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

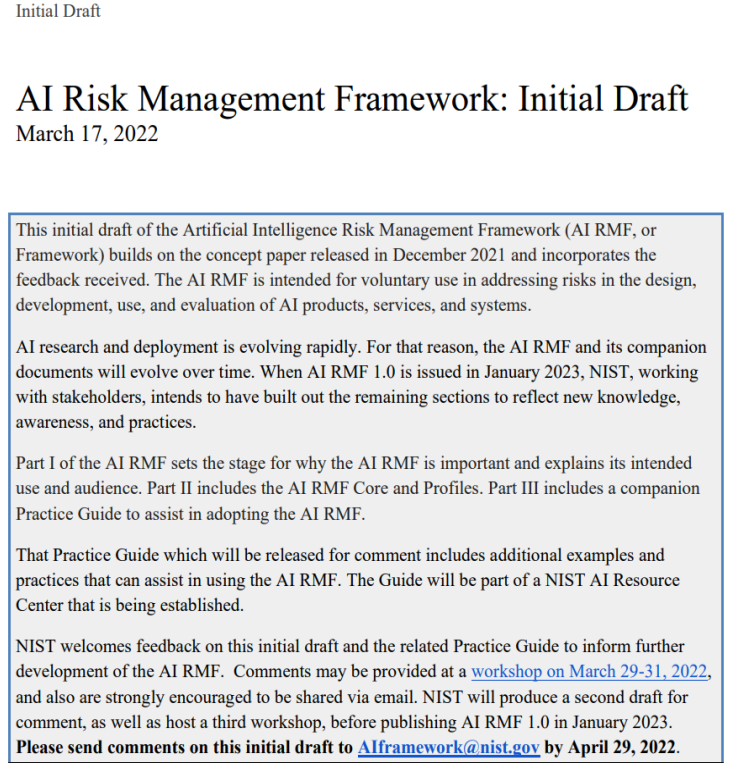
1. **AI Risk Management via Outcome Testing**
2. Weak Spot Identification
   - Error Analysis by Slicing
3. Reliability Evaluation
   - Conformal Prediction
4. Robustness and Stability to Distribution Drift
   - Noise-Perturbation Robustness
   - Worst-Sampler Resilience

AI Risks and Trustworthiness

NIST's latest release (03/17/2022) of AI Risk Management Framework:

> "**Accuracy** indicates the degree to which ML model is correctly capturing a relationship existing within the data. It is examined via standard ML metrics as well as assessment of underfit or overfit."
>
> "**Reliability** indicates whether a model consistently generates the same results, within the bounds of acceptable statistical error."
>
> "**Robustness** [measures] might range from sensitivity of a model's outputs to small changes in its inputs, but might also include error measurements on novel datasets."
>
> "**Resilience** or ML security [is to] withstand adversarial attacks, or more generally, unexpected changes in its environment or use."

These ML aspects can be measured by **outcome testing**.



Initial Draft

AI Risk Management Framework: Initial Draft
March 17, 2022

This initial draft of the Artificial Intelligence Risk Management Framework (AI RMF, or Framework) builds on the concept paper released in December 2021 and incorporates the feedback received. The AI RMF is intended for voluntary use in addressing risks in the design, development, use, and evaluation of AI products, services, and systems.

AI research and deployment is evolving rapidly. For that reason, the AI RMF and its companion documents will evolve over time. When AI RMF 1.0 is issued in January 2023, NIST, working with stakeholders, intends to have built out the remaining sections to reflect new knowledge, awareness, and practices.

Part I of the AI RMF sets the stage for why the AI RMF is important and explains its intended use and audience. Part II includes the AI RMF Core and Profiles. Part III includes a companion Practice Guide to assist in adopting the AI RMF.

That Practice Guide which will be released for comment includes additional examples and practices that can assist in using the AI RMF. The Guide will be part of a NIST AI Resource Center that is being established.

NIST welcomes feedback on this initial draft and the related Practice Guide to inform further development of the AI RMF. Comments may be provided at a workshop on March 29-31, 2022, and also are strongly encouraged to be shared via email. NIST will produce a second draft for comment, as well as host a third workshop, before publishing AI RMF 1.0 in January 2023. **Please send comments on this initial draft to AIframework@nist.gov by April 29, 2022.**

Outline

1. AI Risk Management via Outcome Testing
2. **Weak Spot Identification**
   - **Error Analysis by Slicing**
3. Reliability Evaluation
   - Conformal Prediction
4. Robustness and Stability to Distribution Drift
   - Noise-Perturbation Robustness
   - Worst-Sampler Resilience

Underfitting and Overfitting: Bias and Variance

Model Complexity and Errors[1]:

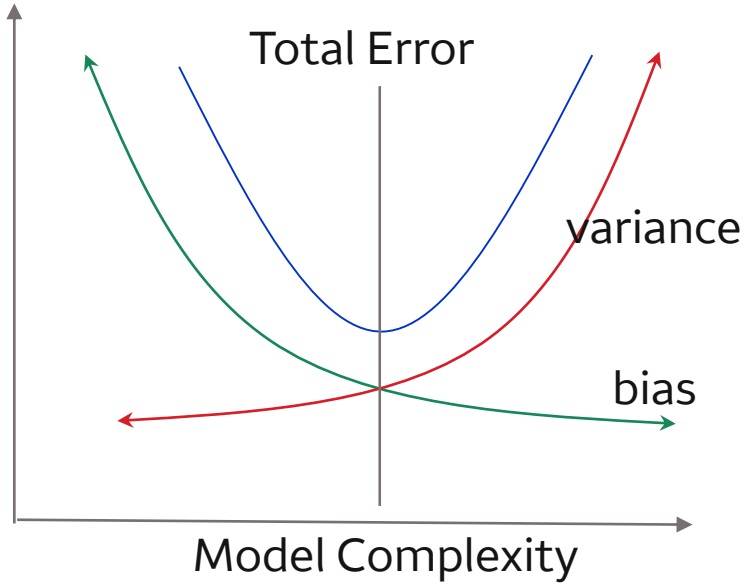
$$Err = err - n\,\sigma^2 + 2\sigma^2 \sum_{i=1}^{n} \frac{\partial f}{\partial y}$$

True error

Fitting error

Observation error

Degree of freedom



- Underfitting: Fitting error is used to evaluate model bias/underfit
- **Overfitting**:
  - The differences between "true" and "fitting" errors can be used to evaluate overfitting: the bigger the gap the more overfit
  - The degree of freedom can be used to evaluate overfitting: The larger the degree of freedom the more flexible (a.k.a. higher variance) and overfit the model is:
    - Perfect interpolation has the degree of freedom equal to the number of training data.

[1]Derived from Stein's Unbiased Risk Estimator

Overfitting Test

- Residual Error Test: Training-Testing Data Split Test
  - Testing data set exclusive of training data set is held out as a proxy for “true errors”
  - The “difference gap” between training error and testing error are indication of overfitting
  - For Machine Learning “slicing” the residual errors to identify regions where large gaps occurred are crucial to identify where the model is overfitting
- Degree of Freedom Test
  - The degree of freedom can be estimated by calculating the effect of model output changes by perturbing the data and compared with actual observation: $\Delta f/\Delta y$ where Δf are the change of model outputs and Δy are the change of true observational responses
  - “Slicing” $\Delta f/\Delta y$ to identify regions where large gaps occurred are crucial to identify where the model is overfitting
- Robustness Test
  - Using testing data set as a proxy for “true errors” may introduce “over-optimism” because samples come from the same set of training data especially when “observational noise” are changing during model use
  - Robustness testing (known as adversarial testing in NLP and computer vision) can be used to evaluate model error sensitivity by perturbing input features can to evaluate overfitting

Error Gaps and Degree of Freedoms

Model Complexity and Errors[1]:

Gap

$$Err - err = n\,\sigma^2 + 2\sigma^2 \sum_{i=1}^{n} \frac{\partial f}{\partial y}$$

H-hat: Change model output with respect to true response

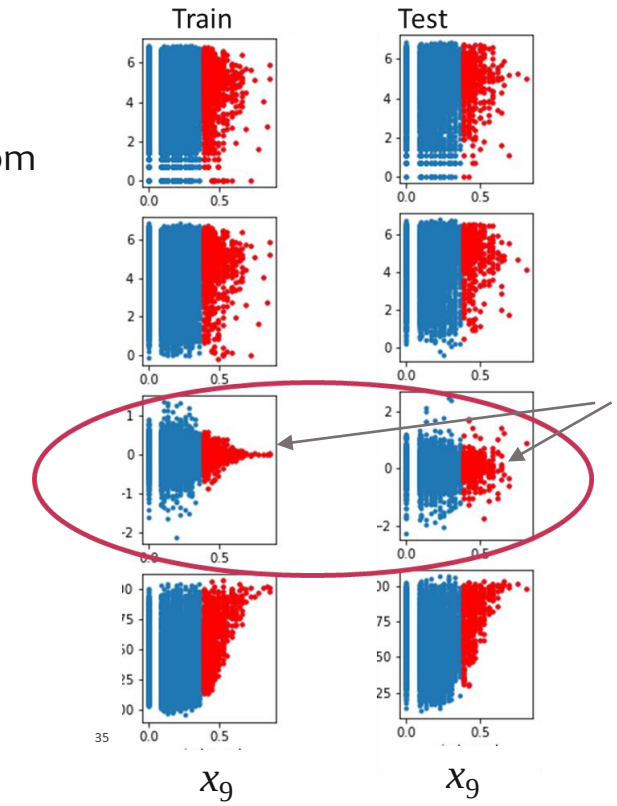
True error

Fitting error

Observation error

Degree of freedom

	train_mse	test_mse	gap	gdf	st_gdf
TreeS	0.320947	0.503853	0.182906	91.786928	90.74
TreeD	0.0	0.513952	0.513952	800.013266	803.352817
LinMod	0.48497	0.593525	0.108555	2.978045	2.90206
XGB2	1.462174	1.904146	0.441972	22.182974	21.226176
XGB5	0.569564	0.740736	0.171173	45.983647	46.717387
XGB10	0.297229	0.417862	0.120634	66.284872	67.104025
XGB25	0.222626	0.339883	0.117257	94.852878	95.06242
XGB50	0.195848	0.323432	0.127584	135.02614	134.419782
XGB100	0.162803	0.319827	0.157024	204.453289	205.796206
XGB200	0.123886	0.332779	0.208894	302.71768	305.651157
XGB400	0.079381	0.342018	0.262637	431.19859	435.300993
XGB800	0.035311	0.362726	0.327415	575.382511	577.760143



WeakSpot – Error Analysis by Slicing

Spot weak regions of fitted ML models

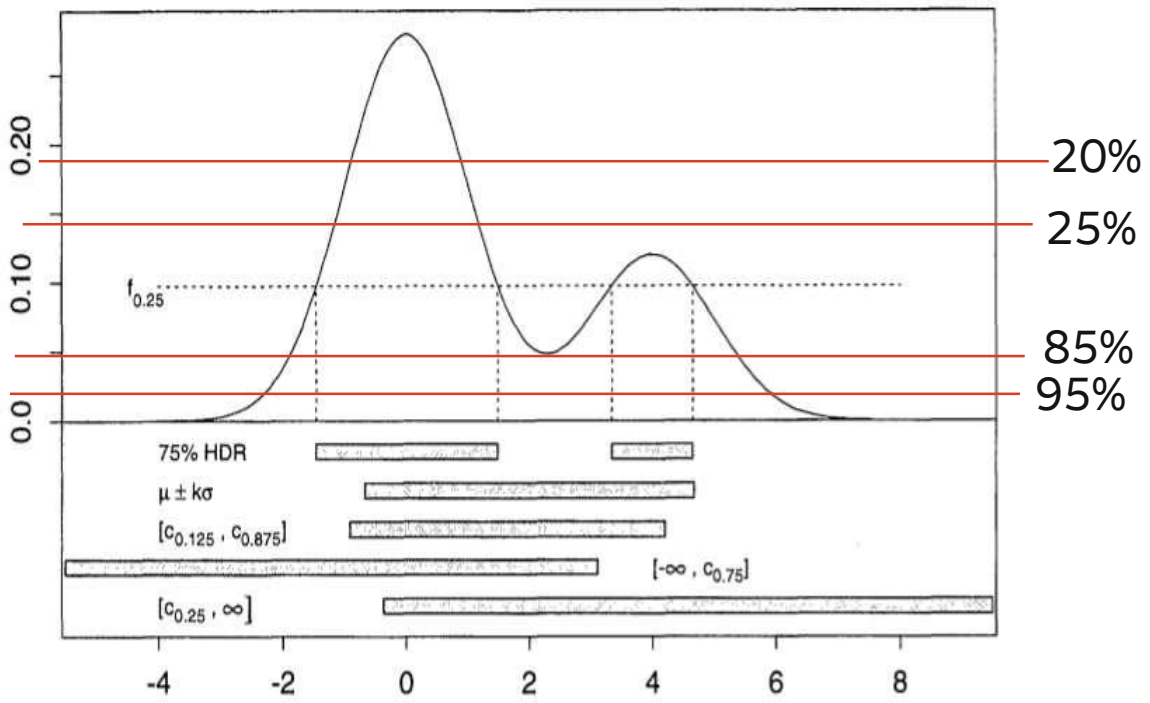
Given a fitted ML model, we have the following 3 methods to spot weak regions in 1 or 2 slicing features.

- **HDR-based slicing:** use Highest Prior Density (HPD) methods to find univariate data slices with low accuracy.
- **Tree-based slicing:** use decision trees to find univariate or bivariate data slices with low accuracy.
- **Ensemble tree-based slicing:** use ensemble trees (e.g., XGBoost) to find univariate or bivariate data slices with low accuracy.

WeakSpot – HDR Slicing

Highest Prior Density (HDR) Slicing

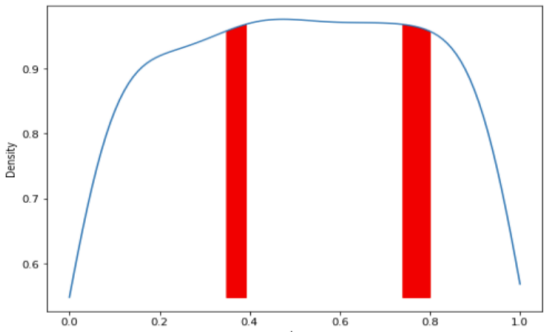
- HDR is method for computing and graphing highest density regions, proposed by Hyndman (1996).
- Estimate the probability density function of the given 1D slicing feature.
- Find the [95%, 90%, …, 5%] highest density regions, i.e., the list of region boundaries.
- Use the region boundaries to form many bins, and evaluate the performance of each bin segment, and report the ones with low accuracy.



WeakSpot – HDR Slicing 1D and 2D

1D HDR Slicing

```
from piml.diagnoser import WeakSpot
weakness = WeakSpot(estimator, feature_names=feature_names,
                    threshold_ratio=2.5, method='hdr', n_estimators=50)
weakness.fit(x_train, y_train, x_test, y_test)
weakness.show('hr')
```

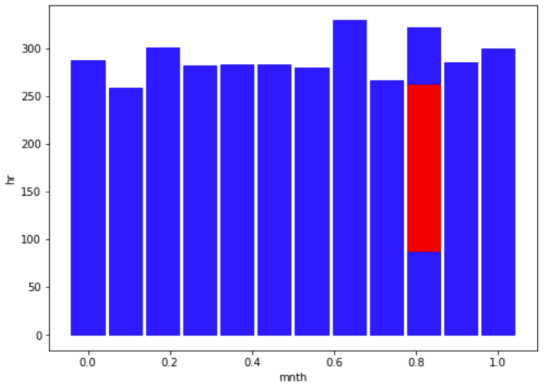


- The blue curve is the probability density function.
- The weak regions are marked by red, i.e., the low and up quantiles of the region boundaries.

	hr_lb	hr_ub	#Test	#Train	MSE_test	MSE_train	Gap
0	0.347826	0.391050	135	592	0.016155	0.013813	0.002342
1	0.739130	0.799255	147	581	0.011143	0.008801	0.002341

2D HDR Slicing

```
from piml.diagnoser import WeakSpot
weakness = WeakSpot(estimator, feature_names=feature_names,
                    threshold_ratio=2.5, method='hdr', n_estimators=50)
weakness.fit(x_train, y_train, x_test, y_test)
weakness.show('hr', 'mnth')
```



- Binning based on the second feature. Within each bin, run 1D HDR slicing for the first feature.
- The weak regions are marked in red, i.e., the low and up quantiles of the region boundaries.

	hr_lb	hr_ub	mnth_lb	mnth_ub	#Test	#Train	MSE_test	MSE_train	Gap
0	0.304348	0.86244	0.772727	0.863636	174	553	0.010326	0.007114	0.003213

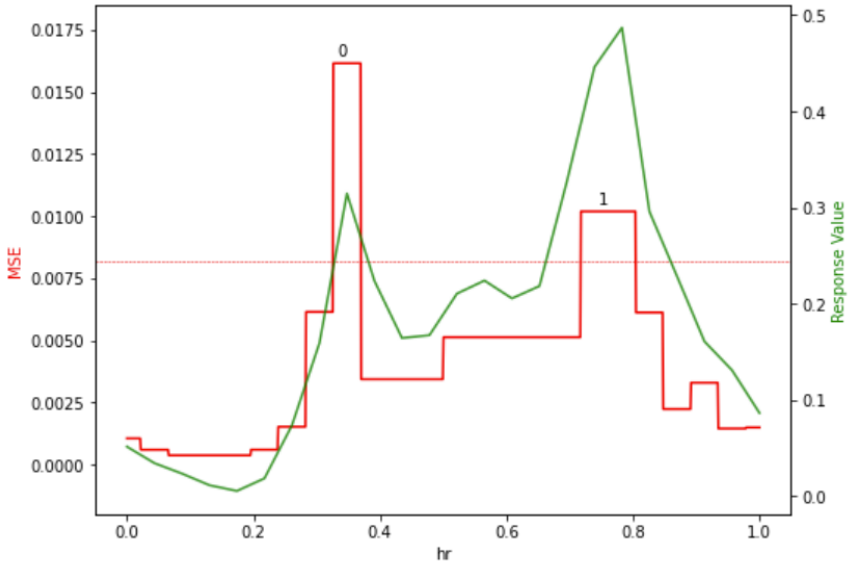
WeakSpot – Tree Slicing

Tree Slicing

- Given a fitted ML model, specify 1 or 2 slicing features.
- Evaluate the loss for each train test sample; e.g., MSE for regression and ACC for classification.
- Fit a Decision Tree with the slicing features as input and the sample-wise loss values as response.
- Spot the weak regions by the leaf nodes with a) Response values (i.e., loss) above certain threshold; and b) Number of samples above certain amount.

```
from piml.diagnoser import WeakSpot

weakness = WeakSpot(estimator, feature_names=feature_names,
                    threshold_ratio=2, method='tree', max_depth=5, n_estimators=50)
weakness.fit(x_train, y_train, x_test, y_test)
weakness.show('hr', plot_response=True)
```



	hr_lb	hr_ub	#Test	#Train	MSE_test	MSE_train	Gap
0	0.326087	0.369565	135	592	0.016155	0.013813	0.002342
1	0.717391	0.804348	290	1168	0.010193	0.009934	0.000260

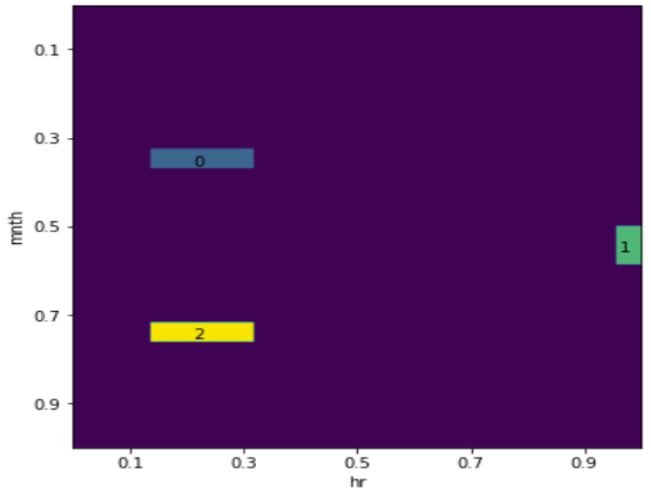
WeakSpot – Ensemble Tree Slicing

Ensemble Tree Slicing

- Specify 1 or 2 slicing features.
- Evaluate the loss for each train test sample; e.g., MSE for regression and ACC for classification.
- Fit an ensemble tree (e.g., XGBoost) with the slicing features as input and the sample-wise loss values as response.
- Get all unique splitting points found by the ensemble tree. Calculate the average loss of each region formed by the splitting points.
- Spot the weak regions a) Response values (i.e., loss) above certain threshold, and b) Number of samples above certain amount.

```
from piml.diagnoser import WeakSpot

weakness = WeakSpot(estimator, feature_names=feature_names,
                    threshold_ratio=2, method='xgb', n_estimators=50)
weakness.fit(x_train, y_train, x_test, y_test)
weakness.show('hr', 'mnth')
```



	hr_lb	hr_ub	mnth_lb	mnth_ub	#Test	#Train	MSE_test	MSE_train	Gap
0	0.326087	0.369565	0.136364	0.318182	20	102	0.013600	0.010589	0.003011
1	0.500000	0.586957	0.954545	inf	26	98	0.009809	0.007206	0.002603
2	0.717391	0.760870	0.136364	0.318182	26	96	0.009163	0.008951	0.000212

Outline

1. AI Risk Management via Outcome Testing
2. Weak Spot Identification
   - Error Analysis by Slicing
3. **Reliability Evaluation**
   - **Conformal Prediction**
4. Robustness and Stability to Distribution Drift
   - Noise-Perturbation Robustness
   - Worst-Sampler Resilience

Conformal Prediction: An Introduction

- A distribution-free UQ framework in machine learning pioneered by Vladimir Vovk since 1990s; see *Vovk, et al., 2005*, or alrw.net

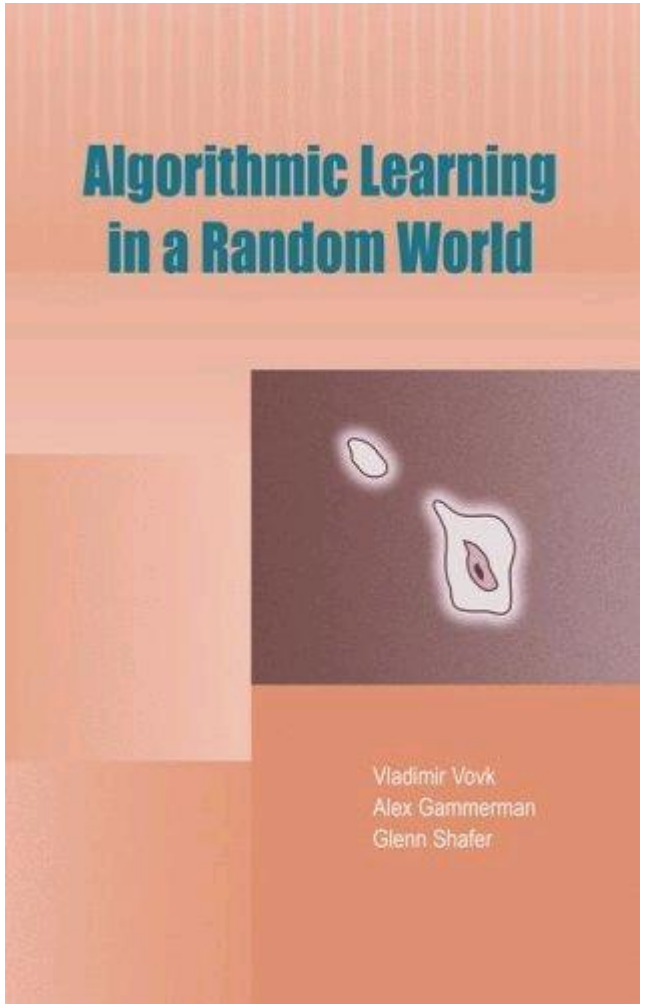
- When $W_1, \ldots, W_n; W_{n+1}$ are exchangeable (weaker than i.i.d.)

$$\mathbb{P}(\text{rank}(W_{n+1}) \leq \lceil (n+1)(1-\alpha) \rceil) = \frac{\lceil (n+1)(1-\alpha) \rceil}{n}, \qquad \alpha \in [0,1]$$

- Given a predictive model $\hat{f}(\boldsymbol{x})$, it is desirable to identify a conformal score measuring the prediction uncertainty

$$S(\boldsymbol{x}, y, \hat{f}) \in \mathbb{R}$$

exchangeable among samples (for calibration and testing data at least)



Split Conformal Prediction

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

1. Calculate the score $S_i = S(\boldsymbol{x}, y, \hat{f})$ for each sample in $\mathcal{X}_{\text{calib}}$;
2. Compute the calibrated score quantile

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

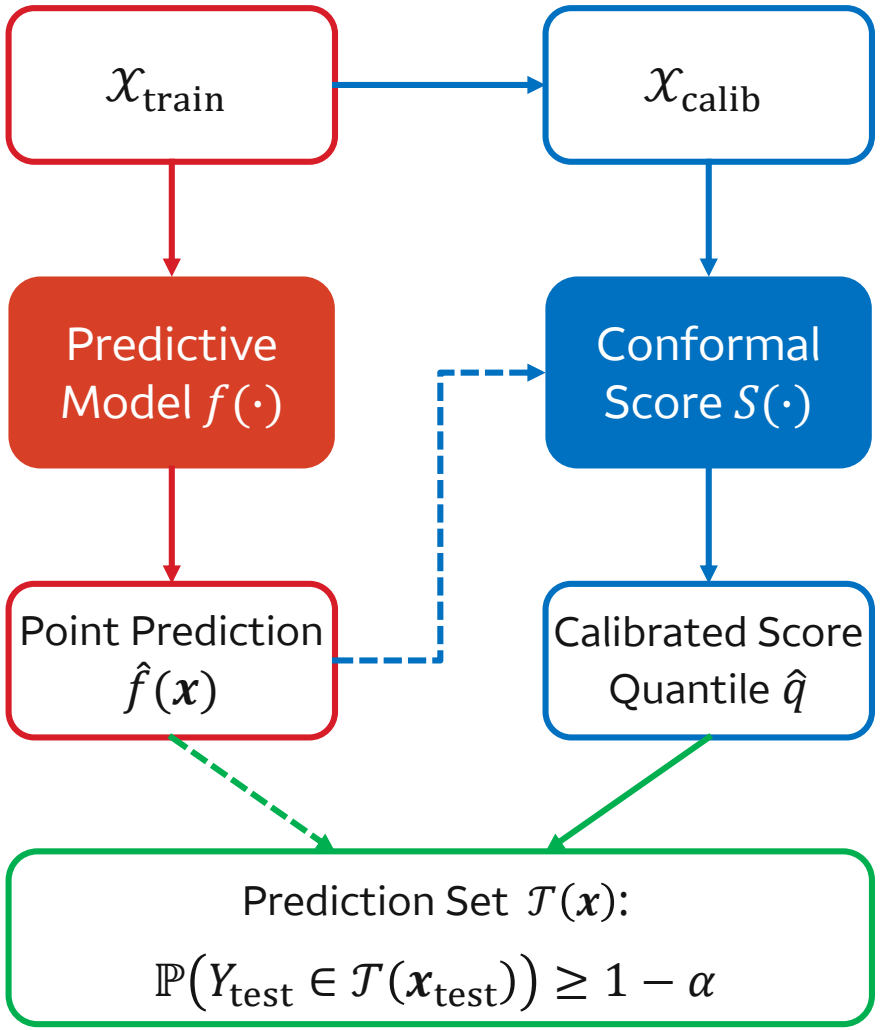
3. Construct the prediction set for the test sample $\boldsymbol{x}_{\text{test}}$ by

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

Under the exchangeability condition of conformal scores, we have that

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

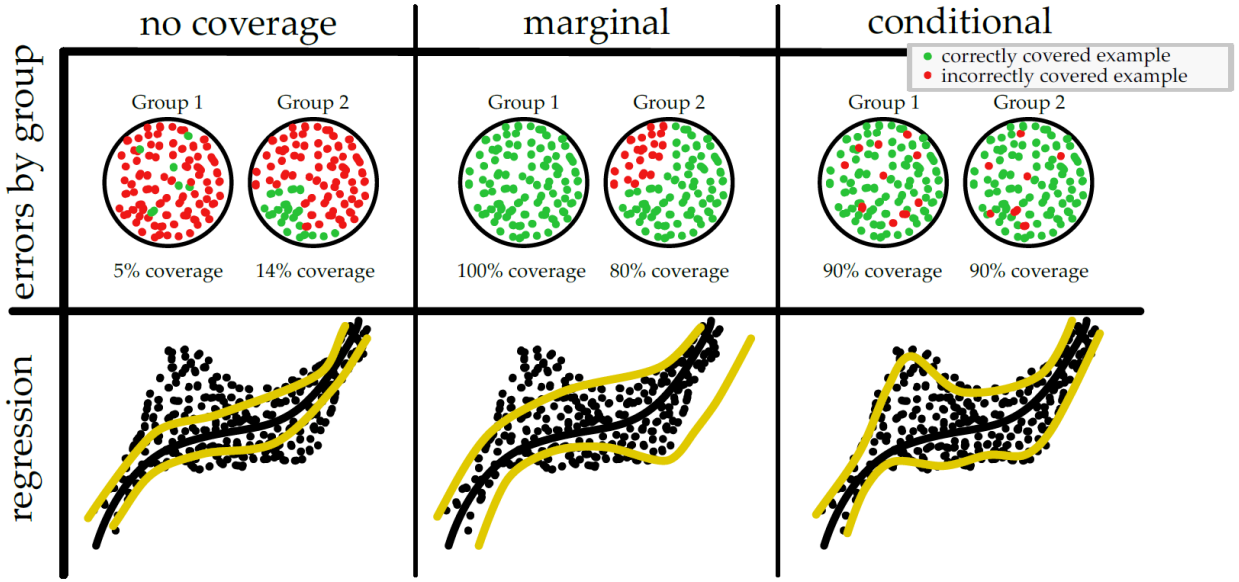
This provides the prediction bounds with α-level acceptable error.



How to Evaluate Conformal Prediction?

There are three aspects to compare the goodness of conformal prediction by different methods:

1. Correct Coverage with statistical guarantee: to be checked with testing data
2. Set size or bandwidth of prediction confidence: smaller/narrower is better
3. Heteroscedastic adaptiveness (approx. conditional coverage): larger bands for harder inputs



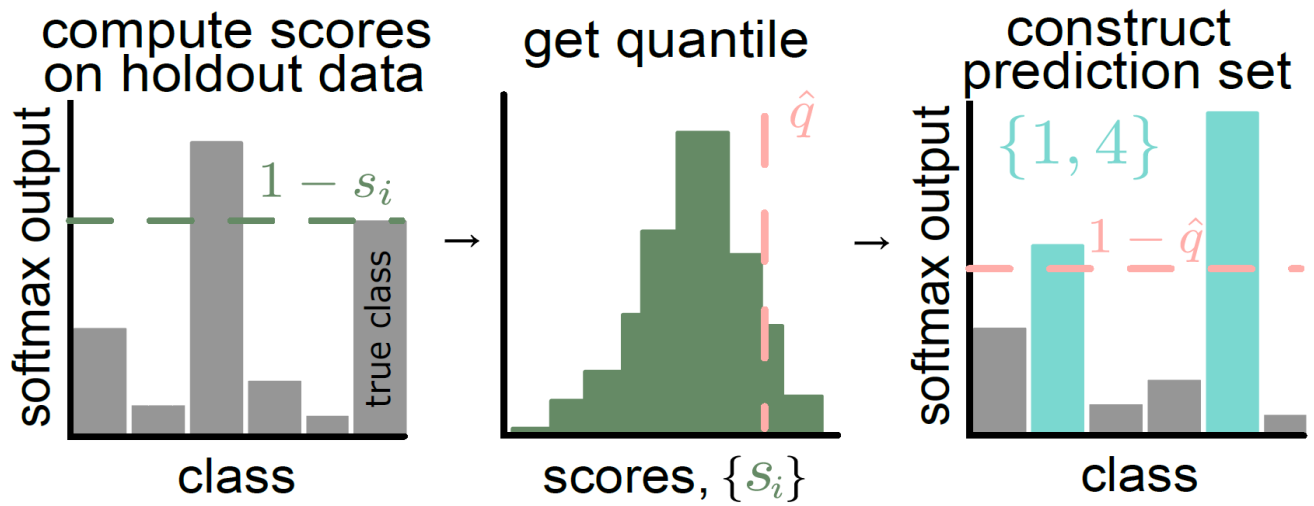
Source: Angelopoulos & Bates, S. (2021). A gentle introduction to conformal prediction and distribution-free uncertainty quantification.

Conformal Prediction for Classification

For a classifier $\hat{f}(\boldsymbol{x})$ pre-trained with softmax loss and prob output, one may use the conformal score

$$S\left(\boldsymbol{x}, y, \hat{f}\right) = 1 - \hat{f}(\boldsymbol{x})_y \quad \text{(one minus the softmax output of the true class)}$$

then conformal prediction works as follows:



Source: Angelopoulos & Bates, S. (2021).

This can be easily extended to "Adaptive Prediction Sets" with a modified score; see Romano, et al. (2020).

Conformal Prediction for Regression

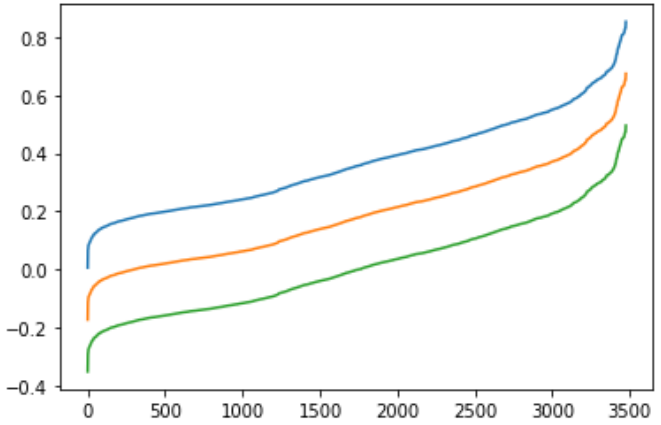
- For regression model $\hat{f}(\boldsymbol{x})$, a vanilla choice of conformal score is the residual

$$S(\boldsymbol{x}, y, \hat{f}) = |y - \hat{f}(\boldsymbol{x})|,$$

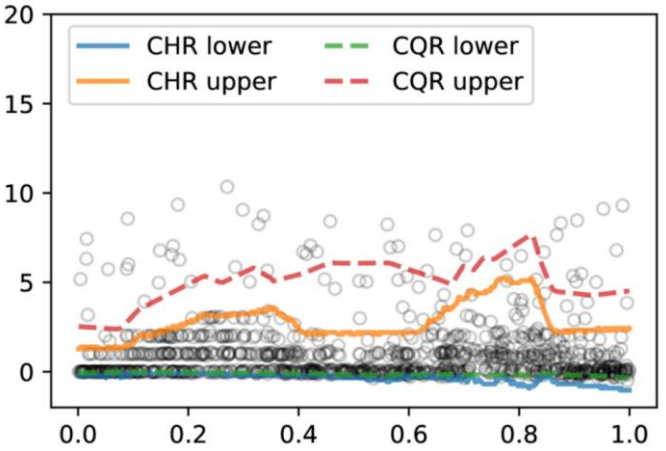
  then use $\mathcal{X}_{\text{calib}}$ to compute the calibrated score quantile $\hat{q}$, and construct

$$\mathcal{T}(\boldsymbol{x}_{\text{test}}) = \{y: \ |y - \hat{f}(\boldsymbol{x}_{\text{test}})| \leq \hat{q}\}.$$

  This naïve method generates the prediction confidence intervals with constant bandwidth for every test sample.



- **Heteroscedastic methods**: Conditional variance normalization (Lei, et al., 2014, 2018), Conformalized quantile regression (Romano, et al., 2019), Conditional histogram regression (Sesia and Romano, 2021).



- Note that the objective of these methods is to **make inference of response Y directly from data**, and they often rely on auxiliary models to predict conditional variance, MAD, quantiles or distributions.
- But we aim at evaluating reliability of a pre-trained AI/ML model.

Conformalized Quantile Regression (CQR)

To use the CQR method by Romano, et al. (2019) for a pre-trained regression model $\hat{f}(\boldsymbol{x})$:

1. **Retrain** $f(\cdot)$ of the same hyperparameter configuration with the quantile loss, in order to predict the quantiles $\left[\widehat{f'}_{\alpha/2}(\boldsymbol{x}), \widehat{f'}_{1-\alpha/2}(\boldsymbol{x})\right]$;

2. Define the conformal score to be the projective distance from y to the predicted quantile interval:

$$S\left(\boldsymbol{x}, y, \widehat{f'}\right) = \max\left\{\widehat{f'}_{\alpha/2}(\boldsymbol{x}) - y, \qquad y - \widehat{f'}_{1-\alpha/2}(\boldsymbol{x})\right\}$$

3. Compute the calibrated scores for $\mathcal{X}_{\text{calib}}$, then calculate $\hat{q} = \text{Quantile}\left(\{S_1, \ldots, S_n\}; \frac{\lceil (n+1)(1-\alpha) \rceil}{n}\right)$;

4. Construct the prediction interval for the test sample $\boldsymbol{x}_{\text{test}}$ by

$$\mathcal{T}(\boldsymbol{x}_{\text{test}}) = \left[\hat{f'}_{\alpha/2}(\boldsymbol{x}_{\text{test}}) - \hat{q}, \qquad \widehat{f'}_{1-\alpha/2}(\boldsymbol{x}_{\text{test}}) + \hat{q}\right].$$

Note that in such CQR procedure, the original pre-trained model $\hat{f}(\boldsymbol{x})$ does not play any role. It is replaced by the $\widehat{f'}(\boldsymbol{x})$ (upon retraining) to predict the conditional quantiles.

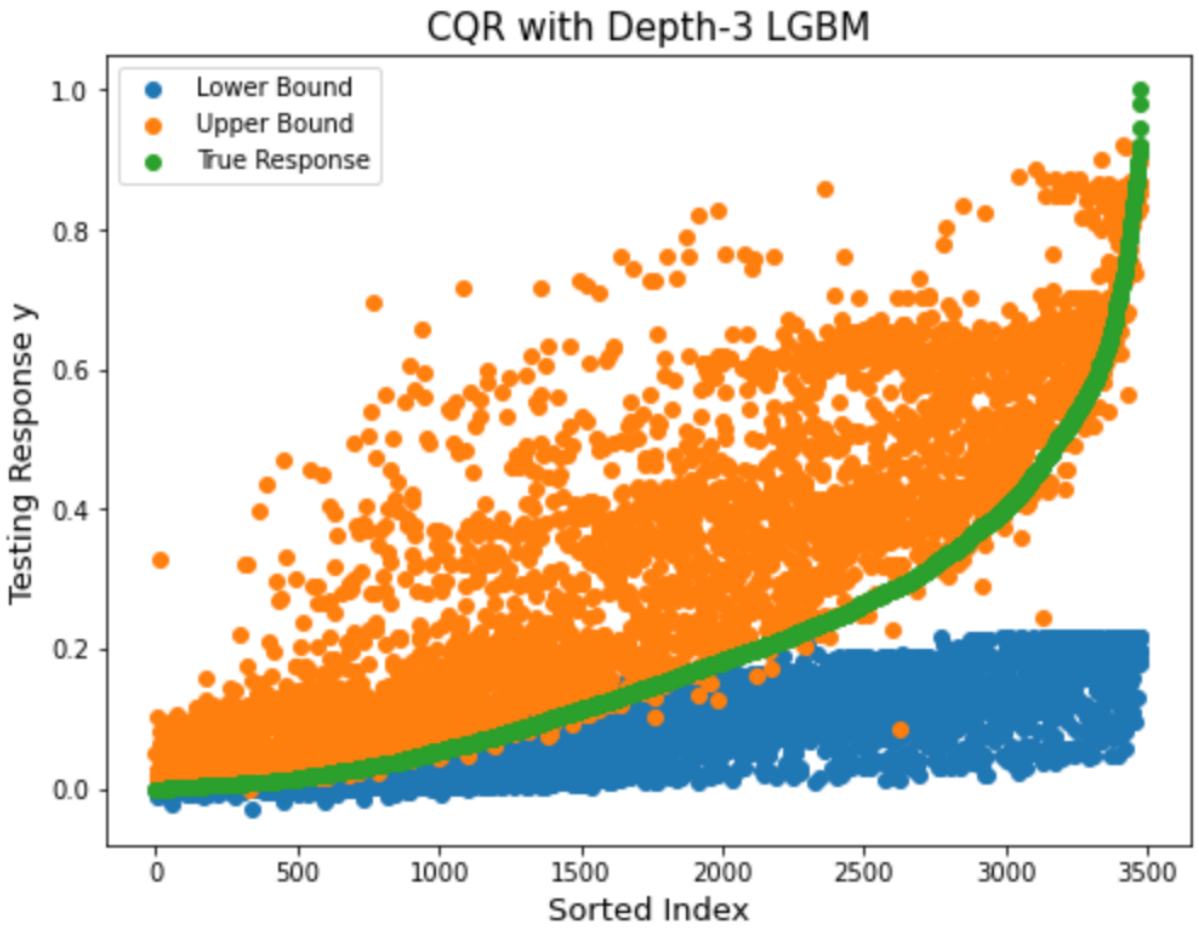
Conformalized Quantile Regression (CQR)

Experimental Setting:

- Bikesharing Dataset w/ MinMaxScaler
- Train : Calib : Test = 6 : 2 : 2
- $\alpha = 0.1$
- LightGBM with depth 3

Results: for an arbitrary random seed

- Correct coverage: 0.9054
- Average bandwidth: 0.2518



New Method: Residual-based CQR

This new method is designed to directly evaluate reliability of any pre-trained regression model $\hat{f}(\boldsymbol{x})$:

1. **Obtain** residuals $y_i - \hat{f}(\boldsymbol{x}_i)$ for each $i \in \mathcal{X}_{\text{train}}$, train a high-performance quantile regressor (LightGBM with quantile loss) to predict the residual quantiles $\left[\hat{g}_{\alpha/2}(\boldsymbol{x}), \hat{g}_{1-\alpha/2}(\boldsymbol{x})\right]$;

2. Define the conformal score $S\left(\boldsymbol{x}, y, \hat{f}\right) = \max\left\{\hat{g}_{\alpha/2}(\boldsymbol{x}) - y + \hat{f}(\boldsymbol{x}),\ y - \hat{f}(\boldsymbol{x}) - \hat{g}_{1-\alpha/2}(\boldsymbol{x})\right\}$

3. Compute the calibrated scores for $\mathcal{X}_{\text{calib}}$, then calculate $\hat{q} = \text{Quantile}\left(\{S_1, \ldots, S_n\}; \frac{\lceil (n+1)(1-\alpha) \rceil}{n}\right)$;

4. Construct the prediction interval for the test sample $\boldsymbol{x}_{\text{test}}$ by

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

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

New Method: Residual-based CQR

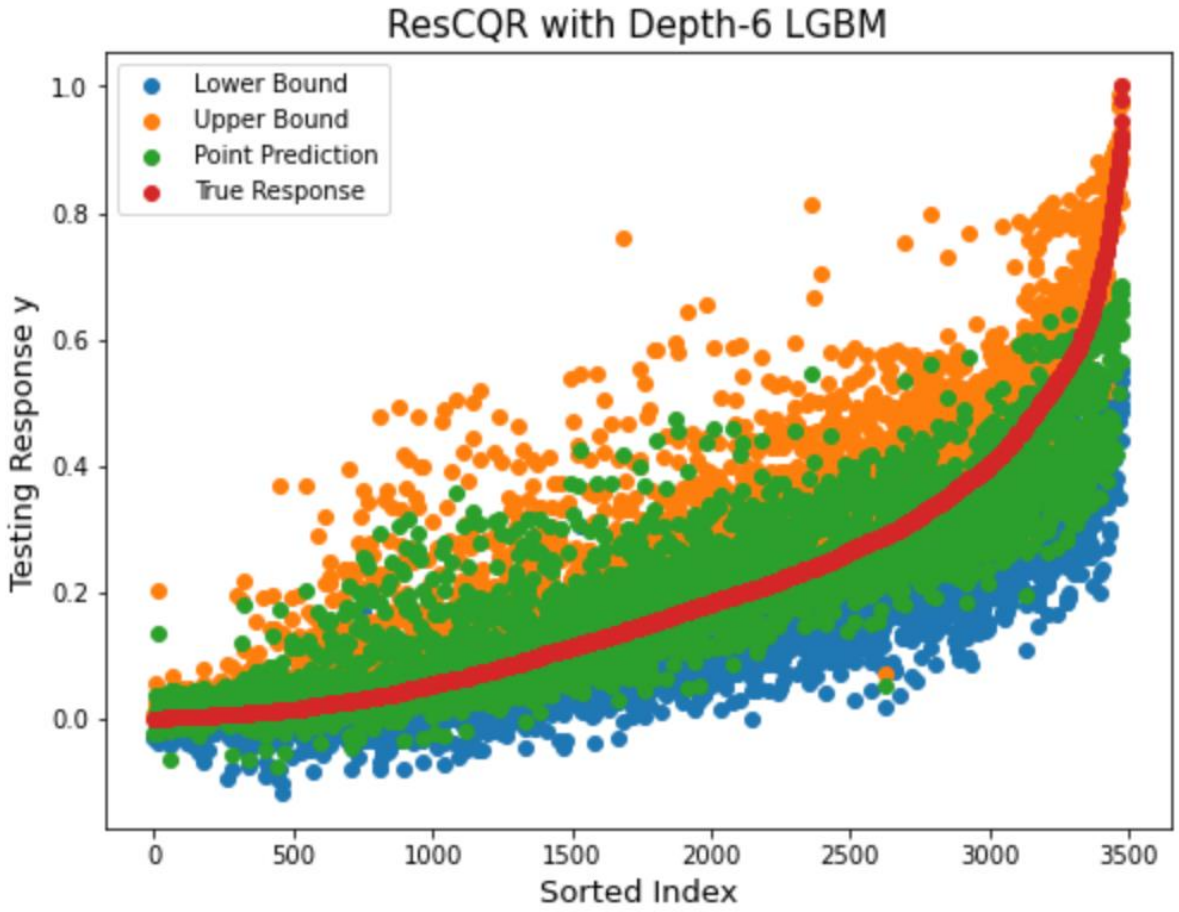
Experimental Setting:

- Bikesharing Dataset w/ MinMaxScaler
- Train : Calib : Test = 6 : 2 : 2
- $\alpha = 0.1$
- LightGBM with depth 3

ResCQR: try LightGBM depth 6

Results: for the random seed

- Correct coverage: 0.9059
- Average bandwidth: 0.1556



Outline

1. AI Risk Management via Outcome Testing
2. Weak Spot Identification
   - Error Analysis by Slicing
3. Reliability Evaluation
   - Conformal Prediction
4. **Robustness and Stability to Distribution Drift**
   - **Noise-Perturbation Robustness**
   - **Worst-Sampler Resilience**

Robustness and Distribution Shift

- Assessment whether a model will perform well when deployed which often differs from the environment in the training data.
- Assessment of model performance under sub-population changes
- How do we do it traditionally?
  - Cross validation or bootstrap – Data still from the same distribution
  - Evaluate models with multiple datasets independently collected – This may not be available
- Two situations will be discussed:
  - Noise corruption to input data
  - Covariate Distribution drift

Distribution and Covariate Shift

- Problem: Distribution of data differ between training and model usage

$$P_{train}(X, Y) \neq P_{use}(X, Y)$$

$$P(X, Y) = P(Y|X)\, P(X)$$

- Concept Drift: $P_{train}(Y|X) \neq P_{use}(Y|X)$

- Covariate Drift: $P_{train}(Y|X) \neq P_{use}(Y|X)$ but $P_{train}(X) \neq P_{use}(X)$
  - Model remain the same
  - Distribution of input data shifts between the training environment and during model usage

Mutable and Immutable Covariates

- Covariates are partitioned into $X = \{X_i, X_m\}$
- Immutable Covariates, X_i
  – Variables that remain unchanged $\rightarrow$ marginal distribution will be fixed
- Mutable Covariates, X_m
  – Variables that experience drift$\rightarrow$ marginal distribution will vary

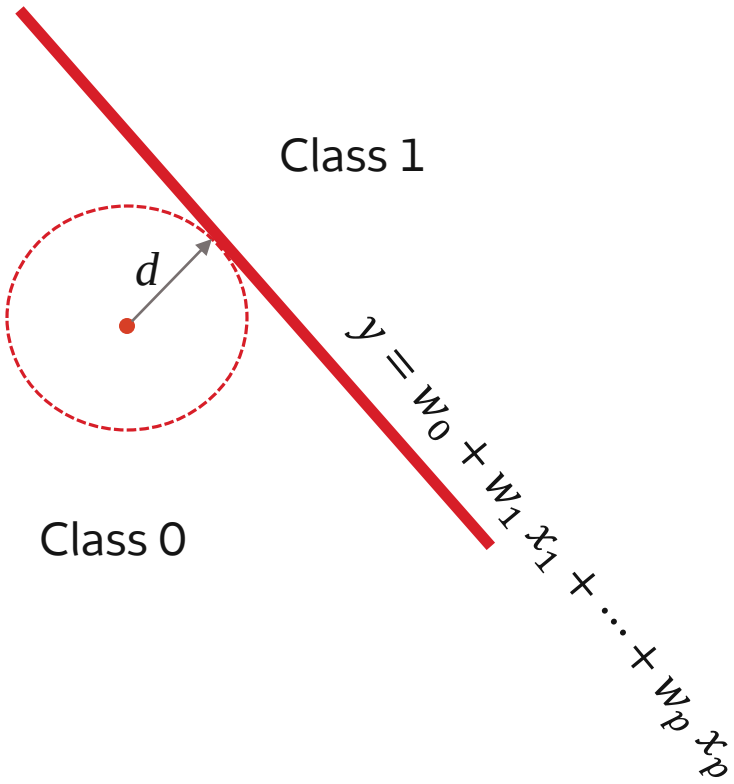
- $P(X, Y) = P(Y|X_m, X_i)\ P(X_m|X_i)\ P(X_i)$
- Model Performance are evaluated by replacing $P_{train}(X_m|X_i) \leftarrow P_{use}(X_m|X_i)$

Robustness against Noise

- Distance to decision boundary, d:

$$d = \frac{|w_0 + w_1 x_1 + \cdots + w_p x_p|}{\|\mathbf{w}\|}$$

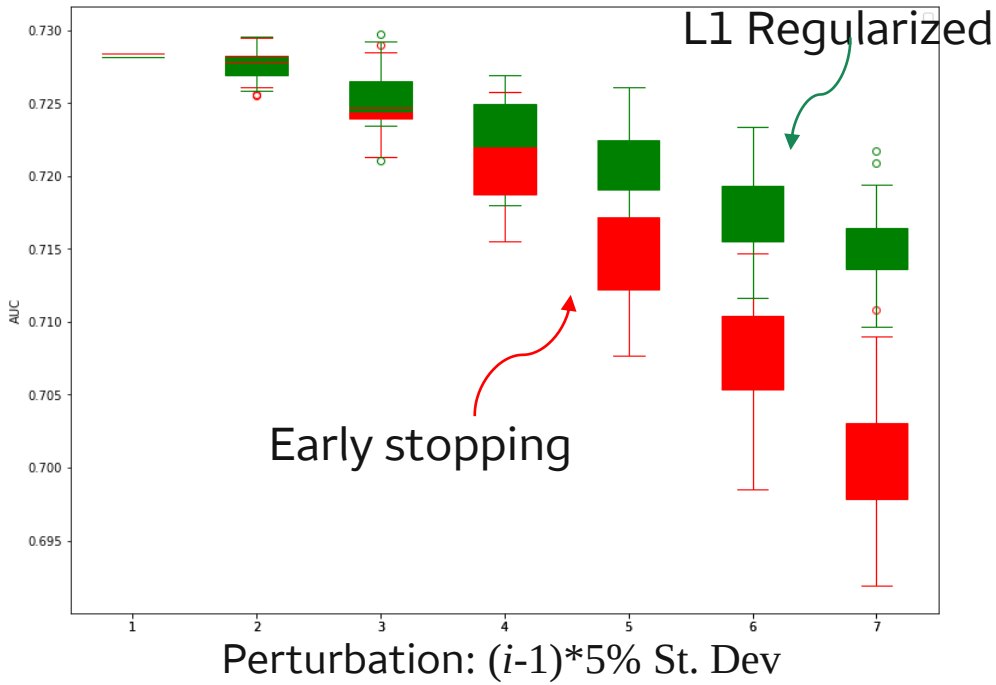
- Perturbation (noise) size d will flip the output of the model $\rightarrow$ wrong classification
  - Adversarial robustness
    - Circumvent model decision
    - Model security
  - Counter factual robustness
- The denominator suggests that minimizing $\|\mathbf{w}\|$ is enlarging distance d, thus improving model robustness
  - Regularized (i.e., simpler) models are more robust less performance degradation when inputs are corrupted



Robustness Test by Noise Perturbation

- Assumption
  - $g: P_{train}(X_m) \rightarrow P_{use}(X_m)$
  - g preserves label samples
- Choose mutable set variables $X_m \subset X$
  1. Apply Random perturbation to X_m in the testing dataset
  2. Calculate Performance
  3. Repeat 1-2 for k times
  4. Box-plot 3

Taiwan Credit Data Example:
Complex vs. Simpler Model Performance



Perturbation: (i-1)*5% St. Dev

Covariate Distribution Shift

- **Sample Reweighting**

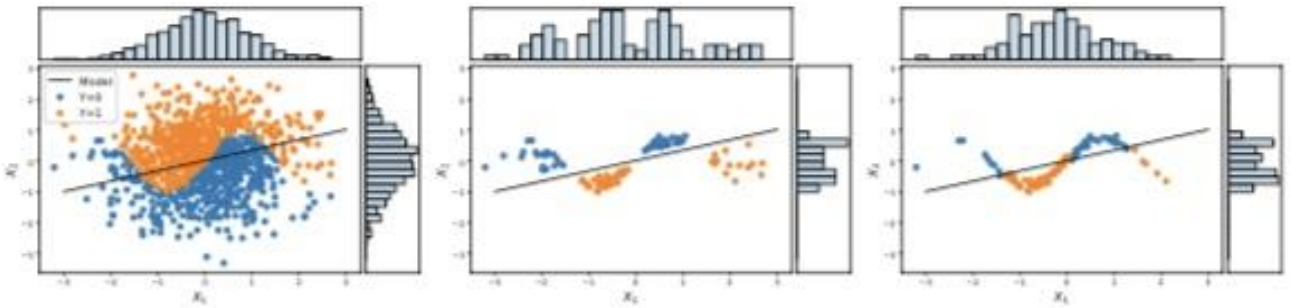
$$E\left[l_{use}\left(x, y, \hat{f}(x;\boldsymbol{\theta})\right)\right] = \frac{1}{n}\sum_{i=1}^{n}\frac{P_{use}(x_i)}{P_{train}(x_i)}\, l_{train}\left(x_i, y_i, \hat{f}(x_i;\boldsymbol{\theta})\right)$$

  - Require assumptions of $P_{use}(x_i)$ during model use or simulation
  - Example: $P_{use}(x_i) = \frac{w(x_i)}{\sum_{j=1}^{n} w(x_j)}$ where $w(x_i) = exp(\boldsymbol{\beta}^{\mathrm{T}} x_i)$

- **Worst-Case Sampling**

$$E_{\alpha-worst}\left[l_{use}\left(x, y, \hat{f}(x;\boldsymbol{\theta})\right)\right] = E\left[h(x)\; l_{train}\left(x, y, \hat{f}(x;\boldsymbol{\theta})\right)\right]$$

  where $h(x)$ is a function that selects datapoints to be in or out of α-worse case



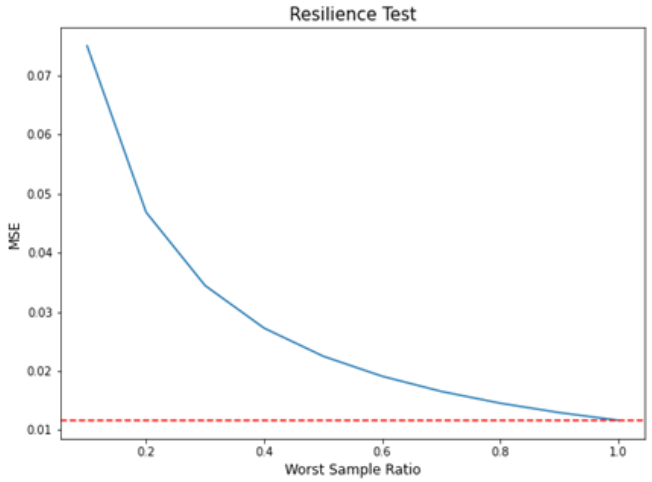
(a) Data distribution (b) Shift in $P(X_1, X_2)$ (c) Shift in $P(X_2|X_1)$

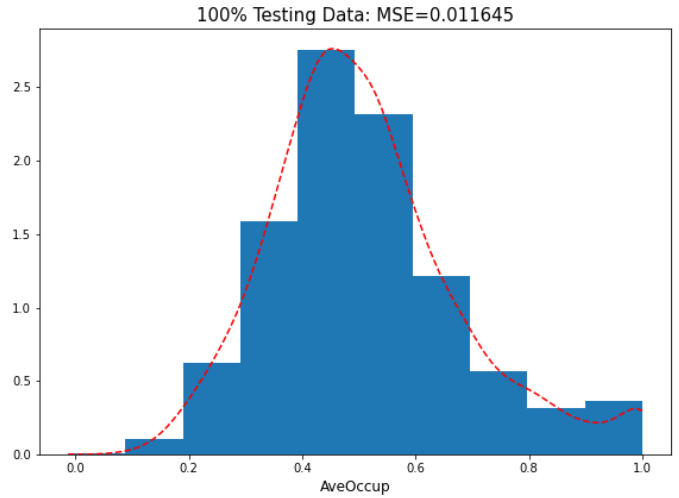
Subbaswamy et. al: http://proceedings.mlr.press/v130/subbaswamy21a/subbaswamy21a.pdf

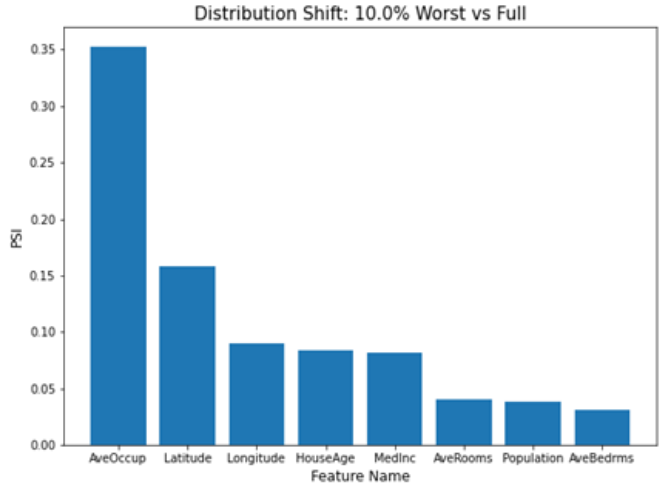
Resilience Test by Worst Subsampling

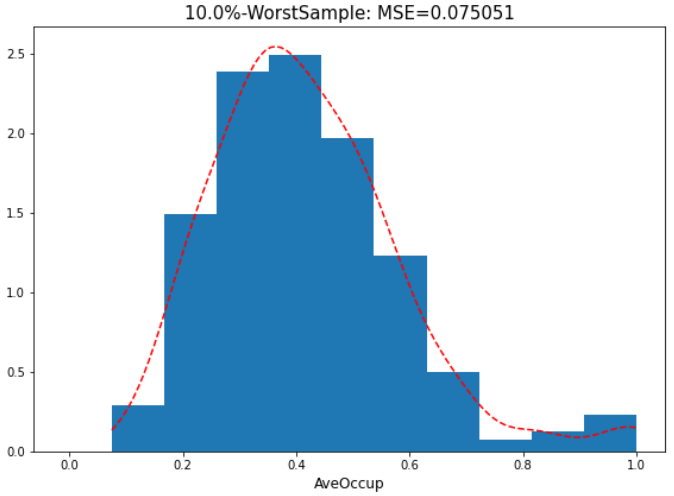
- Anticipating model performance degradation under worst case shift scenario
- Identify vulnerability due to covariate shift
- Rank covariates using distribution shift measure such as Population Stability Index (PSI)

$$PSI = \left(P_{use}(X_m) - P_{test}(X_m)\right) * ln\left(P_{use}(X_m)/P_{test}(X_m)\right)$$









WELLS FARGO

Thank you

Agus Sudjianto, Ph.D.
EVP, Head of Corporate Model Risk (CMoR)

Vijay Nair, Ph.D.
Head of Advanced Technologies for Modeling (AToM), CMoR