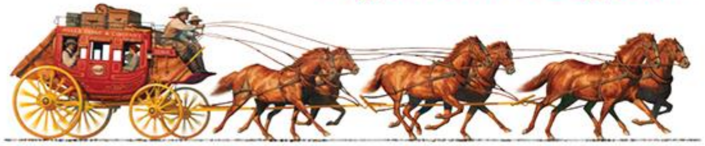
Interpretable Machine Learning with Applications to Banking

Linwei Hu

Advanced Technologies for Modeling, Corporate Model Risk
Wells Fargo

October 26, 2018

Together we'll go far



Agenda

- Machine Learning in Banking
- Machine Learning Model Interpretation
  - Locally Interpretable Model (focus of this talk)
  - Global Diagnostics
  - Explainable Neural Networks
- Example

AI/ML Applications in Banking

Traditionally: Statistical and econometrics techniques used for
- Model development: Core models and challenge models
- Model validation: Benchmarking and comparisons
  - Examples: Credit decision, PD and Revenue modeling, Fraud detection, Fair lending, etc.

Emerging challenges:
- Large data sets (n and p) → deficiency of traditional approaches
- New applications: Text analytics, Natural Language Processing, etc.
  - Examples: Chat bots, complaint analysis, customer assistance, etc.

Rapid adoption of Artificial Intelligence (AI)/Machine Learning (ML) across Financial Institutions

- Address new challenges
- Improve: business decisions, customer experiences and risk management

Examples: Credit Risk Models

Stress Testing

- Predict expected losses: PD, LGD, EAD, (PD*LGD*EAD)
- Predict under multiple time horizons and various micro-economic variables

Statistical Models

- Survival analysis
- Regression: LGD, etc.
- Semi-parametric Models
  - Varying coefficient models

Machine Learning

- Random Forests
- Gradient Boosting Machines
- Neural Nets: LSTM

Examples of Financial Crime Models

- **Anti Money Laundering**: to prevent and detect potential money laundering activities
- **Fraud Detection**: to prevent and detect fraudulent activities

Statistical Model

- Rule-Based System
- Clustering

Machine Learning

- One class SVM
- Supervised and semi-supervised machine learning
- PU learning

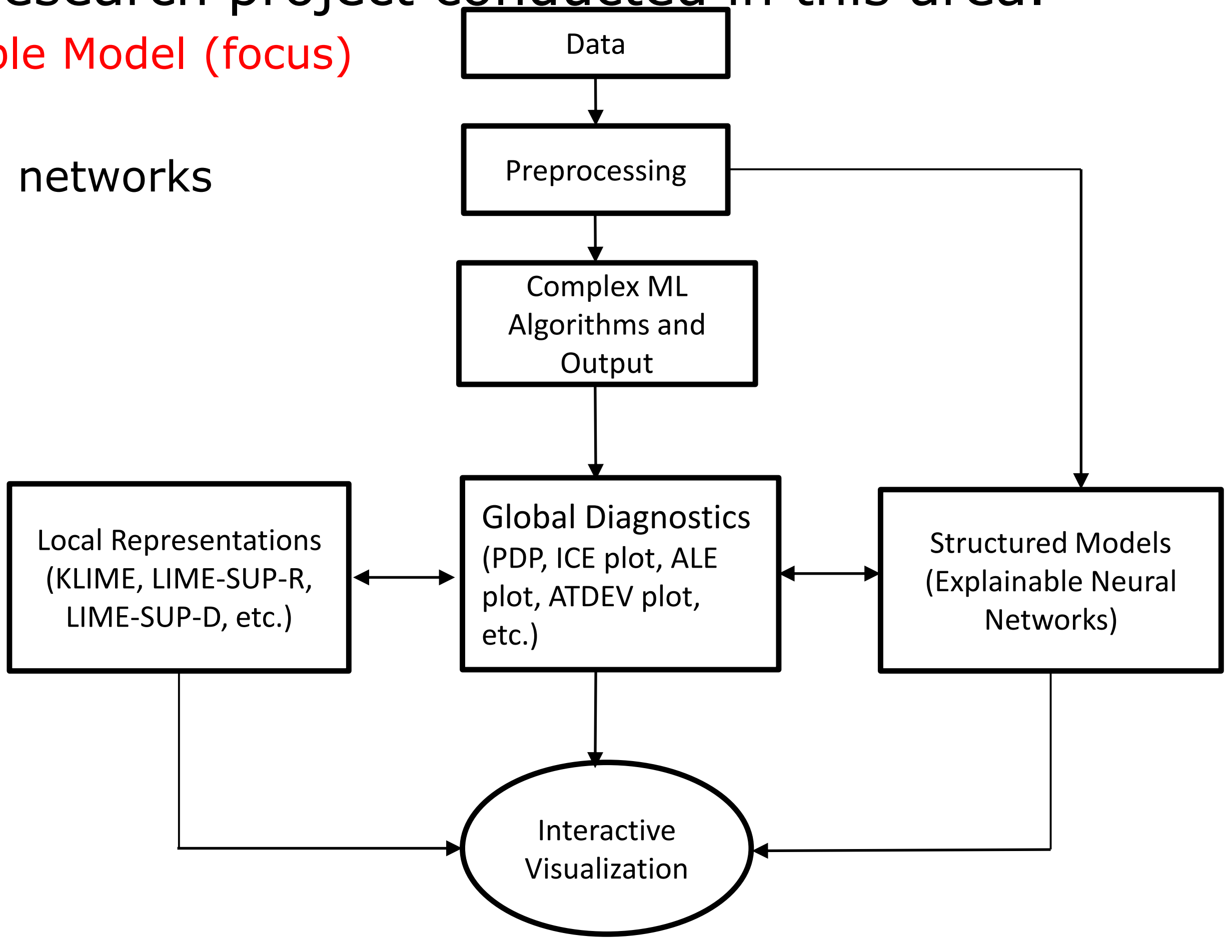
Challenges:

Predictive performance + “automation” come at a cost

- Models are complex and hard to interpret
  - No analytical expressions
- Potential problems with multi-collinearity
- Ambiguities in attribution → credit scoring
- May not conform to subject matter knowledge
  - Inclusion of key variables, monotonicity constraints
- Tuning of “hyper-parameters” is complex and computationally intensive
- Tendency to put too much faith in “automated” algorithms → in fact, now they deserve more scrutiny

Machine Learning Model Interpretation

- Machine learning interpretation is an active research area now. In our team (Advanced Technologies for Modeling), we have a couple of research project conducted in this area.
  - Locally Interpretable Model (focus)
  - Global Diagnostics
  - Explainable Neural networks

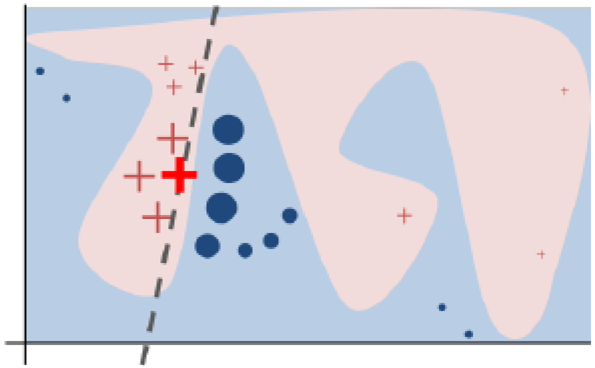


Locally Interpretable Model

- Local interpretation is aimed at interpreting the relationship between input and output over local region, with the idea that a simple parametric model may be used to approximate the input-output relationship, and local variable importance and input-output relationships are easily interpretable from the simple local model.
- Related tools include:
  - LIME (Ribeiro et al. 2016)
  - KLIME (H2o)
  - LIME-SUP (our approach)

LIME

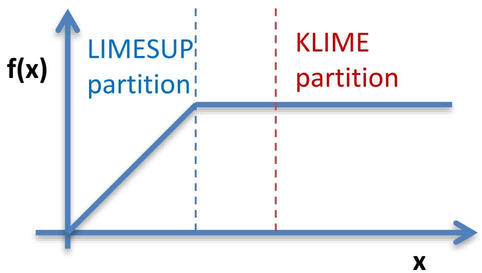
- LIME (*Local* Interpretable *Model-Agnostic Explanations*) is perhaps the first local interpretation method, proposed in Ribeiro et al. (2016).
- The idea is to approximate the model around a given instance/observation using a linear model in order to explain the prediction:
  - Simulate new instances
  - Predict on the new instances using the machine learning model $f(\boldsymbol{x})$
  - Pick a kernel and fit a linear model using the kernel as weight; penalize the complexity of the linear model, for example, fit ridge regression.



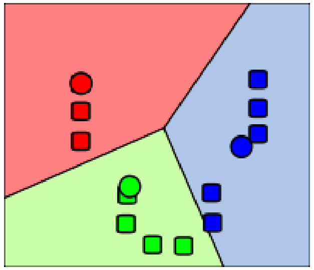
- Available in python (lime package) and R (lime package)

KLIME

- KLIME is a variant of LIME proposed in H2o *Driverless AI*. It divides the input space into regions and fit a linear model in each region.
  - Cluster the input space using a K-Means algorithm
  - Fit a linear model to the machine learning prediction $f(x)$ in each cluster
  - The number of clusters is chosen by maximizing Rsquare
- KLIME can be used as a surrogate model (a less accurate but more interpretable substitute of the machine learning model). However, it has some disadvantages:
  - the unsupervised partitioning approaches can be unstable, yielding different partitions with different initial locations.
  - the unsupervised partitioning does not incorporate any model information which seems critical to preserving the underlying model structure. It is less accurate
  - K-means partitions the input space according to the Voronoi diagrams, it is less intuitive in business environment where modelers are more used to rectangle partitioning (segmentation).



KLIME vs LIMESUP partition

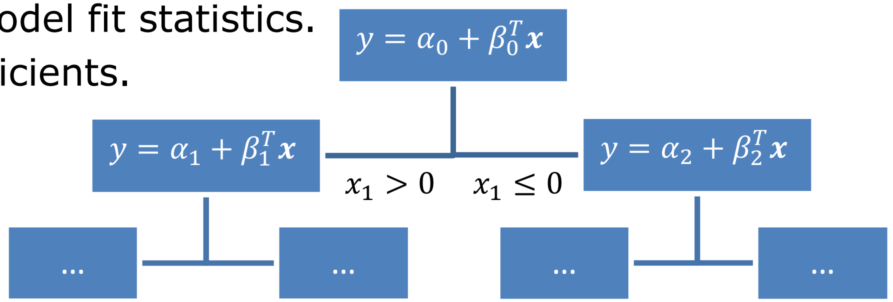

K-means clustering

LIME-SUP

- LIME-SUP is an internal local interpretation method developed by our team.
- Similar to KLIME: it also partitions the X-space and fits a simple model in each partition. The key difference from KLIME: it is a supervised partitioning method using information from the machine learning model.
- The goal is to use supervised partitioning to achieve a more stable, more accurate and more interpretable surrogate model than KLIME.
- There are two implementations of LIME-SUP. One uses model based tree (LIME-SUP-R) and the other uses partial derivatives (LIME-SUP-D). The two have similar principal but different focus, LIMESUP-R fits better but is more computationally expensive.

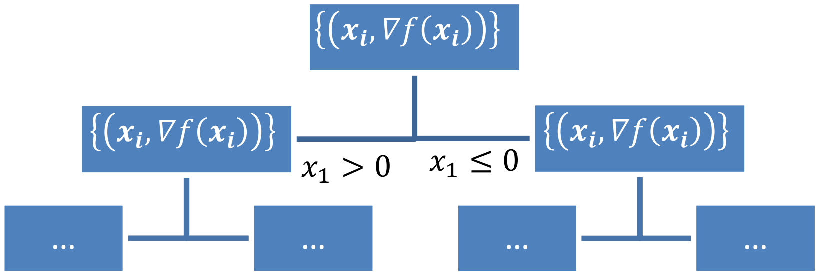
LIME-SUP-R

- LIME-SUP-R uses model based tree. Model based tree differs from traditional classification and regress tree (CART) in that it fits a model instead of constant in each tree node. At each node, it works as follows:
  - Fit a parent model to the node
  - Split the node into two child nodes, fit separate child models. The best split is found so that the combined model fit for the two child models is maximized.
  - Keep splitting until certain stopping criterion is met (depth, leaf node size, etc)
- LIMESUP-R partitions the X-space in a supervised manner by utilizing machine learning model predictions. On the high level it works as follows:
  - Predict on the data using the machine learning model $f(\boldsymbol{x})$. Predictions can be predicted mean (continuous response) or logodds (binary response).
  - For a specified form of parametric model (say linear regression), fit a model based tree to the predictions using machine learning model predictors.
  - Prune the tree using appropriate model fit statistics.
  - Check model fit, plot tree and coefficients.



LIME-SUP-D

- LIME-SUP-D uses partial derivatives $\nabla f(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$ from the ML model $f(\boldsymbol{x})$, derivatives are the coefficients if we fit a linear model to $f(\boldsymbol{x})$ in the neighborhood of $\boldsymbol{x}$.
- We can partition the X-space by grouping the derivatives that are close, since similar derivatives in a region indicates a linear model can fit well in that region.
- On the high level it works as follows:
  - Compute the partial derivatives $\nabla f(\boldsymbol{x})$. The derivatives can be computed using a neural network surrogate model.
  - Fit a regression tree to the multi-dimensional partial derivatives using machine learning predictors.
  - Prune the tree using appropriate model fit statistics.
  - Check model fit, plot tree and coefficients for interpretation.

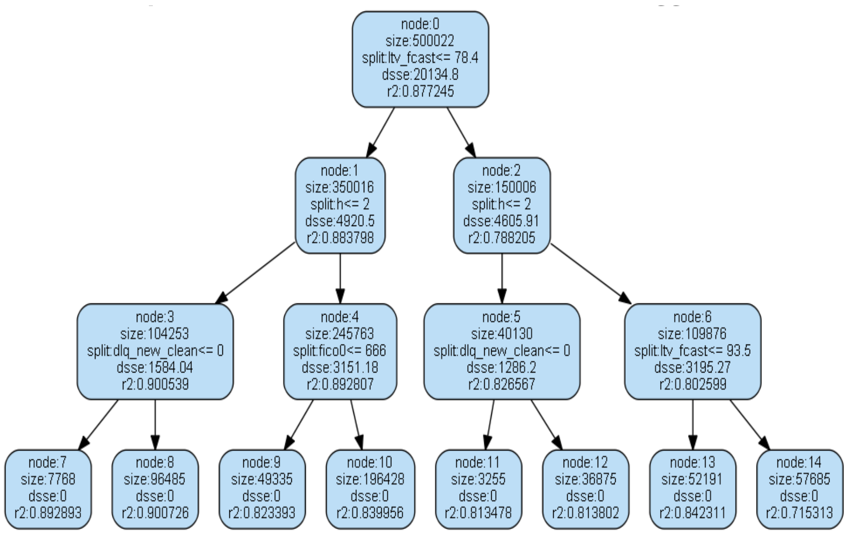


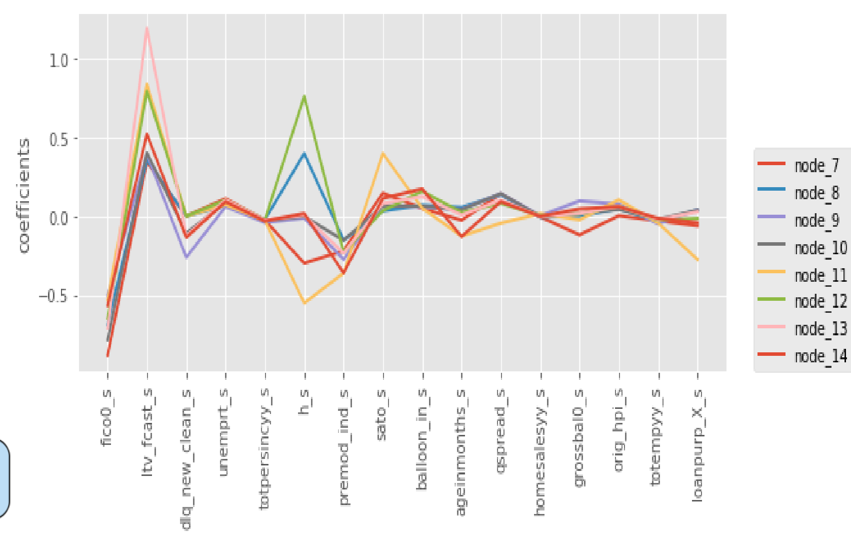
Example

- We use a data with 50 predictor variables and 1 million observations. The response is a binary indicator (default).
- A gradient boosting model is trained using the 50 variables. Then the top 20 excluding 4 variables (due to correlation), are selected to run LIME-SUP and KLIME. So in total there are 16 variables.
- The logodds/logits and partial derivatives of the logits are computed and used for LIMESUP-R and LIMESUP-D.

Example

- Figure below shows the tree structure and the coefficients in the terminal nodes, for LIMESUP-R with depth = 3.
- The strongest patterns in the coefficients exist for ltv_fcast and horizon. Combining the tree structure and the coefficient values, we can see
  - The coefficients for ltv_fcast peaks for middle range ltv (node 13) and are low for low (node 7, 8, 9, 10) and high ltv (node 14).
  - The coefficients for horizon is positive for non-delinquent accounts (dlq_new_clean = 1, node 8 & 12) and negative otherwise (node 7 & 11). The overall effect of horizon is close to 0. This explains the interaction effects.
- The coefficient for fico is also smaller for high ltv (node 14, red curve),





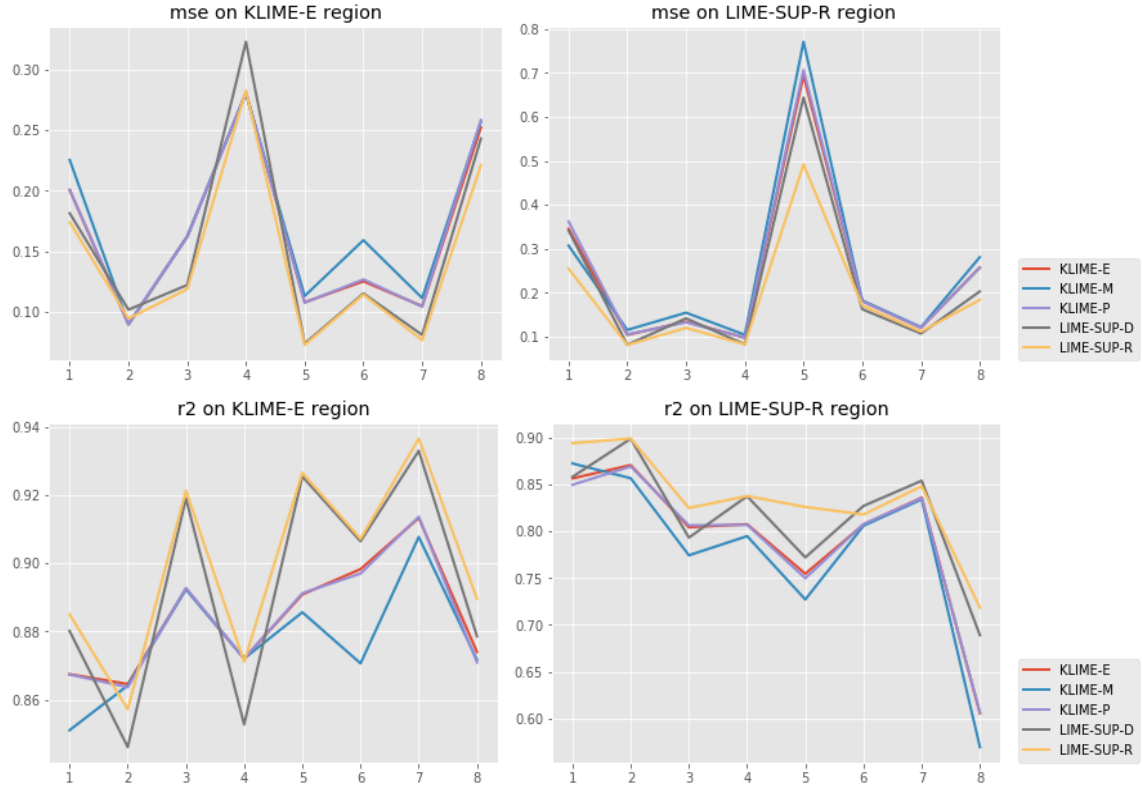
Example

- Similarly we fit KLIME with 8 clusters.
- Table below shows the MSE, and Rsquare for the 5 methods.
- LIME-SUP is better than KLIME, although the difference is not as striking in this case. Besides that, we see LIME-SUP-R fits slightly better than LIME-SUP-D, which is expected.

	LIME-SUP-R	LIME-SUP-D	KLIME-E	KLIME-M	KLIME-P
MSE	0.113	0.118	0.137	0.147	0.138
R^2	0.927	0.924	0.911	0.905	0.911

Example

- Figure below provides a different view of the comparisons: values of MSE and R^2 computed within each of eight local regions.
- The conclusions are similar as before. LIME-SUP does better on all local regions, except LIME-SUP-D has larger mse than KLIME on the KLIME-E regions 2, 4.



Reference

- Liu, Chen, Vaughan, Nair, Sudjianto (2018), Model Interpretation: A Unified Derivative-based Framework for Nonparametric Regression and Supervised Machine Learning, arXiv:1808.07216
- Hu, Chen, Nair, Sudjianto (2018), Locally Interpretable Models and Effects based on Supervised Partitioning (LIME-SUP), arXiv:1806.00663
- Vaughan, Sudjianto, Brahimi, Chen, Nair (2018), Explainable Neural Networks based on Additive Index Models, arXiv:1806.01933