# Machine Learning for Poverty Prediction
## A Comparative Assessment of Classification Algorithms

**Casey A. Fitzpatrick**
DrivenData Inc.*
Berkeley, CA 94704
casey@drivendata.org

**Peter Bull**
DrivenData Inc.
Berkeley, CA 94704
peter@drivendata.org

**Olivier Dupriez**
The World Bank†
Washington, D.C.20433
odupriez@worldbank.org

## Abstract

This report compares the performance of various open source machine learning classification algorithms applied to the task of predicting the poverty status of households in Indonesia and Malawi. We use household survey data from a nationally-representative sample. A selection of categorical variables is used as predictors. The target variable, a label "Poor" or "Non-Poor," indicates whether a household's per capita consumption is below or above the national poverty line. Classical models such as logistic regression are optimized and compared to numerous modern approaches, including advanced tree methods, genetic algorithms for automated machine learning, and new approaches from deep learning. The results of the comparison demonstrate the trade-offs between computational complexity and model performance. We also discuss the use of a machine learning competition to crowdsource building predictive algorithms. This report aims to contribute to the democratization of machine learning by providing examples of good practice in the form of reproducible results and openly accessible documented scripts.

## 1 Introduction

Driven by advances in computational power, open source software development, and new statistical methods, machine learning has changed the way companies, governments, and organizations engage their data. This trend is coupled with greater availability of non-traditional data sources such as sensor data or satellite imagery, and together they enable more extensive and diverse applications of using data for decision making. This is particularly true in the field of development, where underexploited data can serve as an input for improving the design or targeting of policies and interventions. But machine learning is not yet routinely used by statistical or other agencies in developing countries. Providing well-documented scripts that are built on open source solutions and that are applicable to the types of data available across the globe is a step toward the democratization of machine learning approaches. This is the main objective of the project, funded by the World Bank Knowledge for Change Program (Grant TF0A4534).

This report presents the main findings of a comparative assessment of open source machine learning classification algorithms applied to poverty prediction. We explore how well machine learning algorithms perform in identifying "Poor" households in a given population, based on a set of qualitative variables. The assessment only covers classification algorithms used to predict a binary class. Poverty prediction could also be set up as a multi-class problem (for example, with labels "Very poor", "Poor", and "Non poor"), or as a regression problem (predicting the household per capita consumption, which would provide a flexible solution as it would allow users to apply different poverty lines). The scope of this assessment will be expanded in the future to cover such options.

*http://www.drivendata.co

†Additional analytical support for this project was provided by Tefera Bekele Degefu (World Bank).

All scripts (Jupyter Notebooks) used to generate the results presented in the report are openly accessible. They provide a valuable training resource on machine learning techniques.

Measuring poverty is typically done by collecting data on household expenditure, consumption, and/or income through household surveys. These surveys are costly, time-consuming, labor-intensive, and complex. In this project, datasets from two such surveys are used. Qualitative and categorical variables that are easy to collect are selected as features, and a binary class variable—"poor" and "non-poor"—is predicted. Engineering new features or building the best possible model is not the aim of this report. Rather, the focus is on comparing algorithms across multiple metrics.

### 1.1 Measuring Poverty Through Survey Data

Measuring poverty is typically done by conducting nationally-representative sample surveys that collect data on household income or consumption. These surveys are costly, time-consuming, labor-intensive, and complex. As a result, many countries conduct them infrequently, and on relatively small samples which limit the geographic and other disaggregation of the poverty estimates. Modeled estimates of poverty are often generated to complement the measured estimates. Tools like poverty mapping, small area estimation, survey-to-survey imputation, or proxy means testing are extensively used.

In this project, we use datasets from two such surveys: the National Socioeconomic Survey 2012, March (Survei Sosial Ekonomi Nasional - SUSENAS – 2012, Maret) from Badan Pusat Statistik Indonesia, and the Third Integrated Household Survey 2010-2011 from the National Statistical Office of Malawi.[3] From each survey, we extract a set of qualitative and categorical variables as features, and a binary class variable—"Poor" and "Non-poor"—is predicted. Households were assigned a status of "Poor" if their per capita consumption was less than or equal to the poverty line, and a status of "Non-Poor" if it was greater than the poverty line. Malawi was chosen because it has a relatively even number of "Poor" and "Non-Poor" households (47 percent poor, weighted estimate). Indonesia, on the other hand, has far fewer "Poor" households, with only 9 percent categorized as "Poor" (weighted estimate) in 2012 (see Table 1.4). This is known as an imbalanced dataset, and it requires special consideration when using machine learning approaches. The features were selected—somewhat arbitrarily—among variables deemed to be easy to collect. This includes information on household consumption, in the form of dummy variables indicating whether a household consumed or not each specific product or service listed in the survey questionnaire. Appendix A provides more information on the surveys and selected variables. We assess how, using this set of qualitative variables, we can predict household status.

The kind of models assessed in this project, through which we seek to predict the poverty status of households and not just the country's poverty rate, can have multiple applications. They could for example be used for survey-to-survey imputation, to add a "poverty status" variables in survey datasets in which only predictors are available. Or a statistical agency could consider using two versions of a questionnaire in their sample survey: a long questionnaire with full consumption or income module, administered to a subset of the sample and used to train a poverty prediction model, and a short form containing only the predictors but administered to the full sample. The cost of collecting data could be significantly reduced, allowing for larger samples and higher data disaggregation.

### 1.2 What is Machine Learning?

Machine learning techniques center on pattern recognition, relying on statistical models that map input data to some output. Inputs from various sources are routinely integrated in order to effectively leverage available data. Outputs can range from classical regression values or class labels to autonomously controlled vehicle movements. As models consume more data, they "learn" how to produce better outputs. (The meaning of "better" is a context-specific function of the metric used to evaluate the model, as discussed in Sec. 2.1) The process of extracting a mapping of input to output from data rather than writing down an explicit set of rules that maps the input to the output is

[3]For assessing the robustness of the predictive models built for Indonesia, we also make use of the SUSENAS 2011, 2013, and 2014 (March data). Information on the SUSENAS surveys is available at https://microdata.bps.go.id/mikrodata/index.php/home. Information on the Malawi data can be found at http://microdata.worldbank.org/index.php/catalog/1003.

the simple but powerful idea underlying machine learning. Classical statistical regressions——for example, logistic regression or linear regression——are examples of machine learning. However, over the last 20 years machine learning techniques have expanded beyond these classical statistical techniques, and in many situations the new techniques, many of which are explored in this report, are better at learning and representing the underlying structure in the data.

Machine learning holds great potential for social impact. Not only are the volume and diversity of available data greater than ever, but machine learning algorithms have never been easier or cheaper to use. Every algorithm considered in this report has an open source implementation, and most can be run on a basic laptop computer (made within the last decade). This means less money and less training can go further than ever before toward generating data-driven insights for organizations that need them the most.

### 1.3 Machine Learning Applied to Poverty Prediction

This report compares machine learning techniques applied to the problem of poverty prediction. As described above, the task is to predict whether a household is above or below the poverty line, given household and individual-level survey data. The type of problem is therefore known as a *classification* problem, meaning that the outputs are class labels: *Poor* or *Non-Poor*. This is opposed to a *regression* problem, in which the output would be some continuous variable, e.g., household income.

Producing a machine learning solution to a classification problem can be broken down into three parts:

1. Preprocessing
2. Training
3. Evaluation

Within each dataset considered, every model undergoes the **same evaluation** (3). That is, the same test data are used to generate class label predictions, and the same metrics are used to measure the quality of the predictions. A detailed explanation of these metrics and ranking scheme is given in Sec. 2.1.

Each dataset considered undergoes **mostly the same preprocessing** (1). The Malawi and Indonesia data are *split* in the same way, respectively, for every model, meaning that the same set of *training samples* are used to fit each model, and therefore the *same set of testing samples* are used to test each model. There are two exceptions to this. First, in the results of Sec. 2, the number of features varies across models for each dataset: some models use a reduced number of features while others use the full feature set, and the number of features is different for each dataset. Second, in the results of both Secs. 2 and 3, class-balancing techniques are applied to the Indonesia data—consequently, the number of samples (rows) used to train each model varies. This is because techniques to deal with class imbalance either remove data or add synthetic data as a means of balancing the class distribution to increase (decrease) a model's exposure to underrepresented (overrepresented) classes.

Finally, differences in **approaches to training** (2) are a primary focus of this report. An empirical assessment of machine learning algorithms applied to poverty prediction is offered, in which the learning algorithm plays the role of the independent variable while the evaluation plays the role of the dependent variable. Many model calibration approaches are considered. Some are specific to a certain class or subclass of model, such as "number of estimators" for ensemble methods, and others are available to every class of model, such as "cross-validation" to reduce over-fitting. More than 10 types of models are considered, and it is beyond the scope of this writing to present a thorough introduction to each model. However, as part of this project a set of open source Jupyter [15] notebooks was developed using the Python [22] programming language. The notebooks are divided by algorithm, and each notebook contains not only the code to generate the results reported here, but also a short introduction to each algorithm under consideration [26]. For a comprehensive introduction to every technique reported on here (and more), see [16, 6, 10].

### 1.4 What's In This Report

The following sections present results from an empirical comparison of various machine learning models, or, algorithms, applied to the survey data from Malawi and Indonesia. In Sec. 2, various well-known models are trained using standard sampling and validation techniques and then compared to each other across six metrics. The models explored all have convenient "out-of-the-box" implementations in standard Python programming libraries. "Out-of-the-box" here means that someone wishing to use the method does not need to implement the method from scratch. It is strongly recommended to use models implemented in widely-used libraries, since these models are well-tested and free from the mathematical, numerical, or programming errors that can plague implementations from scratch.

In Sec. 3, more advanced techniques are applied to the Indonesia (imbalanced) dataset, in an effort to explore the limits of the predictive power of the data, while introducing additional modern modeling techniques. First however, in Sec. 3.1, the results of a data science competition using the the Malawi dataset of this report are summarized and motivate the discussion of ensembling techniques. Next, Sec 3.2.1 presents a simple ensemble average of the top performing Indonesia models from Sec. 2. The results of a more advanced ensembling technique, *stacking*, using the same models and data are presented in Sec. 3.2.2. Sec. 3.3 presents the results of an automated machine learning pipeline based on genetic algorithms to the Indonesia data. Sec. 3.4 applies to the Indonesia data a recently developed deep learning model meant to take advantage of the categorical nature of the features.

Sec. 4 explores the misclassification trends across all models applied to the Indonesia data. This entails a closer look at model performance, the structure of the data, and choices in how performance can be assessed.

Finally, in Sec. 5, the top Indonesia models are applied to survey data from three other years: 2011, 2013, and 2014. Using these results, a final ranking of Indonesia model performance is presented.

## 2 Comparison of Standard Machine Learning Approaches

This section reports the results from an empirical comparison of various machine learning models applied to the Malawi and Indonesia datasets. Models are compared across six metrics, and applied to both the balanced (Malawi) and unbalanced (Indonesia) class distributions introduced in Sec. 1.4 and summarized in Table 1.4. The naming schemes for the models as well as the model classes considered are presented in Table. 2. For a more detailed yet hands-on introduction to the models considered in this section, see the Jupyter notebooks associated to this report [26]. For a traditional academic introduction, see [10].

The metrics by which the models are ranked are outlined in Sec. 2.1, and the model results are presented in Sec. 2.2.

### 2.1 Metrics

Classification accuracy—the percentage of observations for which a correct prediction is made—is often a misleading metric upon which to judge overall model performance. For example, given 100 total samples, 3 of which are class A and 97 of which are class B, a classifier can achieve 97% accuracy by predicting class B for every sample. The consequence of predicting a single false

Summary of Consumption Survey Data

| | Year Collected | Number of Households | Number of Features Including Derived (Original) | Percent "Poor" | Evaluation % |
|---|---|---|---|---|---|
| Malawi | 2010 | 12,244 | 484 (346) | 47% | 25% |
| Indonesia | 2012 | 70,843 | 453 (359) | 9% | 25% |

Table 1: High-level summary of the Malawi and Indonesia data used throughout this report. As the Percent Poor column makes clear, the Malawi data has a balanced class distribution, and the Indonesia data has an unbalanced class distribution.

Abbreviations Used In Model Naming Scheme Throughout This Report

| Algorithm Type | | Model Options | |
|---|---|---|---|
| Abbreviation | Algorithm | Abbreviation | Meaning |
| dl | deep learning network | full | trained using full feature set |
| dt | decision trees | | |
| knn | k-Nearest Neighbors | simple | trained using simplified feature set |
| lda | linear discriminant analysis | | |
| lr | logistic regression | cv | hyperparameters optimized using cross-validation |
| mlp | multi-layer perceptron | | |
| nb | naive bayes | wts | trained using sample weights |
| rf | random forest | | |
| svm | support vector machine | classwts | trained using class weights |
| xgb | extreme gradient boosted trees | ada | AdaBoost, additive learning technique |
| | | oversample | class balancing technique, increases dataset size |
| | | undersample | class balancing technique, decreases dataset size |
| | | isotonic | model calibration by isotonic regression |

Table 2: Throughout this report, algorithms are indicated by their model abbreviation (left columns), and further specified by one or more additional model options (right columns).

negative will vary with application, but in many contexts such as using a classifier to select households for social benefit programs, there may be a high cost for this error. It is therefore useful to measure model performance on metrics more sensitive to types of error.

We never expect a model eliminate all error. The "Poor" and "Non-Poor" labels can have errors stemming from erroneous reporting and data capture that we cannnot anticipate with the model, and the structure of the problem as binary classification means that households close to the poverty line will be difficult to characterize definitively in one bucket or the other. Our goal is instead to reduce the error as much as possible and, through application of many approaches, get a better sense of the error inherent in the data that a model cannot overcome.

Throughout this report, models are compared across six different metrics, defined below. To provide an overall model ranking, each model's *mean rank* across the six metrics is computed.

**Mean rank** The *mean rank* is calculated in two steps. First, $N$ models are ranked according to $M$ metrics, producing $N \times M$ values where value $n_m$ corresponds to model $n$'s rank according to metric $m$. Model $n$'s mean rank is then $(n_1 + ... + n_M)/M$. Models can be roughly sorted according to this aggregate statistic, making it a reasonable first pass for gaining an overall sense of performance across models. Note that the value of a model's mean rank depends on the number of models $N$ it is being compared to. Throughout this report, any table with a mean rank column also parenthetically states $N$ to offer a sense of an upper bound on this quantity.

Most of the metrics used to calculate mean rank rely on the widely-used *confusion matrix notation* (a confusion matrix is sometimes called a *contingency table*). This notation is reviewed before introducing the metrics.

**Confusion matrix notation** To simplify expressions, confusion matrix notation is used whenever possible. For binary classification problems with "positive" and "negative" classes, a confusion matrix is $2 \times 2$ and consists of diagonal elements $TP$ and $TN$, representing *true positives* (correctly classified as positive) and *true negatives* (correctly classified as negative) respectively. The off-diagonal elements $FP$, and $FN$, represent the *false positives* (incorrectly classified as positive) and *false negatives* (incorrectly classified as negative) respectively. The total number of samples that contribute to the confusion matrix is the sum $S$ of samples classified, where $S = TP + TN + FP + FN$. The number of positive samples $P$ is given by the sum $P = TP + FN$, and the number of negative

samples $N$ is given by the sum $N = TN + FP$. Throughout this report, "Poor" is taken to be the positive class and "Non-Poor" is the negative class.

**Accuracy** In classification tasks, accuracy provides a measure of the proportion of samples that were correctly predicted. As mentioned above, this quantity can be misleading, especially in cases where the class distribution is imbalanced. However, it remains useful because of its interpretability. In confusion matrix notation, the accuracy, $A \in [0, 1]$, is given by

$$A = \frac{TP + TN}{P + N} \tag{1}$$

**Recall** Also called the *true positive rate* and *sensitivity*, *recall* is the ratio of correctly classified samples to the total number of samples in that class. It can be thought of as a probability of detection (of the positive class). The recall, $R \in [0, 1]$, is given by,

$$R = \frac{TP}{P} \tag{2}$$

Recall is particularly useful in cases of imbalanced classes. In the example at the beginning of this section, the classifier predicts all samples to be class B and achieves 97% accuracy. Its recall, however, would be 0 (as opposed to, say, a recall of 0.667 if it has predicted 2 of the 3 instances of class A to be class A).

**Precision** Also called the *positive predictive value*, *precision* answers the question, "Of all the identified positive instances, how many identifications were correct?" Precision, $PPV \in [0, 1]$, is given by

$$PPV = \frac{TP}{TP + FP} \tag{3}$$

As the number of the false positives increases, the precision decreases. A classifier may achieve high recall by predicting many false positives. As such, precision a useful counterbalance to recall.

**F1** The *F1* score is the harmonic mean of precision and recall, as defined in Eqs. 3 and 2. $F1$ ($F1 \in [0, 1]$) is given by

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{4}$$

It is commonly used in cases where both the precision and recall of a model should be considered.

**Cohen's Kappa** This quantity provides a measure of accuracy that controls for the accuracy of a random classifier using a term called expected accuracy. Expected accuracy is defined as the accuracy that a random classifier would be expected to achieve based on the confusion matrix of predictions. Cohen's kappa statistic, $\kappa \in [-1, 1]$ can be calculated by

$$\kappa = \frac{A - EA}{1 - EA} \tag{5}$$

where the *expected accuracy*, $EA$, is given by

$$EA = \frac{(TP + FN)(TP + FP) + (TN + FP)(FN + TN)}{TP + FN + FP + TN} \tag{6}$$

and the accuracy $A$ is given by Eq. 1. The Cohen's kappa statistic can be thought of as a normalized accuracy, where normalization occurs with respect to $EA$. The maximum $\kappa$ of 1 indicates complete agreement between classifier performance and ground truth, where a $\kappa \in [-1, 0]$ indicates chance agreement.

**Cross Entropy** The cross entropy metric is not defined using confusion matrix notation because it involves the probabilities underlying a classified sample, if available (many classifiers predict class probabilities, and the most probable class taken to be the predicted class). For binary classification with $N$ samples, if the true label $y_i$ is predicted with probability $p_{y_i}$, the cross entropy score, $H \in [0, \infty)$ takes a simplified form,

$$H = -\sum_{i=1}^{N} (y_i \log(p_{y_i})) + (1 - y_i) \log(1 - p_{y_i}) \tag{7}$$

This metric is particularly well-suited for penalizing highly confident *wrong* answers, as can be seen from the second term when the correct class $y_i = 0$ is incorrectly predicted to be class 1 with high probability $p_{y_i}$. Therefore, unlike any of the other metrics considered here, the *lower* the cross entropy the better.

**Receiver Operating Characteristic Area Under Curve** The *receiver operating characteristic* (ROC) curve is one of the most widely used tools for evaluating classifiers, especially visually. The curve is constructed by plotting the recall, or, true positive rate $TPR$ (i.e., probability of detection), defined in Eq. 2, against the false positive rate (i.e., probability of false alarm), defined as $FPR = FP/(FP+TN)$, as a function of the threshold $T$ used by the classifier to map probabilities to classes. This means that a single confusion matrix, computed using a particular classifier threshold and yielding a single $TPR$ and $FPR$, produces a single point in the ROC curve associated with that classifier. For perfect classification, the ROC curve will consist of a false positive rate of 0 and a true positive rate of 1, regardless of the threshold. This leads to an area of 1 under the ROC curve integrated with respect to the false positive rate. For the worst possible classifier, the opposite is true: the false positive rate will be 1 and the true positive rate will be 0, leading to an area of 0 under the same integration. $ROCAUC \in [0, 1]$ is formally given by

$$ROCAUC = \int_0^1 ROC(TPR, FPR, T)\, d(FPR) \tag{8}$$

**Poverty Rate Error** Another metric that is not systematically used throughout this paper but can nevertheless be informative, is the *poverty rate error*. Using the sample weights associated to each household, a poverty headcount estimate can be constructed by first making predictions on the entire dataset, then retrieving the sample weights for each positive sample (household labeled "Poor"), summing them to get a poverty estimate, and finally subtracting the true poverty headcount from the estimate to obtain an error measure. Even for models with relatively low accuracy and recall, this error can be surprisingly small, so even models with "Poor" predictive power at the level of individual samples may be useful for national-level estimates. It is worth noting that because this rate is an aggregation of predictions, errors in opposite directions can cancel each other out.

Each of the above metrics, with the exception of the non-standard poverty rate error, is computed for every classifier considered in this report. For poverty rate error performance, see Appendix C.

Throughout the report, in order to rank the classifiers across all metrics, the mean rank computation is used. Due to its simplicity, the mean rank sometimes leads to ties, but it still gives a sufficient idea of overall classifier performance.

### 2.2 Comparative Results on Malawi and Indonesia Data

The outcomes for the Malawi data are shown in Tables 3, 4, and 5. The outcomes for the Indonesia data are shown in Tables 6, 7, and 8. These are based on exploring model building across a wide range of modeling and preprocessing choices. In Tables 3 and 6, a simple subset of features from the survey data was used and models were trained without any preprocessing and default hyperparameters. In Tables 4 and 7, all available features but no class-balancing or cross-validation techniques were used to train the models. In Tables 5 and 8, the techniques listed in Table 2 were used in model training. Table 5 will be useful in comparing the results for Malawi for the top competitors in the competition, which is discussed in Sec. 3.1. Table 8 will be useful for comparison against the more advanced algorithms presented in Sec. 3.

A full list of models and their results is available in the Appendix B. For all of the results reported in this section and the rest of this report, the top ten best performing models from Appendix B is used, and mean rank *relative to these top 10* is computed. (For this reason, the upper bound on mean rank is larger in Appendix B than it is in the rest of the tables in this report.)

In order to understand differences in algorithm performance, each algorithm class was fit with default hyperparameter choices on a reduced dataset. The reduced dataset has a small number of well-understood features that appear in the data for both countries: number of children under ten years old, number of males over ten years old, number of females over ten years old, number of literate members of the household, number of household members employed in last year, if the household has electricity, if the household does not have a toilet, if the household consumes chicken, and if the household consumes beef. These results serve as a baseline, which helps illustrate the strengths and weaknesses of various metrics. Comparing the results of the full data approach with the baseline evidences the benefit of more sophisticated modeling decisions.

The next step is to use all of the available features in training each model. While these models are much more complex (and likely less interpretable given that there are 484 features in the Malawi dataset and 453 in the Indonesia data, and issues of multicollinearity), the inclusion of all features improves the models' ability to separate "Poor" from "Non-Poor" households.

In addition to comparing different algorithm types, this report assesses the added benefit of a set of preprocessing techniques outlined in Table 2. For Malawi these include hyperparameter optimization through cross-validation, isotonic calibration of model results, and the use of AdaBoost [9] to do gradient boosting on an ensemble of fitted models. Techniques for Indonesia include undersampling and oversampling to create more balanced populations of "Poor" and "Non-Poor" households for the algorithms to learn from.

### 2.2.1 Malawi

**Simple Features** For Malawi (Table 3), the best performing model is the multi-layer perceptron, a simple artificial neural network. This algorithm achieves a classification accuracy of 77%. For most of the metrics that have a range of $[0, 1]$, simple models fall around 0.75, which means they get about 3 out of 4 cases "correct," where "correct" is defined differently for each metric. Note that precision and recall are often in the same range. The worst performing model according to the mean rank is a decision tree.

**Full Features** For Malawi, as shown in Table 5, the maximum accuracy increases from 77% to 87%. Likewise, all other metrics have improved for the top models. The best model according to mean rank is a support vector machine, and again the worst performing model is a decision tree. It is worth noting that logistic regression, which is relatively simple compared to other algorithms, has consistently fallen in the top three models so far.

**Full Features and Tuning** In Malawi, the best performing algorithm is a support vector machine that uses all of the features as well as cross validation for hyperparameter optimization (Table 5). It is worth noting that the mean rank for the best performing model is 5th, and within the top 10,

Malawi Results Using Simple Feature Set

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=11) |
|---|---|---|---|---|---|---|---|---|
| mlp_simple | 0.773 | 0.784 | 0.733 | 0.757 | 0.468 | 0.854 | 0.545 | 1.750 |
| dl_simple | 0.768 | 0.801 | 0.718 | 0.757 | 0.482 | 0.851 | 0.536 | 2.625 |
| lr_simple | 0.767 | 0.739 | 0.743 | 0.741 | 0.479 | 0.848 | 0.529 | 3.750 |
| xgb_simple | 0.765 | 0.753 | 0.733 | 0.743 | 0.477 | 0.848 | 0.526 | 3.750 |
| svm_simple | 0.763 | 0.776 | 0.721 | 0.748 | 0.488 | 0.841 | 0.526 | 4.750 |
| lda_simple | 0.763 | 0.735 | 0.739 | 0.737 | 0.488 | 0.845 | 0.522 | 5.625 |
| lr_logit_simple | 0.757 | 0.663 | 0.767 | 0.711 | 0.483 | 0.847 | 0.503 | 7.125 |
| nb_simple | 0.740 | 0.729 | 0.705 | 0.716 | 0.528 | 0.805 | 0.476 | 8.500 |
| knn_simple | 0.741 | 0.722 | 0.710 | 0.716 | 2.137 | 0.799 | 0.478 | 8.750 |
| rf_simple | 0.738 | 0.715 | 0.708 | 0.711 | 1.105 | 0.809 | 0.471 | 9.000 |
| dt_simple | 0.732 | 0.679 | 0.714 | 0.696 | 3.023 | 0.781 | 0.457 | 10.375 |

Table 3: Malawi model performance across six metrics, using only a small subset of features. Models are sorted by their *mean rank*, as defined in Sec. 2. See Table 2 for list of abbreviations, Table 1.4 for summary of data.

Malawi Results Using Full Feature Set

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=10) |
|---|---|---|---|---|---|---|---|---|
| svm_full | 0.864 | 0.886 | 0.868 | 0.877 | 0.298 | 0.945 | 0.733 | 2.625 |
| xgb_full | 0.860 | 0.886 | 0.862 | 0.874 | 0.322 | 0.937 | 0.732 | 3.375 |
| lr_full | 0.874 | 0.870 | 0.854 | 0.862 | 0.288 | 0.949 | 0.746 | 3.625 |
| dl_full | 0.860 | 0.930 | 0.834 | 0.879 | 0.602 | 0.930 | 0.721 | 4.125 |
| lda_full | 0.859 | 0.884 | 0.863 | 0.873 | 0.327 | 0.934 | 0.719 | 4.750 |
| mlp_full | 0.853 | 0.852 | 0.877 | 0.864 | 0.669 | 0.940 | 0.717 | 5.750 |
| rf_full | 0.830 | 0.860 | 0.835 | 0.848 | 0.392 | 0.920 | 0.650 | 6.500 |
| knn_full | 0.780 | 0.960 | 0.727 | 0.827 | 0.753 | 0.890 | 0.527 | 6.750 |
| nb_full | 0.794 | 0.859 | 0.785 | 0.821 | 2.472 | 0.873 | 0.509 | 8.000 |
| dt_full | 0.779 | 0.782 | 0.809 | 0.795 | 3.577 | 0.825 | 0.568 | 9.500 |

Table 4: Malawi model performance across six metrics, using all available features. Models are sorted by their *mean rank*, as defined in Sec. 2. See Table 2 for list of abbreviations, Table 1.4 for summary of data.

meaning no single algorithm and preprocessing pipeline consistently performs the best across all metrics. In fact, there is far less consistency in a pipeline's performance relative to the other pipelines it is ranked against in the table. That said, 3 distinct support vector machine pipelines have shown up 3 times in the top 10, and XGBoost pipelines have appeared twice. Given that these model classes performed well across all of the iterations, these are worth considering as a first approach for predicting household level poverty where there is close to a balanced dataset of "Poor" and "Non-Poor" households.

### 2.2.2 Indonesia

**Simple Features** For Indonesia, an imbalanced dataset, the ordering of models is very different (Table 6). This is an initial reminder of Wolpert and Macready's "no free lunch" theorems [25], one of which states "what an algorithm gains in performance on one class of problems is necessarily offset by its performance on the remaining problems." Or, more simply, there is not one algorithm that performs best across all classes of problems. While accuracy is much higher for these models, inspection of the other metrics indicates that none of the models is actually performing that well. A model that identified every household as "Non-Poor" would be 91% accurate, but the recall would be 0. The deep learning model in particular does exactly this, and while it optimizes to reduce cross entropy by making less confident predictions, recall, precision, and f1-score are all 0. Unlike Malawi, the models using simple features for Indonesia perform quite poorly at identifying "poor" households.

**Full Features** For Indonesia, as shown in, Table 8, the logistic regression model performs the best with 91% accuracy, which is not substantially better than the results from the simple feature set. However, performance against other metrics indicates that while the algorithm correctly classifies the same percentage of overall households, it is substantially better at identifying "Poor" households, with a recall of 0.46 instead of 0.09.

**Full Features, Class Balancing, and Tuning** Indonesia shows similar results, as shown in Table 8, with even less consistency for particular algorithms across pipelines across metrics (the top scoring

Top 10 Malawi Results with Full Feat., Class Balancing, Tuning, CV, and Feat. Selection

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=35) |
|---|---|---|---|---|---|---|---|---|
| svm_full_cv | 0.874 | 0.894 | 0.878 | 0.886 | 0.287 | 0.949 | 0.758 | 5.000 |
| mlp_full_cv | 0.871 | 0.895 | 0.874 | 0.884 | 0.278 | 0.952 | 0.752 | 6.125 |
| xgb_feats | 0.872 | 0.892 | 0.877 | 0.884 | 0.289 | 0.949 | 0.754 | 7.375 |
| svm_full_cv_isotonic | 0.871 | 0.891 | 0.876 | 0.883 | 0.288 | 0.949 | 0.754 | 7.625 |
| lr_full_wts_cv | 0.873 | 0.892 | 0.879 | 0.885 | 0.301 | 0.944 | 0.734 | 7.750 |
| xgb_full_cv | 0.869 | 0.894 | 0.870 | 0.882 | 0.296 | 0.948 | 0.751 | 9.125 |
| svm_full | 0.864 | 0.886 | 0.868 | 0.877 | 0.298 | 0.945 | 0.733 | 10.625 |
| lr_full | 0.874 | 0.870 | 0.854 | 0.862 | 0.288 | 0.949 | 0.746 | 12.750 |
| rf_full_wts_cv_ada | 0.866 | 0.878 | 0.878 | 0.878 | 0.580 | 0.947 | 0.744 | 13.000 |
| dt_full_wts_cv_ada | 0.866 | 0.878 | 0.878 | 0.878 | 0.353 | 0.941 | 0.737 | 13.000 |

Table 5: Malawi model performance across six metrics, using all available features and various class-balancing techniques as well as cross-validated training. Models are sorted by their *mean rank*, as defined in Sec. 2. See Table 2 for list of abbreviations, Table 1.4 for summary of data.

Indonesia Results Using Simple Feature Set

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=9) |
|---|---|---|---|---|---|---|---|---|
| lr_simple | 0.912 | 0.085 | 0.564 | 0.148 | 0.257 | 0.782 | 0.127 | 2.750 |
| lda_simple | 0.907 | 0.165 | 0.459 | 0.243 | 0.263 | 0.779 | 0.205 | 3.000 |
| mlp_simple | 0.911 | 0.080 | 0.552 | 0.140 | 0.254 | 0.786 | 0.120 | 3.250 |
| xgb_simple | 0.911 | 0.054 | 0.597 | 0.099 | 0.257 | 0.782 | 0.085 | 4.375 |
| dt_simple | 0.903 | 0.128 | 0.380 | 0.191 | 1.316 | 0.657 | 0.153 | 5.250 |
| nb_simple | 0.907 | 0.067 | 0.401 | 0.115 | 0.275 | 0.735 | 0.091 | 5.625 |
| knn_simple | 0.909 | 0.057 | 0.446 | 0.101 | 0.694 | 0.712 | 0.082 | 6.500 |
| svm_simple | 0.910 | 0.003 | 0.833 | 0.006 | 0.294 | 0.565 | 0.006 | 6.625 |
| dl_simple | 0.910 | 0.000 | 0.000 | 0.000 | 0.278 | 0.769 | 0.000 | 7.625 |

Table 6: Indonesia model performance across six metrics, using only a small subset of features. Models are sorted by their *mean rank*, as defined in Sec. 2. See Table 2 for list of abbreviations, Table 1.4 for summary of data.

Indonesia Results Using Full Feature Set

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=9) |
|---|---|---|---|---|---|---|---|---|
| lr_full | 0.910 | 0.456 | 0.662 | 0.540 | 0.213 | 0.923 | 0.483 | 2.500 |
| mlp_full | 0.908 | 0.584 | 0.607 | 0.595 | 0.503 | 0.927 | 0.546 | 2.750 |
| lda_full | 0.906 | 0.405 | 0.648 | 0.499 | 0.231 | 0.912 | 0.457 | 3.875 |
| svm_full | 0.902 | 0.208 | 0.782 | 0.329 | 0.204 | 0.932 | 0.312 | 4.250 |
| knn_full | 0.904 | 0.372 | 0.647 | 0.472 | 0.541 | 0.865 | 0.423 | 5.125 |
| xgb_full | 0.898 | 0.184 | 0.743 | 0.295 | 0.224 | 0.917 | 0.285 | 5.750 |
| nb_full | 0.807 | 0.603 | 0.322 | 0.420 | 1.893 | 0.828 | 0.238 | 6.000 |
| dt_full | 0.858 | 0.381 | 0.386 | 0.383 | 4.891 | 0.651 | 0.303 | 6.500 |
| dl_full | 0.884 | 0.000 | 0.000 | 0.000 | 0.578 | 0.852 | 0.000 | 8.250 |

Table 7: Indonesia model performance across six metrics, using all available features. Models are sorted by their *mean rank*, as defined in Sec. 2. See Table 2 for list of abbreviations, Table 1.4 for summary of data.

algorithm has a mean rank of 7.875). Half of the top 10 models are based on logistic regression, which seems to be the most effective algorithm type for imbalanced data, even when preprocessing is included to account for the imbalance. While accuracy decreases from the simple model, the more complex models are able to achieve high accuracy while maintaining good results for recall and decent results for precision. F1 score provides a balance of precision and recall. Notably, the best performing model for F1 (when preprocessing techniques are included) does not improve upon the best performing model without preprocessing techniques.

## 3 Comparison of Advanced Machine Learning Approaches

This section explores advanced techniques that go beyond the "out-of-the-box" approaches presented in Sec. 2.2. This section considers four approaches that require more expertise and model complexity but often perform well for optimizing predictive performance: crowd-sourced algorithms, ensemble methods, automated machine learning, and deep learning. The crowd-sourced algorithm results discussed in Sec. 3.1 include Malawi but not Indonesia. Because simple models were less

Top 10 Indonesia Results with Class Balancing, Tuning, CV, and Feature Selection

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=32) |
|---|---|---|---|---|---|---|---|---|
| lr_full_oversample | 0.856 | 0.840 | 0.436 | 0.574 | 0.344 | 0.927 | 0.477 | 7.875 |
| mlp_full_undersample_cv | 0.825 | 0.902 | 0.389 | 0.543 | 0.370 | 0.930 | 0.434 | 8.375 |
| lr_full_oversample_cv | 0.852 | 0.841 | 0.429 | 0.568 | 0.348 | 0.926 | 0.474 | 8.500 |
| xgb_full_undersample_cv | 0.832 | 0.874 | 0.397 | 0.546 | 0.379 | 0.928 | 0.448 | 8.625 |
| lr_full_undersample | 0.826 | 0.892 | 0.389 | 0.542 | 0.394 | 0.927 | 0.434 | 9.625 |
| lr_full_classwts | 0.831 | 0.867 | 0.394 | 0.542 | 0.385 | 0.923 | 0.428 | 11.000 |
| mlp_full_undersample | 0.828 | 0.887 | 0.392 | 0.544 | 0.894 | 0.922 | 0.450 | 11.625 |
| lda_full_oversample_cv | 0.815 | 0.890 | 0.374 | 0.526 | 0.429 | 0.922 | 0.407 | 12.000 |
| lr_l1_feats_oversample_cv | 0.833 | 0.835 | 0.395 | 0.536 | 0.378 | 0.915 | 0.408 | 12.250 |
| lda_full_oversample | 0.813 | 0.886 | 0.371 | 0.523 | 0.426 | 0.922 | 0.406 | 12.375 |

Table 8: Indonesia model performance across six metrics, using all available features and various class-balancing techniques as well as cross-validated training. Models are sorted by their *mean rank*, as defined in Sec. 2. See Table 2 for list of abbreviations, Table 1.4 for summary of data.

accurate in Indonesia, evaluation focuses these more sophisticated methods on that dataset in Sec. 3.2, Sec. 3.3, and Sec. 3.4. Compared to the previous section, which focuses on common algorithm types, more detail on model types is provided since many of these approaches are rarely considered in the context of predicting household level poverty.

### 3.1 Crowd-sourced Algorithms

Data science competitions have a history of being used to solve complex predictive problems. One of the most well-known predictive modeling competitions was the Netflix Prize [3], which offered $1,000,000 USD to the team that could improve their recommendation algorithm by 10%. The competition ran from 2006 - 2009, and had participation from researchers around the world. Since then popularity of such competitions, known generally as innovation tournaments, has increased in commercial and academic spheres [4]. These competitions offer prizes to teams or individuals who can solve a problem, the results of which are typically assessed by predictions on held-out subsets of the data. Online platforms such as Kaggle or DrivenData (http://www.drivendata.org) engage data scientists and other quantitative experts from around the world in building models for a combination of prize money, recognition, practice, and real-world impact. These challenges enable participants to try out thousands of models for a given problem using whatever backgrounds, skills, and approaches they see fit, with the solutions that perform best climbing to the top of the leaderboard. At a high level, these competitions offer broad coverage of a wide solution space.

DrivenData, Inc., in partnership with the World Bank, hosted a data science competition where competitors were asked to optimize poverty prediction using survey data from three countries, Malawi, Namibia, and Tajikistan [8]. Submissions were ranked using the mean log loss metric. The data were anonymized in order to minimize opportunities to cheat by using publicly available versions of the data that might exist. Aside from cash prizes for the top three leaderboard submissions, there was a bonus prize for the top competitor from a low-income or lower-middle-income country. 84 qualifying countries were determined using World Bank Country and Lending group data [2]. The winners of the competition came from Portugal, Russia, Germany, and the Philippines.

The top competitors all used ensembles of different algorithms, and every ensemble included tree-based methods. The most popular model was an implementation of gradient boosted trees called "LightGBM" [14], which appeared in all of the four solutions that were awarded prizes. Three of the 4 prize winners used both XGBoost and some form of artificial neural network. Finally, CatBoost [7], a method of gradient boosting developed at the Russian search engine company Yandex, and random forests appeared in one solution each. Most of the winners chose a simple blending approach to ensembling model predictions, which consists of a weighted combination of the predictions of each model.

For the balanced Malawi data, the top competitors' submissions produce results that are slightly better than the initial survey of methods on the Malawi data reported in Table 5. These competition results are shown in Fig. 1 and Table 9. The figures are generated using a different subset of the Malawi data than that of the models in Fig. 5. Both samples are representative however, therefore these results suggest that competitions compare well with a more systematic comprehensive search of the exploration of possible models.

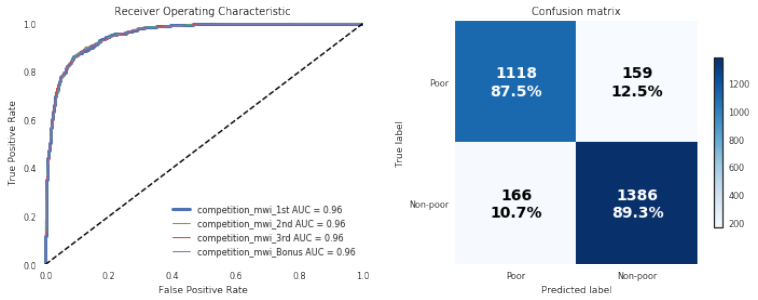



Figure 1: Malawi Private Test Results of the DrivenData competition [8]. (Left) AUC of 1st, 2nd, 3rd, and Bonus (top performer from a developing country). Slightly improved performance to the top models in Table. 5. (Right) Confusion matrix of first place.

Table 9: DrivenData Competition Results: Malawi (Balanced)

| | competition_mwi_1st | competition_mwi_2nd | competition_mwi_3rd | competition_mwi_Bonus |
|---|---|---|---|---|
| accuracy | 0.885 | 0.885 | 0.889 | 0.885 |
| recall | 0.875 | 0.897 | 0.886 | 0.883 |
| precision | 0.871 | 0.856 | 0.869 | 0.865 |
| f1 | 0.873 | 0.876 | 0.878 | 0.874 |
| cross_entropy | 0.264 | 0.269 | 0.262 | 0.264 |
| roc_auc | 0.957 | 0.956 | 0.957 | 0.957 |
| cohen_kappa | 0.768 | 0.770 | 0.776 | 0.767 |

These results indicate that across household survey datasets which contain mostly categorical features, modern tree based models (random forests, XGBoost, CatBoost) can be the most effective. They also demonstrate that ensembling can be a useful tool for getting the most predictive power out of a dataset without overfitting.

### 3.2 Ensemble Techniques

As seen in the previous section, a common way to improve predictive performance is to create a model ensemble, combining the predictions of more than one model to form a so-called "meta-prediction." Although some models, such as tree-based ones like XGBoost or random forests, are already ensembles of many weak learners (a weak learner overfits a certain property of the data, becoming an "expert" that contributes to the ensemble), it is possible to perform ensembling of heterogeneous algorithm types. There are a number of techniques for creating ensembles including blending, model averaging, and stacking.

This section presents the results of two ensembling approaches. The first, presented in Sec. 3.2.1, is a simple ensemble average of the top performing models. The second, presented in Sec. 3.2.2, is a more advanced technique called *stacking*, where the best models from Secs.8 and 10 are combined to produce predictions that are used as features to train a meta-estimator which in turn generates the final predictions.

#### 3.2.1 Simple Ensemble Average

The simplest approach to ensembling is to take an average of predictions from already-trained models. If class labels rather than probabilities are desired, the newly computed average can be thresholded, just like a normal classifier.

For the Indonesia data, a simple ensemble average was created using the models from Table 8. The results are shown in Fig. 5.

Due to the heterogeneous nature of the models in this ensemble (including variations in class balancing techniques and weighting), the simple ensemble average performs very well. The advantage of combining heterogeneous model types is evidenced here as a simple average outperforms any of its constituent models (see the mean ranking of "simple_ensemble" in Table 10).

#### 3.2.2 Stacking (Meta-Ensembling)

Model stacking, or meta-estimation, is a popular technique used in machine learning competitions. Indeed, the winners of [8] all used versions of this technique. In model stacking, rather than averaging the predictions of a model ensemble, the predictions of each model in the ensemble are used as features, often referred to as *meta features*, to train a new classifier (or, meta classifier). In this way, the "stack" is composed of two *levels*: level 1 composed of independent estimators, and level 2 which combines the level one estimates using the standard train-test work flow of fitting a model.

In order to avoid overfitting the training set, it is important to use a proper validation technique when training a stacked model. To train a stacked model using $l$ base estimators to generate features for meta estimator $M$, a very common approach consists of,

1. **Preprocess** Split training data $X$ into $k$ folds $(X_1, ..., X_k)$.

2. **Train Level 1** For each model $L_i$ where $i : 1, ..., l$, fit $L_i$ using $k - 1$ folds and predict on the remaining fold. Repeat this cycle $k$ times, until $L_i$ has predicted on *every* fold.
3. **Train Level 2** Train the meta estimator $M$ using the $L$-dimensional features generated in step 2. If $X$ has 1000 samples then the meta estimator training data will take the shape $1000 \times L$.
4. **Optional** After meta features are generated, refit level 1 models using full training set. Do not do anything to level 2 models!

The key point is that level 2 features need to be generated using out-of-fold predictions from level 1. Otherwise the level 2 meta estimator will overfit the training data severely, and the stacked model may perform worse than any of its components. If executed appropriately, it can be shown that stacked modeling will, with probability approaching 1, never perform worse than its best base estimator [24].

In order to generate a stacked classifier for the Indonesia data, the top performing algorithms *classes*, reported in Table 8, were used as level 1 estimators, with the exception of XGBoost which was used as the meta estimator. Level 1 estimators therefore include logistic regression, multi-layer perceptron, and linear discriminant analysis algorithms. Although many different instances of these estimators appear in Table 8, only one instance of each model *class* is used in the stack. One reason for this is that the open source software [1] that was used to implement the training process described above requires all models to use the same training data, making it difficult to use combinations of under and over sampling class-balancing techniques.

The results for the stacked classifier are shown in Table 10, under the model name "ensemble_stacked." The model was trained using a grid search over all hyperparameters considered in the training of algorithms in Table 8. This means that for each algorithm in level 1 of the stack, various model configurations were considered while the other model configurations were held fixed. This led to a long training time (about 24 hours on a 32 core machine), but sought to exhaustively search for the best combination of the model classes considered in the Indonesia full feature, class balanced and tunes results Table 8. Surprisingly, the stacked ensemble performed worse than the simple ensemble when measured across all metrics using the mean rank comparison. This is likely due to the substantially higher heterogeneity of the simple ensemble, which uses various class balancing techniques. However, it is worth noting that the stacked ensemble does have a higher recall, which means it learned better not to miss targets of interest (poor households).

### 3.3 Automated Machine Learning

Another approach applied to the Indonesia data was automated machine learning (AutoML). In AutoML, the entire machine learning pipeline—from preliminary data processing to model selection and training—is framed as a minimax optimization problem that seeks to maximize model performance while minimizing the complexity of the pipeline. This means that cumbersome steps such as feature engineering, which typically require subject matter experts, are not performed manually, as indicated by the AutoML diagram shown in Fig. 3. However, AutoML has its drawbacks. Because the space of solutions is composed of complex pipelines which must be executed, AutoML is more computationally intensive than optimizing coefficients for a single model.[4] Further, the resulting pipeline found by an AutoML approach, either after a certain condition is met or after a certain amount of time, can be difficult to interpret or justify.

**Genetic programming** The optimization approach used here is an instance of *genetic programming* (GP), a technique inspired by evolutionary biology to search the solution space. While genetic programming was first introduced in 1973 by Rechenberg [20], it has not been widely used in the machine learning community until recently. To implement a genetic algorithm for the Indonesia data, an open source python framework called TPOT [19, 18, 17] was used. A concise explanation of the optimization process is offered by one of the TPOT publications [17]:

> The GP algorithm generates 100 random tree-based pipelines and evaluates their accuracy on the dataset. For every generation of the GP algorithm, the algorithm

[4] That said, manually searching the model space is also computationally intensive. For much of this project work we used GNU parallel [23] to make better use of computational resources.

selects the top 20 pipelines in the population according to the NSGA-II selection scheme, where pipelines are selected to simultaneously maximize classification accuracy on the dataset while minimizing the number of operators in the pipeline. Each of the top 20 selected pipelines produces five offspring into the next generation's population, 5 of those offspring experience crossover with another offspring, then 90 percent of the remaining unaffected offspring experience random mutations. Every generation, the algorithm updates a Pareto front of the non-dominated solutions discovered at any point in the GP run. The algorithm repeats this evaluate-select-crossover-mutate process for 100 generations—adding and tuning pipeline operators that improve classification accuracy and pruning operators that degrade classification accuracy—at which point the algorithm selects the highest accuracy pipeline from the Pareto front as the representative "best" pipeline from the run.

An example tree-based pipeline from [17] is shown in Fig. 2.

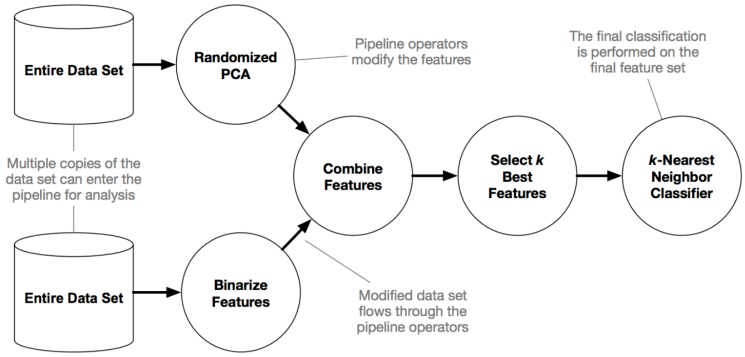



Figure 2: An example tree-based pipeline from TPOT. Each circle corresponds to a machine learning operator, and the arrows indicate the direction of the data flow. (Figure and caption from [17]).

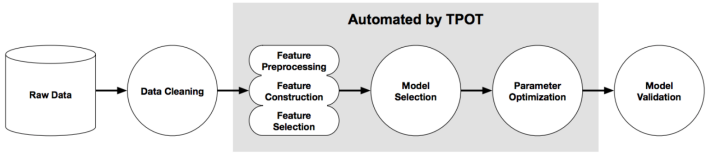



Figure 3: A typical machine learning pipeline, indicating the portion automated by the chosen AutoML package. (Figure from [17]).

**Results using TPOT** After 200 generations of training on a 32 core machine (about 2 days) the best performing TPOT pipeline used an XGBoost classifier with 6 preprocessing steps. It is worth noting that at the end of an optimization the TPOT software conveniently exports fully executable python code that implements the winning pipeline. This is very useful in cases where the programmer may want to tweak the pipeline manually to improve performance further. In the current case, the optimal preprocessing steps were not all standard transformations such as normalization and interaction terms; preprocessing included use of a TPOT transformation called the StackingEstimator. The StackingEstimator uses other classifiers—in this case logistic regression and XGBoost—to produce *synthetic features*, comprised of predictions and class probabilities. The particular pipeline output from the search augments the feature space with two StackingEstimators using $l_2$ penalties before normalizing the features, then passing them through two more StackingEstimators using $l_1$ penalties and finally binarizing the features based on a $0.25$ threshold. The result is passed into an XGBoost classifier. This counter-intuitive pipeline demonstrates both the strength and weakness of AutoML approaches: seemingly creative solutions may lead to better performance, but the reason for each step in the pipeline—normally justified by the human modeling the problem—is not clear.

The performance of the TPOT model is shown in Table 10. While it was far from the top performing model using cross-validation, it does achieve the highest f1 score. This is particularly

notable since the model is optimized with respect to its f1 score during training. However, the deep learning models considered in Sec.3.4 outperform TPOT in every other metric.

### 3.4 Deep Learning

In recent years, there has been a resurgence of artificial neural network approaches, due to a combination of increased computational power and improved training methods. In a general sense, a neural network can be viewed as a chain of functional compositions, often alternating between linear operations (neurons) and non-linear operations (activation functions). It can be shown that artificial neural networks with at least one additional layer between input and output can arbitrarily approximate any continuous function [13]. Like any other machine learning model, the output of these networks during training is evaluated according to a chosen objective function. The result of the evaluation is then "back-propagated" through the network, leading to changes in model parameters along the way. Typically, these changes are related to the gradient of the objective function with respect to the particular model parameter under consideration. Upon reaching its stopping condition, the neural network can be thought of as any other machine learning model: mapping input data to an output class or regressed value.

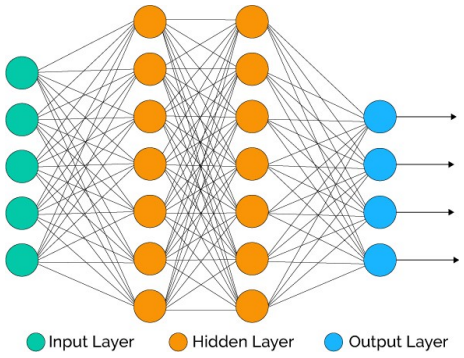



Figure 4: Sample neural network architecture from an introductory article.

The popular term *deep learning* picks out a particular class of neural network architectures. Specifically, it refers to neural networks with more than one layer between input and output, as depicted in Fig. 4. The simplest networks can be viewed as functional operations in series, but much more complicated architectures with loops, parallel paths, and sophisticated sub-modules exist [11]. The network is *deep* in the sense that many—sometimes hundreds of—operations may be applied to the sample, providing a complicated transformation of the input data before classification. At the same time, throughout the computation the dimensionality of the sample is upper-bounded by a reasonably small constant, i.e., the network does not get too *wide*. One standard way of conceptualizing the tradeoff between deep and wide networks is the well-known bias-variance tradeoff. Wider networks are better at *memorizing* patterns in observed data in the same way that a high-order polynomial is better at fitting a complicated set of two-dimensional points; with more parameters available, a closer fit to the training data is possible by exploiting correlations in the historical data. The negative side of such biased models is that they tend to fail to capture the true variance of the underlying data distribution, instead memorizing co-occurrences of the features in the training set. Deep networks still have many parameters, but are better at handling variance because they map the input data to an abstract latent space before classification, with the many layers effectively acting as feature extractors capable of detecting invariant relationships between the data. In fact, it can be useful to conceive of deep models as essentially a series of automated feature extractors with a simple linear classifier or regressor in the final layer. Such networks have been especially successful in applications to computer vision and speech processing.

Recently, a model introduced by researchers at Google proposed combining the strengths of wide and deep networks [5]. The original aim of this work was to utilize the handling of high-dimensional sparse data common in click-through rate (CTR) analysis to make recommendations to users in the Google Play store. The model's *wide* part makes recommendations based on specific co-occurrences chosen by the programmer; this approach does not generalize to correlations that have not appeared in the data. The *deep* part makes recommendations based on the abstract correlations

learned by the hidden layers of a neural network. When these modules are coupled together and jointly trained, the model achieves significant improvements compared to each decoupled component predicting alone.

The household poverty surveys considered in this paper have a similar structure to the CTR data. Namely, both are high dimensional datasets of primarily binary features. While it may appear that the wide and deep approach could be useful for the high-dimensional sparse poverty survey data, there is an important bottleneck to consider: the original network architecture requires manual selection of the features in the wide component. That is, the programmer must decide *a priori* which interaction terms the model will consider. This problem can be circumvented by considering all possible pairs of feature interactions, but such an approach is often extremely inefficient if not intractable due to the dimensionality of the feature vectors.

A solution to manually engineering feature interactions was offered by Guo et al. [12], which proposes a *deep factorization machine* (DeepFM). The DeepFM model makes use of a dense embedding layer as a singular input to both the wide and deep modules of the model. The dense embedding transformation maps discrete inputs to a lower-dimensional continuous space. From here, the wide component can efficiently compute higher-order feature pairs by using a factorization machine (FM) [21] layer as an operation in the wide part of the network: instead of learning feature interactions explicitly, FM layers represent features using a fixed size latent vector and learn the inner products of latent interactions, effectively acting as another dense encoding. These latent vectors can be pre-computed and reduce feature interaction complexity to linear time. The wide part of the model is then able to assign non-zero weights to interactions in linear time. In the same way that this is approach is useful for the CTR data, it is useful for the poverty data, since it is computationally expensive to compute all interactions explicitly.

**Results using DeepFM** The DeepFM model is tested using all features as input and with 3-fold cross-validation. The same configuration is tested using under-sampling as well. After a relatively short training period (50 epochs, about 15 minutes on a GPU-enabled machine), the "DeepFM" models achieved good performance, as shown in Table 10. When compared across other years however (see Sec. 5), these models achieve the top rank.

Note that the non-under-sampled deep model does not perform as well as other cross-validated models on recall. This failure is crucial when the model task is the identification of unbalanced classes, and is particularly important in the context of poverty. The under-sampled DeepFM model, however, solves this issue and outperforms other top models in recall, although, consistent with the rest of the results reported here, the improvement to recall comes at the price of other metrics, as indicated in Table 10.

Table 10: Results Using Advanced Models: Simple Ensemble, Stacked, Tpot & DeepFM

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=15) |
|---|---|---|---|---|---|---|---|---|
| deepfm | 0.932 | 0.549 | 0.648 | 0.594 | 0.163 | 0.943 | 0.558 | 3.429 |
| ensemble_simple | 0.878 | 0.856 | 0.415 | 0.559 | 0.273 | 0.945 | 0.498 | 4.143 |
| lr_full_oversample | 0.856 | 0.840 | 0.436 | 0.574 | 0.344 | 0.927 | 0.477 | 5.429 |
| lr_full_oversample_cv | 0.852 | 0.841 | 0.429 | 0.568 | 0.348 | 0.926 | 0.474 | 6.143 |
| ensemble_stacked | 0.852 | 0.898 | 0.368 | 0.522 | 0.316 | 0.944 | 0.452 | 6.857 |
| xgb_full_undersample_cv | 0.832 | 0.874 | 0.397 | 0.546 | 0.379 | 0.928 | 0.448 | 7.143 |
| automl_tpot | 0.899 | 0.700 | 0.551 | 0.616 | 0.626 | 0.813 | 0.554 | 7.143 |
| mlp_full_undersample_cv | 0.825 | 0.902 | 0.389 | 0.543 | 0.370 | 0.930 | 0.434 | 7.571 |
| lr_full_undersample | 0.826 | 0.892 | 0.389 | 0.542 | 0.394 | 0.927 | 0.434 | 8.714 |
| lr_full_classwts | 0.831 | 0.867 | 0.394 | 0.542 | 0.385 | 0.923 | 0.428 | 9.429 |
| mlp_full_undersample | 0.828 | 0.887 | 0.392 | 0.544 | 0.894 | 0.922 | 0.450 | 9.429 |
| lr_l1_feats_oversample_cv | 0.833 | 0.835 | 0.395 | 0.536 | 0.378 | 0.915 | 0.408 | 10.143 |
| deepfm_full_undersample_cv | 0.794 | 0.937 | 0.297 | 0.451 | 0.442 | 0.932 | 0.363 | 11.143 |
| lda_full_oversample_cv | 0.815 | 0.890 | 0.374 | 0.526 | 0.429 | 0.922 | 0.407 | 11.429 |
| lda_full_oversample | 0.813 | 0.886 | 0.371 | 0.523 | 0.426 | 0.922 | 0.406 | 11.857 |

## 4 Misclassification

In this section, misclassifications in the 2012 Indonesia test data are considered. A household is said to *have a model in error* if at least 25% (5) of the top 20 performing models made *either* a false positive *or* a false negative prediction for that household. (No model was 100% accurate, but some households were correctly classified by all models, so not every household has a model in

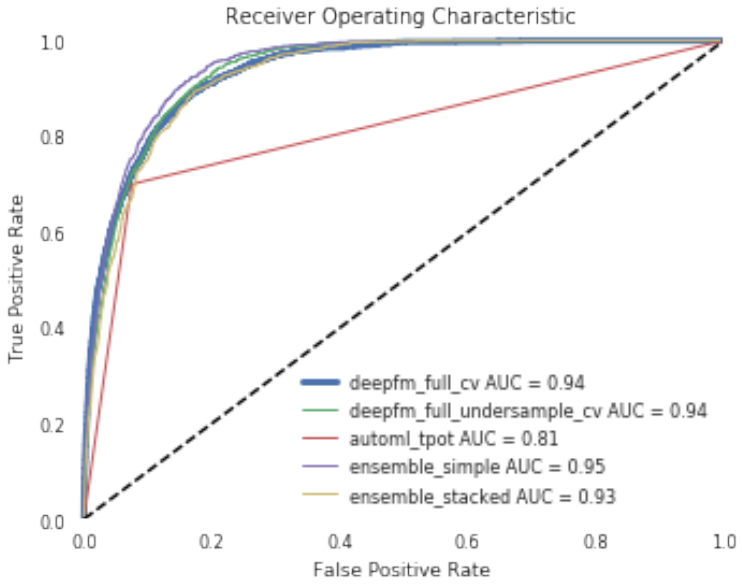



Figure 5: ROC curve of advanced models reported in Sec. 3.

error.) As defined in Sec. 2.1, these are the two types of misclassifications that a binary classifier can make. In the case of the poverty survey data, the "positive" class is "Poor" and the "negative" class is "Non-Poor," with the classes determined by each household's per capita consumption relative to a fixed poverty line for each country. So, a false positive prediction is one in which the model predicts a household to be "Poor" when the household label is "Non-Poor", and a false negative prediction is one in which the model predicts a household to be "Non-Poor" when the household label is "Poor".

Beyond their number, we are interested in the concentration of the false positives and false negatives around the poverty line. On one hand, misclassifying households that are in close proximity to the poverty line may have less impact than misclassifying households that are significantly above or below the poverty line. On the other hand, data on misclassified households that are far from the poverty line may need particular scrutiny. Indeed, we may not assume that the survey data are exempt from error. Understanding patterns in misclassifications can help uncover improvements to the modeling, data collection or data quality issues, and bugs in the software.

Figure 6 shows centered distance from the poverty line for all households with at least 5 of the top 20 models in error. By subtracting the poverty line from the per capita expenditure the data are centered around zero. Thus, the distance from the poverty line is given by the distance from zero. This is a useful check of data quality; false negative households (predicted "Non-Poor", labeled "Poor") are below the centered poverty line and false positive households (predicted "Poor", labeled "Non-Poor") are above the poverty line.

For a higher-level view of model performance, the inter-model agreement for misclassifications is plotted in Fig. 7. In general, more households are consistently incorrectly predicted to be "Poor" across all models (false positive) than incorrectly predicted to be "Non-Poor" (false negative), as is evidenced by the fact that the false positive density is shifted towards a higher fraction of models in error. What this means is that with proper model tuning, it should be easier for ensemble methods to correct for false negatives than false positives, since heterogeneity (the primary advantage of ensembling) is less useful when all models are making the same error. This reflects the model results where recall, which measures the prevalence of false negatives, is one of the metrics that is hardest to optimize.

To give a sense of the distribution of errors across the top models, Fig. 8 shows the spread of model confidences for each error type, grouped by model, and sorted from left to right according to the average median error of the model. The top classifiers produce a probability that a household is poor. A model that predicts 0.5 is not confident whether a household is "Poor" (1) or "Non-Poor" (0). This means that there is a possible error range of 0.5: a model that predicts a household labeled

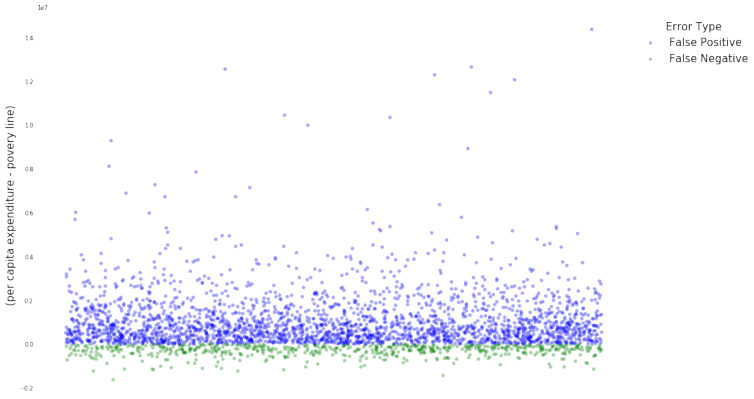



Figure 6: Distance From Poverty Line - Households With At Least 5 Models In Error.

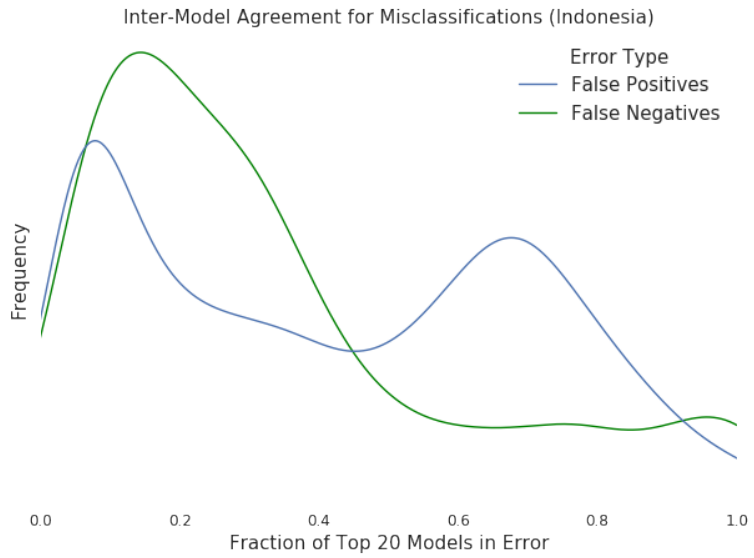



Figure 7: Inter-Model Agreement for Misclassifications

"Poor" to be "Non-Poor" (false negative) with probability 0 is the maximum possible distance of 0.5 away from the threshold that *would* have led to a correct prediction (true positive). Similarly, a model that predicts a household labeled "Non-Poor" to be "Poor" (false positive) is the maximum possible distance of 0.5 away from the threshold that *would* have led to a correct prediction (true negative).

In Fig. 8 this measure of a model's confidence in its incorrect preditions is plotted on the Error axis. A tighter box indicates less variability in model confidence, so for example the "svm_full_classwts" model consistently *almost* correctly predicted its false negative households, because the false negative box (green) is close to 0 error and has a maximum error of less than 0.1. The same model's false positive box (blue) however, has much more variability, with minimum errors close to 0 and maximum errors at the global maximum of 0.5. The spread of a box in these plots can be taken to correspond to model consistency. Additionally, note that the "deepfm_full_cv," which is one of the overall top performing models reported (see Sec. 5), consistently makes almost the maximum possible error in its false positive (blue) predictions, indicating a high model confidence but incorrect predictions.

Another approach to misclassification analysis is to investigate properties of the misclassified samples themselves, for example comparing the structure of misclassified samples to the aggregated structure of each class, which we do in Figs. 10 and 9. These figures compare the consumption patterns of households predicted false positive (or negative) by at least 25% of the top models to

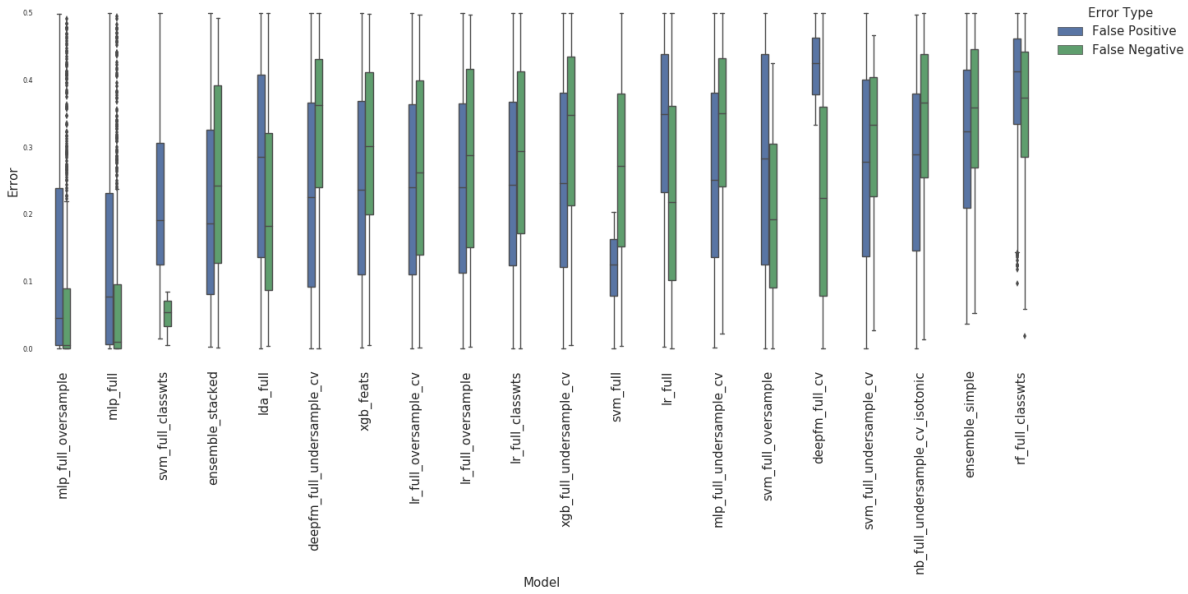



Figure 8: Distribution of Errors As Measured By Model Confidence

the aggregated consumption patterns of "Poor" and "Non-Poor" households as determined by the original expenditure data and poverty line threshold.

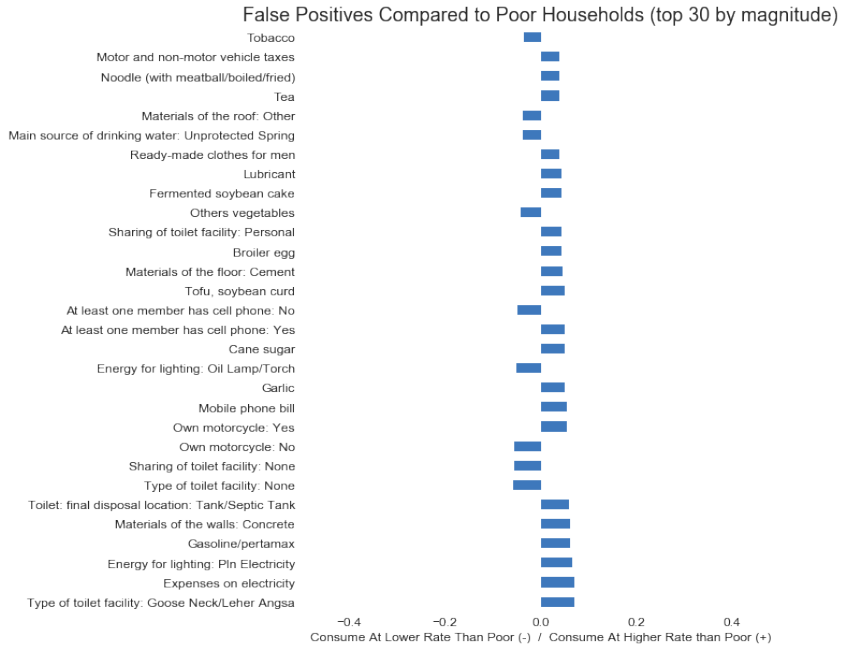
The *consumption rate* of any categorical survey question is determined by taking the normalized value count of affirmative answers, e.g. "$X\%$ of households answered "Yes" to owning a refrigerator." For the plots below, we will compare the consumption behavior of misclassified exampled to both 'Poor' and 'Non-Poor' consumption behaviors of the aggregated population. For each type of misclassification, false positives (predicted 'Poor', labeled 'Non-Poor') and false negatives (predicted 'Non-Poor', labeled 'Poor'), we will compute the sample's consumption rates and compare them to the consumption rates of both 'Poor' and 'Non-Poor' members of the population. In Figs. 9 and 10, a positive value indicates that the sample's consumption rate is smaller than the comparison group. The top 30 largest magnitude effects are shown for each type of misclassification as compared to the overall "Poor" and "Non-Poor" populations.

The false positives and false negatives both exhibit consumption patterns more similar to the "Poor" population. The false negatives are slightly less like the poor households and slightly moor like the non-poor households than the false positives, and this difference is likely what some of the models which misclassified these households identified.

## 5 Model Robustness

One of the best practical tests of a machine learning model is to measure its performance on an entirely new dataset. Throughout this paper, 2012 Indonesia expenditure survey data are used to train and validate many models. In this section, similarly structured survey data from other years are used as alternative test sets. Specifically, survey data from 2011, 2013 and 2014 are used to test the Indonesia models reported in Table 8, as well as the more advanced models in Secs. 3.

Tables 11, 12, and 13 report the results of the mean rank calculation defined in Sec. 2.1. Table 14 shows the result of the mean rank calculation for each model across the 2011-2014 data. When compared over all years, it is clear that the neural network models are the most robust (recall that "mlp" stands for multi-layer perceptron—see Table 2—which is a simple neural network). The deep learning models in particular consistently rank in the top two, except in 2012, the year used to train the models. This suggests that while the deep learning models do a good job of capturing the overall structure of the data distribution over many years.



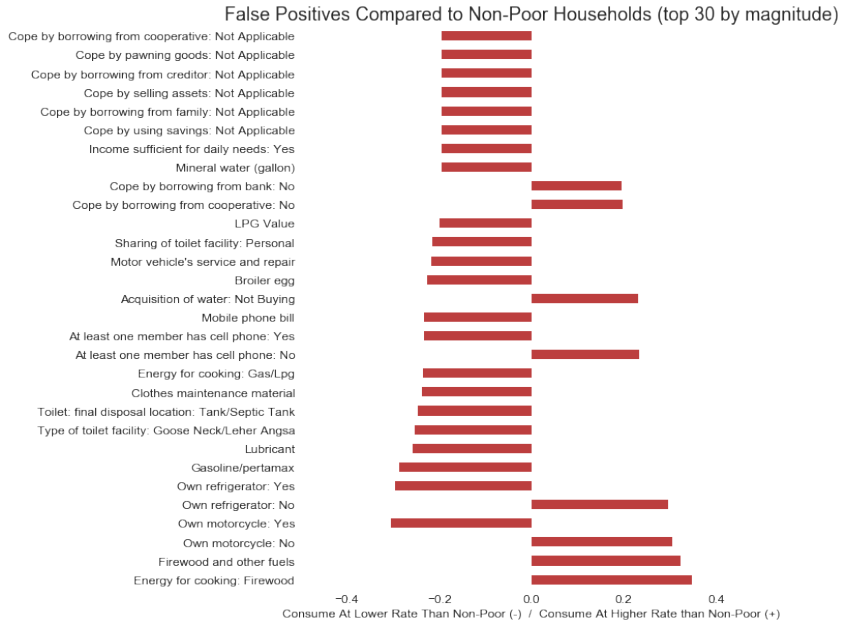
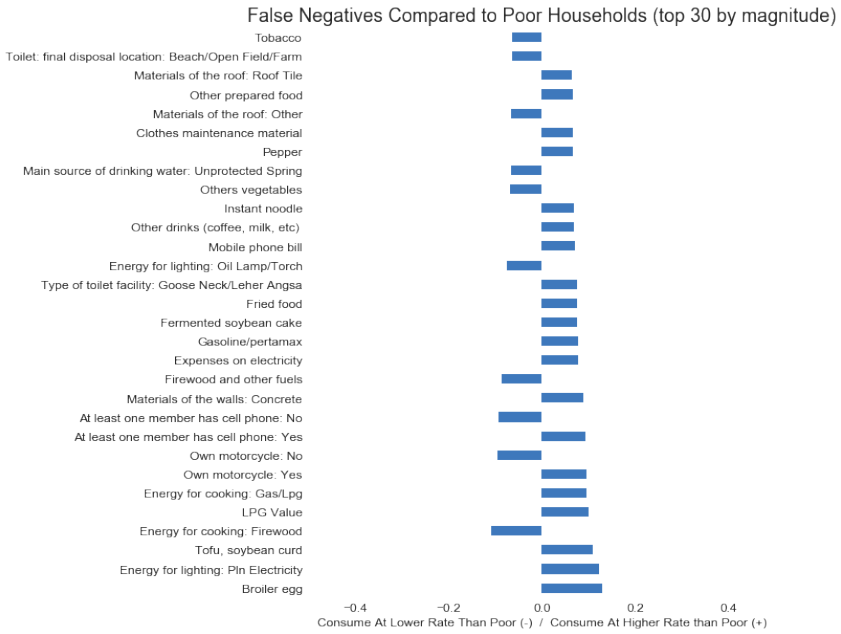




Figure 9: Comparison of Consumption of False Positive Households to Poor and Non-Poor



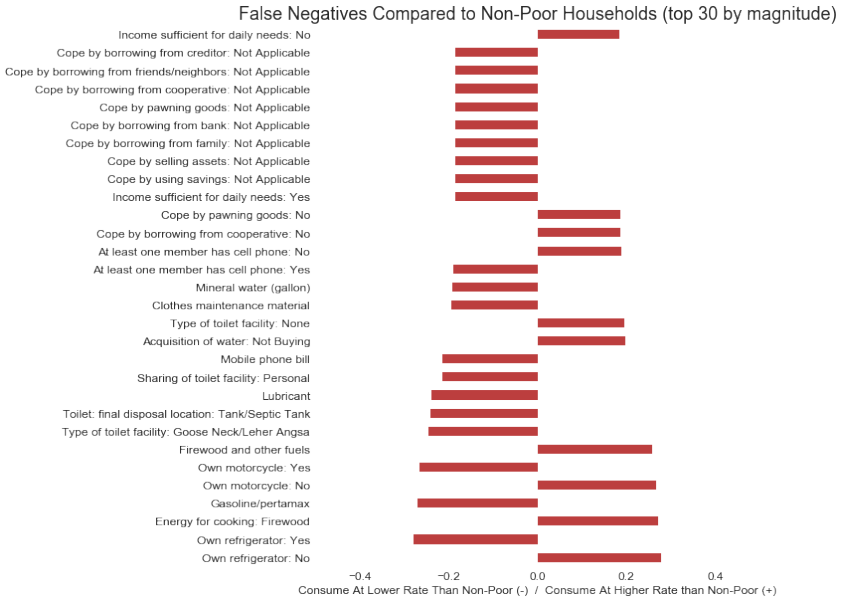




Figure 10: Comparison of Consumption of False Negative Households to Poor and Non-Poor

Table 11: Robustness: 2011

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=14) |
|---|---|---|---|---|---|---|---|---|
| deepfm_full_cv | 0.889 | 0.608 | 0.501 | 0.550 | 0.280 | 0.908 | 0.487 | 3.143 |
| deepfm_full_undersample_cv | 0.885 | 0.688 | 0.488 | 0.571 | 0.281 | 0.915 | 0.507 | 3.286 |
| mlp_full_undersample_cv | 0.906 | 0.274 | 0.280 | 0.277 | 0.250 | 0.779 | 0.226 | 4.857 |
| ensemble_stacked | 0.887 | 0.243 | 0.482 | 0.323 | 0.314 | 0.855 | 0.269 | 5.429 |
| ensemble_simple | 0.893 | 0.074 | 0.677 | 0.134 | 0.239 | 0.905 | 0.114 | 6.429 |
| lr_full_undersample | 0.888 | 0.320 | 0.239 | 0.273 | 0.356 | 0.757 | 0.214 | 7.000 |
| xgb_full_undersample_cv | 0.855 | 0.415 | 0.205 | 0.275 | 0.332 | 0.768 | 0.205 | 7.143 |
| lr_full_oversample | 0.929 | 0.085 | 0.348 | 0.137 | 0.573 | 0.736 | 0.114 | 7.857 |
| lr_full_oversample_cv | 0.927 | 0.088 | 0.309 | 0.137 | 0.720 | 0.717 | 0.111 | 8.571 |
| mlp_full_undersample | 0.897 | 0.210 | 0.214 | 0.212 | 0.971 | 0.724 | 0.157 | 8.571 |
| lr_full_classwts | 0.907 | 0.154 | 0.215 | 0.179 | 0.833 | 0.680 | 0.132 | 8.714 |
| automl_tpot | 0.891 | 0.045 | 0.707 | 0.085 | 0.627 | 0.526 | 0.072 | 9.857 |
| lda_full_oversample | 0.554 | 0.390 | 0.060 | 0.103 | 4.492 | 0.496 | -0.012 | 11.857 |
| lda_full_oversample_cv | 0.547 | 0.392 | 0.059 | 0.102 | 4.281 | 0.489 | -0.014 | 12.286 |

Two of the ensemble-based models—the simple ensemble average and the under-sampled XGBoost models—also consistently outrank many other top models.

It is somewhat surprising that the simple ensemble average, which consists only of the average probability over all of the top 10 models reported in, outperforms the stacked ensemble. The simple ensemble average has more heterogeneity in its base learners than any of the other models, including the stacked ensembles. Although in general stacked ensemble models are thought to be much more robust than simple ensemble averages, for these particular data the simple ensemble always outperforms the stacked ensemble according to the mean metric, likely because of the variety of methods it incorporates.

Table 12: Robustness: 2013

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=14) |
|---|---|---|---|---|---|---|---|---|
| deepfm_full_undersample_cv | 0.890 | 0.707 | 0.383 | 0.497 | 0.253 | 0.918 | 0.441 | 2.143 |
| deepfm_full_cv | 0.865 | 0.674 | 0.322 | 0.436 | 0.337 | 0.882 | 0.370 | 3.714 |
| mlp_full_undersample_cv | 0.906 | 0.274 | 0.280 | 0.277 | 0.250 | 0.779 | 0.226 | 3.857 |
| xgb_full_undersample_cv | 0.855 | 0.415 | 0.205 | 0.275 | 0.332 | 0.768 | 0.205 | 5.857 |
| lr_full_undersample | 0.888 | 0.320 | 0.239 | 0.273 | 0.356 | 0.757 | 0.214 | 5.857 |
| lr_full_oversample | 0.929 | 0.085 | 0.348 | 0.137 | 0.573 | 0.736 | 0.114 | 6.571 |
| lr_full_oversample_cv | 0.927 | 0.088 | 0.309 | 0.137 | 0.720 | 0.717 | 0.111 | 7.429 |
| lr_full_classwts | 0.907 | 0.154 | 0.215 | 0.179 | 0.833 | 0.680 | 0.132 | 7.571 |
| mlp_full_undersample | 0.897 | 0.210 | 0.214 | 0.212 | 0.971 | 0.724 | 0.157 | 7.714 |
| ensemble_simple | 0.905 | 0.053 | 0.155 | 0.078 | 0.295 | 0.677 | 0.041 | 9.857 |
| automl_tpot | 0.906 | 0.054 | 0.164 | 0.081 | 0.622 | 0.516 | 0.045 | 10.143 |
| ensemble_stacked | 0.863 | 0.104 | 0.106 | 0.105 | 0.461 | 0.617 | 0.031 | 10.429 |
| lda_full_oversample | 0.554 | 0.390 | 0.060 | 0.103 | 4.492 | 0.496 | -0.012 | 11.714 |
| lda_full_oversample_cv | 0.547 | 0.392 | 0.059 | 0.102 | 4.281 | 0.489 | -0.014 | 12.143 |

Table 14 also shows that the logistic regression with oversampled class-weighting stays in the top half of the overall ranking. This is a particularly notable result given the simplicity of the model. Logistic regression is straightforward to understand, fast to train, and more performant than many more complicated approaches such as the TPOT optimization described in Fig. 2.

The models based on linear discriminant analysis, while also simple, are not robust across years. They trail far behind the other models in every one of the metrics considered in this report.

Table 13: Robustness: 2014

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=14) |
|---|---|---|---|---|---|---|---|---|
| deepfm_full_undersample_cv | 0.917 | 0.612 | 0.410 | 0.491 | 0.203 | 0.920 | 0.448 | 1.714 |
| deepfm_full_cv | 0.878 | 0.659 | 0.303 | 0.415 | 0.305 | 0.888 | 0.357 | 4.000 |
| mlp_full_undersample_cv | 0.906 | 0.274 | 0.280 | 0.277 | 0.250 | 0.779 | 0.226 | 4.571 |
| ensemble_simple | 0.923 | 0.178 | 0.342 | 0.235 | 0.267 | 0.748 | 0.198 | 5.714 |
| xgb_full_undersample_cv | 0.855 | 0.415 | 0.205 | 0.275 | 0.332 | 0.768 | 0.205 | 6.571 |
| lr_full_undersample | 0.888 | 0.320 | 0.239 | 0.273 | 0.356 | 0.757 | 0.214 | 6.571 |
| ensemble_stacked | 0.901 | 0.216 | 0.230 | 0.223 | 0.317 | 0.762 | 0.170 | 7.000 |
| automl_tpot | 0.927 | 0.134 | 0.357 | 0.195 | 0.619 | 0.560 | 0.165 | 7.714 |
| lr_full_oversample | 0.929 | 0.085 | 0.348 | 0.137 | 0.573 | 0.736 | 0.114 | 8.143 |
| lr_full_oversample_cv | 0.927 | 0.088 | 0.309 | 0.137 | 0.720 | 0.717 | 0.111 | 9.143 |
| mlp_full_undersample | 0.897 | 0.210 | 0.214 | 0.212 | 0.971 | 0.724 | 0.157 | 9.571 |
| lr_full_classwts | 0.907 | 0.154 | 0.215 | 0.179 | 0.833 | 0.680 | 0.132 | 9.857 |
| lda_full_oversample | 0.554 | 0.390 | 0.060 | 0.103 | 4.492 | 0.496 | -0.012 | 12.000 |
| lda_full_oversample_cv | 0.547 | 0.392 | 0.059 | 0.102 | 4.281 | 0.489 | -0.014 | 12.429 |

Table 14: Mean Rank Over All Years (2011-2014)

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=14) |
|---|---|---|---|---|---|---|---|---|
| deepfm_full_cv | 0.891 | 0.568 | 0.473 | 0.466 | 0.274 | 0.903 | 0.411 | 2.286 |
| deepfm_full_undersample_cv | 0.884 | 0.723 | 0.410 | 0.517 | 0.275 | 0.923 | 0.458 | 2.429 |
| mlp_full_undersample_cv | 0.884 | 0.432 | 0.305 | 0.341 | 0.285 | 0.817 | 0.278 | 5.429 |
| ensemble_simple | 0.900 | 0.290 | 0.397 | 0.251 | 0.268 | 0.819 | 0.213 | 5.714 |
| xgb_full_undersample_cv | 0.850 | 0.535 | 0.254 | 0.344 | 0.343 | 0.809 | 0.265 | 6.429 |
| lr_full_undersample | 0.871 | 0.463 | 0.274 | 0.339 | 0.370 | 0.800 | 0.269 | 6.857 |
| ensemble_stacked | 0.872 | 0.362 | 0.307 | 0.303 | 0.367 | 0.791 | 0.231 | 7.286 |
| lr_full_oversample | 0.910 | 0.274 | 0.369 | 0.245 | 0.516 | 0.784 | 0.203 | 8.143 |
| automl_tpot | 0.906 | 0.233 | 0.445 | 0.244 | 0.624 | 0.603 | 0.209 | 8.714 |
| lr_full_oversample_cv | 0.909 | 0.278 | 0.340 | 0.247 | 0.626 | 0.769 | 0.202 | 8.857 |
| mlp_full_undersample | 0.880 | 0.379 | 0.259 | 0.295 | 0.952 | 0.774 | 0.228 | 9.000 |
| lr_full_classwts | 0.888 | 0.332 | 0.260 | 0.270 | 0.721 | 0.741 | 0.206 | 9.429 |
| lda_full_oversample | 0.619 | 0.515 | 0.138 | 0.209 | 3.475 | 0.602 | 0.092 | 12.000 |
| lda_full_oversample_cv | 0.614 | 0.516 | 0.137 | 0.208 | 3.317 | 0.597 | 0.092 | 12.429 |

# 6 Discussion

In this study, a range of machine learning methods were applied to predicting household level poverty. Ten commonly used models were evaluated across datasets from two different countries with different poverty rates. The models were combined with common preprocessing techniques to assess their performance. In addition to these standard machine learning approaches, crowd-sourced algorithm development, and three more recently developed techniques—model ensembling, automated machine learning, and deep learning—were explored.

1. There are substantial gains to machine learning approaches over simple regressions when it comes to identifying "Poor" households accurately.
2. While there is no single, stand-out algorithm type, it has been shown that any new models should be benchmarked against logistic regression and gradient-boosted trees.
3. Ensembling methods consistently created more accurate models across the reported investigations as well as the winners of the crowd-sourced algorithm competition.
4. Newer models such as DeepFM hold promise for accurate predictions across many years of household level poverty based on survey data.
5. Neural-network based models performed well both for the years that they were trained on and for other survey years that were used as test sets.

There is much important work still to be done on making these models useful in the context of understanding poverty. For example, an exploration of which variables were most important across model classes could reduce the number of survey questions asked. An analysis of households that were consistently misclassified could suggest the ways in which our collected data could be improved. A comparison across more countries and survey years will illuminate the most consistently accurate approaches. In addition, a complementary comparative assessment that treats the problem as a regression or multi-class classification is an important future direction.

Ultimately, if our goal is to make measurement of poverty more efficient through machine learning, we need to invest in reducing the cost of data acquisition while retaining highly accurate models. Furthermore, these models need to be calibrated in a way that they can be integrated into the work of policymakers and administrators of social benefit programs. Accurate predictions are just one of the import tools in the toolbox of these decision-makers. This report aims to contribute to the democratization of these tools by providing an empirical assessment of these methods and reproducible, openly accessible scripts.[26]

# References


[1] C. Analytics. civisml-extensions. https://github.com/civisanalytics/civisml-extensions, 2017.

[2] T. W. Bank. World bank country and lending groups. https://datahelpdesk.worldbank.org/knowledgebase/articles/906519, 2018. Accessed: 2018-02-13.

[3] J. Bennett, S. Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.

[4] K. J. Boudreau and K. R. Lakhani. Using the crowd as an innovation partner. *Harvard business review*, 91(4):60–9, 2013.

[5] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM, 2016.

[6] M. B. Christopher. *PATTERN RECOGNITION AND MACHINE LEARNING*. Springer-Verlag New York, 2016.

[7] A. V. Dorogush, V. Ershov, and A. Gulin. Catboost: gradient boosting with categorical features support.

[8] I. DrivenData. Pover-t tests: Predicting poverty, 2018. URL https://www.drivendata.org/competitions/50/worldbank-poverty-prediction/.

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[10] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[11] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[12] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. Deepfm: A factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[13] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[14] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3149–3157, 2017.

[15] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[16] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

[17] R. S. Olson and J. H. Moore. Identifying and harnessing the building blocks of machine learning pipelines for sensible initialization of a data science automation tool. *arXiv preprint arXiv:1607.08878*, 2016.

[18] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 485–492, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4206-3. doi: 10.1145/2908812.2908918. URL http://doi.acm.org/10.1145/2908812.2908918.

[19] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore. *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I*, chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pages 123–137. Springer International Publishing, 2016. ISBN 978-3-319-31204-0. doi: 10.1007/978-3-319-31204-0_9. URL http://dx.doi.org/10.1007/978-3-319-31204-0_9.

[20] I. Rechenberg. Evolutionsstrategie–optimierung technisher systeme nach prinzipien der biologischen evolution. 1973.

[21] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010. URL https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf.

[22] G. Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[23] O. Tange. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1): 42–47, Feb 2011. URL http://www.gnu.org/s/parallel.

[24] D. Wolpert. Stacked generalization. 5:241–259, 12 1992.

[25] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[26] Worldbank and Drivendataorg. Ml-comparative-assessment. https://github.com/worldbank/ML-classification-algorithms-poverty, 2018.

# A List of Features

This appendix contains a list of the features used in the Malawi and Indonesia datasets considered throughout this report. Table A lists the Indonesia features. Table A lists the Malawi features. Due to their extensive lengths, both tables are listed on the following pages.

Table 15: INDONESIA – National Socioeconomic Survey – SUSENAS - 2012 (March) by Badan Pusat Statistik (Note: the same features were extracted from the SUSENAS 2011, 2013, and 2014 March datasets)

| Variable | Meaning | Variable | Meaning |
|---|---|---|---|
| hid | Household ID | hid | Household ID in source dataset |
| wta_hh | Household weighting coefficient | wta_hh | Household weighting coefficient |
| poor | Poverty status | poor | Poverty status |
| geo_province | Province | geo_province | Province |
| geo_district | District/City | geo_district | District/City |
| geo_subdistrict | Sub-district | geo_subdistrict | Sub-district |
| geo_urbrur | Area of residence (urban/rural) | geo_village | Village/kelurahan |
| der_hhsize | Household size | geo_urbrur | Area of residence (urban/rural) |
| geo_village | Village/kelurahan | iid | Individual sequential number within household |
| hld_helpers | Household has maid/guard/driver | ind_relat | Relationship to the head |
| hld_building | Type of building (residential or mix use) | ind_sex | Sex |
| hld_occstat | Building occupancy status | ind_age | Age |
| hld_landstat | Land status | ind_marital | Marital status |
| hld_roof | Materials of the roof | ind_certif | Possession of a birth certificate (if age>=17) |
| hld_wall | Materials of the walls | ind_travel | Travel in past 3 monts |
| hld_floor | Materials of the floor | ind_health1 | Had health problem in past month |
| hld_dwater | Main source of drinking water | ind_health2 | Health issue had impact on work/school/activities |
| hld_dwatershr | Sharing of water facility | ind_health3 | Ever performed self-medication |
| hld_dwateracq | Acquisition of water | ind_health4 | Self-medication: traditional medicine |
| hld_bwwater | Source of water for bathing/washing | ind_health5 | Self-medication: modern medicine |
| hld_toiletshr | Sharing of toilet facility | ind_health6 | Outpatient in past 6 months |
| hld_toilet | Type of toilet facility | ind_health7 | Inpatient in past 12 months |
| hld_disposal | Toilet: final disposal location | ind_health8 | Attendance of last birth |
| hld_lighting | Energy for lighting | ind_immu0 | Received any vaccine |
| hld_cooking | Energy for cooking | ind_immu1 | Nb vaccines: BCG |
| hld_bicycle | Own bicycle | ind_immu2 | Nb vaccines: DPT |
| hld_motorcycle | Own motorcycle | ind_immu3 | Nb vaccines: Polio |
| hld_boat | Own boat | ind_immu4 | Nb vaccines: Measles |
| hld_cabletv | Has cable TV | ind_immu5 | Nb vaccines: Hepatitis B |
| hld_aircon | Has air conditioning | ind_educ01 | Ever attended preschool (if 0<=age<=6) |
| hld_waterheat | Own water heater | ind_educ02 | Attended preschool in past 3 months (if 3<=age<=6) |
| hld_gascont | Own gas container >12kg | ind_educ03 | Transport to preschool (if 3<=age<=6 and attended) |
| hld_refrigera∼r | Own refrigerator | ind_educ04 | School attendance (if age >=5) |
| hld_motorboat | Own motorboat | ind_educ05 | Highest education level (passed or currently attended) (if age>=5) |
| hld_car | Own car | ind_educ06 | Highest class (passed or currently attended) (if age>=5) |
| hld_incomeok | Income sufficient for daily needs | ind_educ07 | Highest diploma obtained (if age>=5) |
| hld_cope1 | Cope by using savings | ind_educ08 | Member attended school in past 3 months (if age>=5) |
| hld_cope2 | Cope by selling assets | ind_educ09 | Transport to school (if attended in past 3 months and age>=5) |
| hld_cope3 | Cope by borrowing from family | ind_educ10 | Reason for not attending school (if age 5 to 24) |
| hld_cope4 | Cope by borrowing from friends/neighbors | ind_educ11 | Can read and write in any language (if age>=5) |
| hld_cope5 | Cope by borrowing from creditor | ind_educ12 | Can read and write in latin alphabet (if age>=5) |
| hld_cope6 | Cope by borrowing from bank | ind_educ13 | Can read and write in arab alphabet (if age>=5) |
| hld_cope7 | Cope by borrowing from cooperative | ind_educ14 | Can read and write in other alphabet (if age>=5) |
| hld_cope8 | Cope by pawning goods | ind_educ15 | Accessed internet (anywhere) in past 3 months |

Continued on next page

**Table 15 — continued from previous page**

| Variable | Meaning | Variable | Meaning |
|---|---|---|---|
| hld_anyphone | Any phone in household | ind_educ16 | Accessed internet (home) in past 3 months |
| hld_anycell | At least one member has cell phone | ind_educ17 | Accessed internet (kiosk) in past 3 months |
| hld_nbcell | Number of members with cell phone | ind_educ18 | Accessed internet (office) in past 3 months |
| hld_computer | Household has computer (PC or laptop/notebook) | ind_educ19 | Accessed internet (school) in past 3 months |
| cons_002 | Rice (local, super quality, import) | ind_educ20 | Accessed internet (cell phone) in past 3 months |
| cons_003 | Glutinous rice | ind_work1 | Worked in past week (if age >=10) |
| cons_004 | Fresh corn with skin | ind_work2 | Looked for work in past week (if age >=10) |
| cons_005 | Dryshelled corn/corn rice | ind_work3 | Worked in past 3 months (if age >=10) |
| cons_006 | Rice meal | ind_work4 | Mode of transport to work (if age >=10) |
| cons_007 | Corn flour (cornmeal) | ind_work5 | Days worked in past week (if worked and age >=10) |
| cons_008 | Wheat flour | ind_work6 | Type of business (if worked in past week and age >=10) |
| cons_009 | Other cereals | ind_work7 | Status in occupation (if worked in past week and age >=10) |
| cons_011 | Cassava | ind_age1mar | Age at first marriage (women age 10+ married/divorced/widow) |
| cons_012 | Sweet potato | ind_cborn | Children born alive (women age 10+ married/divorced/widow) |
| cons_013 | Sago flour (non-cassava) | ind_calive | Children still alive (women age 10+ married/divorced/widow) |
| cons_014 | Taro | ind_cdead | Children deceased (women age 10+ married/divorced/widow) |
| cons_015 | Potato | cons_173 | Nutmeg |
| cons_016 | Dried cassava | cons_174 | Clove |
| cons_017 | Flour dried cassava | cons_175 | Fish paste |
| cons_018 | Cassava flour (tapioca) | cons_176 | Soya sauce |
| cons_019 | Other tubers | cons_177 | Monosodium glutamate |
| cons_021 | Yellow tail/fusiliers | cons_178 | Chili sauce/tomato sauce |
| cons_022 | Eastern tuna/skipjack tuna | cons_179 | Spice |
| cons_023 | Mackerel | cons_180 | Other spice |
| cons_024 | Trevallies | cons_182 | Instant noodle |
| cons_025 | Indian Mackerel | cons_183 | Wheat noodle |
| cons_026 | Anchovies | cons_184 | Rice noodle |
| cons_027 | Milk fish | cons_185 | Macaroni |
| cons_028 | Snake head murrel (gabus) | cons_186 | Crisps |
| cons_029 | Mozambique tilapia | cons_187 | Fried chips |
| cons_030 | Common carp | cons_188 | Seaweed |
| cons_031 | Catfish | cons_189 | Porridge in package |
| cons_032 | Barramundi | cons_190 | Others |
| cons_033 | Baronang | cons_192 | Ordinary bread |
| cons_034 | Other fresh fish | cons_193 | Other bread |
| cons_035 | Shrimp | cons_194 | Cookies |
| cons_036 | Common squid/cuttle fish | cons_195 | Boil or steam cake |
| cons_037 | Mud crab/swim crab | cons_196 | Fried food |
| cons_038 | Cockle/snail | cons_197 | Porridge of mung bean |
| cons_039 | Other fresh shrimp and other water animals | cons_198 | Kind of salad with peanuts sauce |
| cons_040 | Indian Mackerel/Peda | cons_199 | A plate of rice accompanied by a mixture of dishes |
| cons_041 | Mackerel | cons_200 | Fried rice |
| cons_042 | Eastern tuna/skipjack tuna | cons_201 | Rice |
| cons_043 | Anchovies | cons_202 | Rice steamed in a banana leaf or coconut leaf |
| cons_044 | Trevallies | cons_203 | Soup Bowl |
| cons_045 | Snakeskin gourame | cons_204 | Roasted meat on skewer/satay |

Continued on next page

**Table 15 — continued from previous page**

| Variable | Meaning | Variable | Meaning |
|---|---|---|---|
| cons_046 | Milk fish | cons_205 | Noodle (with meatball/boiled/fried) |
| cons_047 | Snake head murrel | cons_206 | Instant noodle |
| cons_048 | Canned fish | cons_207 | Snack for children |
| cons_049 | Other preserved fish | cons_208 | Fish (fried, roasted, etc) |
| cons_050 | Shrimp (ebi) | cons_209 | Chicken/meat (fried, roasted, etc) |
| cons_051 | Common squids | cons_210 | Other prepared food |
| cons_052 | Other shrimp and other preserved seafoods | cons_211 | Mineral water (bottle) |
| cons_054 | Beef | cons_212 | Mineral water (gallon) |
| cons_055 | Buffalo meat | cons_213 | Packed tea |
| cons_056 | Lamb meat | cons_214 | Packed juice |
| cons_057 | Pork | cons_215 | CO2 drink |
| cons_058 | Broiler meat | cons_216 | Health drink |
| cons_059 | Local chicken meat | cons_217 | Other drinks (coffee, milk, etc) |
| cons_060 | Other poultry meat | cons_218 | Ice cream |
| cons_061 | Other meat | cons_219 | Other ice |
| cons_062 | Dried beef | cons_220 | Beer |
| cons_063 | Shredded fried meat | cons_221 | Wine |
| cons_064 | Canned meat | cons_222 | Other alcoholic beverage |
| cons_065 | Other preserved meat | cons_224 | Clove filter cigarettes |
| cons_066 | Liver | cons_225 | Clove non filter cigarettes |
| cons_067 | Innards excluding liver | cons_226 | Cigarettes |
| cons_068 | Trimming | cons_227 | Tobacco |
| cons_069 | Bone (untrimmed) | cons_228 | Betel/areca nut |
| cons_070 | Others | cons_229 | Others tobacco and betal |
| cons_072 | Broiler egg | cons_232 | In the case of own / rent-free house, an estimated monthly rent |
| cons_073 | Local chicken egg | cons_233 | In the case of leased house (annually), the average monthly lease |
| cons_074 | Duck egg | cons_234 | In the case of rented house, the monthly rent |
| cons_075 | Quail egg | cons_235 | In the case of free of rent/official house/other, the estimated monthly rent |
| cons_076 | Other egg | cons_236 | Maintenance costs of the house |
| cons_077 | Salted egg | cons_238 | Expenses on electricity |
| cons_078 | Fresh milk | cons_240 | Water (PAM/retail purchase): Value |
| cons_079 | Preserved milk | cons_242 | LPG Value |
| cons_080 | Sweet canned liquid milk | cons_244 | Expenses in city gas |
| cons_081 | Canned powder milk | cons_246 | Expenses on gasoline |
| cons_082 | Baby powder milk | cons_248 | Generator fuel oil |
| cons_083 | Cheese | cons_250 | Expenses on generator fuel used |
| cons_084 | Milk product | cons_251 | Generator maintenance and repair |
| cons_086 | Spinach | cons_253 | Expenses on charcoal/coal/briquette |
| cons_087 | Swamp cabbage | cons_254 | Firewood and other fuels |
| cons_088 | Cabbage | cons_255 | Others (batteries, storage batteries, matches,..) |
| cons_089 | Chinese cabbage | cons_256 | Home telephone bill |
| cons_090 | Mustard greens | cons_257 | Mobile phone bill |
| cons_091 | Beans | cons_258 | Phone card/public phone/phone shop |
| cons_092 | String bean | cons_259 | Post stuff (stamp, etc.) |
| cons_093 | Tomato | cons_260 | Other (Internet Cafe, Internet, etc) |
| cons_094 | Carrot | cons_262 | Bathing soap, toothpaste, toothbrush, and shampoo |

Continued on next page

**Table 15 — continued from previous page**

| Variable | Meaning | Variable | Meaning |
|---|---|---|---|
| cons_095 | Cucumber | cons_263 | Cosmetic articles and sanitary napkin |
| cons_096 | Cassava leaf | cons_264 | Treatment of skin, face, nails, hair |
| cons_097 | Aubergine | cons_265 | Laundry soap (bars, powders, creams, and liquid) |
| cons_098 | Bean sprout | cons_266 | Clothes maintenance material |
| cons_099 | Squash | cons_267 | Newspapers, magazine, books, and stationeries (exluding for education) including |
| cons_100 | Unripe corn | cons_268 | Other stuffs (tissue, baby diaper, satai stick, etc.) |
| cons_101 | Soup/stir-fried vegetables | cons_269 | Public hospitals |
| cons_102 | Sour vegetable soup | cons_270 | Private hospitals |
| cons_103 | Young jackfruit | cons_271 | Sub ordinary Public Health Center |
| cons_104 | Unripe papaya | cons_272 | Medical Doctor private/public hospital |
| cons_105 | Mushroom | cons_273 | Paramedical |
| cons_106 | Petai beans | cons_274 | Traditional Treatment |
| cons_107 | Stink beans | cons_275 | Traditional birth attendant |
| cons_108 | Onion | cons_276 | Take medicine with recipe |
| cons_109 | Garlic | cons_277 | Self treatment/take medicine without recipe |
| cons_110 | Chillies | cons_278 | Purchasing traditional medicine |
| cons_111 | Green chili | cons_279 | Purchasing glasses, hand/leg artificial, and wheel chair |
| cons_112 | Cayenne pepper | cons_280 | Pregnancy examination cost |
| cons_113 | Canned vegetable | cons_281 | Children Under-fives immunization cost |
| cons_114 | Others vegetables | cons_282 | Medical check-up |
| cons_116 | Peanuts without shell | cons_283 | Contraception cost |
| cons_117 | Peanuts with shell | cons_284 | Take care of health (vitamin, medicine herbs, etc.) |
| cons_118 | Soybean | cons_285 | Development school contribution/admission fee |
| cons_119 | Mungbean | cons_286 | School fee |
| cons_120 | Red kidney bean | cons_287 | Other cost of school contribution |
| cons_121 | Other bean | cons_288 | Textbooks, photocopies of learning materials |
| cons_122 | Tofu, soybean curd | cons_289 | Stationery (pen, pencil, eraser, ruler, calculator, dividers, etc) |
| cons_123 | Fermented soybean cake | cons_290 | Non-formal education cost |
| cons_124 | Fermented soybean paste | cons_292 | Gasoline/pertamax |
| cons_125 | Fermented soya cake | cons_294 | Diesel fuel |
| cons_126 | Others legumes | cons_296 | Lubricant |
| cons_128 | Orange | cons_297 | Motor vehicle's service and repair |
| cons_129 | Mango | cons_298 | Transport expenses |
| cons_130 | Apple | cons_299 | Hotel, inn, cinema, theater, sports, set-top box, cable TV subscriptions |
| cons_131 | Avocado | cons_300 | Domestic servant, security, and driver (salary or wages) |
| cons_132 | Rambutan | cons_301 | Financial service charge |
| cons_133 | Lanzon | cons_302 | Other services (ID card, driver's license, birth certificate, copy, photo, etc.) |
| cons_134 | Durian | cons_304 | Ready-made clothes for men |
| cons_135 | Zalacca | cons_305 | Ready-made clothes for women |
| cons_136 | Pineapple | cons_306 | Ready-made clothes for children |
| cons_137 | Ambon | cons_307 | Materials clothing for men, women, and children |
| cons_138 | Raja | cons_308 | Wages sewing, repairing clothes, sewing thread, and other goods for the purposes |
| cons_139 | Other banana | cons_309 | Footwear |
| cons_140 | Papaya | cons_310 | Headgear for men, women, and children (hat, cap, scarf, etc..) |
| cons_141 | Rose-apple | cons_311 | Others (towel, belt, shoe polish, tie, laundry, etc.) |
| cons_142 | Sapodilla | cons_313 | Furniture |

Continued on next page

**Table 15 — continued from previous page**

| Variable | Meaning | Variable | Meaning |
|---|---|---|---|
| cons_143 | Carambola | cons_314 | Household appliances (sewing machines, refrigerators,..) |
| cons_144 | Spanish plum | cons_315 | Household equipments (mattresses, pillows, tablecloths, ..) |
| cons_145 | Watermelon | cons_316 | Home appliances (iron, broom, scissors, knives, machetes, hoes, saws, vacuum cle |
| cons_146 | Melon | cons_317 | Kitchen utensils |
| cons_147 | Jack fruit | cons_318 | Decoration stuff |
| cons_148 | Tomato | cons_319 | Furniture and utensils repairs |
| cons_149 | Canned fruit | cons_320 | Hand phone and other accessories |
| cons_150 | Others fruits | cons_321 | Watch, clock, camera, glasses, and repairs |
| cons_152 | Coconut oil | cons_322 | Umbrella, bag, and repairs |
| cons_153 | Corn oil | cons_323 | Jewelry and repairs |
| cons_154 | Other frying oil | cons_324 | Toys and repair, imitation jewelry |
| cons_155 | Coconut | cons_325 | Electronics (Television, radio,..) and repair |
| cons_156 | Margarine | cons_326 | Tools and sports equipment and repairs |
| cons_157 | Others oil and fats | cons_327 | Vehicles (cars, motorcycles, bicycles, etc..) and major repairs |
| cons_159 | Cane sugar | cons_328 | Domestic animal and plant maintenance |
| cons_160 | Brown sugar | cons_329 | Other durable goods (electrical installation / phone / tap, swing, stroller, etc |
| cons_161 | Tea | cons_331 | Buildings and land taxes |
| cons_162 | Powdered/bean coffee | cons_332 | Motor and non-motor vehicle taxes |
| cons_163 | Instant cocoa | cons_333 | Other contributions (dues RT / RW, trash, security, cemetery, parking, etc..) |
| cons_164 | Powdered cocoa | cons_334 | Health insurance |
| cons_165 | Syrup | cons_335 | Live insurance and general insurance |
| cons_166 | Others beverages stuff | cons_336 | Others (ticket, Income Tax, etc..) |
| cons_168 | Salt | cons_338 | Wedding |
| cons_169 | Candlenut | cons_339 | Circumcision and birthday |
| cons_170 | Coriander | cons_340 | Religious festival (chair rental, tent rental, etc..) |
| cons_171 | Pepper | cons_341 | Pilgrimage cost (ONH) |
| cons_172 | Tamarind | cons_342 | Religious/traditional ceremony |
| | | cons_343 | Funeral expenses |

Table 16: MALAWI – Third Integrated Household Survey 2010-2011 by the National Statistics Office

| Variable | Meaning | Variable | Meaning |
|---|---|---|---|
| hld_toilet | What kind of toilet facility does your household use? | ind_educ07 | Are you currently attending school or, if school is not now in session, did you |
| hld_toiletshr | Is this toilet facility for the use of : | ind_educ08 | Why did you not continue your education? |
| hld_rubbish | What kind of rubbish disposal facilities does your household use | ind_educ09 | What type of school do you attend? |
| hld_bednet | Do any members of your household sleep under a bed net to protect against mosqui | ind_educ10 | How do you get to school each day? |
| geo_urbrur | Area of residence (urban/rural) | ind_educ11 | At any time in the past 12 months, did you ever temporarily withdraw from school |
| geo_district | District | ind_educ12 | What was the main reason you temporarily withdrew from school? |
| hld_nbguests | Past 7 days, any people not listed as HH memebr eat any meal in your HH | ind_health1 | During the last 12 months, were you hospital-ized or had an overnight stay(s) in |
| cons_0101 | Maize ufa mgaiwa (normal flour) | ind_health2 | Did you or other members of your household have to borrow money or sell assets i |
| cons_0102 | Maize ufa refined (fine flour) | ind_health3 | During the last 12 months, did you stay over-night(s) at a traditional healer's |
| cons_0103 | Maize ufa madeya (bran flour) | ind_health4 | Did you or other members of your household have to borrow money or sell assets i |
| cons_0104 | Maize grain (not as ufa) | ind_health5 | Enumerator: check questions HH_D24 through HH_D29. Did the respondent have any d |
| cons_0105 | Green maize | ind_health6 | Does this difficulty reduce the amount of work you can do at home? |
| cons_0106 | Rice | ind_health7 | Does this difficulty reduce the amount of work you can do at school? |
| cons_0107 | Finger millet (mawere) | ind_health8 | Does this difficulty reduce the amount of work you can do at work? |
| cons_0108 | Sorghum (mapira) | ind_breakfast | What did you have for breakfast yesterday? |
| cons_0109 | Pearl millet (mchewere) | ind_birthplace | Where did you deliver your last child born in the last 24 months? |
| cons_0110 | Wheat flour | ind_birthattend | Who assisted in delivering this child? |
| cons_0111 | Bread | ind_work1 | How many hours in the last seven days did you do any work for a wage, salary, co |
| cons_0112 | Buns, scones | ind_work2 | Enumerator: review questions HH_E07 to HH_E11. Did this person, [NAME], work for |
| cons_0113 | Biscuits | ind_work3 | Even though you did not do any activities in the last seven days, do you have a |
| cons_0114 | Spaghetti, macaroni, pasta | ind_work4 | At any time in the past 12 months, were you employed for a wage, salary, commiss |
| cons_0115 | Breakfast cereal | ind_work5 | Is your main employer for your main occupation in the last 12 months ... |
| cons_0116 | Infant feeding cereals | ind_work6 | At any time over the last 12 months, did you work on any farm(s) owned by anothe |
| cons_0117 | Other (specify) | wta_hh | Household weighting coefficient |
| cons_0201 | Cassava tubers | poor | Poverty status |
| cons_0202 | Cassava flour | cons_1317 | Lady's blouse/shirt |
| cons_0203 | White sweet potato | cons_1318 | Chitenje cloth |
| cons_0204 | Orange sweet potato | cons_1319 | Lady's dress/skirt |
| cons_0205 | Irish potato | cons_1320 | Lady's undergarments |
| cons_0206 | Potato crisps | cons_1321 | Lady's other clothing |
| cons_0207 | Plantain, cooking banana | cons_1322 | Boy's shoes |
| cons_0208 | Cocoyam (masimbi) | cons_1323 | Men's shoes |
| cons_0209 | Other (specify) | cons_1324 | Girl's shoes |
| cons_0301 | Bean, white | cons_1325 | Lady's shoes |
| cons_0302 | Bean, brown | cons_1326 | Cloth, thread, other sewing material |
| cons_0303 | Pigeonpea (nandolo) | cons_1327 | Laundry, dry cleaning, tailoring fees |
| cons_0304 | Groundnut | cons_1328 | Bowls, glassware, plates, silverware, etc. |
| cons_0305 | Groundnut flour | cons_1329 | Cooking utensils (cookpots, stirring spoons and wisks, etc.) |
| cons_0306 | Soyabean flour | cons_1330 | Cleaning utensils (brooms, brushes, etc.) |
| cons_0307 | Ground bean (nzama) | cons_1331 | Torch / flashlight |
| cons_0308 | Cowpea (khobwe) | cons_1332 | Umbrella |
| cons_0309 | Macademia nuts | cons_1333 | Paraffin lamp (hurricane or pressure) |
| cons_0310 | Other (specify) | cons_1334 | Stationery items (not for school) |
| cons_0401 | Onion | cons_1335 | Books (not for school) |
| cons_0402 | Cabbage | cons_1336 | Music or video cassette or CD/DVD |

Continued on next page

**Table 16 — continued from previous page**

| Variable | Meaning | Variable | Meaning |
|---|---|---|---|
| cons_0403 | Tanaposi/Rape | cons_1337 | Tickets for sports / entertainment events |
| cons_0404 | Nkhwani | cons_1338 | House decorations |
| cons_0405 | Chinese cabbage | cons_1339 | Night's lodging in rest house or hotel |
| cons_0406 | Other cultivated green leafy vegetables | cons_1401 | Carpet, rugs, drapes, curtains |
| cons_0407 | Gathered wild green leaves | cons_1402 | Linen - towels, sheets, blankets |
| cons_0408 | Tomato | cons_1403 | Mat - sleeping or for drying maize flour |
| cons_0409 | Cucumber | cons_1404 | Mosquito net |
| cons_0410 | Pumpkin | cons_1405 | Mattress |
| cons_0411 | Okra / Therere | cons_1406 | Sports & hobby equipment, musical instruments, toys |
| cons_0412 | Tinned vegetables (Specify) | cons_1407 | Film, film processing, camera |
| cons_0413 | Mushroom | cons_1408 | Cement |
| cons_0414 | Other vegetables (Specify) | cons_1409 | Bricks |
| cons_0501 | Eggs | cons_1410 | Construction timber |
| cons_0502 | Dried fish | cons_1411 | Council rates |
| cons_0503 | Fresh fish | cons_1412 | Insurance - health (MASM, etc.), auto, home, life |
| cons_0504 | Beef | cons_1413 | Losses to theft (value of items or cash lost) |
| cons_0505 | Goat | cons_1414 | Fines or legal fees |
| cons_0506 | Pork | cons_1415 | Lobola (bridewealth) costs |
| cons_0507 | Mutton | cons_1416 | Marriage ceremony costs |
| cons_0508 | Chicken | cons_1417 | Funeral costs, household members |
| cons_0509 | Other poultry - guinea fowl, doves, etc. | cons_1418 | Funeral costs, nonhousehold members (relatives, neighbors/friends) |
| cons_0510 | Small animal - rabbit, mice, etc. | cons_1419 | Woodpoles, bamboo |
| cons_0511 | Termites, other insects (eg Ngumbi, caterpillar) | cons_1420 | Grass for thatching roof or other use |
| cons_0512 | Tinned meat or fish | own_501 | Mortar/pestle (mtondo) |
| cons_0513 | Smoked fish | own_502 | Bed |
| cons_0514 | Fish Soup/Sauce | own_503 | Table |
| cons_0515 | Other (specify) | own_504 | Chair |
| cons_0601 | Mango | own_505 | Fan |
| cons_0602 | Banana | own_506 | Air conditioner |
| cons_0603 | Citrus - naartje, orange, etc. | own_507 | Radio ('wireless') |
| cons_0604 | Pineapple | own_508 | Tape or CD/DVD player; HiFi |
| cons_0605 | Papaya | own_509 | Television |
| cons_0606 | Guava | own_510 | VCR |
| cons_0607 | Avocado | own_511 | Sewing machine |
| cons_0608 | Wild fruit (masau, malambe, etc.) | own_512 | Kerosene/paraffin stove |
| cons_0609 | Apple | own_513 | Electric or gas stove; hot plate |
| cons_0610 | Other fruits (specify) | own_514 | Refrigerator |
| cons_0701 | Fresh milk | own_515 | Washing machine |
| cons_0702 | Powdered milk | own_516 | Bicycle |
| cons_0703 | Margarine - Blue band | own_517 | Motorcycle/scooter |
| cons_0704 | Butter | own_518 | Car |
| cons_0705 | Chambiko - soured milk | own_519 | Mini-bus |
| cons_0706 | Yoghurt | own_520 | Lorry |
| cons_0707 | Cheese | own_521 | Beer-brewing drum |
| cons_0708 | Infant feeding formula (for bottle) | own_522 | Upholstered chair, sofa set |
| cons_0709 | Other (specify) | own_523 | Coffee table (for sitting room) |

Continued on next page

**Table 16 — continued from previous page**

| Variable | Meaning | Variable | Meaning |
|---|---|---|---|
| cons_0801 | Sugar | own_524 | Cupboard, drawers, bureau |
| cons_0802 | Sugar Cane | own_525 | Lantern (paraffin) |
| cons_0803 | Cooking oil | own_526 | Desk |
| cons_0804 | Other (specify) | own_527 | Clock |
| cons_0810 | Salt | own_528 | Iron (for pressing clothes) |
| cons_0811 | Spices | own_529 | Computer equipment & accessories |
| cons_0812 | Yeast, baking powder, bicarbonate of soda | own_530 | Sattelite dish |
| cons_0813 | Tomato sauce (bottle) | own_531 | Solar panel |
| cons_0814 | Hot sauce (Nali, etc.) | own_532 | Generator |
| cons_0815 | Jam, jelly | hld_foodsecur∼y | In the past 7 days, did you worry that your household would not have enough food |
| cons_0816 | Sweets, candy, chocolates | farm_601 | Hand hoe |
| cons_0817 | Honey | farm_602 | Slasher |
| cons_0818 | Other (specify) | farm_603 | Axe |
| cons_0820 | Maize - boiled or roasted (vendor) | farm_604 | Sprayer |
| cons_0821 | Chips (vendor) | farm_605 | Panga knife |
| cons_0822 | Cassava - boiled (vendor) | farm_606 | Sickle |
| cons_0823 | Eggs - boiled (vendor) | farm_607 | Treadle pump |
| cons_0824 | Chicken (vendor) | farm_608 | Watering can |
| cons_0825 | Meat (vendor) | farm_609 | Ox cart |
| cons_0826 | Fish (vendor) | farm_610 | Ox plough |
| cons_0827 | Mandazi, doughnut (vendor) | farm_611 | Tractor |
| cons_0828 | Samosa (vendor) | farm_612 | Tractor plough |
| cons_0829 | Meal eaten at restaurant | farm_613 | Ridger |
| cons_0830 | Other (specify) | farm_614 | Cultivator |
| cons_0901 | Tea | farm_615 | Generator |
| cons_0902 | Coffee | farm_616 | Motorized pump |
| cons_0903 | Cocoa, millo | farm_617 | Grain mill |
| cons_0904 | Squash (Sobo drink concentrate) | farm_618 | Other (specify) |
| cons_0905 | Fruit juice | farm_619 | Chicken house |
| cons_0906 | Freezes (flavoured ice) | farm_620 | Livestock kraal |
| cons_0907 | Soft drinks (Coca-cola, Fanta, Sprite, etc.) | farm_621 | Poultry kraal |
| cons_0908 | Chibuku (commercial traditional-style beer) | farm_622 | Storage house |
| cons_0909 | Bottled water | farm_623 | Granary |
| cons_0910 | Maheu | farm_624 | Barn |
| cons_0911 | Bottled / canned beer (Carlsberg, etc.) | farm_625 | Pig sty |
| cons_0912 | Thobwa | hld_busin1 | Over the past 12 months, has anyone in your household operated any non-agricultu |
| cons_0913 | Traditional beer (masese) | hld_busin2 | Owned a non-agricultural business or provided a non-agricultural service from ho |
| cons_0914 | Wine or commercial liquor | hld_busin3 | Processed and sold any agricultural by-products, including flour, starch, juice, |
| cons_0915 | Locally brewed liquor (kachasu) | hld_busin4 | Owned a trading business on a street or in a market? |
| cons_0916 | Other (specify) | hld_busin5 | Offered any service or sold anything on a street or in a market, including firew |
| cons_1101 | Charcoal | hld_busin6 | Owned a professional office or offered professional services from home as a doct |
| cons_1102 | Paraffin or kerosene | hld_busin7 | Drove a household-owned taxi or pick-up truck to provide transportation or movin |
| cons_1103 | Cigarettes or other tobacco | hld_busin8 | Owned a bar or restaurant? |
| cons_1104 | Candles | hld_busin9 | Owned any other non-agricultural business, even if it is a small business run fr |
| cons_1105 | Matches | inc_101 | Cash Transfers/Gifts from individuals |
| cons_1106 | Newspapers or magazines | inc_102 | Food Transfers/Gifts from Individuals |

Continued on next page

**Table 16 — continued from previous page**

| Variable | Meaning | Variable | Meaning |
|---|---|---|---|
| cons_1107 | Public transport - Bicycle Taxi | inc_103 | Non-Food In-Kind Transfers/Gifts form Individuals |
| cons_1108 | Public transport - Bus/Minibus | inc_104 | Savings, Interest or Other Investment Income |
| cons_1109 | Public transport - Other (Truck, Oxcart, Etc..) | inc_105 | Pension Income |
| cons_1201 | Milling fees, grain | inc_106 | Income from Non-Agricultural Land Rental |
| cons_1202 | Bar soap (body soap or clothes soap) | inc_107 | Income form Apartment, House Rental |
| cons_1203 | Clothes soap (powder) | inc_108 | Income from Sho, Store Rental |
| cons_1204 | Toothpaste, toothbrush | inc_109 | Income from Car, Truck, Other Vehicle Rental |
| cons_1205 | Toilet paper | inc_110 | Income from Real Estate Sales |
| cons_1206 | Glycerine, Vaseline, skin creams | inc_111 | Income from Household Non-Agricultural Asset Sales |
| cons_1207 | Other personal products (shampoo, razor blades, cosmetics, hair products, etc.) | inc_112 | Income from Household Agricultural/Fishing Asset Sales |
| cons_1209 | Light bulbs | inc_113 | Inheritance |
| cons_1210 | Postage stamps or other postal fees | inc_114 | Lottery/Gambling Winnings |
| cons_1211 | Donation - to church, charity, beggar, etc. | inc_115 | Other Income (Specify) |
| cons_1212 | Petrol or diesel | gifts201 | Cash Transfers/Gifts |
| cons_1213 | Motor vehicle service, repair, or parts | gifts202 | Food Transfers/Gifts |
| cons_1214 | Bicycle service, repair, or parts | gifts203 | Non-Food In-Kind Transfers/Gifts |
| cons_1215 | Wages paid to servants | hld_credit1 | During the last 12 months did you try to borrow from someone outside the househo |
| cons_1216 | Mortgage - regular payment to purchase house | hld_credit2 | What was main reason for trying to obtain the loan? |
| cons_1217 | Repairs & maintenance to dwelling | hld_adeqfood | Concerning your household's food consumption, over the past months which of the |
| cons_1218 | Repairs to household and personal items (radios, watches, etc., excluding batter | hld_adeqhous | Concerning your housing which of the following is true? |
| cons_1219 | Expenditures on pets | hld_adeqcloth | Concerning your household's clothing, which of the following is true? |
| cons_1220 | Batteries | hld_adeqhealth | Concerning the standard of health care you receive for household members which o |
| cons_1221 | Recharging batteries, cell phones | hld_selfscale | On which step are you today? |
| cons_1301 | Infant clothing | hld_selfincome | Which of the following is true? Your current income ... |
| cons_1302 | Baby nappies/diapers | hld_headsleep | What do you (HH head) sleep on? |
| cons_1303 | Boy's trousers | com_roadtype | What is the type of main access road surface in this community? |
| cons_1304 | Boy's shirts | com_vehicles | Do vehicles pass on the main road in this community throughout the year? |
| cons_1305 | Boy's jackets | com_bus | Do public buses, mini-buses, or regular matola stop in this community? |
| cons_1306 | Boy's undergarments | com_urbancenter | Is the community in a major urban centre? |
| cons_1307 | Boy's other clothing | com_dailymrkt | Is there a daily market in this community? |
| cons_1308 | Men's trousers | com_weeklymrkt | Is there a larger weekly market in this community? |
| cons_1309 | Men's shirts | com_postoffice | Is there a post office in this community |
| cons_1310 | Men's jackets | com_publicphone | Is there a place to make a telephone call here?(Public phone,phone bureau,etc.) |
| cons_1311 | Men's undergarments | com_distprimary | Distance to nearest government primary school |
| cons_1312 | Men's other clothing | com_classrooms | At nearest gov prim school,are all classrooms built of brick w/iron sheet roofs? |
| cons_1313 | Girl's blouse/shirt | com_schoolelec | Is the nearest government primary school electrified? |
| cons_1314 | Girl's dress/skirt | com_medicines | Is there a place here to purchase common medicines(pain killers,malaria tablets) |
| cons_1315 | Girl's undergarments | com_clinic | Is there a health clinic (Chipatala) is this community? |
| cons_1316 | Girl's other clothing | com_distclinic | Distance to nearest health clinic |
| | | com_bank | Is there a commercial bank in this community (NBM, Savings Bank, Stanbic, etc.)? |

# B Full Results from Comparison of Algorithms Using Class Balancing, Tuning, CV, and Feature Selection

Full Malawi Results with Full Feat., Class Balancing, Tuning, CV, and Feat. Selection

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=35) |
|---|---|---|---|---|---|---|---|---|
| svm_full_cv | 0.874 | 0.894 | 0.878 | 0.886 | 0.287 | 0.949 | 0.758 | 5.000 |
| mlp_full_cv | 0.871 | 0.895 | 0.874 | 0.884 | 0.278 | 0.952 | 0.752 | 6.125 |
| xgb_feats | 0.872 | 0.892 | 0.877 | 0.884 | 0.289 | 0.949 | 0.754 | 7.375 |
| svm_full_cv_isotonic | 0.871 | 0.891 | 0.876 | 0.883 | 0.288 | 0.949 | 0.754 | 7.625 |
| lr_full_wts_cv | 0.873 | 0.892 | 0.879 | 0.885 | 0.301 | 0.944 | 0.734 | 7.750 |
| xgb_full_cv | 0.869 | 0.894 | 0.870 | 0.882 | 0.296 | 0.948 | 0.751 | 9.125 |
| svm_full | 0.864 | 0.886 | 0.868 | 0.877 | 0.298 | 0.945 | 0.733 | 10.625 |
| lr_full | 0.874 | 0.870 | 0.854 | 0.862 | 0.288 | 0.949 | 0.746 | 12.750 |
| rf_full_wts_cv_ada | 0.866 | 0.878 | 0.878 | 0.878 | 0.580 | 0.947 | 0.744 | 13.000 |
| dt_full_wts_cv_ada | 0.866 | 0.878 | 0.878 | 0.878 | 0.353 | 0.941 | 0.737 | 13.000 |
| xgb_full | 0.860 | 0.886 | 0.862 | 0.874 | 0.322 | 0.937 | 0.732 | 13.125 |
| dl_full | 0.860 | 0.930 | 0.834 | 0.879 | 0.602 | 0.930 | 0.721 | 14.625 |
| rf_feats | 0.863 | 0.865 | 0.884 | 0.874 | 0.467 | 0.944 | 0.744 | 15.000 |
| dt_feats | 0.861 | 0.877 | 0.871 | 0.874 | 0.353 | 0.940 | 0.732 | 15.375 |
| lda_full | 0.859 | 0.884 | 0.863 | 0.873 | 0.327 | 0.934 | 0.719 | 15.500 |
| lda_full_cv | 0.859 | 0.884 | 0.863 | 0.873 | 0.327 | 0.934 | 0.719 | 15.500 |
| lr_l2_feats_wts_cv | 0.852 | 0.879 | 0.855 | 0.867 | 0.312 | 0.939 | 0.690 | 16.500 |
| lr_l1_feats_wts_cv | 0.863 | 0.891 | 0.864 | 0.877 | 2.008 | 0.843 | 0.712 | 17.750 |
| lr_full_wts | 0.859 | 0.871 | 0.872 | 0.871 | 0.365 | 0.934 | 0.706 | 18.750 |
| lr_feats40_wts_cv | 0.844 | 0.875 | 0.846 | 0.860 | 0.328 | 0.933 | 0.684 | 18.875 |
| lda_feats_cv | 0.836 | 0.873 | 0.836 | 0.854 | 0.357 | 0.919 | 0.668 | 20.000 |
| mlp_full | 0.853 | 0.852 | 0.877 | 0.864 | 0.669 | 0.940 | 0.717 | 20.750 |
| rf_full_wts | 0.832 | 0.864 | 0.835 | 0.849 | 0.393 | 0.920 | 0.657 | 21.875 |
| rf_full_wts_cv | 0.814 | 0.874 | 0.804 | 0.837 | 0.407 | 0.911 | 0.602 | 22.000 |
| dl_full_cv | 0.845 | 0.806 | 0.900 | 0.850 | 0.798 | 0.938 | 0.702 | 22.125 |
| rf_full | 0.830 | 0.860 | 0.835 | 0.848 | 0.392 | 0.920 | 0.650 | 22.375 |
| knn_full | 0.780 | 0.960 | 0.727 | 0.827 | 0.753 | 0.890 | 0.527 | 22.375 |
| knn_full_cv | 0.779 | 0.933 | 0.736 | 0.823 | 2.104 | 0.848 | 0.527 | 23.875 |
| nb_full_cv_platt | 0.795 | 0.853 | 0.791 | 0.821 | 0.520 | 0.873 | 0.513 | 26.000 |
| nb_full_cv | 0.794 | 0.859 | 0.786 | 0.821 | 2.475 | 0.873 | 0.509 | 27.062 |
| nb_l2_feats_cv_platt | 0.768 | 0.866 | 0.750 | 0.804 | 0.549 | 0.838 | 0.446 | 27.250 |
| nb_full | 0.794 | 0.859 | 0.785 | 0.821 | 2.472 | 0.873 | 0.509 | 27.312 |
| dt_full_wts_cv | 0.801 | 0.806 | 0.827 | 0.817 | 0.634 | 0.867 | 0.619 | 29.125 |
| dt_full_wts | 0.763 | 0.791 | 0.781 | 0.786 | 3.369 | 0.815 | 0.528 | 32.125 |
| dt_full | 0.779 | 0.782 | 0.809 | 0.795 | 3.577 | 0.825 | 0.568 | 32.375 |

Table 17: Indonesia model performance across six metrics, using all available features and various class-balancing techniques as well as cross-validated training. Models are sorted by their *mean rank*, as defined in Sec. 2. See Table 2 for list of abbreviations, Table 1.4 for summary of data.

# C Poverty Rate Error

The poverty rate error measurement discussed in Sec 2.1 is non-standard and therefore not included in the primary results of this report. However, it is sometimes very small (meaning the models can accurately estimate national poverty) even though precision and recall may not be very high. These values are reported in this appendix. For convenience, the results are not resorted, but reported with the same sorting as the tables in Appendix B.

Full Indonesia Results with Class Balancing, Tuning, CV, and Feature Selection

| | accuracy | recall | precision | f1 | cross_entropy | roc_auc | cohen_kappa | mean_rank (N=32) |
|---|---|---|---|---|---|---|---|---|
| lr_full_oversample | 0.856 | 0.840 | 0.436 | 0.574 | 0.344 | 0.927 | 0.477 | 7.875 |
| mlp_full_undersample_cv | 0.825 | 0.902 | 0.389 | 0.543 | 0.370 | 0.930 | 0.434 | 8.375 |
| lr_full_oversample_cv | 0.852 | 0.841 | 0.429 | 0.568 | 0.348 | 0.926 | 0.474 | 8.500 |
| xgb_full_undersample_cv | 0.832 | 0.874 | 0.397 | 0.546 | 0.379 | 0.928 | 0.448 | 8.625 |
| lr_full_undersample | 0.826 | 0.892 | 0.389 | 0.542 | 0.394 | 0.927 | 0.434 | 9.625 |
| lr_full_classwts | 0.831 | 0.867 | 0.394 | 0.542 | 0.385 | 0.923 | 0.428 | 11.000 |
| mlp_full_undersample | 0.828 | 0.887 | 0.392 | 0.544 | 0.894 | 0.922 | 0.450 | 11.625 |
| lda_full_oversample_cv | 0.815 | 0.890 | 0.374 | 0.526 | 0.429 | 0.922 | 0.407 | 12.000 |
| lr_l1_feats_oversample_cv | 0.833 | 0.835 | 0.395 | 0.536 | 0.378 | 0.915 | 0.408 | 12.250 |
| lda_full_oversample | 0.813 | 0.886 | 0.371 | 0.523 | 0.426 | 0.922 | 0.406 | 12.375 |
| lda_full_undersample | 0.796 | 0.908 | 0.351 | 0.507 | 0.464 | 0.922 | 0.382 | 12.500 |
| mlp_full_oversample | 0.895 | 0.600 | 0.542 | 0.569 | 0.660 | 0.912 | 0.529 | 12.625 |
| xgb_full_oversample | 0.878 | 0.601 | 0.476 | 0.531 | 0.285 | 0.899 | 0.456 | 12.750 |
| xgb_full_undersample | 0.798 | 0.892 | 0.352 | 0.505 | 0.406 | 0.915 | 0.378 | 12.750 |
| dl_full_oversample | 0.846 | 0.789 | 0.413 | 0.543 | 0.761 | 0.901 | 0.454 | 13.875 |
| dl_full_undersample_cv | 0.781 | 0.934 | 0.338 | 0.497 | 0.555 | 0.917 | 0.374 | 14.000 |
| dt_full_undersample_cv_ada | 0.795 | 0.882 | 0.347 | 0.498 | 0.436 | 0.910 | 0.387 | 14.625 |
| dl_full_undersample | 0.729 | 0.939 | 0.291 | 0.445 | 0.578 | 0.900 | 0.322 | 16.125 |
| lda_feats_oversample_cv | 0.779 | 0.851 | 0.326 | 0.471 | 0.461 | 0.885 | 0.304 | 17.000 |
| knn_full_undersample_cv | 0.626 | 0.959 | 0.231 | 0.372 | 1.098 | 0.878 | 0.204 | 20.625 |
| dt_full_undersample_cv | 0.700 | 0.848 | 0.258 | 0.395 | 0.766 | 0.836 | 0.258 | 20.750 |
| lda_full_classwts | 0.506 | 0.989 | 0.188 | 0.316 | 1.166 | 0.912 | 0.150 | 20.750 |
| knn_full_undersample | 0.636 | 0.927 | 0.231 | 0.370 | 2.245 | 0.856 | 0.210 | 21.750 |
| dt_full_classwts | 0.789 | 0.726 | 0.319 | 0.443 | 1.485 | 0.774 | 0.321 | 22.375 |
| nb_l2_feats_oversample_cv_platt | 0.730 | 0.749 | 0.264 | 0.391 | 0.492 | 0.814 | 0.183 | 22.750 |
| dt_full_undersample | 0.727 | 0.805 | 0.271 | 0.405 | 3.877 | 0.791 | 0.270 | 22.875 |
| nb_full_oversample_cv_platt | 0.767 | 0.652 | 0.281 | 0.392 | 0.490 | 0.814 | 0.217 | 23.125 |
| nb_full_undersample | 0.788 | 0.669 | 0.308 | 0.422 | 2.213 | 0.827 | 0.223 | 23.125 |
| nb_full_oversample | 0.764 | 0.665 | 0.281 | 0.395 | 2.056 | 0.814 | 0.214 | 23.625 |
| nb_full_oversample_cv | 0.764 | 0.665 | 0.280 | 0.394 | 2.055 | 0.814 | 0.215 | 23.750 |
| knn_full_oversample | 0.412 | 0.984 | 0.163 | 0.279 | 8.634 | 0.792 | 0.103 | 24.000 |
| dt_full_oversample | 0.690 | 0.646 | 0.217 | 0.325 | 8.419 | 0.657 | 0.199 | 30.000 |

Table 18: Indonesia model performance across six metrics, using all available features and various class-balancing techniques as well as cross-validated training. Models are sorted by their *mean rank*, as defined in Sec. 2. See Table 2 for list of abbreviations, Table 1.4 for summary of data.

Malawi Poverty Rate Error Tuning, CV, and Feature Selection

| | recall | precision | pov_rate_error |
|---|---|---|---|
| svm_full_cv | 0.894 | 0.878 | 0.013 |
| mlp_full_cv | 0.895 | 0.874 | 0.014 |
| xgb_feats | 0.892 | 0.877 | 0.008 |
| svm_full_cv_isotonic | 0.891 | 0.876 | 0.013 |
| lr_full_wts_cv | 0.892 | 0.879 | 0.014 |
| xgb_full_cv | 0.894 | 0.870 | 0.010 |
| svm_full | 0.886 | 0.868 | 0.018 |
| lr_full | 0.870 | 0.854 | 0.010 |
| rf_full_wts_cv_ada | 0.878 | 0.878 | 0.000 |
| dt_full_wts_cv_ada | 0.878 | 0.878 | 0.003 |
| xgb_full | 0.886 | 0.862 | 0.019 |
| dl_full | 0.930 | 0.834 | 0.044 |
| rf_feats | 0.864 | 0.884 | -0.002 |
| dt_feats | 0.877 | 0.871 | 0.004 |
| lda_full | 0.884 | 0.863 | 0.015 |
| lda_full_cv | 0.884 | 0.863 | 0.015 |
| lr_l2_feats_wts_cv | 0.879 | 0.855 | 0.016 |
| lr_l1_feats_wts_cv | 0.891 | 0.864 | 0.016 |
| lr_full_wts | 0.871 | 0.871 | 0.008 |
| lr_feats40_wts_cv | 0.875 | 0.846 | 0.018 |
| lda_feats_cv | 0.873 | 0.836 | 0.035 |
| mlp_full | 0.852 | 0.877 | -0.004 |
| rf_full_wts | 0.864 | 0.835 | 0.019 |
| rf_full_wts_cv | 0.874 | 0.804 | 0.051 |
| dl_full_cv | 0.806 | 0.900 | -0.027 |
| rf_full | 0.860 | 0.835 | 0.018 |
| knn_full | 0.960 | 0.727 | 0.178 |
| knn_full_cv | 0.933 | 0.736 | 0.131 |
| nb_full_cv_platt | 0.853 | 0.791 | 0.040 |
| nb_full_cv | 0.859 | 0.786 | 0.049 |
| nb_l2_feats_cv_platt | 0.866 | 0.750 | 0.093 |
| nb_full | 0.859 | 0.785 | 0.050 |
| dt_full_wts_cv | 0.806 | 0.827 | -0.012 |
| dt_full_wts | 0.791 | 0.781 | 0.011 |
| dt_full | 0.782 | 0.809 | -0.011 |

Table 19: Malawi Poverty Rate Error. Estimated using weighted household predictions.

Indonesia Poverty Rate Error Class Balancing, Tuning, CV, and Feature Selection. Estimated using weighted household predictions.

| | recall | precision | pov_rate_error |
|---|---|---|---|
| lr_full_oversample | 0.840 | 0.436 | 0.106 |
| mlp_full_undersample_cv | 0.902 | 0.389 | 0.150 |
| lr_full_oversample_cv | 0.841 | 0.429 | 0.108 |
| xgb_full_undersample_cv | 0.874 | 0.397 | 0.139 |
| lr_full_undersample | 0.892 | 0.389 | 0.145 |
| lr_full_classwts | 0.867 | 0.394 | 0.137 |
| mlp_full_undersample | 0.887 | 0.392 | 0.142 |
| lda_full_oversample_cv | 0.890 | 0.374 | 0.154 |
| lr_l1_feats_oversample_cv | 0.835 | 0.395 | 0.126 |
| lda_full_oversample | 0.886 | 0.371 | 0.152 |
| lda_full_undersample | 0.908 | 0.351 | 0.176 |
| mlp_full_oversample | 0.600 | 0.542 | 0.008 |
| xgb_full_oversample | 0.601 | 0.476 | -0.028 |
| xgb_full_undersample | 0.892 | 0.352 | 0.177 |
| dl_full_oversample | 0.789 | 0.413 | 0.092 |
| dl_full_undersample_cv | 0.934 | 0.338 | 0.201 |
| dt_full_undersample_cv_ada | 0.882 | 0.347 | 0.169 |
| dl_full_undersample | 0.939 | 0.291 | 0.247 |
| lda_feats_oversample_cv | 0.851 | 0.326 | 0.178 |
| knn_full_undersample_cv | 0.959 | 0.231 | 0.358 |
| dt_full_undersample_cv | 0.848 | 0.258 | 0.255 |
| lda_full_classwts | 0.989 | 0.188 | 0.479 |
| knn_full_undersample | 0.927 | 0.231 | 0.343 |
| dt_full_classwts | 0.726 | 0.319 | 0.139 |
| nb_l2_feats_oversample_cv_platt | 0.749 | 0.264 | 0.213 |
| dt_full_undersample | 0.805 | 0.271 | 0.226 |
| nb_full_oversample_cv_platt | 0.652 | 0.281 | 0.148 |
| nb_full_undersample | 0.669 | 0.308 | 0.132 |
| nb_full_oversample | 0.665 | 0.281 | 0.154 |
| nb_full_oversample_cv | 0.665 | 0.280 | 0.154 |
| knn_full_oversample | 0.984 | 0.163 | 0.536 |
| dt_full_oversample | 0.646 | 0.217 | 0.062 |

Table 20: Indonesia Poverty Rate Error