Costa Rica Multi-Class Poverty Prediction

Gregory Ho[*†]
MSCAPP
gregoryh@uchicago.edu

Jonathan Juarez
MSCAPP
jonathanjuarez@uchicago.edu

Sarah Walker
MSCAPP
swalker10@uchicago.edu

Yueyue Wang
MSCAPP
yueyue@uchicago.edu

Abstract

In this study, we propose and evaluate the effectiveness of four supervised machine learning models in predicting poverty levels in Costa Rican households. This research aims to augment policy-making and strategic planning by offering an efficient, data-driven approach to improve household well-being. Utilizing the proxy means test data from the Inter-American Development Bank, we benchmark the performances of Random Forest, Gradient Boosted Trees, Naive Bayes, K-Nearest Neighbors, and Logistic Regression models. Preliminary results indicate that the Random Forest and Gradient Boosted Trees models outperformed the other models, boasting an F1 score of 0.74 and 0.74, respectively, while the Naive Bayes, K-Nearest Neighbors, and Logistic Regression models had more modest F1 scores of 0.54, 0.65, and 0.65. The conclusion of this paper presents a comprehensive discussion of the successes and limitations of the different models in the context of poverty prediction.

1 Introduction

The ability to accurately predict the poverty levels of households is pivotal to the design and execution of public policies aimed at alleviating poverty and enhancing household well-being. This capability is particularly crucial in developing and implementing social welfare programs, where the objective is to target households experiencing varying dimensions and levels of deprivation.

The advent of machine learning techniques has spurred interest in harnessing computational methods to tackle complex socio-economic challenges, including poverty. However, the application of these techniques to predict poverty levels is not without its difficulties, often stemming from issues related to data availability, quality, and complexity.

In this study, we delve into the application of four supervised machine learning models—Random Forest, Gradient Boosted Trees, Naive Bayes, and K-Nearest Neighbors (KNN)—with the goal of predicting poverty status in Costa Rican households. These models were selected due to their varying characteristics and effectiveness at handling classification tasks in a diverse set of domains. We present a comparative analysis of these models to determine the most effective approach and contribute to the body of research in this domain.

The primary dataset employed in this study originates from the proxy means a test conducted by the Inter-American Development Bank, subsequently published on Kaggle, a platform renowned for

[*] Authors are listed in alphabetical order. All authors contributed equally to this work.

[†] Special thanks is due to Zander Meitus for his careful review and insightful suggestions, which significantly improved our work. All errors remain our own.

Preprint. Under review.

hosting competitions aimed at fostering model development for the common good. The richness of this dataset provides our models with a comprehensive source of information, enabling a robust analysis.

The organization of this paper is as follows: Section 2 offers an overview of the dataset and outlines the task setup. Section 3 delves into the methodology, detailing the specific models utilized and the experimental setup. Section 4 presents the results of our experiments, followed by a discussion on our findings and their implications in Section 5. The paper concludes with a summary of the key takeaways and potential avenues for future research.

2 Overview of Dataset and Task Setup

The primary dataset employed in this study originates from the Costa Rican Household Poverty Level Prediction competition on Kaggle. Sponsored by the Inter-American Development Bank, this competition seeks to enhance traditional poverty prediction methods by promoting the development and application of machine learning models. The dataset was compiled through a proxy means test[3], which comprises a mix of self-reported answers concerning household composition, educational outcomes, and observable physical characteristics pertaining to their housing conditions (e.g., overcrowding, roof type), and asset ownership (computers, electronic devices, among others).

The publishers of this dataset have made the implicit decision to embed these household-level characteristics at the individual level. Hence, each row in the dataset represents both individual characteristics and the household they belong to, rendering the dataset partially hierarchical. The target variables are ordinal variables representing 'extreme poverty', 'overall poverty', 'vulnerable households', and 'non-vulnerable households'.

2.1 Cultivating Domain Expertise

To contextualize this machine learning project, we initiated our study with a thorough literature review of the poverty metrics in the context of Costa Rica. This informed our understanding of the dataset and our approach to model building. We supplemented this with a socio-economic and demographic description of Costa Rica, as the Kaggle dataset was originally published in 2018. To align our understanding with the dataset, our review focused on pre-Covid studies conducted between 2014 and 2019.

Costa Rica, a Central American country, has a population of approximately 4.9 million. In 2016, it had a GDP per capita of USD 13,876, categorizing it as an upper-middle-income country. Roughly three-quarters of its population lives in urban areas. According to the World Bank, Costa Rica employs two income-based poverty lines [10]:

- **Target 1 - Extreme poverty** is defined based on the food poverty line index (fPLI), representing the amount a household requires per member to meet a minimum caloric intake. This category refers to individuals with a harmonized per capita income of less than USD 2.50 per day.
- **Target 2 - Overall poverty** considers other basic resources beyond food. This category includes individuals with a harmonized per capita income of less than USD 4.00 per day.

  Collectively, Targets 1 and 2 account for around 10% of Costa Rica's population.
- **Target 3 - Vulnerable Households** refers to households that were able to fulfill basic needs, but remain at risk of falling back into poverty due to unexpected circumstances. Approximately one-third of the population (36%, representing those with per capita incomes between USD 4.00 and USD 10.00 per day).
- **Target 4 - Non-Vulnerable Households** refers to non-vulnerable and non-poor households, constituting 49% of the population, earning above USD 10.00 per day.

[3] A proxy means test is a method of estimating household income or welfare level using observable characteristics of the household or its members. It is often used in contexts where reliable income or expenditure data is hard to collect.

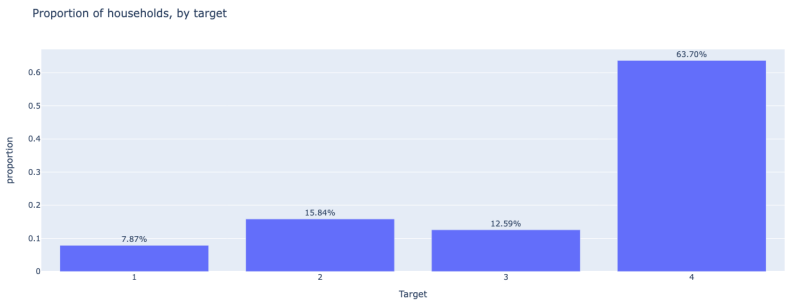

Figure 1: Household Proportion by Target Label

Spatially, poverty rates are lowest in the Central region, despite having the highest headcount poverty due to the large population (approximately half of Costa Rica's population resides in this region). Significant regional disparities exist, with notably higher poverty rates in border areas.

2.2 Exploratory Data Analysis and Pre-processing

Our initial step involved an exploratory data analysis to comprehend the structure and content of the dataset provided by Kaggle. This process revealed several discrepancies and inconsistencies[4]. Particularly, we noticed a large number of null values in the columns representing 'Monthly rent payment', 'Number of tablets household owns', and 'Years behind in school'.

Our general rule for handling null values is to generate contingency tables to determine whether or not null values exist as a result of aggregation errors, or if they were the result of survey non-response. In most cases, null values in this dataset were the result of aggregation errors presumably made when the publishers aggregated household-level features and embedded them unto individual-level rows.

Hence, since we know that the missing values in 'Number of tablets' and 'Years behind in school' were due to the aggregation of 0 values, we merely replaced each missing value with 0, effectively rectifying the issue.

On the other hand, handling null values in 'Monthly rent payment', required a more comprehensive approach. Null values for 'Monthly rent payment' only existed for households who:

- own the house they live in
- were living under precarious housing
- were living in assigned/borrowed housing

For these households, we imputed 0 monthly rent paid. However, in order to quantify the benefits of not having to pay rent, we developed an imputation strategy guided by the established guidelines from the United Nations Statistics Division (UNSD) [14, 6] [5].

2.3 Addressing Distributional issues

Our next step focused on analyzing the distribution of data within the dataset. We sought to understand the distribution of various features and the relationships between them. A key observation was the severe class imbalance exhibited by the target features (Figure 1). Among the four classes, **Target 4 - Non-Vulnerable Households** constituted approximately 63.7% of the dataset. This significant imbalance signaled the need for data balancing, ensuring that our machine learning models would not be unduly biased towards the majority class.

Addressing class imbalance In order to address the imbalanced class in the target variable, we use SMOTE (Synthetic Minority Oversampling Technique), which creates synthetic data points that

[4]We have outlined the specifics of these discrepancies and inconsistencies in our code, along with our pre-processing steps. These may be accessed at the following link: <>

[5]We provide a more detailed description in subsequent segments

are slightly different from the original data points[4]. This will allow an even representation of the poverty target variables and potentially increase predictive performance for each target without generating exact duplicates of the data. SMOTE re-balanced data is used after a model is chosen for further analysis.

3 Methodology: Picking and Training ML models

3.1 Feature Engineering

Generating new features based on domain expertise The primary dataset presented for analysis encompasses approximately 140 features. A comprehensive review of these features revealed substantial redundancy and overlap, suggesting the opportunity for effective feature engineering. A strategic conversion of these overlapping and redundant features into standalone variables was deemed necessary to enhance the coherence, readability, and efficiency of the dataset.

The feature generation process for our study was based on several critical dimensions, each contributing to a comprehensive understanding of the dataset:

Monthly Rent Payments The variable 'v2a1', representing monthly rent payments, is a key variable in Kaggle's dataset. It stands out as one of the main variables reflecting a portion of the household's monetary expenditure. However, our preliminary data checks revealed that there were a substantial number of missing values that had to be filled. To address this, we applied an imputation strategy guided by the established guidelines from the United Nations Statistics Division (UNSD) [14, 6].

- **Handling Missing Values in v2a1** We assigned 0 for households who owned the homes they live in ('tipovivi1'), households living under precarious conditions ('tipovivi4'), and households whose homes were assigned or borrowed ('tipovivi5').
- **Imputed Rent** Imputed rent represents the rent that homeowners would have paid if they had rented a house similar to the one they owned. For these same households who owned the homes they live in ('tipovivi1'), households living under precarious conditions ('tipovivi4'), or whose homes were assigned or borrowed ('tipovivi5'), imputed rent is computed as the median rent of other households within the same region and poverty level, further adjusted based on household size. This offers an estimate of the monetary savings derived from not paying rent.

Incorporating Equivalence Scales Larger households generally require more resources to maintain a comparable standard of living. However, the requirement doesn't scale linearly with the number of individuals, due to the economies of scale effect: individuals can share resources, leading to cost savings [8, 9]. To account for this effect, we apply a square-root scale to distributed value in our features, as recommended by Atkinson [3].

- **Mean Per Capita Income** For households who own their homes ('tipovivi1'), households living under precarious conditions ('tipovivi4'), or whose homes are assigned or borrowed ('tipovivi5'), we distribute the value added savings from imputed rent across household members. This distribution employs the use of equivalence scales, meaning that benefits gained are sub-linearly scaled based on the household size.

Poverty Dimensions Poverty has been recognized to be multidimensional and multifaceted[13, 2].Our feature generation also considered the multi-faceted nature of poverty. Specifically, we examined the following dimensions:

- **Housing Outcomes** We analyzed various factors related to housing, such as living conditions as represented by the physical conditions of walls, flooring, roofs, toilet system, water conditions, as well as whether the household faced overcrowding. We also quantified housing stability, and ownership status, to generate corresponding features.
- **Educational Outcomes** Acknowledging education as a key factor in poverty, we incorporated educational outcomes into our feature set, considering variables such as educational attainment and years of education lost.

- **Digitization and Access to Technology** Recognizing the role of digital access in modern society, we included features related to technology access and usage, such as access to and ownership of mobile phones, computers, and tablets.
- **Other Dimensions** We also examined various other aspects of poverty to generate additional features, ensuring a comprehensive and holistic representation of the dataset.

We undertook a rigorous process of generating new variables derived from the original dataset. This encompassed the creation of 25 new features at the household level, each designed as either ordinal or categorical variables. The newly constructed variables include:

- mean per capita income: Adjusted monetary savings from not needing to pay rent
- years of lost education: Total number of years of Household lost education
- mobilephone per capita: Ratio of mobile phones to household size
- computer per capita: Ratio of computers to household size
- tablet per capita: Ratio of tablets to household size
- asset owned: Assets owned by each household including refrigerator, computer, tablet, mobile phone, and television which is computed by the sum of the binary assets variables
- region: The region the household live in, which is encoded by the region binary variables
- wall, floor, and roof material: Three categorical variables indicating the material of the wall, floor, and roof by encoding related variables
- marital status: A categorical variable signifies whether an individual is single, married, divorced, or widowed, and another status by encoding related variables
- rubbish disposal: A categorical variable representing the method of rubbish disposal used by a household by encoding related variables
- water provision: A categorical variable representing the source of a household's water by encoding related variables
- electricity source: A categorical variable representing the source from which a household gets its electricity by encoding related variables
- wall, roof, floor status: Three categorical variables representing the condition or quality of the walls, roofs, and floors in a household by encoding related variables

Following the generation of these new features, the dataset was restructured, resulting in a final version containing 87 necessary features for model development and evaluation. With that in place, we then focused on the processing of specific categorical data. This was accomplished using a one-hot encoding technique. This approach to feature engineering and data processing has led to the creation of a more concise and robust dataset.

3.2 Evaluating Strength and Weaknesses of Classification Models

3.2.1 Logistic Regression

Logistic Regression (LR) was selected as our baseline model due to its simplicity, interpretability, and suitability for classification problems. LR is a statistical model that uses a logistic function to model a binary dependent variable. In the context of our study, the dependent variable is the poverty level of a household.

LR models the relationship between a set of independent variables, which in our case are the various socio-economic and demographic features and a binary target variable. This target variable represents whether a household belongs to a certain poverty level. By estimating the probability of a household belonging to a specific poverty level, LR allows us to make probabilistic predictions and understand the factors that significantly influence the outcome.

Mathematical Formulation In logistic regression, the probability of the outcome belonging to a particular class is modeled as a logistic function of a linear combination of input features:

$$p(Y = 1|X = x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{1}$$

LR exhibits several limitations that should be considered when utilizing it for analysis. In the case of this particular survey data, our examination has revealed certain issues that may impact its applicability. First, there is a potential violation of the independence assumption, which assumes that the observations are independent of each other. Additionally, the assumption of no multi-collinearity, where predictor variables are not highly correlated, may not hold true as exploratory analysis reveals several highly correlated features. Another limitation arises from the inability of logistic regression to handle newly introduced categorical variables, as the model can only accommodate continuous or binary variables. Although it is possible to transform these features into a continuous data type, we have deliberately refrained from pursuing this option when creating the LG model to maintain a focused analysis aimed at comparing baseline model accuracy at this stage.

3.2.2 K-Nearest Neighbors

Mathematical Formulation In k-nearest neighbor, a data point is classified by a majority vote of its neighbors, with the data point being assigned to the class most common among its K-nearest neighbors:

$$y = \arg\max_{c} \sum_{i \in N_k(x)} I(y_i = c) \tag{2}$$

The k-nearest neighbor algorithm was chosen to be tested due to its effectiveness with small to medium-sized datasets and its non-parametric nature, which makes no assumptions about the functional form of the data. We evaluated k-nearest neighbor and weighted k-nearest neighbor algorithms using the dataset with generated new features.

Our initial foray into this predictive task involved deploying an unweighted k-nearest neighbor model. After thorough experimentation, the model produced its optimal results when using 3 nearest neighbors. The F1 score, a metric used to assess the balance between precision and recall, reached a peak value of 0.7846.

Although the unweighted KNN model's performance was relatively satisfactory, there were concerns about potential overfitting due to the imbalanced nature of the dataset's target labels. As a mitigation measure, we turned our attention to a weighted KNN model, which differentially weighs instances based on their relevance. The aim was to offer minor labels, which were underrepresented in the data, a higher weight.

After tuning the hyperparameters, the weighted k-nearest neighbor model performed best with 5 nearest neighbors. This model yielded an F1 score of 0.6642, which, while lower than the unweighted k-nearest neighbor model, potentially offers a more realistic representation of the model's capacity to generalize to unseen data. The drop in the F1 score is indicative of the challenges inherent in handling imbalanced data.

Despite the performance of both the unweighted and weighted k-nearest neighbor models, we concluded that the k-nearest neighbor might not be the best-suited model for our poverty prediction task. The reasoning behind this conclusion stems from the 'curse of dimensionality', a phenomenon that affects performance and efficiency in high-dimensional spaces. Given our dataset's high-dimensional nature, this issue could lead to a decrease in model performance due to the increased sparsity of the data and the inflated distance computations inherent in the k-nearest neighbor approach.

To conclude, although k-nearest neighbor and its weighted variant are potent algorithms in certain contexts, their applicability to our poverty prediction task seems limited.

3.2.3 Naive Bayes

The Naive Bayes classifier was employed for its computational efficiency and suitability for datasets with high dimensionality. Naive Bayes classifiers are a family of simple "probabilistic classifiers"

based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods. It is particularly suitable for large datasets due to its linear time complexity, which is a significant advantage over other methods like support vector machines or more sophisticated ensemble methods.

Mathematical Formulation The naive Bayes classifier is based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features:

$$P(Y = y|X = x) = \frac{P(Y = y)\prod_{i=1}^{n} P(X_i = x_i|Y = y)}{P(X = x)} \tag{3}$$

Naive Bayes has similar limitations as logistic regression but performs even worse when there are strong correlations among the features. It requires making the 'naive' assumption that each feature is independent of one another - an assumption that is unrealistic in the context of poverty assessment. Poverty-stricken households often struggle with a myriad of interdependent deprivation sets which together form a self-perpetuating cycle that continues to weigh them down, restricting the prospect of having social mobility. This mutual reinforcement of adversities invalidates the assumption of independence, thus negatively affecting the performance of the Naive Bayes model in this particular context.

3.2.4 Decision Trees

Since Decision Tree algorithms split predictive features based on the information gained, they can be the simplest to understand. Additionally, Decision Trees are useful for filtering irrelevant features because they find splits that maximize the information gain [12]. This is important for the task of this project, especially since there are so many features present in the data. Decision Trees are also advantageous because they are very easy to build without very much data cleaning or normalization, and missing data does not greatly affect the ability to build the model.

Mathematical Formulation The Decision Tree algorithm first considers all features and calculates their entropy with the target column. Entropy is a measure of impurity, so it determines whether the target outcomes are highly varied (high entropy) or more homogeneous (low entropy) after the feature split:

$$Entropy = -\sum_{i=1}^{c} P(x_i) log_b P(x_i) \tag{4}$$

When building Decision Trees, we will split the feature with the lowest entropy at the given iteration. Lower entropy contributes to higher information gain about the feature's predictive power with the target class. We can determine the information gain from the split, and choose the feature with the highest information gain:

$$IG(Y, x_i) = Entropy(Y) - \sum_{v \in x_i} \frac{|Y_v|}{Y} * Entropy(Y_v) \tag{5}$$

However, Decision Tree models potentially overfit the training data, and may not be generalizable between samples (i.e., poor predictive power outside the training data). This also means that their models are unstable as small changes to the training data could change the entire structure of the tree. Another limitation is that decision trees cannot be fitted with categorical variables, thus certain features needed to be transformed into continuous and/or numerical.

From the cleaned data and using the featured engineered variables, we generate 4 decision tree models based on the following restrictions:

- first_tree uses all the features
- second_tree only includes features with at least +/- 15% correlation with the Target

- third_tree only includes features that represent individual-level characteristics
- fourth_tree only includes features that represent characteristics of the house

3.3 A comparison of Model Performance

It is important to determine a successful metric when analyzing the models. We decided on reviewing F1 scores as it offers a robust examination of accuracy since the original data is highly imbalanced at the poverty target variable. Selecting F1 when reviewing the models considers both precision and recall and provides a more comprehensive evaluation of a model's performance by considering the trade-off between correctly predicting positive instances and minimizing false positives and false negatives.

The tested model scores are given in the table below. The parameters column notes the parameters used to construct each model which helps to correct for over-fitting. C denotes the size of the regularization term and the penalty equates to ridge regression for LG to reduce over-fitness. The hyper-tuned weighted KNN finds 5 as the best k-nearest neighbor for predicting. For Naive Bayes, α is applied to prevent zero probabilities when estimating the conditional probabilities of the features, σ gives a portion of the largest variance of all features that is added to variances for calculation stability. The max depth parameter for the decision trees restricts the depth of the tree. Smaller depth leads to simpler and interpretable models. Larger values of depth allow the tree to grow deeper and capture more complex relationships in the data.

The results of these evaluations guided our decision on which model(s) to further optimize and evaluate in the subsequent sections. The decision trees have the highest accuracy and F1 scores compared to those given in the Kaggle competition, but this might change by implementing a random forest model that may prevent over-fitness.

Because of the importance of predicting classification targets for those in extreme poverty and overall poverty, choosing a model that has good performance at these ranges was important in deciding a model to further analyze and hypertune. Figure 2 visualizes the F1 scores of each model by poverty target and demonstrates decision trees are better at predicting all targets, including those of major concern. To address the decision tree models in over-fitting we created random forest models that help to minimize this issue.

Model Accuracy			
Model	Parameters	Accuracy Score	F1 Score
Logistic Regression	C = 0.01, penalty = 'l2'	0.66	0.65
Weighted KNN	K = 5	0.63	0.65
Naive Bayes	α = 10,σ = 0.01	0.60	0.54
Decision Tree 1	max depth=none, use all variables	0.74	0.82
Decision Tree 2	max depth=26, use correlated variables	0.74	0.74
Decision Tree 3	max depth=29, use household level variables	0.74	0.8

4 Random Forest Model Implementation

In order to create the final model for predicting a household's poverty class, we needed to overcome a few limitations. First, we label encoded our categorical variables into numerical values so that the decision tree algorithm could handle these features. Then, we used Synthetic Minority Over-sampling Technique (SMOTE) to address our data's class imbalance. Finally, we limited the number of features to consider and trained a random forest model to help mitigate overfitting. After comparing results from multiple decision tree models with various implementation methods, our final model was built from the random forest algorithm, trained on selected features from the dataset balanced using

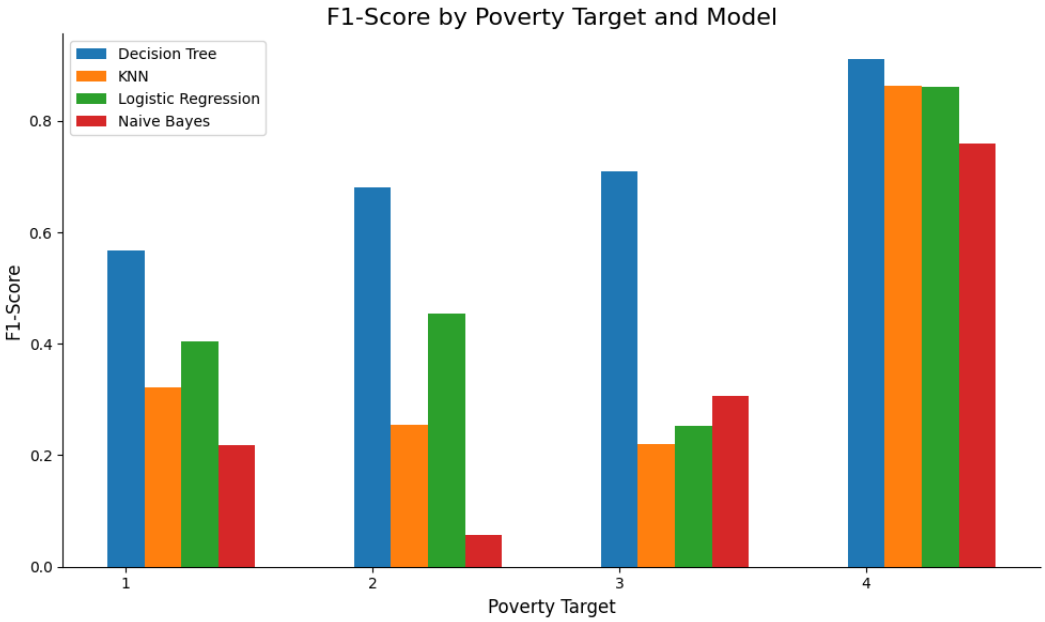

Figure 2: F1-Score by Poverty Target and Model

SMOTE. We considered this final random forest model to be the best for predicting the target class at the household level, especially for the most vulnerable households identified in classes 1 and 2. After testing on the test data, our accuracy results came to .74.

4.1 Label Encoding

Because decision trees are unable to train categorical data, we needed to consider how to transform the categorical features in our dataset into numerical representations that the decision tree algorithm could handle. One consideration was to use a one-hot encoding. Another option was to implement the label encoding algorithm, which simply reassigned each of the categories within a variable to a numerical value. Importantly, the numbers had no weight or ranking (i.e. they are not ordinal). Label encoding would allow the original feature to be used in building the decision tree by representing all of the categories as numbers.

We chose the label encoding method for two reasons. The first was that one-hot encoding increased the number of features to be considered in the training data set. Since our goal of data cleaning and feature engineering was to reduce and combine features into a smaller, more descriptive subset of variables, we did not want to reverse this work with one-hot encoding. The second reason to use label encoding was to help limit the depth of the decision tree. Since reducing the depth of decision trees is one use to prevent overfitting, we favored a non-binary tree that could be built around label encoded features, versus a binary tree which would result from one-hot encoding. Again, this method would reduce the number of features the tree would need to split, thus helping to limit the depth of the tree and mitigate overfitting.

4.2 Training/Validation/Testing Data

We separated our data into a train, validation, and test set using an 80/10/10 split. The training data would be used by the decision tree and random forest algorithms to construct the model. Each model would be tested with the validation set. This process would hyper-tune our parameters as we compared the accuracy of predictions made from each model against the validation set. Finally, once the final model had been chosen, we would run it on the testing set to see how well our final model performed. As a note, separating our data into training, validation, and testing sections is useful to mitigate overfitting. It is important not to use the full dataset to build the model so that it can be more generalizable when encountering new data.

Variable	Correlation to Target
yrs_edu_lost	-0.170611
r4h1	-0.186530
r4m1	-0.209479
SQBovercrowding	-0.219318
overcrowding	-0.234954
wall_material	-0.256747
SQBhogar_nin	-0.256824
r4t1	-0.260917
hogar_nin	-0.266309
v2a1	0.349605
escolari	0.333791
meaneduc	0.333593
SQBescolari	0.314397
cielorazo	0.295249
asset_owned	0.287570
SQBmeaned	0.276190
SQBedjefe	0.241272
instlevel8	0.235102
mean_per_capita_income	0.215767
tablet_per_capita	0.214161
rooms	0.201019
v18q1	0.197493
qmobilephone	0.168685
computer_per_capita	0.159212

Figure 3: Features with the Highest Correlation to the Target

4.3 Feature Selection

As mentioned previously, limiting the depth of the decision tree helps prevent overfitting. To try and achieve this, we considered feature selection to be a useful method. We wanted to restrict the full data set to a subset based on features with the most predictive power. We ran correlations of all the features with the target class and only included the variables with at least +/- 15 percent correlation to the target to train our model. Figure 3 displays the features used in the selected data set, along with their correlation to the target.

4.4 Synthetic Minority Over-sampling Technique (SMOTE)

The original data set had a high class imbalance, meaning there was an overrepresentation of households labeled as class 4, and an underrepresentation of class 1 and 2 households. Due to the nature of decision tree algorithms for splitting into majority classes, this imbalance skews our results to be more favorable to class 3 and 4 targets, but less able to accurately predict and identify class 1 and 2 households. Furthermore, this means that we only have a small sample of class 1 and 2 households, so it is difficult for the decision tree algorithm to learn characteristics that are highly predictive for identifying vulnerable households.

To address this imbalance, we implemented SMOTE on our data. SMOTE creates synthetic data points that are slightly different from the original data points [7]. This will allow an even representation of the poverty target variables and potentially increase predictive performance for each target without generating exact duplicates of the data.

Classification Report:

	precision	recall	f1-score	support
1.0	0.67	0.87	0.76	208
2.0	0.71	0.62	0.66	202
3.0	0.82	0.63	0.72	183
4.0	0.79	0.83	0.81	189
accuracy			0.74	782
macro avg	0.75	0.74	0.74	782
weighted avg	0.75	0.74	0.73	782

Figure 4: Classification Report for the Final Model ("select smote rf")

4.5 Random Forest

Implementing random forest is the final specification we included in the final model. Random forests are created from random subsets of data. Individual decision trees are constructed for each sample and an average result of the output is found. After considering many different trees (built using many random subsets of data), random forest selects the model with the majority ranking in results [11]. The random forest algorithm solves the problem of overfitting and also identifies features that are important for predicting the target class.

We know that random forest models will perform more poorly on the training and validation sets, but will make better predictions on the testing set as well as new data introduced to the model. This is a good sign that the model has not been overfitted to the training set alone, and is generalizable to new data. When we reviewed our results, we found this to be the case, with the random forest models scoring lower on precision and recall than models build without random forest. However, we still consider the random forest implementation to be a better choice for our final model because of its applicability to new data.

4.6 Selecting a Final Model

Going in order of complexity, we built and compared four models to determine which methods of implementation contributed to improving the accuracy of predicting household poverty levels. The main models we compared were built from the full data set, the selected data set only using the most correlated features, the selected data set with SMOTE, and the selected data set with SMOTE and random forest.

As mentioned previously, we were aiming to optimize recall in our model accuracies (more on this in the next section), and our model for the selected data set with SMOTE performed the best for all classes. However, we recognize that a classic decision tree is likely to be overfitted, so we perform a random forest to generalize what we have. We select our final model to be “select smote rf”: the model built from the random forest algorithm, trained on selected features from the dataset balanced using SMOTE.

After we selected our final model to be “select smote random forest”, we tested the results on the testing data. Our recall for each class (from 1 through 4) was .87, .62, .63, and .83, respectively. The f-1 score was .74, with an overall accuracy of .74. Importantly, our accuracies are roughly balanced across classes. (See Figure 4)

5 Accuracy Implications

When considering the use of a random forest model for social benefit targeting, reviewing recall scores is important, especially when identifying those in extreme poverty and those who are vulnerable. Various researchers that aim to classify extreme poverty in Indonesia note these items as important for proper inclusivity [1]:

- **Identifying the Most Vulnerable** Recall focuses on minimizing false negatives, meaning it emphasizes capturing as many true positive instances as possible. In the context of social benefit targeting, false negatives represent missed opportunities to provide support to individuals who genuinely require assistance. By prioritizing recall, the model aims to identify and include as many truly vulnerable individuals as possible in the target group.
- **Ensuring Inclusive Social Programs** By maximizing recall, the model aims to reduce the exclusion of individuals who may fall into extreme poverty or vulnerability. This is important for ensuring the inclusivity and effectiveness of social benefit programs, as leaving out eligible individuals could perpetuate inequality and hinder poverty alleviation efforts.
- **Mitigating Social Costs** By capturing a higher proportion of true positive instances, the model can help prevent cases where vulnerable individuals are overlooked or not included in the target group. This can mitigate the social costs associated with not providing adequate assistance to those who need it the most.
- **Targeting Limited Resources** In social benefit targeting, resources are often limited, and efficient allocation is crucial. Maximizing recall helps ensure that the available resources reach those who are most in need, improving the targeting efficiency and effectiveness of social programs.

We can see the importance of applying a high recall scoring model by reviewing a study conducted in Brazil that tries to determine the effectiveness of targeting poor individuals for social program benefits [5]. The Bolsa Famila program (BFP) aims to give cash transfer subsidies to individuals and households deemed poor and vulnerable. The BFP study aimed to evaluate the effectiveness of the current selection process by assessing the program's under-coverage and leak rates of the cash transfer program through a learning logistic model and measuring a focus indicator. Under-coverage measures the proportion of impoverished households that are mistakenly excluded from the program (exclusion errors). On the other hand, leaks occur when non-poor households receive program benefits (inclusion errors). Both types of errors introduce inefficiencies. The former deprives needy families of necessary resources, while the latter leads to wastage and potential scarcity of resources for the poorest families, necessitating a larger program budget to achieve the desired impact on poverty. The researchers found that there was under-coverage of cash transfers of 24-40%, depending on which poverty percentiles are selected when estimating the potential eligibility of cash transfers to applicants by comparing with data from Brazil's CadÚnico registration system. Inclusion errors were measured to be around 13.6% Overall, this study provides evidence that targeting those that are in extreme poverty using precise measuring techniques is particularly important if a program has budget constraints in delivering poverty alleviation.

6 Reflections and Takeaways

6.1 Handling imbalanced classes

Synthetic Minority Over-sampling Technique (SMOTE) played a crucial role in handling class imbalance. Our best classifier (decision trees) displayed substantial enhancements in both precision and recall of the under-sampled target classes, largely attributed to the application of SMOTE. (See Figure 5 for increases in precision and Figure 6 for increases in recall. The green and red bars indicate data used with SMOTE sampling versus the blue and yellow bars without SMOTE). This technique fostered the development of a more robust and accurate classifier.

6.2 Feature Engineering

Feature engineering can significantly contribute to the performance of a model. In particular, we have found that the engineered features "asset owned", "mean per capita income", "tablet per capita",

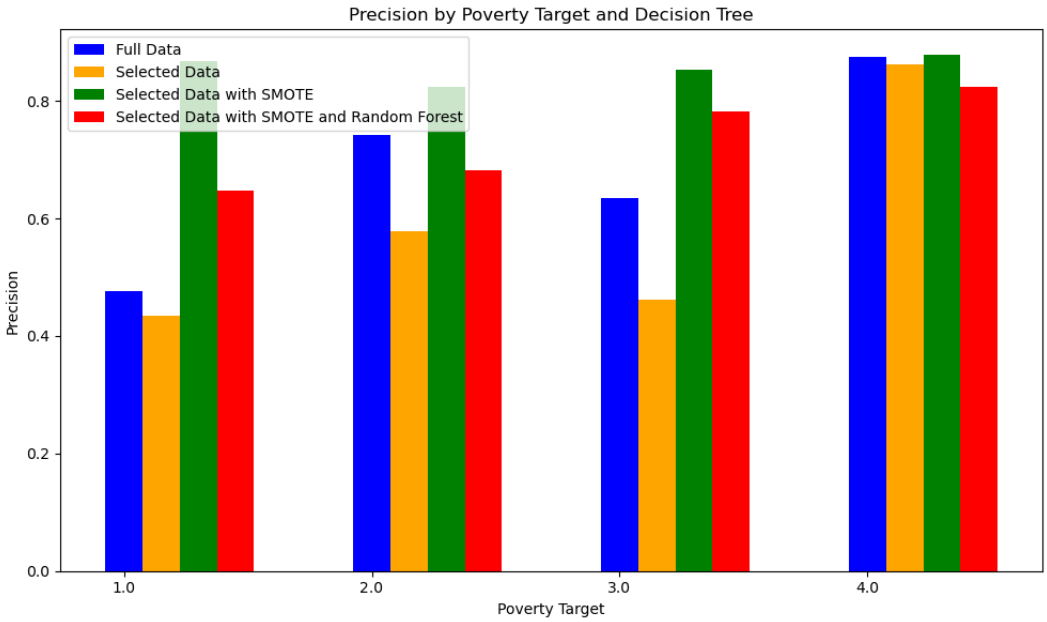

Figure 5: Precision by Poverty Target and Decision Tree

"computer per capita", "yrs edu lost", and "wall material" had at least a 15 percent (positive or negative) correlation with the target outcome. These were among the features we included in the selected model. Furthermore, after the best-fitting random forest model was selected, we were able to view the importance of the features included in the model. (See Figure 7 for the feature and its importance score generated from the random forest model). These results identified "asset owned" and "mean per capita income" within the top five most important features for predicting target class.

We also cleaned and imputed the monthly rent variable v2a1, which was the most important, or predictive, feature in our decision and random forest trees. So, our choices for filling null values in this variable were crucial not only for the predictive performance of our model but also for the real-world implications of predicting poverty class.

By utilizing domain-specific knowledge, we were able to craft features that enhanced model performance and enriched the existing dataset.

6.3 Using Machine Learning for Public Policy

In a span of slightly less than seven weeks, we have successfully showcased the viability of using Machine Learning methods in the realm of poverty targeting. The immense potential that Machine Learning harbors can lead to a significant increase in the effective utilization of resources. The experiment we conducted manifests that even models of relatively basic complexity can substantially equip decision-makers with the ability to pinpoint households requiring aid with remarkable precision. This approach optimizes resource distribution while simultaneously curtailing inefficiencies.

6.3.1 Machine Learning interpretability

If we had more time, we would have liked to explore the capability of Machine Learning models to also help describe and characterize the spectrum of deprivations (both general and specific) that is experienced by households at different levels of poverty. It is crucial that Machine Learning models be not only good and accurate predictors but also interpretable to policymakers. This attribute enables Machine Learning to serve as more than just a tool for efficient resource allocation, but also to contribute to providing a more nuanced understanding of the realities of poverty. With this, Machine Learning can support the formulation of more holistic public policies and interventions aimed at dismantling systematic barriers that impede social mobility and perpetuate poverty.

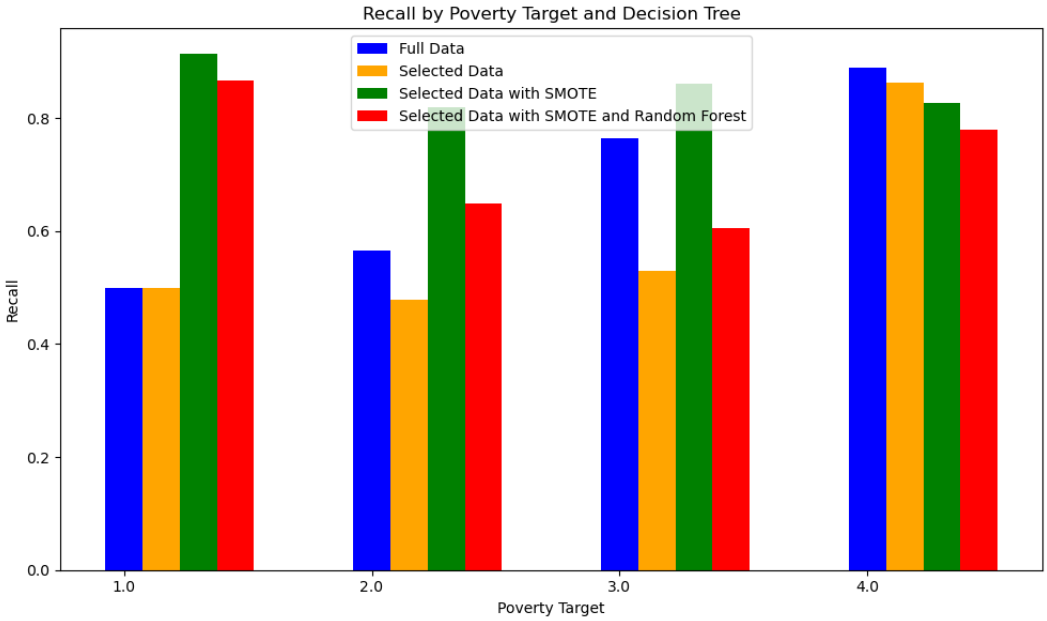

Figure 6: Recall by Poverty Target and Decision Tree

Varname	Imp
* v2a1	0.393229
* mean_per_capita_income	0.120004
SQBescolari	0.049391
* asset_owned	0.048275
cielorazo	0.047837
escolari	0.045338
SQBedjefe	0.038075
meaneduc	0.036185
qmobilephone	0.027671
SQBmeaned	0.027423
instlevel8	0.020179
SQBhogar_nin	0.019539
overcrowding	0.019345
SQBovercrowding	0.018991
* wall_material	0.016949
v18q1	0.016651
rooms	0.012289
r4t1	0.011516

Figure 7: Feature Importance in Forest Tree

References

[1] Vivi Alatas, Abhijit Banerjee, Rema Hanna, Benjamin A. Olken, and Julia Tobias. Targeting the poor: evidence from a field experiment in indonesia. *American Economic Review*, 102(4):1206–1240, 2012.

[2] Sabina Alkire and James Foster. Counting and multidimensional poverty measurement. *Journal of public economics*, 95(7-8):476–487, 2011.

[3] AB Atkinson, L Rainwater, and TM Smeeding. Income distribution in oecd countries.‖ oecd social policy studies, 1995.

[4] Jason Brownlee. *Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning*. Machine Learning Mastery, 2020.

[5] Juviliana Pereira Corrêa, Marcel de Toledo Vieira, Ricardo da Silva Freguglia, and Admir Antônio Betarelli Junior. Focus on cash transfer programs: assessing the eligibility of the bolsa família program in brazil. *Quality & Quantity*, pages 1–25, 2022.

[6] Marco Mira d'Ercole, Bob McColl, and Bindi Kindermann. Development of international guidelines and frameworks for micro statistics on household income, consumption and wealth. 2012.

[7] Joos Korstanje. Smote, 2021.

[8] Peter Lanjouw and Martin Ravallion. Poverty and household size. *The economic journal*, 105(433):1415–1434, 1995.

[9] Arthur Lewbel and Krishna Pendakur. Estimation of collective household models with engel curves. *Journal of econometrics*, 147(2):350–358, 2008.

[10] Ana Maria Oviedo, Susana M. Sanchez, Kathy A. Lindert, and J. Humberto Lopez. Costa rica's development: From good to better. systematic country diagnostic, 2015. License: CC BY 3.0 IGO.

[11] Shruthi E R. Understand random forest algorithms with examples (updated 2023), 2021.

[12] scikit learn. Decision trees, 2023.

[13] Amartya Sen. Development as freedom (1999). *The globalization and development reader: Perspectives on development and global change*, 525, 2014.

[14] United Nations. Economic Commission for Europe. *Canberra group handbook on household income statistics*. 2011.