Responsible Machine Learning
Assignment 1
10 points

May 14, 2022

In Assignment 1, you will work with your group to train interpretable machine learning (ML) models following the instructions below. A template has been provided as an example of how to train and compare a few different interpretable models. For those of you who use Python virtual environments, a basic `requirements.txt` file is also available for the template.

Please let me know immediately if you find typos or mistakes in this assignment or related materials.

1 Download Data.

Download Home Mortgage Disclosure Act (HMDA) data as zip files from this folder in the class repository. The folder includes two data files:

- `hmda_train_preprocessed.zip` – Zipped CSV HMDA *labeled* training data.
- `hmda_test_preprocessed.zip` – Zipped CSV HMDA *unlabeled* test data.

Later you will score the unlabeled test data with your models and submit these scores as part of your assignment deliverable. See cell 3 in the template.

2 Load and Explore Data.

Load the data into modeling software. Training data contains 160338 rows and 23 columns. Test data contains 19831 rows and 22 columns. The features to use for Assignment 1 are as follows:

- `high_priced`: Binary target, whether (1) or not (0) the annual percentage rate (APR) charged for a mortgage is 150 basis points (1.5%) or more above a survey-based estimate of similar mortgages. (High-priced mortgages are legal, but somewhat punitive to borrowers. High-priced mortgages often fall on the shoulders of minority home owners, and are one of many issues that perpetuates a massive disparity in overall wealth between different demographic groups in the US.)
- `conforming`: Binary numeric input, whether the mortgage conforms to normal standards (1), or whether the loan is different (0), e.g., jumbo, HELOC, reverse mortgage, etc.
- `debt_to_income_ratio_std`: Numeric input, standardized debt-to-income ratio for mortgage applicants.
- `debt_to_income_ratio_missing`: Binary numeric input, missing marker (1) for `debt_to_income_ratio_std`.
- `income_std`: Numeric input, standardized income for mortgage applicants.
- `loan_amount_std`: Numeric input, standardized amount of the mortgage for applicants.
- `intro_rate_period_std`: Numeric input, standardized introductory rate period for mortgage applicants.

- `loan_to_value_ratio_std`: Numeric input, ratio of the mortgage size to the value of the property for mortgage applicants.
- `no_intro_rate_period_std`: Binary numeric input, whether or not a mortgage does not include an introductory rate period.
- `property_value_std`: Numeric input, value of the mortgaged property.
- `term_360`: Binary numeric input, whether the mortgage is a standard 360 month mortgage (1) or a different type of mortgage (0).

See cell 4 in the template for modeling roles.

This data contains no major quality issues, so no preprocessing is required. Please familiarize yourself with the data using basic exploration techniques – see cells 5–6 in the template. You may optionally try to improve your model with feature engineering or other preprocessing approaches.

Training data should be used to create training and validation partitions. Test data will only be used to evaluate your models by the instructor. See cell 7 in the template.

3 Train Interpretable Models

Train at least two types of interpretable models, ensuring best practices like reproducibility, validation-based early-stopping, and grid search are used. (Scikit-learn does not necessarily make applying such best practices easy.) You are encouraged to try packages like `h2o`, `interpret`, and `XGBoost`, but you may use any interpretable or self-explainable approach on which you will be able to apply explanation, discrimination testing and remediation, and model debugging approaches in coming weeks. Please reach out with questions about appropriate modeling techniques.

The template contains examples for elastic net logistic regression using `h2o` (see cells 8–10), monotonic gradient boosting machines (GBM) using `XGBoost` (see cells 12–14), and explainable boosting machines (EBM) using `interpret` (see cells 16–18).

4 Submit Code Results

Your deliverable for this assignment has two parts. Each part is worth 5 points.

- You must check in your code to a public GitHub repository by the deadline below. Code should be available as a commented script, Jupyter notebook, R markdown or other polished and professional format.
- You must create submission files with output probabilities for each row of the test data. The submission file should have one column named `phat`. Each model should have a separate submission file named using a `<group_indentifier>_<model_type>.csv` convention, similar to the example submission file. See cells 11, 15, and 19 for examples of writing submission files. Your group's submission will be ranked using the cross-validated ranking method discussed during Lecture 1. The remaining five points for the assignment will be issued based on this ranking. You will have opportunities to increase your rank each week of the class.

Your deliverables are due Wednesday, May 25th, at 11:59:59 PM ET.

Please send an email to jphall@gwu.edu by that deadline with the link to your group's GitHub page and with your zipped submission files.