Problem Set 2: Predicting Poverty

"Wars of nations are fought to change maps. But wars of poverty are fought to map change" M. Ali

MECA 4107

Due Date: July 10 at 23:59 on Bloque Neón

1 Introduction

This problem set was inspired by a recent competition hosted by the world bank: Pover-T Tests: Predicting Poverty. The idea is to predict poverty in Colombia. As the competition states, *"measuring poverty is hard, time consuming, and expensive. By building better models, we can run surveys with fewer, more targeted questions that rapidly and cheaply measure the effectiveness of new policies and interventions. The more accurate our models, the more accurately we can target interventions and iterate on policies, maximizing the impact and cost-effectiveness of these strategies."*

Predictions have to be made at the household level only. Data, however, are provided at the household and individual levels. You can use individual-level information to build extra variables to improve your prediction. You can use the variable `id` to merge households with individuals.

The data folder contains four data sets coming from the GEIH. Training and testing data sets at the household and the individual level. You will note that some variables are missing in the testing data sets. This is by design to make things a bit more challenging.

A document describing the variables is available in the data folder. You can also check them on the DANE website.

An essential dimension for policymakers is they can *rapidly and cheaply* measure poverty. When building your model, aim for a model that uses the minimum number of variables.

There are two expected outputs:

1. A `.pdf` document.
2. And a `.csv` file with your predictions.

1.1 General Instructions

The main objective is to construct a predictive model of household poverty. Note that a household is classified as

$$Poor = I(Inc < Pl) \tag{1}$$

where I is an indicator function that takes one if the family income is below a certain poverty line. This suggests two ways to go about predicting poverty. First, approach it as a classification problem: predict zeros (no poor), and ones (poor). Second, as an income prediction problem. With the predicted income, you can use the poverty line and get the classification. You will explore both routes in this problem set.

The document should not be longer than 3 (three) pages and include at most 6 exhibits (tables and/or figures). You are welcome to add an appendix, but the main document must be self-contained. Specifically, a reader should be able to follow the analysis in the paper and be convinced it is correct and coherent from the main text alone, without consulting the appendix.

The document must contain the following sections:

- Introduction. The introduction briefly states the problem and if there are any antecedents. It describes the data and its suitability for the issue at hand. It contains a preview of the results and main takeaways.

- Data. Treat this section as an opportunity to present a compelling narrative to justify or defend your data choices, walk the reader through your reasoning of how you "cleaned" the data, and describe it accordingly with descriptive stats, graphs, etc. At a minimum, you should include a descriptive statistics table. Use your professional knowledge to add value to this section. Do not present it as a "dry" list of ingredients.

- Models and Results. Present the models that you are using. When selecting the best models, argue why these are the best models for your task.

  - Classification Models. This section should include (not necessarily in this order):
    - A detailed explanation of the final chosen model. The explanation must include how the model was trained, hyper-parameters selection, and other relevant information.
    - Include comparisons to at least 5 other models. You can compare them in terms of ROC, AUC, False Positives, or False Negatives.
    - Describe the variables that you used in the model and a measure of their relative importance in the prediction.
    - Describe any sub-sampling approach used to address class imbalances.
  - Income regression Models. This section should include (not necessarily in this order)

    - A detailed explanation of the final chosen model. The explanation must include how the model was trained, hyper-parameters selection, and other relevant information.
    - Include comparisons to at least 5 other models. Compare them in terms of MSE.
    - Convert the predicted income to a binary indicator and show the performance in terms of the ROC, AUC, False Positives, or False Negatives.
    - Describe the variables that you used in the model and a measure of their relative importance in the prediction.

- Conclusions and recommendations. In this section, you state the main takeaways of your work.

2 Additional Guidelines

I expect the following things from the problem set, omission of any of these guidelines will be penalized.

- Turn in your document and a submission `.csv` file with predictions in bloque neón.
    - An example of how the submission file should look like is in the data folder: `submission_template.csv`. This file includes three columns, one with the variable *id*, one with your prediction using a classification model, and one with a prediction using an income model.
    - I will judge predictions based on false-positive rates, false-negative rates, and the model's sparsity. In the final score, false negatives will have more weight (75%), i.e. poor families classified as non-poor.
    - Please follow the following convention for your `.csv` file name. The file name must include the name `predictions`, followed by your teammates' last names, and the number of variables you used for prediction, all separated by underscores, for example, `predictions_gomez_matinez_sarmiento_4.csv`, where `4` indicates the number of variables in your final model. The more variables you use, the lower your score.
    - I will assign bonus points based on the relative rankings.
- The document should point and include a link to your GitHub Repository.
- Follow the repository template.
- The repository should include a README file. A good README helps your project stand out from other projects and is the first file a person sees when they come across your repository. Therefore, this file should be detailed enough to focus on your project and how it does it, but not so long that it loses the reader's attention. For example, Project Awesome has a curated list of interesting READMEs.

- The repository should have at least five (5) substantial contributions from each team member.
- Tables, figures, and writing must be as neat as possible. Label all variables that you include. Describe all variables, statistics, etc., included in your figures or tables.
- Your code should be readable and include comments. In coding, like in writing, a good coding style is critical for readable code. I encourage you to follow the tidyverse style guide.