WELLS FARGO

Machine Learning Model Validation

Risk Americas Workshop
New York, NY

Agus Sudjianto and Vijay Nair
Corporate Model Risk, Wells Fargo
May 9, 2022

Agenda

- **9:00 – 9:30: Introduction** – Agus Sudjianto
- **9:30-10:45: Machine Learning and Explainability** – Vijay Nair and Sri Krishnamurthy
- 10:45-11:00: **Break**
- **10:45-11:45: Unwrapping ReLU Networks** – Agus Sudjianto
- **11:45-12:45 Inherently Interpretable Models** – Vijay Nair and Sri Krishnamurthy
- **12:45-1:15: Lunch Break**

- **1:15-2:15: Outcome Testing** – Agus Sudjianto
- **2:15-3:15 Hands-on Exercises** – Sri Krishnamurthy
- **3:15-3:30: Break**
- **3:30-4:30 Bias and Fairness** – Nick Schimdt
- **4:30-5:00: ModelOp Presentation** – Jim Olsen

Overview

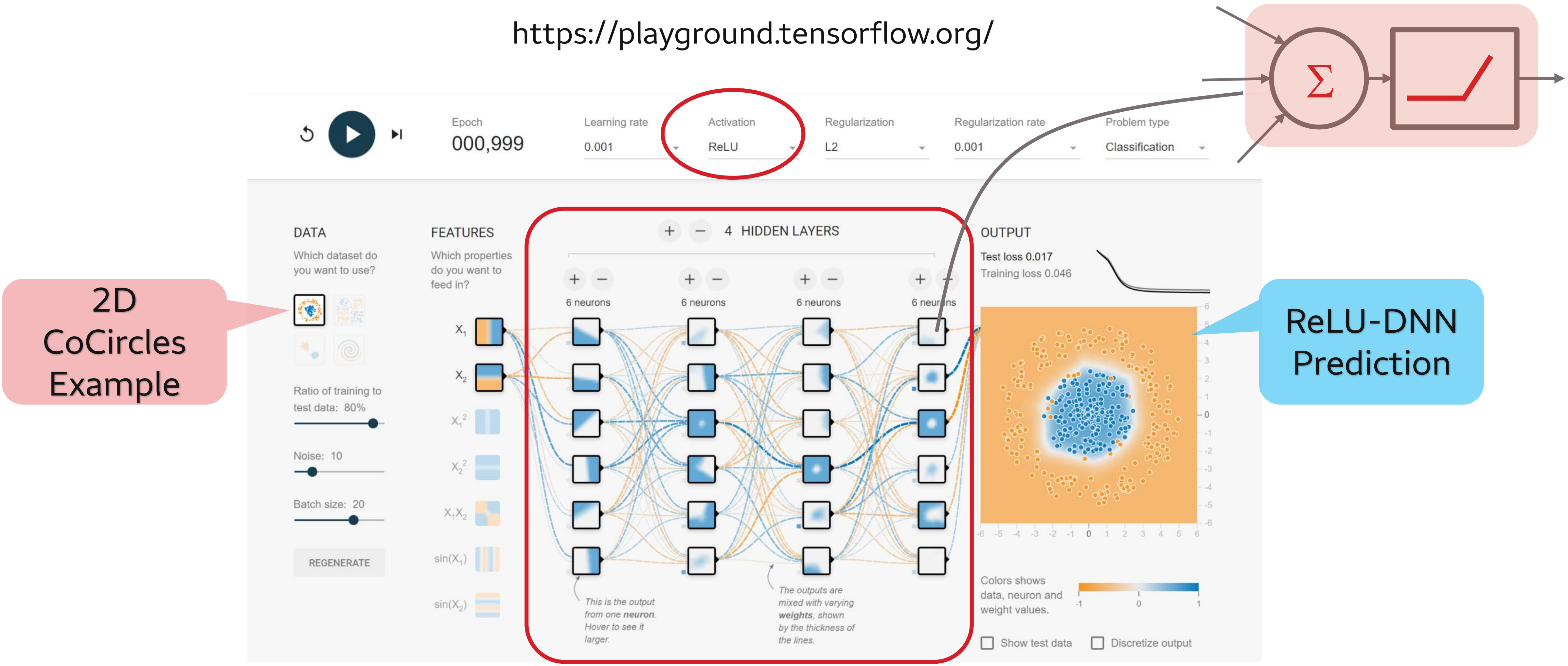
1. Introduction: Risk Dynamics, Conceptual Soundness and Outcome Testing
2. Supervised Machine Learning: Algorithms and Explainability
3. **Deep ReLU Networks and Inherent Interpretation**
4. Inherently Interpretable Models
5. Outcome Testing

Outline

- Deep Neural Networks
- Unwrapping Deep ReLU Networks
  - Activation Pattern and Region Partitioning
  - Local Linear Models
- Network Simplification by Sparse Regularization
- Inherent Interpretation for ReLU-DNNs
- PiML Low-code and High-code Demo

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

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

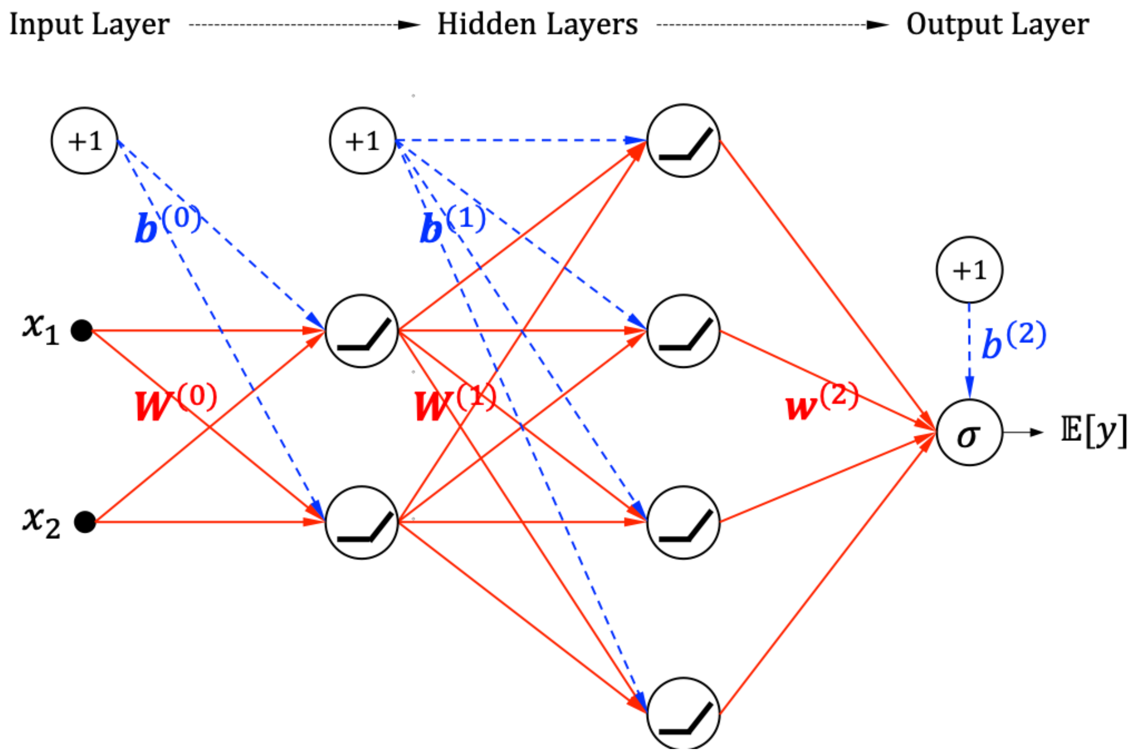
- Nonlinear: ReLU activation

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

Output layer:

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

GLM (generalized linear model)

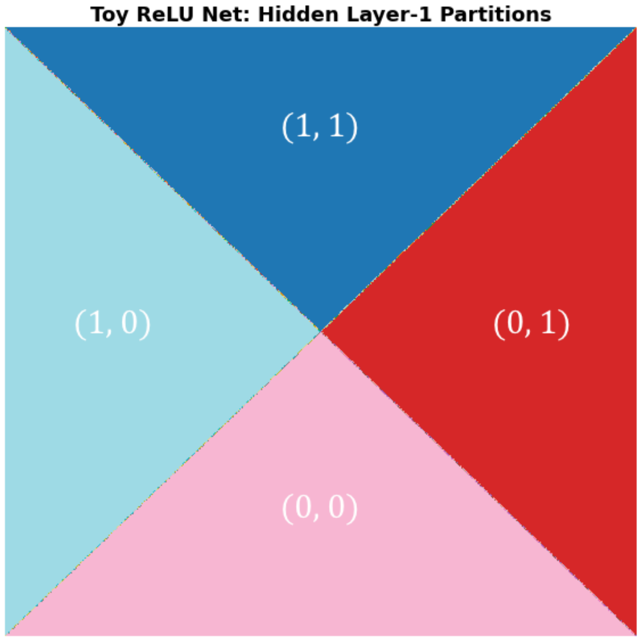


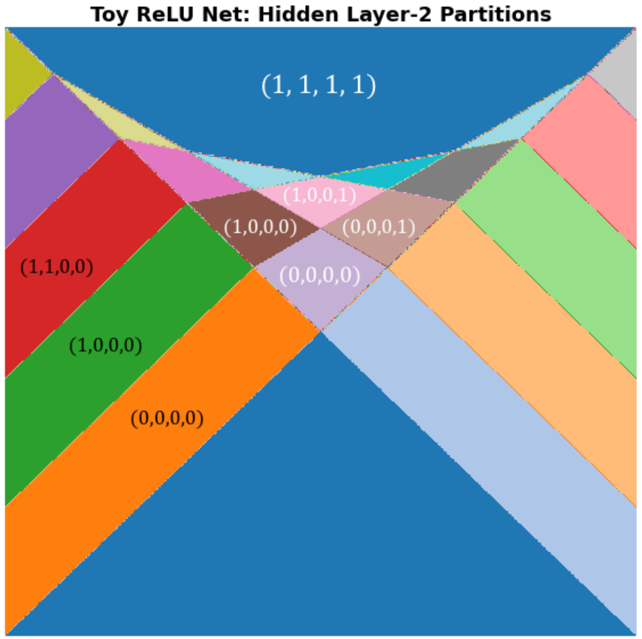
$$\boldsymbol{W}^{(0)} = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}, \quad \boldsymbol{b}^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \quad \boldsymbol{W}^{(1)} = \begin{pmatrix} 1 & 1/4 \\ 1/2 & 1/3 \\ 1/3 & 1/2 \\ 1/4 & 1 \end{pmatrix}, \quad \boldsymbol{b}^{(1)} = \frac{3}{10}\begin{pmatrix}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

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 $\boldsymbol{D}^{(l)} = \text{diag}(\boldsymbol{P}^{(l)}), l = 1, \ldots, L,$ 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*

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

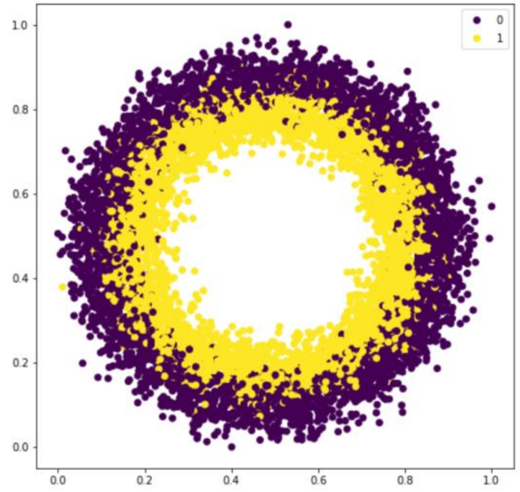
with the following closed-form parameters

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

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

Deep Neural Networks: CoCircles Example

CoCircles data:



- True decision function:

$$x_1^2 + x_2^2 = \alpha$$

- sklearn.datasets.make_make_circles

  with n_samples=10000 and noise=0.1.

```python
from sklearn.neural_network import MLPClassifier, MLPRegressor
from sklearn.metrics import roc_auc_score, mean_squared_error

model = MLPClassifier(hidden_layer_sizes=[40] * 2, activation="relu",
                      solver='adam', learning_rate_init=0.001,
                      max_iter=2000, early_stopping=True,
                      n_iter_no_change=100, validation_fraction=0.2,
                      random_state=0)
model.fit(train_x, train_y)
print("Training AUC:", roc_auc_score(train_y, model.predict_proba(train_x)[:,1]).round(4))
print("Testing AUC:", roc_auc_score(test_y, model.predict_proba(test_x)[:,1]).round(4))
```

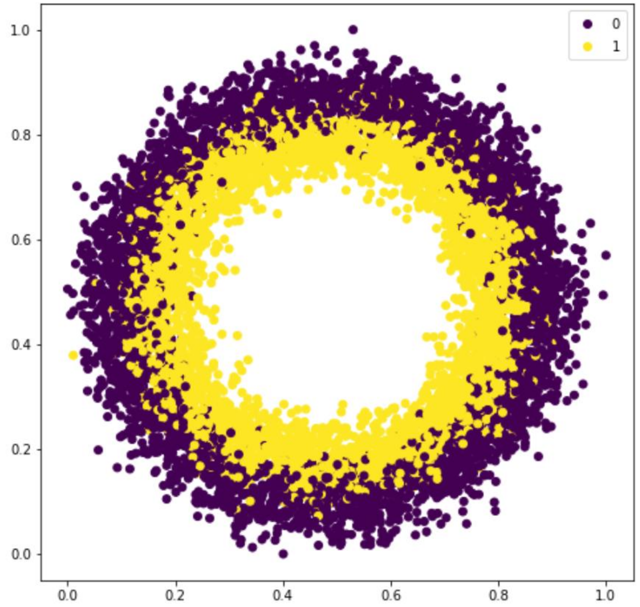
```
Training AUC: 0.9209
Testing AUC: 0.9285
```

	Count	Response Mean	Response Std	Local AUC	Global AUC
0	637.0	0.502355	0.500387	0.905747	0.497014
1	596.0	0.521812	0.499944	0.907813	0.504353
2	571.0	0.549912	0.497939	0.866713	0.496603
3	510.0	0.494118	0.500456	0.923373	0.501198
4	447.0	0.472036	0.499777	0.922705	0.500129
..	...	...	...	...	...
222	1.0	1.000000	NaN	NaN	0.496506
223	1.0	1.000000	NaN	NaN	0.501111
224	1.0	1.000000	NaN	NaN	0.503415
225	1.0	1.000000	NaN	NaN	0.495903
226	1.0	1.000000	NaN	NaN	0.504317

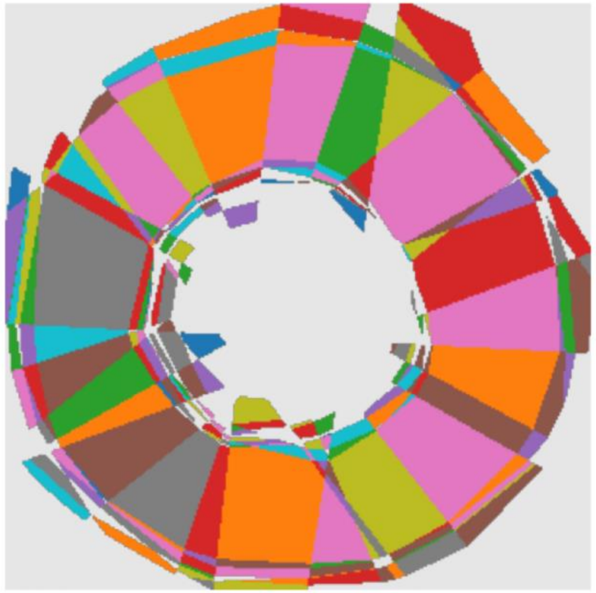
```
Total LLMs: 227
Trivial regions: 44.0%
```

Transparency of ReLU-DNN: Data Segmentation and LLMs

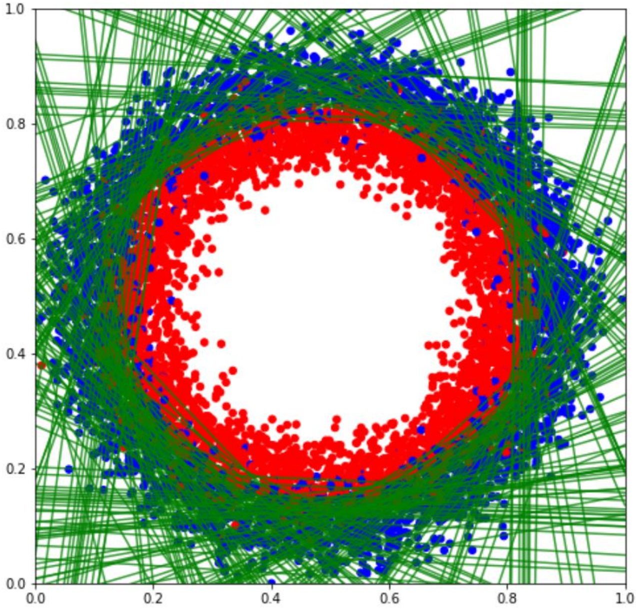
Simulated Data



Space Partitioning
into activation regions

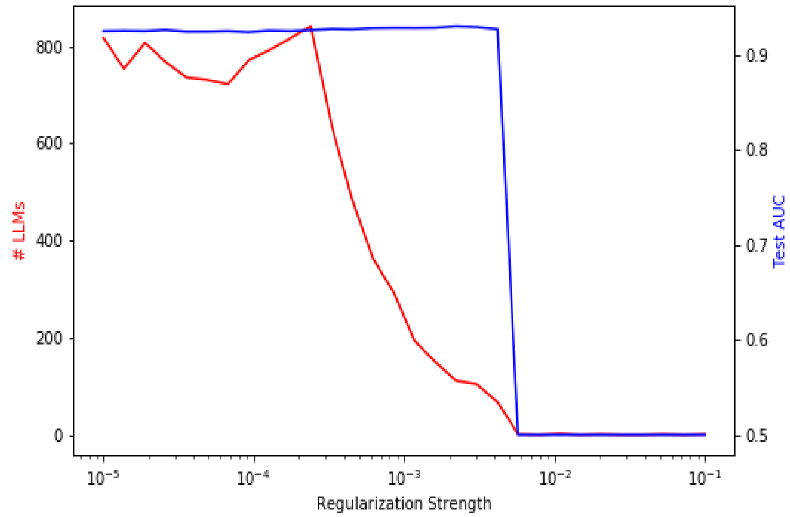


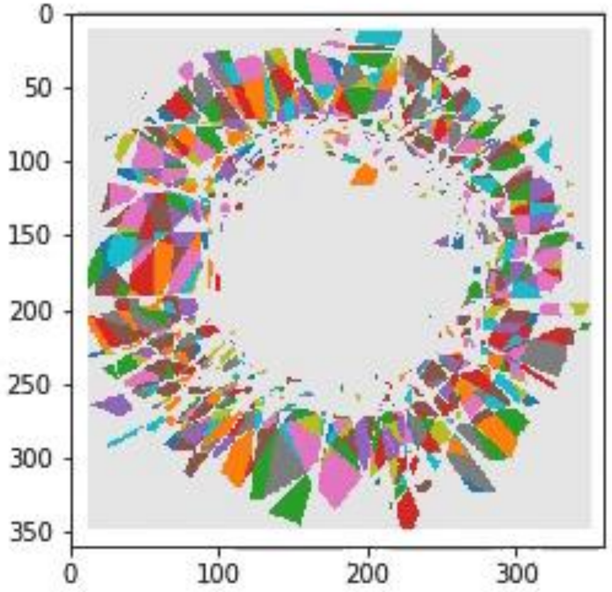
Local Linear Models



- 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.

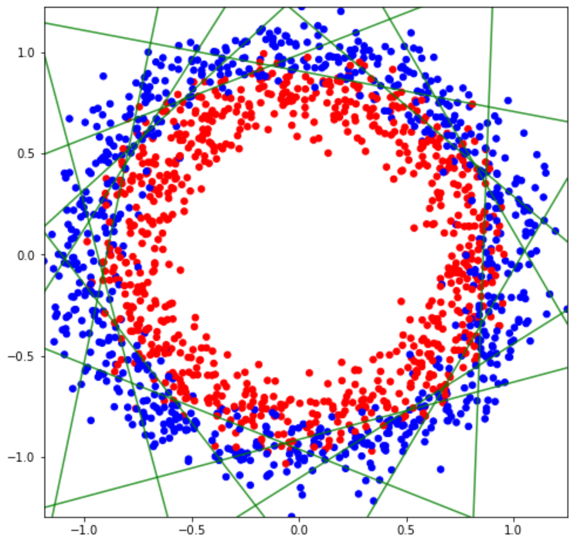
Network Simplification by L1-Regularization

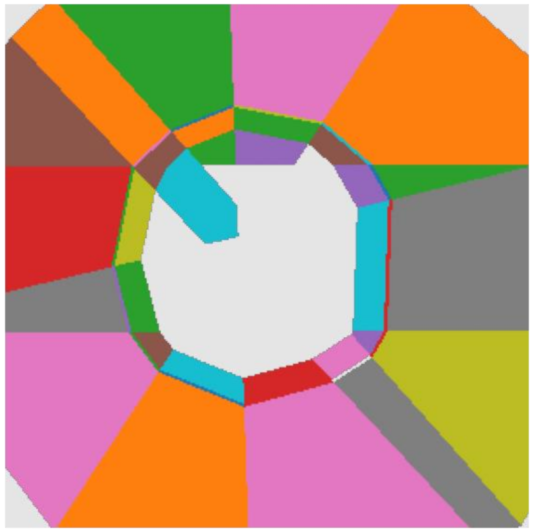




via an appropriate

L1-hyperparameter



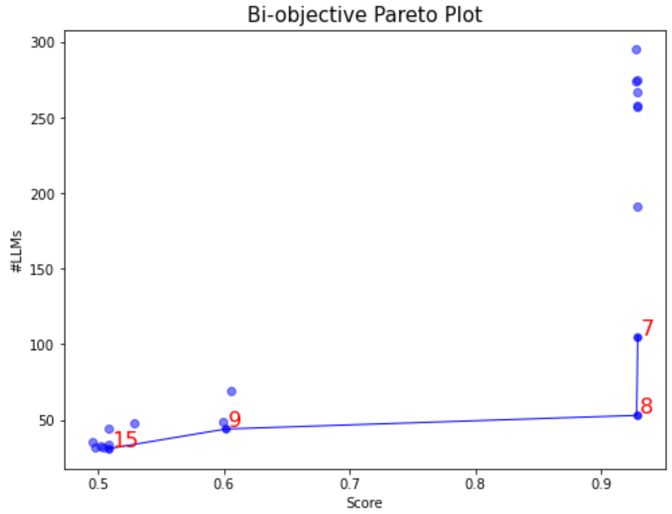


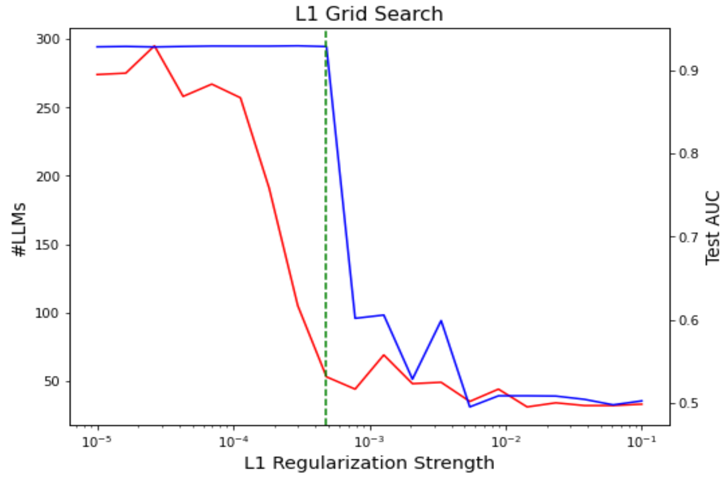
Network Simplification by L1-Regularization

- **PiML Toolbox** supports L1-regularization to shrink the node weights towards zero, so as to reduce total number of LLMs.
- We may perform **bi-objective optimization** for tuning such L1 hyperparameter, then visualize by Pareto plot.
- For ReLU-DNN [40, 40] on the CoCircles data, using L1 regularization strength 0.0005, would reduce the number of LLMs to 50, while maintaining the same level of predictive performance.

```
reg = 0.0005
tmp = ReluDNNClassifier([40, 40], l1_reg = reg, random_state = 0)
tmp.fit(train_x, train_y)
clf = UnwrapperClassifier(tmp.coefs_, tmp.intercepts_)
clf.fit(train_x, train_y)
auc = roc_auc_score(test_y, tmp.predict_proba(test_x)[:,1])
nllms = clf.nllms
print("AUC =", auc.round(4), "\n#LLMs =", nllms)
```

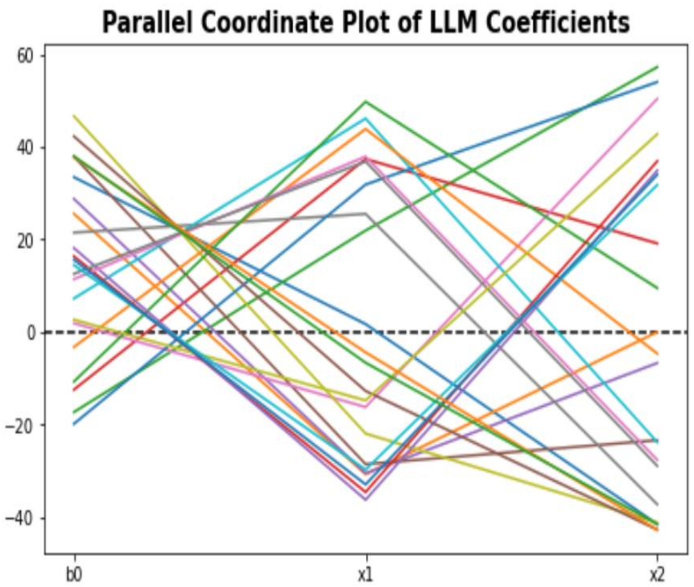
```
AUC = 0.9294
#LLMs = 50
```

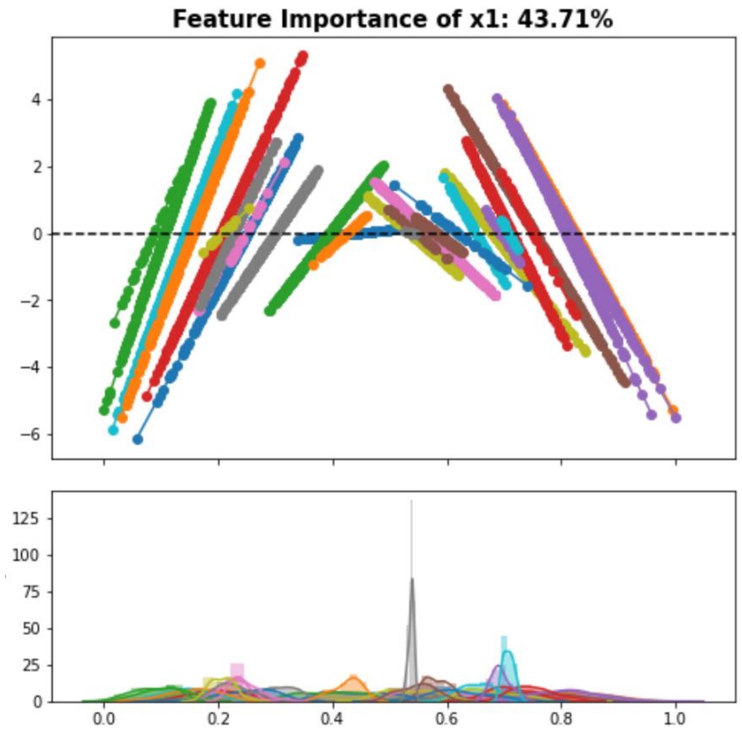


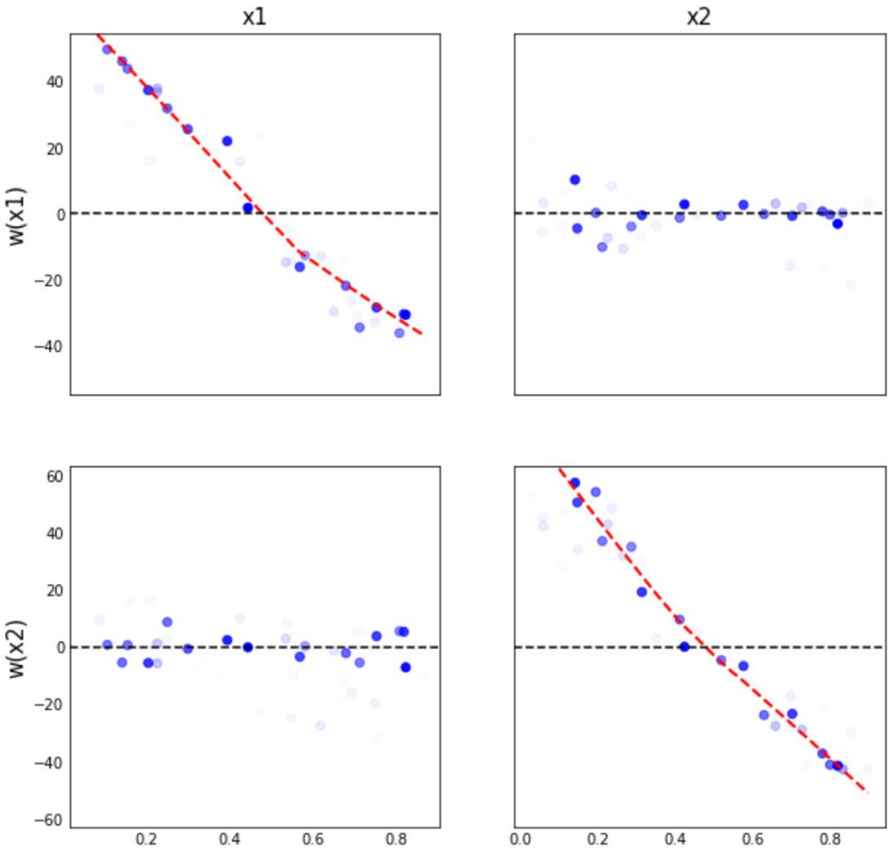


Deep Neural Networks: Inherent Interpretation

- PiML Toolbox supports the rich Aletheia functionalities for ReLU DNNs (Sudjianto, et al. 2020).
- **LLM-based interpretation**: Parallel Coordinates, Local Linear Profiles, Pairwise Interaction plots.

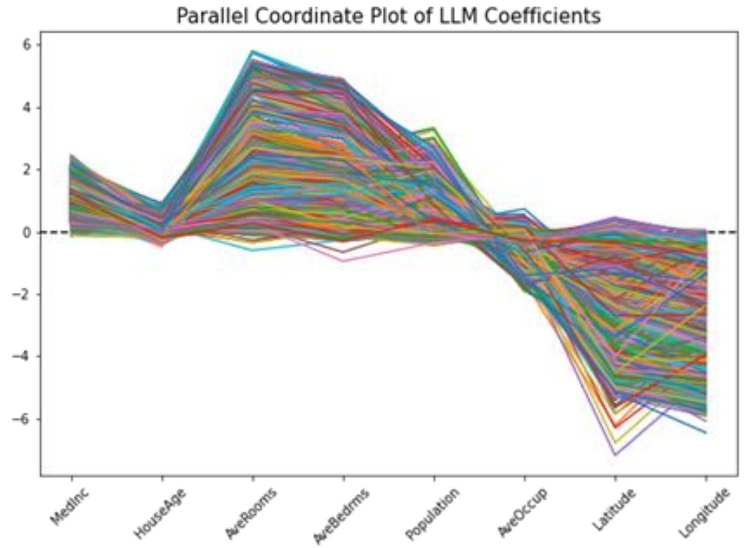






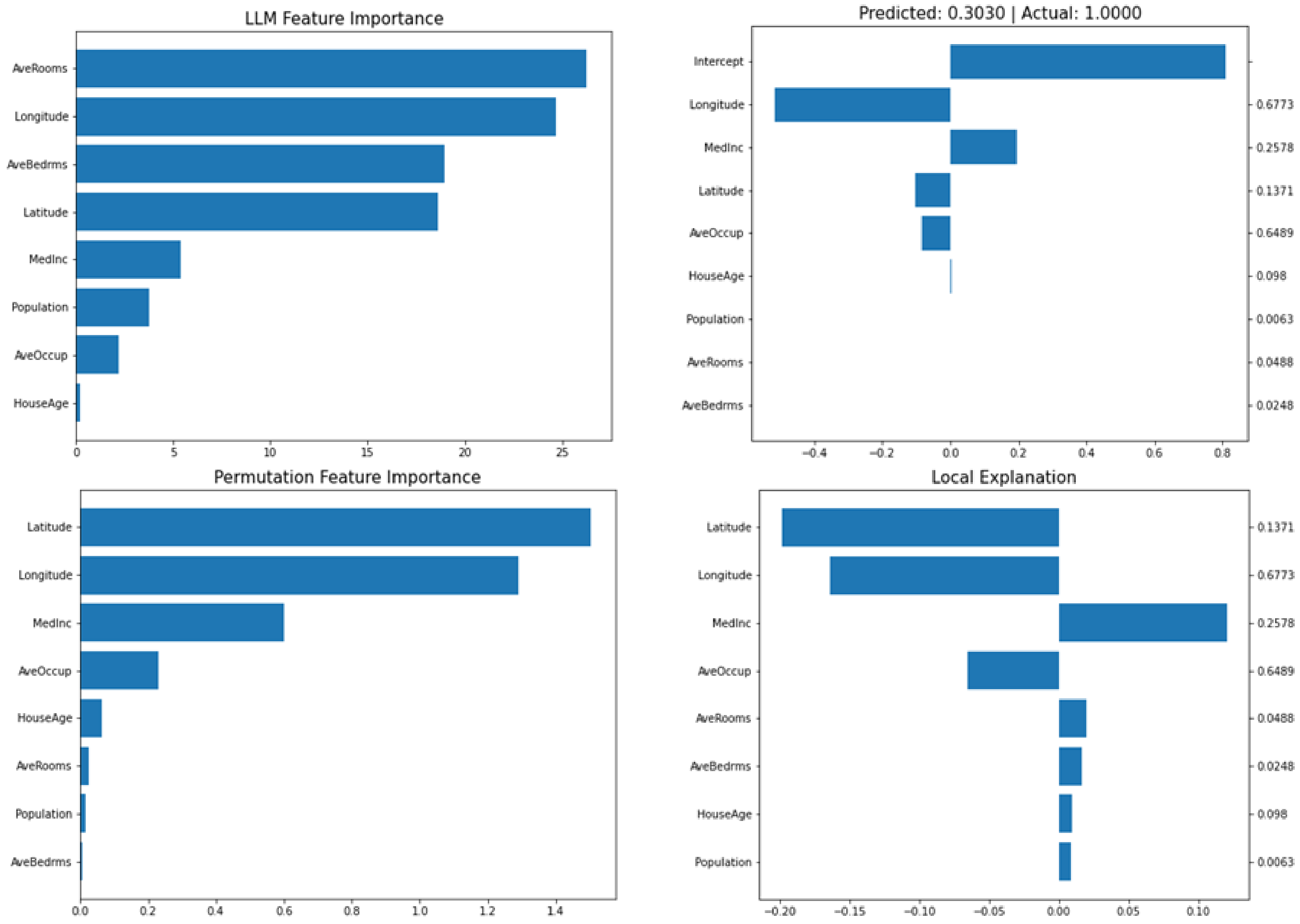
PiML for California Housing Price

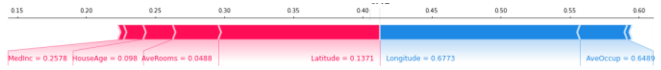
Inherent Interpretability vs. Post Hoc



Exact from model

Post-hoc explainer





PiML Demo: Low-code Mode

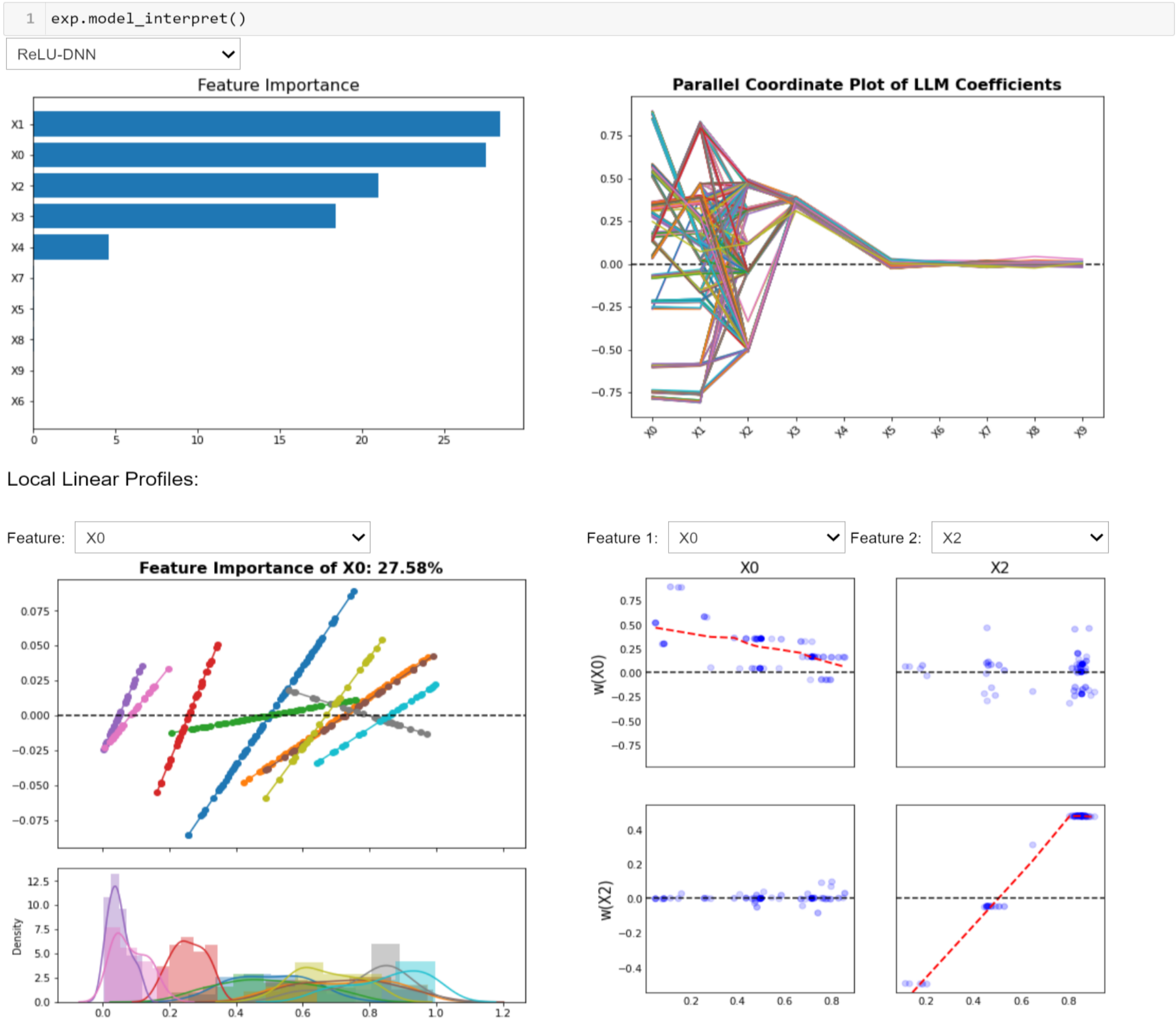
Friedman1 data:

- Multivariate independent features $\boldsymbol{x}$ uniformly distributed on [0,1]
- Continuous response generated by

$$y(\boldsymbol{x}) = 10\sin(\pi x_0 x_1) + 20(x_2 - 0.5)^2$$
$$+20x_3 + 10x_4 + \epsilon$$

  depending only $x_0 \sim x_4$.

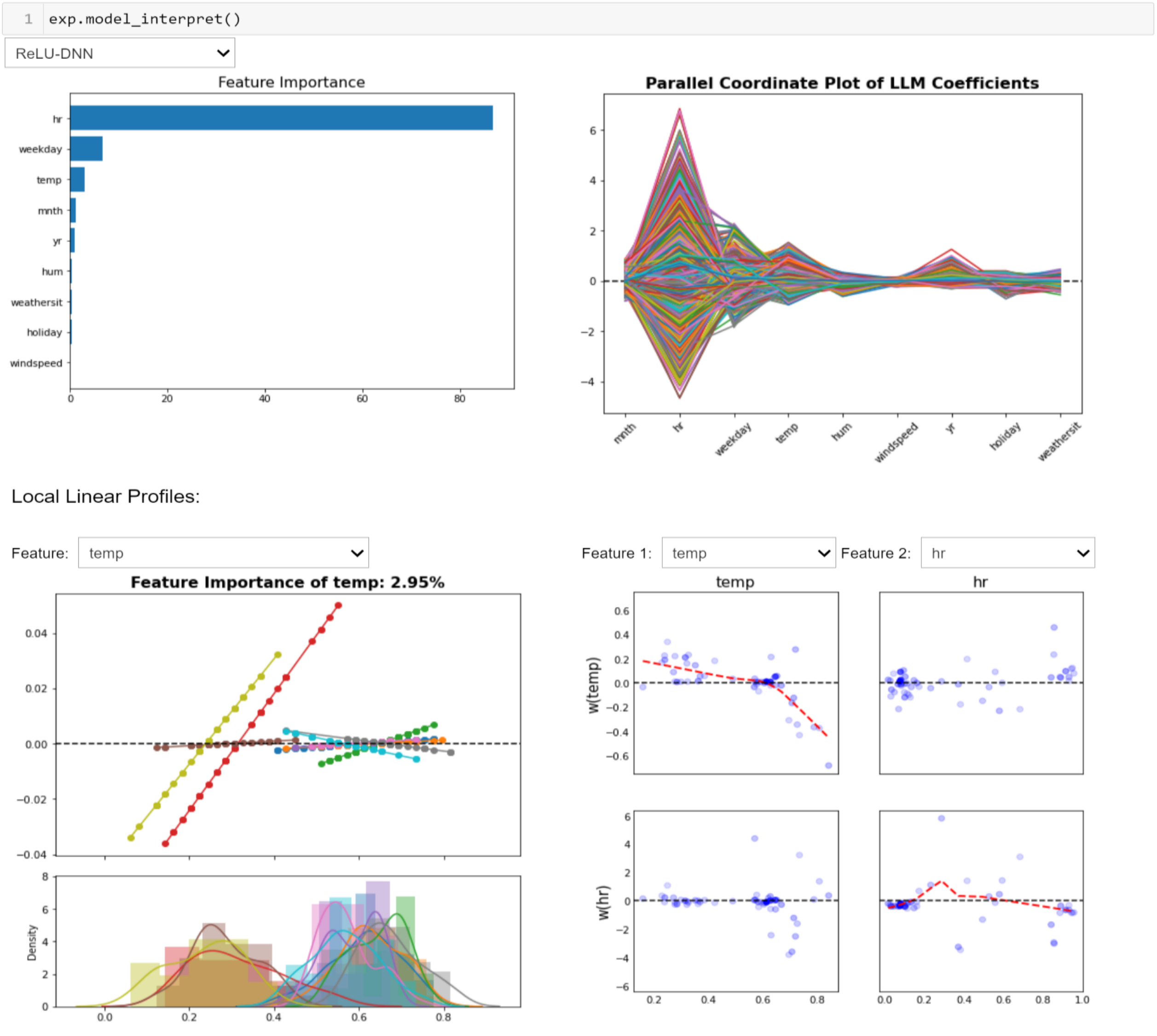
- sklearn.datasets.make_friedman1 with n_samples=10000, n_features=10, and noise=0.1.
- **PiML toolbox** calls “ReluDNNRegressor” with network size [40, 40]. Test R2 = 99.68%



PiML Demo: Low-code Mode

Bike Sharing data:

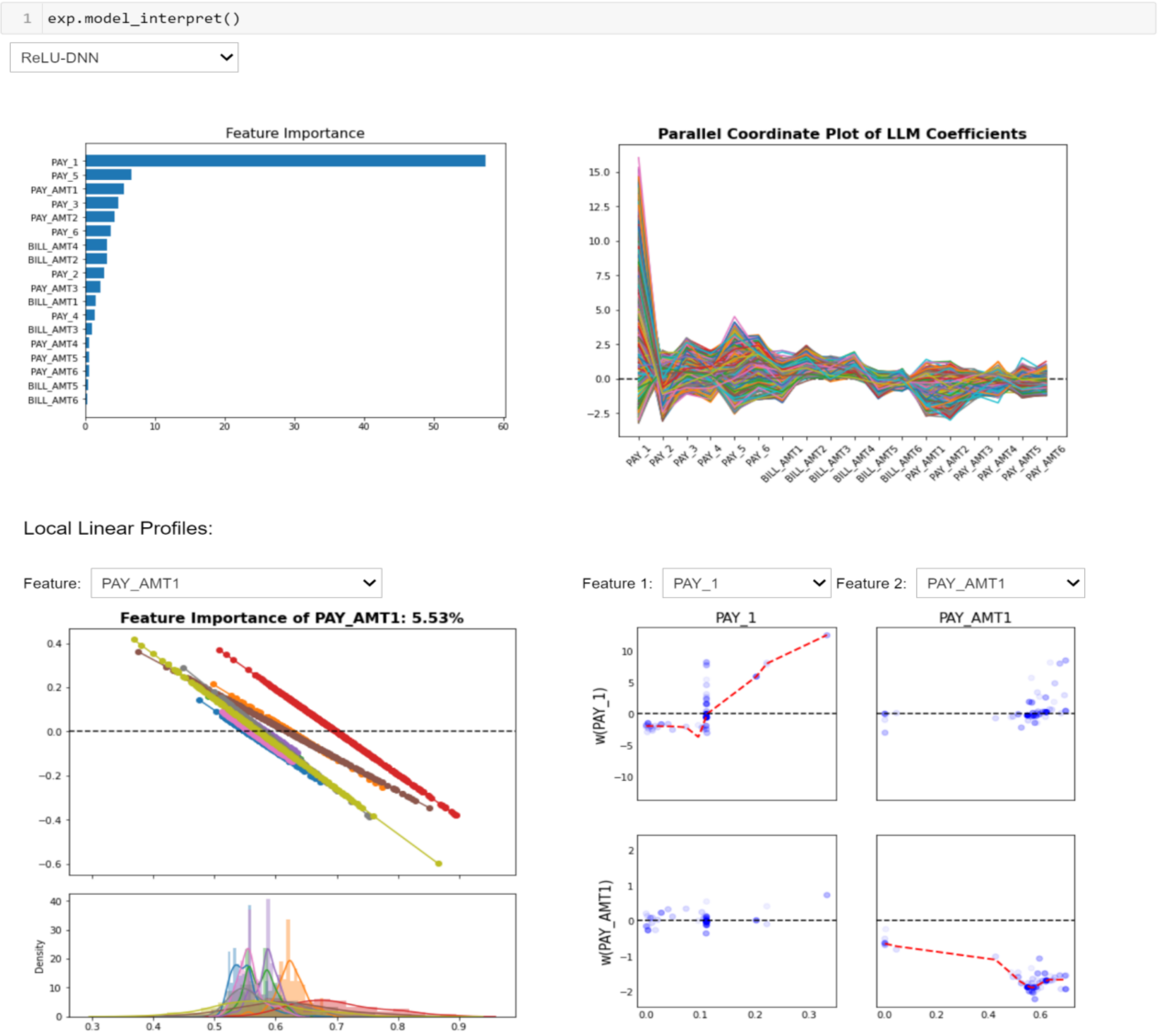
- Another popular benchmark UCI dataset consisting of hourly count of rental bikes between years 2011 and 2012 in Capital bikeshare system.
- Sample size: 17379
- The features include weather conditions, precipitation, day of week, season, hour of the day, etc.
- The response is count of total rental bikes.
- **PiML toolbox** calls "ReluDNNRegressor" with network size [40, 40]. Test R2 = 88.50%



PiML Demo: Low-code Mode

Taiwan Credit data:

- A popular benchmark UCI/Kaggle dataset: default of credit card clients (n=30000) in Taiwan from 200504 to 200509.
- We use only non-demographic features as predictors: Pay1~6 status, Bill_AMT1~6, and Pay_AMT1~6.
- The response is the indicator of default payment in next month.
- **PiML toolbox** calls “ReluDNNClassifier” with network size [40, 40]. Test AUC = 0.7741.



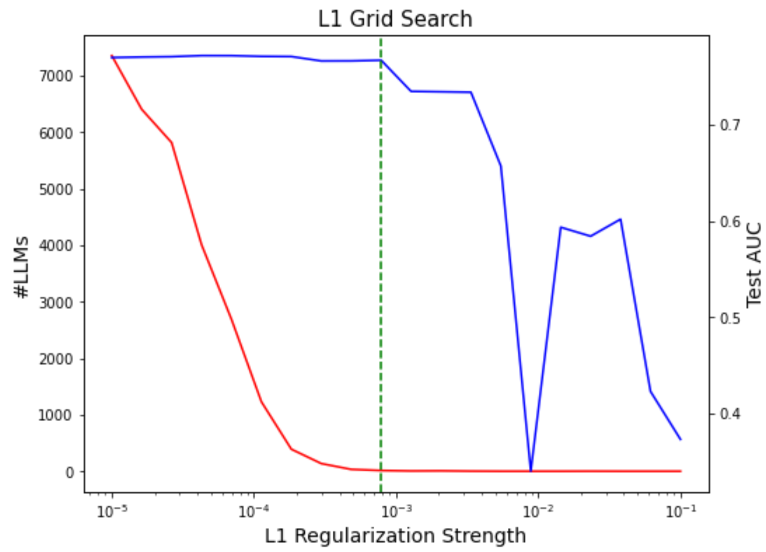
PiML Demo: High-code Mode

- Previously, the raw ReLUDNNClassifier [40,40] on Taiwan Credit data:

Raw DNN: TestAUC = 0.7741, #LLMs = 4333

- Let's simplify it by increasing the L1-regularization strength.
- **PiML toolbox** may run ReLUDNNClassifier [40,40] with L1 grid search, and finds L1_reg = 0.000785, resulting

Sparse DNN: TestAUC = 0.7671, #LLMs = 8



- Which DNN should we use in practice?