University of Zurich^UZH

Department of Economics

Prof. Damian Kozbur

Machine Learning for Economic Analysis

WRITE-UP FOR THE REPLICATION PROJECT ON TARGETED TRANSFERS: ANTI-POVERTY PROGRAM IN PERU

BASED ON HANNA & OLKEN (2018)

Group:

Paolo Neri	paolo.neri@uzh.ch	16-729-287
Gabriele Brunini	gabriele.brunini@uzh.ch	20-742-219
Maxime Vandierendounck	maxime.vandierendounck@uzh.ch	16-730-137
Kevin Hardegger	kevin.hardegger@uzh.ch	12-758-785
Josia Schellenberg	josia.schellenberg@uzh.ch	13-608-351

14th December 2020

ABSTRACT

In the course Machine Learning for Economic Analysis, we as a group are assigned to replicate Hanna & Olken's (2018) Figure 5 for Peru, which plots social welfare evaluated against the amount of inclusion error for a fixed budget and allows to infer which amount of inclusion error maximizes overall social welfare. In order to do so, we introduce the reader with a background on (targeted) transfer programs and disclose the data display the trade-offs associated with a proxy-means test and the accuracy assessments we use to evaluate our prediction models. Following that we describe the prediction models we implemented. Because the MSE of our two linear models lasso and ridge are almost identical to the baseline OLS, we compare the accuracy results of our baseline OLS solely to two different random forest models. To this end, we additionally vary the cutoff between the 20th, 25th and 30th percentile in order to assess the sensitivity of our models. We present the results of these accuracy tests, as well as our replication of Hanna & Olken's (2018) Figure 5, where overall social welfare is plotted against the inclusion error. Our main finding is that the baseline OLS model is sufficient as a targeting tool and that more flexible models, such as the random forest, improve the accuracy of predictions only by a small margin.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

BPAC	Balanced Poverty Accuracy Coefficient
CRRA	Constant Relative Risk Aversion
ENAHO	Peruvian National Household Survey
GBC	Governmental Budget Constraint
LE	Leakage (i.e., Inclusion Error)
ML	Machine Learning
MSE	Mean Squared Error
OLS	Ordinary Least Squares
PA	Precision Accuracy
PMT	Proxy-Means Test / Proxy-Means Targeting
SISFO	Sistema de Focalization de Hogares
TA	Total Accuracy
TTS	Targeted Transfer Scheme
UC	Undercoverage (i.e., Exclusion Error)

1. Introduction

As reported by the Worldbank (2020), Peru was one of the fastest-growing countries in Latin America between 2002 and 2013 with an average GDP growth rate of 6.1% per year. Further, it was celebrating a decline in the poverty rate from 52.2% in 2005 to 26.1% in 2013. Although this rapid growth and reduction of poverty is much acclaimed, Hanna & Olken (2018) state that a substantial amount of poverty remains even in rapidly growing economies and that sustained economic growth to the aforementioned extent is the exception and not the norm. There remains an important role for national-level transfer programs that assist the poorest families in a developing country such as Peru (p. 201).

As a matter of principle each government's spending choices i.e., resource transfers programs, is constrained by their budget, called the government budget constraint (GBC). In a nutshell, it is an accounting identity that links monetary and fiscal policy, creating intertemporal links with a large set of possible outcomes (Leeper & Nason, 2005, p. 1f.). Assuming that a transfer program would be financed through taxation (and not by issuing government bonds or receiving international financial aid), there is an inherent trade-off between a government's tax schedule and the transfer program, that depends on how the rest of the tax schedule is adjusted to satisfy the GBC. In the course of this project, we will not be varying or optimizing the tax schedule nor will we consider intensive or extensive labor supply elasticities thereof, but instead we assume a fixed governmental budget of $274 Mio. per year in Peru, following Hanna & Olken (2018, p. 212) and consider only (predicted and actual) pre-tax income, as well as trade-offs in overall social welfare between different targeting schemes.

To this end we first introduce the reader to the challenge of constructing tools for poverty assessment. Subsequently, we illustrate the inherent trade-offs between inclusion and exclusion errors associated with a prediction tool and how we use these true/false positives/negatives to construct accuracy metrics which will be applied in section 4 to assess the accuracy of the models we built in section 3. Closing section 4 we illustrate how the overall social welfare is dependent on the choice of the eligibility threshold.

1.1 Universal Basic Income and Targeted Transfer Schemes

In its most basic form, a universal basic income (UBI) can be conceived as a fixed transfer given to every person or household. UBIs have their merits, such as ease of implementation and low administrative costs (no need to evaluate income or establish and maintain targeting schemes). In practice however, UBIs are rare, due to reasons such as the GBC and the overall aim of a government to target the poorest of the poor.

The main challenge for implementing a transfer program that targets the poorest of the poor, is to correctly target said demographics. This challenge is even higher for developing countries

– such as Peru – because governments in those countries do not readily observe income for those who work in the informal sector and are hence outside of the tax net, which is most common for the poor population. Thus, a preliminary task when attempting to develop a means to target resources towards the poorest of the poor, is to develop a "proxy-means test" or "proxy-means targeting" (PMT) system. This however, will inevitably lead to imperfect targeting in the form of exclusion and inclusion errors. We will discuss these errors in more detail in section 2 and section 4.

A PMT is usually based on large, periodic quasi-census of the population, where government enumerators go door-to-door focusing on households assumed most likely to be poor. These enumerators collect data on the assets of households (i.e., ownership of items, rooms in the house, materials used in house, etc.). The data on these assets is then used to predict incomes, or more precisely the "per-capita consumption of households". In effect, the individual's eligibility for government benefits is based on predicted income since the actual income might be unobservable (Hanna & Olken, 2018, p. 206). The income prediction procedure is described in section 3.

According to Hanna & Olken (2018), the eligibility for governmental programs in Peru is determined by their national PMT called Sistema de Focalization de Hogares (SISFOH). The authors focus on a targeted program called "Juntos", which is "a conditional cash transfer to mothers designed to subsidize child health and education. Beneficiary households receive a monthly transfer of 100 soles (approximately $30)" (p. 208).

1.2 Data

A dataset encompassing household-level data from the Peruvian National Household Survey (ENAHO) during the years 2010-2011 was used. We first split our full sample 50:50, depending on if the outcome variable (*lnpercapitaconsumption*) was blinded or not. The sample split with the blinded outcome variables is to be used only to generate a .csv file with predictions which will be used to evaluate our model by the professor. We randomly split the remaining (unblinded) sample 80:20 into training and validations samples.

Some features must be excluded since they should not be taken into account. The following columns are dropped because either they represent outcome variables or they do not help to predict the income: *training, percapitaconsumption, poor, h_hhsize, id for matlab, hhid, lncaphat_OLS, percapitahat_OLS.* With *lnpercapitaconsumption* as the outcome variable, we have 72 explanatory variables remaining (see section 3). All explanatory variables are binary (dummy variables) and are comprised of the following categories[1]: Fuel, water, drain, wall, roof, floor, education, max. education, insurance, crowd, lux.

[1] A more precise description of each category is presented in Appendix I

For our linear regression models (section 3.1) we drop one dummy-variable per category. We do this in order to offset the dummy variable trap, which would cause perfect multicollinearity. The dropped variables are: *d_crowd_lessthan1, d_lux_0, d_fuel_other, d_water_other, d_wall_other, d_roof_other, d_floor_other, d_h_educ_none, d_insurance_0*. However, for our random forest models (section 3.2) we do not drop these features as tree models aren't prone to this issue, since they treat every variable independently.

2. Trade-Offs and Assessments

As mentioned briefly in the introduction, there is already an underlying trade-off between the UBI (or any targeted transfer schemes (TTS) and the government's (income) tax scheme which will not be investigated in detail. However, there are further important trade-offs to be considered when constructing a PMT and also for the subsequent assessment of how much of the population is optimal to target for a given program (and its corresponding budget). Sub-section 2.1 and sub-section 2.2 describe some important considerations of these trade-offs. Sub-section 2.3 discloses our underlying criteria for assessing our model accuracy.

2.1 Exclusion and inclusion errors in the context of the eligibility threshold

As mentioned in sub-section 1.1, a PMT is prone to errors since the predictive accuracy of such a test is never 100%. These errors are commonly referred to as inclusion and exclusion errors. The former refers to the situation where the transfer program fails to deliver the transfer (i.e., the resources, or cash and cash-equivalents) to the truly poor, because they were misclassified by the PMT. Oppositely, the latter refers to the situation where transfers are delivered to those who are not poor, i.e., who should not have been eligible for the transferred resources (Hanna & Olken, 2018, p. 208). These errors are also referred to as "false positives" and "false negatives" (respectively inclusion and exclusion errors) and are best visualized using a confusion matrix:

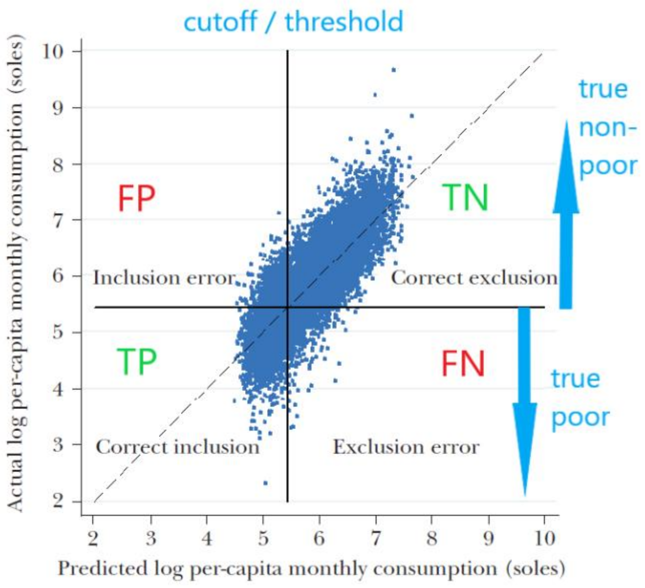


Figure 1: Inclusion vs. Exclusion Errors

Most governments do not vary the level of benefits among the "very poor" group that is eligible, because of the imprecision associated with estimating income based on assets and also because it is more transparent and politically feasible to provide all eligible households below the cutoff with the same amount of benefits (Hanna and Olken, 2018, p. 207). It is however feasible and common amongst certain governments to link benefits levels to predicted income levels by having programs with different eligibility cutoffs. Therefore, depending on the governments' threshold for eligibility, more or less households will benefit from the transfer scheme. This threshold is directly linked to the program being widely or narrowly targeted. There is an inherent trade-off when keeping the total budget for the transfer program fixed. This is most concisely illustrated by receiver operating characteristics (ROC) curves, as shown in Hanna & Olken (2018, p. 212). The higher the cutoff, i.e. the wider the program gets and hence nearer to a UBI, the larger the inclusion error because more and more non-poor households will be receiving a portion of the benefit. At the same time, with a rising threshold the exclusion error and benefits per household get smaller, because the same budget must be equally distributed among more households.

In the setting of this project, we assumed the benefits to be constant, however with 3 different cutoffs (i.e., eligibility thresholds) at the 20th, 25th and 30th percentile. We chose this approach in order to display the sensitivity of correct and incorrect predictions when varying the cutoff, and as well to average out between them for our final assessment of prediction accuracy in section 4.1.

2.2 PMT accuracy assessment

To assess and compare the predictive ability of the PMT models that we describe in further detail in section 3, there are a multitude of accuracy tests or metrics that can be used. The most common metrics are the MSE and the R-squared.

The MSE is the most commonly used measure in the regression setting and is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Where $\hat{f}(x_i)$ is the prediction that the function $\hat{f}$ gives for observation i. The more accurate the predicted values are (i.e., the closer the predictions are to the actual values) the smaller the MSE. Since we want to know whether $\hat{f}(x_0)$ is approx. equal to y_0, where (x_0, y_0) is an out-of-bag, i.e., previously unseen test observation that was not used to train the prediction model, we want to choose the model that has the lowest test MSE (James et. al, 2013, 29f). 1 minus MSE can also be interpreted as total accuracy (TA). However, McBride and Nichols (2016, p. 7f.) display that TA of a model by itself may not be an adequate metric for assessing the accuracy of a poverty tool. For example, if a sample of 1000 households would include 150 poor households, a tool that would (falsely) classify all households as non-poor would have a TA of 85%. This result might appear as accurate, yet it fails to identify even one single poor household means a total failure of the PMT. In light of this, we need more context and hence more accuracy metrics. Thus, we evaluate our models in section 3 by the following sets of accuracy criteria, which can be interpreted as a systematic consideration of each possible outcome and error type in the confusion matrix:

- Total Accuracy (TA) $TA = \frac{TP+TN}{TP+TN+FP+FN}$
- Poverty Accuracy (PA) $PA = \frac{TP}{TP+FN}$
- Undercoverage (UC) $UC = \frac{FN}{TP+FN}$
- Leakage (LE) $LE = \frac{FP}{TP+FN}$
- Balanced Poverty Accuracy Criterion (BPAC) $BPAC = \frac{TP}{TP+FN} - \left|\frac{FN}{TP+FN} - \frac{FP}{TP+FN}\right|$

The BPAC depicts the correctly predicted poor as a percentage of the true poor minus the absolute difference between the UC and LE rates. The BPAC thus combines the three metrics, PA, UC and LE by penalizing the rate which the tool correctly predicts the true poor (PA) with the amount that the LE- and UC-rates exceed one another.

The BPAC will be the main metric we compare in section 4 in order to choose the final model we use for the predictions on the testing sample (which is to be evaluated by the professor).

3. Models

We start off by predicting the dependent variable with a linear regression model that gives an initial guess and baseline to compare more elaborated ML models with. For this, we apply a linear OLS regression in the same manner as Hanna & Olken (2018) for our baseline (results in section 3.1.3). For each subsequent model, we provide an explanation on how these are implemented and why it is thought to increase the accuracy.

3.1 Linear Regressions

3.1.1 Lasso

This project first implements a lasso regression. Lasso regression is a shrinkage method which adds a shrinkage term to the minimization problem of a standard OLS regression. The aim of this ML technique is to prevent overfitting by reducing the model complexity. The advantage here is that the lasso penalty can shrink parameters to zero, thus pruning the model of variables which it considers as nonessential. In other words, the lasso has a variable selection property (James et al., 2013).

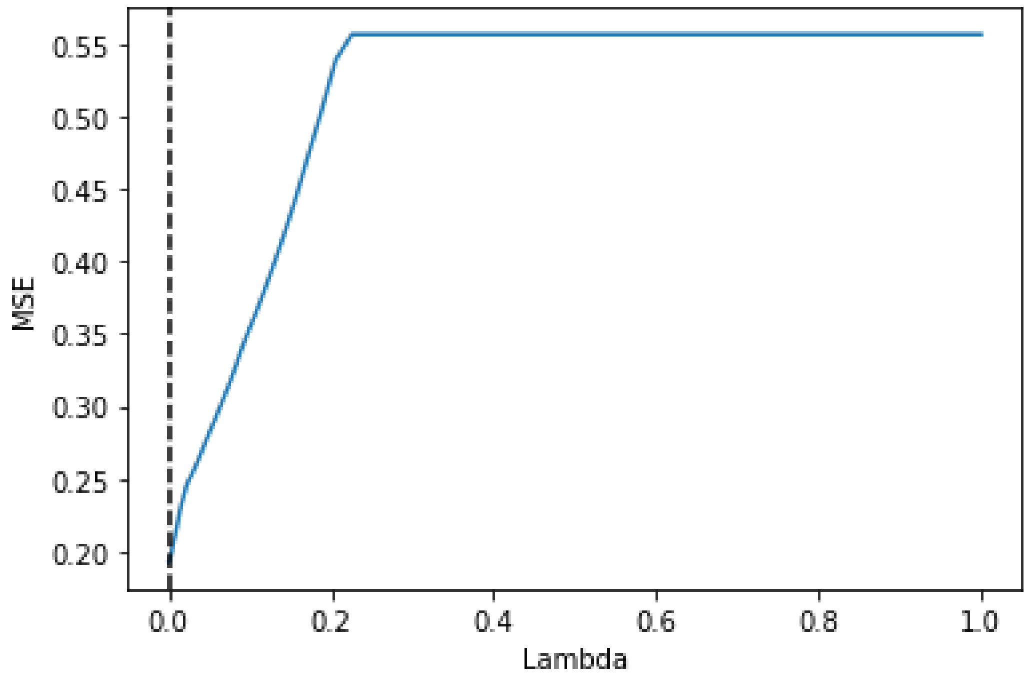
The formula for the lasso estimate looks as follows, where $t \geq 0$ is a tuning parameter:

$$(\hat{\alpha}, \hat{\beta}) = arg\ arg \left\{ \sum_{i=1}^{N} \left(y_i - \alpha - \sum_j \beta_j x_{ij} \right)^2 \right\} \quad subject\ to \quad \sum_j |\beta_j| \leq t$$

It has been shown that subset selection, ridge regression and lasso have advantages and disadvantages, when it comes to different scenarios. Subset selection works best (followed by lasso) when there is a small number of large effects. Lasso does best when there is a moderate number of moderate-sized effects and ridge regression (followed by lasso) does best when there is a large number of small effects (Tibshirani, 1996). The selection of an adequate penalty parameter λ is of major importance for lasso. When λ is equal to zero, then the predictions from lasso are equal to those from OLS. In order to choose a good penalty parameter, it is suggested to perform cross-validation (James et al., 2013).

A 10-fold cross validation procedure was conducted, in order to find the optimal penalty parameter lambda that minimized the mean squared error (MSE), across the training sample. The results suggested that the optimal lambda was really close to zero, as the range for lambda was set to be between 0.0001 and 1 and the optimal lambda ended up to be 0.0001. The evolution of the MSE, given different levels of lambda, is shown in figure 2.

Figure 2: Mean squared error by 10-fold cross validation



Note: The minimum penalty parameter lambda that minimized the MSE was very close to zero. The range for lambda was set to be from 0.0001 to 1. The alpha that minimized the MSE resulted to be 0.0001, given our predefined range. However, the optimal lambda might be even lower, but the procedure to find a lower lambda would have been too computationally intense.

The prediction with lasso resulted in a MSE of 0.190062. Given that the value of lambda was very close to zero, it can be concluded that there is not a significant difference between OLS and Lasso.

3.1.2 Ridge

Because of this, we test another shrinkage method namely the ridge regression. The difference lies in how the penalty term is calculated and thus changes how parameters become smaller. In fact, the penalty shrinks them towards zero, without actually setting them equal to zero. Hence, in contrast to lasso regression, ridge regression includes all regressors in the final model.

The following formula presents the minimization problem for the ridge regression estimation, where $t \geq 0$ is a tuning parameter (James et. al., 2013):

$$(\hat{\alpha}, \hat{\beta}) = arg\ min \left\{ \sum_{i=1}^{N} \left(y_i - \alpha - \sum_j \beta_j x_{ij} \right)^2 \right\} \quad subject\ to \quad \sum_j \beta^2 \leq t$$

Similarly, to the lasso regression, we compute the ridge regression with a 10-fold cross validation. The resulting λ delivering the best performance is 0.01, which is the lowest value of the possible range. Moreover, the model provides an MSE of 0.189974 which is practically the same as the MSE for the OLS model. Thus, the ridge regression doesn't provide any clear improvement compared to the OLS or lasso regression.

3.1.3 Linear Regressions Results

Model	*MSE*
OLS:	0.189975
Ridge Regression:	0.189974
Lasso Regression:	0.190062

Table 1: Linear Regression Results

As we can see from the results depicted above, the ridge and lasso's MSE are practically identical to the MSE of the baseline OLS. Put differently, both ridge and lasso favor a miniscule lambda that approximates towards zero. As a consequence, a lambda of zero essentially converts both models into a standard OLS model.

3.2 Random Forests

3.2.1 Theory

Due to this unfavorable outcome, we implement another technique that follows a different approach than linear regressions; the random forest. Random forests belong to tree-based models. They vary from other models in their method of producing predictions, as they partition the feature space into a set of high dimensional regions following specific decision rules. The set of these splitting rules can be illustrated in an upside-down tree, hence the class name: decision tree.

This paper focuses on regression thus this section covers regression trees and random forest regression, in particular. Regression trees follow a recursive binary splitting algorithm. The goal of the algorithm is to split each given region in two, which in the end concludes in J non-overlapping Regions. The final goal of this algorithm is to reduce the residual sum of squares (RSS) given by:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

In other words, it solves a minimization problem, in which each split minimizes the summed squared distance between the mean response variable and the actual response variable by the greatest extent for each region. First, the algorithm selects the feature X_j and a cut point s which splits the feature space into the regions

$$R_1(j,s) = \{X | X_j < s\} \text{ and } R_2(j,s) = \{X | X_j \geq s\},$$

seeking values for j and s that minimize the equation

$$\sum_{i:\, x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 \sum_{i:\, x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2,$$

where $\hat{y}_{R_1}$is the mean response for observations in $R_1(j,s)$, and $\hat{y}_{R_2}$is the mean response for observations in $R_2(j,s)$ (James et. al., 2013).

Regression trees tend to perform better compared to more classic approaches like linear regression when there is a highly non-linear and complex relationship given between the input and the output variables. However, regression trees possess little robustness, thus little changes in the data can lead to large changes in outcome. In other words, they suffer from high variance. In addition, especially deep regression trees are prone to overfitting. These issues can be treated with random forests (James et. al., 2013).

Random forests differ from their regression forest counterpart by tackling high variance and overfitting by using a modified version of bootstrap aggregation called bagging. Bagging is a method that involves bootstrapping over a number of identically distributed regression trees B and then averaging across them to reduce overall variance of the model. However, as bagging uses the same complete feature space for every split the resulting trees are highly correlated. As a result, the final model carries a persistent variance which can't be mitigated further by increasing B. To solve this problem, random forests improve bagging by using only a random subset of the feature space. Thus, the trees are decorrelated and the persistent variance can be reduced (Hastie et. al., 2009).

The random forest regression algorithm operates as follows (Hastie et. al., 2009):

1. For b = 1 to B:

   a) Draw a bootstrap sample Z^* of size N from the training data.

   b) Grow a random-forest tree $T_b$to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached:

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$ to make a prediction at a new point x:

Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$

3.2.2 Methodology

We implement the random forest for our project with the scikit-learn package in python. Their function for applying random forest regression gives us the possibility to experiment with different hyperparameters. Hyperparameters are parameters that the algorithm doesn't approximate itself. In other words, these have to be tuned manually. However, instead of fine tuning our model manually, we follow an automated approach called grid search.

For this we create a metaphorical grid and insert various values for certain hyperparameters. The total number of possible settings that could be tested, results out of the multiplication of every number of values for each hyperparameter multiplicated by each other. For example, if we test three hyperparameters, which each all contain five values to test, we get a total of $5 \cdot 5 \cdot 5 = 125$ total settings. However, in this paper we experiment a total of five hyperparameters each containing between three to six values. As this leads to a much larger search space, we start with a randomized search.

A randomized search randomly chooses a given number of settings, whereas the specific number of settings to be tested has to be chosen manually. It then tests these combinations with a cross validation approach and presents us the ones that performed best. For illustration, in this project we set the number of iterations to 100 and the cross validation to three. Put differently, the randomized search will randomly test 100 different combinations of the search space and fit each three times totaling in 300 fits. With the help of the randomized search, we can narrow down the optimal hyperparameters. Knowing these, we can then apply the grid search using values near the results the randomized search gave us and testing again for the optimal hyperparameters.

The hyperparameters that we use for our grid search are:

1. The maximum depth of the tree.
2. The maximum features for the subset.
3. The minimum number of samples required to be a leaf node.
4. The minimum number of samples required to split an internal node.
5. The number of trees in the forest.

3.2.3 Random Forest Models

In this paper, we test two different random forest models. The first model *maxDepth* experiments with the maximum depth of trees, while the second model *lessDepth* uses three different values for maximum depth. By doing this, we take overfitting into consideration, as random forest can overfit as well as any other statistical learning model.

The hyperparameters for the randomized search are set as follows, where *p* equals the total number of features:

	maxDepth	*lessDepth*
Max. depth:	max.	50, 60, max.
Max. features:	$p/2, \sqrt{p}, \ ln\ p$	$p/2, \sqrt{p}, \ ln\ p$
Min. samples leaf:	1, 2, 4	1, 2, 4
Min. samples split:	2, 5, 10	2, 5, 10
No. of trees:	1000,1200,1400,1600,1800,2000	1000,1200,1400,1600,1800,2000
Total Settings:	162	486
Settings tested:	61.73%	20.58%

Table 2: Hyperparameters Random Search

As we run the randomized search for 100 iterations totalling in 300 fits, as mentioned before this takes a lot of computation power and time. Unfortunately, this leads to the fact that we only test 20.58% of possible settings for the *lessDepth* model compared to 61.73% for the max*Depth* model. Yet, the computation time for testing an equal rate of settings would increase too drastically.

3.2.4 Results

The table below presents the results for the randomized search:

	maxDepth	*lessDepth*
Max. depth:	max.	max.
Max. features:	$\sqrt{p}$	$\sqrt{p}$
Min. samples leaf:	1	1
Min. samples split:	10	10
No. of trees:	1600	1800
MSE:	0.190138	0.190169

Table 3: Results Random Search

The results present an almost identical outcome for all hyperparameters. Except for the number of trees, the models do not vary from each other. Both prefer the number of features in the subsets to equal the square root of all features. Furthermore, both models indicate that minimum samples used for a leaf node is sufficient with one and minimum samples required for an internal node split should be 10. Considering these minor differences, their MSE differs only by a very small margin of 0.000031.

These presented results now help us narrow down the optimal hyperparameters, as we now employ a small range of values near the proposed values for our grid search. As our goal is to create an alternative tree with less depth, we include new depth ranges near the maximum value of 72. However, we fix the value of 1 for the minimum samples used for splitting a leaf node.

As we have reduced the search space, in contrast to the randomized search, we are now able to test all possible settings for both models.

The hyperparameters for the grid search are set as follows:

	maxDepth	*lessDepth*
Max. depth:	max.	68, 70, 72
Max. features:	$\sqrt{p}$	$\sqrt{p}$
Min. samples leaf:	1	1
Min. samples split:	9, 10, 11	9, 10, 11
No. of trees:	1500, 1600, 1700	1700, 1800, 1900
Total Settings:	9	27
Settings tested:	100%	100%

Table 4: Hyperparameters Grid Search

The results of the grid search are presented below:

	maxDepth	*lessDepth*
Max. depth:	max.	68
Max. features:	$\sqrt{p}$	$\sqrt{p}$
Min. samples leaf:	1	1
Min. samples split:	11	11
No. of trees:	1700	1900
MSE:	0.189691	0.189662

Table 5: Results Grid Search

We can see that both models change their number of trees to 1700 and 1900 respectively. Furthermore, both models now use a minimum number of eleven samples for an internal node split. However, the *lessDepth* model chooses a maximum depth of 68. This change leads to an overall larger improvement to the random search model compared to the *maxdepth* model. In fact, the MSE for lessDepth with a value of 0.189662 is now smaller than the MSE of *maxDepth* with a value of 0.189691. Having said that, the difference is still very small.

4. Results

Section 4 presents the results of the developed models. These are evaluated using confusion matrices, MSE and BPAC scores. Because both our Ridge and lasso models produce almost identical results as the baseline OLS, we will only consider the OLS in comparison to the Random Forest model.

4.1 Confusion matrices and accuracy results

The following table discloses our accuracy metrics for the OLS and both random forest models, for different cutoffs (i.e., eligibility thresholds) at the 20th, 25th and 30th percentile.

	TA	PA	LE	UC	BPAC
Cutoff 20%					
OLS20	0.861521	0.653804	0.346196	0.346196	0.653804
Lessdepth20	0.862845	0.657111	0.342889	0.342889	0.657111
Maxdepth20	0.863726	0.659316	0.340684	0.340684	0.659316
Cutoff 25%					
OLS25	0.861521	0.723104	0.276896	0.276896	0.723104
lessDepth25	0.861521	0.723104	0.276896	0.276896	0.723104
maxDepth25	0.860639	0.721340	0.278659	0.278659	0.721340
Cutoff 30%					
OLS30	0.857552	0.762675	0.237325	0.237325	0.762675
lessDepth30	0.856229	0.760470	0.239530	0.239530	0.760470
maxDepth30	0.856229	0.760470	0.239530	0.239530	0.760470
Avg. BPAC OLS		0.713194	MSE OLS		0.189975
Avg. BPAC lessDepth		0.713562	MSE lessDepth		0.189662
Avg. BPAC maxDepth		0.713709	MSE maxDepth		0.189691

Table 6: Accuracy Tests for Cutoffs at 20, 25 and 30th percentile.

The table displays that both random forests possess smaller MSE than the OLS model. Yet, the differences are minor. Further, our accuracy tests stemming from the confusion matrix suggest that on average our random forest maxDepth has the highest accuracy, albeit with a miniscule and insignificant difference to the OLS and second random forest lessDepth. The results also confirm the intuition concerning the eligibility threshold described in section 2.1, that the higher the cutoff the higher the PA. Interestingly however, the LE- and UC-rates seem to be getting smaller with a higher threshold, implying that the more widely targeted the TTS, the less prone to error our PMT is. We see furthermore, that both for OLS and for random forests, independent of the cutoff, the LE- and UC-rates are the same. Hence, the BPAC is the same as the PA. This implies that our PMT always equally splits the sample in a manner such that exactly the same amount of households are falsely classified as poor (inclusion error) and non-poor (exclusion error).

4.2 Overall social welfare versus inclusion error

As mentioned in the introduction, our main goal is to derive the optimal threshold for the TTS, i.e., the cutoff which maximizes overall social welfare parsing the trade-offs between inclusion error, exclusion error and per-capita benefits. Given a fixed governmental budget and the premise that each household characterized as poor receives the same amount of benefits b, per-capita benefits are de facto dependent on how many households are eligible for benefits, i.e., where the cutoff is set.

The optimal cutoff point, however, is dependent on how the weights are assigned to the inclusion- and exclusion errors i.e., of how costly it is for the government to deliver benefits to households that are not poor and fail to deliver the benefits to households that are poor respectively. Furthermore, the weight of the implied reduction in per-capita benefits to the poor when raising the cutoff-threshold must be assessed as well as the difference in marginal utilities for poor and rich. Because we want to enable a pure comparison between our replicated figure 5 with that of Hanna & Olken (2018, p. 213f.), we do not assess and apply the aforementioned weights independently, but rather apply the same weights as Hanna & Olken (2018b). Along those lines we evaluate the total social welfare with the following constant relative risk aversion (CRRA) function:

$$U = \frac{\sum \ (y_i + b_i)^{1-\rho}}{1-\rho}$$

Where y_i is household i's pre-tax income, b_i is the per-capita benefit for household i and ρ a coefficient of risk-aversion that is greater the higher a weight we assign to transfers received by the very poor.

We plot the social welfare evaluated against the amount of inclusion error for the fixed budget of $274 Mio. per year and obtain the following:

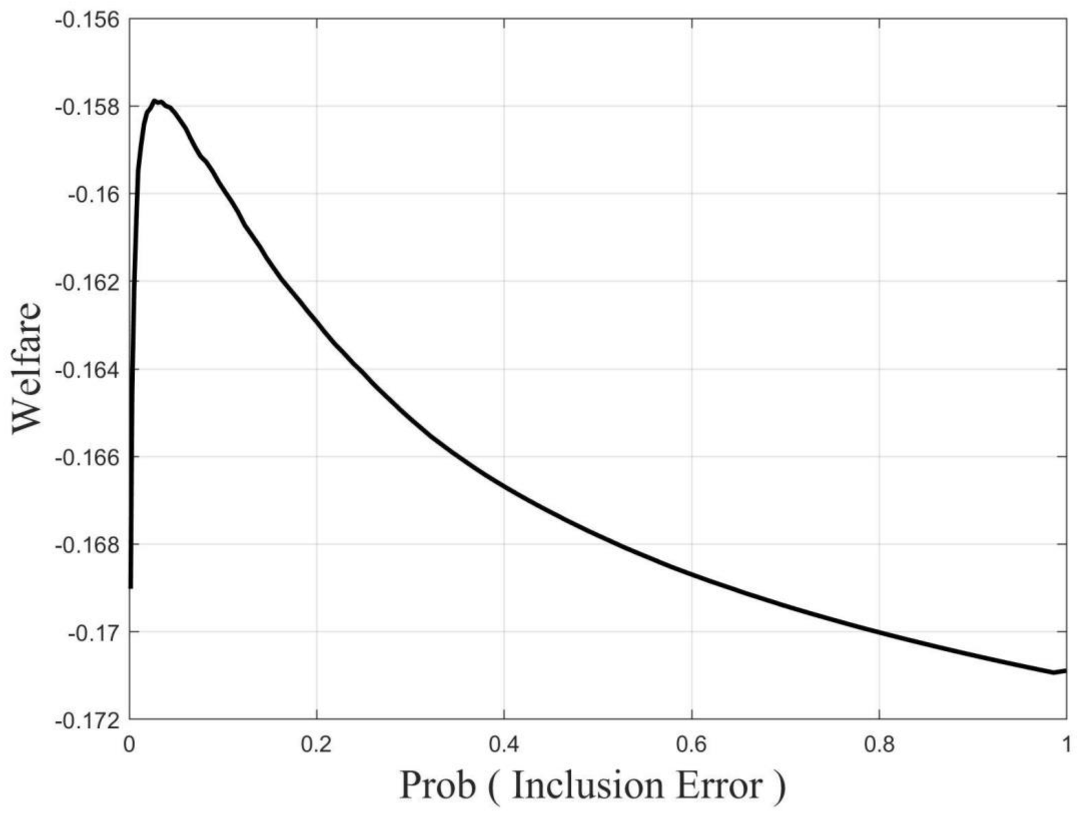


Figure 3: Overall social welfare vs. inclusion error

We find that the socially optimal program targets around 10% of the whole population of Peru and induces an inclusion error of 2.67%. This conclusion is dependent on the weights assigned, the accuracy of our prediction model and the number of households assumed (6.75 Mio.) for the whole population. If we were to increase the number of households for Peru, then we would witness a drop in the optimal inclusion error.

Comparing with Hanna & Olken (2018, p. 214f.) who obtained a socially optimal target of 18% of the population with an inclusion error of 6.4%, our PMT targets a lower percentage of the population and thus tolerates less inclusion error, implying a narrower TTS.

5. Discussion and Conclusion

In the course of this project, we have outlined the challenge of constructing and applying PMTs for a developing country. We used Hanna & Olken (2018) as a guideline and blueprint, essentially replicating their results with our OLS model, whereby we also expanded upon their approach by constructing more flexible ML models. We evaluated the accuracy of these models with a broader array of metrics such as the BPAC and found little to no difference between the more flexible, less interpretable ML-models, and the standard OLS. This might indicate that a strong linear relationship between the features and the response variable exists. Nevertheless, due to small improvements in prediction, we proceeded with our random forest maxDepth model and generated predictions for the blinded sample of the dataset and used these predictions to replicate Hanna & Olken's (2018) "figure 5". As seen in figure 3, we plot overall welfare as a function of inclusion error dependent on the varying the eligibility threshold. We find that our predictions imply a narrower targeting of the transfer scheme at the optimal social welfare than did Hanna & Olken (2018). We attribute these differences to the different model of choice, the smaller training sample, the plotting on a smaller sample set as well as the random splitting into training and validation samples that remained after dropping all observations with NaN entries.

Although our model predictions lack a distinct improvement upon the original standard OLS model, we are not disappointed since we were able to apply our knowledge from this course on a real-world example. Moreover, we were able to dive into the contemporary political debate of UBIs and targeted transfers that may very well become a future standard tool for most governments, and thereby gained insights on the do's and don'ts when trying to build a cutting-edge PMT.

For further analysis we suggest implementing a larger range of thresholds as well as using further ML techniques, such as neural nets and splines which have not been investigated in this work.

BIBLIOGRAPHY

Hanna, R. & Olken, B. A. (2018). Universal Basic Income versus Targeted Transfers: Anti-Poverty Programs in Developing Countries. *Journal of Economic Perspectives, 32*(4), 201-226.

Hanna, R., & Olken, B. A. (2018b). Replication data for: Universal Basic Incomes versus Targeted Transfers: Anti-Poverty Programs in Developing Countries: 181009 Programs to publish: Coding do files: 181009 Peru Coding.do. Nashville, TN: American Economic Association [publisher]. Ann Arbor, MI: Inter-university Consortium for Political and Social Research [distributor], 2019-10-12. https://doi.org/10.3886/E114021V1-72082

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). New York: springer. https://doi.org/DOI 10.1007/978-1-4614-7138-7

Kozbur, D. (2020). Analysis of testing-based forward model selection. Econometrica, 88(5), 2147-2173. https://doi.org/10.3982/ECTA16273

Leeper, E. M. & Nason, J. M. (2005). *Government Budget Constraint*. 10.1057/978-1-349-95121-5_2041-1

McBride, N. & Nichols, A. (2016). Retooling Poverty Targeting Using Out-of-Sample Validation and Machine Learning. The World Bank Economic Review, 32(3), 531-550. https://doi.org/10.1093/wber/lhw056

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288. Retrieved from https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.x

Worldbank. (2020, October 9). *The World Bank in Peru*. Retrieved from https://www.worldbank.org/en/country/peru/overview

APPENDIX I

Category	Description	Features
Fuel	Type of fuel used for cooking	other, wood, coal, kerosene, gas, electric, none
Water	Water source	other, river, well, truck , pylon, outside, inside
Drain	Drainage source	none, river, cesspool, septic, outside, inside
Wall	Wall type	other, woodmat, stonemud, quincha, tapia, adobe, stonecement, brick cement
Roof	Roof type	other, straw, mat, tile, wood, concrete, other, earth, cement, wood
Floor	Floor type	sheets, parquet
Education	Education of head of household	pre, prim, sec, higher no-uni, higher uni, post
Max. Education	Highest education of anyone in household	None, prim, sec, higher no-uni, higher uni
Insurance	Number of household members affiliated to health insurance	0, 1, 2, 3, 4+
Crowd	Number of household member per room	< 1, 1-2, 2-4, 4-6, 6+
Lux	Luxury goods owned (internet connection, cable connection, computer, refrigerator and washing machine)	0, 1, 2, 3, 4, 5

Table 7: Description of the Features