CAPP 30254 Project Submission

Costa Rican Household Poverty Level Prediction

Pooja Verulkar and Gayathri Jayarman

OVERVIEW

The objective of this exercise is to build a machine learning model that predicts the poverty level of a household. The target variable has four income levels:

1 = extreme poverty
2 = moderate poverty
3 = vulnerable households
4 = non vulnerable households

The prediction process began with preliminary data exploration, followed by data cleaning to handle missing values and the application of feature engineering techniques to create representative variables for prediction. To achieve optimal prediction, three machine learning models were employed: Multilayer Perceptron, Logistic Regression, and K-Nearest Neighbors (KNN). The performance of each model was evaluated using various metrics to determine the optimal level of predictive power in terms of accuracy.

DATA EXPLORATION, CLEANING AND FEATURE SELECTION

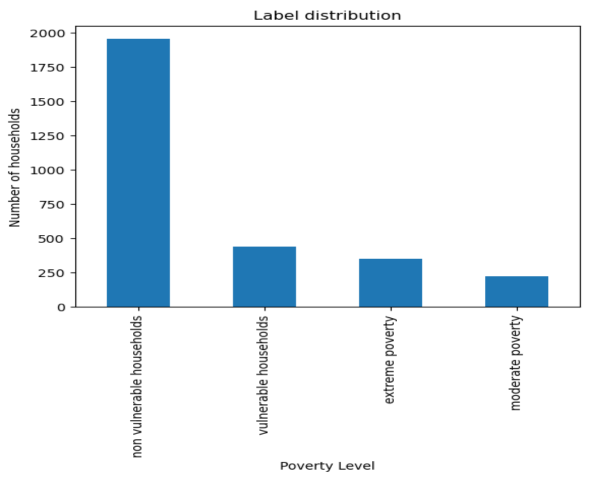
A. Summary Statistics and Features

The dataset comprised 139 features, including a mix of continuous, Boolean, and integer variables at both individual and household levels and data for 9557 individuals and 2988 households. Key summary statistics were calculated to understand the dataset's distribution:

- Age: Mean age was approximately 34 years.
- Monthly Rent: Average rent was 165,231 Costa Rican colóns.
- Household Size: Average size was 4 persons.
- Education: Individuals had an average of 7.20 years of schooling.
- Mobile Phones: Households had an average of 2.82 mobile phones.

As shown in Figure 1, the target variable is highly imbalanced, with 66% households under the non-vulnerable income category.

Figure 1. Label Distribution



B. Data Quality Checks

Basic sanity checks were performed on the data to ensure quality such as checking if all individuals with the same household have the same target label.

C. Handling Missing Values

As a preliminary step, the variables of the dataset were studied to identify missing values.

1. **Dropping Columns with High Missing Values:** The column rez_esc (Years behind in school) had 82.95% missing values, making it impractical to impute without compromising data integrity.

Table 1: Columns with Missing Values

Variable	Number of missing values	%
Years behind in school	7928	82.95
Number of tablets household owns	7342	76.82

Monthly rent payment	6860	71.77
Square of mean adult education	5	0.05
Mean adult education	5	0.05

2. **Imputing Missing Values with Zero:** For v18q1 (number of tablets household owns), missing values were treated as 0, considering the potential marginal impact on poverty determination. For v2a1 (monthly rent payment), missing values were addressed based on the type of house ownership. If a household owned a house (tipovivi1 = 1) or was paying in installments (tipovivi2 = 1), the rent was set to 0. This approach ensured that only households living in rented places had a rent value.
3. **Cross-Association for Imputation:** Missing rent values were cross-referenced with ownership status (tipovivi3). If a household was marked as owning the house (tipovivi3 = 1), the rent was set to 0. This method maintained consistency and logical coherence in the data.

FEATURE ENGINEERING

1. **Dividing features:** To make feature selection easier, data was divided into household level and individual level to explore the features independently and combine them after selection.
2. **Creating New Features:**

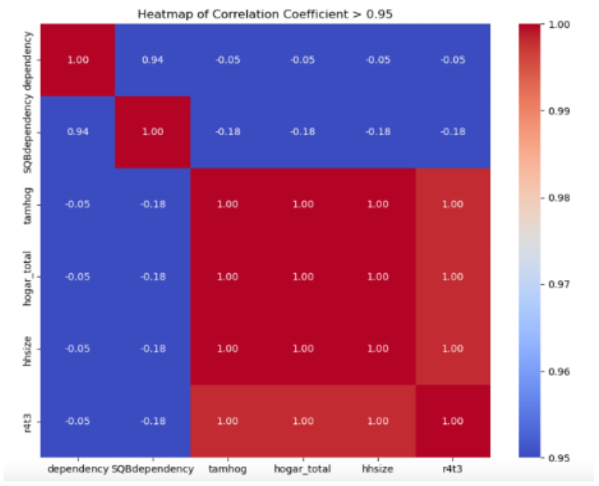
a. Boolean variables related to the same feature were compressed into ordinal variables to reduce dimension of the input. For example, abastaguadentro, abastaguafuera, and abastaguano were combined into a single variable *water_provision*. After creating the ordinal variables, the original variables were removed.

b. Few relevant variables were combined to generate a new feature, to improve predictive power of features. For example, variables for refrigerator, mobile, television and tablet were combined to create a feature that indicates if the household has extra facilities.

Using the individual level variables, new features were created by taking aggregated min, max, sum and average at household level.

3. **Removing features correlated with each other:** Features with high correlation with others can lead to multicollinearity, which can adversely affect the model's performance. To eliminate these redundancies, Pearson's correlation coefficient was calculated for all the pairs of features. Figure 2 shows the heatmap of few of the features. In the pairs with a correlation coefficient above 0.95, the feature with a weaker relationship with the target variable was removed, ensuring that the most significant predictors were preserved.

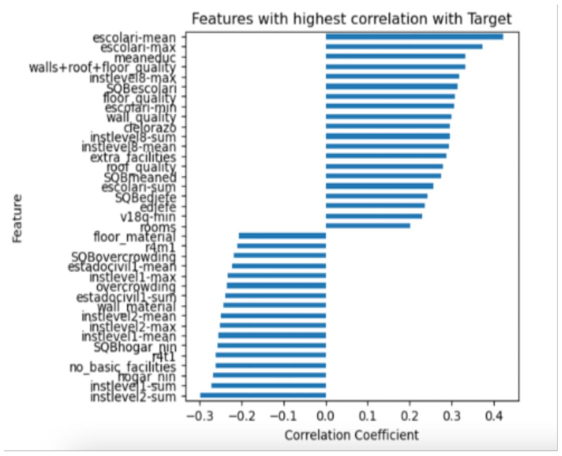
Figure 2: Heatmap of Features Correlation



4. **Feature Selection:** After removing the highly correlated features, household level and individual level features were combined. To select the best features from this set, they were evaluated for their relevance to the target variable (poverty level) using Pearson's correlation coefficient. household level and are individual level.

Figure 3 shows the top features with highest correlation with the target variable. Features with high positive correlation (more than 0.2) or high negative correlation (less than -0.2) were selected.

Figure 3: Correlation of features with Poverty Level



For instance, overcrowding and refrigeration were deemed irrelevant due to low correlation, while years of schooling and wall+roof+floor_quality showed high correlation and were retained. 38 features were selected through this process, out of which 22 are household level and 16 are aggregated individual level features.

MODEL IMPLEMENTATION

Choosing evaluation matrics

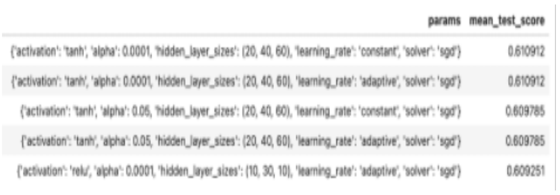
Given the imbalanced nature of the target variable, accuracy and F1 score were chosen as the parameters for assessing classification performance of the model. While accuracy is the ratio of correct labels and total number of labels, weighted F1 score calculates precision and recall for the entire dataset by summing up the true positives, false negatives, and false positives across all classes, and then uses these sums to compute the F1 score.

For training the models, data was split into 80% training dataset and 20% test dataset.

A. Multi-layer Perceptron

In addressing the non-linear nature of the target variable, a Multi-layer Perceptron (MLP) was employed to uncover complex patterns within the data, thereby enhancing predictive accuracy. To optimize the MLP, extensive hyperparameter tuning was conducted, focusing on the configuration of hidden layer size, activation function, solver, alpha, and learning rate. Figure 4 presents a comparison of the performance across different hyperparameter settings, revealing the most effective model configuration. This optimal model was subsequently validated using 5-fold cross-validation to ensure robust performance assessments.

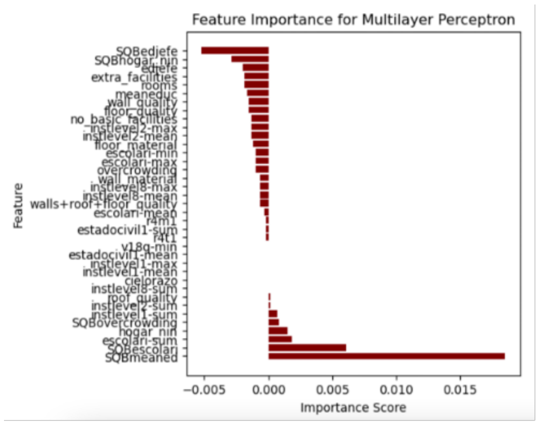
Figure 4: Performance of MLP for various hyperparameters



params	mean_test_score
{'activation': 'tanh', 'alpha': 0.0001, 'hidden_layer_sizes': (20, 40, 60), 'learning_rate': 'constant', 'solver': 'sgd'}	0.610912
{'activation': 'tanh', 'alpha': 0.0001, 'hidden_layer_sizes': (20, 40, 60), 'learning_rate': 'adaptive', 'solver': 'sgd'}	0.610912
{'activation': 'tanh', 'alpha': 0.05, 'hidden_layer_sizes': (20, 40, 60), 'learning_rate': 'constant', 'solver': 'sgd'}	0.609785
{'activation': 'tanh', 'alpha': 0.05, 'hidden_layer_sizes': (20, 40, 60), 'learning_rate': 'adaptive', 'solver': 'sgd'}	0.609785
{'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (10, 30, 10), 'learning_rate': 'adaptive', 'solver': 'sgd'}	0.609251

Additionally, an evaluation of feature selection strategies was undertaken to further enhance model performance. The importance and predictive power of various features were analyzed by testing multiple feature combinations. Figure 5 illustrates the importance scores assigned to each feature in the MLP model. However, since most features exhibited low importance scores, pruning features based on their importance did not yield any significant improvement in model performance.

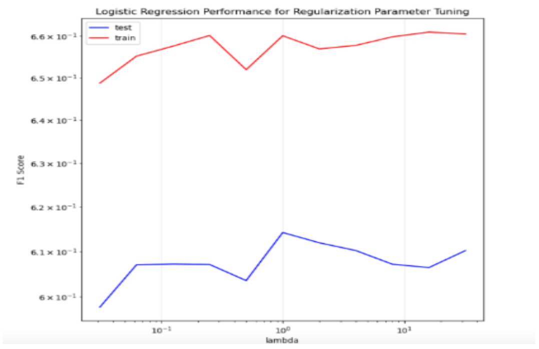
Figure 5: Feature Importance for Multilayer Perceptron



B. Logistic Regression

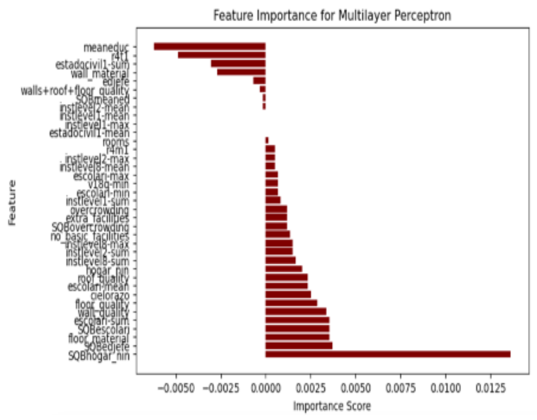
To understand linear relationships between the features and target variable, logistic regression model was implemented. We experimented with various regularization parameter values, ranging from 5 to -5 on logscale through hyperparameter tuning to identify the most effective setting. The best performing value of lambda was found to be 1. Figure 6 depicts how the model's performance varies with different regularization parameters. The figure indicates that changes in the regularization parameter have a minimal impact on model performance. Notably, the training performance consistently exceeds that observed on the test data, suggesting that the model is overfitting the training set and lacks generalizability. To better assess the model's performance in generalizing, 5-fold cross-validation was conducted using the optimal regularization parameter.

Figure 6: Logistic Regression Performance for Regularization Parameter Tuning



Additionally, feature selection strategies were explored to optimize model performance. The significance and predictive power of different features were assessed through the examination of various feature combinations. Figure 7 displays the importance scores for each feature within the MLP model. However, the features with low importance scores are very few, removing features based on their significance did not lead to any substantial enhancement in model performance.

Figure 7: Feature Importance for Logistic Regression

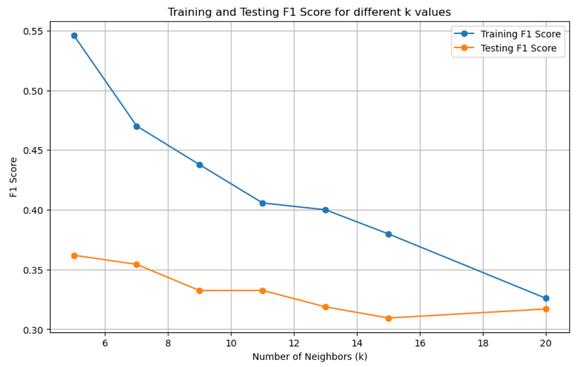


C. KNN

The K-Nearest Neighbors (KNN) algorithm was implemented to classify the dataset with the goal of predicting household poverty levels. The KNN algorithm was implemented on selected features consisting of household and individual data points. The implementation involved evaluating the model by hypertuning the parameters using various numbers of neighbors (k) ranging from 5 to 20. Accuracy and F1-score (macro average) were the primary metrics used to assess model performance, with the F1-score providing a balanced view of precision and recall, which is crucial for imbalanced datasets.

During hyperparameter tuning, it was observed that increasing the number of neighbors generally led to a decline in both training and testing F1-scores. The best performance on the testing data was achieved with 5 neighbors, resulting in an F1-score of 0.362. To ensure the robustness of this finding, a 5-fold cross-validation was conducted on the best model (k=5), revealing a mean test F1-score of 0.227 and a mean test accuracy of 0.429, compared to a mean train F1-score of 0.273 and a mean train accuracy of 0.489.

Figure 8: Training and Testing F1 for k values



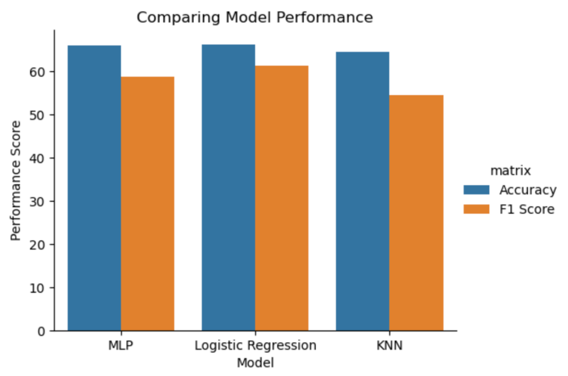
These results suggest that the model with 5 neighbors provided a good balance between bias and variance, although there was an indication of potential overfitting as the training scores were higher than the testing scores.

The cross-validation results indicated that the model might be overfitting to the training data, as evidenced by the higher training scores compared to testing scores.

COMPARING MODEL PERFORMANCE

After generating the optimized models using hyperparameter tuning, performance of all three models, Multilayer Perceptron, Logistic Regression and KNN was compared using the mean test F1 scores for cross-validation. Figure 9 shows the comparison of model performance.

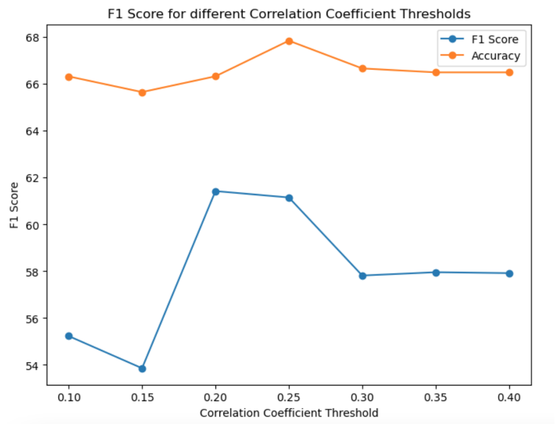
Figure 9: Comparison of model performance



With 66% accuracy and 61% weighted F1 score, logistic regression is the best performing model. Multilayer perceptron performed second-best with 66% accuracy and 58% weighted F1 score, while KNN was the least performing model with 64% accuracy and 54% weighted F1 score. Hence, a logistic regression model was chosen to perform the prediction.

Performance of Final Model

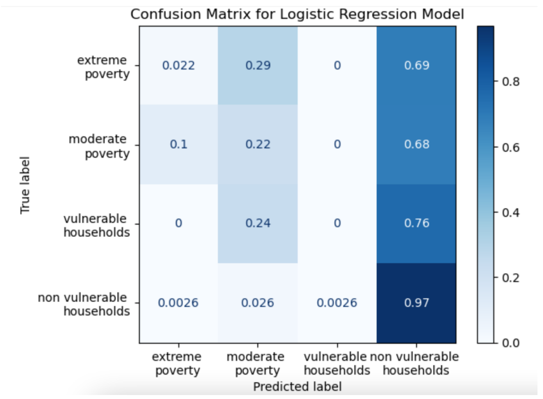
Figure 10: F1 Score for Different Correlation Coefficient Thresholds



To optimize the model, its performance was assessed based on various thresholds for the correlation coefficient used in feature selection. Figure 10 illustrates the model's performance across different correlation coefficient thresholds, indicating that a threshold of 0.2 yields the best results.

After finalizing the model, it was trained on the training dataset and then used to generate predictions for the test dataset. Figure 11 displays the normalized confusion matrix for the final logistic regression model with respect to the test data. The matrix shows that the model is highly effective at predicting the 'non-vulnerable household' income category, achieving 97% accuracy in correctly identifying true labels for this group. However, it incorrectly classifies over half of the labels from other categories as 'non-vulnerable household'. For these other categories, the model's accuracy in predicting true labels is less than 0.5%, highlighting a substantial need for improvement in its ability to accurately classify these categories.

Figure 11: Confusion Matrix for Logistic Regression



Overall, the results indicate that imbalanced classification challenges, especially with limited data, are notably difficult to address. Several strategies can be employed to mitigate this, including oversampling or training distinct models on various subsets of the data. However, ultimately, the most effective approach might be to collect more data.

Please find the github code here.