Predicting Poverty in Costa Rican Households through Proxy Means Testing

Lee-Or Bentovim, Katherine Dumais, Andrew Dunn, Kathryn Link-Oberstar
CAPP 30254
May 2023

Abstract

Using a Kaggle dataset from the Inter-American Development Bank, we design a machine learning model to classify household-level poverty using a Proxy Means Tests methodology. After data cleaning and collapsing the data to the household level, we use several oversampling techniques and cross validation to improve model performance given imbalances in poverty categories. After testing random forests, logistic regression, naive bayes, and k-nearest neighbors, as well as different combinations of hyperparameters, we select a logistic regression as our best performing model. We also test ensemble methods and explore using a binary poverty categorization. Finally, we note limitations of our approach and recommendations for further exploration.

1. Introduction

Accurately measuring poverty is an essential yet often intractable policy challenge. In many countries, poverty is mitigated through means-tested programs, which aim to direct aid to those most in need. However, these programs face a significant hurdle: the poorest individuals often struggle to provide the official documentation needed to demonstrate their financial circumstances, complicating the process of effectively distributing resources.

In response to this challenge, many Latin American countries have adopted a 'Proxy Means Test' (PMT) as a method to estimate a household's income indirectly. This is done based on observable characteristics or proxy variables, which can include aspects like housing conditions, asset ownership, educational attainment, access to basic services, and other socio-economic indicators. These variables are used to construct a predictive model that estimates the household's economic status.

PMTs are commonly utilized in means-tested social assistance programs and targeting mechanisms. However, their effectiveness can be compromised by the quality of the proxy variables used and the underlying assumptions of the model.

1.1 Project Goals

In 2018, the Inter-American Development Bank, the preeminent source of development financing for Latin America and the Caribbean, initiated a Kaggle Competition. The objective was to devise improved methods for using Proxy Means Tests (PMTs) to classify household-level poverty. The project with the highest score achieved a Macro F1 score of 44% on the test data provided in the competition.

Our project has dual objectives: to explore techniques for enhancing prediction accuracy and to propose recommendations for improving policy decision-making. As we refine our models, we aim to not only elevate their predictive capacity but also use our findings to foster more precise decision-making to allow efficient allocation of resources to the households most in need.

1.2 Literature Review

We consulted the significant literature on poverty measurement to inform our approach and feature engineering. Academic researchers have investigated ways to measure poverty using PMTs in developing contexts and individual-level characteristics to measure household poverty; research on these approaches was informative for our project. For example, some measurement approaches recommend using the highest individual level of socioeconomic characteristics like education as the household-level characteristics. Others recommended using household head characteristics to represent the household.

There are a number of salient features recommended by previous research. Potential features like sex ratio, child/woman ratio, the proportion of female workers to total workers as potential features, as well as income per consumption unit, rather than per capita, and the presence of a

marriage in a household. A number of studies cautioned against using the sex of the head of household as a potential feature. Finally, some papers on poverty research suggested that principal components analysis could be a potentially useful approach to reduce the dimensionality of the raw data.

1.3 Overview of Analysis and Findings

Key Challenges

Our preliminary analysis identified the two principal challenges:

- *Limited Dataset:* With approximately 3,000 households, the size of the dataset posed a significant constraint.
- *Imbalanced Dataset*: The dataset's distribution was highly skewed, with poverty classified into four target classes:
    - Class 1: Extreme poverty
    - Class 2: Moderate poverty
    - Class 3: Vulnerable households
    - Class 4: Non-vulnerable households

About 66% of the training data belonging to Class 4, which posed challenges for training our model, namely:

- The models might yield deceptively high accuracy by merely classifying most of the data as Class 4.
- Our models were provided with relatively few data points to distinguish between classes 1, 2, and 3.

Strategies

- In our quest to enhance our model, we explored feature engineering and adjustments to hyperparameters for performance improvement. We identified the following techniques as most effective in enhancing our models:
    - Oversampling: To address the class imbalance in the dataset.
    - Feature Selection: We improved the model's performance by eliminating features with low variance and incorporating feature scaling.
    - K-fold Cross Validation: To mitigate the effects of a small training dataset, we employed k-fold cross-validation.

2. Data Management

To avoid the adage “garbage in, garbage out,” we spent some initial time ensuring our the data we used for our models was high quality.

Data Cleaning

We conducted minor data cleaning on the raw Kaggle dataset. For example, the coding of the provided variables on years of education of the household head and the dependency ratio

(variables edjefe, edjefa, and dependency) didn't reflect the coding in the data dictionary. Using information from questions already asked in the Kaggle competition, we recoded the variables. For two other variables – number of tablets owned in the household and monthly rent payment (variables v18q1 and v2a1) – NA values in the variable suggested that this data field didn't apply to the household. In those two variables, we imputed a predicted rent payment and entered the number of tablets owned as 0, respectively. Reducing the number of variables with missing data would allow our machine learning models to use the full set of available data during our training process.

Feature Engineering

The provided dataset already included several engineered features. To further enhance the representativeness of our data and the performance of our model, we engineered new features at the household level using individual-level data:

- *Highest Level of Education in Household:* Captures the highest educational attainment among the members of the household.
- *Marital Status in Household:* Indicates whether there exists a marital bond within the household.
- *Maximum Age in Household:* Maximum age of a person living in the household. The highest age could indicate the generational makeup of the household and potential caretaking roles.
- *Sex Ratio in Household:* Reflects the gender balance in the household. It could have implications for understanding the gender dynamics within a household.
- *Child/Woman Ratio in Household*: Designed to provide a snapshot of the caregiving burden within the household. We created two versions of this variable, one where the cutoff between child and woman was age 12, and another where it was age 19. *Child/Adult Ratio in Household*: Designed to provide a different snapshot of the caregiving burden within the household. We created two versions of this variable, one where the cutoff between child and adult was age 12, and another where it was age 19.
- *Logged Monthly Rent Payment:* After imputing missing values, we created a logged variable which negated the impact of outliers. This recoded variable would likely be more informative than the original, skewed, variable.

Given the factor we were predicting using machine learning techniques was household poverty, the final step in our data management process was to collapse the dataset to the household level. We collapsed on the household head, given this was the individual both the literature and Kaggle competition organizers recommended. In the handful of instances where there was not a designated household head, we defined the household head as the oldest male household member. If this person did not exist, we used the oldest person in the household.

Model Performance With and Without Engineered Features

Testing our best-performing machine learning model (model specifications are explained in section 4.1), we saw that using all of our engineered features did improve model performance. The model with engineered features had higher accuracy, weighed F1, macro F1, and average recall for classes 1 and 2 compared to the model without engineered features, showing our engineered features improved the model's predictive performance.

Model	**Accuracy**	**Weighted F1**	**Macro F1**	**Recall**	**Precision**
Without engineered features	0.58	0.60	0.38	Class 1: 0.17 Class 2: 0.53 Class 3: 0.15 Class 4: 0.72 Avg. Recall for 1 and 2: 0.35	Class 1: 0.28 Class 2: 0.25 Class 3: 0.27 Class 4: 0.84
With engineered features	0.61	0.61	0.39	Class 1: 0.18 Class 2: 0.57 Class 3: 0.15 Class 4: 0.74 Avg. Recall for 1 and 2: 0.37	Class 1: 0.29 Class 2: 0.27 Class 3: 0.29 Class 4: 0.85

3. Model Development

3.1 Models Used

Over the course of our analysis we explored a number of machine learning models.

Random Forest

A Random Forest model is a machine-learning algorithm that creates many decision trees and averages the results from them. Because of this, random forest models have the benefit over decision trees of being less likely to overfit on training data. They are frequently used for classification questions.

Main takeaways:

- Random Forest had the best "out of the box" model performance.

- Random Forest is a good model for this research question because it is fairly robust to outliers and conducts feature selection as each tree is built. This makes it a good fit for our relatively small dataset and allows the algorithm to select features, rather than relying on manual testing.
- However, random forest models are computationally expensive, especially as we automated testing of hyperparameters. The model is also not interpretable, especially when compared to some of the other models we tested.

Logistic Regression

Logistic Regression is a type of supervised learning algorithm used for classification problems which uses a logistic function to model the probability of a binary or categorical outcome. In our case, our 1-4 levels of poverty allow us to make categorical predictions with Logistic Regression

The goal of Logistic Regression is to find the best parameters of a logistic function that minimizes the difference between the predicted probabilities and the actual outcomes. Logistic Regression uses the sigmoid function, which maps any real number into a range between 0 and 1, allowing us to interpret the output as a probability. The input features are weighted and combined linearly, and the resulting value is passed through the logistic function to produce a probability. We then use gradient descent to determine what the best classification is for these categories.

Main Takeaways:

- Providing the out-of-the-box model allowed us to quickly predict class 4, but nothing else meaningfully.
- Logistic regression had one of the highest F1 and recall scores, but it was largely based on hyperparameter tuning, feature selection, and highly contingent on oversampling.
    - The results of certain feature selection methods, such as Recursive Feature Elimination and SMOTE, fluctuated greatly.
    - Variance Thresholds significantly increased the efficacy of these methods, raising our Macro F1 scores from consistently in the 20's into the high 30's.
    - Oversampling significantly increased the reliability of our categorizing of 1-3, moving from under 10% to results as high as 50%,
- One drawback however, is that it assumes linear relationships which, based on the diversity of our data, may not be true in this case. Additionally, given the small size of the dataset and the complexity of the relationships, Logistic Regression may not be complex enough to model them precisely.
- Linear Regression ended up being one of our best predicting models with an average recall of .37, a macro F1 score .42 and a model accuracy is .59 as shown in 4.1

Naive Bayes

The Naive Bayes algorithm assumes that all features are independent of each other, meaning that the presence or absence of one feature does not affect the probability of another feature being present or absent. The algorithm uses Bayes' theorem to calculate the probabilities of different classes given the observed evidence. Bayes' theorem allows us to update our beliefs about the

probability of a hypothesis (such as the class of a data point) based on new evidence (such as the features of the data point).

Main Takeaways:

- Naive Bayes had some of the worst out-of-the-box performance, but was very responsive to scaling and oversampling.
- The benefit of Naive Bayes for this project is that it can handle large datasets and high-dimensional feature spaces efficiently.
- The limitations of Naive Bayes for this project are:
    - The Naive Bayes models assume that features are conditionally independent given the class label. This is almost certainly not true with this data.
    - The performance of Naive Bayes models can suffer when there is not enough data to estimate the probabilities accurately. We know we have a data scarcity problem with this dataset.
    - Naive Bayes models work better with categorical data. We have a lot of binary and continuous data in our dataset.
    - Naive Bayes models may not perform well on an imbalanced dataset, where the classes are not represented equally. This can lead to poor classification performance, as the Naive Bayes algorithm may be biased towards the majority class.
- We attempted to overcome some of these limitations by using Complement Naive Bayes (as discussed in section 3.5 Tuning Hyperparameters), and by employing a binary model as discussed in section 3.6)

K Nearest Neighbors (KNN)

The KNN algorithm is a common supervised learning classification algorithm. Class labels are assigned by plurality vote of the K nearest neighbors (hence the name). Two common methods for this plurality vote system are uniform (1 point 1 vote), and weighted, where points closer to our target point get more weight on their "vote".

Main Takeaways:

- Overall, KNN was our worst performing model on all counts except accuracy,
    - This is despite it being among the better models before tuning.
    - The high accuracy score is due to almost exclusively picking class 4,
- KNN is generally a good fit for smaller datasets where the computing cost is not too great
- However, it was not a good fit for this dataset as with the high number of parameters we had we were evenly valuing predictive and non-predictive parameters.
    - We attempted to filter down which parameters we used with the KNN model, but this was generally ineffective and suggested an alternative approach would be more fruitful.
    - Despite tinkering with hyperparameters (see below section), we were unable to get the model to stop predicting almost exclusively class four.

3.2 Feature Selection

Manual Feature Selection
We experimented with some manual feature selection, which resulted in relatively weak models. This lack of significant improvement might be attributed to limited expertise in the subject matter, indicating the need for more targeted domain knowledge or, potentially, unsupervised learning methods (such as selection on variance).

Variance Thresholds
In the interest of optimizing the feature set and eliminating redundancy, we implemented a variance threshold strategy in our Naive Bayes and Logistic Regression Models. This allowed us to discard features that demonstrated low entropy. These low variance features often contribute little to the predictive power of a model since they do not change significantly from one observation to another. This tuning happens when we clean and load the data before running our models.

3.3 Oversampling

To address class imbalances in our dataset, we adopted an oversampling strategy. This approach involves augmenting the data for underrepresented classes to balance their proportions with that of the majority class, aiming to improve model performance by reducing bias toward the majority class.

We experimented with two distinct oversampling techniques:

- Random Over Sampling (ROS): Using the RandomOverSampler method, we randomly sampled instances from the minority class with replacement to enhance their representation in the dataset. While this approach can help balance classes, one potential downside is that it might lead to overfitting due to the duplication of data points.
- Synthetic Minority Oversampling Technique (SMOTE): Unlike ROS, which simply duplicates existing entries, SMOTE generates synthetic data based on the feature space similarities between existing instances. By creating "synthetic" examples, SMOTE presents a more robust method of balancing classes. However, it's worth noting that SMOTE can potentially introduce noise in the dataset if the minority class instances are not sufficiently close in the feature space.

It's essential to meticulously monitor model performance and validate results when using these oversampling methods, given their potential pitfalls such as overfitting.

3.4 K-Fold Cross Validation

The relatively small size of our dataset presented another significant challenge. Small datasets can often lead to overfitting and a lack of generalizability in the model's predictions. To mitigate this, we implemented a technique known as k-fold cross-validation.

K-Fold Cross Validation

K-fold cross-validation is a robust method used to estimate the performance of machine learning models when the available data is limited. This technique involves splitting the dataset into 'k' equally-sized subsets or 'folds'. The model is trained on 'k-1' folds, and the remaining fold is used for validation. This process is repeated 'k' times, with each fold used exactly once for validation. The model's performance is then assessed based on the average results from all the iterations. This approach not only provides a more reliable estimate of the model's performance but also maximizes the usage of available data for training and validation.

Oversampling with K-Fold Cross-Validation

When oversampling techniques (like ROS or SMOTE) are used in conjunction with k-fold cross-validation, careful attention must be paid to the sequencing of these methods. To prevent data leakage – where information from the validation set "leaks" into the training set – it is essential to perform oversampling within each fold during the cross-validation process. In other words, oversampling should only be applied to the training set of each fold, leaving the validation set untouched. This approach ensures that the synthetic samples generated during oversampling do not appear in both the training and validation sets, thus preserving the integrity of the validation process.

3.5. Tuning Hyperparameters

In addition to feature engineering, oversampling, and k-fold cross-validation, each model also required some model-specific tuning of hyperparameters.

Random Forest

- We used a randomized search on hyperparameters to find the best-performing combination of hyperparameters. We selected reasonable possible values for the number of estimators, maximum features, maximum depth, minimum number of samples to split a node, minimum samples to be a leaf node, and whether bootstrap samples were used when building trees. Given the possible values, there were 4,320 possible combinations of the different hyperparameters. As such, we opted for a randomized search rather than a grid search which would check every possible hyperparameter combination.
- We conducted the randomized search on both SMOTE-oversampled and randomly-oversampled datasets. Given our prioritization of the recall of classes 1 and 2, we selected the average recall of these two classes as the metric to select the best combination of metrics and best oversampling method

Logistic Regression

- Regularization helps to improve the generalization performance of the model by balancing the bias-variance trade-off and reducing overfitting. By penalizing the weights of the input features, regularization encourages the model to focus on the most important features that are most relevant to the target variable. Here, L2 regularization encourages small and non-zero coefficients for all irrelevant features.
- Solver algorithm: When creating a logistic regression, the solver algorithm determines how logistic regression solves the optimization problem. Different solvers have different

computational efficiency and are suited for different types of problems. We selected 'Liblinear' because it is specifically tuned for small datasets.
- Number of iterations: We generally went with the default of 100 iterations, but in early models, prior to oversampling, we increased this number if the model did not converge.
- We chose not to provide any particular class weighting.

Naive Bayes

There are 4 primary naive bayes models. Each of these models have specific assumptions about the data and can be considered as hyperparameters in the context of model selection.

1. Gaussian Naive Bayes assumes that the continuous data follows a normal distribution. In our dataset this includes things like age and years of education.
2. Multinomial Naive Bayes is used for discrete data. In our dataset this includes things like number of people in the household and number of people above or below a certain age.
3. Bernoulli Naive Bayes is also used for discrete data, the presence or absence of a certain attribute or feature. In our dataset this includes all the binary variables such as whether or not the floors are good and whether or not the dwelling has a toilet.
4. Complement Naive Bayes classifier was designed to correct the "severe assumptions" made by the standard Multinomial Naive Bayes classifier. It is particularly suited for imbalanced data sets.

After running tests on all four Naive Bayes models, we selected the Complement Naive Bayes classifier to proceed with model turning. Complement Naive Bayes performed best among our Naive Bayes models which makes sense as it's designed to address the problem of imbalance datasets.

KNN

We ran a variety of KNN models with several hyperparameters we tuned:
1. Nearest Neighbors vs Radius: We considered whether a nearest neighbors approach, where we considered the k neighbors closest to a given point would be better than selecting a radius of distance around the closest point and considering all points in that radius would be more effective. We generally found that nearest neighbors was more effective, as especially larger radii defaulted to class 4
2. K: we experimented with different sizes of K: [3,5,10,15,20]. Generally speaking k values 10,15,20 were more effective.
3. Vote system: We considered uniform vs. distance-weighted voting. Generally both performed similarly, but distance-weighted voting was a slightly better fit for the data.

After considering these hyperparameters, we selected a 15-nearest neighbor distance-weighted model as our best KNN model to compare with the other best models we found. We would have liked to consider creating a custom voting system that devalued class four to address its over-representation but that proved challenging especially in light of other models providing more promising results.

4. Model Results and Recommendations

The core of our project was dedicated to predicting household poverty levels in Costa Rica by leveraging the Proxy Means Test methodology. PMTs play a crucial role in administering need-based programs, particularly for those who face challenges in providing the necessary documentation to demonstrate their need. The analysis was confronted with a significant hurdle - class 4 (the non-poverty group) was overrepresented in the dataset. This led to an increased likelihood of models predicting this class disproportionately to enhance their overall accuracy, a tendency we aimed to curb.

To navigate this issue, our selection criteria for the best model was driven predominantly by its recall performance on Class 1 (extreme poverty) and Class 2 (moderate poverty). In our context, recall - a metric indicating the proportion of actual positive cases the model correctly identified - served as an imperative measure. Given our mission to accurately classify households most in need, predicting Class 1 and Class 2 accurately became our prime performance indicator.

Using the method highlighted in section 3, the following reflects the outcomes of our out-of-the-box model performance as well as our tuned models. While there were different models that might have been selected as the top performer based on these scores, because we sought to have the highest performing recall of Class 1 and 2 while assuring that other models were fairly similar in terms of F1 (discussed below in considerations), we consider Logistic Regression our highest performing model.

4.1 Summary of Model Performance

Figure 1. Summary of Model Performance on Validation Data

Model	Base Accuracy	Final Model Accuracy	Δ Accuracy	Base Weight F1	Final Weight F1	Δ Weight F1	Base Macro F1	Final Macro F1	Δ Macro F1	Base Avg Recall Class 1 and 2	Final Avg Recall Class 1 and 2	Δ Final Recall of Class 1 and 2
Random Forest	0.68	0.59	-.09	0.62	0.62	0	0.36	0.42	.06	0.19	0.35	.16
Logistic Regression	0.66	0.59	-.07	0.56	0.61	.05	0.25	0.39	.14	0.06	0.37	.31
Naive Bayes	0.51	0.63	.12	0.50	0.60	.10	0.25	0.34	.09	0.29	0.29	0
KNN	0.61	0.66	.05	0.55	0.57	.02	0.28	0.28	0	0.09	0.08	-0.01

** Results return the average estimate across 5-fold cross-validations.*

4.2 Recommendations

Among various models tried, the Logistic Regression model stood out in performance. Notably, the use of the following hyperparameters and adjustments played a significant role in enhancing the performance of our Logistic Regression model:

- *Variance Threshold:* Adjusting the variance threshold helped to filter out features that lacked sufficient variability, improving the model's focus on more meaningful predictors of poverty. We used features that had a variance between 0.2 and 0.8.
- *SMOTE Oversampling:* Synthetic Minority Over-sampling Technique (SMOTE) helped to balance our imbalanced dataset, reducing the model's bias towards the overrepresented Class 4.
- *L2 Regularization:* This helped to prevent overfitting by penalizing large coefficients in the model, promoting generalizability.
- *Liblinear Algorithm:* Suited to smaller datasets.

4.3 Other Considerations

Our decision to emphasize the recall of Class 1 and 2 when selecting our model stemmed from the belief that it was crucial to correctly identify all individuals within these categories as living in poverty. We acknowledged the potential consequence of this approach, which might lead to a higher number of Class 3 and Class 4 households being incorrectly classified as living in poverty. Nevertheless, we saw this as an acceptable trade-off, driven by the goal to ensure that everyone living in poverty had access to the resources they needed, even if there was some degree of spillover.

In coming to this conclusion, we considered a few other potential metrics:

- *Precision for Class 4:* Precision penalizes false positives, thus this evaluation metric helps to avoid models that incorrectly classify households as not in poverty if they were indeed in need, thereby minimizing the risk of misallocated resources. A model with high precision for Class 4 would be more effective in ensuring that households not in poverty aren't mistakenly flagged as in need, thereby preserving resources for households that are genuinely in need.

- *Macro F1:* We considered Macro F1 largely due to its use as the evaluation metric in the Kaggle competition. By utilizing this metric, we were able to contextualize our model performance within the broader scope of the competition, providing a comparative reference point with other submitted models. Nonetheless, we found that the Macro F1 score, while valuable for a balanced performance assessment, did not perfectly align with our policy-oriented objectives.

4.4 Potential Alternate Model Structures.

When moving out of the Kaggle Competition structures, there are other potential models that are promising in terms of predicting poverty in Costa Rica:

Binary Model

The over-reliance on classifying class 4 poses significant policy challenges. One key goal of this project is to build a model that helps perform a Proxy Means Test and, as discussed in the introduction, it is most important to be able to classify households in poverty (Classes 1 and 2), as *this is the group that has the most difficult time demonstrating their need for services and the group for whom Proxy Means testing is most important.*

The address this challenge, we build a binary model that grouped households into two categories:

1. Experiencing poverty (1) which included households coded as:
    - 1 = extreme poverty
    - 2 = moderate poverty
2. Not Experiencing Poverty (0) which included households coded as:
    - 3 = vulnerable households
    - 4 = non-vulnerable households

Figure 3. Summary of Binary Model Performance

Model	Accuracy	Weighted F1	Macro F1	Recall
Random Forest (Binary)	0.8072	0.786	0.666	Recall Poverty: 0.352
Logistic Regression (Binary)	0.7843	0.73276	0.5574	Recall Poverty: 0.155
Naive Bayes (Binary)	0.729	0.74	0.673	Recall Poverty: 0.710
KNN (Binary)	0.7843	0.732	0.5574	Recall Poverty: 0.1551

The binary models performed well across the board. Unsurprisingly, given they are probabilistic models, Logistic Regression and Naive Bayes performed the best. Accuracy and F1 were higher compared to models which predicted the 4-class poverty measure, but perhaps more importantly, Logistic Regression and Naive Bayes did a significantly better job than any other models identifying people in poverty (classes 1 and 2), which was the main goal of this project. Meanwhile, KNN and Random Forest still leaned heavily on classifying 3 and 4 and thus may be less useful from a policy perspective.

Two-Stage Model

Despite extensive tuning of hyperparameters and the deployment of various strategies to manage class imbalance, we still struggled to build a model that didn't lean primarily on classifying only the overrepresented class, Class 4.

In an attempt to address models propensity to rely on the overrepresented class, we built a two-stage model to:

1. Classify 4
2. Then classify all not predicted as 4 as 1, 2, and 3

Figure 2. Summary of Two Stage Model Performance on Test Data

Model	Accuracy	Weighted F1	Macro F1	Recall	Precision
Logistic Regression, Random Forest	0.61	0.62	0.43	Class 1: 0.3 Class 2: 0.36 Class 3: 0.29 Class 4: 0.76 Avg. Recall for 1 and 2: 0.33	Class 1: 0.75 Class 2: 0.34 Class 3: 0.28 Class 4: 0.75

This allows us to predict non-vulnerable populations and then understanding there may be different features that categorize vulnerable populations and more nuance between classes with smaller bands of income difference in the vulnerable class, we strived to better predict this variation between the other 3 categories.

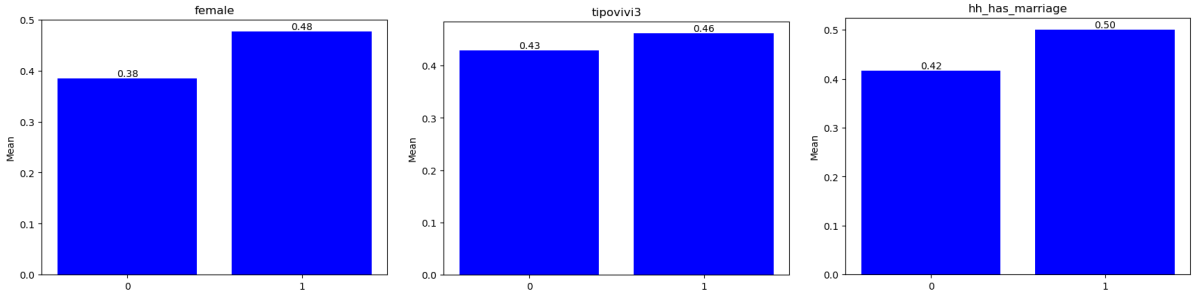
For this model, we had both the highest weighted F1, tied with Random Forest, and a fairly robust average recall of 1 and 2. Our precision model for model 1 was significantly higher than any of our models and while most of our high-achieving models just had good recall for 2 in previous models, it was heartening that there was relatively equal recall for both 1 and 2 in this model. Additionally, this model has the highest macro F1. As such, this model is promising to explore more. With more time to tune hyperparameters, we may be able to improve these results; however, given our goal is to create the highest average recall for 1 and 2, our logistic regression still reflects our best model for our goals.

5. Overcoming Limitations and Next Steps

5.1 Limitations

During our model development we encountered several limitations: limited data, difficulty distinguishing between target classes, and limited subject matter expertise that limited our ability to engineer new and meaningful features.

- *Limited Data:* Our project was constrained by the limited quantity of available data. The dataset contained only 3,000 households, and over half of those belonged to a single class. This not only left us with a small dataset overall, but also a disproportionately small amount of data to distinguish between classes 1, 2, and 3. In an ideal world, gaining access to a more detailed and expansive dataset, distributed more evenly across classes, could enhance the robustness of our model.
- *Difficulty Distinguishing Between Target Classes, Particularly 1, 2, and 3:* It is unclear whether this issue stemmed from the limited data available for these classes or if it was a problem inherent in the target classification scheme itself. As non-subject matter experts, we cannot definitively judge the validity of the target class labels. However, it is noteworthy that across multiple iterations of our model, we observed a recurring pattern: models were generally adept at distinguishing between Class 4 and the others, or at classifying everything as either 2 or 4. Typically, models struggled to differentiate between Class 1 and 2 or Class 3 and 4 with any substantial success. This pattern suggests there may not be sufficient differences between these groups to allow for effective classification.
- *Limited Domain Area Expertise Constrained Feature Engineering:* Time constraints limited the extent of our literature review, which might have impacted our selection of features. Given more time, a more comprehensive literature review could have allowed us to identify and include features that have been empirically shown to correlate more strongly with poverty status.
- *Model Performance by Household Characteristics*: to understand the implications of a potential real-world implementation of our best performing Logistic Regression model, we checked its performance by some of the available household characteristics. Poverty researchers in Costa Rica would likely want to understand if the model performed better for certain groups of people than others, for example. The graphs below display the accuracy of the model by sex of the head of household, whether the house is rented, and whether someone in the household is married.



We see that the model is 10 percentage points more accurate for female-headed households than for male-headed households, performs relatively comparably for renters versus owners, and is more accurate for households where someone is married compared to where there isn't a marriage. Examining accuracy by household characteristics was an exploratory activity we conducted at the end of our process, but government officials would want to pay close attention to these performance differences in a real-world application and minimize potential bias.

5.2 Additional Information

While working with the Kaggle dataset, we encountered several areas where we would have liked to gain more information. Additional information in the following areas may have been useful in feature engineering and evaluation of our model.

- *Domain Area Knowledge:* A deeper understanding of poverty in Costa Rica could have informed our approach significantly. Knowledge on whether the dataset is representative of the population, in terms of proportions and demographics, would have provided a better context for interpreting our model's results.
- *Quality of Data:* Information on the quality of the data, such as potential biases in data collection, could have helped us anticipate and account for any skewed patterns in the data. We know very little about how the data is collected, and if imbalance in the dataset represents actual imbalances in the population, or is the product of how data collection was conducted.
- *Discrepancies Between Household and Individual Level Data:* We encountered discrepancies between household and individual level data. A clearer understanding of how these discrepancies came about would have made it easier for us to consolidate the data accurately and effectively.
- *Impact of Covid-19 and Ongoing Changes:* Given that the data predates the Covid-19 pandemic, it would be interesting to understand how poverty has been impacted by the pandemic and its ongoing health, social and economic effect three years out. It's possible that the patterns we discovered in the dataset might have changed substantially. Additionally, in 2021, Costa Rica joined the OECD, and it will be interesting to see if their membership impacts poverty in the country.

5.3 Recommendations for Future Research

With more time and knowledge, there are several avenues we recommend for further exploration:

- *Revisiting Classification Categories:* One potential avenue of exploration would be to question the appropriateness of the four-class classification system we've used. It would be valuable to investigate if these categories align optimally with our policy goals. Perhaps there could be a more suitable classification schema that better captures the nuances of poverty, but determining this would require a deeper understanding of the domain.
- *Enhanced Feature Engineering:* This challenge could benefit from more robust feature engineering. This could potentially involve developing new composite features, performing more extensive feature selection, or exploring different ways to transform the data such as OneHot encoding for categorical values.
- *More Thorough Testing of Ensemble Methods:* Ensemble methods could potentially improve the performance of our models. We could further explore different types of ensemble methods.
- *Refinement of Current Models*: We could further refine the hyperparameters of our current models, including the ensemble model for predicting classes 1,2,3.
- *Exploration of Additional Modeling Techniques:* Given more time, we could explore other modeling techniques such as XGBoost or neural networks. These methods have their own strengths and could potentially offer improvements over the models we've used so far.