Machine Learning: Conceptual Soundness

Explainability and Interpretability

Vijay Nair

Advanced Technologies for Modeling

Corporate Model Risk

Biographical Sketch: Vijay

- R&D group in Corporate Model Risk within in Wells Fargo
- **Advanced Technologies for Modeling (AToM)**
  - **Statistical Modeling and Machine Learning** (SMMaL) → Tabular Data; VIPER-COM
  - **Machine Learning and Validation Engineering** (MLiVE) → Pi-ML and Validation of Demand
  - **Artiifical Intelligence and Automation** (AIA) → NLP and conversational AI; VIPER-NLP
  - **Market Modeling Technologies** (MMT) → Capital Markets; MLTD
  - **CoPRA** →CMoR computing platform
- 1993-2016: University of Michigan
  - Donald A. Darling professor of statistics and professor of industrial & operations engineering
  - Distinguished data scientist at the University of Michigan, Ann Arbor
  - Chair, Statistics Department for 12 years
- 1978-1993: Research Scientist at Bell Labs, New Jersey
- PhD (Statistics) – University of California, Berkeley

Outline

- ❑ Machine Learning and Algorithms: Overview
- ❑ PiML: Python Interpretable Machine Learning Toolkit
- ✓ ML Model Risk and Validation
- ✓ Conceptual Soundness: Explainability and Interpretability
  - Post hoc methods
  - Inherently Interpretable Models

Machine Learning and Algorithms: Overview

Data-centric areas:

- **Credit Risk:** Predicting **losses from loans**
- **Credit Decisions:** credit scoring, marketing, collections, …
- **Revenue and Transactions:** Interest, servicing fees, deposits, withdrawals, electronic payments, etc.
- **Financial Crimes:** Fraud detection, Money laundering
- **Fair Lending**
- **Compliance and staffing**

- **NLP, Text and speech**: Conversations, complaints, emails, voice messages, chat-bots for assisting customers and employees

Statistical and econometric techniques

- Dimension reduction; clustering, anomaly detection
- Parametric modelling for regression and classification
- Semi- and non-parametric regression models
- Regularization: Lasso, ridge, …
- Survival analysis; Time series forecasting

ML/AI techniques

- Auto-encoders → Dimension reduction
- Isolation Forest → Anomaly detection
- **Supervised ML**: Support vector machines; **Random forests; Gradient boosting; Neural networks** (FF and Deep NNs)
- **Natural language processing** of Text Data → Deep NNs
- Conversational AI: Chatbots, …

Computing Environment

- Python, C++, Java, R, SAS, and R
- Open Source Libraries, Tensorflow, PyTorch
- CPU and GPU Clusters, Cloud

Natural Language Processing (NLP)

- Methods, algorithms, and systems for **analyzing "human language" data** (text, speech, conversations)
  - **Very challenging** …
- **Interdisciplinary area** that combined computer science, statistics, optimization, AI, linguistics, logic …
  - Earlier version → computational linguistics, speech recognition, …
- **Evolution:**
  - Rule-based, statistical …now **largely driven by deep neural networks**
- **Diverse applications**
  - Jeopardy and IBM Watson

Text Summarization

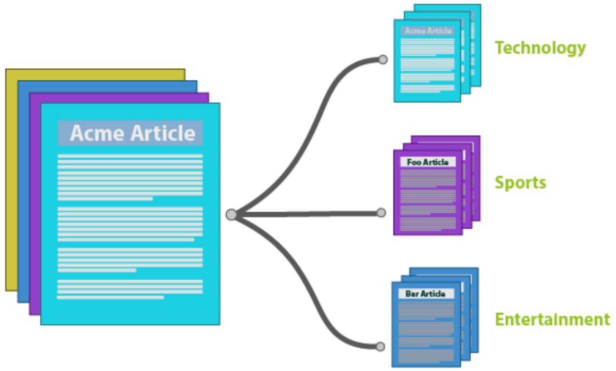
Machine Translation

Chatbots

- Alexa and Siri-like
- Conversational AI

Natural Language Generation

Text classification



Sentiment Analysis

Machine Learning and Artificial Intelligence
(wiki and other sources)

- **Machine Learning:**
  - Term coined in 1959 by **Arthur Samuel** (IBM)
  - study and construction of algorithms that can learn from data, summarize features, recognize patterns, make predictions, and take actions …
  - A key pathway to AI

- **Artificial Intelligence:** concerned with making computers behave like humans
  - Long history: formal reasoning in philosophy, logic, …
  - Term coined in 1956 by John McCarthy (MIT)
  - study of " intelligent agents" – devices that perceive the environment and take actions that maximize its chance of success at some goal.
  - Resurgence of AI techniques in the last decade: advances in computing power, computing and data architectures, sizes of training data, and theoretical understanding
  - Deep Neural Networks: At the core of recent advancements in AI, specifically for certain classes of ML tasks
  - Applications:
    - Pattern recognition, Autonomous systems, Recommender systems, …

Machine Learning Tasks

- Supervised Learning
  - Data with **“labels”**
  - **Regression and classification**
  - Goal: use labelled data to train a predictor $\hat{y} = \hat{f}(\boldsymbol{x})$
  - Given a new observation $\boldsymbol{x}^*$, predict $\widehat{y^*} = \hat{f}(\boldsymbol{x}^*)$

- Unsupervised Learning
  - Data with **no labels**
  - **Discover patterns or structure** in the data
  - Anomalies, clusters, lower-dimensional representation

- Reinforcement Learning
  - **Experiment** and **exploit** to **make “optimal” decisions** based on **reward** structure

- Many Others
  - Semi-supervised, Representation Learning, Transfer Learning

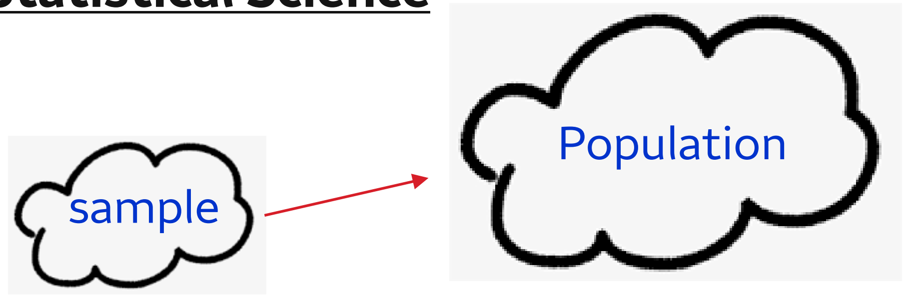
Statistical Techniques for Supervised Learning

- Data $\{(Y_i, \boldsymbol{X}_i), i = 1, \dots n\}$
- Input-output relationship: example, for continuous response: $Y = f(\boldsymbol{X}) + \epsilon$
- **Approaches:**
  - Parametric models: $f(\boldsymbol{X}) = g(\boldsymbol{X}; \beta) \rightarrow g$ a smooth function (linear, logistic, etc.)
  - Non-linear models: $f(\boldsymbol{X}) = \beta_0 + h_1(X_1; \beta_1) + h_2(X_2; \beta_2) \; \dots + h_k(X_k; \beta_k)$
  - Fitting methods: **OLS, WLS, MLE, robust**, …
  - Extensive techniques and software: model selection; diagnostics; inference
  - **More flexible techniques**:
    - Generalized Additive Models (GAM): $f(\boldsymbol{X}) = g_1(X_1) + (X_2) + \; \dots + g_k(X_k)$
    - Splines (regression splines and smoothing splines), MARS, …
    - Kernel regression, nearest-neighbors, Local smoothing (Loess);
  - **Regularized regression**: Ridge, Lasso, Elastic Net, …
    - Minimize OLS subject to penalty: $PL(f; \lambda) = \sum_i \left(Y_i - f(\boldsymbol{X}_i)\right)^2 + J(f; \lambda)$

- Many of these are now called ML these days…
- Distinction is blurring …

Supervised Learning: Statistics vs ML paradigms

- **Leo Breiman (2001) *Statistical Modeling: The Two Cultures*, Statistical Science**
  - Two paradigms: data model and algorithmic model

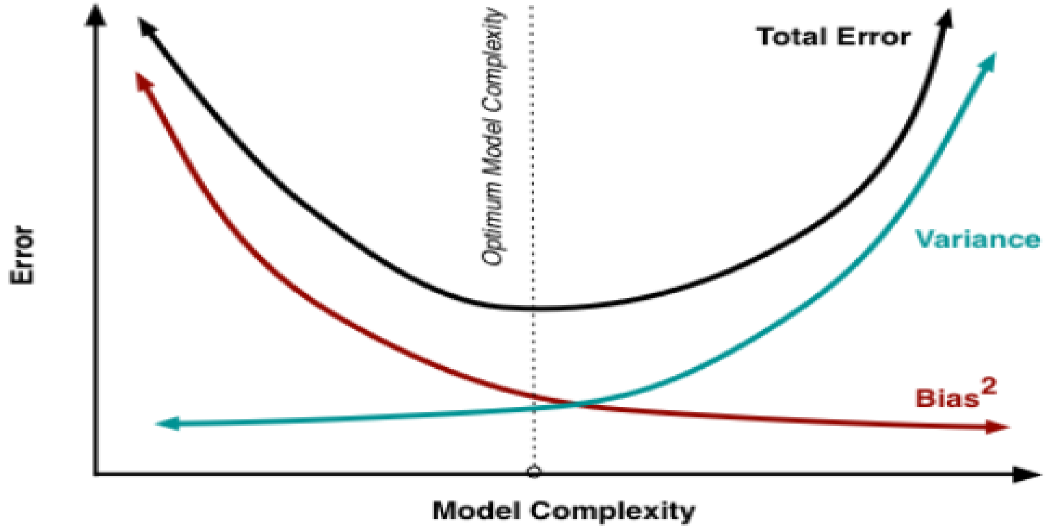


- Traditional statistics
  - Goal: "understand" the generative model
    - Identify **key drivers** and **input-output relationships**
    - **Uncertainty → Estimate** model **parameters, confidence intervals, p-values, …**
- Machine Learning
  - Goal: best predictive performance … generalization assessed on hold-out data
    - **Algorithmic** approach and **automation** of model building
      → variable selection, feature engineering, model training
    - Split dataset into training and test sets
    - Train model on former; assess performance on latter
    - Little focus on distributions, CI, hypothesis testing, …
  - **No** intrinsic **interest in** the **data generation process** (even if there's such a thing!)
  - Main goal: Optimize predictive performance (Auto ML, Kaggle, …)

- For us: Model interpretability is just as important

Supervised Machine Learning: Bias vs Variance Trade-Off

- Machine learning algorithms usually come with hyper-parameters: control performance and complexity
  - Trees: depth, number of terminal nodes, etc. to define the tree structure
  - Neural networks: number of layers, number of neurons per layer, activation functions, etc.
- Complexity is related to bias-variance trade-off.
  - Prediction MSE can be decomposed into $bias^2$ and variance.
  - Bias: $[f(x) - E\left(\hat{f}(x)\right)]$. Simpler models typically have larger bias …
  - Variance: $var\left(\hat{f}(x)\right)$. Simpler models typically have smaller variance, …
  - Good model achieves balance between bias and variance → hyper-parameter tuning



Supervised Machine Learning: Tuning

- Hyper-parameter tuning: find the best hyper parameter configuration for predictive performance: a critical component of machine learning algorithms

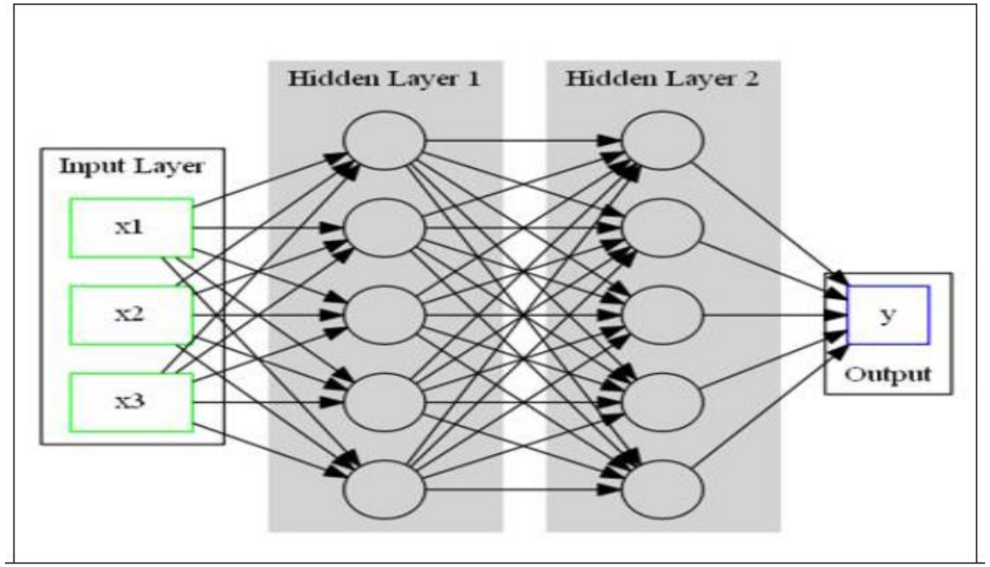
- Choice of “best” hyper-parameters is data dependent
- Tuning involves a search routine and an evaluation routine

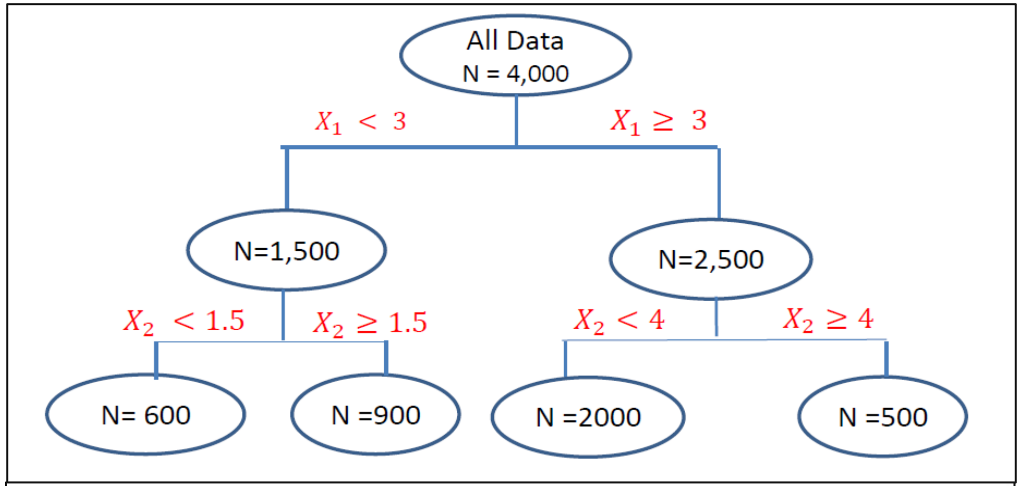
- For each hyper-parameter setting:
  - Fit the model and evaluate model performance;
  - Use search routine to find the hyper-parameter setting with that performs “best”

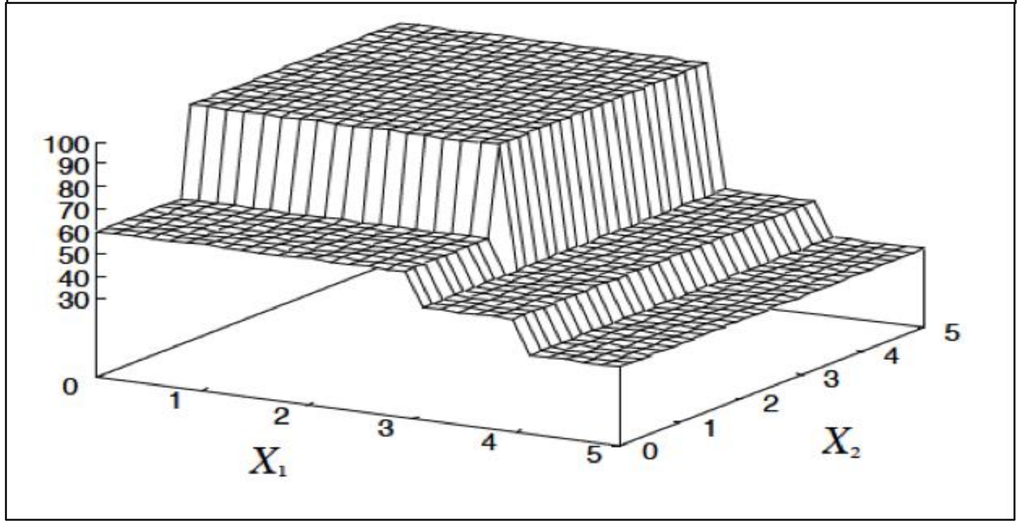
- Caution: ML models are stochastic
  - Model results depend on the random starting point
  - Many other sources of randomness

Supervised ML Algorithms

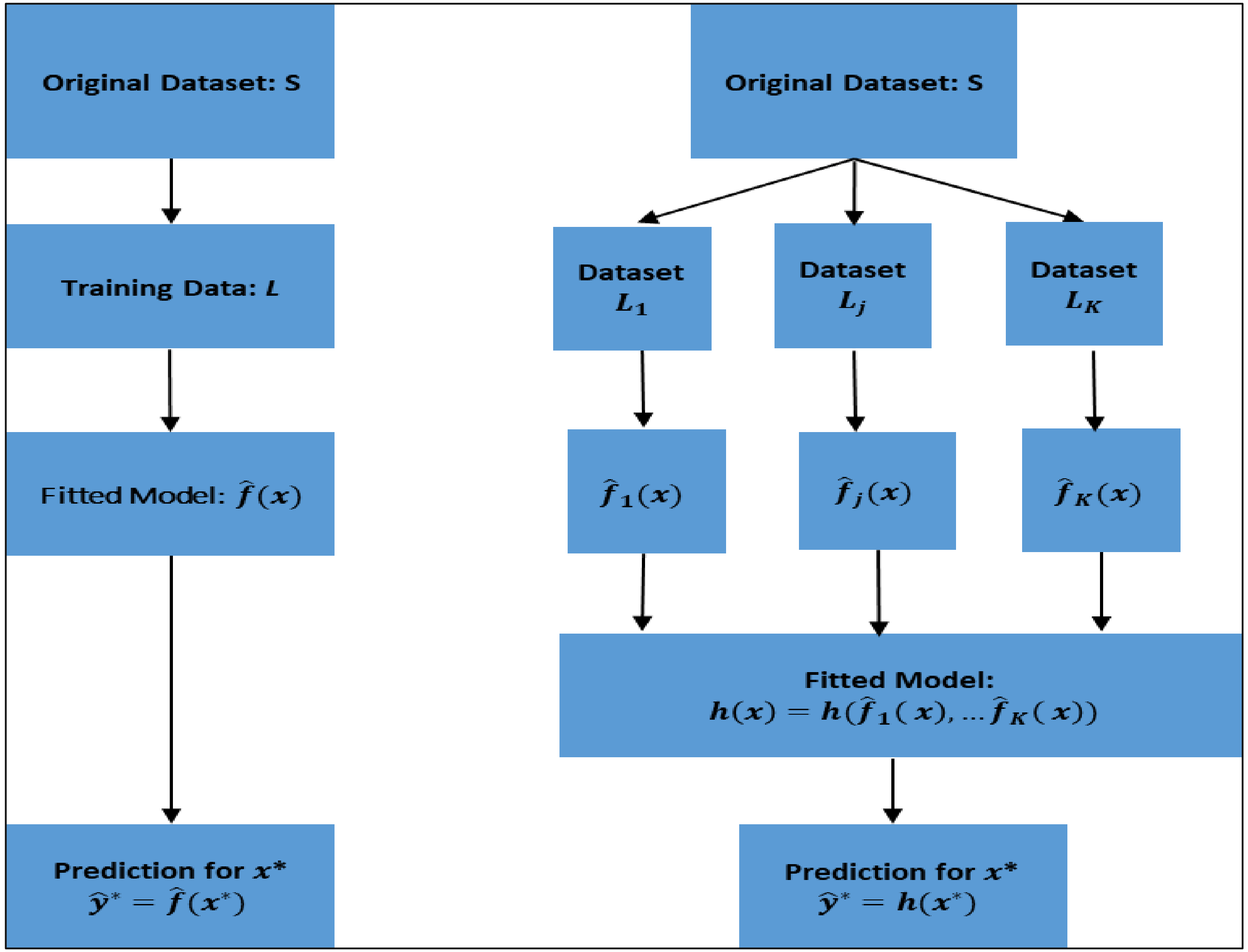
- **Ensemble algorithms**
  - Random Forests (RFs)
  - Gradient Boosting Machines (GBMs)
    - eXtreme Boosting (XGBoost)
  - Tree-based models
  - Piecewise constant within nodes
- **Feedforward Neural Networks**







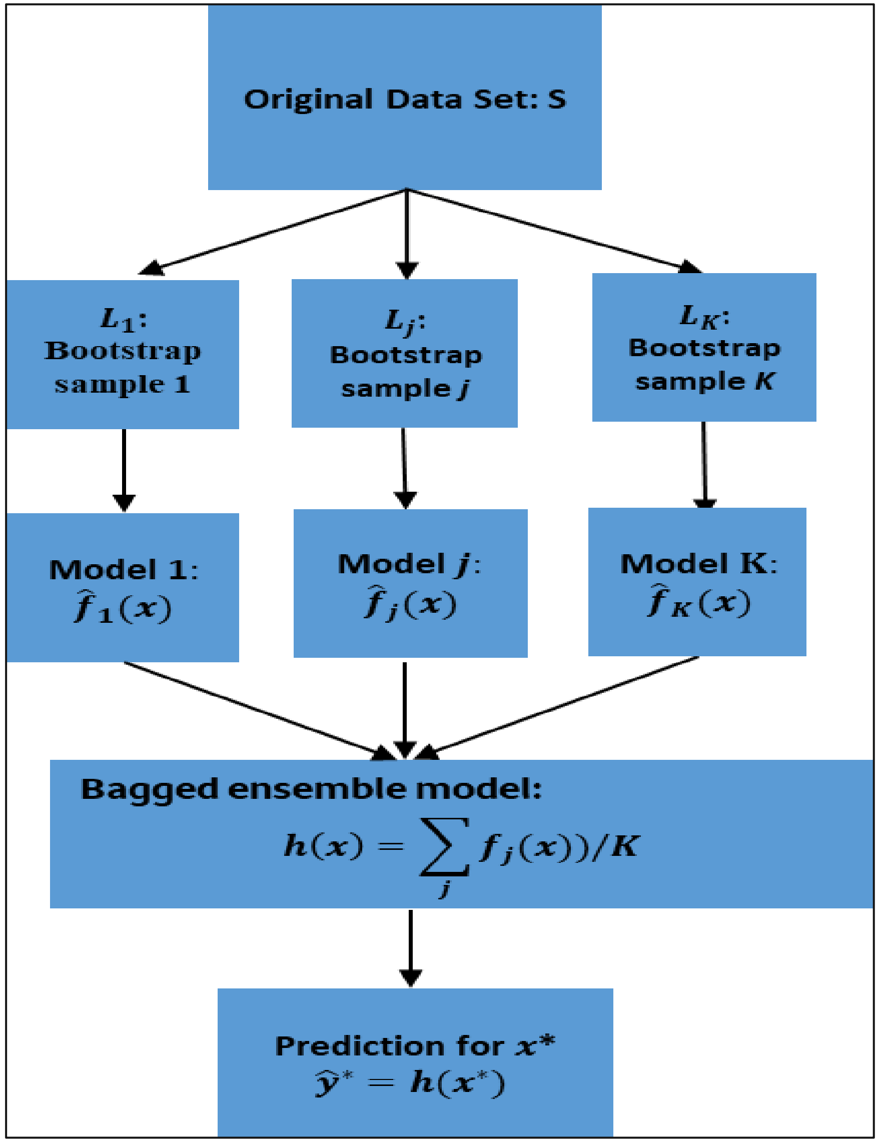
Ensemble Algorithms



Improve performance by **combining** outputs of **several individual algorithms** (“weak learners”):

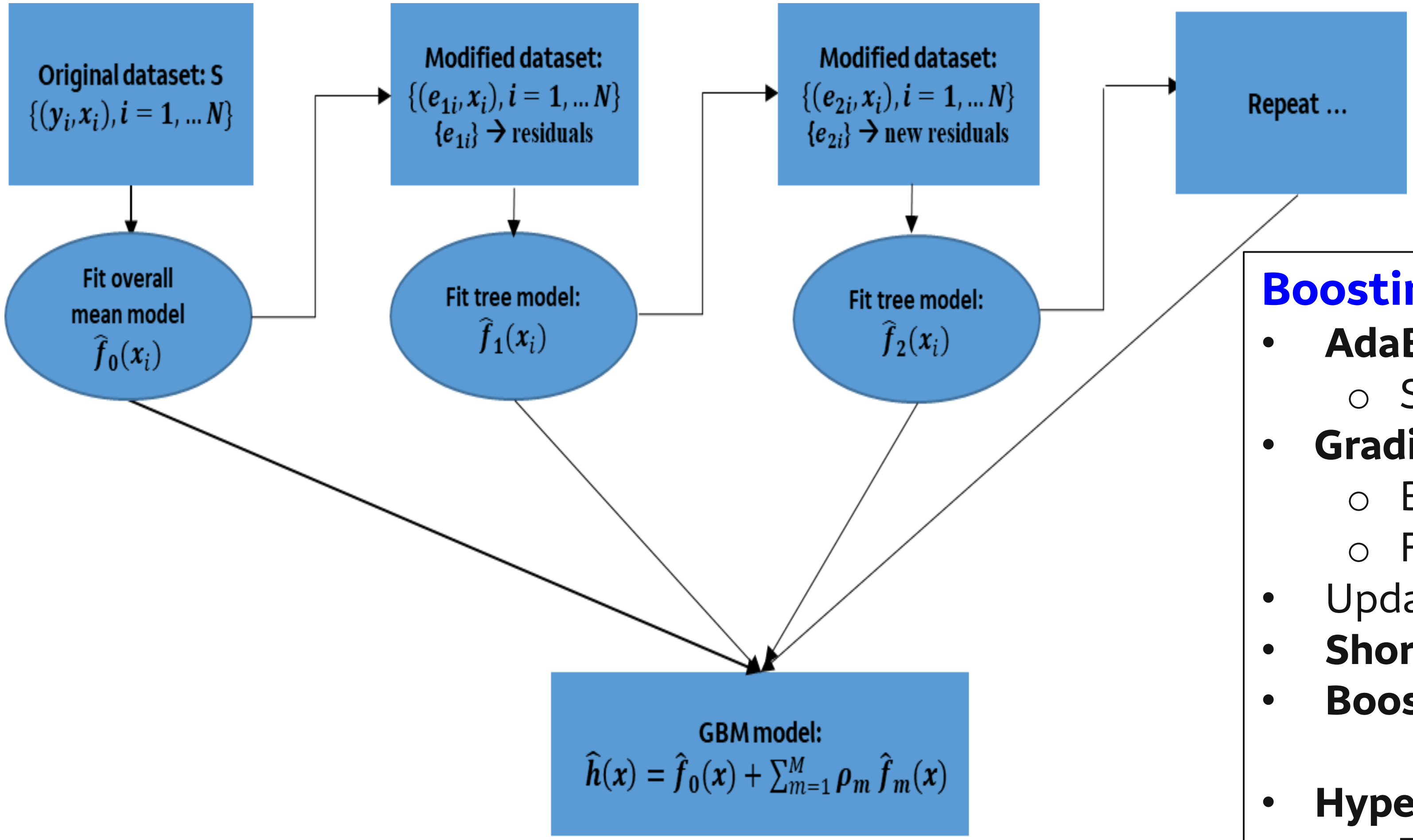
- **Bagging and Random Forest**
- **Boosting**
- **Other ensemble approaches:**
  - Model Averaging
  - Majority Voting
  - Stacking

Random Forest



- **Random Forest** (Breiman and Cutler, 1994)
  - Create **multiple datasets** by **bootstrap sampling of rows**
  - Build **deep trees** for each dataset
    - → fit piecewise constant models
    - → each tree has small bias (deep) but large variance
  - **Average** results across trees
    - → **reduce variance** and instability
- **Bootstrap aggregating** (bagging)
  - Column sub-sampling
    - → reduce correlations across trees
- **Hyper-parameters**
  - Tree depth
  - Number of trees
  - Row sampling ratio
  - Colum sampling ratio

Gradient Boosting Machine

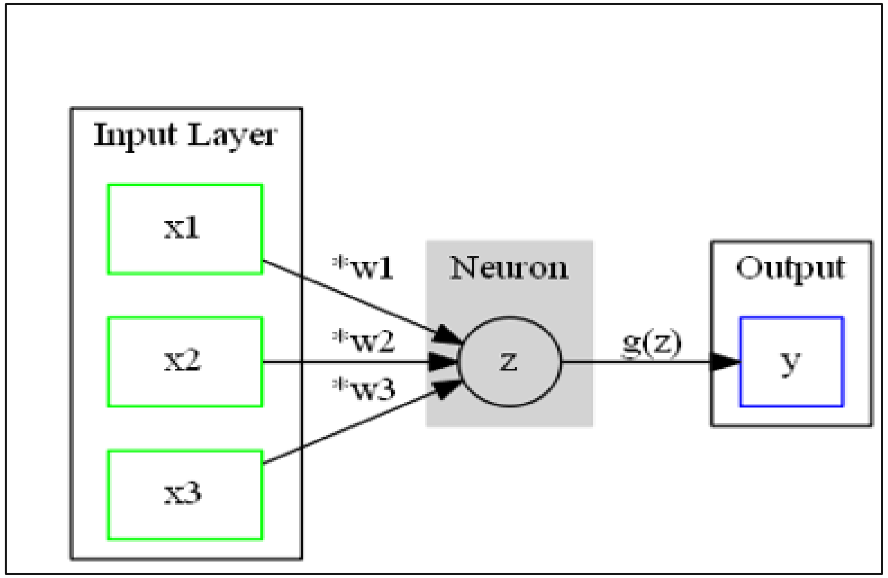


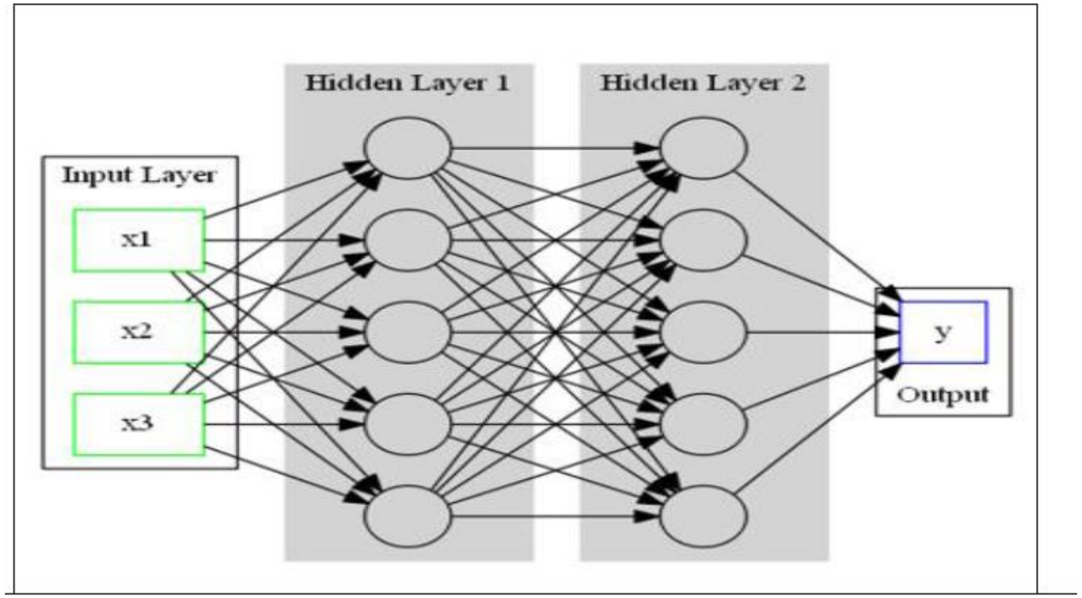
Boosting

- **AdaBoost**
  - Schapire (1990), Freund and S (1995)
- **Gradient boosting**
  - Breiman (1996), Friedman (2001)
  - Fit trees to **residuals sequentially**
- Updates in the **direction of negative gradient**
- **Short trees** → low variance, big bias
- **Boosting reduces bias**

- **Hyper-parameters** (similar to RF)
  - Tree depth
  - Number of trees
  - Learning rate
  - Row sampling ratio
  - Colum sampling ratio

Feedforward Neural Networks (FFNNs)





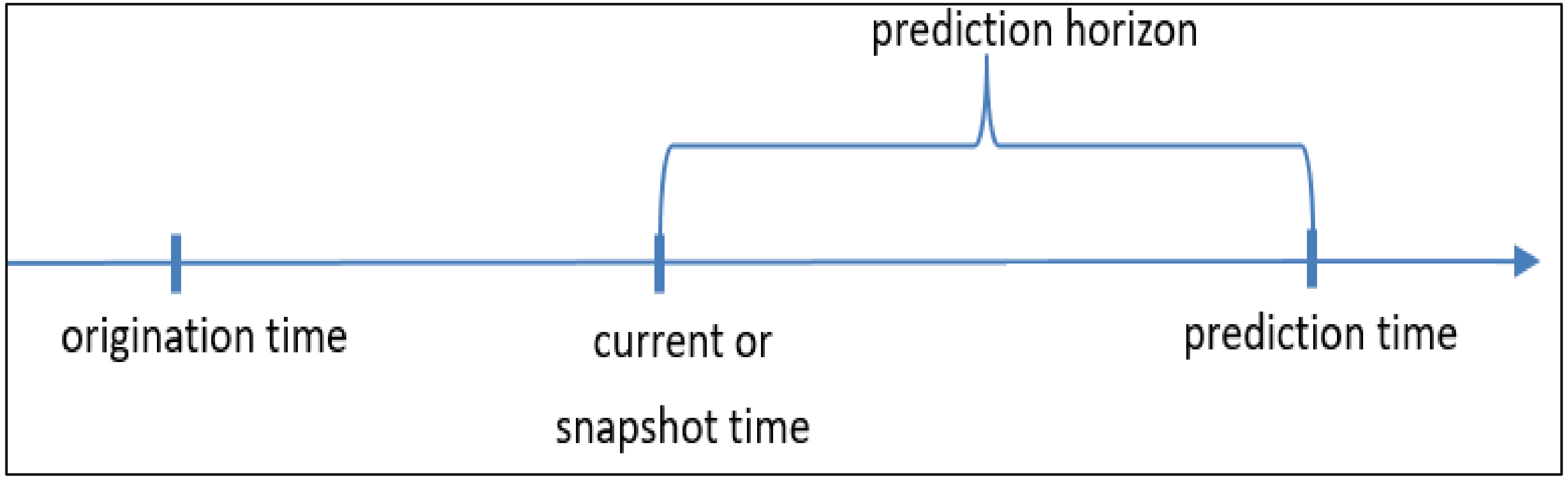
- **Mimic neuronal networks**
- **Activation function:** $g(w^T x)$
  - Sigmoidal, Hyperbolic Tan, ReLU
  - Connection to **additive index models:** $f(\boldsymbol{x}) = g(w_1 x_1 + \ ... + w_P x_P)$

- **FFNN architecture**
  - Nodes (Neurons)
  - Input, Output, and Hidden Layers
  - All nodes connected with others in next layer
- **Deep NNs**
  - Many layers
  - CNN, RNN, LSTM, …
  - BERT (Bidirectional Encoder Representations from Transformers)

Application to Home Mortgage: Modeling "In-Trouble" Loans

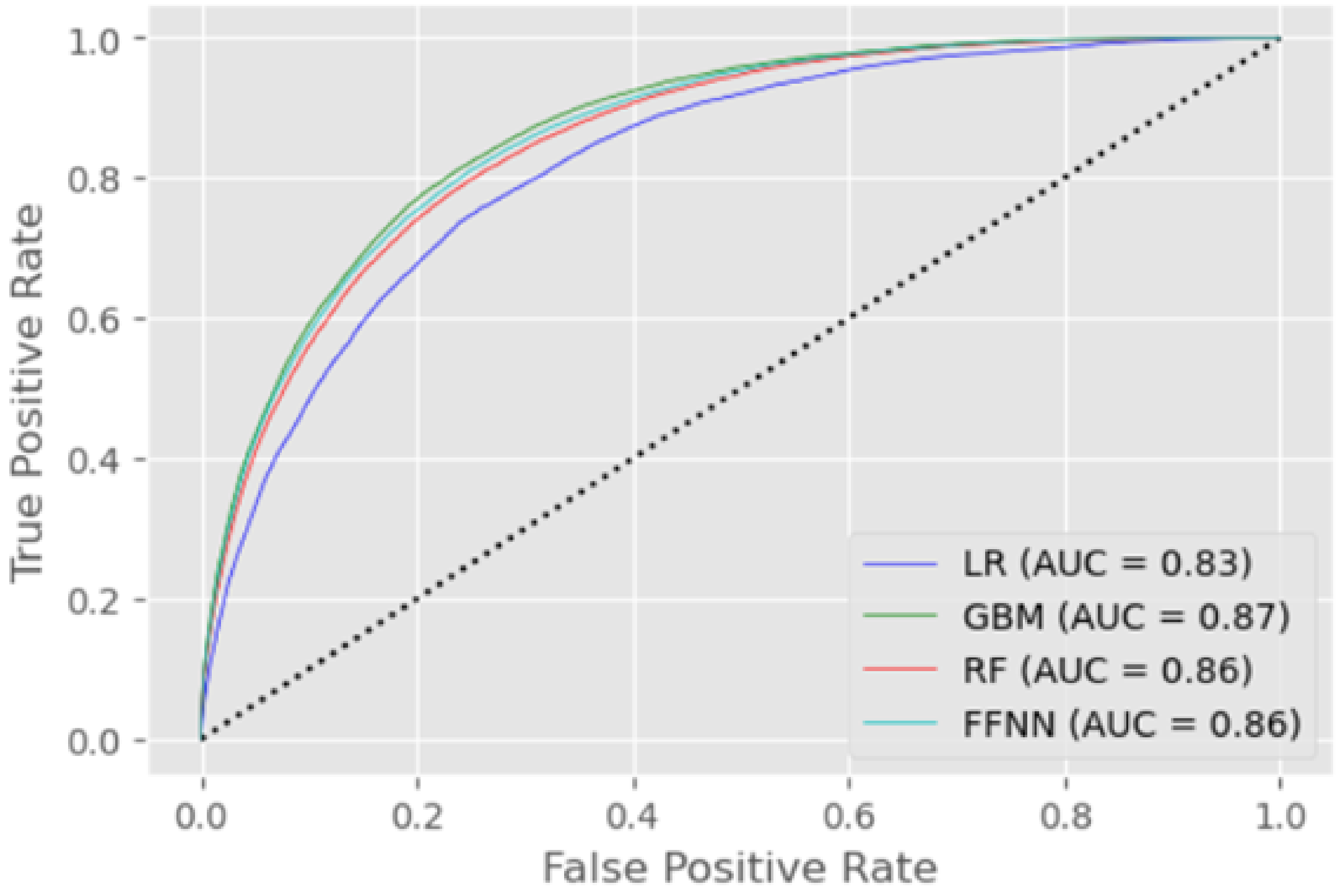
- **One portfolio: ~ 5 million observations and 22 predictors**
- **Response: binary = loan is "in trouble"** (multiple failures and connections to competing risks)
- **Predictors:** credit history, type of loan, loan amount, loan age, loan-to-value ratios, interest rates at origination and current, loan payments up-to-date, etc. (origination and over time)

Modeling framework



Loan origination, current (snapshot) and prediction times

Comparison of Predictive Performance: ROC and AUC on Test Data



- **ML with 22 predictors**
- **LR model: eight carefully selected variables**
  - ltv (loan-to-value ratio);
  - fico (credit history) at snapshot;
  - ind_financial-crisis;
  - pred_unemp_rate;
  - pred_income;
  - two delinquency status variables;
  - horizon

How typical is this "lift" in our applications?

Outline

- ❑ Machine Learning and Algorithms: Overview
- ❑ PiML: Python Interpretable Machine Learning Toolkit
- ✓ ML Model Risk and Validation
- ✓ Conceptual Soundness: Explainability and Interpretability
  - Post hoc methods
  - Inherently Interpretable Models
- ✓ Outcome Analysis
  - In Distribution Test: Weakspot and Reliability
  - Out of Distribution Test: Robustness and Resilient
- ✓ Model Fairness

Interpretable Machine Learning: Python Toolbox

✓ Low-code Interface

PiML

Python Interpretable Machine Learning

✓ High-code Programming

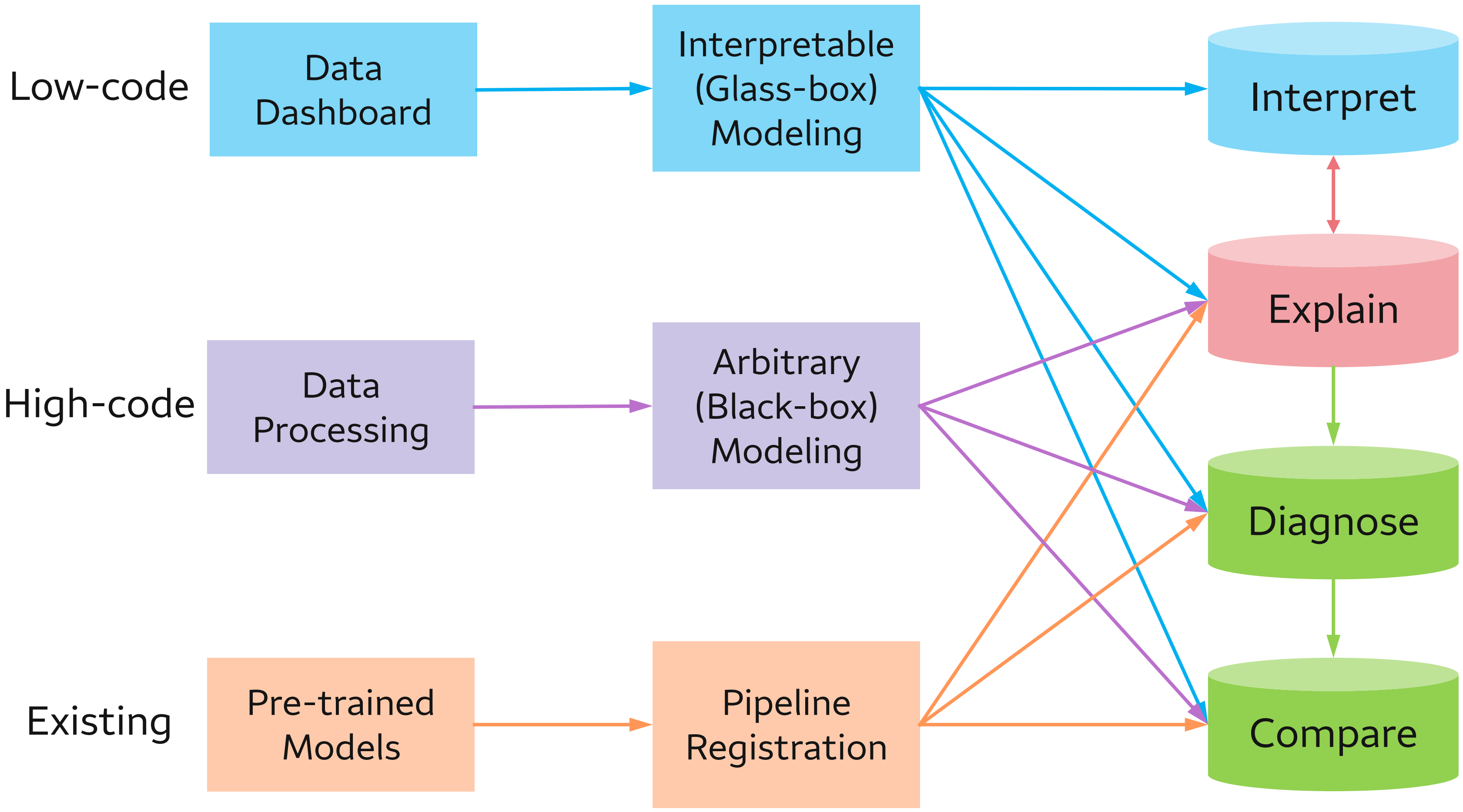
Model Development

- **Inherently interpretable ML models**
  - GLM, GAM, Tree, FIGS, XGB2
  - Explainable Boosting Machine
  - GAMI Neural Networks
  - Sparse ReLU Neural Networks
  - More advanced developments
- **Model-inherent Interpretability**
- **Post-hoc Explainability Tools**
- **Causal Feature Selection**

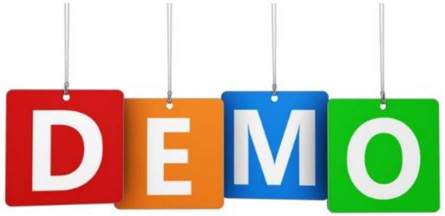
Model Validation

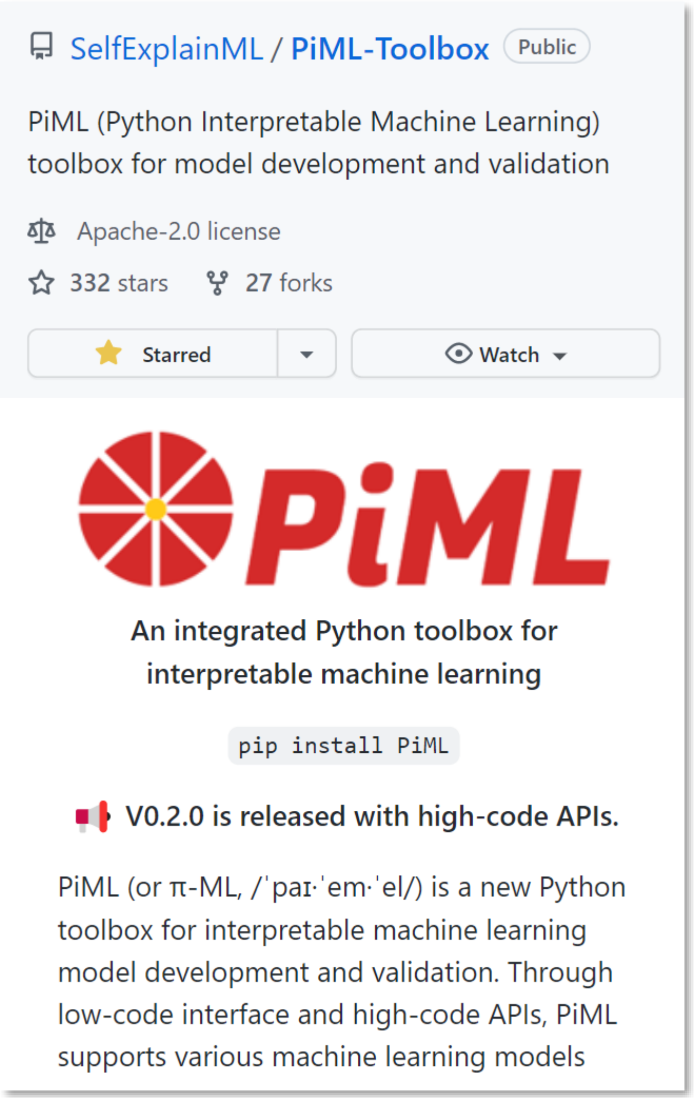
- ML Model Diagnostics and Outcome Testing
  - Performance
  - WeakSpot
  - Overfit/Underfit
  - Reliability
  - Robustness
  - Resilience
- Model Fairness
- Model Comparison and Benchmarking

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

PiML

An integrated Python toolbox for interpretable machine learning

`pip install PiML`

V0.2.0 is released with high-code APIs.

PiML (or π-ML, /ˈpaɪ·ˈem·ˈel/) 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: V0.4.2 (December 15, 2022)
- Low-code and high-code examples
- We'll demonstrate applications to:
  - Post-hoc Explainability
  - Inherently Interpretable Models
  - Sparse ReLU Deep Neural Networks
  - FANOVA-Interpretable Models: GAMI-Net and EBM
  - ML Model Diagnostics and Validation, etc.

Outline

- ❑ Machine Learning and Algorithms: Brief Overview
- ❑ PiML: Python Interpretable Machine Learning Toolkit
- ✓ ML Model Risk and Validation
- ✓ Conceptual Soundness: Explainability and Interpretability
  - Post hoc methods
  - Inherently Interpretable Models
- ✓ Outcome Analysis
  - In Distribution Test: Weakspot and Reliability
  - Out of Distribution Test: Robustness and Resilient
- ✓ Model Fairness

Opportunities with ML

General:

- **Advent of "Big Data"**
  - ✓ **New sources of data**: social media, sensor networks, intelligent systems, …
    - o Text, conversations, images, …

- **Advances in computing and data storage technologies**
  - ✓ Infrastructure for data collection, warehousing, transfer, and management
  - ✓ Efficient and scalable algorithms and associated technologies for analyzing large datasets
  - ✓ Open-source algorithms
  - ✓ Cloud storage and computing

    → Democratization of Data Science

Specific:

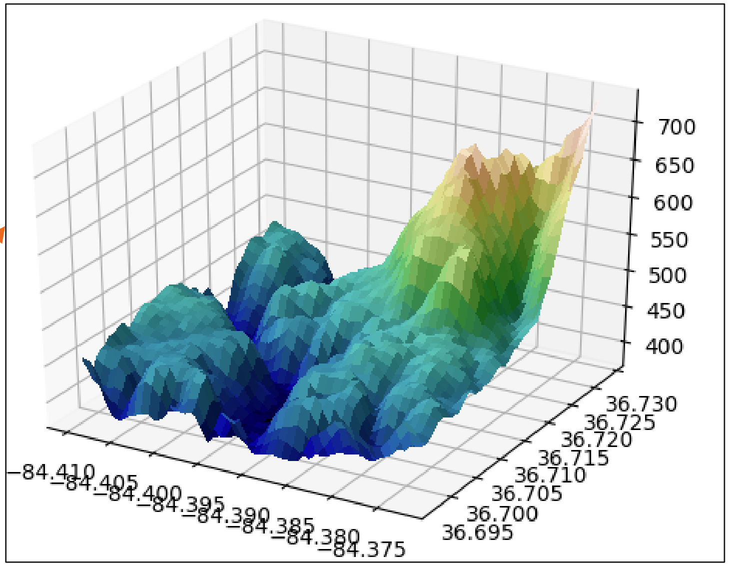
- Availability of large datasets and fast algorithms

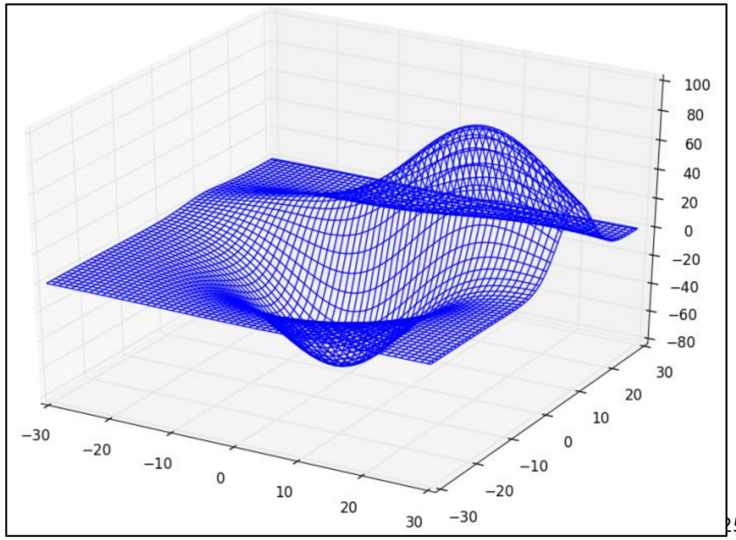
  → flexible modeling … move away from restrictive parametric models
- SML model:
  - ➢ Improved predictive performance
  - ➢ Semi-automated approach to feature engineering and model training → ideal for Big Data
- New data sources and computing technologies open up new opportunities
  - ➢ Text, speech, images, …
  - ➢ More timely information and decision making

Challenges with Use of ML: Model Risks

- **Model is a black box and results are opaque**
  - Ensure results “make sense”
  - Explain/interpret results and conclusions to multiple stakeholders
  - Consistent with subject-matter knowledge (right variables included, monotonicity, etc.)
- **Potential for model bias**
  - Challenge in assessing fairness
- **Model is highly flexible: danger of overfitting training data**
  - Assess model robustness
- **Generalizability**
  - Traditional approach → select test data to represent training data
  - But … what will be the performance if there is covariate shift?
- **Uncertainty quantification**
  - Many sources of variation and randomness in ML models
  - Identify and quantify

What is a model?
Function-Fitting vs Modeling





Outline

- ❑ Machine Learning and Algorithms: Brief Overview
- ❑ PiML: Python Interpretable Machine Learning Toolkit
- ✓ **ML Model Risk and Validation**
- ✓ Conceptual Soundness: Explainability and Interpretability
    - Post hoc methods
    - Inherently Interpretable Models
- ✓ Outcome Analysis
    - In Distribution Test: Weakspot and Reliability
    - Out of Distribution Test: Robustness and Resilient
- ✓ Model Fairness

Explainability and Interpretability

Approaches:

I. Post hoc explanations: Techniques for understanding results after fitting model

II. Inherently interpretable algorithms
   a) Additive index models
   b) Low-order functional ANOVA models

Some questions in trying to understand model results

- Main drivers in the model?
- Input-output relationship important variables?
  Nonlinearity? Interaction? Impact of correlations? Strategies?
- Is model consistent with known relationships?

- What are the characteristics defining fraud accounts?
- Why did a particular customer not get their loan approved
  - Adverse action explanation

- Bias and fairness

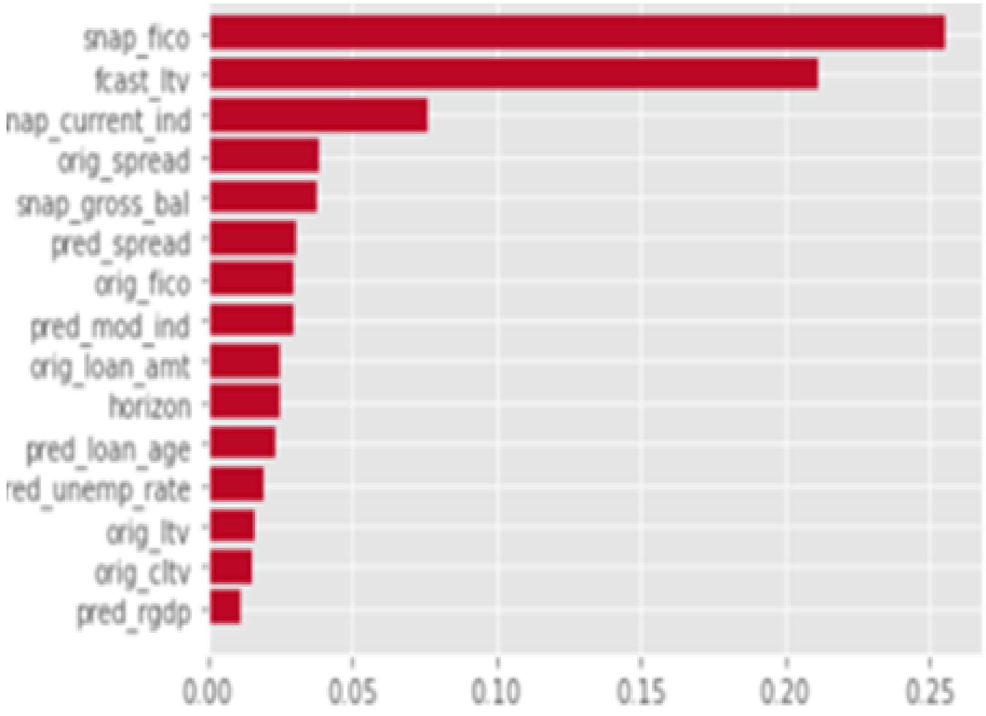
Post hoc global: Identifying important predictors/features

- **Permutation based: Model agnostic**
  - Randomly permute the rows for variable (column) of interest while keeping everything else unchanged
  - Compute the change in prediction performance as the measure of importance.
  - Issue:
    - Double counting of interactions

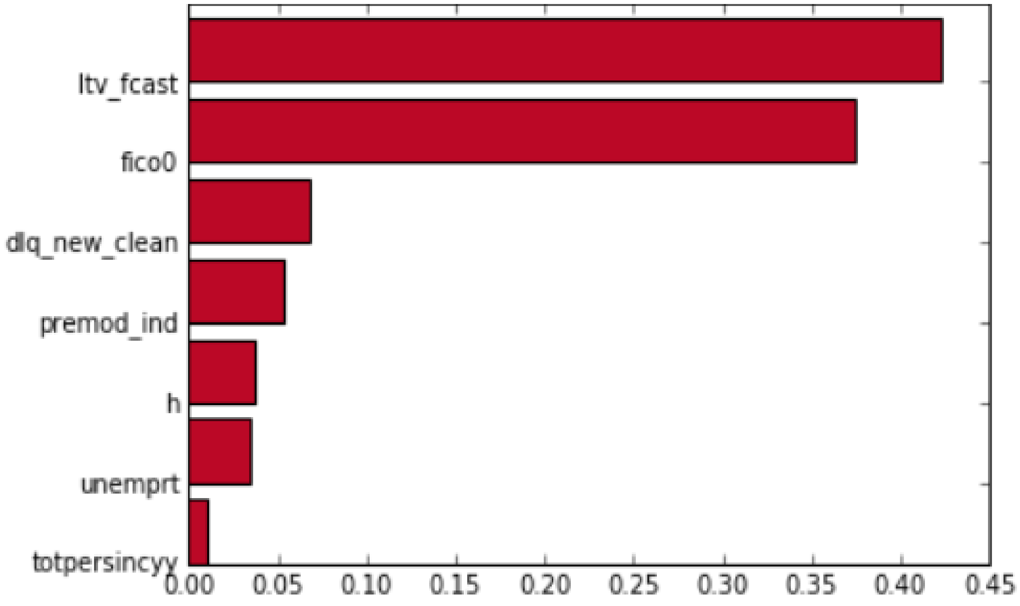
Y	X1	X2	X3	X4	X5
2	1.5	0	4.5	10.2	3.0
4	2.7	1	5.3	8.7	4.2
8	3.3	1	7.2	19.3	17.6
3	1.9	0	3.3	7.8	21.2

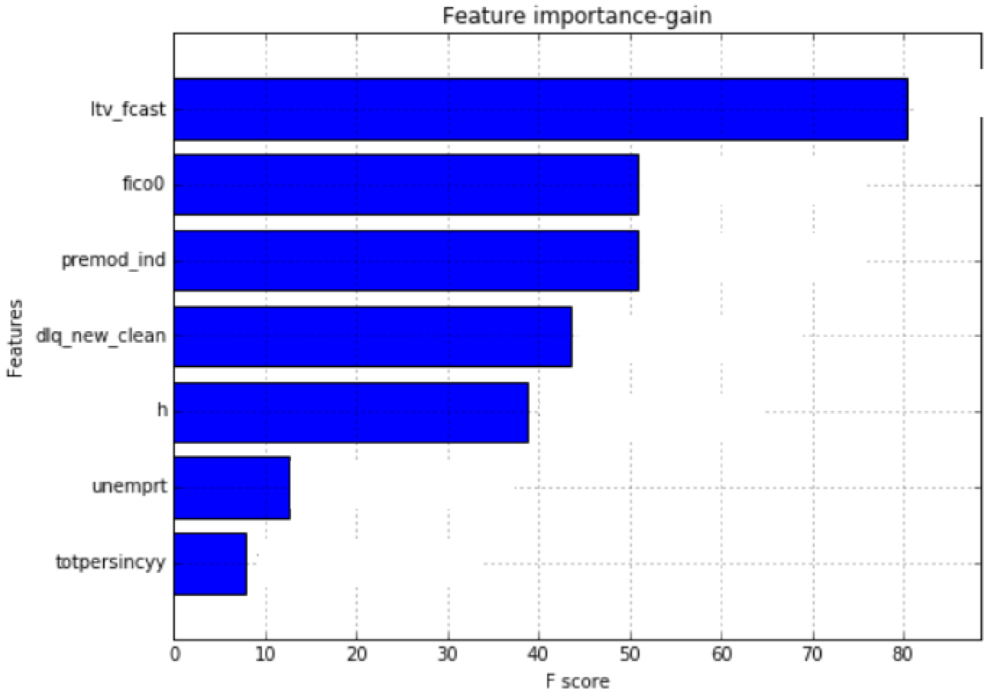
- **Selected Others**
  - **Tree-based** importance metrics
    - Importance of a variable x_j → total reduction of impurity at nodes where x_j is used for splitting
    - For ensemble algorithms, average over all trees
  - **Global Shapley**
    - Based on Shapley decomposition (1953); Owen (2014)
    - Model agnostic but **computationally intractable**

Home Mortgage-XGB



Variable importance: Home Mortgage Data Comparison of permutation and tree-based





LTV_fcast and fico0 are both identified as top important variables
Ranking of others vary

Shapley effects for global variable importance

- Shapley (1951-53): game theory: evaluate contribution of a player in a cooperative game.
- Proposed by Owen and others to explain variable importance in ML (contribution of variables)

$$\phi_k = \sum_{\boldsymbol{S} \subseteq K \setminus k} \frac{|\boldsymbol{S}|!\,(|\boldsymbol{K}| - |\boldsymbol{S}| - 1)!}{|\boldsymbol{K}|!} \big(val(\boldsymbol{S} \cup k) - val(\boldsymbol{S}) \big)$$

– $val(\cdot)$ is conditional variance

– We wont' discuss this application in detail here as computation is prohibitive
– But will come back to it later for local explanation

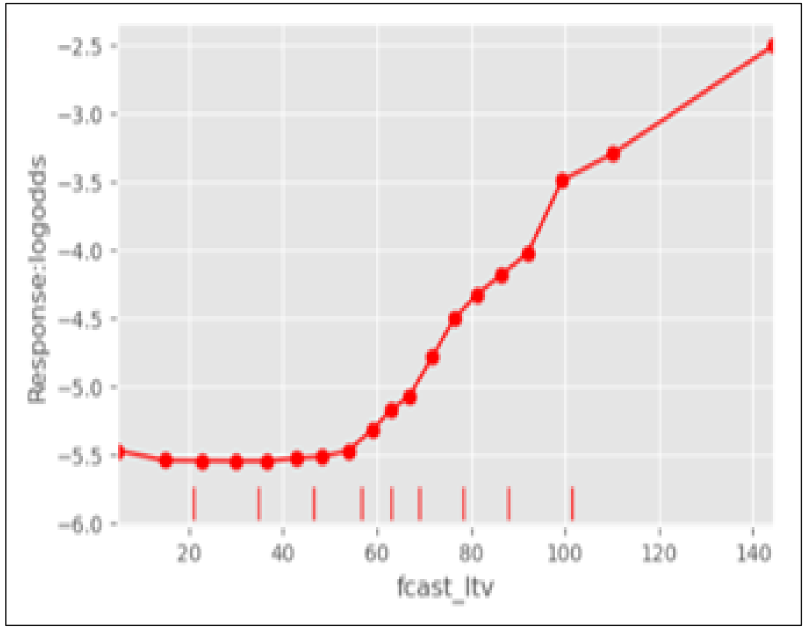
Understanding input-output relationships: 1-dimensional partial dependence plots

- PDPs → visualize the input-output relationship: Friedman (2001).
- Removes the effect of other variables by averaging over them
  → the "partial effect" of the variable of interest
- Theoretical PD Function for single input variable X_j:

  Partition $\boldsymbol{X}$ into $(X_j, \boldsymbol{X}_{-j})$ where $\boldsymbol{X}_{-j}$ is the complementary set.

$$f_{PD}(x_j) = \int f(x_j, \boldsymbol{x}_{-j})\, p(\boldsymbol{x}_{-j})\, d\boldsymbol{x}_{-j}$$

Home Mortgage
1-D PDP for forecast_LTV



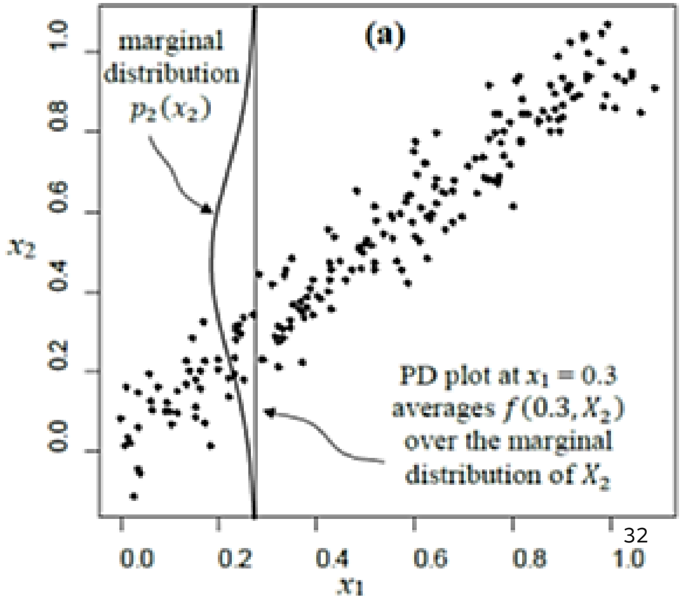
PDP (empirical version)

- Write $\hat{f}(\boldsymbol{X}) = \hat{f}(X_j, \boldsymbol{X}_{-j})$ be the fitted model
- For each X_j fixed at c, compute the average value of $\hat{f}$ over the entire data

$$\hat{f}_{PD}(x_j = c) = \frac{1}{N}\sum_{i=1}^{N} \hat{f}(x_j = c, \boldsymbol{x}_{-j,i})$$

- Plot $f_{PD}(x_j)$ against x_j over the grid

- **Extrapolation →**



Source: Apley, D. W. (2016)

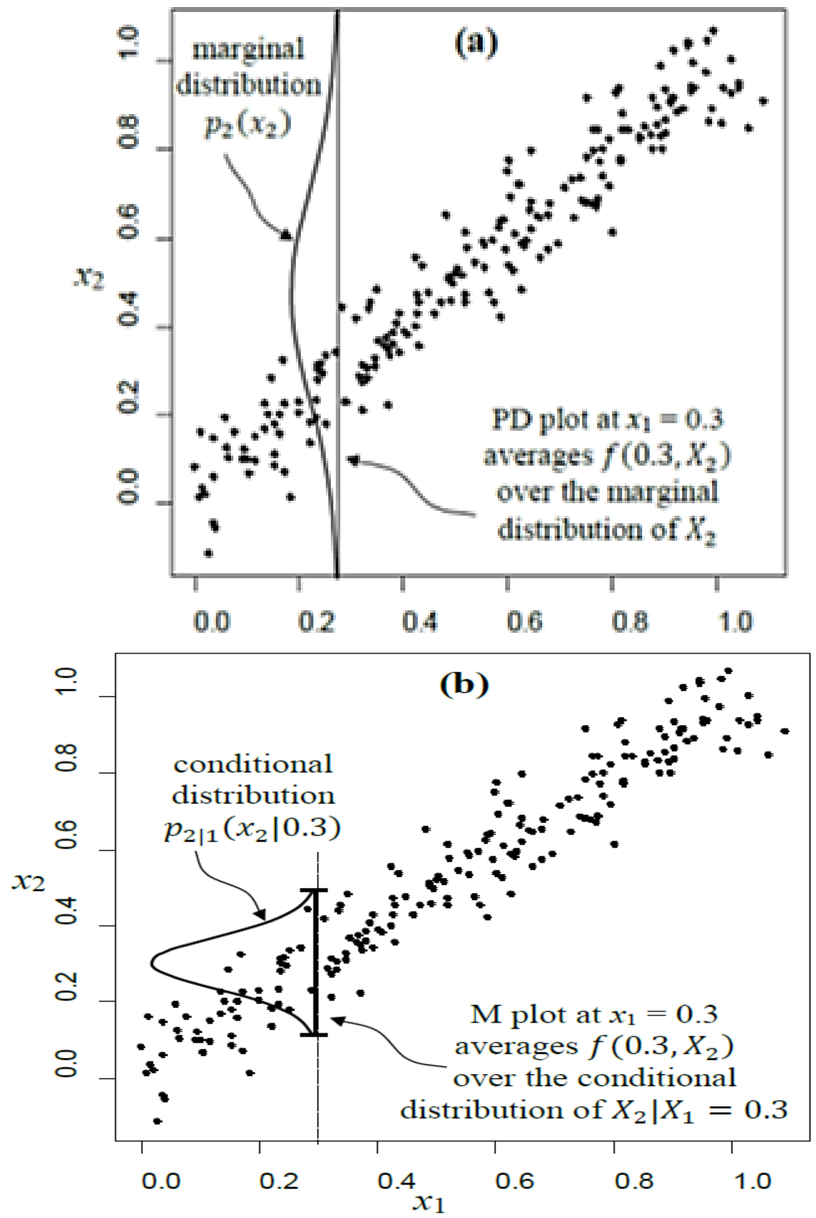
Accumulated Local Effects (ALE) Plot for highly correlated data

Need technique that is not affected by correlations

ALE Plot (Apley, 2016)

$$f_{ALE}(x_j) = \int_{z_{j,0}}^{x_j} E_{\boldsymbol{X}_{-j}|X_j} \left\{ \frac{\partial f(\boldsymbol{X})}{\partial X_j} | X_j = z_j \right\} dz_j$$

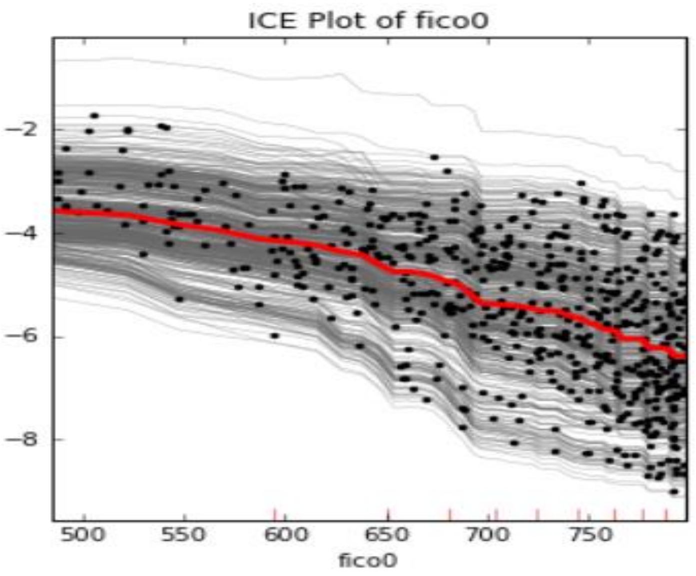
- Focuses on conditional distribution to avoid extrapolation
- Definition uses partial derivatives
- Can be approximated by finite differences, but …
- Reduces to PDP when predictors are independent



Techniques to detect two-way interactions

- Individual conditional expectation (ICE) plots
- Two-dimensional PDP Plots
- H-statistics

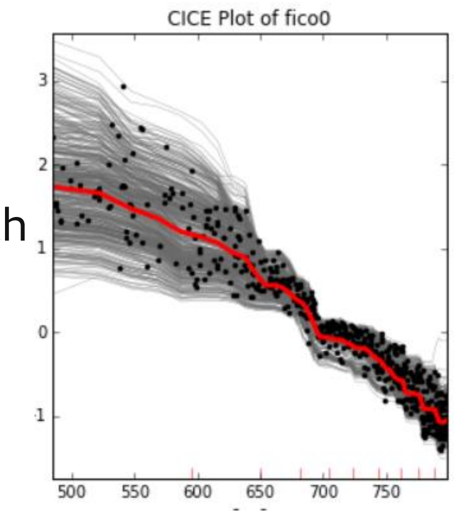
Individual conditional expectation (ICE) plots

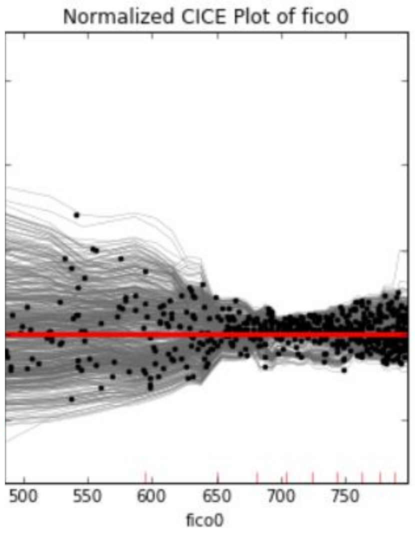


- ICE plot: more detailed version of PDP (Goldstein et al., 2015)
  - 1d-PDP $\hat{f}_{PD}(x_j = c) = \frac{1}{N}\sum_{i=1}^{N} \hat{f}(x_j = c, \boldsymbol{x}_{-j,i})$
  - average of $\hat{f}(x_j, \boldsymbol{x}_{-j,i})$ over the entire data
  - when there are interaction effects, $\hat{f}(x_j, \boldsymbol{x}_{-j,i})$ will have different patterns for different $\boldsymbol{x}_{-j,i}$.
  - Averaging loses interaction information

- **ICE plot: plot all N** curves $\hat{f}(x_j, x_{-j,i}), i = 1, 2, \dots, N$, conditional on $\boldsymbol{x}_{-j,i}$.
- Each curve is localized for a single ith observation.
  - Can see if there is any change in the input-output relationships for x_j
  - Including possible interaction effect
  - Cannot identify nature of interaction

- **Centered ICE (CICE) Plots**
  - Given the sample n, CICE plot for X_i is to subtract each ICE curve with the weighted mean of the curve.
- **Normalized CICE plot**
  - subtracting CICE curves with the corresponding centered partial dependence curves.





H-statistics for 2D-Interactions

- H_{jk} to measure the interaction between X_j and X_k (Friedman and Popescu; 2005)

$$H_{jk}^2 = \frac{\sum_{i=1}^{N}\left[f_{PD}(x_{j,i}, x_{k,i}) - f_{PD}(x_{j,i}) - f_{PD}(x_{k,i})\right]^2}{\sum_{i=1}^{N} f_{PD}^2(x_{j,i}, x_{k,i})}, \quad H_{jk} = \sqrt{H_{jk}^2}$$

  - $f_{PD}(x_{j,i}, x_{k,i})$, $f_{PD}(x_{j,i})$, $f_{PD}(x_{k,i})$ are the centered partial dependence functions
  - H_{jk}^2 is the proportion of variation in $f_{PD}(x_{j,i}, x_{k,i})$ unexplained by an additive model: relative measure.

- Challenge:
  - When two variables are irrelevant, both denominator and nominator are small
  - H_{jk} can be high due to instability

- Unnormalized version of H-statistic:

$$\widetilde{H}_{jk}^2 = \frac{1}{N}\sum_{i=1}^{N}\left[f_{PD}(x_{j,i}, x_{k,i}) - f_{PD}(x_{j,i}) - f_{PD}(x_{k,i})\right]^2, \ \widetilde{H}_{jk} = \sqrt{\widetilde{H}_{jk}^2}$$

- Computation is expensive
- No calibration

Application to Home Lending Data

- Unscaled H-statistics

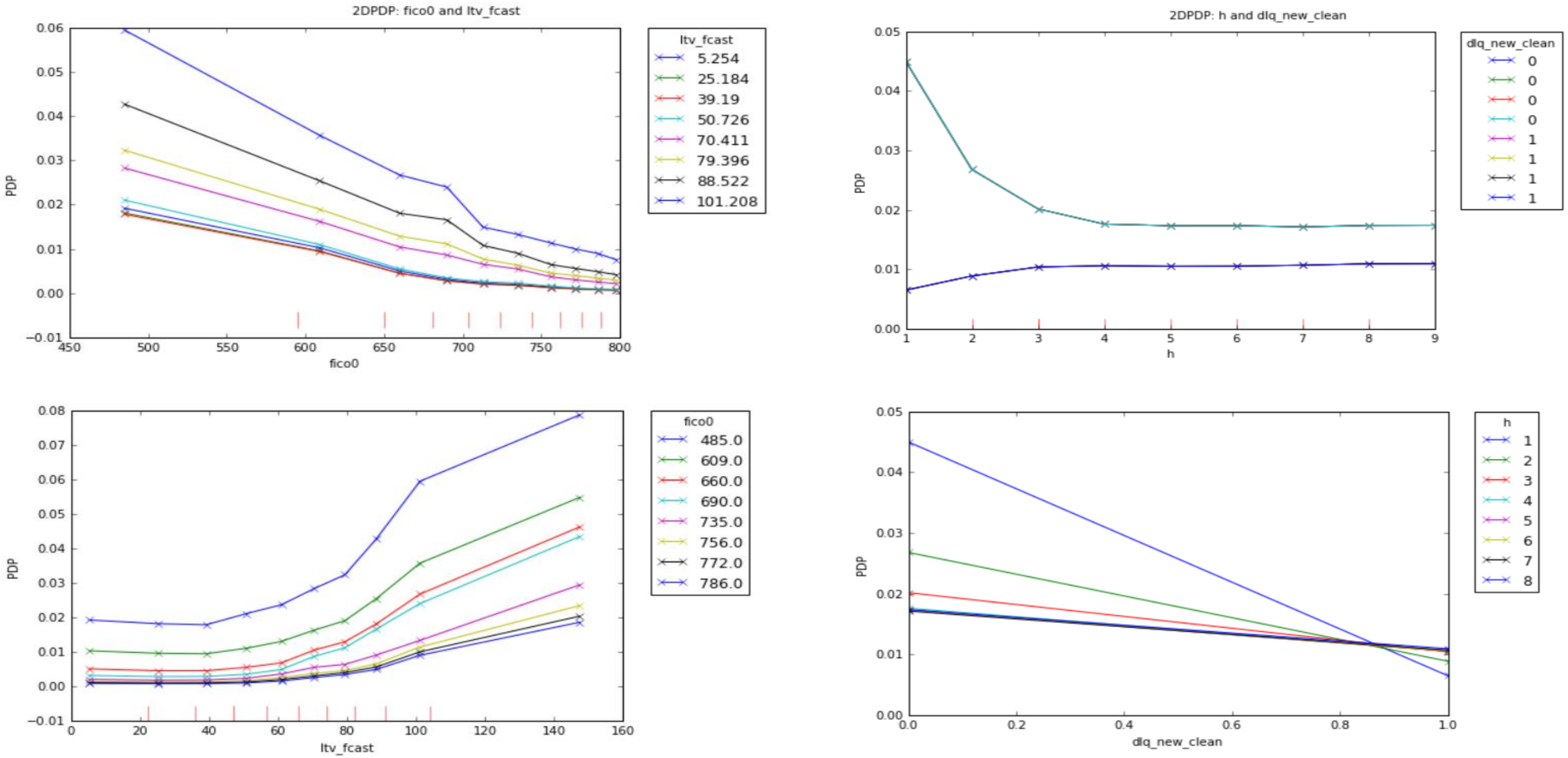
	fico0	ltv_fcast	dlq_new_clean	unemprt	totpersincyy	h	premod_ind
fico0	NaN	0.1630	0.1224	0.0820	0.0360	0.0339	0.1107
ltv_fcast	0.1630	NaN	0.0518	0.0291	0.0286	0.0186	0.0843
dlq_new_clean	0.1224	0.0518	NaN	0.0101	0.0071	0.2296	0.0003
unemprt	0.0820	0.0291	0.0101	NaN	0.0232	0.0122	0.0094
totpersincyy	0.0360	0.0286	0.0071	0.0232	NaN	0.0068	0.0661
h	0.0339	0.0186	0.2296	0.0122	0.0068	NaN	0.0192
premod_ind	0.1107	0.0843	0.0003	0.0094	0.0661	0.0192	NaN

Relatively stronger interactions between:

- horizon and dlq_xxx
- fico0 and LTV_fcast

2-dimensional PDPs for home lending case

- Can plot 2-D PDPs for interactions identified as important
- Many ways to display interaction information



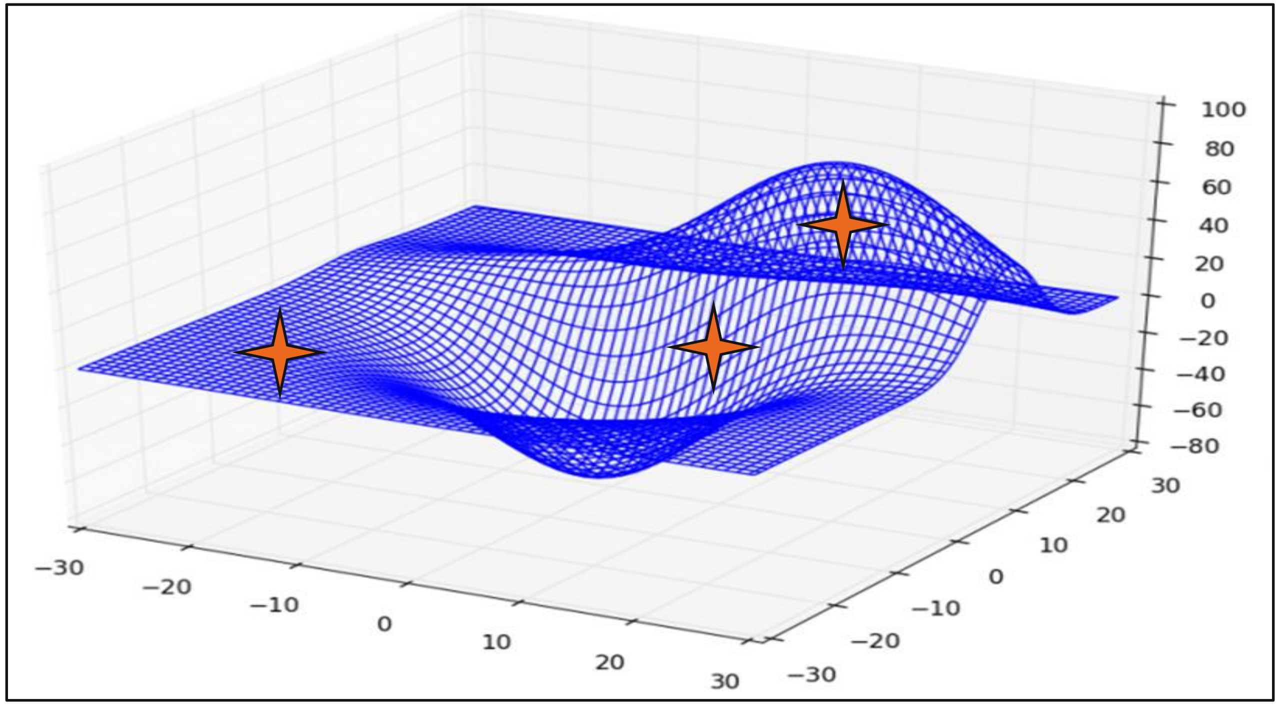
Post hoc techniques for local explanation

- Questions of Interest:
  1. How can we interpret the response surface locally at selected points of interest?
     - If fitted model is linear: $\hat{f}(\boldsymbol{x}) = b_0 + b_1 x_1 + \ \dots b_K x_k$, magnitudes and signs of coefficiencts
  2. Given the predicted value at a point of interest $\hat{f}(\boldsymbol{x}^*) = \hat{f}(x_1^*, \dots, x_K^*)$, what are the contributions of the different variables $\{x_1, \dots x_K\}$ to the prediction?
     - If fitted model is linear: $\hat{f}(\boldsymbol{x}) = b_0 + b_1 x_1 + \ \dots b_K x_k$, contribution of x_j^* is $b_j x_j^*$

- How to extend these interpretations to fitted models from ML algorithms?

Techniques for local explanation: Question 1

- How can we interpret the response surface locally at selected points of interest?
  - Local interpretable model-agnostic explanations
  - LIME (Ribeiro et al. 2016)
  - Fit a linear model locally around the point $b_0 + b_1 x_1^* + \ \ldots b_K x_K^*$
  - Use this for interpretation
  - Many ways to sample points locally and fit the linear model



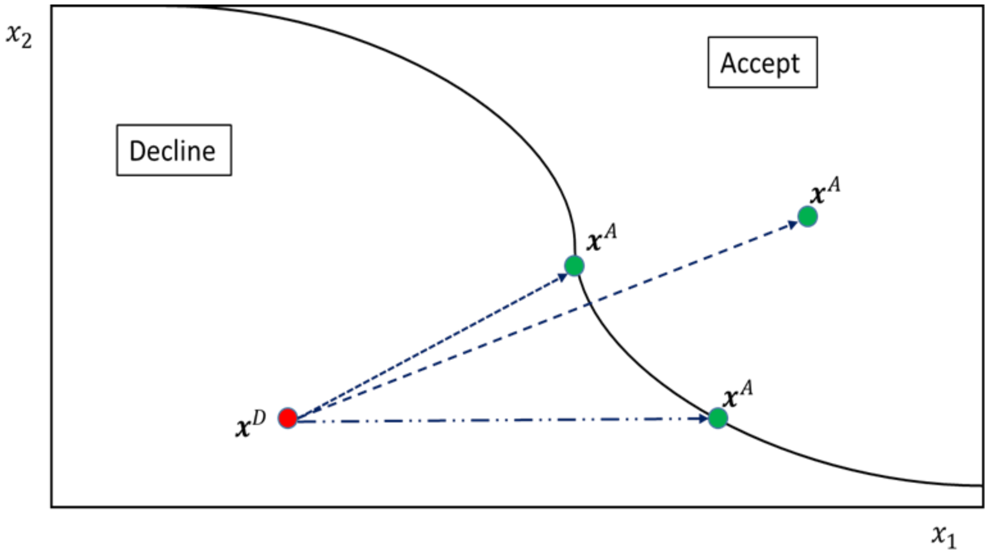
Techniques for local explanation: Question 2

Given $\hat{f}(\boldsymbol{x}^*) = \hat{f}(x_1^*, \dots, x_K^*)$ at the point $\boldsymbol{x}^*$:

- What are contributions of $\{x_1, \dots x_K\}$ to that prediction?
- Must compute contribution with and without each of the variables
- Usually done with respect to "average" – reference point
- This can vary with on the application
- Adverse action →

- Many approaches
  - Integrated Gradient, …
  - SHAP (local version of Shapley)
  - Illustration using Baseline Shapley → next

"Adverse" action (AA) explanation on declined decisions



- Declined customers are entitled to an "explanation"

- Use historical data $\{y_i, \boldsymbol{x}_i\}, i = 1, \ldots n$ to develop model for probability of default (PoD) - $p(\boldsymbol{x})$
- Decision:
  – Accept application with $\boldsymbol{x}^*$ if $p(\boldsymbol{x}^*) \leq \boldsymbol{\tau}$;
  – Decline otherwise
- Model based on complex algorithm
- How to explain the decision?
  - Compute the difference: $[p(x^D) - p(x^A)]$
  - Attribute the difference to the (important) predictors
  - Better to do in terms of $f(x) = logit\ p(x)$
  - Decompose $[f(x^D) - f(x^A)] = E_1(x^D, x^A) + E_2(x^D, x^A) + \ldots + E_K(x^D, x^A)$

    $E_k(x^D, x^A)$ is allocation to (contribution by) k –th predictor

General expression for AA with Baseline Shapley

$$[f(\boldsymbol{x}^{\boldsymbol{D}}) - f(\boldsymbol{x}^{\boldsymbol{A}})] = E_1 + \cdots + E_K,$$

$$E_k = E_k(x^D;\, x^A) = \sum_{S_k \subseteq K \setminus \{k\}} \frac{|S_k|!\,(|K| - |S_k|)!}{|K|!} \left(f\left(x_k^D;\, x_{S_k}^D; x_{K \setminus S_k}^A\right) - f\left(x_k^A; x_{S_k}^D; x_{K \setminus S_k}^A\right) \right)$$

- Application of Shapley concept (Shapley, 1951+)
  - Adapted to global explanation in ML (Owen 2014; and others)
  - Local explanation (Lundberg et al. 2018, others)
  - Computationally intractable
- Baseline Shapley (Sundararajan, M. and Najmi, A. (2020) – easier to compute
- Adaptation to adverse action (Nair et al. 2022)

AA explanation with two predictors

Linear model: $f(\boldsymbol{x}) = b_0 + b_1 x_1 + \cdots + b_K x_K$

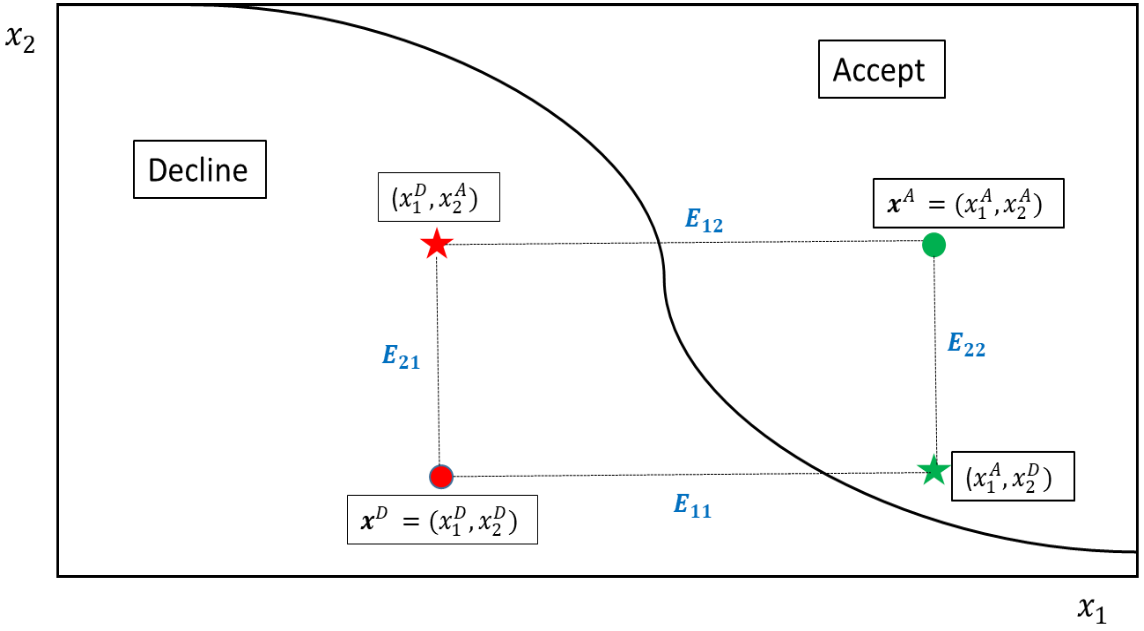
$[f(\boldsymbol{x^D}) - f(\boldsymbol{x^A})] = b_1(x_1^D - x_1^A) + b_2(x_2^D - x_2^A) + \ \ldots$

GAM: $f(\boldsymbol{x}) = g_1(x_1) + \cdots + g_K(x_K)$

$f(\boldsymbol{x^D}) - f(\boldsymbol{x^A})$

$= [g_1(x_1^D) - \ g_1(x_1^A)] + \cdots + [g_K(x_K^D) - \ \ g_K(x_K^A)]$

$E_k = g_k(x_k^D) - g_k(x_k^A)$



Interactions? $f(\boldsymbol{x}) = b_0 + b_1 x_1 + b_2 x_2 + b_{12} x_1 x_2$

$b_1(x_1^D - x_1^A) + b_2(x_2^D - x_2^A) + b_{12}(x_1^D x_2^D - x_1^A x_2^A)$

General: (Nair et al. 2022)

- $E_1 = \frac{1}{2}(E_{11} + E_{12}) \rightarrow \frac{1}{2}\{[f(\ x_1^D, x_2^D) - f(x_1^A, x_2^D)] + [f(x_1^D, x_2^A) - f(x_1^A, x_2^A)]\}$
- $E_2 = \frac{1}{2}(E_{21} + E_{22}) \rightarrow \frac{1}{2}\{[f(x_1^D, x_2^D) - f(x_1^D, x_2^A)] + [f(x_1^A, x_2^D) - f(x_1^A, x_2^A)]\}$

Illustrative Application

Variable Name	Description	Monotone in probability of default
Response: $y =$ **default indicator**	$y = 1$ if account defaulted and $y = 0$ if it did not default	
Predictors		
x1 = avg bal cards *std*	Average monthly debt standardized: amount owed by applicant) on all of their credit cards over last 12 months	**N**
x2 = credit age *std*	Age in months of first credit product standardized: first credit cards, auto-loans, or mortgage obtained by the applicant	**Y = Decreasing**
x3 = pct over 50 uti	Percentage of open credit products (accounts) with over 50% utilization	**N**
x4 = tot balance *std*	Total debt standardized: amount owed by applicant on all of their credit products (credit cards, auto-loans, mortgages, etc.)	**N**
x5 = uti open card	Percentage of open credit cards with over 50% utilization	**N**
x6 = num acc 30d past due 12 months	Number of non-mortgage credit-product accounts by the applicants that are 30 or more days delinquent within last 12 months (Delinquent means minimum monthly payment not made)	**Y = Increasing**
x7 = num acc 60d past due 6 months	Number of non-mortgage credit-product accounts by the applicants that are 30 or more days delinquent within last 6 months	**Y = Increasing**
x8 = tot amount currently past due *log*	Total debt standardized: amount owed by applicant on all of their credit products – credit cards, auto-loans, mortgages, etc.	**Y = Increasing**
x9 = num credit inq 12 month	Number of credit inquiries in last 12 months. An inquiry occurs when the applicant's credit history is requested by a lender from the credit bureau. This occurs when a consumer applies for credit.	**Y = Increasing**
x10 = num credit card inq 24-month	Number of credit card inquiries in last 24 months. An inquiry occurs when the applicant's credit history is requested by a lender from the credit bureau. This occurs when a consumer applies for credit.	**Y = Increasing**

Simulated Data:
- 50,000 accounts
- Default or not in 18 months
- 10 predictors
- Distributions of predictors mimic bureau data

Fitted Model:
- Feedforward NN
- Constrained to be monotone in indicated variables

Correlation

High correlation among similar features

x1 = avg bal cards (std)
x2 = credit age std (flip)
x3 = pct over 50 uti
x4 = tot balance std
x5 = uti open card
x6 = num acc 30d past due 12 months
x7 = num acc 60d past due 6 months
x8 = tot amount currently past due (log)
x9 = num credit inquiries is past 12 months
x10 = num credit card inquiries in past 24-months

- Block correlation among similar features
- High-levels

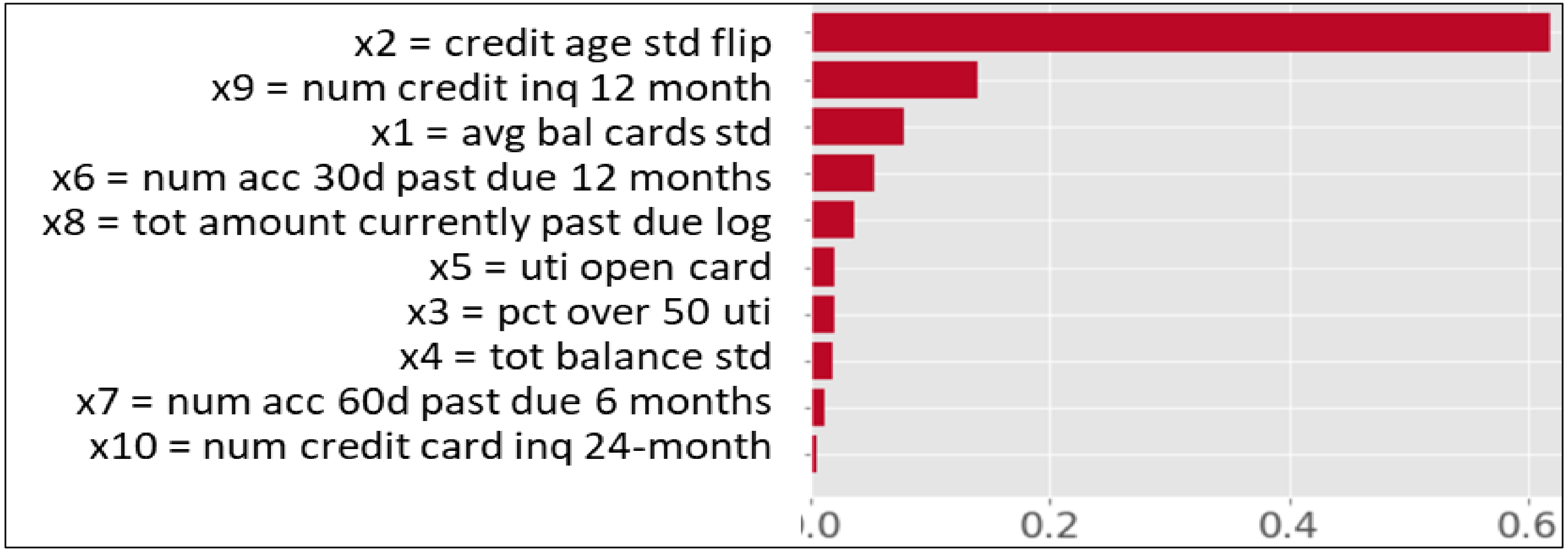
Training Monotone Neural Network

- Iterative algorithm: Fit with a penalty for monotonicity; certify; and iterate
- 50,00 accounts → training: 80%, validation and training: 10% each
- Final model: three hidden layers with dimensions [35, 15, 5]; learning rate (LR) = 0.001
- For comparison:
  - fitted unconstrained Feedforward Neural Network (FFNN) [23, 35, 15]; LR = 0.004

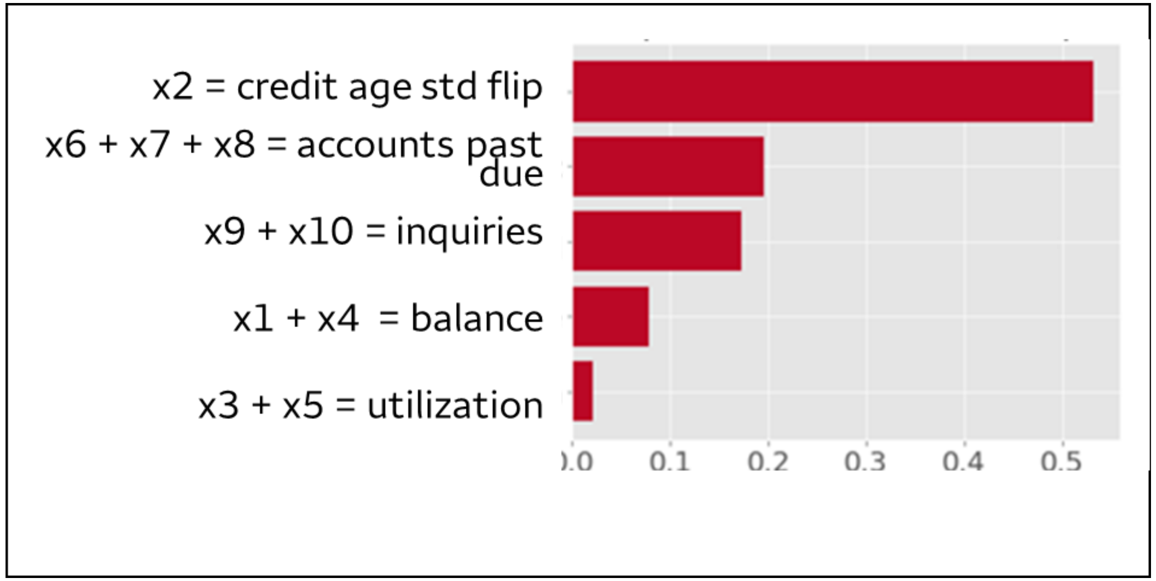
Training and Test AUCs for the Two Algorithms

Algorithm	Training AUC	Test AUC
FFNN	0.810	**0.787**
M-NN	0.807	**0.797**

Variable Importance for Mono-NN: All and Combined



Individual Variable Importance



Variable Importance for Combined

AA explanation: Decision rule – decline if $p(x) > 0.25$

Predictors	$\boldsymbol{x^A}$	$\boldsymbol{x_1^D}$	M-NN Attributions for $\boldsymbol{x_1^D}$
x1 = avg bal cards std	-0.006	0.674	0.112 (3.5%)
x2 = credit age std flip	-0.733	0.886	**1.928 (59.5%)**
x3 = pct over 50 uti	0.518	0.531	0.001 (0.0%)
x4 = tot balance std	-0.001	0.562	−0.008 (0.2%)
x5 = uti open card	0.501	0.577	0.012 (0.4%)
x6 = num acc 30d past due 12 months	0.000	0.000	0.0 (0.0%)
x7 = num acc 60d past due 6 months	0.000	0.000	0.0 (0.0%)
x8 = tot amount currently past due std	0.000	0.000	0.0 (0.0%)
x9 = num credit inq 12 month	0.000	3.000	**1.010 (31.2%)**
x10 = num credit inq 24 month	0.000	4.000	0.186 (5.7%)
$\widehat{\boldsymbol{p}}(\boldsymbol{x})$	0.016	0.294	
$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{logit}(\widehat{\boldsymbol{p}}(\boldsymbol{x}))$	−4.117	−0.876	

Modified to get combined explanation for groups of correlated predictors

Groups of predictors	M-NN Attributions for $\boldsymbol{x_1^D}$
balance	0.126 (3.9%)
credit age std flip	**1.925 (59.4%)**
utilization	0.018 (0.5%)
num acc	0.000 (0.0%)
num inq	**1.173 (36.2%)**

AA explanation: Decision rule – decline if $p(x) > 0.25$

Predictors	$\boldsymbol{x^A}$	$\boldsymbol{x_2^D}$	M-NN Attributions for $\boldsymbol{x_2^D}$
x1 = avg bal cards std	-0.006	0.519	0.028 (0.5%)
x2 = credit age std flip	-0.733	0.431	**1.565 (26.5%)**
x3 = pct over 50 uti	0.518	0.522	-0.001 (0.0%)
x4 = tot balance std	-0.001	1.968	-0.201 (−3.4%)
x5 = uti open card	0.501	0.525	-0.024 (−0.4%)
x6 = num acc 30d past due 12 months	0.000	4.000	**1.850 (31.3%)**
x7 = num acc 60d past due 6 months	0.000	2.000	**0.984 (16.6%)**
x8 = tot amount currently past due std	0.000	4.379	**1.712 (28.9%)**
x9 = num credit inq 12 month	0.000	0.000	0.0 (0.0%)
x10 = num credit inq 24 month	0.000	0.000	0.0 (0.0%)
$\boldsymbol{\hat{p}(x)}$	0.016	0.858	
$\boldsymbol{f(x) = logit(\hat{p}(x))}$	−4.117	1.797	

Modified to get combined explanation for groups of correlated predictors

Groups of predictors	M-NN Attributions for $\boldsymbol{x_2^D}$
balance	-0.328 (-5.5%)
credit age std flip	**1.785 (30.2%)**
utilization	-0.018 (-0.3%)
past due	**4.476 (75.7%)**
num inq	0.000 (0.0%)

Issues with post hoc explanation

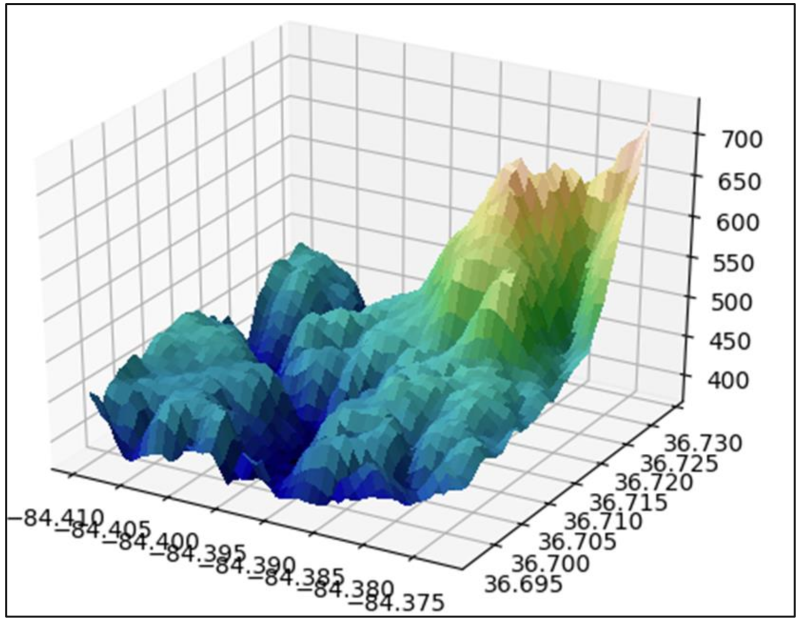
- **Most post-hoc tools** for studying input-output relationships are **lower-dimensional summaries**
  - **Limited** in **ability** to **characterize complex models** that may have different local behaviors

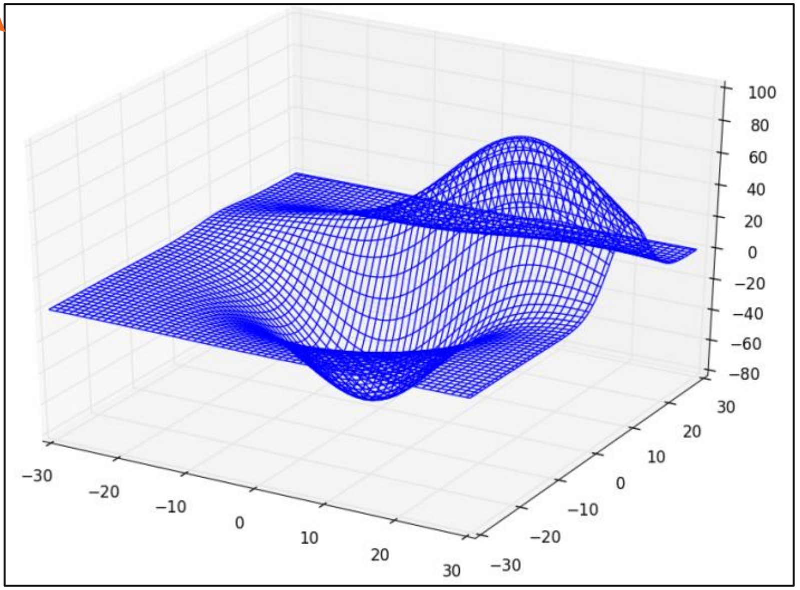
- ML algorithms: function-fitting vs modeling
  - High-dimensional ML – can do very good function fitting with large samples
  - What is a **role of a model**?

- Correlation causes havoc!
  - Model identifiability

$$f(x_1, x_2) = \beta_1 x_1 + \beta_2 x_2 + \beta_{12}\, x_1 x_2$$

- The view in ML is to throw as many predictors as possible into the mix and automate model building
- Bound to be high correlation among some predictors





Inherently interpretable models

- Goals and challenges of complex ML models
  - **Extract as much predictive performance** as possible
  - **No emphasis on interpretation** → lots of variables, complex relationships and interactions
  - No analytic expressions → **rely on low dimensional summaries** → don’t present the full picture
- Emerging view:
  - **Low-order functional** (nonparametric) **models** are **adequate** in most of our applications
    → tabular data in banking
  - Directly interpretable
  - Reversing emphasis on complex modeling
    → **trade-off**: improvements in predictive performance vs interpretation

Examples of "Low Order" Models

- **Functional ANOVA Models:**

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

  - FANOVA models with low-order interactions are adequate for many of our applications
  - **Focus** on models with **functional main effects and second order interactions**
  - Stone (1994); Wahba and her students (see Gu, 2013)

    → use **splines** to estimate low-order functional effects non-parametrically
  - **Not scalable** to large numbers of observations and predictors
  - Recent approaches

    → use **ML architecture and optimization algorithms** to develop fast algorithms

FANOVA framework

$$f(\boldsymbol{x}) = g_0 + \sum_j g_j(x_j) + \sum_{j<k} g_{jk}(x_j, x_k)$$

- Model made up of mean g_0, **main effects $g_j(x_j)$, two-factor interactions $g_{jk}(x_j, x_k)$**
- **Interpretability**
  - Fitted model is **additive,** effects are enforced to be **orthogonal**
  - Components can be **easily visualized** and **interpreted directly**
  - Regularization or other techniques used to keep model parsimonious

- Two state-of-the-art ML algorithms for fitting these models:
  - **Explainable Boosting Machine** (Nori, et al. 2019) → boosted trees (EBM)
  - **GAMI Neural Networks** (Yang, Zhang and Sudjianto, 2021) → specialized NNs (GAMI-NET)
  - **GAMI-Linear-Tree** (Hu, Chen, and Nair, 2022) → boosted linear trees (GAMI-Lin-T)

EBM: Nori, Jenkins, Koch and Caruana (2019) arXiv: 1909.09223
GAMI-Net.: Yang, Zhang and Sudjianto (2021, arXiv: 2003.07132
GAMI-Lin-TL Hu, L., Chen, J. and Nair, V. (2022), arXiv:2207.06950

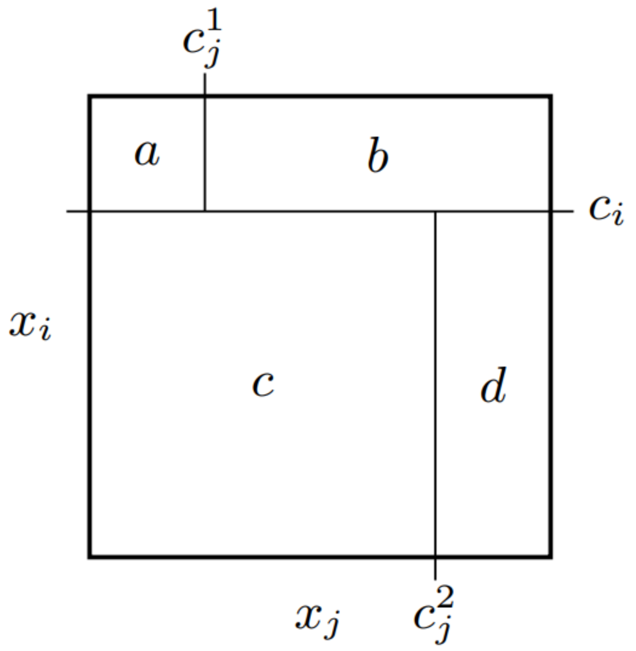
Explainable boosting machine

- **EBM** – Boosted-tree algorithm by Microsoft group (Lou, et al. 2013)

$$f(\boldsymbol{x}) = g_0 + \sum g_j(x_j) + \sum g_{jk}(x_j, x_k)$$

  – Microsoft InterpretML (Nori, et al. 2019)
  – fast implementation in C++ and Python

- **Multi-stage model training :**
  – 1: fit functional main effects non-parametrically
    – Shallow tree boosting with splits on the same variable for capturing a non-linear main effect
  – 2: fit pairwise interactions on residuals:
    a. Detect interactions using **FAST** algorithm
    b. For each interaction (x_j, x_k), fit function $g_{jk}(x_j, x_k)$ non-parametrically using a tree with depth two: 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)
    c. Iteratively fit all the detected interactions until convergence



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

Explainable boosting machine: Example

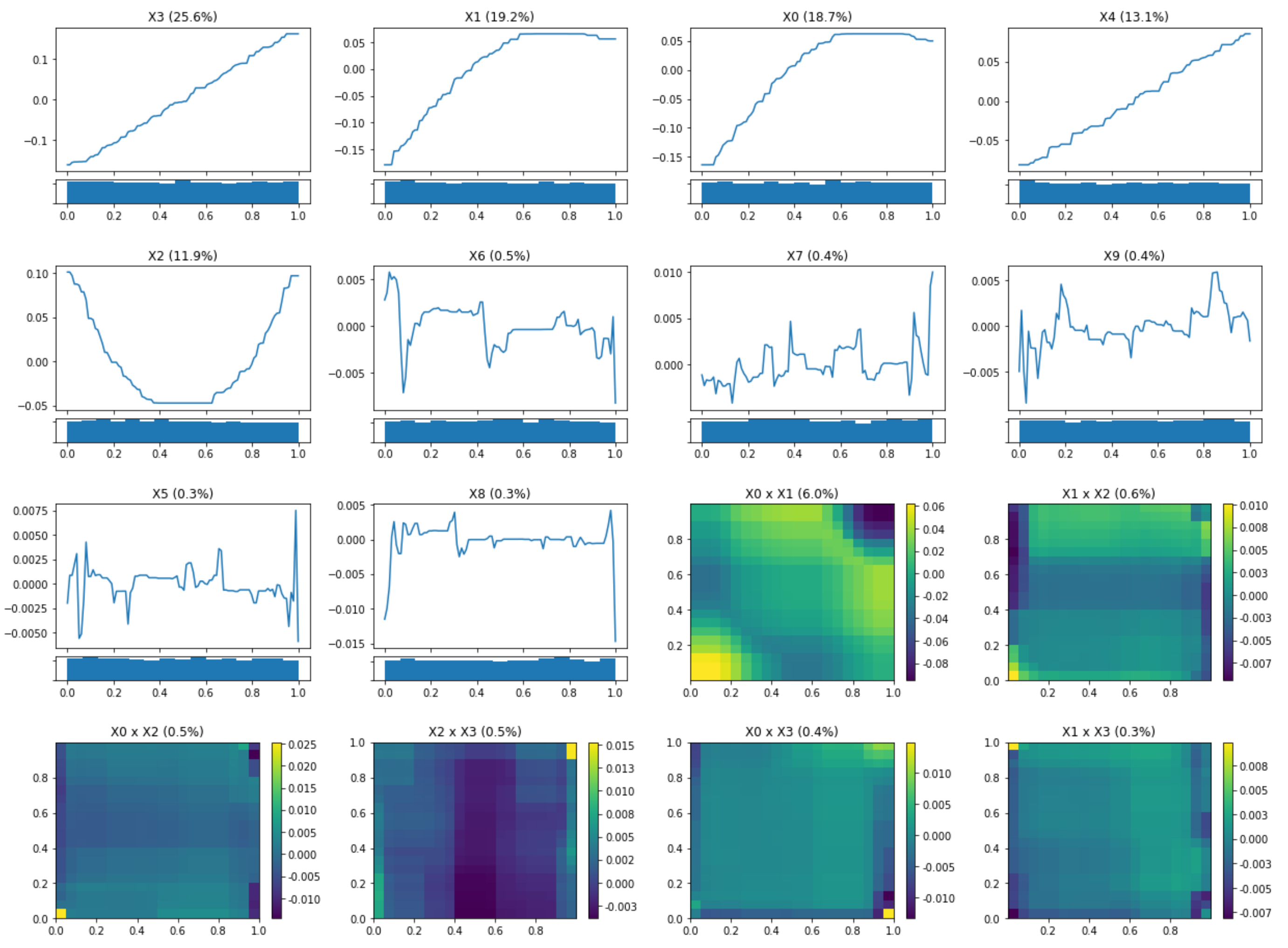
Friedman1 simulated data:

- sklearn.datasets.make_friedman1 n_samples=10000, n_features=10, and noise=0.1.
- 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 $\boldsymbol{x_0} \sim \boldsymbol{x_4}$

EBM Output with Test RMSE = 0.0284 and R2 = 97.39%



GAMI-Net

- NN-based algorithm for non-parametrically fitting

$$f(\boldsymbol{x}) = g_0 + \sum g_j(x_j) + \sum g_{jk}(x_j, x_k)$$

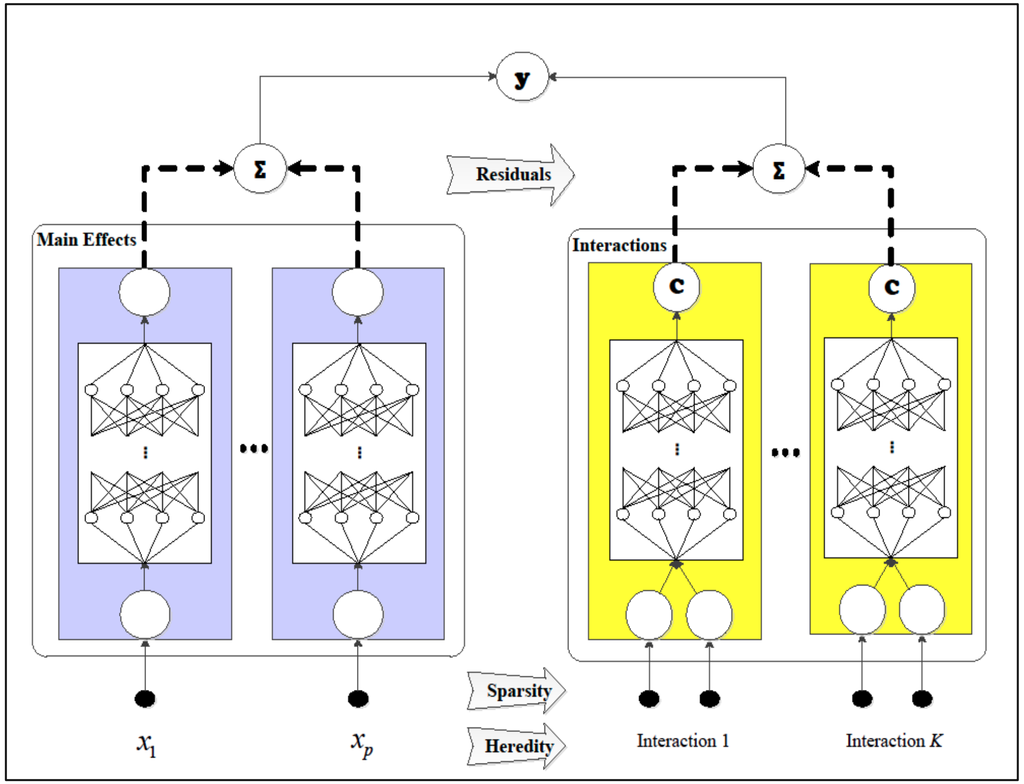
- **Multi-stage training algorithm:**

  1: estimate $\{g_j(x_j)\}$ → train main-effect subnets and **prune** small main effects

  2: estimate $\{g_{jk}(x_j, x_k)\}$ → compute residuals from main effects and train pairwise interaction nets

  - Select candidate interactions using heredity constraint
  - Evaluate their scores (by FAST) and select top-K interactions;
  - Train the selected two-way interaction subnets;
  - Prune small interactions

  3: retrain main effects and interactions simultaneously



Yang, Zhang and Sudjianto (2021, Pattern Recognition): GAMI-Net. arXiv: 2003.07132

Diagnostics: Effect importance and feature importance

- Each **effect importance** (before normalization) is given by

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

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

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

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

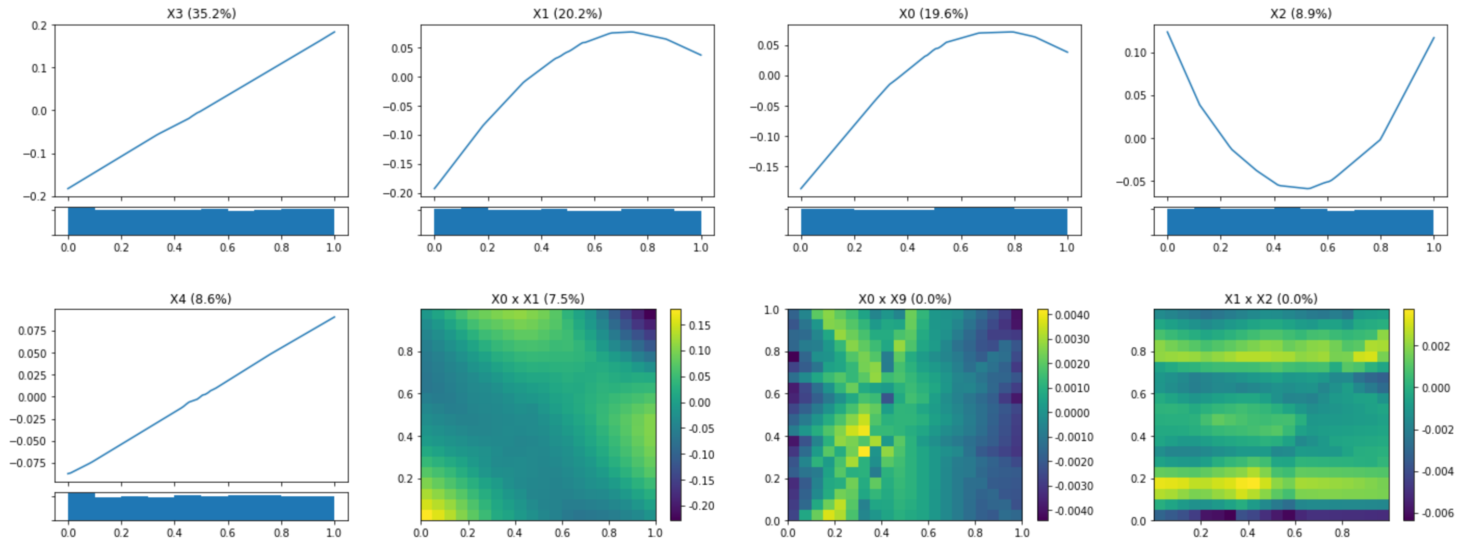
GAMI-Net: Example

Friedman1 data:

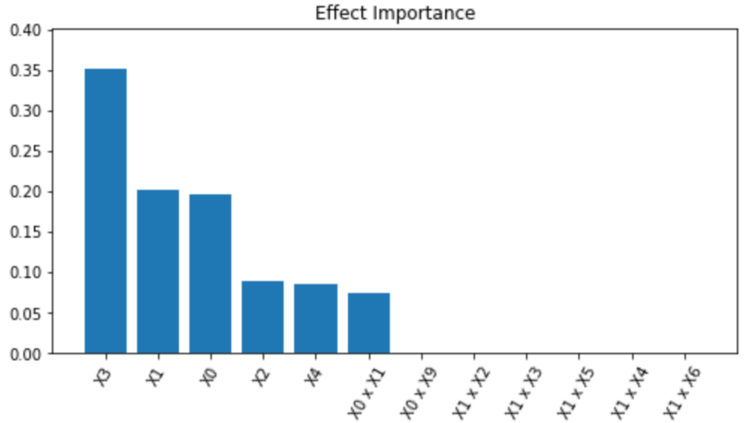
$$y(\boldsymbol{x}) = 10\sin(\pi x_0 x_1) + 20(x_2 - 0.5)^2 + 20x_3 + 10x_4 + \epsilon$$

Same data generated as for EBM example.

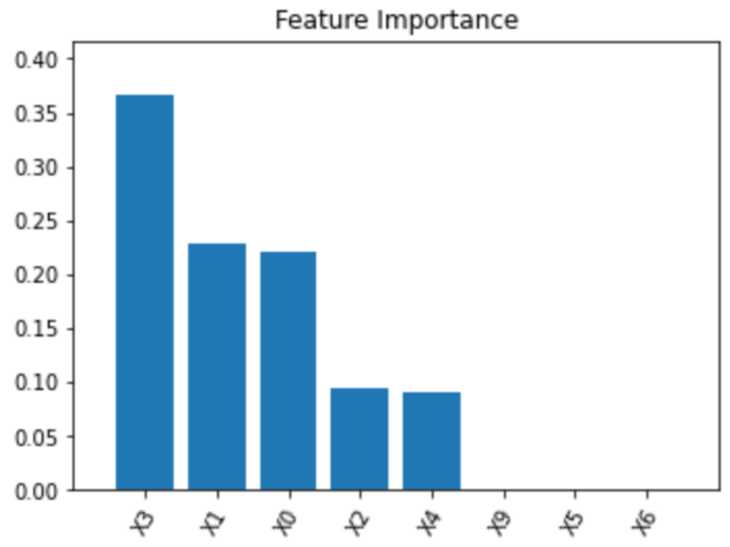
GAMI-Net Output with Test RMSE = 0.0058 and R2 = 99.89%




```
model_gaminet.show_effect_importance()
```




```
model_gaminet.show_feature_importance()
```



Summary

- Explainability & Interpretability
  - Post-hoc techniques
  - Inherently interpretable algorithms

- PI-ML Demo