WELLS FARGO

Machine Learning Model Validation

Part 1: Machine Learning Interpretability

Aijun Zhang, Ph.D.
Corporate Model Risk, Wells Fargo

QU-ML Model Validation Workshop, Session 1, June 29, 2022.

Machine Learning Model Validation

Session 1 (today)

Machine Learning Interpretability

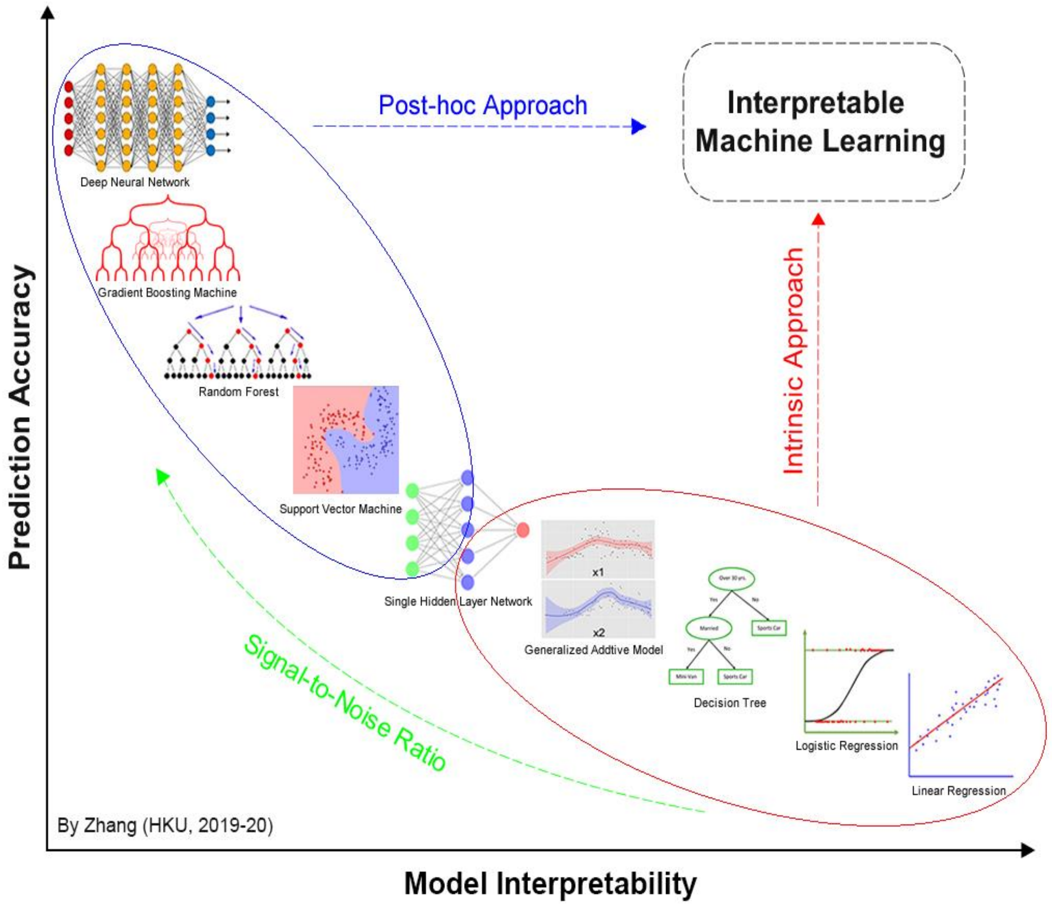
1. Interpretable ML and PiML Toolbox
2. Post-hoc Explainability Tools/Puzzles
3. Designing Interpretable ML Models
4. ReLU Deep Neural Networks
5. FANOVA Models: EBM and GAMI-Net

Session 2 (July 6)

Model Diagnostics and Validation

1. AI Model Risk and Trustworthiness
2. Accuracy, WeakSpot and Overfit
3. Reliability Testing
4. Robustness and Resilience Testing
5. Model Comparison

Interpretable Machine Learning



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

Hypothesis Data

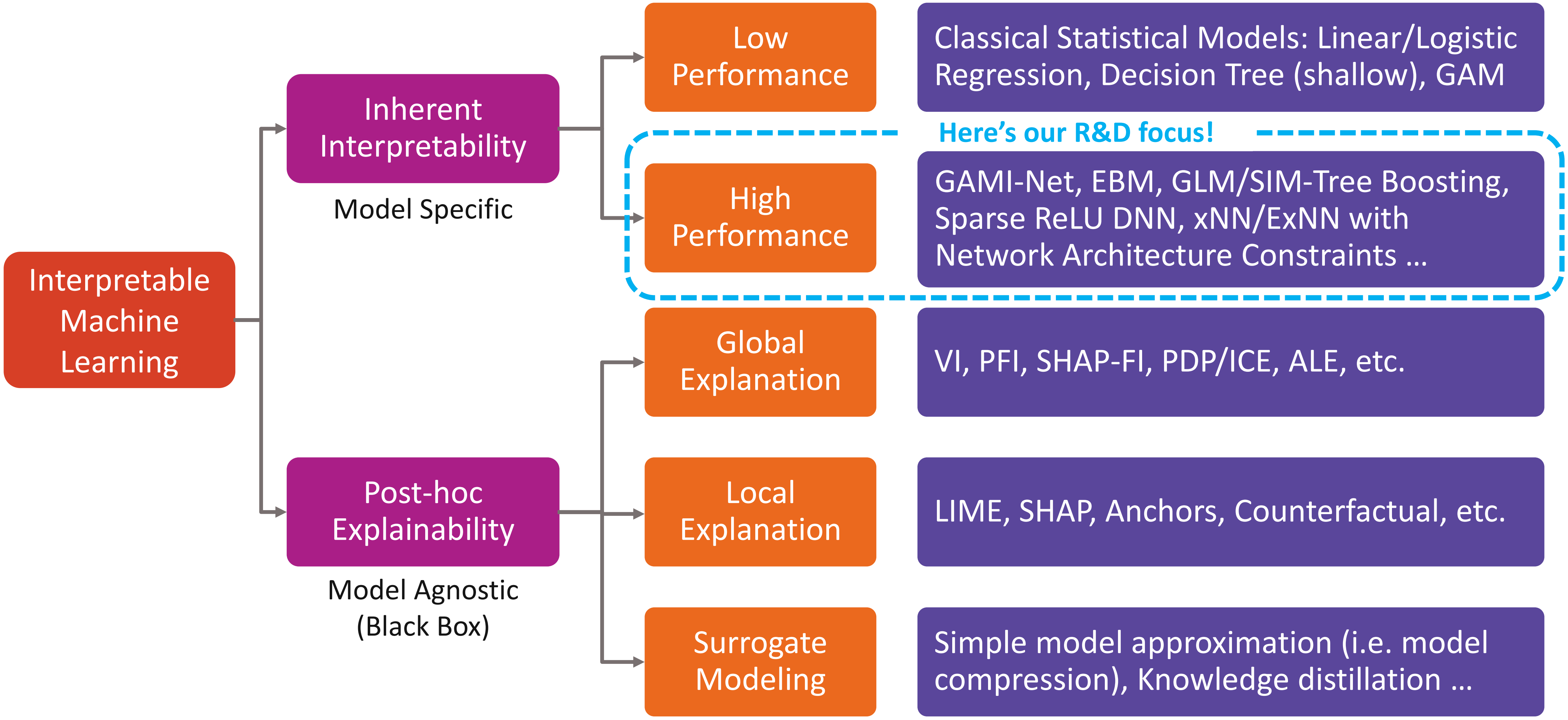
Model

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** covers most of IML methods, with focus on inherently-interpretable high-performance models.

Interpretable Machine Learning: Python Toolbox

✓ Low-code Interface

PiML

Python Interpretable Machine Learning

✓ High-code Programming

Model Development

- **Inherently interpretable ML models**
  - GLM, GAM, Tree/Rule (to add)
  - Explainable Boosting Machine
  - GAMI Neural Networks
  - Sparse ReLU Neural Networks
  - More advanced developments
- **Model-inherent Interpretability**
- **Post-hoc Explainability Tools** (use with caution)

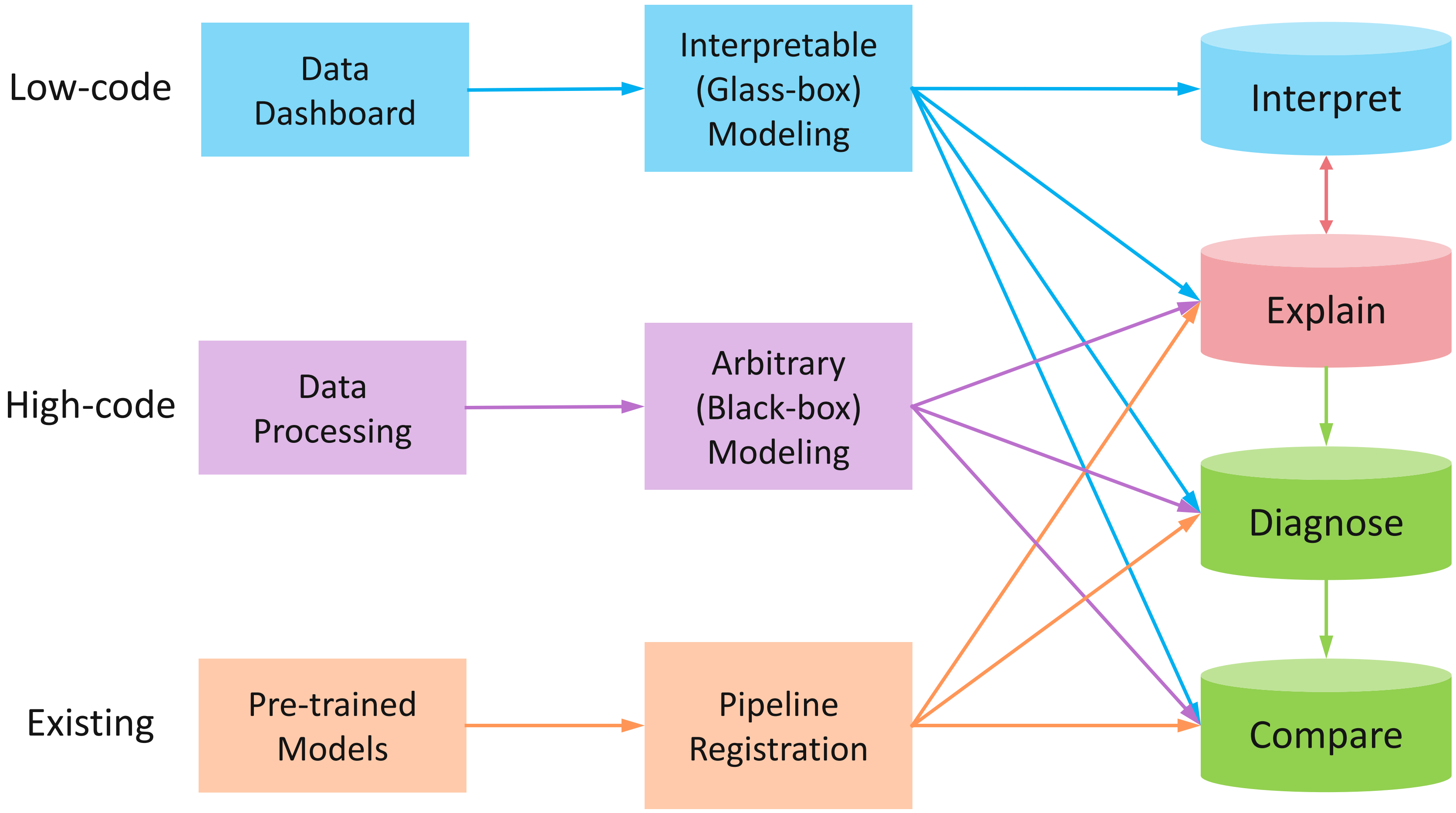
Model Validation

- ML Model Diagnostics and Outcome Testing
  - Accuracy
  - WeakSpot
  - Overfit/Underfit
  - Reliability
  - Robustness
  - Resilience

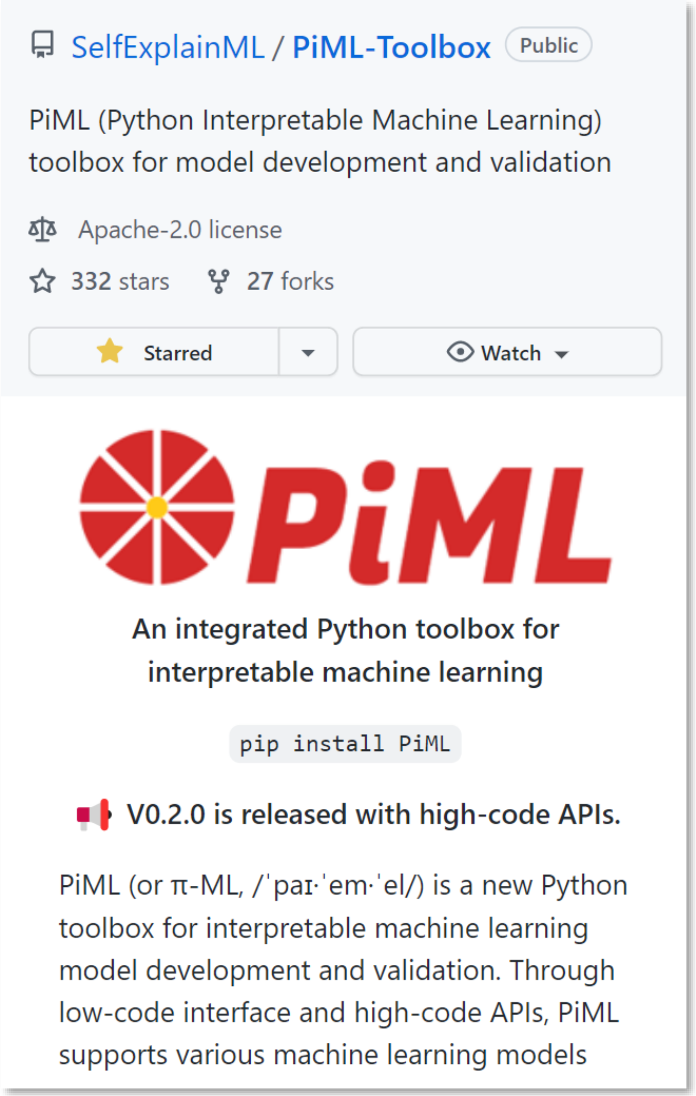
Trustworthy AI

- Model Comparison and Benchmarking

PiML Toolbox: Workflow Design



PiML Toolbox: Github Repo



- URL: https://github.com/SelfExplainML/PiML-Toolbox
- Installation: **pip install PiML**
- First Release: V0.1.0 (May 4, 2022)
- Latest release: V0.2.0 (June 26, 2022)
- Low-code and high-code examples, freely through Google Colab
- We'll provide **a series of reproducible PiML tutorials**, including
  - Post-hoc Explainability Puzzles
  - Inherently Interpretable Models
  - Sparse ReLU Deep Neural Networks
  - FANOVA-Interpretable Models: GAMI-Net and EBM
  - ML Model Diagnostics and Validation, etc.

Post-hoc Explainability Tools/Puzzles

for black-box models (Must use with caution)

Post-hoc Explainability Tools

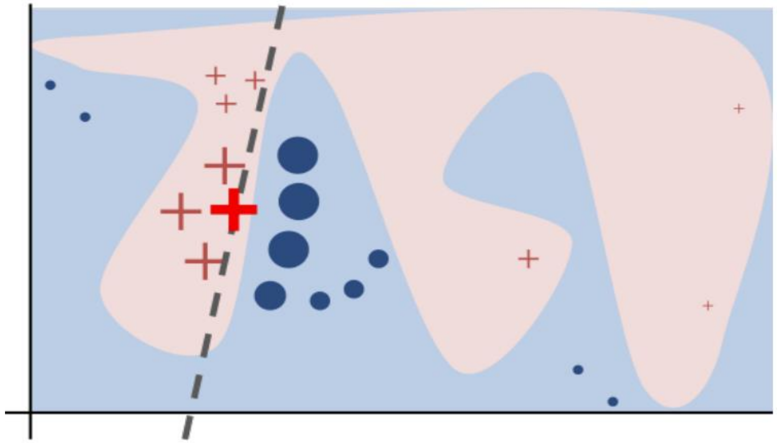
- Model-agnostic approach, applied after model development
  - Useful for explaining black-box models; but need to use with caution.
  - Most of post-hoc explainability tools (below) have potential limitation, pitfalls and puzzles
- **Local explainability tools** for explaining an individual prediction
  - 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)
  - Understand input-output relationships: **PDP** (Partial Dependence Plot)/**ICE** (Individual Conditional Expectation), **ALE** (Accumulated Local Effects), **H-statistic** for feature interactions

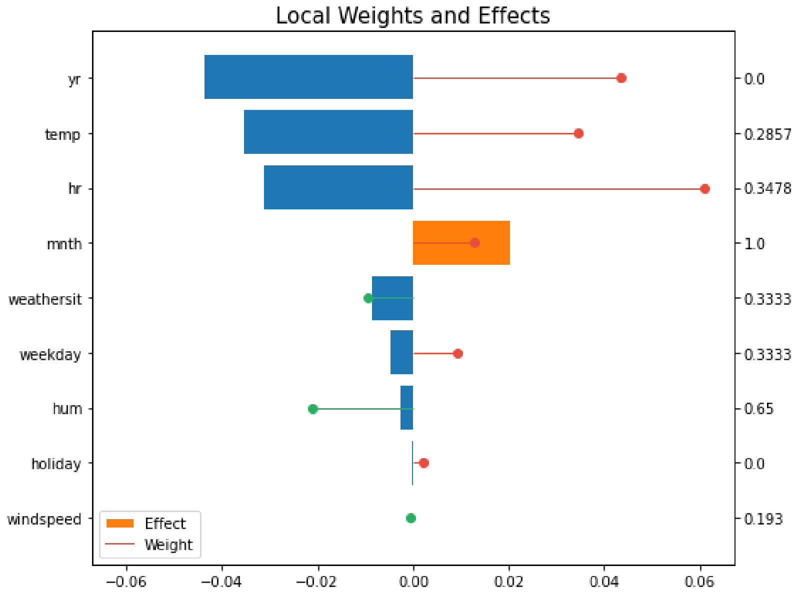
Local Explainability Tool: LIME

- For linear model $\hat{f}(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$, we know how to explain the prediction by the regression coefficients β_j for each unit change of x_j, or marginal effects $\beta_j x_j$

- For complex ML model, **LIME** by Ribeiro et al. (2016) is to fit a local linear model around the individual prediction:
  - Given point of interest $\boldsymbol{x}^*$, simulate points $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$ in the neighbourhood, and compute $\hat{f}(\boldsymbol{z}_1), \ldots \hat{f}(\boldsymbol{z}_m)$;
  - Fit a weighted linear regression model to points $\{\boldsymbol{z}_i, \hat{f}(\boldsymbol{z}_i)\}_{i=1}^{m}$ with weights inversely proportional to distance$(\boldsymbol{z}_i, \boldsymbol{x}^*)$;
  - Use the fitted local linear model $\beta_0 + \beta_1 x_1^* + \cdots + \beta_d x_d^*$ for local explanation.

- Local explanation results depend on the neighbourhood size and Gaussian sampling points (ignoring feature correlations).





Local Explainability Tool: SHAP

- **Shapley** decomposition proposed by Shapley (1953)
  - Properties: efficiency, symmetry, additivity, etc.
  - Exponential complexity in feature dimensionality.

$$\phi_i = \sum_{S \subset N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} \left(f_{S \cup \{i\}}(x) - f_S(x) \right).$$

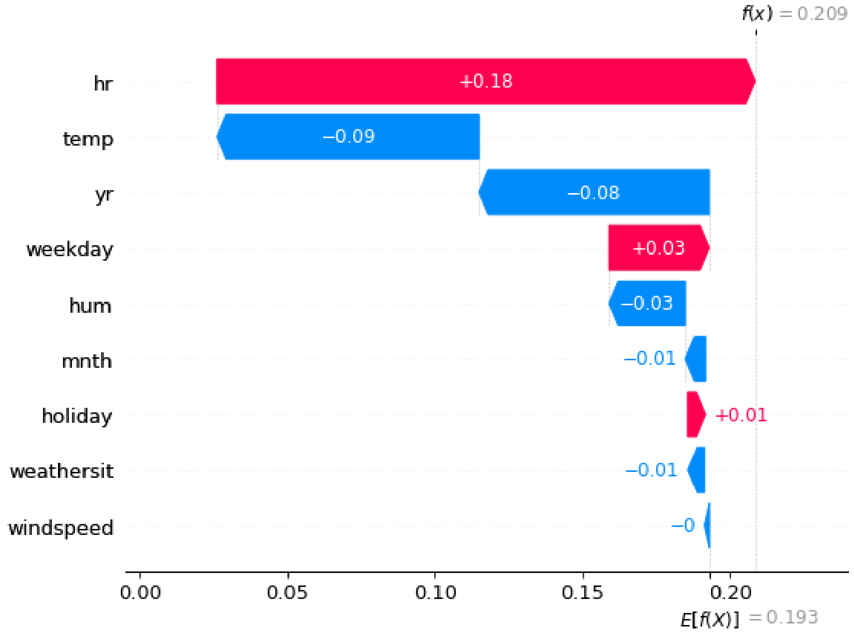
i-th feature

Sum over all sub-models

Normalization factor

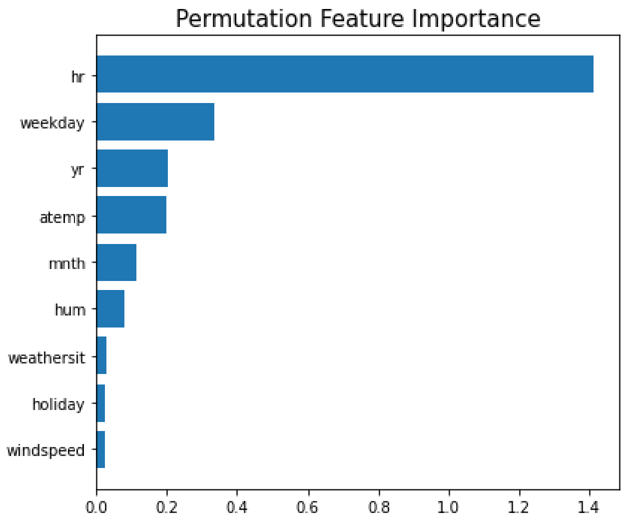
Difference in contribution to prediction with and without i-th feature

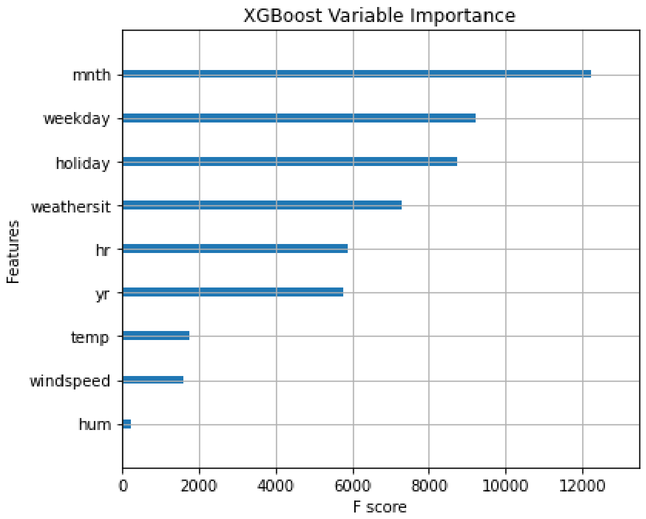
- **SHAP** by Lundberg and Lee (2016) provides multiple ways to approximately compute Shapley values:
  - KernelSHAP (slow) and TreeSHAP (fast)
  - Based on unrealistic assumptions
  - Differing results that are often not reliable.
  - Explanation results depend on input data and can sometimes be manipulated/attacked.



Global Explainability Tools: Feature Importance

- **PFI** (Permutation Feature Importance) for any prediction model
  - Randomly permute the rows for column or interest while keeping other columns unchanged
  - Compute the change in prediction performance as the measure of feature importance
- **VI** (Variable Importance) for tree-based models
  - For a single tree, importance of a variable x_j is measured by total reduction of impurity at nodes where x_j is used for splitting
  - For tree-ensemble methods, average over all trees
- **SHAP-FI** based on Shapley values
  - Average the absolute Shapley values per feature across the data
  - Super slow for computing Shapley values for all data points





Global Explainability Tools: PDP, ICE and ALE

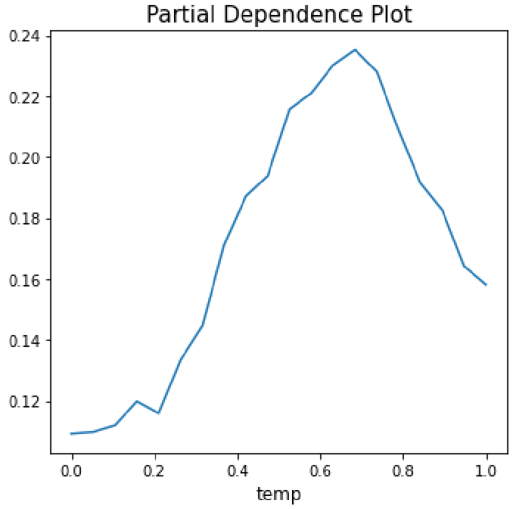
- **PDP** (partial dependence plot) is to understand how the prediction varies as a function of variables of interest, by averaging over other variables.

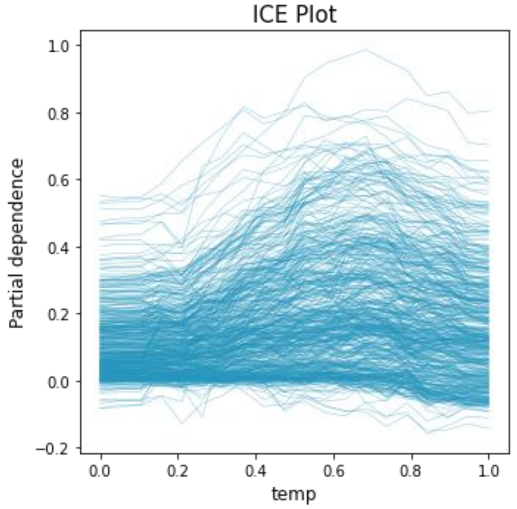
- One-dimensional PDP is most popular:

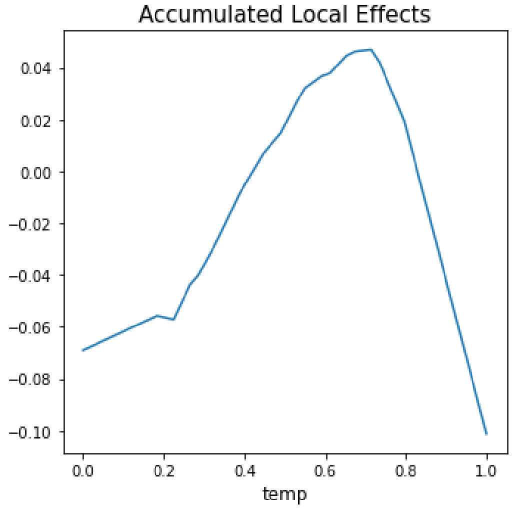
$$h_{PD}(x_j) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_j, \boldsymbol{X}_{-j,i}) \text{ (average on the data } \boldsymbol{X}\text{)}$$

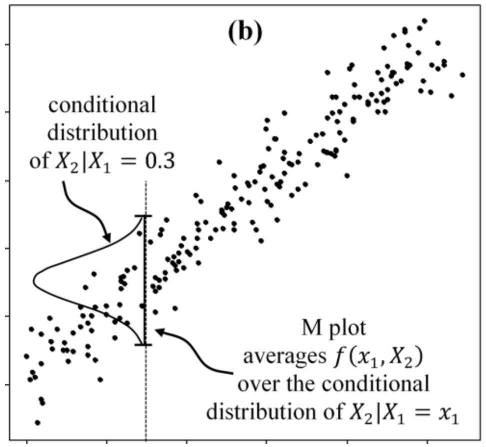
- **ICE** (individual conditional expectation) plots one curve per instance $\boldsymbol{x}_i$ in the similar fashion

- **ALE** (accumulated local effects) by Apley and Zhu (2020) is a promising alternative to PDP by avoiding extrapolation based on conditional expectations.



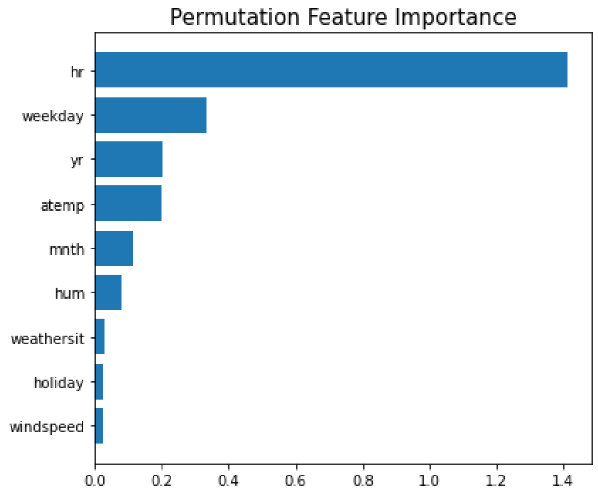


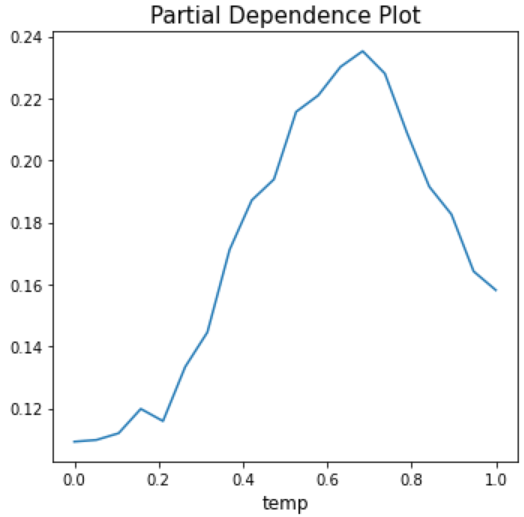


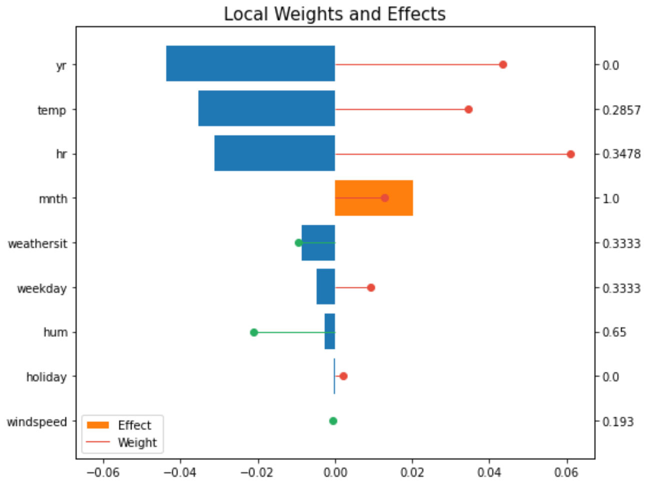


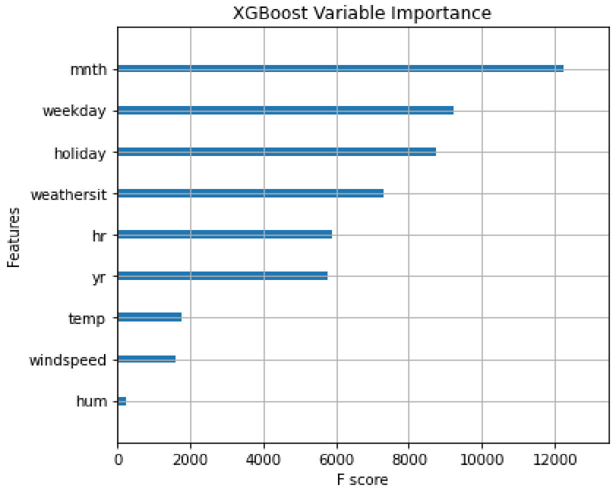
Source: Apley and Zhu (2020)

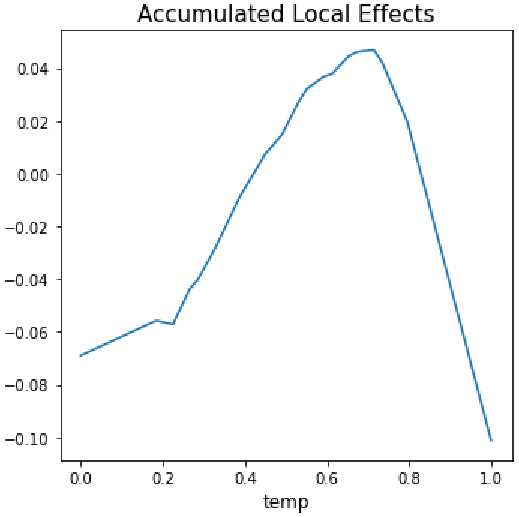
Post-hoc Explainability Puzzles

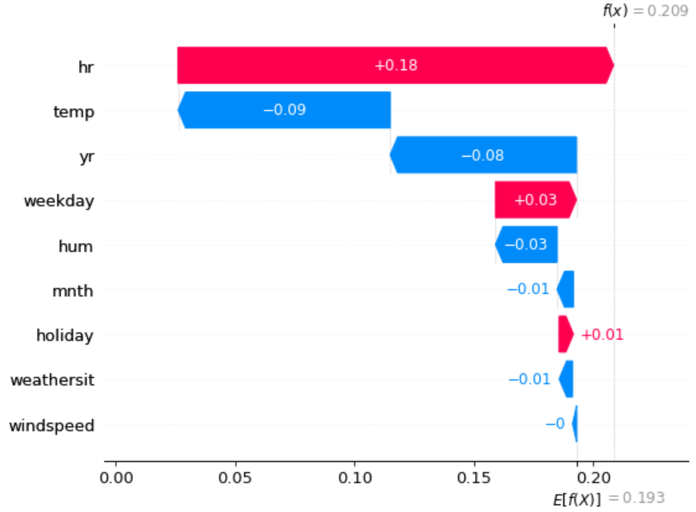












PiML Demo: BikeSharing data fit by `XGBRegressor(max_depth=7, n_estimators=500)`

Designing Interpretable ML Models

with inherently designed architecture constraints

Inherent Interpretability vs. Post-hoc Explainability

- **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 a whole set of inherently interpretable models, including GAMI-Net, EBM, and Sparse ReLU-DNNs.

- **Post-hoc explainability** is not exact and can produce misleading information. According to Cynthia Rudin, use of auxiliary post-hoc explainers creates **"double trouble"** for black-box models.
- Model-agnostic approach, one-fits-all explainability; but there is no fee lunch in interpretable ML
  - Global explainability tools: VI/FI, PDP, ALE, …
  - Local explainability tools: LIME, SHAP, …
- Post-hoc explainability tools **often produce results with disagreements** (as just shown by PiML).
- Lots of discussions recently about challenges and potential risks of using post-hoc explainers.

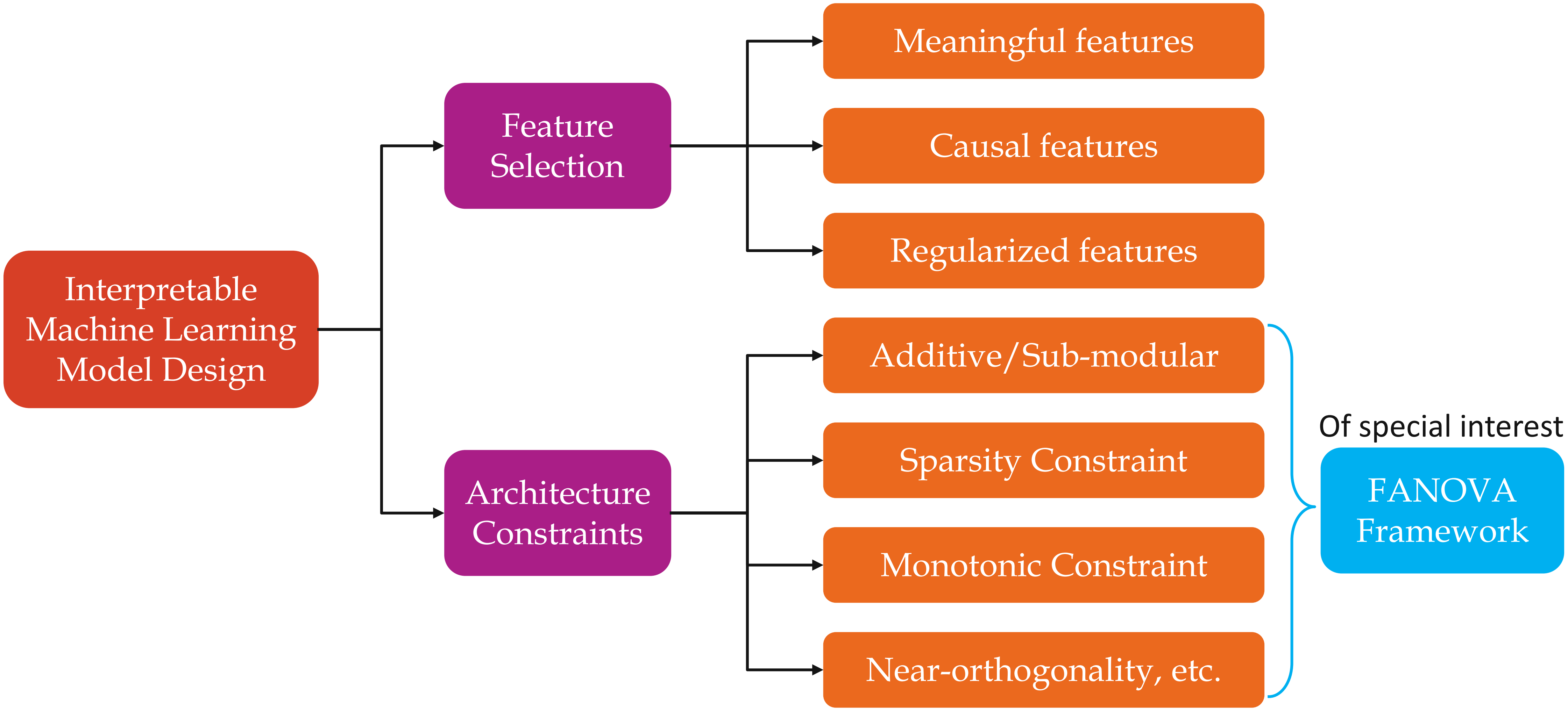
Designing Inherently Interpretable Models

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

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

Designing Inherently Interpretable Models



[1] Sudjianto and Zhang (2021): Designing Inherently Interpretable Machine Learning Models. arXiv: 2111.01743

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

FANOVA Model Design Framework

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

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

  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 …

- Two state-of-the-art interpretable models up to two-factor interactions:
  – **Explainable Boosting Machine** (Nori, et al. 2019)[3]
  – **GAMI Neural Networks** (Yang, Zhang and Sudjianto, 2021)[4]

- **PiML Toolbox** integrates EBM and GAMI-Net with Interpret/Explain/Diagnose/Compare functionalities.

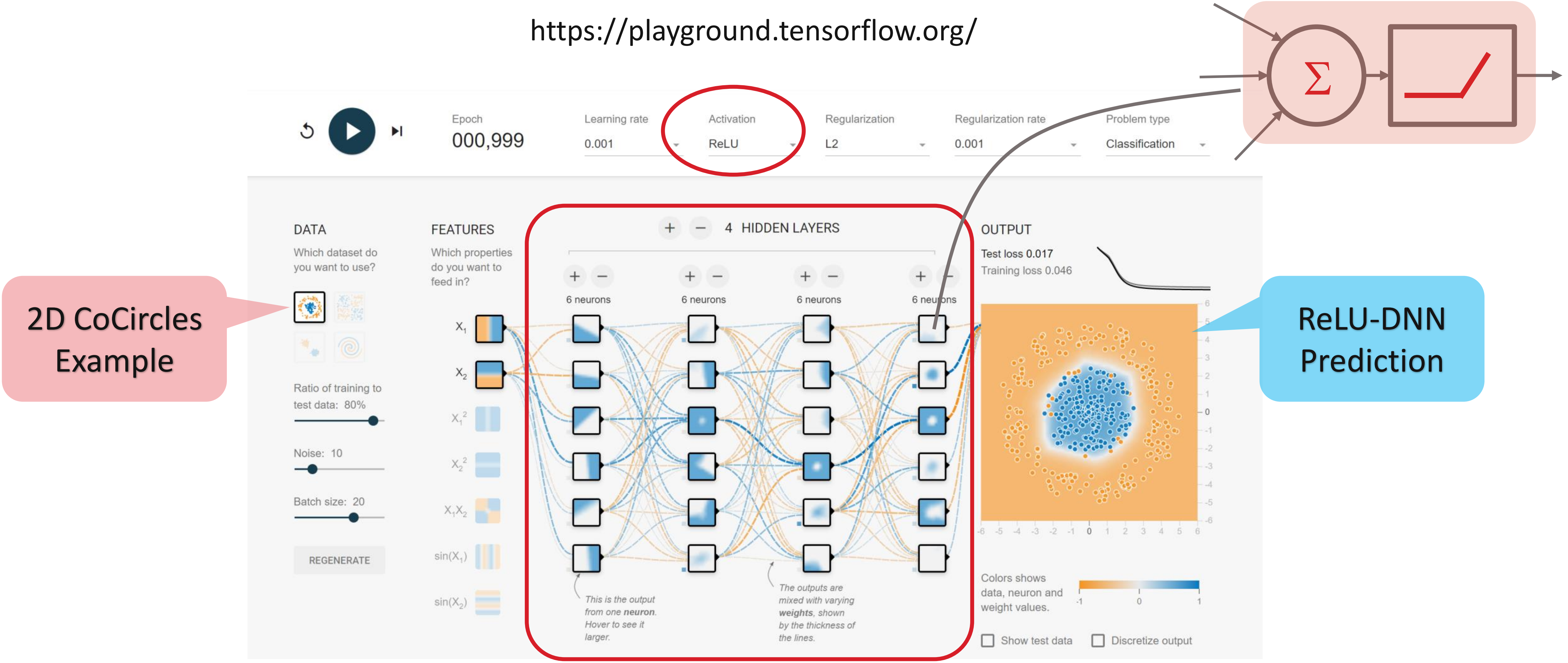
[3] Nori, Jenkins, Koch and Caruana (2019). InterpretML: A Unified Framework for Machine Learning Interpretability. arXiv: 1909.09223
[4] Yang, Zhang and Sudjianto (2021, Pattern Recognition): GAMI-Net. arXiv: 2003.07132

ReLU Deep Neural Networks

through Aletheia unwrapper and sparsification

Deep Neural Networks with ReLU Activation

https://playground.tensorflow.org/



Question: how to interpret deep neural networks (DNNs) with ReLU activation?

Deep Neural Networks: Simple 2-Layer Example

Each hidden layer:

- Linear: affine transformation

$$z_i^{(l)} = \boldsymbol{w}_i^{(l-1)} \boldsymbol{\chi}^{(l-1)} + b_i^{(l-1)}$$

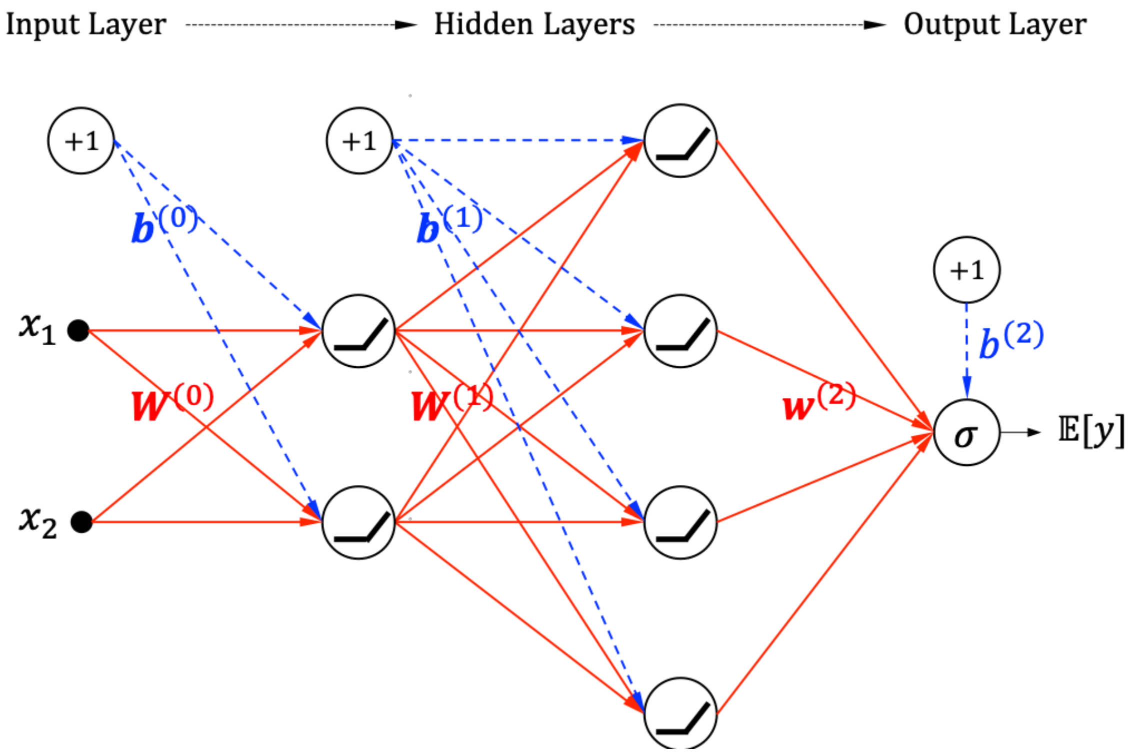
- Nonlinear: ReLU activation

$$\chi_i^{(l)} = \max\left\{0, z_i^{(l)}\right\}$$

Output layer:

$$\mathbb{E}[y] = \sigma\left(\boldsymbol{w}^{(L)} \boldsymbol{\chi}^{(L)} + \boldsymbol{b}^{(L)}\right)$$

GLM (generalized linear model)

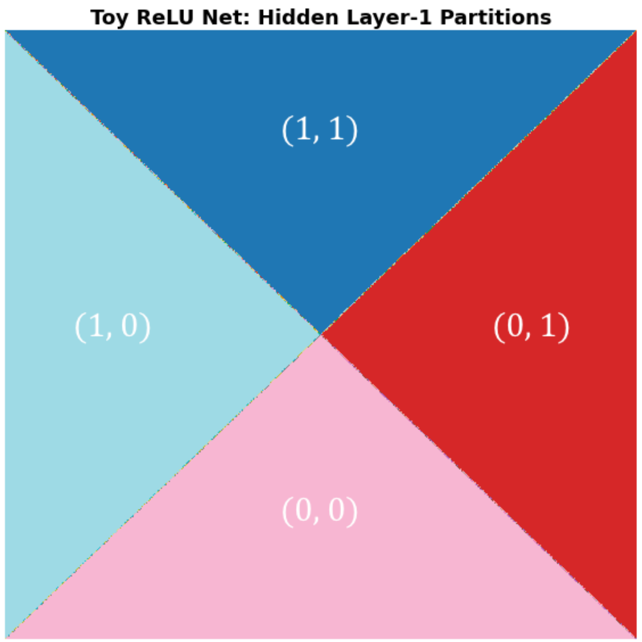


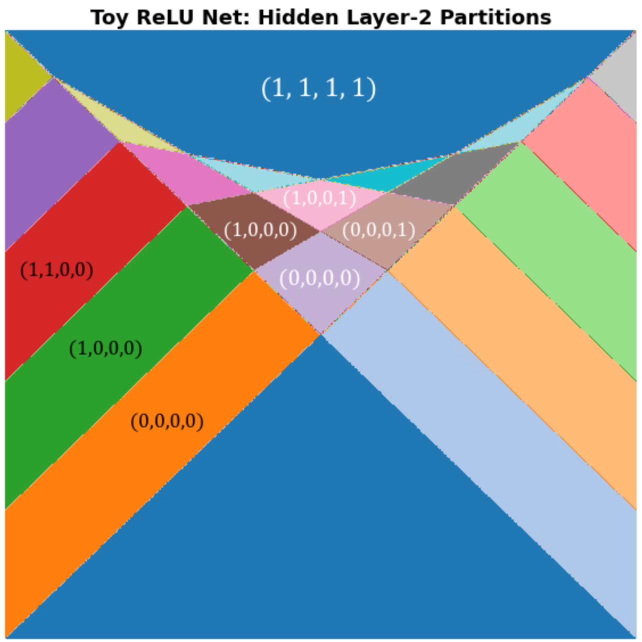
$$\boldsymbol{W}^{(0)} = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}, \quad \boldsymbol{b}^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \quad \boldsymbol{W}^{(1)} = \begin{pmatrix} 1 & 1/4 \\ 1/2 & 1/3 \\ 1/3 & 1/2 \\ 1/4 & 1 \end{pmatrix}, \quad \boldsymbol{b}^{(1)} = \frac{3}{10}\begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}.$$

Deep Neural Networks: Activation Pattern

Activation pattern: binary vector with entries indicating the on/off state of each neuron.

$$\boldsymbol{P} = [\boldsymbol{P}^{(1)}; \ldots ; \boldsymbol{P}^{(L)}] \in \{0,1\}^{\sum_{i=1}^{L} n_l}$$





Each activation pattern results in a **convex region partitioning** of the input domain.

Deep Neural Networks: Local Linear Models

Using the binary diagonal matrix induced from the layerwise activation pattern

$$\boldsymbol{D}^{(l)} = \mathrm{diag}(\boldsymbol{P}^{(l)}), \quad \text{for } l = 1, \ldots, L.$$

we obtain the closed-form local linear representation for deep ReLU networks.

Theorem 1 (Local Linear Model) *For a ReLU DNN and any of its expressible activation pattern* $\boldsymbol{P}$, *the local linear model on the activation region* $\mathcal{R}^{\boldsymbol{P}}$ *is given by*

$$\eta^{\boldsymbol{P}}(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^{\boldsymbol{P}}\boldsymbol{x} + \tilde{b}^{\boldsymbol{P}}, \quad \forall \boldsymbol{x} \in \mathcal{R}^{\boldsymbol{P}}$$

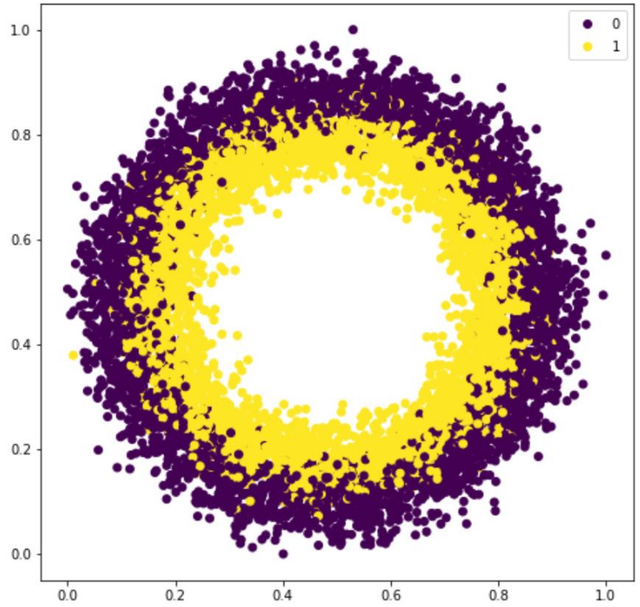
with the following closed-form parameters

$$\tilde{\boldsymbol{w}}^{\boldsymbol{P}} = \prod_{h=1}^{L} \boldsymbol{W}^{(L+1-h)}\boldsymbol{D}^{(L+1-h)}\boldsymbol{W}^{(0)}, \quad \tilde{b}^{\boldsymbol{P}} = \sum_{l=1}^{L} \prod_{h=1}^{L+1-l} \boldsymbol{W}^{(L+1-h)}\boldsymbol{D}^{(L+1-h)}\boldsymbol{b}^{(l-1)} + b^{(L)}.$$

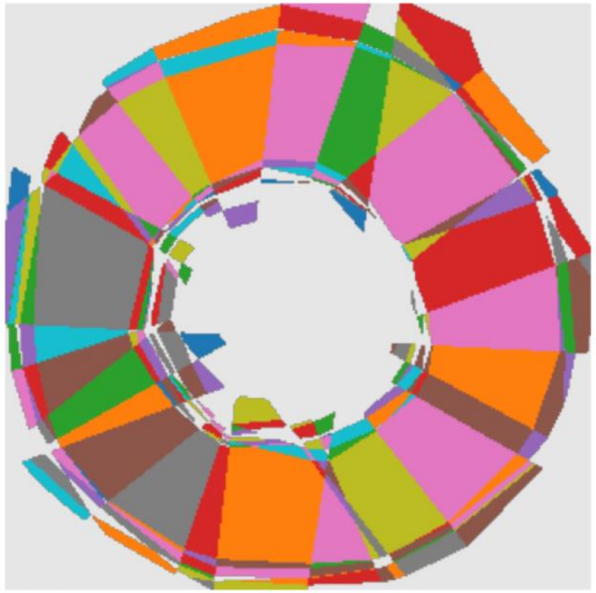
More details in our Aletheia paper (**Sudjianto, et al. 2020**) at: https://arxiv.org/abs/2011.04041

Transparency of ReLU-DNN: Data Segmentation and LLMs

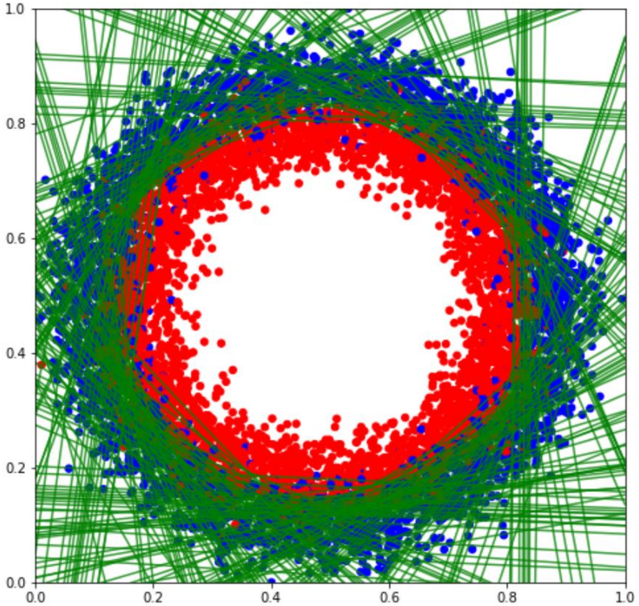
Simulated Data



Space Partitioning
into activation regions

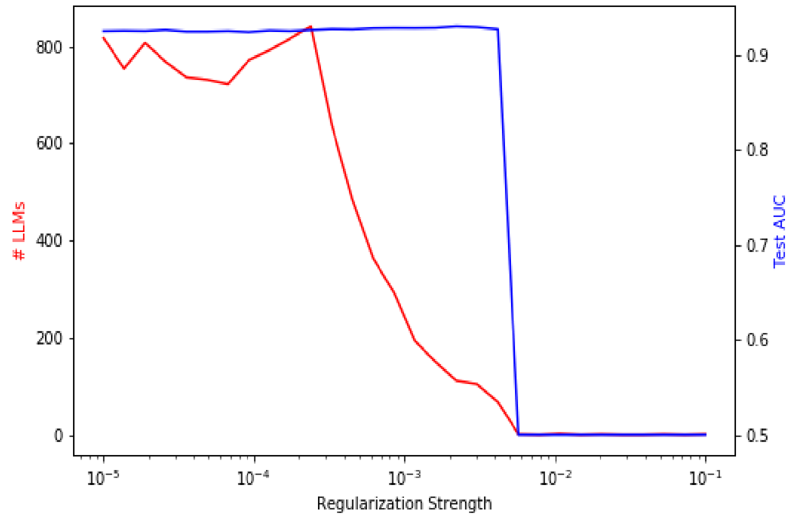


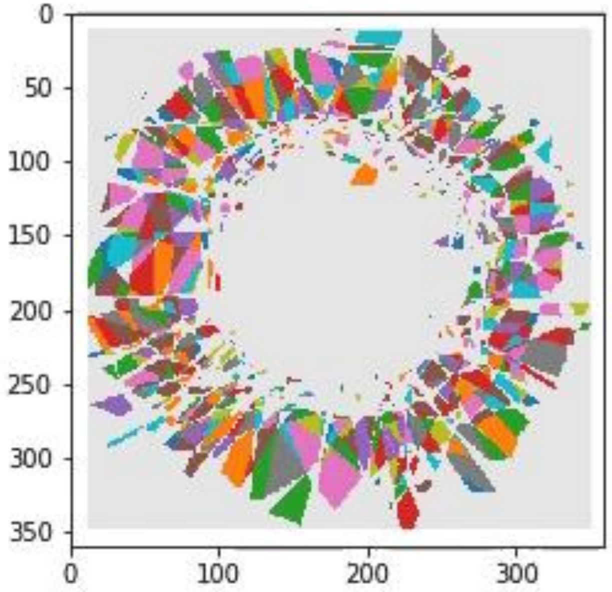
Local Linear Models



- ReLU DNN with 2 hidden layers (each 40 nodes) leads to **high performance** (AUC ~0.93) upon SGD training.
- **Unwrapped Transparency:** it generates 227 regions; over 40% LLMs have only a single instance per region.
- **Transparency ≠ Interpretability/Robustness:** raw DNNs are overparameterized with lots of unreliable LLMs.

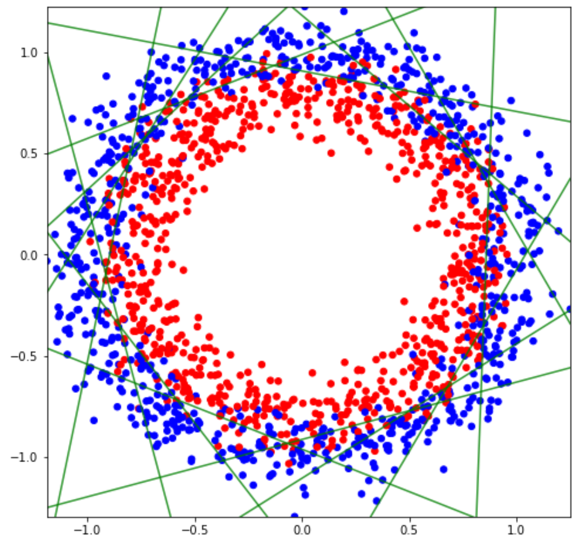
Network Simplification by L1-Regularization





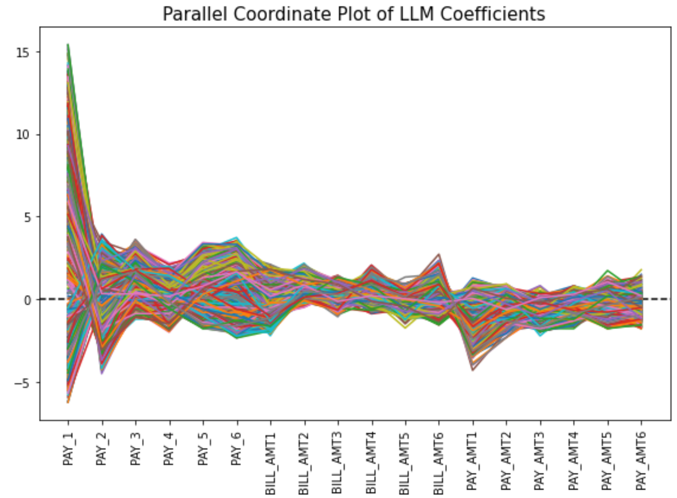
via an appropriate

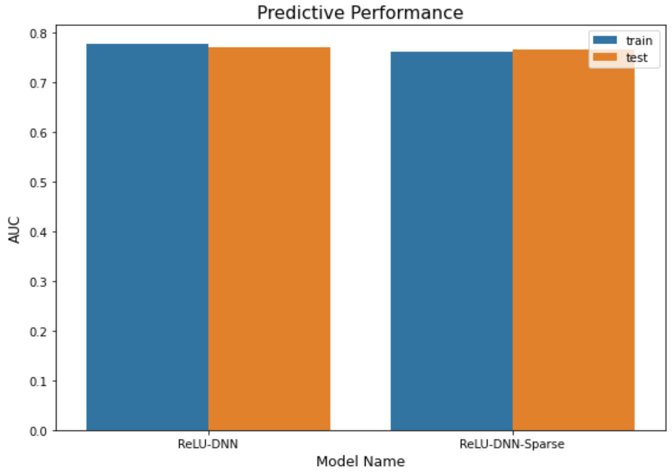
L1-hyperparameter



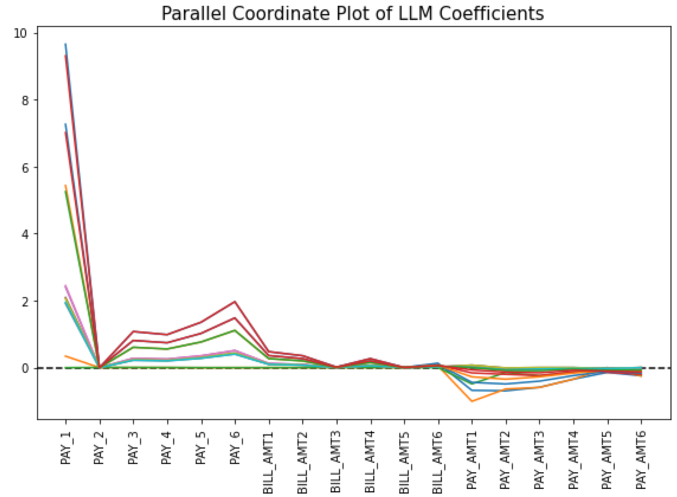
PiML Demo: TaiwanCredit Data Modeling

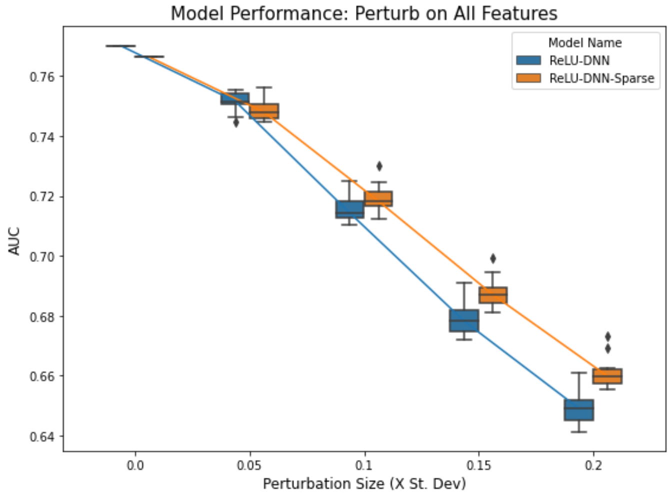
L1-reg = 0.00001
#LLMs = 6362
TestAUC = 0.7701





L1-reg = 0.0008
#LLMs = 16
TestAUC = 0.7663





PiML Demo: TaiwanCredit data fit by ReLU-DNN with L1-regularization 0.00001 vs. 0.0008

FANOVA-Interpretable Models

EBM and GAMI-Net with pairwise interaction pursuit

Explainable Boosting Machine

- The original **GA2M** (Lou, et al. 2013)[5]

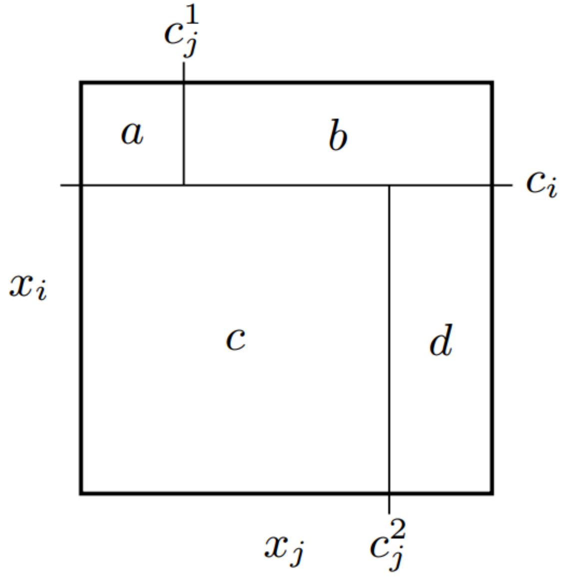
$$g(\mathbb{E}(y|\boldsymbol{x})) = \sum h_j(x_j) + \sum f_{jk}(x_j, x_k)$$

- Called "Explainable Boosting Machine" (**EBM**) by Microsoft InterpretML (Nori, et a 2019)[3], with fast implementation in C++ and Python.

- **Two-stage training algorithm:**
  - Stage 1: fit main effects by **shallow-tree boosting** in round-bin fashion. Each shallow tree splits only one variable for capturing a main effect.
  - Stage 2: fit pairwise interactions on residuals, by
    - Detect interactions by a **FAST** version of depth-2 tree algorithm;
    - For each interaction (x_j, x_k), model it by a **depth-2 tree**, either 1 cut in x_j and 2 cuts in x_k, or 2 cuts in x_j and 1 cut in x_k (pick the better one)
    - Iteratively fit all the detected interactions until convergence.

Algorithm 1 GA^{2}M Framework

$\mathcal{S} \leftarrow \emptyset$
$\mathcal{Z} \leftarrow \mathcal{U}^2$
while not converge **do**
  $F \leftarrow \arg\min_{F \in \mathcal{H}^1 + \sum_{u \in \mathcal{S}} \mathcal{H}_u} \frac{1}{2} E[(y - F(\boldsymbol{x}))^2]$
  $R \leftarrow y - F(\boldsymbol{x})$
  for all $u \in \mathcal{Z}$ **do**
    $F_u \leftarrow E[R|x_u]$
  $u^* \leftarrow \arg\min_{u \in \mathcal{Z}} \frac{1}{2} E[(R - F_u(x_u))^2]$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{u^*\}$
  $\mathcal{Z} \leftarrow \mathcal{Z} - \{u^*\}$

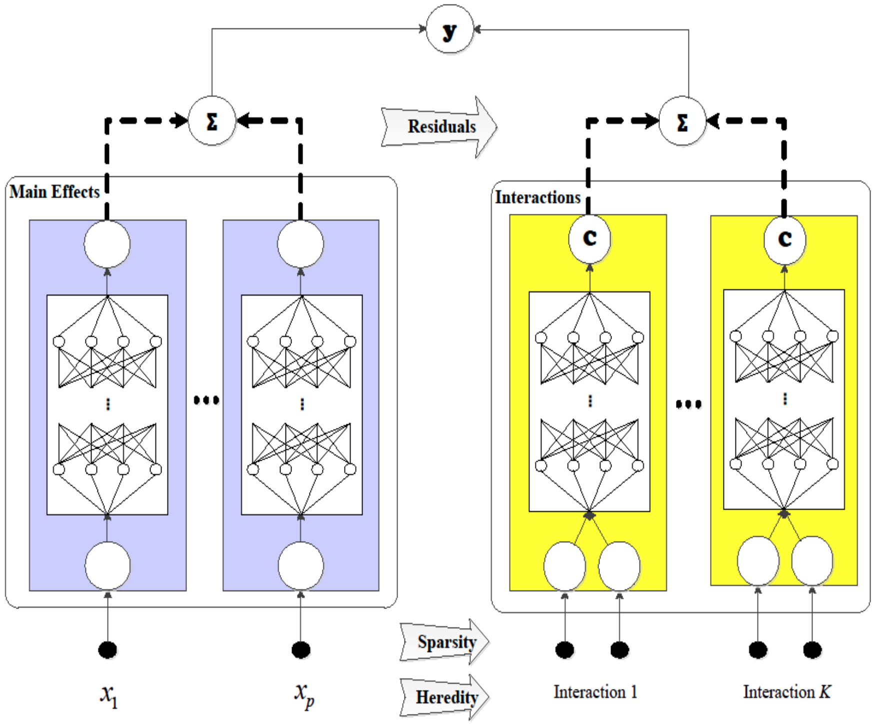


[5] Lou, Caruana, Gehrke and Hooker (2013). Accurate Intelligible Models with Pairwise Interactions. [Microsoft Research]

GAMI-Net and Interpretability Constraints

- **GAMI-Net** (Yang, Zhang and Sudjianto, 2021)[4] considered the same FANOVA form as GA2M but used neural networks instead of tree-boosting.

$$g\big(E(y|\boldsymbol{x})\big) = \mu + \sum h_j(x_j) + \sum f_{jk}(x_j, x_k)$$



- **Three-stage training algorithm:**
  - Stage 1: train the main effect subnetworks and **prune** the trivial ones by validation performance.
  - Stage 2: train pairwise interactions on residuals, by
    - Select candidate interactions by heredity constraint;
    - Evaluate their scores (by FAST) and select top-K interactions;
    - Train the selected two-way interaction subnetworks;
    - Prune trivial interactions by validation performance.
  - Stage 3: retrain main effects and interactions simultaneously for fine-tuning network parameter.

GAMI-Net and Interpretability Constraints

GAMI-Net incorporates the following constraints inherently.

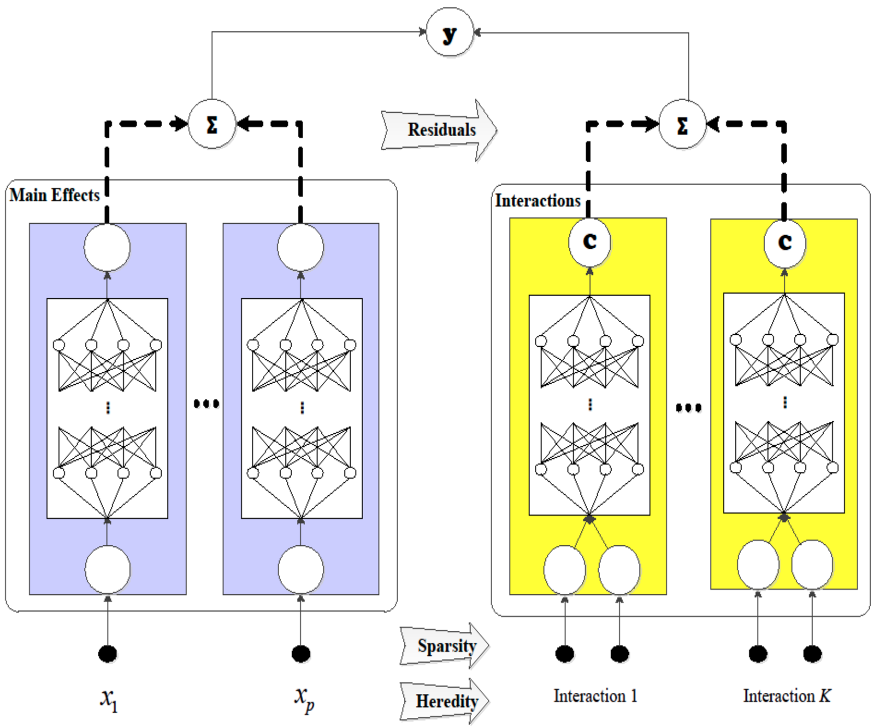
- **Sparsity**: select only the most important main effects and pairwise interactions.
- **Heredity**: a pairwise interaction is selected only if at least one (or both) of its parent main effects is selected.
- **Marginal Clarity**: enforce the pairwise interactions to be nearly orthogonal to the main effects, by imposing penalty

$$\Omega(h_j, f_{jk}) = \left|\frac{1}{n}\sum h_j(x_j) f_{jk}(x_j, x_k)\right|$$

- **Monotonicity**: certain features can be constrained to be monotonic increasing or decreasing, by imposing penalty

$$\Omega(x_j) = \max\left\{-\frac{\partial g}{\partial x_j}, 0\right\} \text{(if inceasing) or } \max\left\{\frac{\partial g}{\partial x_j}, 0\right\} \text{(if deceasing)}$$

$$g(E(y|\boldsymbol{x})) = \mu + \sum h_j(x_j) + \sum f_{jk}(x_j, x_k)$$



Effect Importance and Feature Importance

- In GAMI-Net, each **effect importance** (before normalization) is given by

$$D(h_j) = \frac{1}{n-1}\sum_{i=1}^{n} h_j^2(x_{ij}), \qquad D(f_{jk}) = \frac{1}{n-1}\sum_{i=1}^{n} f_{jk}^2(x_{ij}, x_{ik})$$

- For prediction at $\boldsymbol{x}_i$, the **local feature importance** is given by

$$\phi_j(x_{ij}) = h_j(x_{ij}) + \frac{1}{2}\sum_{j \neq k} f_{jk}(x_{ij}, x_{ik})$$

- For GAMI-Net (or EBM), the **global feature importance** is given by

$$\mathrm{FI}(x_j) = \frac{1}{n-1}\sum_{i=1}^{n}\left(\phi_j(x_{ij}) - \overline{\phi_j}\right)^2$$

- The effect can be visualized by a line plot (for main effect) or heatmap (for pairwise interaction).

EBM and GAMI-Net: Pros and Cons

- Both EBM and GAMI-Net are inherently interpretable models of FANOVA form up to two-factor interactions:

$$g(E(y|\boldsymbol{x})) = \mu + \sum h_j(x_j) + \sum f_{jk}(x_j, x_k)$$

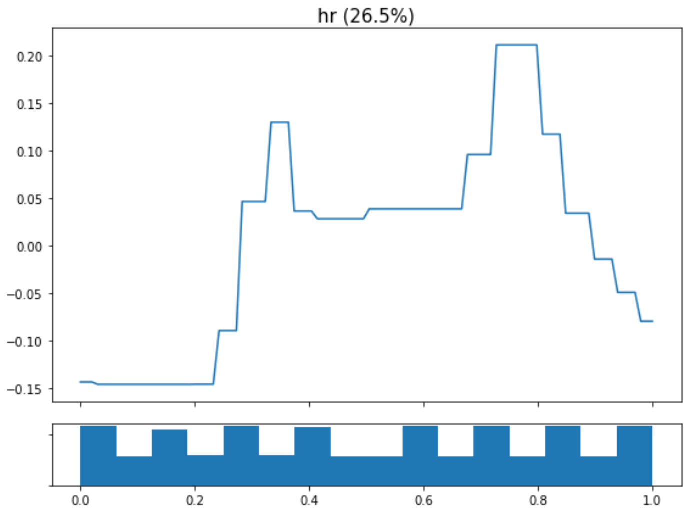
	Pros	Cons
EBM	1. Fast computation; 2. Nice visualization; 3. Good support from Microsoft Research.	1. Non-smooth and jumpy shape functions; 2. Lacking monotonicity constraint; 3. Lacking pruning for main effects.
GAMI-Net	1. Support constraints like sparsity, heredity, marginal clarity and monotonicity; 2. Continuous and smooth shape functions; 3. Nice visualization; 4. Importance at effect and feature levels; 5. TensorFlow and PyTorch implementations.	1. Subnetwork training is slow, but can be accelerated by warm initialization; 2. Sometimes slight sacrifice on predictive performance.

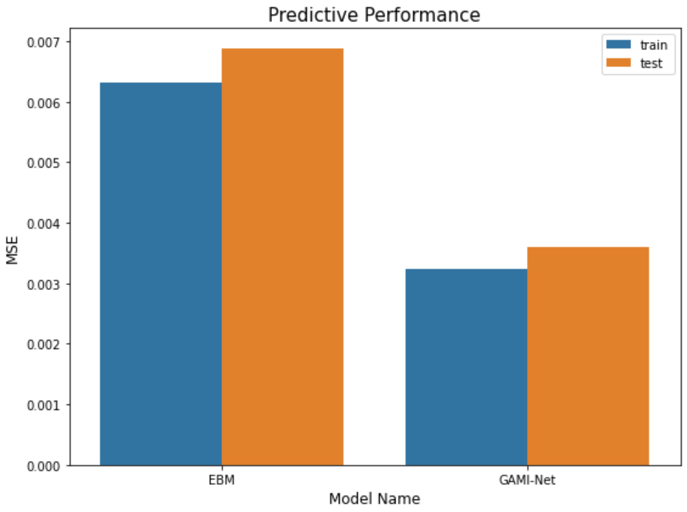
PiML Demo: BikeSharing Data Modeling

EBM:

1 + 9 + 10 effects

TestMSE = 0.0069

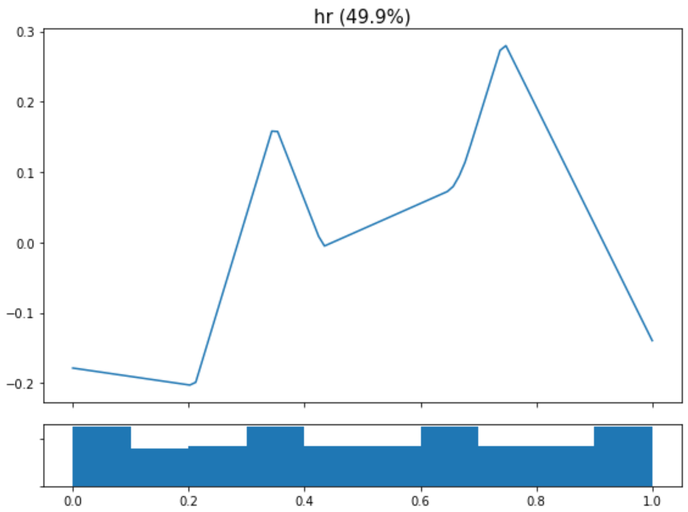


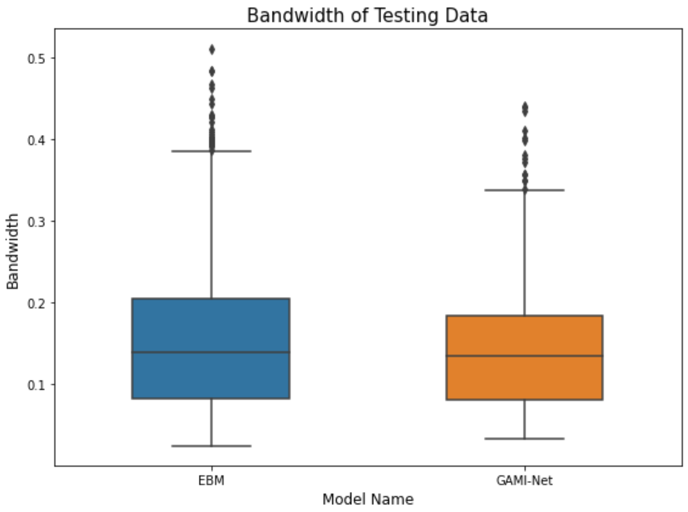


GAMI-Net:

1 + 7 + 9 effects

TestMSE = 0.0036





PiML Demo: BikeSharing data fit by FANOVA-interpretable EBM and GAMI-Net

WELLS FARGO

Thank you

Aijun Zhang, Ph.D.
Senior Lead, Advanced Technologies for Modeling (AToM), CMoR
https://www.linkedin.com/in/ajzhang/

Agus Sudjianto, Ph.D.
EVP, Head of Corporate Model Risk (CMoR)
https://www.linkedin.com/in/agus-sudjianto-76519619/