Increasing Trust and Understanding in Machine Learning with Model Debugging

© Patrick Hall*

June 11, 2020

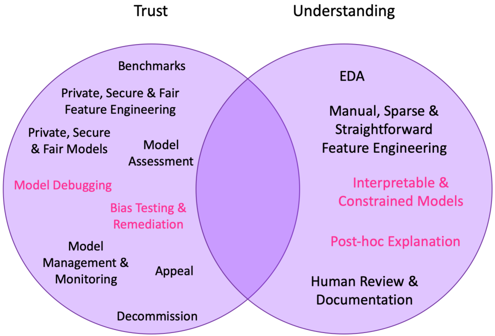
Contents

What is Model Debugging?

- Model debugging is an emergent discipline focused on discovering and remediating errors in the internal mechanisms and outputs of machine learning models.[†]
- Model debugging attempts to test machine learning models like code (because the models are code).
- Model debugging promotes trust directly and enhances interpretability as a side-effect.
- Model debugging is similar to regression diagnostics, but for nonlinear machine learning models.

[†] See https://debug-ml-iclr2019.github.io/ for numerous model debugging approaches.

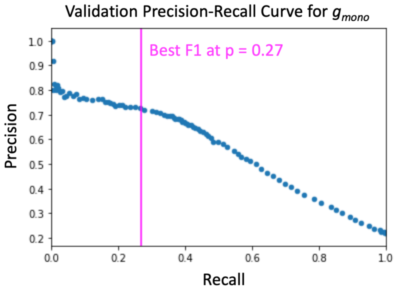
Trust and Understanding



Trust and understanding in machine learning are different but complementary goals, and they are technically feasible *today*.

Why Debug?

- Constrained, monotonic GBM probability of default (PD) classifier, g_{mono}.
- Grid search over hundreds of models.
- Best model selected by validation-based early stopping.
- Seemingly well-regularized (row and column sampling, explicit specification of L1 and L2 penalties).
- No evidence of over- or under-fitting.
- Better validation logloss than benchmark GLM.
- Decision threshold selected by maximization of F1 statistic.
- BUT traditional assessment can be inadequate ...

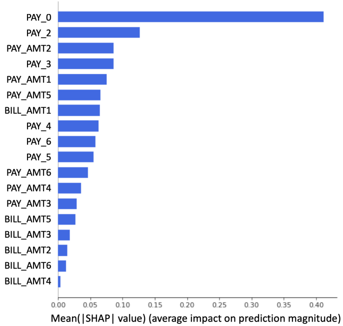


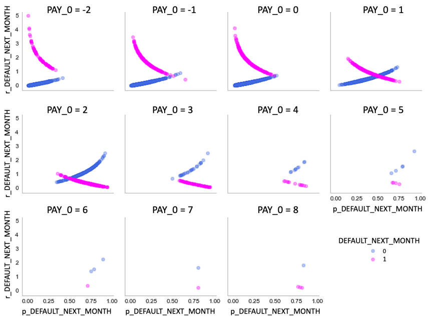
Validation Confusion Matrix at Threshold:

	Actual: 1	Actual: 0
Predicted: 1	1159	827
Predicted: 0	1064	6004

Why Debug?

Machine learning models can be **inaccurate**.





g_{mono} PD classifier over-emphasizes the most important feature, a customer's most recent repayment status, PAY_0.

g_{mono} also struggles to predict default for favorable statuses, $-2 \leq \texttt{PAY\_0} < 2$, and often cannot predict on-time payment when recent payments are late, PAY_0 ≥ 2.

Why Debug?

Machine learning models can perpetuate **sociological biases** [1].

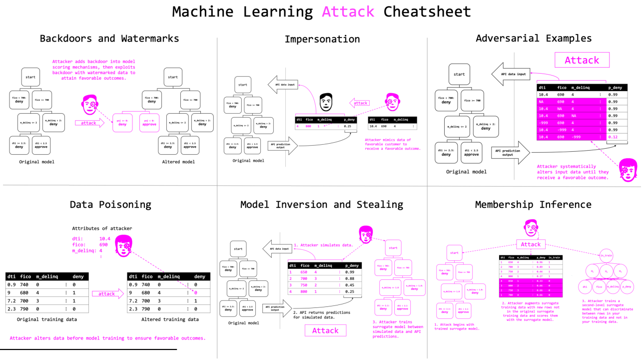
	Adverse Impact Disparity	Accuracy Disparity	True Positive Rate Disparity	Precision Disparity	Specificity Disparity
single	0.885	1.029	0.988	1.008	1.025
divorced	1.014	0.932	0.809	0.806	0.958
other	0.262	1.123	0.62	1.854	1.169

Group disparity metrics are out-of-range for g_{mono} across different marital statuses.

This *does not* address localized instances of discrimination.

Why Debug?

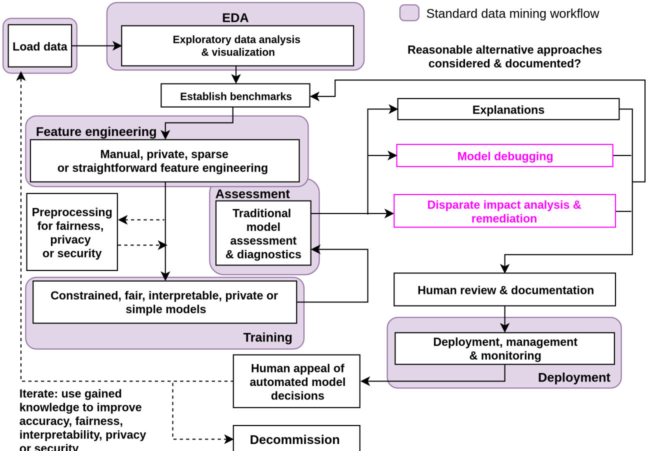
Machine learning models can have **security vulnerabilities** [2], [5], [6].‡



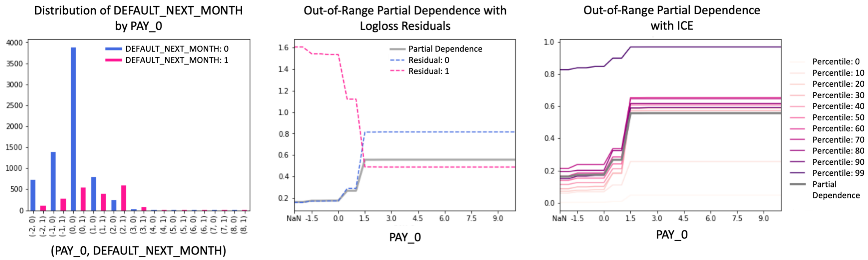
‡See https://github.com/jphall663/secure_ML_ideas for full size image and more information.

How to Debug Models?

As part of a holistic, low-risk approach to machine learning.[§]

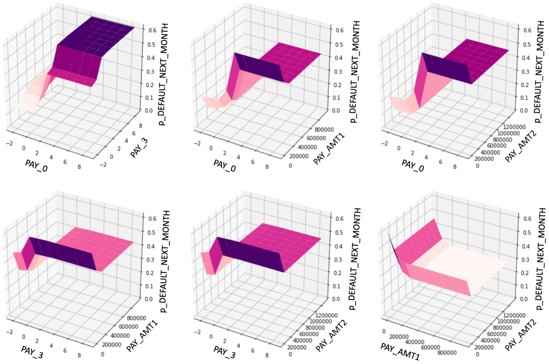


Sensitivity Analysis: Partial Dependence and ICE



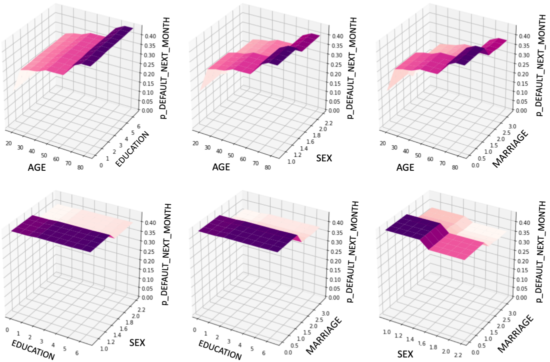
- Training data is very sparse for PAY_0 > 2.
- Residuals of partial dependence confirm over-emphasis on PAY_0.
- ICE curves indicate that partial dependence is likely trustworthy and empirically confirm monotonicity, but also expose adversarial attack vulnerabilities.
- Partial dependence and ICE indicate g_{mono} likely learned very little for PAY_0 ≥ 2.
- PAY_0 $=$ `missing` gives lowest probability of default.

Sensitivity Analysis: Search for Adversarial Examples



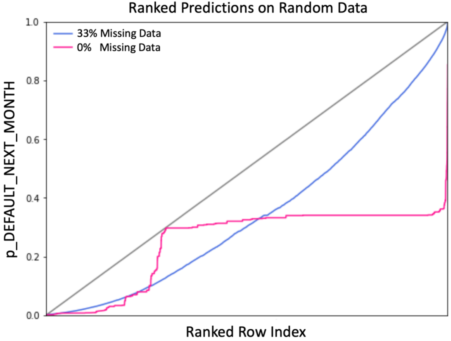
Adversary search confirms multiple avenues of attack and exposes a potential flaw in $g_{\textbf{mono}}$ scoring logic: default is predicted for customer's who make payments above their credit limit. (Try the cleverhans package for finding adversarial examples.)

Sensitivity Analysis: Search for Adversarial Examples



g_{mono} appears to prefer younger, unmarried customers, which should be investigated further with disparate impact analysis - see slide 7 - and could expose the lender to impersonation attacks. (Try the AIF360, aequitas, or themis packages for disparate impact audits.)

Sensitivity Analysis: Random Attacks



- In general, random attacks are a viable method to identify software bugs in machine learning pipelines. **(Start here if you don't know where to start.)**
- Random data can apparently elicit all probabilities $\in [0, 1]$ from g_{mono}.
- Around the decision threshold, lower probabilities can be attained simply by injecting missing values, yet another vulnerability to adversarial attack.

Residual Analysis: Disparate Accuracy and Errors

Error Metrics for PAY_0

PAY_0	Prevalence	Accuracy	True Positive Rate	Precision	Specificity	Negative Predicted Value	False Positive Rate	False Discovery Rate	False Negative Rate	False Omissions Rate
-2	0.124	0.864	0.099	0.333	0.972	0.884	0.028	0.667	0.901	0.116
-1	0.168	0.816	0.206	0.406	0.939	0.854	0.061	0.594	0.794	0.146
0	0.121	0.867	0.107	0.341	0.972	0.888	0.028	0.659	0.893	0.112
1	0.325	0.491	0.903	0.381	0.292	0.862	0.708	0.619	0.097	0.138
2	0.709	0.709	1	0.709	0	0.5	1	0.291	0	0.5
3	0.748	0.748	1	0.748	0	0.5	1	0.252	0	0.5
4	0.571	0.571	1	0.571	0	0.5	1	0.429	0	0.5
5	0.444	0.444	1	0.444	0	0.5	1	0.556	0	0.5
6	0.25	0.25	1	0.25	0	0.5	1	0.75	0	0.5
7	0.5	0.5	1	0.5	0	0.5	1	0.5	0	0.5
8	0.75	0.75	1	0.75	0	0.5	1	0.25	0	0.5

Error Metrics for SEX

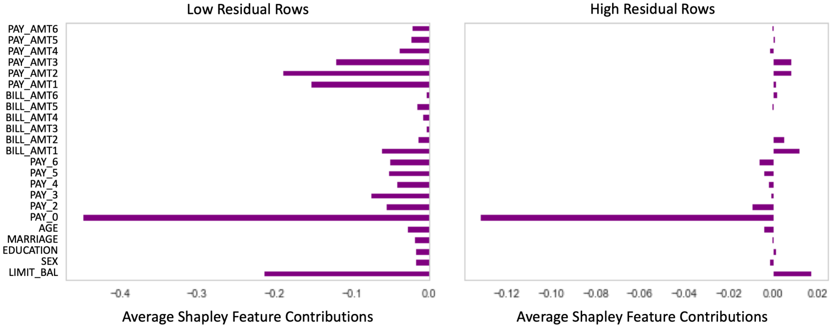
SEX	Prevalence	Accuracy	True Positive Rate	Precision	Specificity	Negative Predicted Value	False Positive Rate	False Discovery Rate	False Negative Rate	False Omissions Rate
Male	0.235	0.782	0.626	0.531	0.83	0.879	0.17	0.469	0.374	0.121
Female	0.209	0.797	0.552	0.514	0.862	0.879	0.138	0.486	0.448	0.121

For PAY_0:

- Notable change in accuracy and error characteristics for PAY_0 ≥ 2.
- 100% false omission rate for PAY_0 ≥ 2. (Every prediction of non-default is incorrect!)

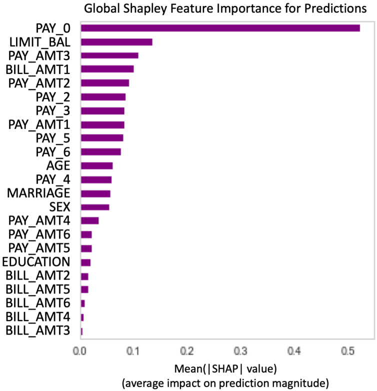
For SEX, accuracy and error characteristics vary little across individuals represented in the training data. Non-discrimination should be confirmed by disparate impact analysis.

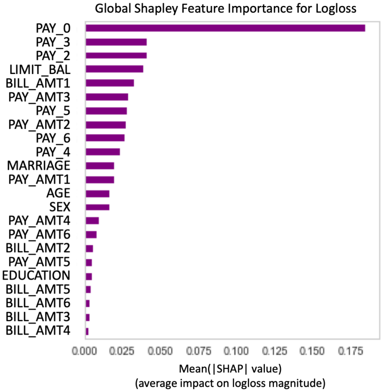
Residual Analysis: Mean Local Feature Contributions



Exact local Shapley feature contributions [4], which are available at scoring time for unlabeled data, are noticeably different for low and high residual predictions. (Both monotonicity constraints and Shapley values are available in h2o-3 and XGBoost.)

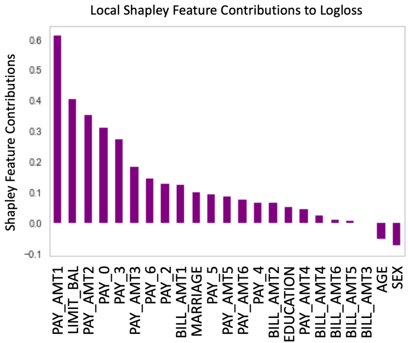
Residual Analysis: Non-Robust Features





Globally important features PAY_3 and PAY_2 are more important, on average, to the loss than to the predictions. (Shapley contributions to XGBoost logloss can be calculated using the shap package. This is a **time-consuming** calculation.)

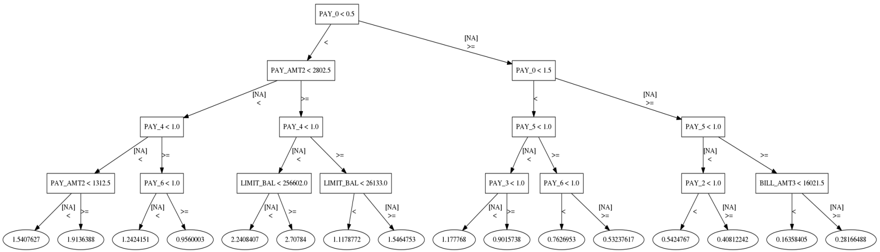
Residual Analysis: Local Contributions to Logloss



Exact, local feature contributions to logloss can be calculated, enabling ranking of features contributing to logloss residuals for **each prediction**.

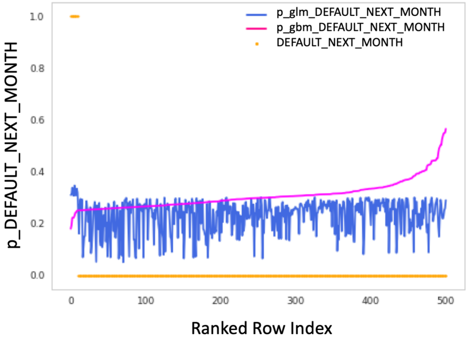
Residual Analysis: Modeling Residuals

Decision tree model of g_{mono} DEFAULT_NEXT_MONTH $= 1$ logloss residuals with 3-fold CV MSE $= 0.0070$ and $R^2 = 0.8871$.



This tree encodes rules describing when g_{mono} is probably wrong.

Benchmark Models: Compare to Linear Models



For a range of probabilities $\in (\sim 0.2, \sim 0.6)$, g_{mono} displays exactly incorrect prediction behavior as compared to a benchmark GLM.

Remediation: for g_{mono}

- **Over-emphasis of PAY_0**:
  - Engineer features for payment trends or stability.
  - Missing value injection during training or scoring.
- **Sparsity of PAY_0 > 2 training data**: Increase observation weights.
- **Payments $\geq$ credit limit**: Scoring-time model assertion [3].
- **Disparate impact**: Adversarial de-biasing [7] or model selection by minimal disparate impact.
- **Security vulnerabilities**: API throttling, authentication, real-time model monitoring.
- **Large logloss importance**: Evaluate dropping non-robust features.
- **Poor accuracy vs. benchmark GLM**: Blend g_{mono} and GLM for probabilities $\in (\sim 0.2, \sim 0.6)$.
- **Miscellaneous strategies**:
  - Local feature importance and decision tree rules can indicate additional scoring-time model assertions, e.g. alternate treatment for locally non-robust features in known high-residual ranges of the learned response function.
  - Incorporate local feature contributions to logloss into training or scoring processes.

Remediation: General Strategies

- Calibration to past data.
- Data collection or simulation for model blindspots.
- Detection and elimination of non-robust features.
- Missing value injection during training or scoring.
- Model or model artifact editing.

References

This presentation:
https://www.github.com/jphall663/jsm_2019

Code examples for this presentation:
https://www.github.com/jphall663/interpretable_machine_learning_with_python
https://www.github.com/jphall663/responsible_xai

References

[1] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. URL: http://www.fairmlbook.org. fairmlbook.org, 2018.

[2] Marco Barreno et al. "The Security of Machine Learning." In: *Machine Learning* 81.2 (2010). URL: https://people.eecs.berkeley.edu/~adj/publications/paper-files/SecML-MLJ2010.pdf, pp. 121–148.

[3] Daniel Kang et al. *Debugging Machine Learning Models via Model Assertions*. URL: https://debug-ml-iclr2019.github.io/cameraready/DebugML-19_paper_27.pdf.

[4] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf. Curran Associates, Inc., 2017, pp. 4765–4774.

[5] Reza Shokri et al. "Membership Inference Attacks Against Machine Learning Models." In: *2017 IEEE Symposium on Security and Privacy (SP)*. URL: https://arxiv.org/pdf/1610.05820.pdf. IEEE. 2017, pp. 3–18.

References

[6] Florian Tramèr et al. "Stealing Machine Learning Models via Prediction APIs." In: *25th {USENIX} Security Symposium ({USENIX} Security 16)*. URL: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_tramer.pdf. 2016, pp. 601–618.

[7] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. "Mitigating Unwanted Biases with Adversarial Learning." In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. URL: https://arxiv.org/pdf/1801.07593.pdf. ACM. 2018, pp. 335–340.