A Cascading Machine Learning Model for FERC and EIA Data Linkage
Andrew Bartnof & Alex Engel

Andrew Bartnof, for RMI

April 30, 2025

1 Introduction

The FERC Form 1 - Electric Utility Annual Report is a "comprehensive financial and operating report submitted for Electric Rate regulation and financial audits"[1]. Every year, power plant operators are obliged to fill out this form, and summarize their power plants' operations. There is little, if any, validation for these forms, and no consistent identifiers beyond the one for the reporting entity.

Likewise, each year, the EIA puts out a series of datasets (forms 860 and 923) which describe the power plants in operation in the nation. In contrast, EIA forms involve a review and validation process and use internally consistent identifiers for utilities, power plants, generators, and other items reported in EIA forms.

Regrettably, there is no common ID that can be used to look up a single power plant in both the FERC and EIA records. The FERC forms are completed manually by the power plant operators, while the EIA is completed by administrators at the federal government. Consequently, even if these two forms refer to the same power plant by name, they can differ in all sorts of ways. A few examples of how these records can *nearly* align and make data-linkages really hard for a computer:

- The FERC entry can have a typo; the FERC entry can refer to the new corporate owners while the EIA one refers to the old ones
- The FERC entry can do something like clustering plants I through IX, while the EIA doesn't cluster plants and doesn't use roman numerals
- The FERC entry can refer to an entire facility, while the EIA records refer to a distinct component of the facility.

[1] https://www.ferc.gov/general-information-0/electric-industry-forms/form-1-1-f-3-q-electric-historical-vfp-data

The list goes on.

A single FERC entry could be matched to a single EIA entry in a few minutes, if you're doing it by hand. But there are thousands of entries to go through, and every year brings more.

2 A Cascading Machine-Learning Solution

Our process required comparing two things. First each hand-written FERC power plant record that could represent, at the discretion of the utility and form-filer, a single generator at a power plant, an arbitrary collection of generators at a power plant, or a whole power plant (we refer to these various aggregations as "plant parts"). And the second, a set of EIA power plant data aggregations constructed to correspond with those possible plant parts used by FERC respondents. The only blocking rule that we imposed was that they had to refer to the same calendar year.

We've designed a set of machine-learning models that can essentially mimic the ways in which a human would match these by hands. Our model looks at each FERC entry individually, just like a human would, and compares it to all of the EIA entries that it *could* link to. The only blocking rule that we imposed was that they had to refer to the same calendar year. Finally, our model gives a probability for each FERC entry and all possible EIA entries for that year– the FERC:EIA linkage with the highest probability wins, and is our preferred match. Easy!

For our FERC and EIA input files, we use the Catalist-Coop PUDL datasets[2]. Cleverly, Catalist-Coop has broken out each EIA entity into its constituent parts– so, for a given EIA plant, we can also recognize each generator therein as a separate row, and a possible match. We then compare each FERC entry to every possible broken out EIA record. This means that for each FERC entry, we are matching against (the number of plants in the nation for that year) x (the number of ways in which the parts of said plants can be broken out)– a very large number.

We designed two feature encodings that represented two different ways of thinking about this problem: in encoding A, we focused on the kinds of proper nouns that a human might focus on. In encoding B, in contrast, we focused on the kinds of technical metrics that effectively describe the plant's structure and performance. Then, for each feature encoding, we fit both an artificial neural network, and a gradient boosting model (using Keras/Tensorflow, and LightGBM, respectively).

Stage 1 Models:

- Model A, Artificial Neural Network
- Model A, Gradient Boosting Model
- Model B, Artificial Neural Network

[2]https://github.com/catalyst-cooperative/pudl/?tab=readme-ov-file

- Model B, Gradient Boosting Model

Finally, we fit a second-stage model, a gradient boosting model, which takes all of the above data (the input data for each model, as well as each model's fitted values), and judges which match between FERC and EIA records is most probable.

Stage 2 Model:

- Gradient Boosting Model

We have had very good results from this architecture. Our goodness-of-fit metrics exceeds existing data-linkage options, and generally mimics the kinds of choices we would make manually.

3 Hyperparameter Search

The neural network models each had 2 hidden layers, with dropout in order to avoid overfit. All of the models were trained in the same general way: we used the RayTune package, using the Optuna optimizer, to perform an initial hyper-parameter search, using a 4/5 of the data for training, and 1/5 as testing. The results of these hyper-parameter searches can be seen in figures 1, 2, 5, and 6. From this list of tested hyper-parameters, we selected the most promising hyper-parameter options, and we performed five-fold cross-validation on them (80% of the data is used to train a model, and 20% is used to test). Finally, we manually looked at the virtues and shortcomings of each cross-validated option, and chose the hyper-parameters we liked best. The results of these cross-validations can be seen in figures 3, 4, 7, and 8.

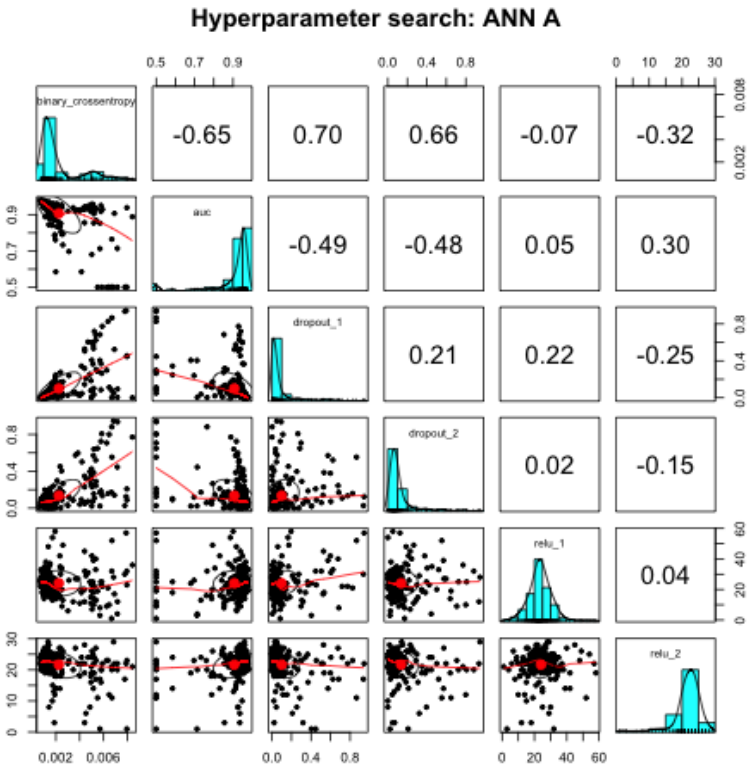

Figure 1: Distribution of the hyperparameter search for ANN model A

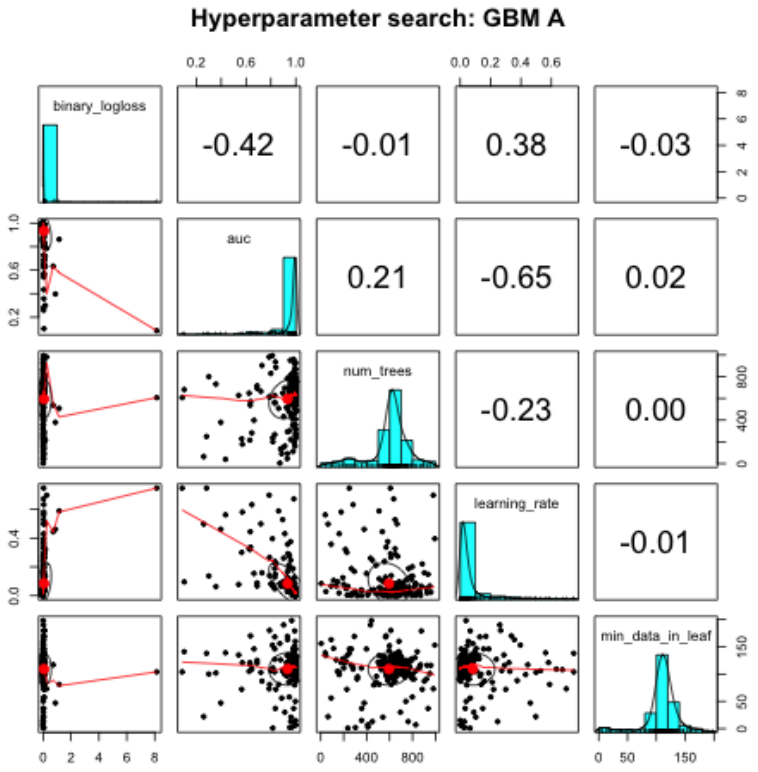

Figure 2: Distribution of the hyperparameter search for GBM model A

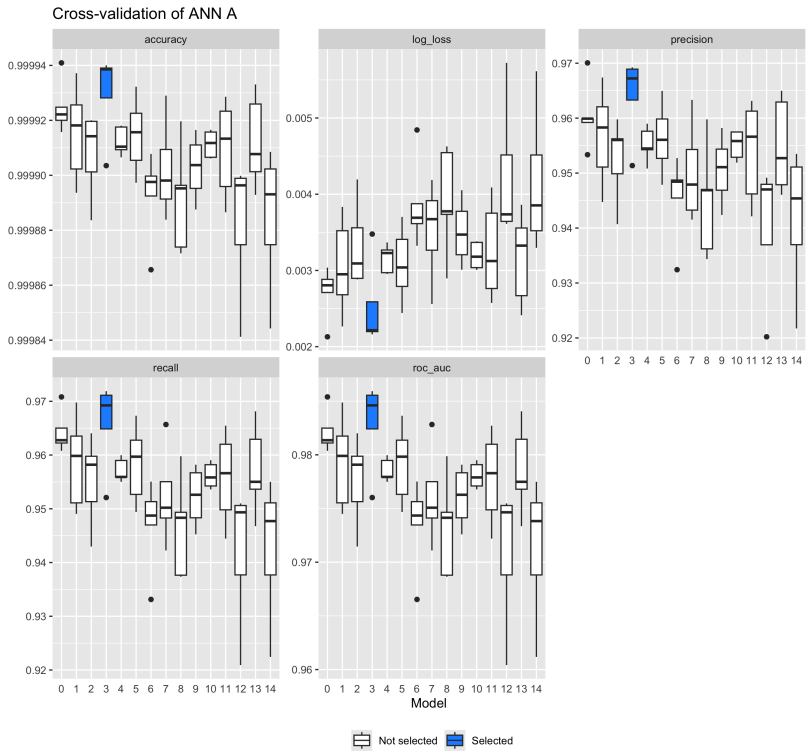

Figure 3: Cross-validation metrics for ANN model A

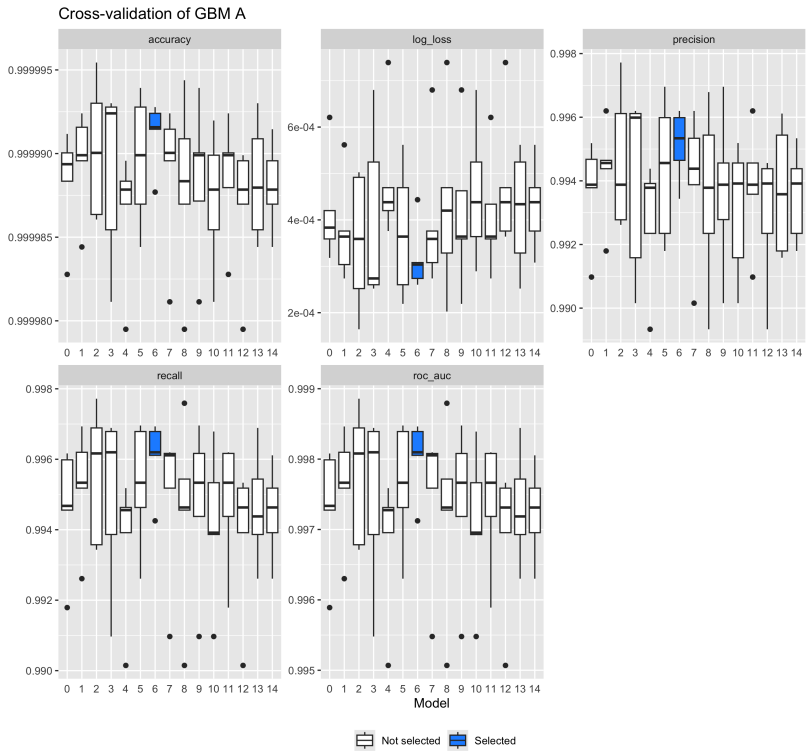

Figure 4: Cross-validation metrics for GBM model A

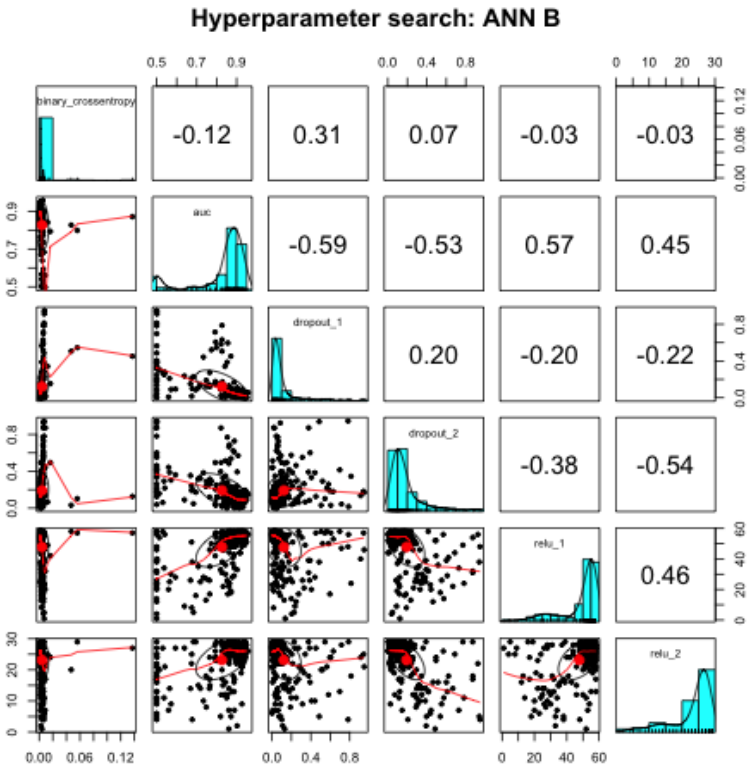

Figure 5: Distribution of the hyperparameter search for ANN model B

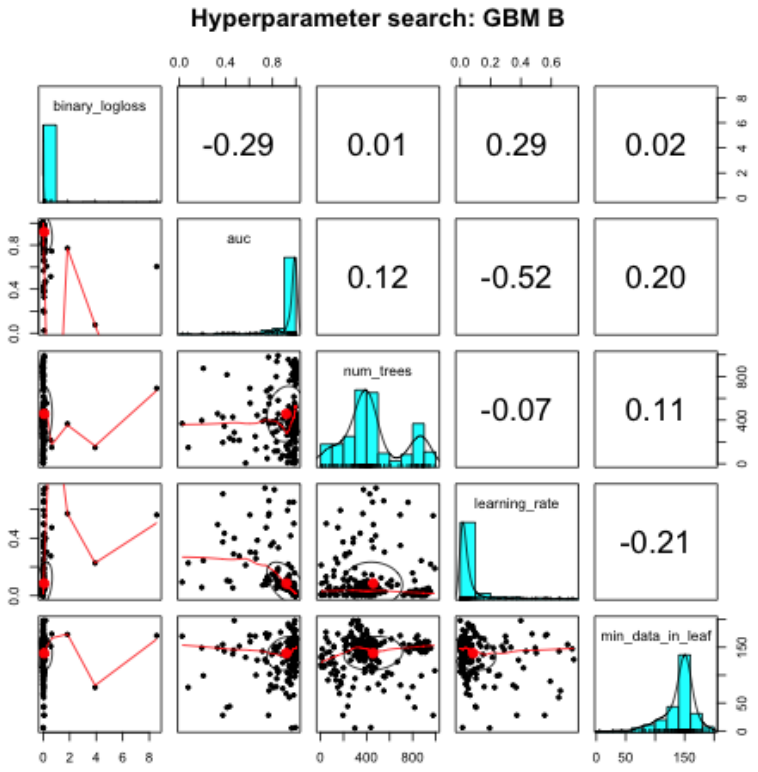

Figure 6: Distribution of the hyperparameter search for GBM model B

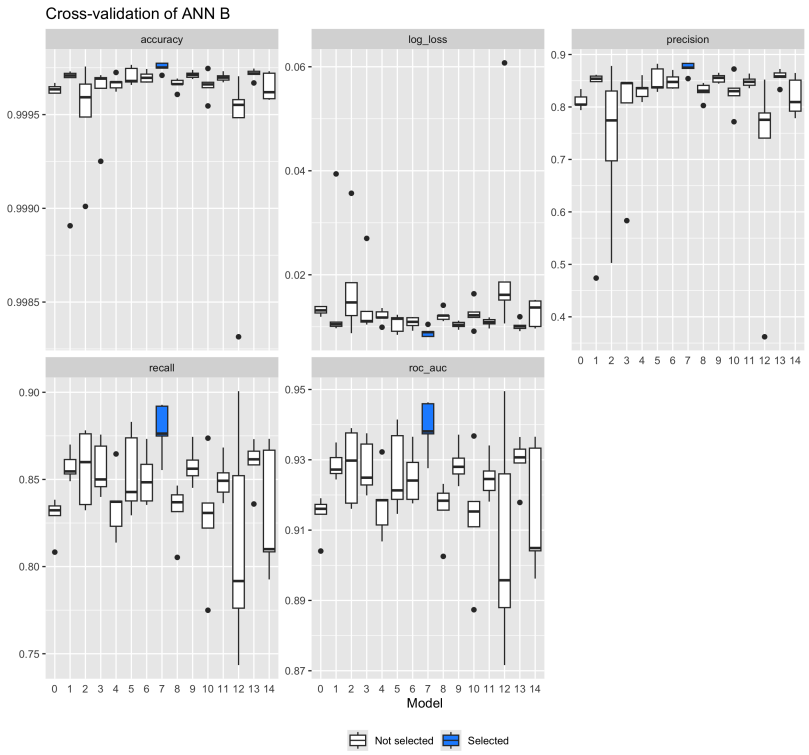

Figure 7: Cross-validation metrics for ANN model B

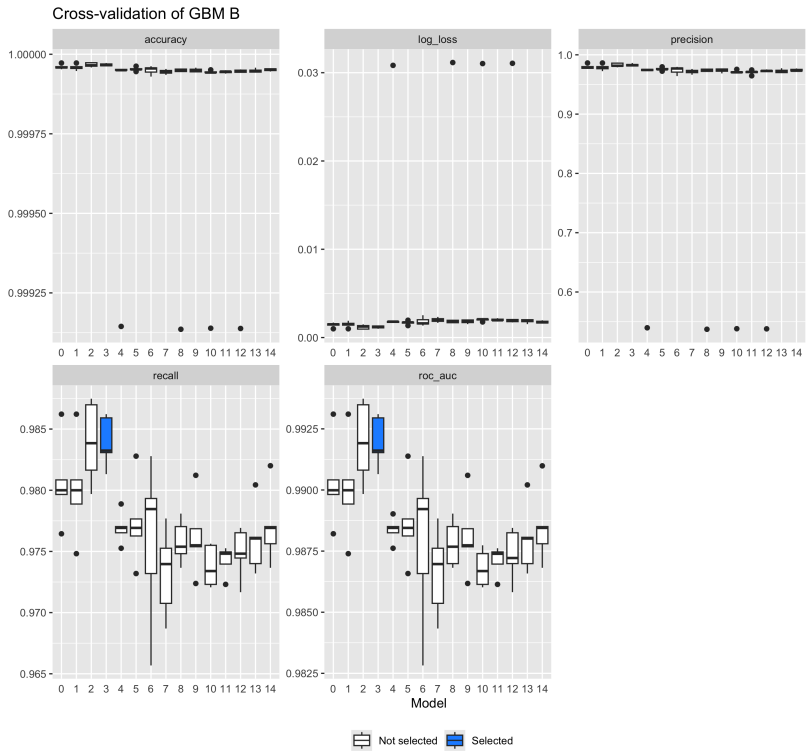

Figure 8: Cross-validation metrics for GBM model B

Interestingly enough, even though each of the 1st stage models performed admirably, if we compared their responses to each other, we found that their responses didn't correlate very highly (cf figure 9)! This was highly reassuring that a cascade architecture was the way to go. The stage 2 model would be a GBM. In our model, each of the initial four models gives its fitted values; then, we feed those fitted values, along with some characteristics about the FERC entries in question, into a second-tier model, which gives us our final linkages.

While all of the metrics on the stage 1 models are based on normal 5-fold cross validation (4 folds for training, 1 for testing), for the stage 2 models, 2/5 of the data was held for stage 1 model training; 2/5 for stage 2 model training/ and 1/5 for stage 2 model validation. Consequently, the stage 2 model was actually evaluated based on a model fit on half as many data. We have no doubt that this gives us an extremely conservative set of goodness-of-fit metrics; the final model, which is fit on the entire dataset, should perform even better.

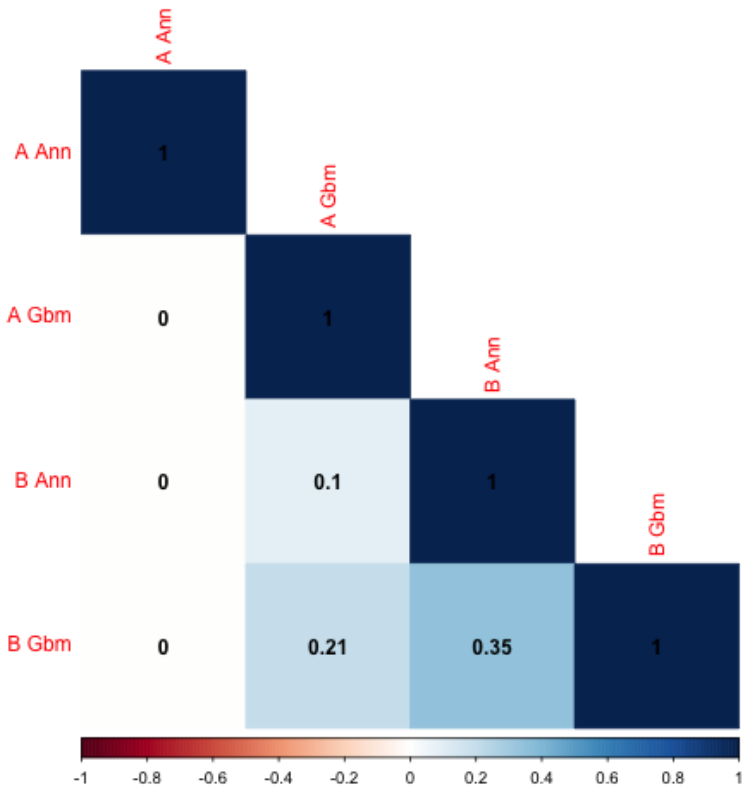

Figure 9: Even though each 1st stage model performed admirably, their responses didn't really correlate very highly with each other. This was a good indication that a cascade-model was the right path.

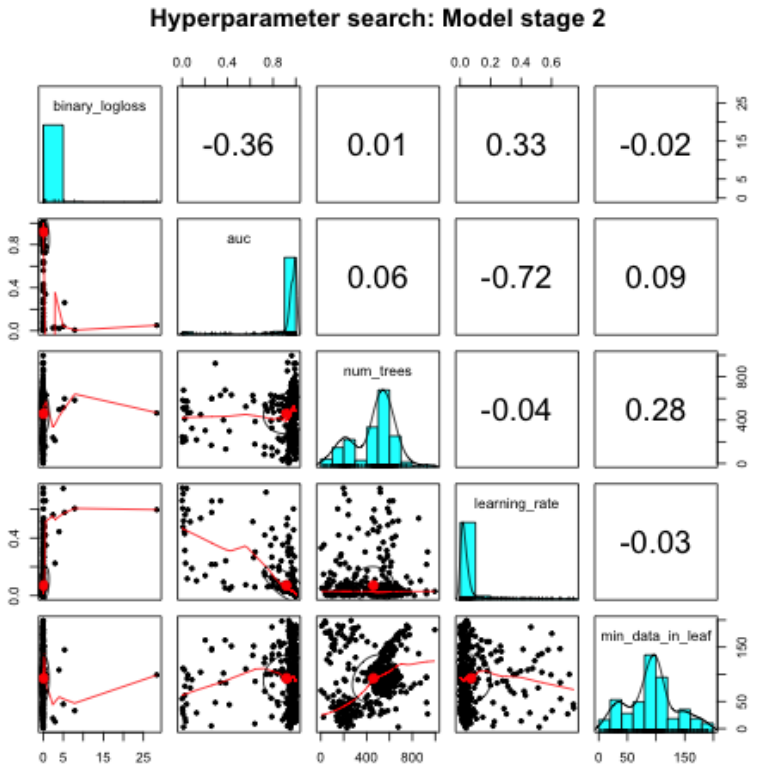

Figure 10: Distribution of the hyperparameter search for stage 2 GBM

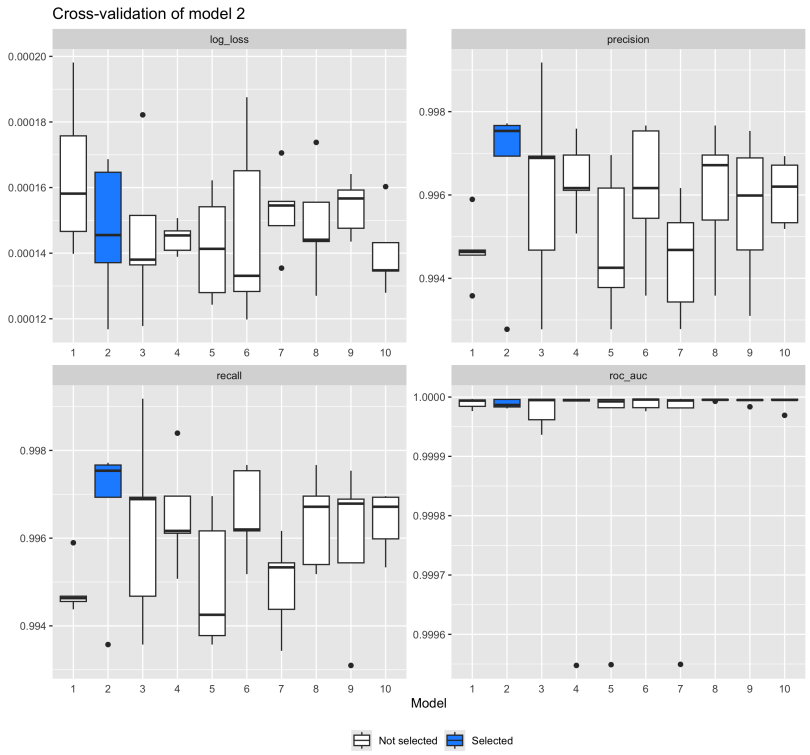

Figure 11: Cross-validation metrics for stage 2 GBM

4 User Guide

All of the above scripts can be run from within pixi, a package management tool. Please note that for every python file, there is both a jupyter notebook, which is handy for doing interactive work, and a derived . py file, which can just be run from terminal. Consequently, we're only listing the . py files in this workflow.

Also please note that at the top of each script, there is a reference to a directory, **data_dir**. This is a reference to where you place your working files (eg. on an external harddrive, in your Documents folder, etc). Please change the referent location for **data_dir** to wherever your project is located.

The files should be run in the following order:

section	file #	directory	filename
General Scripts	1	general_scripts	collect_data.R
General Scripts	2	general_scripts	clean_positive_matches.R
General Scripts	3	general_scripts	create_training_matches_and_mismatches.R
General Scripts	4	general_scripts	entire_dataset_cartesian_product_of_ferc_and_eia_ids.R
Prepare training data, models A	1	model_a	model_a_create_comparable_metrics_training.R
Prepare training data, models A	2	model_a	model_a_feature_engineering_training.R
Prepare training data, models A	(not run)	model_a	model_a_create_comparable_metrics_functions.R
Prepare training data, models A	(not run)	model_a	model_a_feature_engineering_functions.R
Prepare training data, models B	1	model_b	model_b_precreate_comparison_tables.R
Prepare training data, models B	2	model_b	model_b_feature_engineering_training.R
Prepare training data, models B	(not run)	model_b	model_b_feature_engineering_functions.R
Create model A, ANN	1	model_a/model_a_ann/model_a_ann_training	model_a_ann_hyperparameter_search.ipynb
Create model A, ANN	1	model_a/model_a_ann/model_a_ann_training	model_a_ann_hyperparameter_search.py
Create model A, ANN	2	model_a/model_a_ann/model_a_ann_training	model_a_ann_hyperparameter_search_dig_into_best_candidat
Create model A, ANN	2	model_a/model_a_ann/model_a_ann_training	model_a_ann_hyperparameter_search_dig_into_best_candidat
Create model A, ANN	3	model_a/model_a_ann/model_a_ann_training	model_a_ann_hyperparameter_search_final_metrics_analysis.I
Create model A, ANN	4	model_a/model_a_ann/model_a_ann_training	model_a_ann_fit.ipynb
Create model A, ANN	4	model_a/model_a_ann/model_a_ann_training	model_a_ann_fit.py
Create model A, GBM	1	model_a/model_a_gbm/model_a_gbm_training	model_a_gbm_hyperparameter_search.ipynb
Create model A, GBM	1	model_a/model_a_gbm/model_a_gbm_training	model_a_gbm_hyperparameter_search.py
Create model A, GBM	2	model_a/model_a_gbm/model_a_gbm_training	model_a_gbm_hyperparameters_search_2_cv.ipynb
Create model A, GBM	3	model_a/model_a_gbm/model_a_gbm_training	model_a_train_rank_hyperparameters.R
Create model A, GBM	4	model_a/model_a_gbm/model_a_gbm_training	model_a_gbm_fit.ipynb
Create model A, GBM	4	model_a/model_a_gbm/model_a_gbm_training	model_a_gbm_fit.py
Create model B, ANN	1	model_b/model_b_ann/model_b_ann_training	model_b_ann_hyperparameter_search.py
Create model B, ANN	1	model_b/model_b_ann/model_b_ann_training	model_b_ann_hyperparameter_search.ipynb
Create model B, ANN	2	model_b/model_b_ann/model_b_ann_training	model_b_ann_hyperparameter_search_dig_into_best_candidat
Create model B, ANN	2	model_b/model_b_ann/model_b_ann_training	model_b_ann_hyperparameter_search_dig_into_best_candidat
Create model B, ANN	3	model_b/model_b_ann/model_b_ann_training	model_b_ann_fit.ipynb
Create model B, ANN	3	model_b/model_b_ann/model_b_ann_training	model_b_ann_fit.py
Create model B, ANN	4	model_b/model_b_ann/model_b_ann_training	model_b_ann_hyperparameter_search_final_metrics_analysis.I
Create model B, GBM	1	model_b/model_b_gbm/model_b_gbm_training	model_b_gbm_hyperparameter_search.ipynb
Create model B, GBM	1	model_b/model_b_gbm/model_b_gbm_training	model_b_gbm_hyperparameter_search.py
Create model B, GBM	2	model_b/model_b_gbm/model_b_gbm_training	model_b_gbm_hyperparameters_search_2_cv.ipynb
Create model B, GBM	3	model_b/model_b_gbm/model_b_gbm_training	model_b_gbm_hyperparameters_search_choose_hp.R
Create model B, GBM	4	model_b/model_b_gbm/model_b_gbm_training	model_b_gbm_fit.ipynb
Create model B, GBM	4	model_b/model_b_gbm/model_b_gbm_training	model_b_gbm_fit.py
Prepare tranches for models A	1	model_a	model_a_create_comparable_metrics_tranches.R
Prepare tranches for models B	1	model_b	model_b_feature_engineering_tranches.R
Prepare tranches for models A	2	model_a	model_a_feature_engineering_tranches.R
Predict tranches, models A	1	model_a/model_a_ann/model_a_ann_usage	model_a_ann_predict.ipynb
Predict tranches, models A	1	model_a/model_a_ann/model_a_ann_usage	model_a_ann_predict.py
Predict tranches, models A	2	model_a/model_a_gbm/model_a_gbm_usage	model_a_gbm_predict.py
Predict tranches, models A	2	model_a/model_a_gbm/model_a_gbm_usage	model_a_gbm_predict.ipynb
Predict tranches, models B	1	model_b/model_b_ann/model_b_ann_usage	model_b_ann_predict.py
Predict tranches, models B	2	model_b/model_b_gbm/model_b_gbm_usage	model_b_gbm_predict.ipynb
Predict tranches, models B	2	model_b/model_b_gbm/model_b_gbm_usage	model_b_gbm_predict.py
Create 2nd stage model	1	model_stage_2/model_stage_2_training	prepare_for_model_2_hp_search.py
Create 2nd stage model	1	model_stage_2/model_stage_2_training	prepare_for_model_2_hp_search.ipynb
Create 2nd stage model	2	model_stage_2/model_stage_2_training	get_y_fit_model_a.ipynb
Create 2nd stage model	3	model_stage_2/model_stage_2_training	get_y_fit_model_b.ipynb
Create 2nd stage model	4	model_stage_2/model_stage_2_training	model_2_hp_search.ipynb
Create 2nd stage model	5	model_stage_2/model_stage_2_training	cross_validate_best_hp_candidates.py
Create 2nd stage model	5	model_stage_2/model_stage_2_training	cross_validate_best_hp_candidates.ipynb
Create 2nd stage model	6	model_stage_2/model_stage_2_training	model_2_analyze_cv.R
Create 2nd stage model	7	model_stage_2/model_stage_2_training	fit_second_stage_model_gbm.ipynb
Create 2nd stage model	7	model_stage_2/model_stage_2_training	fit_second_stage_model_gbm.py
Create 2nd stage model	8	model_stage_2	get_correlations_and_misc_stats_from_input_models.R
Create 2nd stage model	9	model_stage_2	analyze_correlations_and_misc_stats_from_input_models.R
Predict tranches, 2nd stage	1	model_stage_2/model_stage_2_usage	get_model_stage_2_y_fit.ipynb
Predict tranches, 2nd stage	2	model_stage_2/model_stage_2_usage	analyze_final_fit_qc.R

5 Appendix

fold	accuracy	roc_auc	log_loss	precision	recall
0	0.9999	0.9846	0.0022	0.9692	0.9692
1	0.9999	0.9859	0.0022	0.9672	0.9719
2	0.9999	0.9824	0.0026	0.9633	0.9649
3	0.9999	0.9855	0.0022	0.9689	0.9711
4	0.9999	0.9760	0.0035	0.9514	0.9521

Table 1: Cross-validation metrics for ANN model A

fold	accuracy	roc_auc	log_loss	precision	recall
0	1.0000	0.9984	0.0003	0.9960	0.9968
1	1.0000	0.9981	0.0003	0.9953	0.9961
2	1.0000	0.9985	0.0003	0.9946	0.9969
3	1.0000	0.9971	0.0004	0.9934	0.9943
4	1.0000	0.9981	0.0003	0.9962	0.9962

Table 2: Cross-validation metrics for GBM model A

fold	accuracy	roc_auc	log_loss	precision	recall
0	0.9998	0.9460	0.0081	0.8830	0.8920
1	0.9997	0.9374	0.0091	0.8735	0.8749
2	0.9998	0.9463	0.0081	0.8840	0.8928
3	0.9998	0.9381	0.0090	0.8749	0.8763
4	0.9997	0.9276	0.0105	0.8541	0.8554

Table 3: Cross-validation metrics for ANN model B

fold	accuracy	roc_auc	log_loss	precision	recall
0	1.0000	0.9929	0.0011	0.9836	0.9859
1	1.0000	0.9906	0.0013	0.9813	0.9813
2	1.0000	0.9931	0.0010	0.9862	0.9862
3	1.0000	0.9916	0.0012	0.9825	0.9832
4	1.0000	0.9915	0.0013	0.9808	0.9831

Table 4: Cross-validation metrics for GBM model B

fold	accuracy	roc_auc	log_loss	precision	recall
0	0.9928	0.9936	0.0002	1.0000	0.9859
1	0.9977	0.9977	0.0002	1.0000	0.9813
2	0.9969	0.9969	0.0001	1.0000	0.9862
3	0.9975	0.9975	0.0001	1.0000	0.9832
4	0.9977	0.9977	0.0001	1.0000	0.9831

Table 5: Cross-validation metrics for stage 2 GBM