.

Data → Model

Interpretable Machine Learning: A Taxonomy



PiML Toolbox



An integrated Python toolbox for interpretable machine learning

```
pip install PiML
```

Link: [Hands-on PiML tutorial through Google Colab](#)

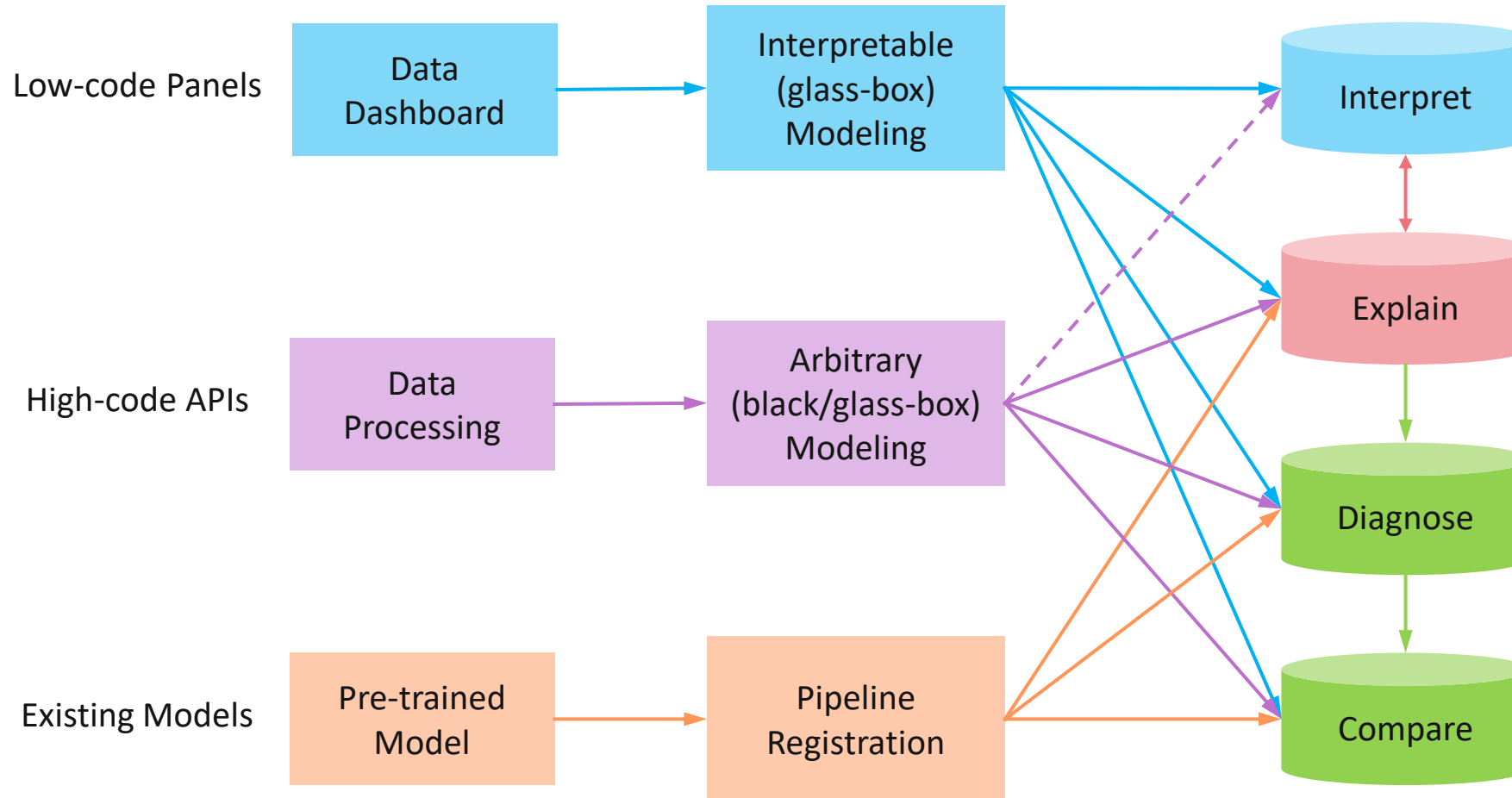
Model Development

- Inherently interpretable models
 - GLM, GAM, Tree/XGB (shallow)
 - Explainable Boosting Machine
 - GAMI Neural Networks
 - Sparse ReLU Neural Networks
 - More advanced developments
- Model-specific Interpretability
- Model-agnostic Explainability

Model Validation

- Model Diagnostics and Outcome Testing
 - Accuracy
 - WeakSpot
 - Uncertainty
 - Robustness
 - Resilience
 - Fairness
- Model Comparison and Benchmarking

PiML Toolbox: Workflow Design



PiML Toolbox: Github Repo


SelfExplainML / PiML-Toolbox Public

PiML (Python Interpretable Machine Learning) toolbox for model development and validation

Apache-2.0 license


332 stars 27 forks

★ Starred Watch



An integrated Python toolbox for interpretable machine learning

```
pip install PiML
```

 V0.2.0 is released with high-code APIs.

PiML (or π -ML, /'paɪ·'em·'el/) is a new Python toolbox for interpretable machine learning model development and validation. Through low-code interface and high-code APIs, PiML supports various machine learning models

- URL: <https://github.com/SelfExplainML/PiML-Toolbox>
- Installation: **pip install PiML**
- First Release: V0.1.0 (May 4, 2022)
- Latest release: V0.5.0 (May 4, 2023), now 600+ stars
- Low-code and high-code examples, freely through Google Colab
- It comes with [a comprehensive PiML user guide with examples](#) including
 - **Data Pipeline:** Data Summary, EDA, Data Quality, etc.
 - **Interpretable Models:** GAM, GAMI-Net, XGB2, etc.
 - **Post-hoc Explainability:** PFI, PDP, LIME, SHAP, etc.
 - **Outcome Testing:** weakness, robustness, fairness, etc.

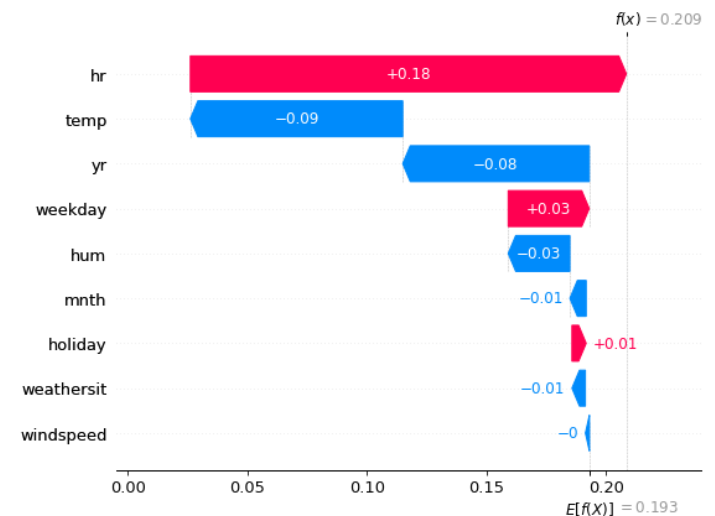
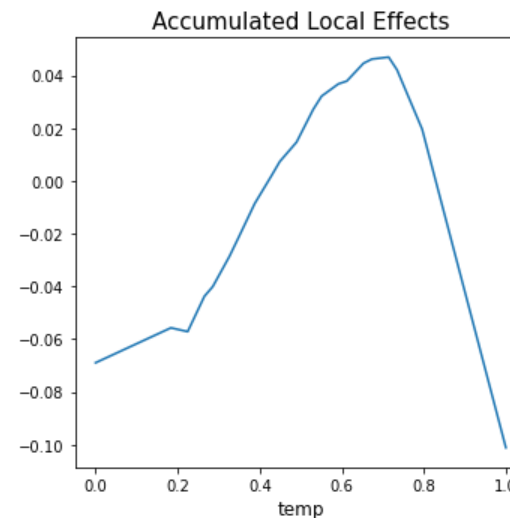
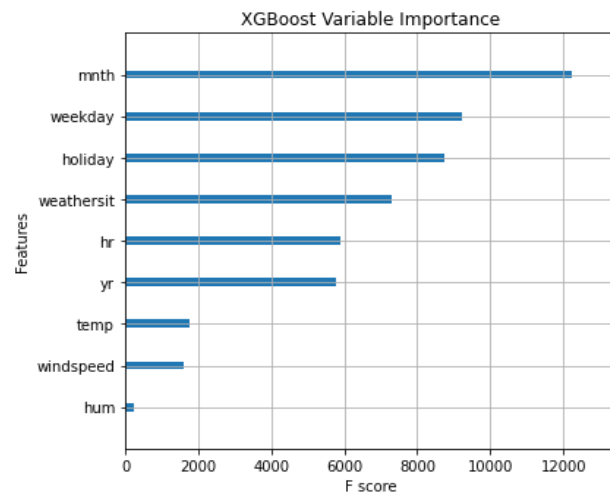
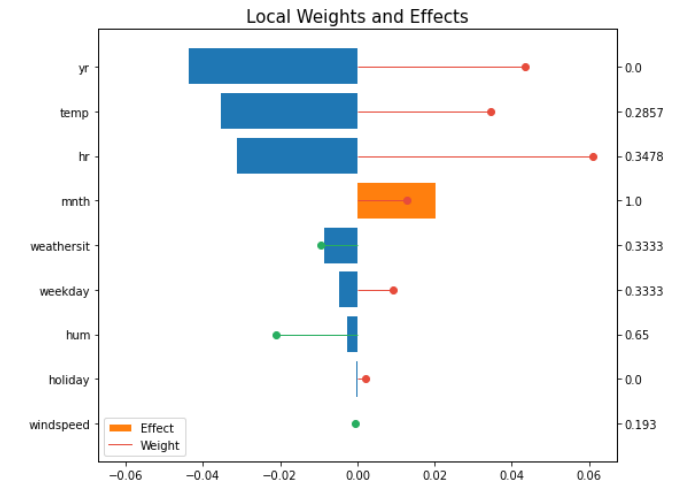
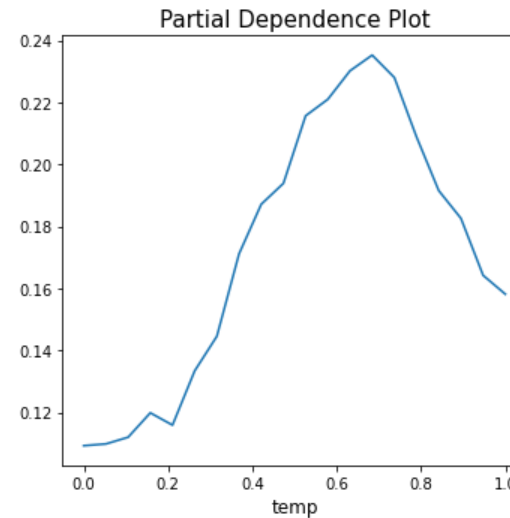
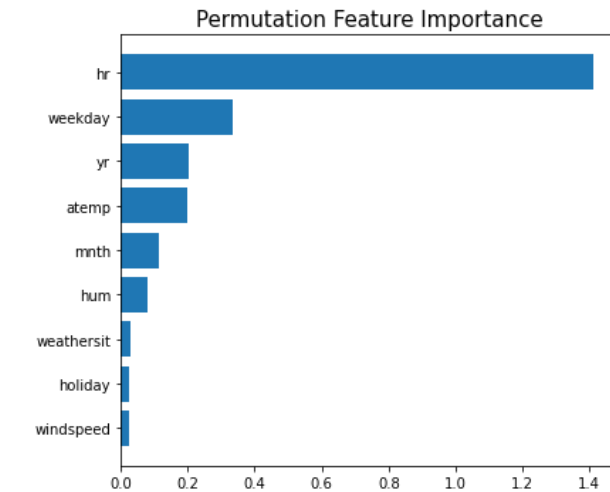
Outline

- **Introduction**
 - Interpretable machine learning
 - PiML toolbox
- **Machine Learning Interpretability**
 - Post-hoc explainability pitfalls
 - Inherently interpretable models
- **Model Diagnostics**
 - Outcome testing
 - Model comparison
- **PiML User Guide and Examples**

Post-hoc Explainability Tools

- Model-agnostic approach, applied after model development
 - Useful for explaining black-box models; but need to use with caution.
 - Most of post-hoc explainability tools (below) have potential limitation, pitfalls and puzzles
- **Local explainability tools** for explaining an individual prediction
 - **LIME** (Local Interpretable Model-agnostic Explanations)
 - **SHAP** (SHapley Additive exPlanations)
- **Global explainability tools** for explaining the overall impact of features on model predictions
 - **Examine relative importance of variables:** **VI** (Variable Importance), **PFI** (Permutation Feature Importance), **SHAP-FI** (SHAP Feature Importance), etc.
 - **Understand input-output relationships:** **PDP** (Partial Dependence Plot)/**ICE** (Individual Conditional Expectation), **ALE** (Accumulated Local Effects), etc.

Post-hoc Explainability Pitfalls



PiML Demo: BikeSharing data fit by XGBRegressor (max_depth=7, n_estimators=500)

Post-hoc Explainability vs. Inherent Interpretability

- **Post-hoc explainability** is model agnostic, but there is no free lunch. According to Cynthia Rudin, use of auxiliary post-hoc explainers creates “double trouble” for black-box models.
- Various post-hoc explanation methods, including VI/FI, PDP, ALE, ... (for global explainability) and LIME, SHAP, ... (for local explainability), **often produce results with disagreements**.
- Lots of discussions about pitfalls, challenges and potential risks of using post-hoc explainers.
- This echoes Footnote 1 of CFPB Circular 2022-03.

- **Inherent interpretability** is intrinsic to a model itself. It facilitates gist and intuitiveness for human insightful interpretation. It is important for evaluating a model’s **conceptual soundness**.
- Model interpretability is a loosely defined concept, without a common quantitative measure.
- Sudjianto and Zhang (2021) proposed qualitative rating assessment for designing inherently interpretable ML models based on model design characteristics.
- **PiML Toolbox** integrates inherently interpretable models, including GAM, EBM, GAMI-Net and XGB2.

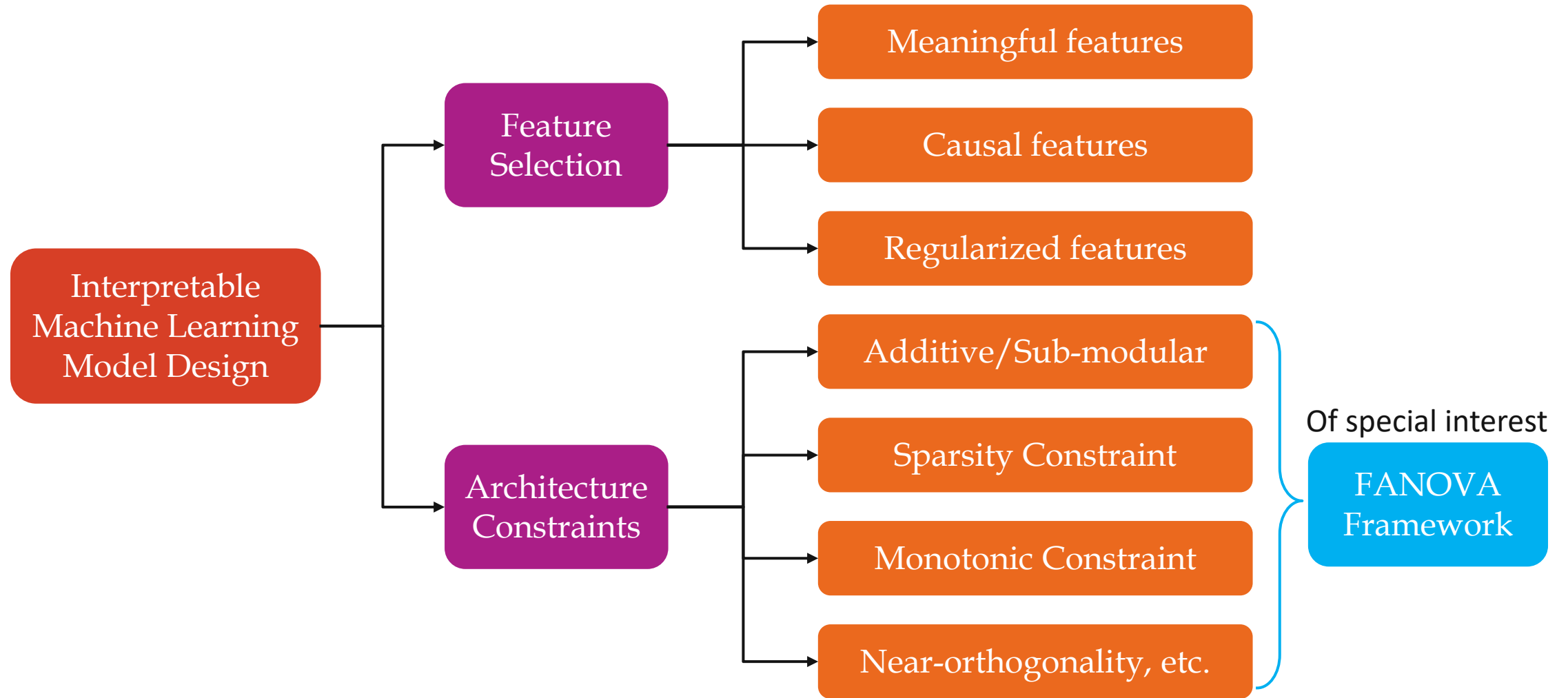
Designing Inherently Interpretable Models

Model Characteristics	Gist for Interpretation
Additivity	Additive decomposition of feature effects tends to be more interpretable
Sparsity	Having fewer features or components tends to be more interpretable
Linearity	Linear or constant feature effects are easy to interpret
Smoothness	Continuous and smooth feature effects are relatively easy to interpret
Monotonicity	Sometimes increasing/decreasing effects are desired by expert knowledge
Visualizability	Direct visualization of feature effects facilitates diagnostics and interpretation
Projection	Sparse and near-orthogonal projection tends to be more interpretable
Segmentation	Having smaller number of segments (heterogeneous data) is more interpretable

¹ Sudjianto and Zhang (2021): Designing Inherently Interpretable Machine Learning Models. [arXiv: 2111.01743](https://arxiv.org/abs/2111.01743)

² Yang, Zhang and Sudjianto (2021, IEEE TNNLS): Enhancing Explainability of Neural Networks through Architecture Constraints. [arXiv: 1901.03838](https://arxiv.org/abs/1901.03838)

Designing Inherently Interpretable Models



¹ Sudjianto and Zhang (2021): Designing Inherently Interpretable Machine Learning Models. [arXiv: 2111.01743](https://arxiv.org/abs/2111.01743)

² Yang, Zhang and Sudjianto (2021, IEEE TNNLS): Enhancing Explainability of Neural Networks through Architecture Constraints. [arXiv: 1901.03838](https://arxiv.org/abs/1901.03838)

FANOVA Model Design Framework

- One effective way is to design inherently interpretable models by the Functional ANOVA representation

$$g(\mathbb{E}(y|\mathbf{x})) = g_0 + \sum_j g_j(x_j) + \sum_{j < k} g_{jk}(x_j, x_k) + \sum_{j < k < l} g_{jkl}(x_j, x_k, x_l) + \dots$$

It additively decomposes a predictive model into the overall mean (i.e., intercept) g_0 , main effects $g_j(x_j)$, two-factor interactions $g_{jk}(x_j, x_k)$, and higher-order interactions ...

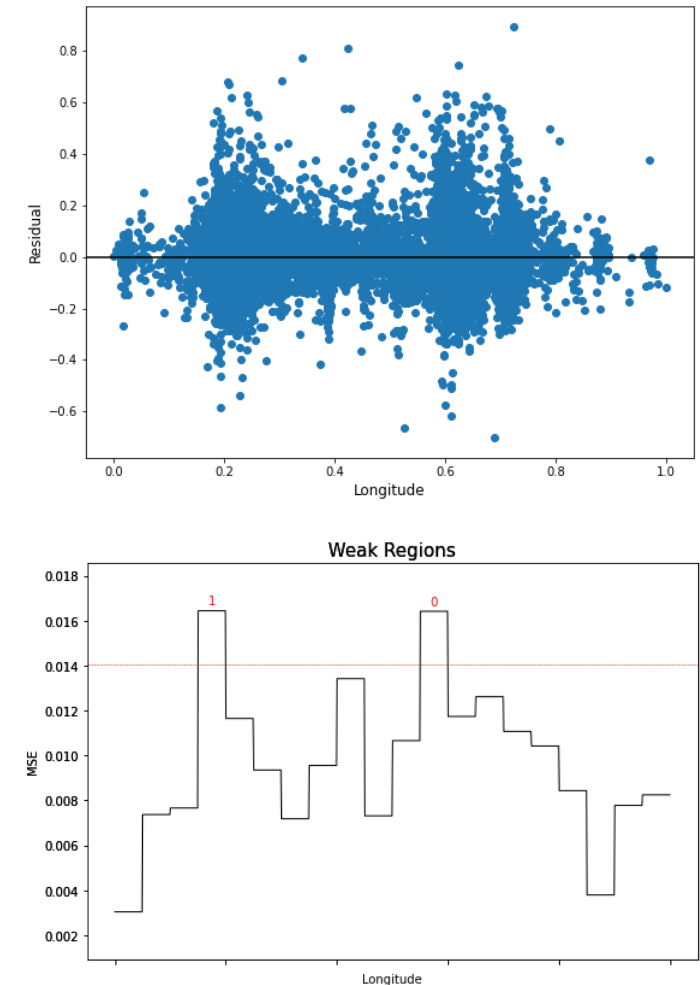
- GAM main-effect models: BinningLogistic, XGBoostDepth1, GAM using Splines, ...
- GAMI main-effect plus two-factor-interaction models:
 - **EBM** (Nori, et al. 2019) → explainable boosting machine with shallow trees
 - **XGB2** (Lengerich, et al. 2020) → boosted trees of depth 2 with effect purification
 - **GAMI-Net** (Yang, Zhang and Sudjianto, 2021) → specialized neural nets
 - **GAMI-Lin-T** (Hu, et al. 2023) → specialized boosted linear model-based trees
- **PiML Toolbox** integrates GAM, XGB1, XGB2, EBM and GAMI-Net, and provides inherent interpretability.

Outline

- **Introduction**
 - Interpretable machine learning
 - PiML toolbox
- **Machine Learning Interpretability**
 - Post-hoc explainability pitfalls
 - Inherently interpretable models
- **Model Diagnostics**
 - Outcome testing
 - Model comparison
- **PiML User Guide and Examples**

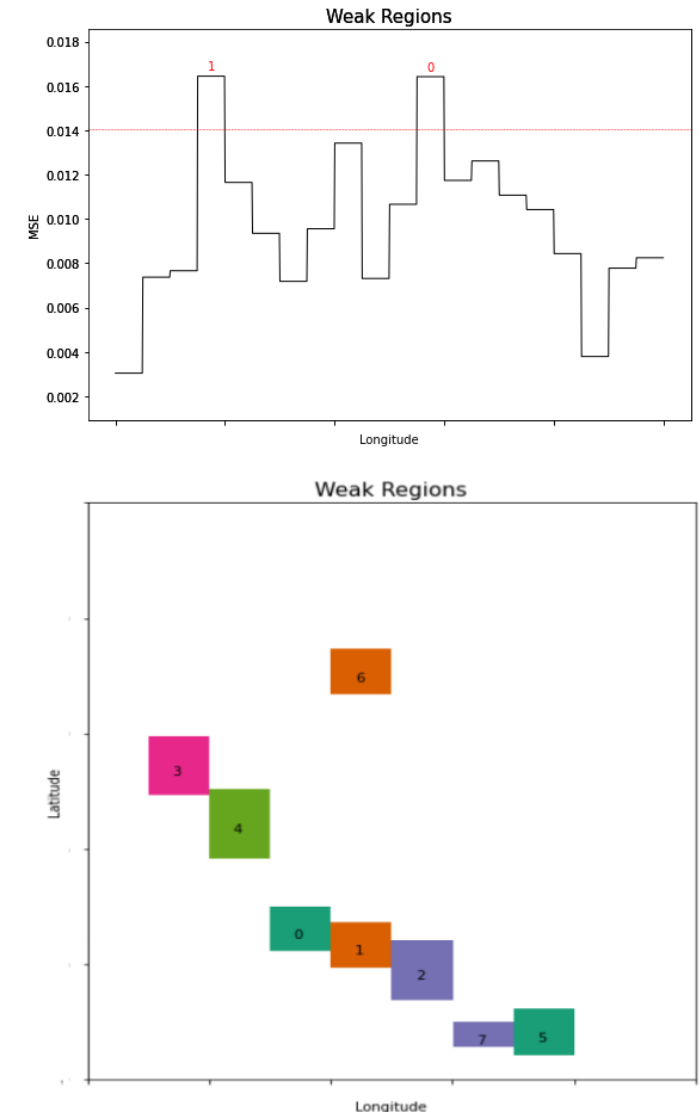
Accuracy, Residuals and WeakSpot

- ML model performance is often measured by **accuracy**, as examined via standard ML metrics (e.g. MSE, MAE, R2, ACC, AUC, F1-score, Precision and Recall).
- However, model assessment by single-valued metrics is insufficient. More granular diagnostics and alternative metrics are needed.
- To check **model underfitting**, perform error analysis based on residuals
 - **Residual plot** marginally for each feature of interest;
 - **WeakSpot** to identify weak regions with high residuals on either training or testing data.
- **PiML toolbox** employs several slicing techniques for WeakSpot.

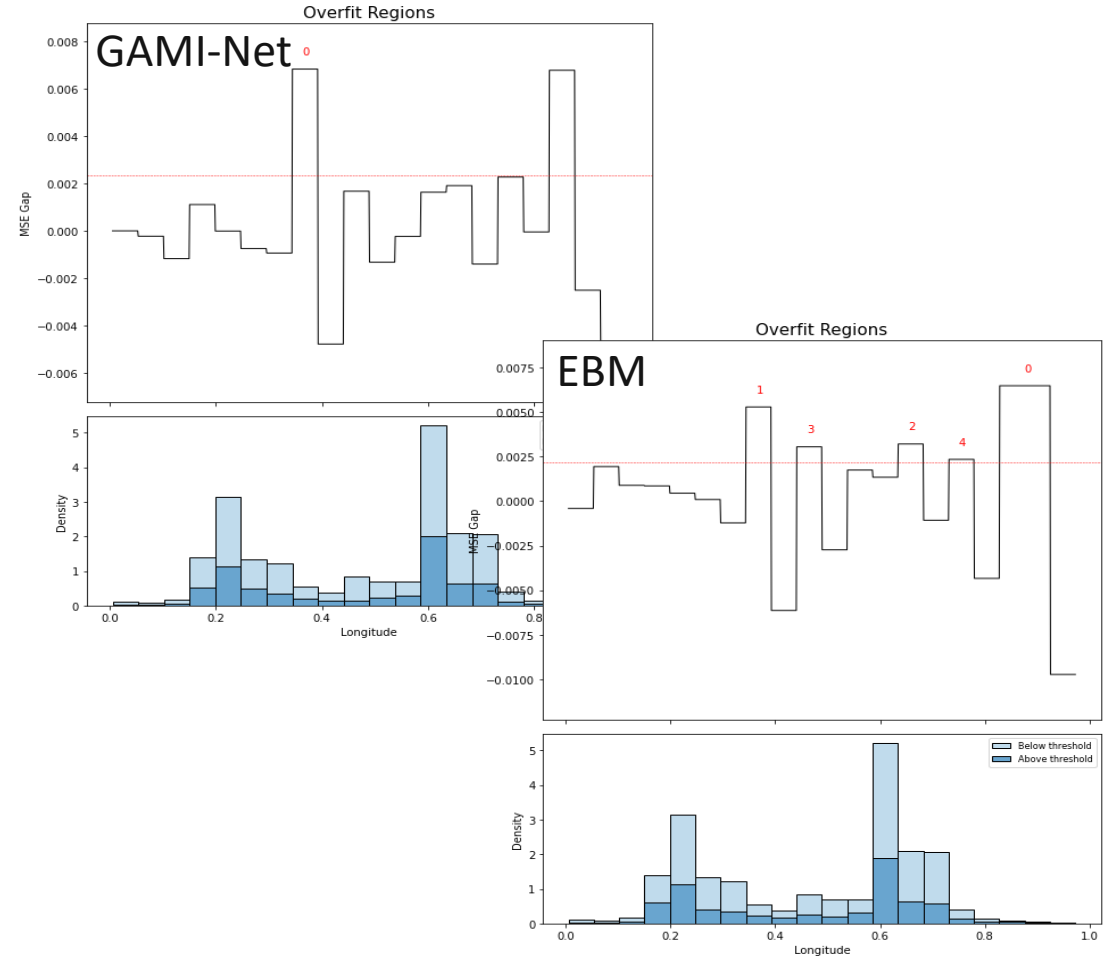
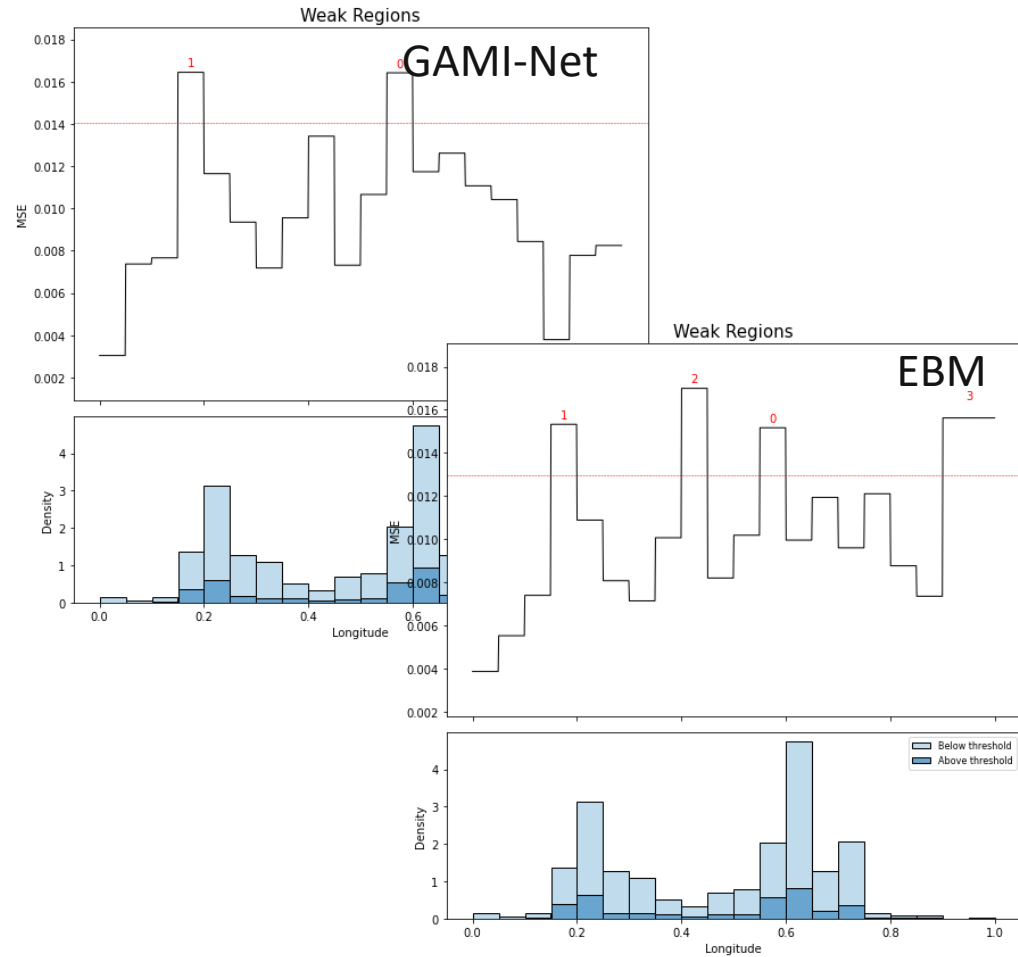


Error Analysis by Slicing Techniques

1. **Specify an appropriate metric** based on individual prediction residuals: e.g., MSE for regression, ACC for classification, train-test performance gap (for checking overfit), uncertainty bandwidth, ...
2. Specify 1 or 2 slicing features of interest;
3. Evaluate the metric for each sample in the target data (training or testing) as pseudo responses;
4. Segment the target data along the slicing features, by
 - a) [Unsupervised] Histogram slicing with equal-space binning, or
 - b) [Supervised] fitting a decision tree or tree-ensemble to generate the sub-regions;
5. **Identify the sub-regions** with average metric exceeding the pre-specified threshold, subject to minimum sample condition.



PiML Demo: WeakSpot and Overfit



PiML Demo: WeakSpot and Overfit analysis for CaliforniaHousing data fit by GAMI-Net and EBM

Uncertainty Quantification

- Prediction uncertainty is important to understand where the model produces less reliable prediction:

Wider prediction interval \rightarrow Less reliable prediction

- Quantification of prediction uncertainty can be done through **Split Conformal Prediction** under the exchangeability assumption:

Given a pre-trained model $\hat{f}(\mathbf{x})$, a hold-out calibration data $\mathcal{X}_{\text{calib}}$, a pre-defined conformal score $S(\mathbf{x}, y, \hat{f})$ and the error rate α (say 0.1)

- Calculate the score $S_i = S(\mathbf{x}_i, y_i, \hat{f})$ for each sample in $\mathcal{X}_{\text{calib}}$;
- Compute the calibrated score quantile

$$\hat{q} = \text{Quantile}\left(\{S_1, \dots, S_n\}; \frac{[(n+1)(1-\alpha)]}{n}\right);$$

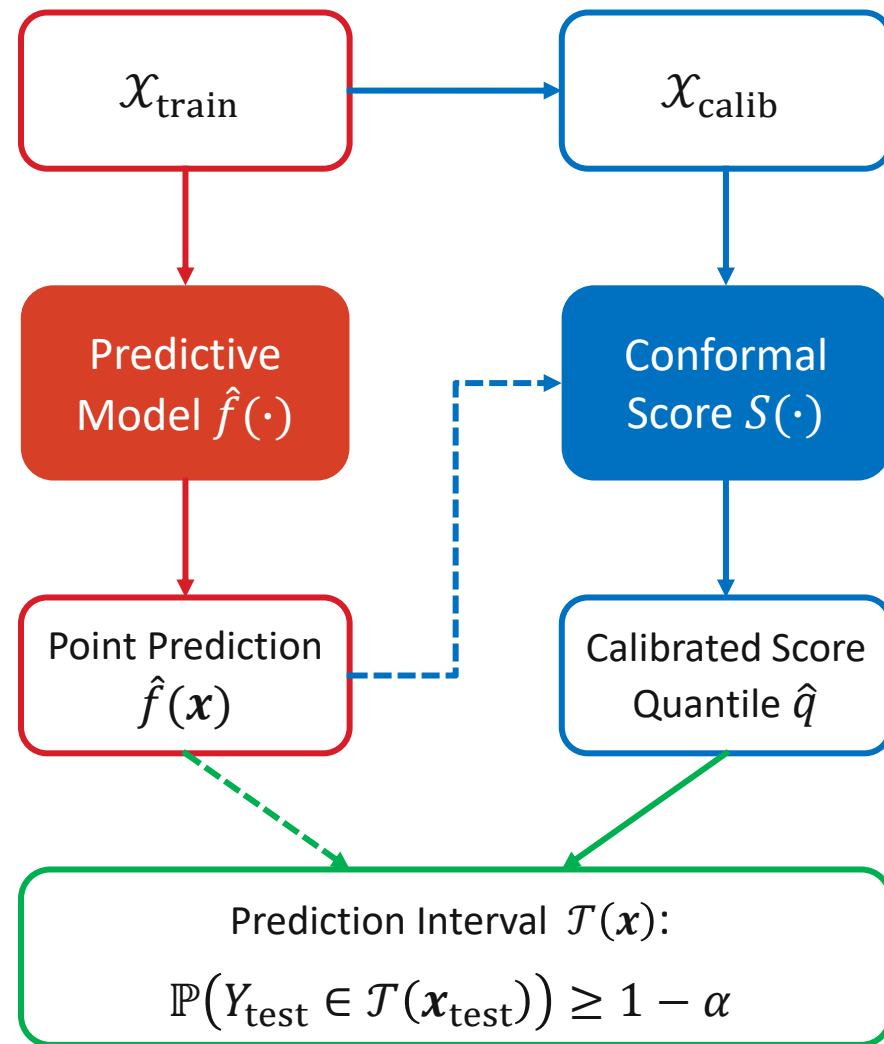
- Construct the prediction set for the test sample \mathbf{x}_{test} by

$$\mathcal{T}(\mathbf{x}_{\text{test}}) = \{y: S(\mathbf{x}_{\text{test}}, y, \hat{f}(\mathbf{x}_{\text{test}})) \leq \hat{q}\}.$$

Under the exchangeability condition of conformal scores, we have that

$$1 - \alpha \leq \mathbb{P}(Y_{\text{test}} \in \mathcal{T}(\mathbf{x}_{\text{test}})) \leq 1 - \alpha + \frac{1}{n+1}.$$

This provides the prediction bounds with α -level acceptable error.



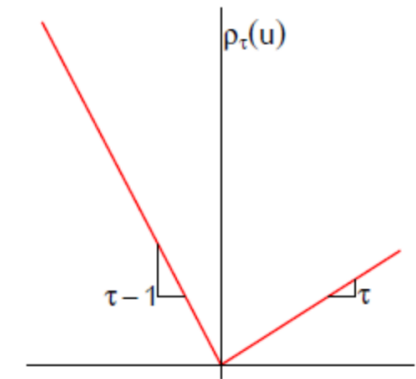
Conformalized Residual Quantile Regression

Directly evaluate prediction uncertainty of a pre-trained regression model $\hat{f}(\mathbf{x})$:

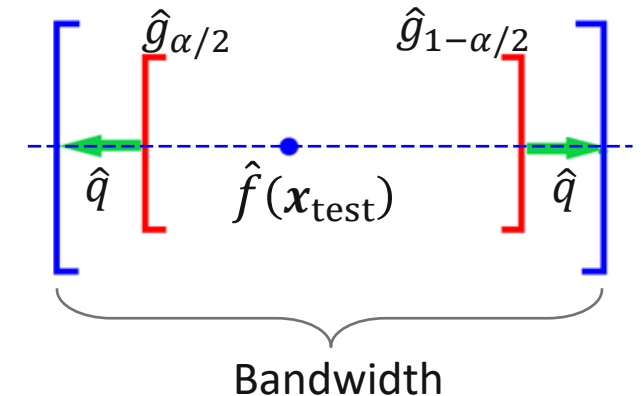
1. Obtain residuals $y_i - \hat{f}(\mathbf{x}_i)$ for each $i \in \mathcal{X}_{\text{train}}$ or $\mathcal{X}_{\text{split}}$, fit a quantile regressor (e.g. LightGBM with quantile loss) for residuals $[\hat{g}_{\alpha/2}(\mathbf{x}), \hat{g}_{1-\alpha/2}(\mathbf{x})]$;
2. Define score $S(\mathbf{x}, y, \hat{f}) = \max\{\hat{g}_{\alpha/2}(\mathbf{x}) - y + \hat{f}(\mathbf{x}), y - \hat{f}(\mathbf{x}) - \hat{g}_{1-\alpha/2}(\mathbf{x})\}$
3. Calculate $\hat{q} = \text{Quantile}\left(\{S_1, \dots, S_n\}; \frac{[(n+1)(1-\alpha)]}{n}\right)$, using $S(\mathbf{x}, y, \hat{f})$ on $\mathcal{X}_{\text{calib}}$
4. Construct the prediction interval for the test sample \mathbf{x}_{test} by

$$\mathcal{T}(\mathbf{x}_{\text{test}}) = [\hat{f}(\mathbf{x}_{\text{test}}) + \hat{g}_{\alpha/2}(\mathbf{x}_{\text{test}}) - \hat{q}, \hat{f}(\mathbf{x}_{\text{test}}) + \hat{g}_{1-\alpha/2}(\mathbf{x}_{\text{test}}) + \hat{q}].$$

Interpretation: the final prediction interval is composed of three terms: original prediction, estimated residual quantiles, and calibrated adjustment.

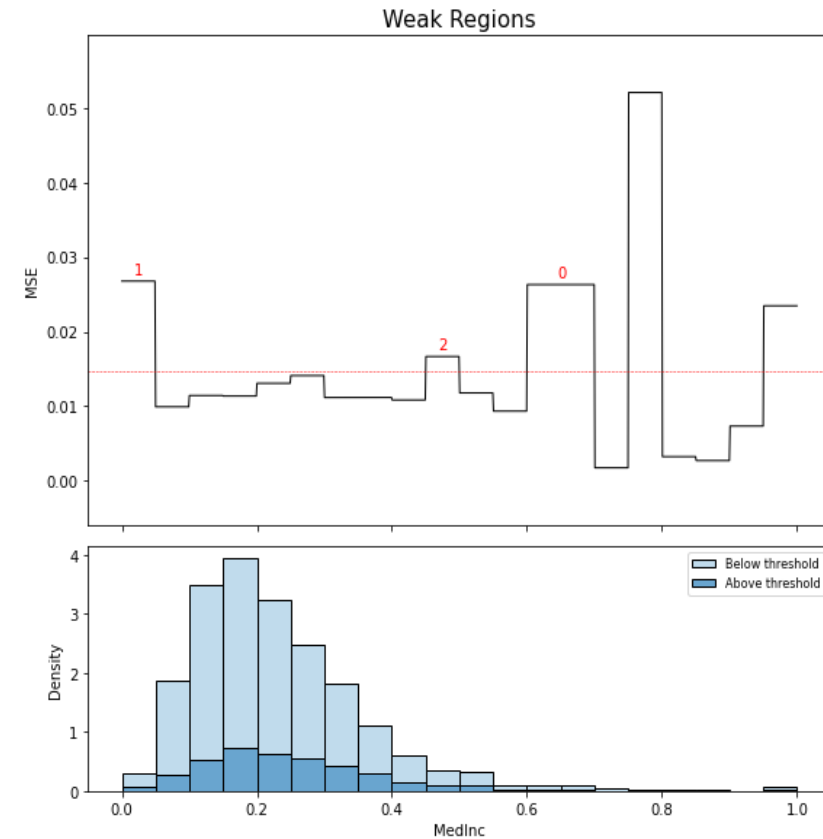
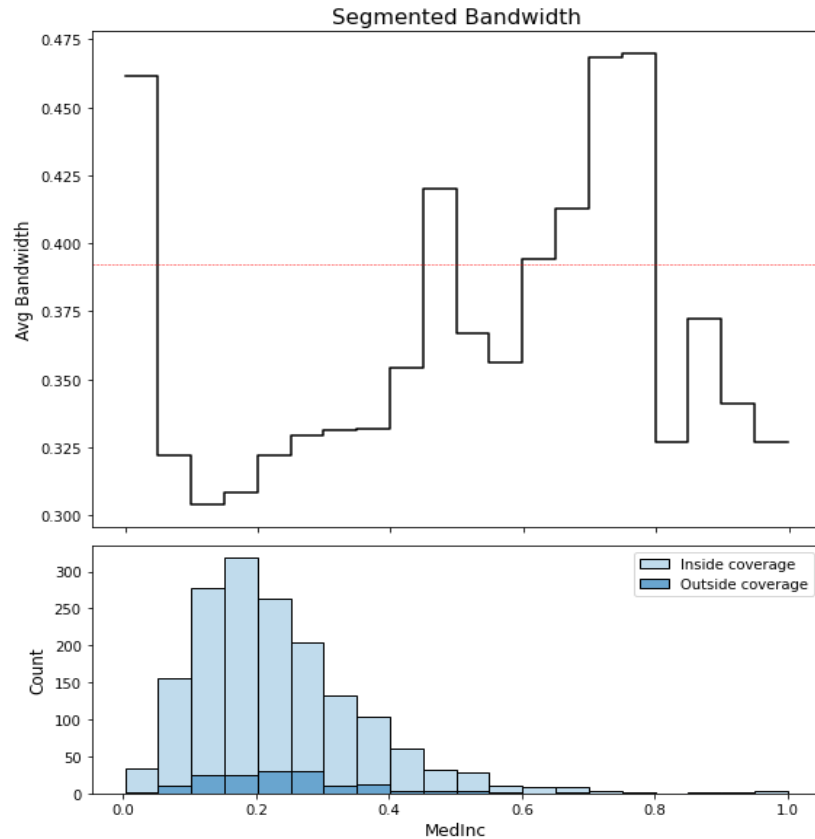


Quantile loss



PiML Demo: Uncertainty Quantification

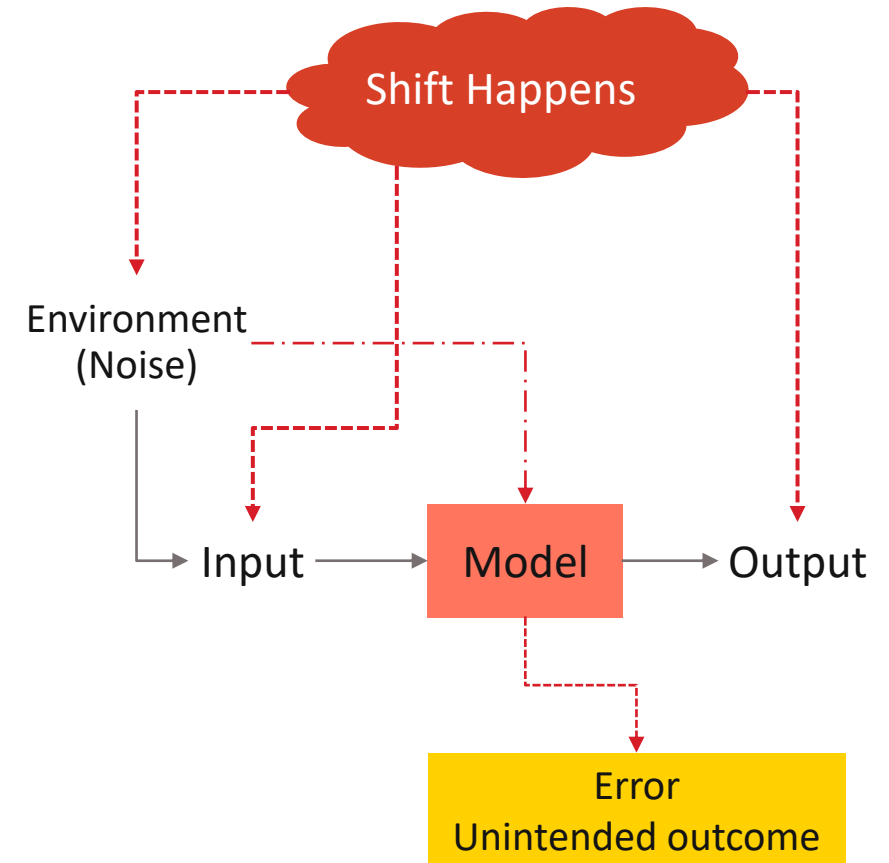
Note that quantile regression makes the interval bandwidth adaptive to heteroscedastic residuals.



PiML Demo: Prediction Uncertainty Testing for CaliforniaHousing data fit by GAMI-Net.

Robustness and Resilience

- Train-test data split for model development often gives over-optimism of model performance, since model in production will be exposed to data distribution shift.
- **Robustness test:** evaluate the performance degradation under covariate noise perturbation:
 - Perturb testing data covariates with small random noise;
 - Assess model performance of perturbed testing data.
 - Overfitting models often perform poorly in changing environments.
- **Resilience test:** evaluate performance degradation under distribution shift scenarios (worst-sample, outer-sample, worst-cluster, hard-sample):
 - Investigate performance degradation in extreme cases;
 - Rank covariates using distribution shift measure such as Population Stability Index (PSI);
 - Identify sensitivity and vulnerability due to covariate shift.



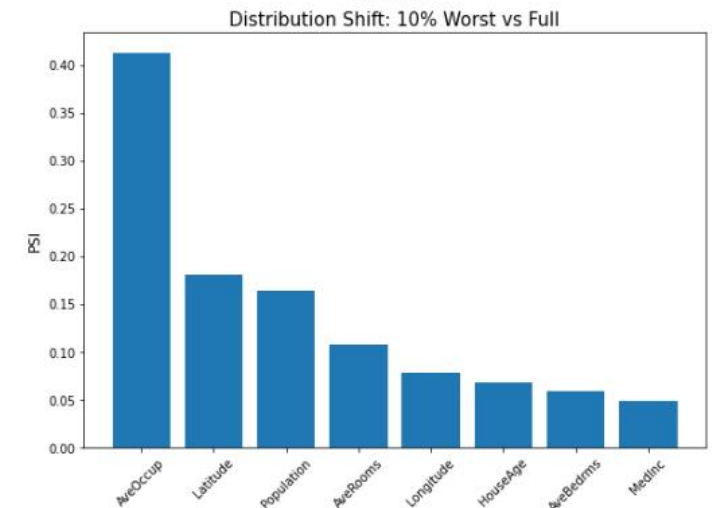
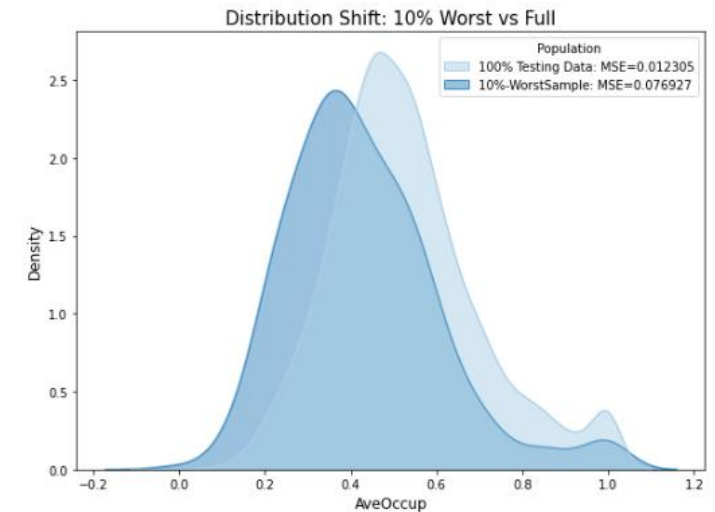
Measuring Distribution Shift

- **Population Stability Index:**

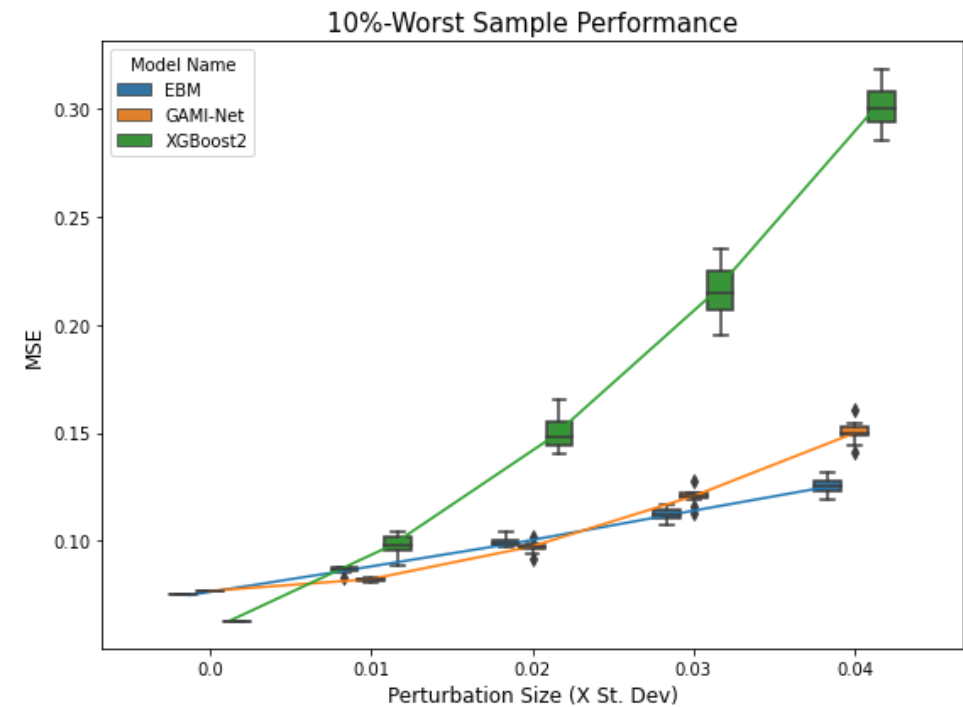
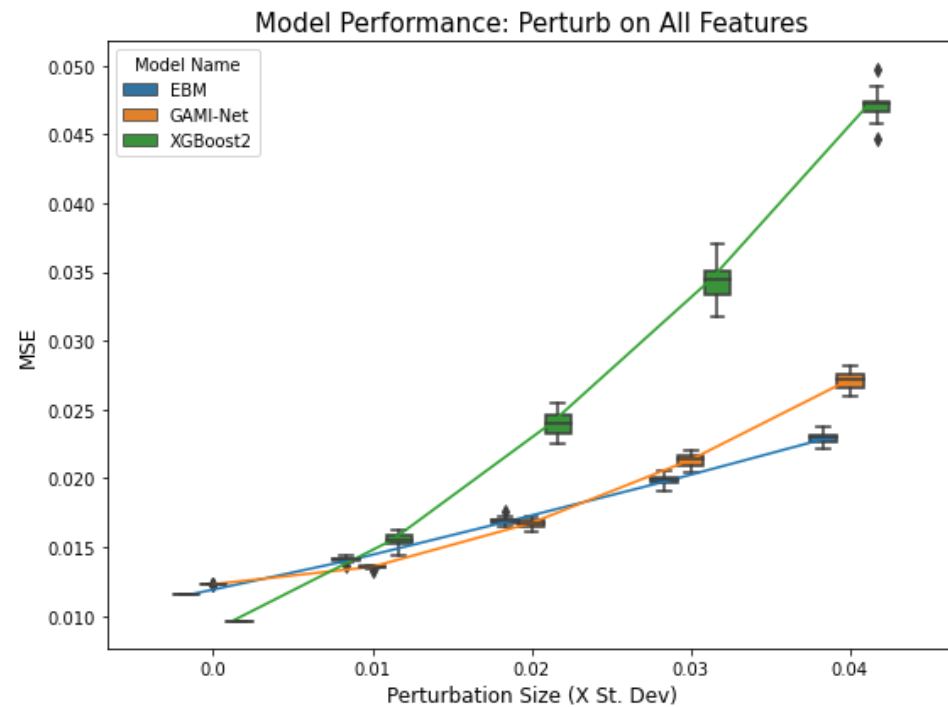
$$PSI = \sum_{i=1}^B (\text{Target}_i\% - \text{Base}_i\%) \ln \left(\frac{\text{Target}_i\%}{\text{Base}_i\%} \right)$$

based on the proportions of samples in each bucket of the target vs. base population. Rule of thumb:

- PSI < 0.1: no significant distribution change
 - PSI < 0.2: moderate distribution change
 - PSI >= 0.2: significant distribution change
- Other two-sample test: KL divergence, Kolmogorov-Smirnov (KS) and Cramer-von Mises (CM) statistics based on empirical distributions.
 - In resilience testing, PSI measures the distribution shift one-feature-at-a-time. One may further use WeakSpot to perform drill-down analysis on sensitive features.

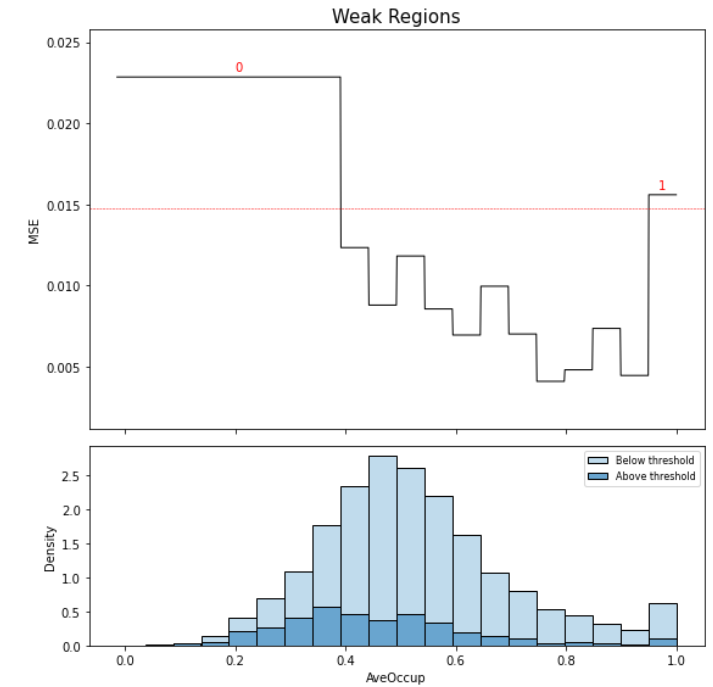
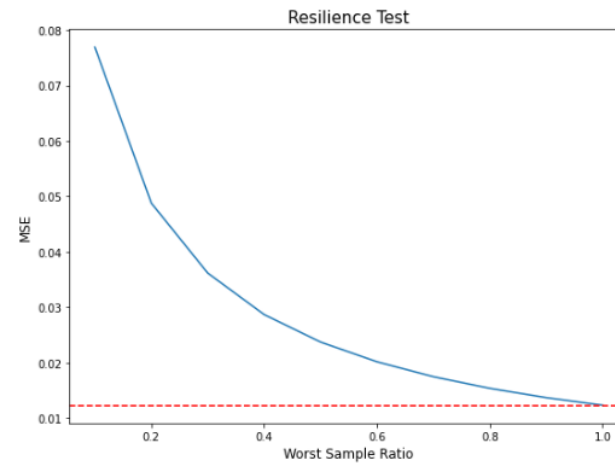
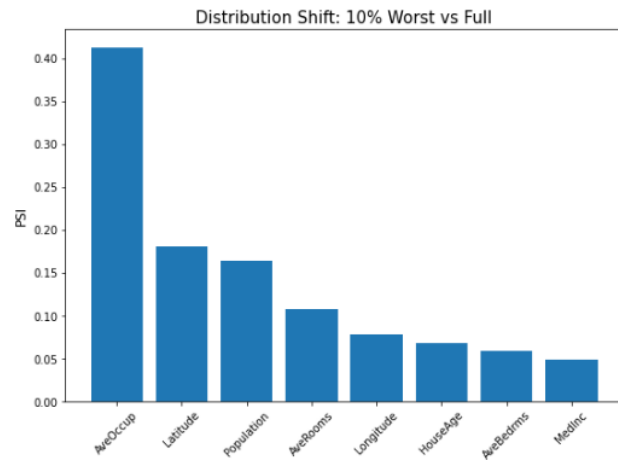
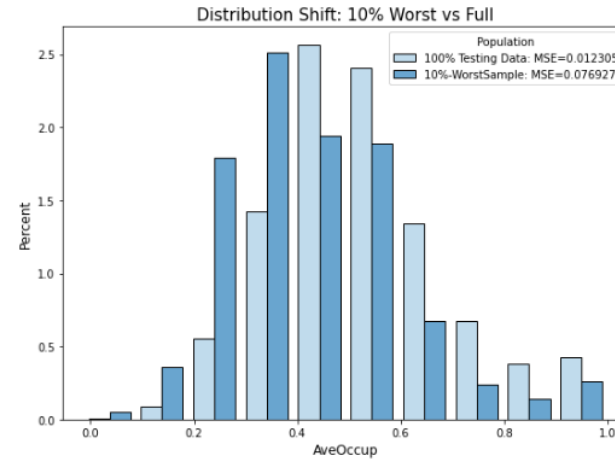
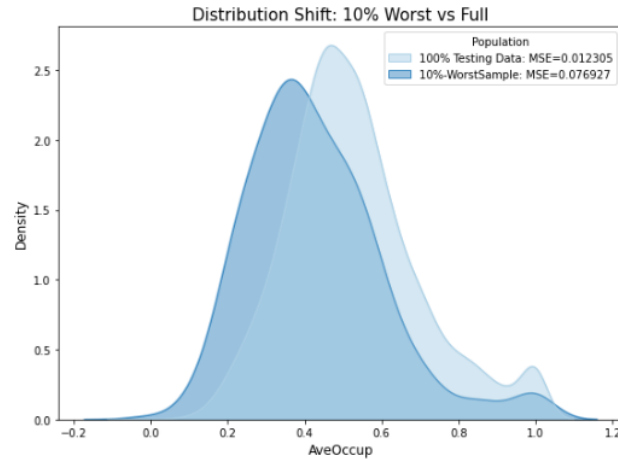


PiML Demo: Robustness Testing



PiML Demo: Robustness Testing for CaliforniaHousing data fit by GAMI-Net, EBM vs XGBoost2.

PiML Demo: Resilience Testing



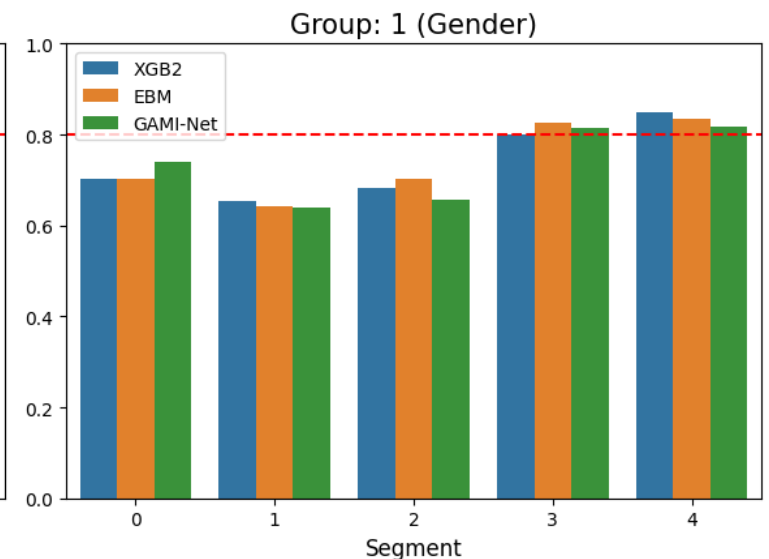
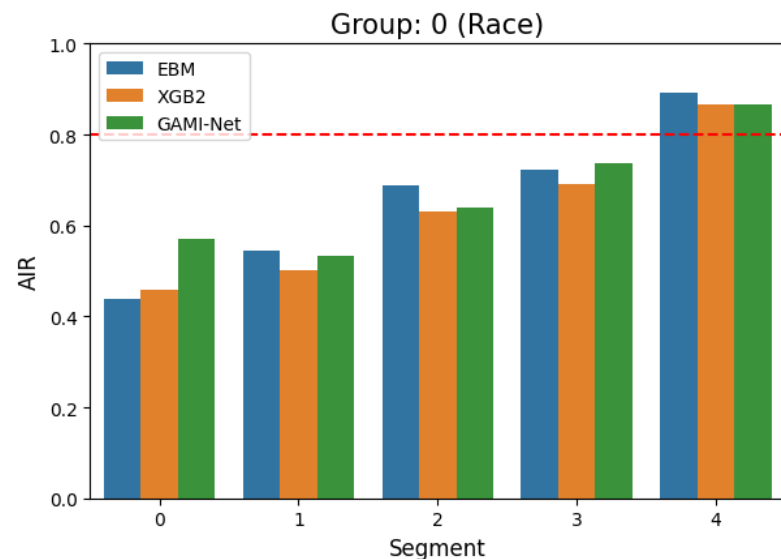
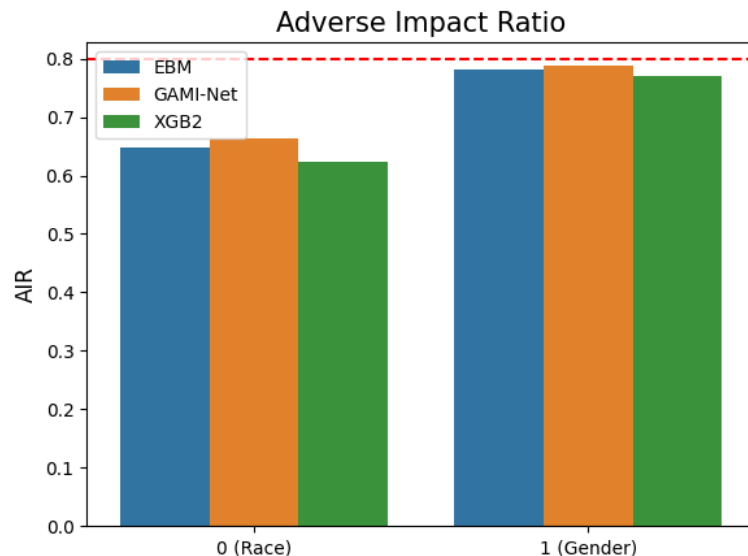
PiML Demo: Resilience Test and WeakSpot for CaliforniaHousing data fit by GAMI-Net

Bias and Fairness

- For each demographic feature (Race, Gender), consider AIR between protected group vs reference group.

$$AIR = \frac{(TP_p + FN_p)/n_r}{(TP_r + FN_r)/n_p}$$

- AIR below 0.8 is a sign of bias and unfairness.
- PiML provides segmented metrics conditional on a modeling variable (e.g., Balance below). It also provides methods to debias through feature binning and decision thresholding.



Outline

- **Introduction**
 - Interpretable machine learning
 - PiML toolbox
- **Machine Learning Interpretability**
 - Post-hoc explainability pitfalls
 - Inherently interpretable models
- **Model Diagnostics**
 - Outcome testing
 - Model comparison
- **PiML User Guide and Examples**

PiML User Guide and Examples

PiML [Install](#) [API](#) [User Guide](#) [Examples](#) [FAQ](#)

Python Interpretable Machine Learning

`pip install PiML`

[User Guide](#) [GitHub](#)

- A Python toolbox for interpretable machine learning
- Supports a growing list of inherently interpretable models
- Supports a whole spectrum of model testing and validation
- Provides easy to use low-code interface and high-code APIs

Data Pipeline

Load, summarize, and prepare data

- PiML Data Pipeline
- Exploratory Data Visualization
- Feature Selection
- Custom Data Loading into PiML

Interpretable Models

Inherently interpretable machine learning

- Classic Statistics Models
- GAMI Neural Networks
- XGBoosted Trees of Depth 2

Post-hoc Explainability

Global and local explainability

- Global Methods: PFI, PDP, ALE
- Local Methods: LIME, SHAP
- Post-hoc Explainability Disagreement

Outcome Testing

Model diagnostics

- WeakSpot by Slicing Techniques
- Reliability Test by Conformal Prediction
- Robustness Test by X-Perturbation
- Resilience Test under OOD Scenarios

Model Comparison

Benchmarking

- Black-box vs. Glass-box Models
- Is XGBoost Benign Overfitting?
- Multi-objective Model Selection

Low-Code Case Studies

PiML workflow and experimentation

- Example: Bikesharing Data
- Example: CaliforniaHousing Data
- Example: TaiwanCredit Data
- Fairness Simulation Study 1
- Fairness Simulation Study 2

<https://selfexplainml.github.io/PiML-Toolbox>



Thank you

Aijun Zhang, Ph.D.

Email: Aijun.Zhang@wellsfargo.com

LinkedIn: <https://www.linkedin.com/in/ajzhang/>