Costa Rican Household Poverty

Ashwin Shukla and Yohan Jhaveri

Fall 2019

1 Abstract

Machine Learning has traditionally not been applied to econometric measures, such as the Proxy Means Test (PMT). This paper outlines how machine learning models can be used to determine these econometric measures, by utilizing a dataset from the Inter-American Development Bank and Kaggle. We present a novel pre-processing approach for the Kaggle competition that employs dimensionality reduction methods on both numerical and categorical data.

This paper attempts to present a systematic analysis of how different machine learning models perform on a multi-class classification task when given a dataset that contains hundreds of numerical and categorical features. We discuss how these models were optimized and evaluated. Through this project, we demonstrate how machine learning can be used to improve the classification of Costa Rican households into the four income levels specified by the PMT, allowing for a more accurate distribution of aid to the most vulnerable households in developing economies.

2 Introduction

The Inter-American Development Bank (IDB) is an organization that seeks to reduce poverty in the developing world, specifically in Latin America and the Caribbean.

In spite of receiving significant funding, the IDB has a difficult time in determining poverty levels of households and ensuring the impoverished households receive enough aid. This task is especially difficult as households are unable to provide the financial records or documentation required to qualify for this aid. As a result, the IDB has previously administered a Proxy Means Test that uses **observable household attributes** and material assets to determine household wealth. The Proxy Means Test assigns households to levels of poverty ranging from 1 to 4, where 1 is represents extreme poverty and 4 represents non-vulnerable households.

The Proxy Means Test is not completely accurate, however, and fails to distinguish between more subtle differences in household wealth (i.e. the test is only truly useful to distinguish households on the extreme ends: those that are wealthy and those that live in extreme poverty). It is in finding these subtle differences where machine learning techniques can be helpful.

This goal of this project was to develop machine learning solutions to improve the accuracy of the Proxy Means Test. Since the proxy means test predicts four levels

of poverty, the objective of our model was to predict the category of poverty that a household fell into (represented by 1, 2, 3, or 4). As a result, our project fell into the category of a supervised multiclass classification problem.

In setting out to develop these machine learning models, we have implemented a novel pre-processing and dimensionality approach for dealing with numerous categorical features. We have not seen this approach utilized in any public Kaggle kernels. By sharing our novel pre-processing approach with the broader community, we hope to contribute to the development of more accurate models for this problem.

This project is significant because the machine learning models developed here could be used to improve the lives of the Costa Rican people. The data that these models were tested on was collected from thousands of real people. By improving how aid is distributed to these people, we can reduce acute human suffering, which is perhaps the best use of machine learning technology.

3 Background

The IDB submitted the Costa Rican Household Poverty problem as a Kaggle Competition which received 618 solutions within a year.

In an attempt to understand previous approaches to the problem, we decided to consult a Kaggle notebook titled *A Complete Introduction and Walkthrough* [0]. This notebook presents the author's detailed approach to processing the data, selecting the model, and optimizing the solution in its entirety.

An interesting pre-processing decision the author made was to decode certain pre one hot encoded features and combine them into a single ordinal feature. For example, the dataset contained three dummy features that represented the condition of the walls as bad, regular, and good. Because an inherent ordering (bad $<$ regular $<$ good) existed for these features, the author combined these into a single feature where the values bad, regular and good represented by the values 0, 1 and 2 respectively.

The F1 score earned by the author was 0.43223, which is in the top 20% of all submissions. In comparison, the top scoring submission had an F1 score of 0.44878.

4 Methods

There were five broad categories of models that we selected:

1. **K Nearest Neighbors (kNN):**
   - Assumes that similar things exist in close proximity
   - Lazy and non-parametric model which simply memorizes the training data
   - Classifies test instances by finding the majority label of the **k** closest training instances (k nearest neighbors) to the test instances
   - The closeness is of two instances is calculated as a function (distance metric) of the difference in feature values of the instances.
   - Hyper-parameters: Distance Metric, k (sklearn)

2. **Tree Based Models**
   - Non-parametric models that prioritize features hierarchically in a binary tree like structure
   - Feature value conditions at each node pass data to respective child node based on the boolean output
   - Split condition features and values are chosen such that they maximize information gain at that node
   - Instances are passed down the tree based on their feature values until they hit a leaf node which classifies them as the majority class of the training points that reach that leaf
   - Hyper-parameters: Max Depth, Minimum Leaf instances (sklearn)
3. **Naive Bayes**
   - This novel uses Bayes theorem to determine the probability that a samples belongs to a class.
   - This classifier assumes that all features are unrelated to each other
   - Hyper-parameters: Class Prior, Fit Prior (sklearn)
4. **Ensemble models**
   - Combine the predictions from multiple standalone models to output a final prediction
   - Examples include a voting classifier and a stack

5 Experiments and Results

5.1 Data Description and Exploration

The dataset contained 9,500 unique instances with 142 total features, of which 102 were categorical and 40 were numerical.

The target class contained four class values, labelled from 1 to 4, where 1 represented *poverty*, 2 represented *moderate poverty*, 3 represented *vulnerable households* and 4 represented *non-vulnerable households*.

On further exploration, a class imbalance was found in the dataset. There were nearly twice as many instances with a label of class 4 as there were instances for classes 1, 2 and 3 combined. The instance frequencies of each class can be seen in Figure 1.

We also found a high degree of correlation among the numerical features present in the dataset. The pairwise correlations of the 16 most correlated numerical features can be seen in Figure 2.

Information gain (IG) measures how much "information" a feature tells us about the class. It helps us recognize the categorical features that are most useful in making predictions or categorizing classes. The categorical features that provided the highest information gain can be seen in Figure 3.

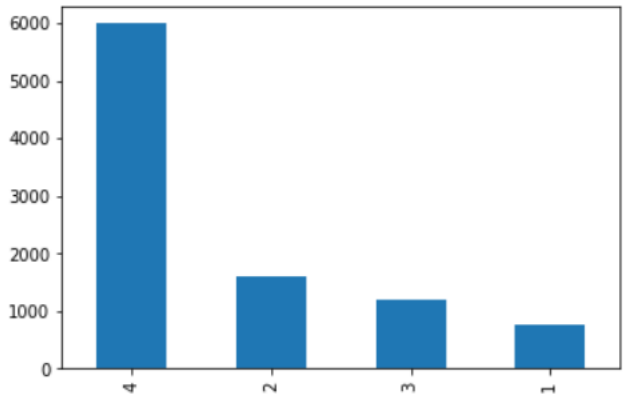

Figure 1: Class Imbalance

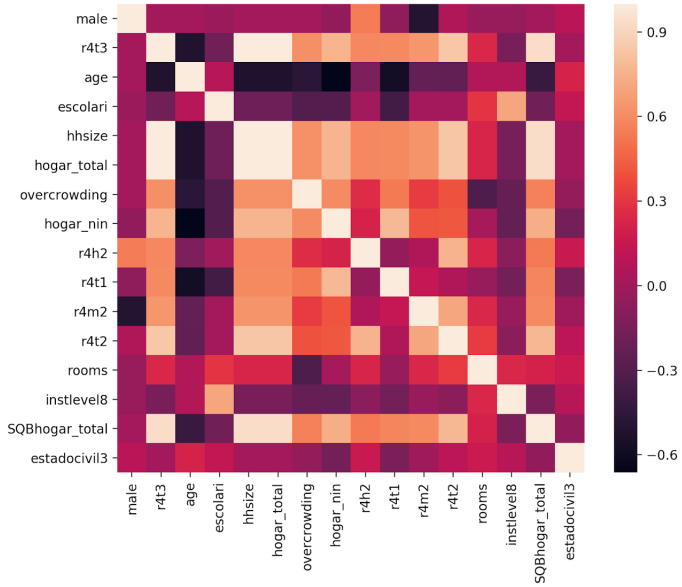

Figure 2: Pearson Correlation Matrix

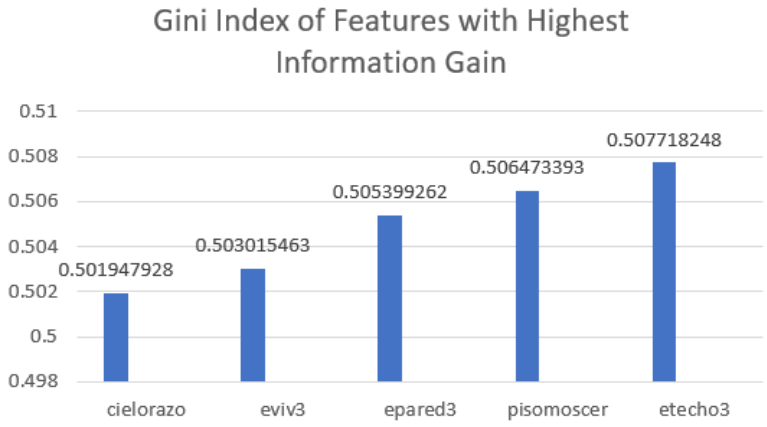

Figure 3: Information Gain

5.2 Pre-Processing

Extensive pre-processing was performed on the dataset to reduce the feature space. Methods ranged from manual processing, scaling, feature selection and dimensionality reduction.

5.2.1 Manual Processing

We recognized certain features to be redundant as they either had a large frequency of missing values, or contained a mix of categorical and numerical values. These features were manually removed for ambiguity and incompleteness.

5.2.2 Grouping into Households

Individual instances were combined into household instances using the Household ID column which identified the household that each individual belonged to. Each household in the household id column has a unique value with at least 1 individual belonging to that household. The individual feature values were combined into household feature values by taking the mean. This was accomplished using pandas' group by function on the idhogar column and taking the mean of the results of this function.

The 9500 present in the dataset were combined into 3000 household instances. This did not alter the class imbalance present in the dataset.

A struggle we faced while combining the data was that there were inconsistencies in labels corresponding to the same Household ID. To resolve this inconsistency, we found the head of the household using the binary feature **parentesco1** and used this individual's label to represent the label of the household.

5.2.3 Feature Scaling

We scaled the features based on the model we were training.

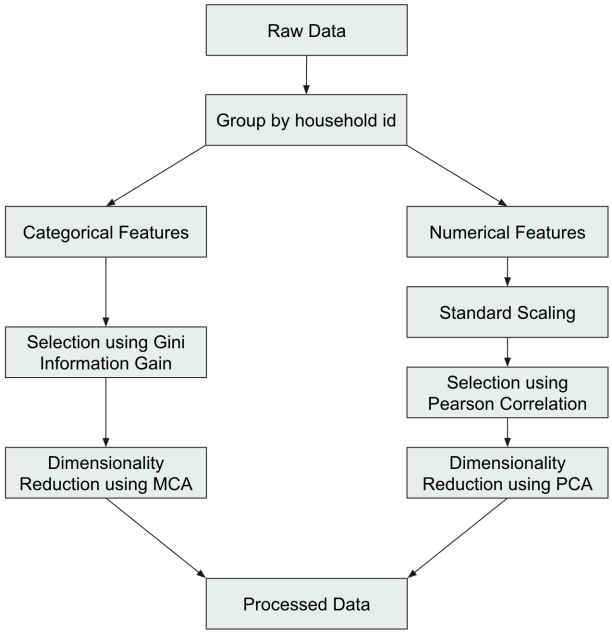

Figure 4: Data Processing Flow

- For models like K-Nearest Neighbors we used a custom-built standard scaler
- For models like Bernoulli Naive Bayes we used a custom-built min-max scaler
- For some models we did not perform any scaling.

5.2.4 Feature Selection

- Performed feature selection on numerical features using the Pearson Correlation Matrix by dropping features with high correlation.
- Performed feature selection on categorical features using the Gini Index by dropping features with low Information Gain.

5.2.5 Dimensionality Reduction

- Performed dimensionality reduction on numerical features using Principal Component Analysis (PCA)
- Performed dimensionality reduction on categorical features using Multiple Correspondence Analysis (MCA)

5.3 Model Selection Process

The models utilized were: K-Nearest-Neighbors, Decision Tree, Gaussian Naive Bayes, Bernoulli Naive Bayes, XGBoost, Random Forest, AdaBoost, a voting classifier ensemble (which utilized all of the models mentioned above), and a Stack that utilized Bernoulli Naive Bayes and Adaboost (which were 1st and 3rd best performing standalone models) as the first level estimators, and XGBoost (which was the 2nd best performing standalone model) as the second level.

We chose tree based models because they perform well on a mix of numerical and categorical data (our data set contains both numerical and categorical data). We wanted to get a benchmark for tree based models so we chose to use an sci kit-learn decision tree classifier. Given that boosted trees and ensembles of trees generally perform better than standalone decision trees, we chose to use XGBoost and Random Forest (an ensemble of decision trees)

We chose kNN because it can work with any type of probability distribution. Since did not know what kind of probability distribution the dataset would fall under (eg. Bernoulli, Gaussian etc.), this makes kNN a good model to consider.We also wanted to experiment with a unique non-parametric model that utilized non-linear decision boundaries.

We chose Naive Bayes because it tends to perform well on varied datasets and because it requires minimal parameter tuning. Additionally, Since Naive Bayes assumes that all features within a dataset are independent, it would be a good baseline to test the pre-processing steps we took to reduce correlation within our dataset. (i.e. Since Naive Bayes performs well when features are independent of one another, a highly performing Naive Bayes model would indicate that we had successfully reduce correlation among features in the dataset).

We implemented a voting classifier (ensemble) of all of the models that we had trained because it reduced the biases and variance of individual models. Voting classifier ensembles generally perform better than individual models.

We built upon the idea of using an ensemble and decided to use a stack to fine tune our multi-model approach. Stacking creates meta-learners that utilize the predictions from the previous layer of the stack. By utilizing the class predictions of the previous layers, the classifier in the final layer of the stack can make better class predictions.

5.4 Model Tuning

Models were tuned by optimizing hyper-parameters. Hyper-parameters were optimized using grid-search. This was accomplished using the built in sklearn grid search library. XGBoost was difficult to tune using sklearn grid search, so we implemented a custom implementation of grid search for the model that gave us more control over parameter tuning. See figure 9 for a list of final hyperparameters chosen for each model.

5.5 Model Evaluation

We evaluated the models using holdout validation. 30 percent of the data was held out to test the model (test data), while 70 percent of the data was used to train the model (training data).

The metric used to evaluate the models was a macro-f1 score.

$$P_M = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FP_i} \tag{1}$$

Equation 1: Macro Precision

$$R_M = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FN_i} \tag{2}$$

Equation 2: Macro Recall

$$F1_M = 2\frac{P_M \cdot R_M}{P_M + R_M} \tag{3}$$

Equation 3: Macro F1 Score

Using an F1-score accounts for the inherent class imbalance data. By taking a macro average of the F1-score for each class, we were able to examine the model performance on all four classes. This was also the metric used by Kaggle to evaluate submissions.

5.6 Classifier Performance

Confusion matrices for the top three best performing models have been provided. These matrices illustrate how certain models are better at predicting certain classes than others.

XGBoost performs best on instances that have a target label of 3 (class 3) when compared to the other models (Figure 5)

Adaboost performs best on instances that have a target label of 4 (class 4) when compared to the other models (Figure 6)

The voting classifier ensemble model performs best on instances that are class 1 or 2, when compared to the other models (Figure 7)

6 Discussion

A significant amount of time and effort went into pre-processing the dataset. The mix of categorical and numerical data made it tricky to process the entire dataset with the same methods and we therefore had to process the numerical and categorical features separately. Some of the novel methods we used were Multiple Correspondence Analysis (MCA) to reduce dimensionality in categorical features and Gini Index to select categorical features.

Implementing and utilizing the Gini Index in categorical feature selection was a novel approach to reduce dimensionality that we hadn't seen used before. This was

Predicted

Actual	XGBoost	1	2	3	4
	1	28	35	13	60
	2	42	80	33	168
	3	18	47	41	152
	4	31	67	59	1218

Figure 5: XGBoost Confusion Matrix

Predicted

Actual	Adaboost	1	2	3	4
	1	31	28	1	76
	2	43	73	15	192
	3	14	51	25	168
	4	27	67	19	1262

Figure 6: Adaboost Confusion Matrix

Predicted

Actual	Ensemble	1	2	3	4
	1	35	50	2	49
	2	64	97	19	143
	3	30	75	28	125
	4	37	86	41	1211

Figure 7: Ensemble Confusion Matrix

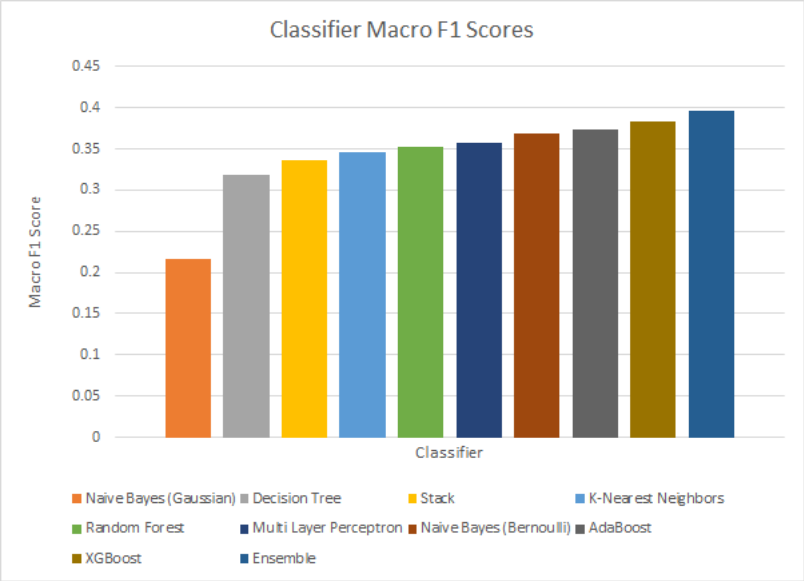

Figure 8: Classifier Performance (F1-Score)

Model	F1-Macro Score	Optimal Hyperparameters	Pre-processing
K-Nearest Neighbors	0.3449	k = 5	Standard Scaling, Gini, MCA
Decision Tree	0.3512	criterion = 'gini' max_depth = 20 min_samples_leaf = 1	Standard Scaling, Gini, MCA
Naive Bayes (Bernoulli)	0.3922	-	Min-max Scaling
XGBoost	0.3829	max_depth = 27 learning_rate = 0.1	Pearson, Gini
Random Forest	0.3523	n_estimators = 26 max_depth = 4 min_samples_leaf = 2	Pearson, Gini
AdaBoost	0.3737	learning_rate = 1.5 n_estimators = 49	Pearson, Gini, PCA
Ensemble	0.3967	models = BNB, XGB, ADA	Min-max Scaling, Pearson, Gini, MCA
Stack	0.3364	models = BNB, XGB, ADA	Pearson, Gini, PCA, MCA

Figure 9: Performance, Optimal Hyper-parameters and Pre-Processing approach by Classifier

inspired by the way the Decision Tree selects features to split at. We similarly used binary features to split the instances and then found the impurity and information gain using the Gini Index. This helped us recognize the features that provided the most information and insights into what the class may be and were likely the most useful features to training the models.

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2 \tag{4}$$

Equation 4: Gini Index

We experimented with making pre-processing specific to the model. For example we performed min-max scaling only for Bernoulli Naive Bayes or performed standard scaling only for kNN. We performed very little feature selection and dimensionality reduction for our Tree Based Models since they work well with highly co-linear data. Overall, pre-processing did improve the performance of most models especially when we made the pre-processing techniques specific to that model. Different amounts of pre-processing affected our model performance significantly and we learned that we needed to strike a balance in terms of how much we processed the data. In our attempt to reduce the feature space, we also learned that too much pre-processing can negatively impact model performance as it leads to information loss.

The structure of the data and characteristics of the models enabled certain models to perform better than others. (see Figure 8 and Figure 9 for a comparison of the f1 scores of all models)

The ensemble, which we picked to be our final classifier, performed the best out of all of the models. Its superior performance was due to the fact that it reduced both the bias and variance of the cumulative prediction by taking the majority of predictions from multiple base models. This combination of models resulted in increased predictive accuracy. The ensemble also performed best on classifying classes 1 and 2 which were particularly important to classify correctly since these were the households that represented the higher level of poverty and required the most aid.

Bernoulli Naive Bayes was the best performing standalone model. We assumed this was the case since one of the objectives of our extensive pre-processing was to eliminate highly correlated features. This reduced feature dependence in our dataset. Since independence is one of the primary assumptions of Naive Bayes, this boosted its performance.

XGBoost was the second best performing individual classifier with Adaboost a close third. These models have been known to perform extremely well on a variety of problems and their excellent performance on our problem was no surprise. When looking at XGBoost in the context of our problem, we believed that the characteristics of our dataset made Tree Based Models an ideal solution for our problem. This can clearly be seen from their overall performance. Tree based models are known to perform well on a mix of categorical and numerical data and when there is high correlation in the dataset, both characteristics that our data possessed. XGBoost performed better than Random Forest because it used to the statistical technique of gradient boosting to train new trees (within the model) based on the mistakes of previous trees (within

the model)

kNN performed badly because it was given a mix of categorical and numerical data. Given that there is no distance metric that can accommodate this kind of data, kNN naturally performs poorly on the dataset.

Decision Tree falls under the category of Tree Based Models. We discussed previously that models in this category were well-suited for our dataset. However, the Decision Tree performed poorly in comparison to its peers. Unlike XGBoost and Adaboost which generate an ensemble of boosted weak learners, the single Decision Tree is highly prone to overfitting. The small size of the final dataset worsened the case of overfitting.

The stacked model is difficult to implement and tune due to the number of model combinations and hyper-parameters to tune and therefore we were unable to optimize the model.

This project showed us the power of ensembles in supervised learning. Ensembles combine the strengths of each model while eliminating the variance and weaknesses of each model which would perform worse individually. It is significant in reducing overfitting and has a significant advantage over any individual model.

We also learned a great deal about pre-processing and the effects of performing too much scaling and dimensionality reduction on model accuracy.

A key takeaway from this project is a set of considerations and a framework for using big data approaches to analyze econometric data (from developing economies). These takeaways could allow for policy initiatives that alleviate poverty in Costa Rica.

7 Contributions

Each team member contributed an equal amount to this project. We worked on every part of the project together, in person. We made the presentation and spotlight together and practiced it together as well. There were were a few times where each of us worked on separate parts but the amount of work that one person completed was equal to the amount that the other person completed.

8 Link to Code

- Github Link: `https://github.com/YohanJhaveri/Costa-Rica-Housing-Poverty/blob/master/Costa%20Rica.ipynb`
- Drive Link (Backup): `https://drive.google.com/open?id=1r6LqPA9j34RStmq8iwOS-XgVKwdCBg9Q`

References

[1] Will Koehrsen: A Complete Introduction and Walkthrough,
`https://www.kaggle.com/willkoehrsen/a-complete-introduction-and-walkthrough`

Costa Rican Household Poverty Level Prediction: Kaggle
`https://www.kaggle.com/c/costa-rican-household-poverty-prediction`