COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

POVERTY ANALYSIS USING MACHINE LEARNING METHODS

BACHELOR THESIS

2016
NATAŠA PLULIKOVÁ

Comenius University in Bratislava

Faculty of mathematics, physics and informatics

Poverty analysis using machine learning methods

Bachelor thesis

Study program:	Mathematics
Field of study:	1113 - Mathematics
Department:	Department of mathematics
Supervisor:	Doc. RNDr. Beáta Stehlíková, PhD.

Bratislava, 2016

Nataša Pluliková

19323727

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Nataša Plulíková
Study programme:	Mathematics (Single degree study, bachelor I. deg., full time form)
Field of Study:	Mathematics
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title:	Poverty analysis using machine learning methods
Aim:	1. Application and comparison of poverty prediction models in Indonesia (regression models, neural networks, trees and random forests) 2. Cluster analysis

Supervisor:	RNDr. Beáta Stehlíková, PhD.	
Department:	FMFI.KAMŠ - Department of Applied Mathematics and Statistics	
Head of department:	prof. RNDr. Daniel Ševčovič, CSc.	
Assigned:	26.10.2015	
Approved:	26.10.2015	prof. RNDr. Ján Filo, CSc. Guarantor of Study Programme

....

Student Supervisor

19323727

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Nataša Plulíková
Študijný program:	matematika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	matematika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Poverty analysis using machine learning methods
Analýza chudoby metódami strojového učenia

Cieľ: 1. Aplikácia a porovnanie modelov predpovedania chudoby v Indonézii (regresné modely, neurónové siete, stromy a náhodné lesy)
2. Zhluková analýza

Vedúci:	RNDr. Beáta Stehlíková, PhD.
Katedra:	FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry:	prof. RNDr. Daniel Ševčovič, CSc.
Dátum zadania:	26.10.2015

Dátum schválenia: 26.10.2015 prof. RNDr. Ján Filo, CSc.
garant študijného programu

....
študent vedúci práce

Acknowledgements: Completing this thesis would have been impossible without support and help from various people. First of all, I am very grateful to my supervisor Doc. RNDr. Beata Stehlikova, PhD. for her patience and encouraging guidance. Secondly, I am thankful to Nadacia SPP which has granted me an educational scholarship. These extra funds enabled me to attend courses to learn knowledge that was crucial for completing this work. Last, I am very thankful to my friends and classmates who were always supporting me and willing to share their ideas with me.

Abstract

Poverty represents a major theme in development practice. Most of the researchers try to understand, measure and predict this phenomenon. The main goal of this paper is to suggest models that effectively predict poverty levels. In order to accomplish this task we also present a way how to measure poverty through a poverty index that we created. The data that we use come from Indonesia and therefore the outcomes mirror condition in this country.

We view poverty as a multidimensional concept. In other words, we do not consider income as a sole predictor of poverty, but incorporate information such as education, health and household condition that all influence whether a person is poor or not. We apply machine learning methods. Our first set of models – multiple correspondence analysis and K-Means - clusters respondents based on the poverty levels. The second set of models – Binary logistic regression, Neural networks, Decision Trees and Random Forests – predicts poverty levels.

When comparing the different predictive models we conclude that Neural networks perform best in terms of accuracy. However the trade-off for that is limited interpretability. It is almost impossible to state the effect of an individual predictor on poverty level. Binary logistic regression and Decision trees can be interpreted quite well but they have lower predictive power.

Key words: machine learning, poverty prediction, poverty measurement, neural networks, decision trees, binary logistic regression, random forests, multiple correspondence analysis, k-means

Abstrakt

Chudoba reprezentuje hlavnú tému v rozvojovej práci. Vedci sa snažia porozumiet', zmerat' a predpovedat' tento fenomén. Hlavným ciel'om tejto práce je navrhnút' modely, ktoré budú efektívne predpovedat' chudobu. Na uskutočnenie tejto úlohy taktiež prezentujeme spôsob ako zmerat' chudobu cez index chudoby, ktorý sme sami vyvinuli. Dáta, ktoré sme použili na testovanie, pochádzajú z Indonézie a preto odzrkadl'ujú podmienky v tejto krajine.

Chudobu vnímame ako viacdimenziálny koncept. Inými slovami, nepovažujeme príjem za jediný indikátor chudoby, ale snažíme sa zahrnút' aj informácie ako je vzdelanie, zdravie a podmienky v domácnosti, ktoré spoločne ovplyvňujú, či je jednotlivec chudobný alebo nie. Aplikujeme metódy strojového učenia. Náš prvý súbor modelov – viacrozmerná korešpondenčná analýza a algoritmus k-priemerov – zhlukuje respondentov podl'a chudoby. Náš druhý súbor modelov – binárna logistická regresia, neurálne siete, rozhodovacie stromy a náhodné lesy – predpovedajú hodnoty chudoby.

Pri porovnaní rôznych metód konštatujeme, že neurálne siete sú najlepšie, čo sa týka presnosti predpovede. Avšak, cenou za túto presnost' je obmedzená schopnost' interpretácie. Je takmer nemožné určit' efekt individuálneho prediktora na hodnotu indexu chudoby. Binárna logistická regresia a rozhodovacie stormy sa dajú dobre interpretovat', avšak majú nízku predikčnú schopnost'.

Kl'účové slová: strojové učenie, predpoved' chudoby, meranie chudoby, neurálne siete, rozhodovacie stormy, binárna logistická regresia, náhodné lesy, viacrozmerná korešpondenčná analýza, algoritmus k-priemerov.

Contents

List of Tables

List of Figures

Introduction

Poverty is a multidimensional problem that has a variety of definitions. For some authors it is measured by income, while other researchers include also health, education, social status and political rights into the picture. However, what connects all researchers is their work in the field of identification of factors that cause poverty, classification of population according to different views of poverty and prediction of future poverty levels (Asselin & Ahn. 2008, Mathiassen. 2008, Vella. 1997)

Our original motivation for this thesis was to use different statistical and machine learning methods, to address the last problem stated above, and to create a poverty prediction model. However, after reviewing the literature on different methods and understanding the nature of our data we decided to slightly change our approach. Prediction still remains in the center of our attention but instead of one model we decided to choose four distinct models and compare their performance, advantages and disadvantages. In this way we answer the following research questions:

1. What advantages and disadvantages are there for different prediction methods?
2. Which models perform best in terms of predictive ability and what is the trade-off for this precision?

The data for this thesis comes from Indonesia. RAND Corporation in collaboration with The Demographic Institute of The University of Indonesia conducted a longitudinal study under the name Indonesian Family Life Surveys. These surveys are unique as they track individuals and households in four consecutive waves and capture their livelihoods for fifteen years. Despite the richness of these datasets we encountered a major limitation which impacted our outcome heavily. The data contains very detailed information about household conditions but has major gaps in the area of household economy. Due to the large number of missing values for numerical variables such as income or expense we had to omit them from our analysis. We believe that this is a reason for the lower predictive power of our models since income is an important, although not exclusive, poverty predictor.

The preliminary step to our analysis was to determine the poverty levels in Indonesia. We have decided to use a multidimensional concept of poverty and for that purpose we needed to find a suitable method that would cluster respondents based on their household condition, levels of education, health and other characteristics that we associate with poverty. The nature of our data which was purely categorical imposed another limitation as we had to seek methods designed for that type of data.

In the end, the thesis combined two main aspects. First, we incorporated the multidimensional concept of poverty and used methods that classified respondents accordingly. Second, we chose four commonly used methods for prediction, created four distinct poverty prediction models and compared how well each of them performed.

The thesis is organized in various sections. In the first section we provide a brief theoretical background on our view of poverty. The second section describes our data and the limitations applicable to this thesis. The third section gives a brief introduction to the methodology. It describes the nature of the used methods, gives a theoretical background of certain methods' specifics and introduces our way of measuring accuracy. The fourth section dives into each particular method. It first provides the theoretical background which is followed by an explanation of the results that were obtained when the respective algorithm was run on our data. All these sections work with the data from Wave 1, i.e. from 1993. Our last section describes the results that were obtained when the created models were run on the Wave 2 data from 1997.

1 Understanding the concept of poverty

For the purposes of this study we view the concept of poverty through the lens of the Sustainable Livelihood Approach. Such types of framework first emerged in the context of rural development but have been widely adapted also to urban conditions (Scoones 2009: 172). The most commonly used definition comes from Chambers and Conway (1992:6):

A livelihood comprises the capabilities, assets (...) and activities for a means of living. A livelihood is sustainable when it can cope with and recover from stresses and shocks, maintain or enhance its capabilities and assets, and provide sustainable livelihood opportunities for the next generation; and which contributes net benefits to other livelihoods at the local and global levels and in the short and the long term.

Basically, the perspective takes into consideration various dimensions of livelihood. These dimensions constitute five different forms of capital that each individual possesses and which contribute to their wellbeing. These include:

- Human capital - education, skills, health
- Financial capital - income, remittances, savings, credits and debts
- Social capital - social network, trust, relations
- Physical capital - infrastructure, technology
- Natural capital - natural resources such as wood or land

The ownership of these assets is shaped by the outside conditions over which an individual has limited power. These include the political, economic and environmental situation as well as the cultural and social context.

Aligned with this framework is also our approach to poverty. However, since we rely on existing datasets, we could identify variables reflecting only three of these dimensions - human assets (education and health), financial assets (savings) and physical assets (household assets, physical conditions of the household).

2 Data and Limitations

The data that was used and analyzed comes from the Indonesian Family Life Surveys (ILFS). These represent a longitudinal sociological study that was conducted in Indonesia in four waves in the years 1993/94, 1997/98, 2000 and 2007/2008. All surveys were conducted by a research organization, the RAND Corporation, in collaboration with The Demographic Institute of The University of Indonesia. Based on their selection criteria, the survey is 83% representative of the population and contains information about the household and its members; household conditions and economy; individual education and health conditions; and community data (RAND Labor and Population).

Our analysis is twofold. First, we use the rich dataset to determine which individuals are poor. Then we use the other set of variables to create a model for poverty prediction. We were inspired by various articles dedicated to this problem (Mathiassen. 2008, Vella. 1997). However we see a major limitation to their approach, namely in how they described the poverty level. They took the household income and based on some threshold value (typically the World Bank definition of poverty) determined which households are poor. According to our view of poverty, income is not a sufficient poverty descriptor. Therefore we decided to extract all the information from the available data and use models that identify patterns in the current dataset.

As mentioned above, we used two almost distinct datasets. Our first dataset served to create an indicator of poverty and we had to use other data for predictions. Otherwise we would be predicting poverty with the same variables that were used to describe poverty which would bring conceptual problems. In reality, the income data is often not available as it represents sensitive information which respondents frequently refuse to give. Our dataset is a great example of this problem as for income variables we had about 60% missing values. Therefore the added value of our approach is that we rely on data that is easily collected. Some of the variables, like those about the household conditions, are easily observable by the interviewer and so represent a valuable and easily obtained information.

One important advantage of our dataset is its longitudinal character. We have information about the same household across fifteen years. This introduces 'time' to the prediction problem which might provide a valuable insight for tracking and predicting poverty across multiple years. Although the scope of this work does not fully acquire this advantage, we propose ways how this research could be expanded.

We provide the overview of the variables that were used. Table 1 describes the descriptors of poverty (a total of 6388 households included) while table 2 describes poverty predictors (a total of 4493 households included).

Table 1: Poverty descriptors

Dimension	Variable name	Values	Frequency
Human capital	Education	grade school	3205
		higher education	1975
		unschooled	1208
	Health	healthy	5571
		unhealthy	831
	Insurance	yes	791
		no	5597
Physical capital	Surface water	yes	978
		no	5410
	Tap water	yes	1007
		no	5381
	Creek toilet	yes	1630
		no	4758
	Own toilet	yes	1993
		no	4395
	Running sewage	yes	2866
		no	3522
	Garbage collector	yes	1435
		no	4953
	Vehicles	yes	2015
		no	4373
Financial assets	Savings	yes	1477
		no	4911

2.1 Limitations

There are multiple limitations to this work. First is the scope of this thesis. The topic we chose is very wide and far from exhausted. During our work we had to identify results that are feasible to obtain within the limited time frame that was dedicated to the creation of this work. We especially believe that conceptual broadening of the topic could introduce interesting insights which might be translated into new inputs integrated within our models.

The time frame also influenced the size of this work. Since our aim was a comparison of multiple models, we could not delve into the details of each algorithm. Still, for an interested reader, all applicable theorems and proofs can be found in the respective literature.

Thirdly, we were limited by the nature of our data. All of our predictors were categorical variables. Although they contain valuable information, the predictive strength of our algo-

Table 2: Poverty predictors

Dimension	Variable name	Levels	Frequency
Human capital	Education - elementary	1-yes	2300
		0-no	2193
	Education - higher	1-yes	1208
		0-no	3285
	Healthy	1-yes	3693
		0-no	800
	Can write a letter in Bahasa	1-yes	2970
		0-no	1523
	Can read a newspaper in Bahasa	1-yes	3043
		0-no	1450
Physical capital	Has electricity	1-yes	3178
		0-no	1315
	Has household appliances	1-yes	3215
		0-no	1278
	Can buy meat	1-yes	898
		0-no	3595
Financial assets	Savings	1- yes	1080
		0-no	3413
	Employed	1-yes	3794
		0-no	699
	Unemployed	1-yes	411
		0-no	4082
Household shocks	Experienced hardship	1-yes	1323
		0-no	3170

rithm could be improved greatly if the income and expense variables were used. Our dataset contained almost 60% missing values for income and expense variables and we could not extract any valuable information from these. Since we worked with panel data, we also had to identify those variables that were common across the years which again limited the number of predictors that we could use.

3 Methodology Overview

We decided to approach our research problem through machine learning methodology. Machine learning provides effective solutions for understanding data patterns and predictions. The main goal of prediction is to accurately determine the outcome based on the causality relationships in the data. (Shalew-Schwartz & Ben-David, 2014) In our case, we wanted to use the household data from Indonesia to determine and predict poverty levels. There are two main groups of machine learning algorithms that are frequently used to provide insights on similar problems:

- unsupervised learning algorithms
- supervised learning algorithms

The **unsupervised learning algorithms** do not presume any structure of the data and their added value is to help understand patterns in the data. We used these types of algorithms to understand and explain the concept of poverty and to consequently determine who are poor households in our dataset.

The **supervised learning algorithms**, as their name suggests, already presume certain pattern in the data. Typically we split the variables to predictors and outcomes. To teach the algorithm we first need to feed it both types of variables. Throughout the learning process the algorithms tune their parameters. With known parameters the algorithms can be used for predictions of the future data. In our case, we used the obtained levels of poverty to teach the algorithm when the household is poor and test how well it predicts the poverty in the future.

3.1 Unsupervised learning

The unsupervised machine learning uses all available data to understand the patterns and relationships in the data (Ng. 2015). A typical example is clustering the data into objects that have some common features or compressing the data set into a lower dimension while still keeping as much information as possible.

The first part of our analysis attempts to identify whether the individual is poor or not, which is a suitable task for unsupervised learning method. We tried various clustering models but our data contained too many missing values for numeric variables such as income or expense. Therefore we decided to use a method which works well with categorical variables - Multiple correspondence analysis (MCA). The output from MCA includes weights or loadings for each level of each categorical variable. We used these weights to create a numerical index "poverty".

Consequently we applied a K-means algorithm to determine which of the households were poor and non-poor. A K-means algorithm is a widely used method for clustering numerical data. It uses Euclidean distance to calculate how close individuals are to each other and clusters them based on their proximity. The disadvantage of this algorithm is that the user needs to identify the number of clusters, here denoted by K. There is not one way how to determine the number of clusters. The decision can be based on the theory (i.e. number of social classes) or experience and could vary according to the research problem (Ng. 2015). Our decision for the number of clusters was based on the algorithm itself. We chose to examine the *scree plot* according to which we chose four clusters. The *scree plot* shows the number of clusters on the x-axis and the within groups sum of squares on the y-axis. Logically, with the increasing number of clusters, the within groups sum of squares decreases as we split the dataset into smaller groups. The aim is to have few clusters with small within groups sum of squares which would correspond to well defined distinct groups in our dataset (Ng. 2015). We will describe in more detail how scree plot helped us in our decision in the section dedicated to the K-means algorithm.

Our last step was to categorize the group with the lowest values of our poverty index as "poor".

3.2 Supervised learning

Supervised learning algorithms are suitable for predictions. Generally, there are multiple choices for prediction models widely used by statisticians and data scientists (Breiman 2001). In this thesis we chose four methods and we discuss their strengths and weaknesses. The chosen models were Binary logistic regression, Neural networks, Decision trees and Random forests.

The idea of the supervised learning algorithms is that we provide it with a learning environment. The learning environment consists of the cases where both predictors and outcomes are available to the algorithm, called the training dataset. Its parameters are then tuned to correctly predict this training data. We measure the correctness by comparing the predicted values, i.e. calculated outcomes and the real values, i.e. those outcomes that we identified in our data.

The second step of the supervised machine learning is to determine whether the obtained model performs well also on the “unseen” data. We therefore feed it with a dataset of predictors, called the test dataset. For this dataset we know the true levels of our outcome variable. The algorithm gives us predicted outcome levels which we compare with the true outcomes. When we reach high levels of accuracy we can claim that our model predicts the concept well. If we obtain poor accuracy levels, we might have overtrained our model and need to go back and update the algorithm accordingly. Thus, the methodology for supervised learning requires first to split our dataset into two parts: training and test which we have done in a 60:40 ratio (Shalev-Schwartz & Ben-David. 2014).

Typically, the learning task involves an iterative method that either minimizes or maximizes a function that represents our problem. We continue with explaining a method that is widely used for the minimization problems.

3.2.1 Gradient Descent

This section describes the basic idea behind the method called Gradient Descent. It is an iterative method that is widely used in machine learning algorithms. The theoretical perspectives in this chapter are based on Shalev-Schwartz & Ben-David (2014) and Ng (2015).

Gradient Descent (GD) is a widely used iterative technique for minimizing risk (also called cost) functions. The algorithm is based on the fact that the direction of the gradient of a particular function points to the direction where the function grows most rapidly. Therefore at each iteration the algorithm takes a small step in the negative direction of the gradient which ensures the convergence at the local minimum. We will now describe the mathematical notation for this algorithm. To illustrate the idea we limit ourselves to only convex functions however the literature explains in detail how the concept can be extended to any function.

Let f be a convex function and let w denote the vector representing the parameters which we want to learn. The gradient of function f is a function $\triangle f : R^d \rightarrow R$ such that

$$\triangle f(w) = (\frac{\partial f(w)}{\partial w_1}, \frac{\partial f(w)}{\partial w_2}, \cdots, \frac{\partial f(w)}{\partial w_d})$$

Since gradient descent is an iterative algorithm, we first define a starting point, for example $w^{(1)} = 0$. Each following iteration is taking a small step in the negative direction of the gradient:

$$w^{(t+1)} = w^{(t)} - \eta \triangle f(w^{(t)})$$

After T iterations we stop the algorithm based on some stopping criteria and we obtain the result which is close to the local minimum. In the equation above, $\eta > 0$ is a positive parameter which influences the speed of convergence. The proof of convergence is not in the scope of this thesis but can be found in Shalev-Schwartz & Ben-David (2014). The simplified algorithm is shown below (Ng. 2015):

```
1. parameters: Scalar $\eta > 0$, integer $T > 0$
2. initialize: $w^{(1)} = 0$
3. for $t = 1, 2, \cdots, T$:
    • calculate $w^{(t+1)} = w^{(t)} - \eta \triangle f(w^{(t)})$
    • if $\left\|w^{t+1} - w^t\right\| < \varepsilon$ return "Converged"
    • if $f(w^{t+1}) > f(w^t)$ return "Diverged"
4. return $f(w^{t+1})$
```

3.2.2 Overfitting and regularization

The frequent problem that occurs with learning algorithms is overfitting. Overfitting happens when the parameters which we obtained as the best fit perform greatly on the training set but poorly on the testing set. That means that our parameters only reflect the unique situation of the training set and cannot be used for further generalizations.

One way to overcome this problem is through regularization. Regularization introduces greater stability to the algorithm. We loosley define the term stability as 'slightly changing the input causes only minor changes in output' (Shalev-Schwartz & Ben-David.2014).

The most frequently used regularization function is Tikhonov regularization defined in the following way (Shalev-Schwartz & Ben-David.2014):

$$R(w) = \lambda \, ||w||^2$$

After we have trained the model we need to evaluate its accuracy. The next section is dedicated to this topic.

3.3 Determining accuracy

There are many ways how to determine the accuracy of the predictive model. We chose to compare the calculated outcomes and the true outcomes in three different ways explained below. The theory that was used for this section is based on Stojanovic et.al (2014)

Our dependent variable is coded as 0 for the non-poor and 1 for the poor. When we compare the predicted values obtained from our models and the true values of poverty we can get four possible scenarios (Table 3):

Table 3: Crosstabulation of true and calculated outcomes

	True Poor(1)	True Non-poor(0)
Predicted Poor(1)	True positives	False positives
Predicted Non-poor(0)	False negatives	True negatives

1. True positives (TP) - these are all the cases where our model predicted the household is poor and the household indeed was poor.

2. True negatives (TN) - these are all the cases where our model categorized the household as non-poor and indeed it was non-poor.

3. False positives (FP) - these are all the cases where our model categorized the household as poor but it was not.

4. False negatives (FN) - these are all the cases where our model classified household as non-poor but indeed it was poor.

When it comes to accuracy, there are three ways how we can look at this table:

1. **Error rate.** Error rate is simply a ratio of the cases which were identified wrongly. For the purposes of this thesis and to better compare the methods, we calculate its opposite, i.e. ratio of all those cases that were classified correctly. In mathematical terms:

   $$Errorrate = \frac{1}{n}\sum(y_i \neq y_i') \; Accuracy = \frac{1}{n}\sum(y_i = y_i')$$

   Where y_i is the observed value and y_i' is the predicted value of our poverty indicator.

2. **Sensitivity.** Sensitivity is the probability that we will classify the poor among those that are truly poor. In mathematical terms, sensitivity can be calculated as follows:

   $$Sensitivity = \frac{TP}{(TP+FN)}$$

3. **Specificity.** Specificity is the fraction of how many of the non-poor were classified as non-poor. In mathematical terms, specificity can be calculated as follows:

   $$Specificity = \frac{TN}{(TN+FP)}$$

4 Algorithms and results

This part of the thesis is divided into several sections. The first part of each section explains the theory on which the algorithms are based. The second part of each section describes the results obtained when the algorithms were run on our dataset. The whole chapter is chronologically ordered and illustrates the process of how we obtained our results.

4.1 Multiple correspondence analysis

Multiple correspondence analysis (MCA) is a proper technique for the construction of poverty indices (Asselin & Anh. 2008). The technique comes from the family of 'factorial' techniques which aim to decrease the dimension of the dataset and understand the data patterns. The idea is the same as Principal Component Analysis (PCA). While PCA works well with numerical data, MCA is a proper technique to use for categorical and ordinal data (Asselin & Anh. 2008).

To understand the MCA technique, we first need to explain its predecessor - Correspondence analysis (CA). MCA is actually an extension of CA and the computational details will be described below. The definitions are inspired by Abdi & Valentin (2007) and Greenacre (2010).

4.1.1 Correspondence analysis

Let K be the amount of nominal variables, each having J_k levels so that $\sum_{k=1}^{K}(J_k) = J$. Let I denote the number of observations. We create a IxJ indicator matrix X. Let N denote the grand total of observations, such that $N = \sum_i \sum_j (x_{ij})$.

The first step is to compute a correspondence matrix Z by dividing X by N, $Z = N^{-1}X$. This corresponds to standardizing all the variables in the matrix. Next, we will be using the following notation: r denotes the vector of row totals of Z and c the vector of column totals of Z, D_r denotes the diagonal matrix of vector r, $D_r = diag\{r\}$ and D_c the diagonal matrix of vector c, $D_c = diag\{c\}$.

The second step is to obtain the factor loadings. To do that, we have to calculate the singular value decomposition (SVD) from matrix S which represents the standard residuals:

$$S = D_r^{-\frac{1}{2}}(Z - rc^T)D_c^{-\frac{1}{2}} = P\Delta Q^T$$

Singular value decomposition is a frequently used technique in 'factorial' methods such as correspondence analysis. The decomposition lies in representing any matrix by the product of three matrices, such that P and Q are the matrices of left and right singular vectors respectively, and the middle matrix Δ contains singular values which are in decreasing order $\alpha_1 \geq \alpha_2 \geq \alpha_3 \geq \cdots$. The benefit of using SVD is linked to the Eckhart-Young theorem (Greenacre. 2010). According to this theorem, if we only choose m vectors from P and

Q matrices and corresponding m singular values in Δ, the matrix $S(m) = P(m)\Delta(m)Q^T(m)$ which we obtain, is the least-squared rank m approximation of S (Greenacre. 2010). In other words, SVD is a computationally effective method for finding low-dimensional subspace that best fits the original space represented by the original matrix.

The output that we wish for is some kind of representation of our data in the low dimensional space. Therefore we need to calculate the projections of our data to the identified subspace. The third step of the algorithm does exactly this. We denote the row and column standard coordinates as Φ and Γ respectively; and the row and column principal coordinates as F and G respectively. The row and column coordinates contain the information of how our variables are represented in the new subspace. In other words, the row coordinates represent the rows of the matrix, i.e. our observations and the columns of the matrix represent the principal axes, i.e. dimensions. There are $min\{I-1, J-1\}$ of these axes.

$$\Phi = D_r^{-\frac{1}{2}} P$$
$$\Gamma = D_c^{-\frac{1}{2}} Q$$
$$F = D_r^{-\frac{1}{2}} P\Delta$$
$$G = D_c^{-\frac{1}{2}} Q\Delta$$

The last step of the algorithm that is important for our cause is calculating the total inertia. This corresponds to the variance that explains our data. The principal inertias $\lambda_1, \lambda_2, \cdots$ are actually squared singular values: $\lambda_i = \alpha_i^2$, $i = 1, \cdots, k$; where $k = min\{I-1, J-1\}$.

The total inertia of the data matrix is then the sum of squares of the matrix S

$$inertia = trace(SS^T)$$

Another interesting information that can be obtained from MCA is the distance of the observations from the barycenter. Unlike PCA which is based on Euclidean distance, CA and MCA are based on χ^2 distance from the barycenter. We use this distance to visualize the output from MCA which can be calculated in the following way:

Distance from rows: $d_r = diag\{FF^T\}$ Distance from columns: $d_c = diag\{GG^T\}$

4.1.2 Extending to multiple correspondence analysis

Multiple correspondence analysis is an extension of the correspondence analysis. There are two forms of MCA that differ in the initial preparation of the data. Suppose that now our original set consists of KxQ categorical variables. The first type of MCA computes the indicator matrix by transforming the categorical data into dummy variables (i.e. variables with only two levels), then performing the CA algorithm. The second type uses the Burt matrix $B = Z^T Z$. The Burt matrix gives the same factor results while being more efficient. Eigenvalues obtained from the Burt matrix are also a better estimation of intertia (Abdi & Valentin. 2007). The algorithm runs in the same manner as in the case of an indicator matrix but used with the Burt matrix.

4.1.3 Creating poverty indicators

We have performed MCA on the variables from three types of assets (human, physical, financial) that were all coded as factor variables with two or three levels. The MCA was performed on the dataset of 6,388 respondents (households). The results show that dimension 1 explains 53.574% of total inertia. We visualize the output in figure 1. The chart represents the scatterplot where on *x*-axis we show Dimension 1 and on *y*-axis we show Dimension 2. The values for these dimensions represent the distances from the columns, i.e. the principal axes.

Upon visual inspection it is clear that there exists a pattern in our data. The categories which we associate with non-poverty (i.e. garbage collection, running sewage, hygienic toilet, savings, etc.) are clustered on the right side of the chart. The indicators which we associate with poverty are clustered in the left part of the dimension plot. We can see that to explain the poverty pattern it is enough to use the first dimension.

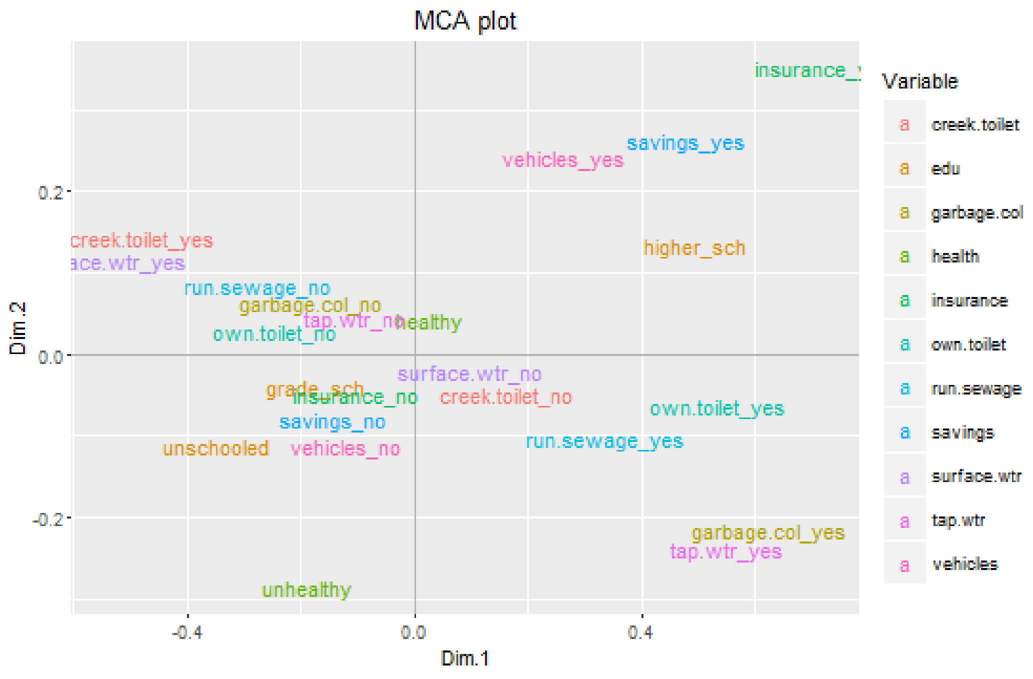

Figure 1: MCA Dimension plot for variables

Multiple correspondence analysis was crucial for us to determine whether the household is poor or not. We used the package *FactoMineR* and function *MCA*. The syntax for the function is the following:

```
    MCA(X, ncp = 5, ind.sup = NULL, quanti.sup = NULL, quali.sup = NULL,
graph = TRUE, level.ventil = 0, axes = c(1,2), row.w = NULL, method = "
Indicator", na.method = "NA", tab.disj = NULL)
```

We used the following function arguments:

- *X* - matrix or data frame of categorical variables coded as factors
- *ncp* - the number of dimensions, we have kept the default value
- *method* - either *Indicator* or *Burt* matrix to use for computation. We have used the *Burt* option.

The important part of the output includes the factor loadings for each category and the total inertia captured by our model. The full output from the MCA analysis is included in the Appendix. To compute the poverty index, we used the factor loadings in the following equation:

$$PI = \sum_j (X_{ij} W_j),$$

where PI stands for Poverty Indicator and W_j stands for the weight of the jth variable obtained from MCA. The weight is actually a factor loading from Dimension 1. The poverty index has distribution visualized by histogram in figure 2.

In other words, the MCA output represents the weights that we give to each category. We have seen on the variable map that the negative scores are associated with the poor and the positive scores with the non-poor.

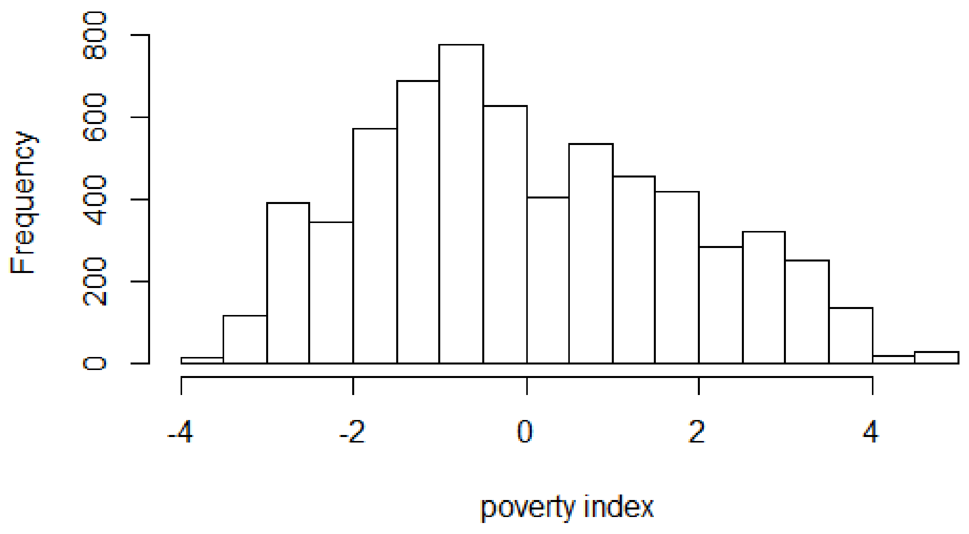

Figure 2: Histogram of the poverty index

Our next step was using the K-means algorithm to recode the poverty index into a binary variable.

4.2 K-means

The second step of our analysis is to determine the poverty levels. K-means clustering is a suitable solution for this problem. The method divides and partitions the data into K clusters which are not overlapping. The criterion for good clustering output is when the within-cluster variation is as small as possible. The definitions that follow are based on James et al. (2013).

Let $C_1, \cdots, C_K$ represent sets which contain indices of our observations. These sets satisfy two conditions:

1. Unification of all sets C_i covers every observation. $C_1 \cup C_2 \cup \cdots \cup C_K = \{1, \cdots, n\}$
2. The sets C_i are not overlapping, i.e. $C_i \cap C_k = \emptyset$, for all $i \neq j$

In other words the sets are such that every observation belongs to exactly one cluster.

We denote the within cluster variation as $W(C_k)$. $W(C_k)$ represents a measure by which observations in one cluster differ from each other. Thus our goal is to find

$$\min_{C_1,\ldots,C_K} \{\textstyle\sum_{k=1}^{n} W(C_k)\}$$

When we work with numeric data it is wise and also most common to use the squared Euclidean distance as a measure of the within cluster variance (Ng. 2015). We first calculate the squared Euclidean distances of all pairs of observations in the kth cluster. Secondly, we sum these and divide by the number of observations found in the kth cluster. Mathematically we can write it in the following way (James et al. 2013):

$$W(C_k) = \tfrac{1}{|C_k|} \sum_{i,i' \in C_k} \textstyle\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

The algorithm for finding the minimum of the function above is very simple and follows three steps:

1. Initialize cluster membership of each observation by random assignment.
2. Iterate until no further change in cluster assignments
   - Compute the centroid of each of the K clusters. The centroid is calculated as a feature mean of all observations in the respective cluster.
   - Assign each observation to the centroid which is closest, as defined by the Euclidean distance

This algorithm is suitable for finding not the global, but the local optimum and has proved to work well in practical exercises. To make sure that the good enough local optimum was found it is important to run the algorithm more times with different initializations.

4.2.1 Determining the poor

One problem with this algorithm is that we need to determine the number of clusters. This is not something that we are always aware of right from the beginning. One way to resolve this problem is to make a scree plot with different choices of K and their respective within cluster variance (Ng. 2015). The scree plot for our data in figure 3 shows how the within cluster variance decreases when more clusters are added. The logical and suitable choice for K is the point after which the variance decreases only minimally. We can spot this point by visual inspection of the scree plot as it resembles an elbow (Ng. 2015).

As we can notice on the scree plot for our dataset, there is a large decrease of variance already by choosing 2 clusters, but after four clusters the slope starts to even out. Therefore we have chosen four groups.

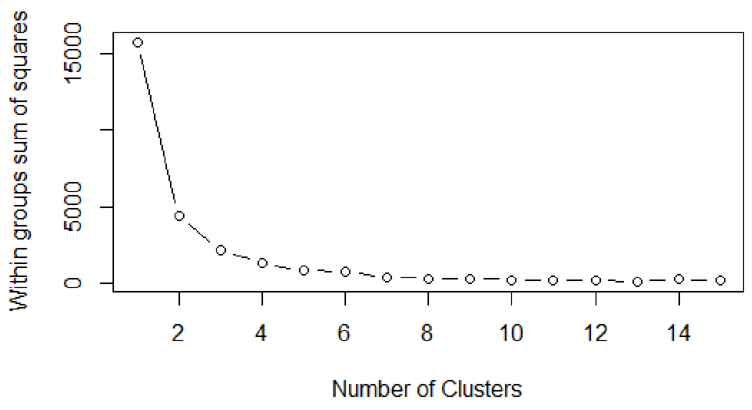

Figure 3: Scree plot for poverty index

We used the K-Means Clustering algorithm in R. The function syntax is the following:

```
kmeans(x, centers, iter.max = 10, nstart = 1, algorithm =
c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace=FALSE)
```

We used the following arguments for the function as they are sufficient to give us the output which we want.

- x - matrix of data, must be numeric; poverty index in our case
- centers - number of clusters, we used the value 4

We also include a scatter plot for our poverty index in figure 4. The x-axis represents an index for all observations and the y-axis shows the poverty index. The color coding is based on the clustering obtained through K-Means. We categorize the poor as those who belong to the black group. In this way we classified about 20% of our dataset as poor.

For the following set of algorithms we create an extra variable called "poverty" which has two values: 0 for non-poor and 1 for poor. This section thus concludes the overview of the algorithms of unsupervised learning.

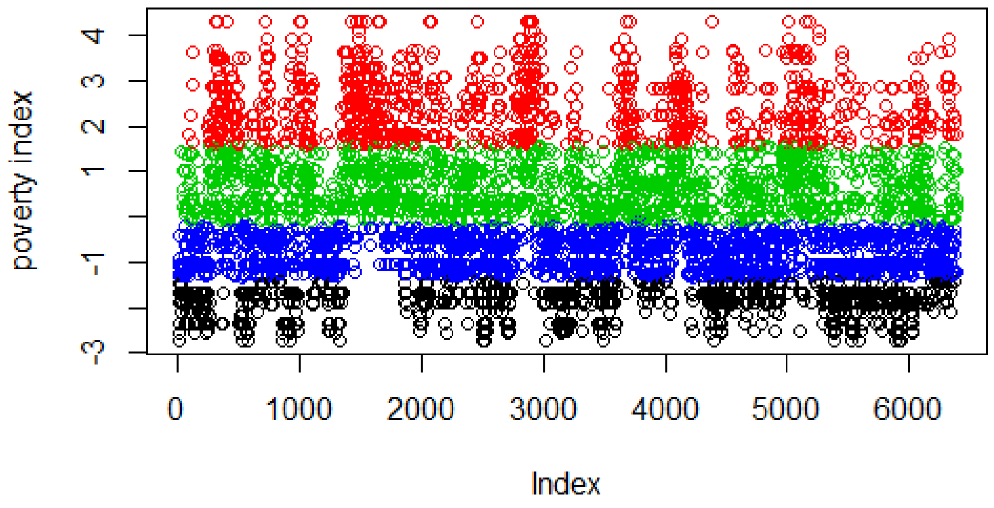

Figure 4: Clusters based on the poverty index

4.3 Binary logistic regression

Logistic regression is a convenient method to use when we have a qualitative dependent variable. The requirement for this method is that the dependent variable is coded as binary, i.e. reaches values of 0 when the characteristic is not present and 1 when the individual has the characteristic. The desired outcome for such scenario would be the probability with which the person has the characteristic. The logistic regression model is thus based on the logistic function:

$$p(X) = \frac{e^{B_0+B_1x_1+\cdots+B_nx_n}}{1+e^{B_0+B_1x_1+\cdots+B_nx_n}}$$

Where $p(X)$ is the probability that a person has the characteristic and B_i are the unknown parameters. The best estimate for the unknown parameters is such that will associate a low probability to respondents which do not possess the feature and vice versa will assign high probability to individuals who have the feature.

There are more ways how we can obtain the best estimate of the parameters. One method is through maximum likelihood which uses the following function

$$l(B_i) = \prod p(x_i) \prod (1 - p(x_{i'}))$$

To obtain the best fitted parameters we maximize this function (James et al. 2013). However we will show an approach that uses the Gradient Descent (Ng. 2015). Firstly, we define our hypothesis function which reflects the learning problem that we wish to solve. For logistic regression we use the sigmoid function $h_w(x) = \frac{1}{1+e^{-w^t x}}$. We can observe that this function has the properties that we want, i.e. with rising x the function reaches values close to 1 and with decreasing x the function reaches values close to 0. Let y denote the vector of outputs and m the number of cases. The cost function can then be defined in the following way (Ng. 2015):

$$f(w) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_w(x^i) + (1-y^{(i)}) \log(1-h_w(x^i))$$

Finding the best estimate for logistic regression is then equivalent to finding the minimum of the cost function. We can use the Gradient Descent algorithm for finding the minimum. Typically a regularization parameter will also be added to the equation and the final function to minimalize will be $min\{f(w)+R(w)\}$, where $R(w) = \lambda \, \|w\|$.

4.3.1 Interpretation of the output

Scrutinizing the logistic function we may notice that it always reaches values between 0 and 1 which was what we desired. The output is typically interpreted either through odds ratio or probabilities. The odds ratio can be calculated easily from the function above where we get:

$$Oddsratio = \frac{p(X)}{1-p(X)} = e^{B_0+B_1x_1+\cdots+B_nx_n}$$

The odds ratio reaches values between 0 and infinity. Values that are close to zero, i.e. the odds ratio is very small, will receive very low probabilities while values that are very high, i.e. odds ratio is very high, will receive high probability rates. The interpretation through odds ratio or probabilities is equivalent. For example, imagine the probability $p(X) = 0.2$. Calculating the odds ratio will produce $\frac{0.2}{1-0.2} = \frac{1}{4}$. This would mean that 1 in 5 people (which corresponds to the odds ration of $1/4$) has the observed feature.

Interpreting the impact of the individual variables is more complex for logistic regression due to its non-linearity in X. With a little bit of manipulation, by taking a logarithm of the odds ratio, we obtain the log-odds. The log-odds are now linear in X and we can interpret the impact of the variables. If x_i changes by one unit and all the other $x_j, j \neq i$ are kept constant, the log-odds increases by B_i. Still, the explanation is rather obscure. The sign of the coefficient is typically enough to give us an idea about the direction of the relationship. If the coefficient is positive, we may say that with increasing values of x_i, the probability of a feature rises. The negative coefficients are associated with decreasing probability.

$$\log \frac{p(X)}{1-p(X)} = B_0 + B_1x_1 + \cdots + B_nx_n$$

4.3.2 Fitting the logistic model

We have run the logistic regression on our training set which contains 60% of all data. We used the *glm* function in R which has the following syntax:

```
    glm(formula, family = gaussian, data, weights, subset, na.action,
start = NULL, etastart, mustart, offset, control = list(...), model =
TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ...)
```

We have used the following arguments:

- formula - formula in the form ’poverty’ ’predictors’
- family - we used ’binomial’ to indicate the binary regression

From the obtained probabilities we chose a threshold of 0.5. That means that any case which had a probability lower than 0.5 was classified as non-poor (0) and vice versa all individuals with a probability higher than 0.5 were classified as Poor (1). Comparing the predicted values and real values we have constructed the following confusion matrix (Table 4.3.2)

Predicted / Truth	Non-poor(0)	Poor (1)
Non-poor(0)	2009	370
Poor (1)	123	193

From the first look we can see that the model does quite a good job when correctly assigning households as not poor (94% for specificity) but does rather poorly when predicting the poverty with only 34% sensitivity. This may be caused by the disproportion of poor and non-poor in our data. The total accuracy for the training set is around 82%. We now use this model to predict poverty on the testing ("unseen") dataset (Table 4.3.2).

Predicted / Truth	Non-poor(0)	Poor (1)
Non-poor(0)	1354	255
Poor (1)	84	105

The specificity for the testing set is around 94%, sensitivity is 29% and accuracy is 81%. The low values for sensitivity raise concerns as the model does not accurately assign the poor. We will now continue with a different algorithm to see how changing the model can improve the accuracy.

4.4 Neural networks

Neural networks is a supervised machine learning method which mirrors how biological neurons work in a human brain. The architecture consists of neurons which are connected through communication lines that transmit the information. We will describe the neural network through a graphical representation where neurons are shown as nodes and communication networks are shown as lines connecting neurons together. The theoretical grounding for this chapter is based on Shalev-Schwartz & Ben-David. 2014.

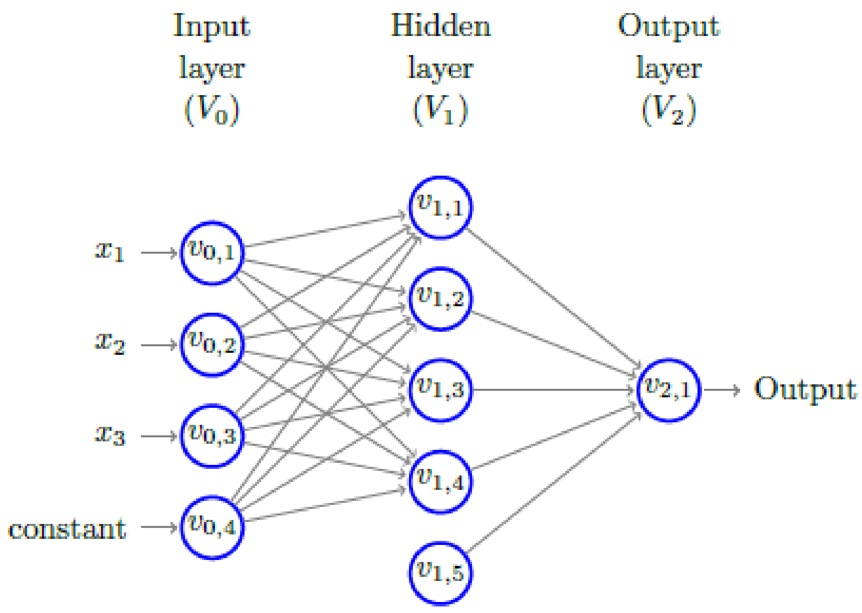

Figure 5: Visualization of neural network

The network is represented by the graph $G = (V,E)$, the weight function $w : E \to R$ and scalar function $\sigma : R \to R$ which models each neuron. An example of the graph G is visualized in figure 5. σ is called the activation function of the neuron. Neurons are typically concentrated in layers $V_t, t = 1, \cdots T$, which represent an organization of the neurons into disjoint subsets. The first layer, V_0 is called an input layer. Every neuron from the input layer connects to every neuron in the following layer V_1. If n is the number of predictors, then the dimensionality of V_0 is $n+1$. The last neuron in the layer is called a constant and outputs to 1. The input for any neuron from layer V_1 is calculated as the weighted sum of the output of the neurons from the preceding layer. The weighting that applies is given by the function w. In the same manner we continue the calculations to the following layers until we reach the last layer - the output layer. Typically, in simple prediction scenarios the output layer V_T consists of one neuron. Layers $V_1, \cdots V_{T-1}$ are called hidden layers (Shalev-Schwartz & Ben-David. 2014).

In mathematical notation, let $v_{t,i}$ denote the ith neuron in the tth layer. By $o_{t,i}(x)$ we denote the output of the $v_{t,i}$ fed by the input vector x. Let $w(v_{t,i}, v_{t+1,j})$ denote the weight or the line that connects the ith neuron in layer t with the jth neuron in layer $t+1$.

Suppose now that we have calculated the network for some tth layer. We want to calculate the output for a fixed neuron in the $t+1$th layer, i.e. neuron $v_{t+1,j}$. First we need to calculate the input to this particular layer, here denoted as $a_{t+1,j}(x)$. We simply sum across the outputs from the neurons of the preceding layer considering their particular weight (Shalev-Schwartz & Ben-David. 2014):

$$a_{t+1,j}(x) = \sum_{i:(v_{t,i},v_{t+1,j})\in E} w(v_{t,i}, v_{t+1,j}) o_{t,i}(x)$$

By x we denote the input vector that is fed to the algorithm of neural networks. When we have calculated the input to the $v_{t+1,j}$th neuron, we apply our σ function to calculate the output (Shalev-Schwartz & Ben-David. 2014):

$$o_{t+1,j}(x) = \sigma(a_{t+1,j}(x))$$

The σ function is typically signum, threshold or sigmoid function. Our model used the sigmoid function, which is the following (Shalev-Schwartz & Ben-David. 2014):

$$\sigma(t) = \frac{1}{1+e^{-t}}$$

The steps described above are steps of the forward propagation algorithm for neural networks. The learning aspect of neural nets is to tune the weights over the edges. The next part explains how the learning of neural networks takes place.

4.4.1 Learning of Neural Networks - Backpropagation

The task of learning neural nets is equivalent to finding a minimum of the risk function $f_w(x)$. Let us denote our hypothesis function of w as $h_w(x) \in R^K$ where $h_w(x))_i$ represents the ith output. Then the cost function is as follows (Ng. 2015):

$$f_w(x) = -\frac{1}{m}[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_w(x^{(i)}))_k + (1-y_k^{(i)})\log(1-(h_w(x^{(i)}))_k)]$$

Finding the minimum of the cost function is again suitable for the Gradient Descent method. One important improvement to the algorithm is the initialization. Recall that frequently the initialization is assigning $w_{(0)} = 0$. However, this will result in having the same weights for neurons in the hidden layer. Consequently, the input to the following layer will be the same and our model will not provide accurate estimates (Ng. 2015, Shalev-Schwartz & Ben-David. 2014). We want to avoid this scenario which is easily done by random initializations. Random initialization is tricky as it might converge in some local minimum which is far away from the result we wish to obtain. We avoid this by repeating the algorithm multiple times each with different initialization and taking the best estimate (Ng. 2015).

The Gradient descent method is dependent on calculating the gradient. Since neural networks is rather a complex method, we will dedicate some space to explaining how the gradient descent is computed. However we will not delve into the details as it is not in the scope of this thesis and we will explain only the main idea.

We introduce another bias term which we denote as $\delta_{i,j}$ referring to the bias at jth node in ith layer. We compute the bias term in a backward fashion, i.e. first we calculate all the outputs up to the last layer. The bias term in the last layer is calculated first as $\delta_T - y^i$. Consequently by using the inverse of our activation function we can calculate all the bias terms down to the first - input layer. Finally, when these are updated we calculate the gradient in the following way (Ng. 2015):

$$\triangle_{t,ij} := \triangle_{t,ij} + o_{t,j}\delta_{t+1,i}$$
$$\triangle f_w(x) := \frac{1}{m}\triangle_{t,ij} + \lambda w_{t,ij} \text{ where } w_{t,ij} = w(v_{t-1,i}, v_{t,j}).$$

We see that the gradient is actually computed by using the bias terms. This is repeated in an iterative manner to obtain the best estimates. The backpropagation serves also another important role. It replaces the regularization function. The details of the backpropagation algorithm can be found in Shalev-Schwartz & Ben-David (2014) or Ng (2015).

4.4.2 Fitting neural networks

Neural networks was the second algorithm tested on our dataset. We have used the R package *nnet* which has the following structure. Since the call function has an option to choose many arguments, we include only those that were important to us and describe their meaning. `nnet(x, x, y, size, Wts, lineout, ...)`

1. x - matrix or data frame of predictors
2. y - matrix or data frame of outcomes, for our case this was the created binary 'poverty' indicator
3. size - number of neurons in a hidden layer, we used 10 nodes.
4. Wts - initial parameter vector by default random
5. lineout - determining the type of outputs by default set at logistic

There are more R packages that can calculate neural networks but we find this the most suitable one. Firstly, it enables data to be input through a matrix of predictors - x and matrix of targets - y; secondly it initializes the weights at random and thirdly, by default if calculates the logistic (sigmoid) output. The disadvantage is that it does not allow for visualizations and that it allows only one hidden layer. To overcome the visualization problem we used the code created by Beck (2016). The limited number of hidden layers was not considered a problem to us since commonly one hidden layer is enough to achieve high levels of accuracy (Ng. 2015).

Figure 6 shows the actual output that we obtained. The thickness of the line represents the strength of the association. The color represents whether the input is negative (grey) or positive (black).

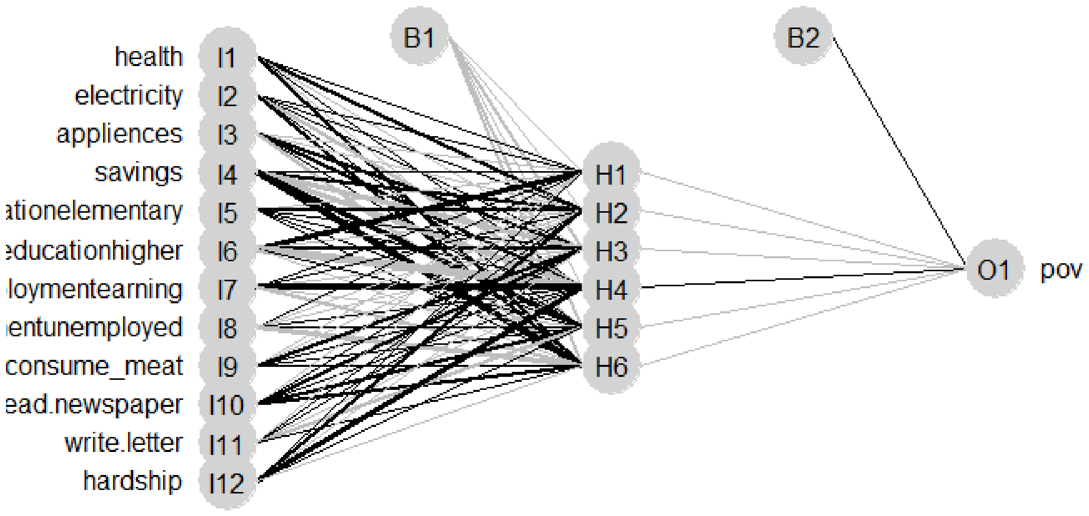

Figure 6: Neural networks

We can observe that the output is rather hard to read. Therefore we show an alternative view on the importance of the variables which is again inspired by the program created by Beck (2016). Firstly, all connections that go from the input variable all the way through the hidden layers to the output layer are identified. Secondly, all the respective weights are scaled relative to all other inputs. Then, the single value is obtained for each input variable. The interpretation will only tell us which of the variables are more important in comparison to the other variables. Figure 7 should not be viewed in light of this explanation.

Lastly, we test the accuracy of our model. The accuracy measures for the testing set are given in the Table 4.4.2.

Predicted / Truth	Non-poor(0)	Poor (1)
Non-poor(0)	1330	226
Poor (1)	108	134

The sensitivity measure is around 37%, specificity is 93% and accuracy is 81%. Typically, neural networks provide higher accuracy than logistic regression and we see that mea-

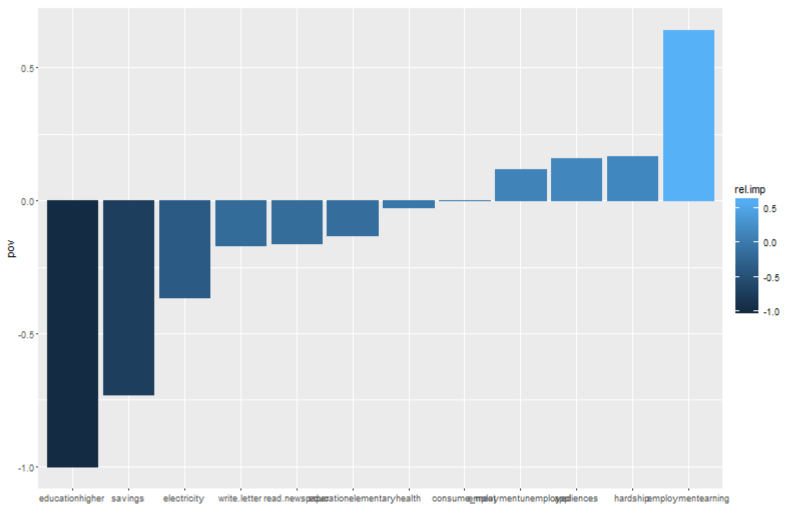

Figure 7: Relative importance of variables in neural networks

sures have increased slightly. To correct the model we could gain more data or re-define input variables.

4.5 Decision Trees

Decision trees represent a simple method for prediction which is easy to interpret and suitable for classification. As the name suggests, composition of the model can be defined as a set of logical decisions that segment the predictor space into simple regions. There are two types of trees that can be constructed - regression and classification trees. Classification trees are suitable to predict a qualitative outcome, as our dependent variable 'poverty'. Therefore we will limit ourselves to defining these. Definitions and explanations in this chapter are based on Shalev-Schwartz & Ben-David (2014) and James et. al (2013).

The way how we grow the tree is called recursive binary splitting. First we take the whole dataset which represents the root of the tree. Based on the first predictor we split the dataset into two regions. Typically the decision is based on some threshold criteria and depending on the value of the feature we either move to the right or to the left node. We consider binary predictors as can be found in our dataset. Therefore we move to the right when the answer is 1 or to the left when it is 0. In the same manner we continue until we have built the tree (James et. al. 2013).

A common problem to this approach is overfitting. If we do not limit the size of our trees we can obtain an almost perfect fit for our trained dataset. However, this does not necessarily mean that it will perform well on the testing dataset. In the next section we describe the algorithm for a decision tree and also ways how to overcome the overfitting problem.

4.5.1 Decision Trees Algorithm

We will now describe a general approach to building a decision tree. First we start with the tree that has only one single leaf. We assign this leaf a label that represents the majority in our training set. We perform iterations in which we split the single leaf. We examine how the split affected the single node. This is defined as a gain measure which will be discussed later. We choose the split that achieves greatest gain and perform the split. Then the process continues. We can also choose not to split the leaf at all. The implementation of this method is described by ID3 (Iterative Dichotomozer 3) (Shalev-Schwartz & Ben-David. 2014).

```
1. input: training set $S$, feature set $A$
2. if all examples in $S$ are labeled by 1, return a leaf 1
3. if all examples in $S$ are labeled 0, return a leaf 0
4. $A = \emptyset$, return a leaf whose value = majority of labels in $S$
5. else:
    • Let $j = argmax_{i \in A} Gain(S, i)$
    • if all examples in $S$ have the same label, return a leaf whose value = majority of labels in $S$
    • else
        – Let $T_1$ be the tree returned by ID3 $((x, y) \in S : x_j = 1, A \setminus \{j\})$
        – Let $T_2$ be the tree returned by ID3 $((x, y) \in S : x_j = 0, A \setminus \{j\})$
        – Return the tree
```

4.5.2 Different approaches to Gain measure

As may be noticed from the algorithm above, the Gain measure is the most crucial aspect of it. There are more approaches and each algorithm addresses this issue differently. Typically, they give similar results and we will discuss few of them below. The theory is based on Shalev-Schwartz & Ben-David (2014).

Train Error: This is the simplest definition of Gain measure. The idea is to calculate the difference between error rate before the split and error rate after the split. Before the split, the training error of feature i is $C(P_S[y = 1])$ where P_S is the probability the event holds considering the distribution over our sample S. Recall that we took the majority vote among the labels of the feature. In this way we can calculate the error after splitting on feature i:

$$P_S[x_i = 1]C(P_S[y = 1|x_i = 1]) + P_S[x_i = 0]C(P_S[y = 1|x_i = 0]).$$

The Gain measure is then the difference of the two error rates:

$$Gain(S,i) = C(P_S[y=1]) - (P_S[x_i=1]C(P_S[y=1|x_i=1]) + P_S[x_i=0]C(P_S[y=1|x_i=0])).$$

Information Gain: This Gain measure is also used in the ID3 algorithm. The idea is similar to Train error but instead of probabilities, the Information Gain uses the entropy function which is defined in the following way:

$$C(a) = -a\log(a) - (1-a)\log(1-a)$$

As can be seen from the chart below, the advantage of the entropy function is that it is smooth and concave. These properties are convenient for some situations (Shalev-Schwartz & Ben-David. 2014).

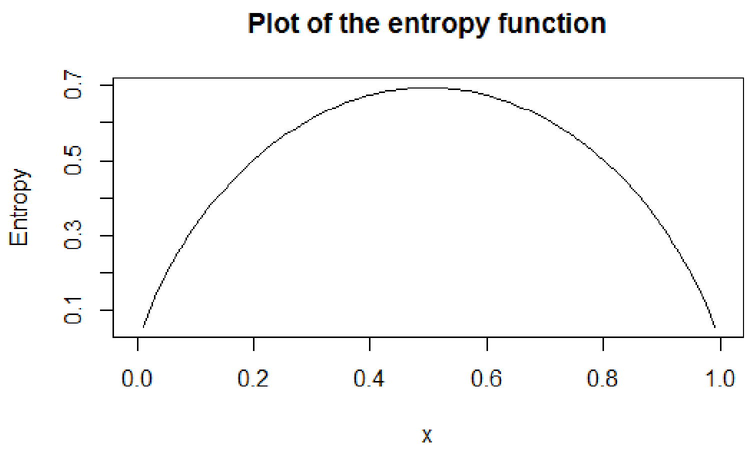

Figure 8: Entropy function

Gini index: The last definition of Gain measure we will explain here is the Gini index which is calculated in the following way. It also has properties of smoothness and concaveness.:

$$C(a) = 2a(1-a)$$

4.5.3 Fitting the Decision Tree

The task of fitting the Decision tree is supported by various packages in R. We have chosen the *rpart* package. The function for fitting the tree is the following:

```
    rpart(formula, data, weights, subset, na.action = na.rpart, method,
model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)
```

The arguments that we used were:

formula - the formula to be fitted, we used ’poverty’ ’predictors’

data - defines data frame

method - specifies which type of decision tree should be fitted, we used ’entropy’ for the classification tree

The results that we obtained are summarized in Table 4.5.3:

Predicted / Truth	Non-poor(0)	Poor (1)
Non-poor(0)	1359	259
Poor (1)	79	101

The sensitivity for this model is 29%, specificity is 94% and accuracy is 81%. There are additions to algorithms to improve the accuracy which we will discuss in the section below.

4.5.4 Pruning of the tree

We have mentioned above that the decision trees might result in a small training error but a large testing error. This is typically caused when we grow a very large tree that overfits our training data. In this section we will introduce the idea of pruning of the tree based on Shalev-Schwartz & Ben-David (2014). The idea of pruning is to reduce the size of the tree after it is built with a much smaller tree by keeping the empirical error. The pruning algorithm is performed from bottom to top, i.e. from leaves to the root. As the algorithm runs, each node is replaced with either its subleaf or a leaf.

We do not perform tree pruning in this thesis. However we dedicate the next section to another technique that improves the model.

4.6 Random Forests

As the name suggests, random forests is a technique that analyzes many decision trees. The disadvantage of the single decision tree is its high variance and dependency on the training dataset. This can be avoided with the increased number of trees that are considered. In a fantasy world we could draw many samples, grow decision trees for each sample and base our conclusions on some averaged value obtained from the individual trees. In reality, this is not feasible since we typically do not have resources for drawing many samples. The technique called ’bagging’ simulates this scenario using only our original sample. We provide a brief overview derived from James et al. (2013).

Bagging is a bootstrap procedure which takes repeated samples from the single data set. In this way we obtain B subsamples and on each we grow a decision tree. As a prediction class for our test observation we take a majority vote (for classification problems) or average (for regression problems).

Growing a random forest takes one step further to the bagging procedure. When a decision split is to take a place, the algorithm takes a random sample of *m* predictors from the total number of *p* predictors. These play the role of the split candidates for that particular split. A new sample of predictors is in this way taken at each decision split. The rule of thumb is to take $m \approx \sqrt{p}$. Since the sample of predictors is taken at random, we avoid the problem of having one strong predictor present in each bagged tree. Thus we limit the correlation of the trees.

4.6.1 Growing random forest

The R package for growing random forests is called *randomForest*. There are many arguments to the function, but we show here those that are relevant for our cause.

```
   randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500, mtry=if
(!is.null(y) && !is.factor(y)) max(floor(ncol(x)/3), 1) else
floor(sqrt(ncol(x))), replace=TRUE, classwt=NULL, nodesize = if
(!is.null(y) && !is.factor(y)) 5 else 1, importance=FALSE, ...)
```

- ntree - number of trees to grow, suggested not to use too small number. We keep the default.
- mtry - number of predictors to be randomly chosen, we choose $\sqrt{p} \approx 4$
- replace - indicates whether the sampling of cases should be done with replacement. We keep the default TRUE.
- nodesize - minimum size for the end nodes. By default this value is 1 for the classification.
- importance - whether the importance of the predictors should be assessed. We set this value as TRUE.
- ... other arguments. We do not use any other arguments.

The output of the random forest algorithm also includes the confusion matrix for the training set shown in Table 4.6.1

Predicted / Truth	Non-poor(0)	Poor (1)
Non-poor(0)	2002	130
Poor (1)	377	186

The accuracy measures for the training dataset are: 58% sensitivity, 84% for specificity and 81% for accuracy. The accuracy measures for the testing dataset are: 31% for sensitivity, 93% for specificity and 81% for accuracy (Table 4.6.1)

Predicted / Truth	Non-poor(0)	Poor (1)
Non-poor(0)	1347	245
Poor (1)	91	115

4.7 Predictive ability of algorithms

In this chapter we have reviewed four algorithms that are commonly used for prediction problems. Now we will summarize their advantages and disadvantages that are inspired by applying them to our dataset.

Logistic regression:

Binary logistic represents a convenient predictive method which can be easily applied to a non-linear problem. Its main disadvantage is the need for the binary outcome variable. Since we recoded our Poverty index into a binary variable we have lost certain information which could have caused the low sensitivity. The main advantage is in its interpretability. Even though it is not the easiest one, we can easily connect the impact of a certain predictor to the outcome. However, we do not have much option to tuning the parameters and to improve the model we would need to replace the predictors or include more data.

Neural networks:

The second algorithm that we used reached the highest levels of accuracy. This is the main strength of the algorithm (Shalev-Schwartz & Ben-David. 2014). On the other hand, its drawback lies in the interpretability. It is almost impossible to connect the input variable to the outcome. We cannot say that by changing values of one predictor we will increase or decrease the output variable by a certain value. This might be problematic to a problem of poverty prediction where this connection might be interesting to the development agencies or governments. The second problem is connected to random initialization of the model. Since we need to run it multiple times, this can be computationally expensive, especially with large datasets.

Decision trees:

Regarding the accuracy models this algorithm performs similarly to binary logistic regression. However, unlike the logistic regression, the decision tree algorithm has very high variance. It is prone to overfitting and pruning or other algorithm is necessary to correct for this. On the other hand, it does not have such strict demands on the form of the outcome.

Random forests:

Like neural networks, the main disadvantage of random forests is their interpretability. However, unlike neural networks, their output includes the importance of the predictors. Random forests were the second best algorithm that we have tried.

5 Poverty in 1997

As the last step of our analysis we used the trained models on the data from the following wave that was collected in 1997. This chapter is dedicated to briefly summarizing the results. First, we have used the same variables in 1997 that were used as descriptors and predictors in 1993. In total we had 9906 household responses. Therefore our results are comparable. The details can be found in the Appendix. We have obtained the following accuracy measures for the distinct models in 1997 (Table 5):

Model	Sensitivity	Specificity	Accuracy
Binary logistic regression	16%	97%	85%
Neural nets	24%	95%	84%
Decision trees	14%	98%	85%
Random Forests	18%	97%	85%

From the table above it is obvious that the prediction accuracy dropped rapidly by more than 10%, especially when it comes to the Sensitivity measure. We can conclude that the model itself is not accurate enough and will require certain tuning. One way would be to include more predictors or to replace the current ones. This will require further research on the theoretical part as the variables that are included should be conceptually accurate. The other solution would be to tune the parameters of the respective models. There are a few suggestions found in the literature in James et al. (2013) and in Shalev-Schwartz & Ben-David (2014).

6 Conclusion

The aim of this thesis was to look deeper into the problematic nature of predictive models. In order to do that, we had to take several steps to prepare our data to be suitable for the methods that we wanted to use. In the light of this we used Multiple Correspondence analysis to work with categorical variables and created a poverty index that is a numerical representation of information extracted from our categorical descriptor variables. We chose categorical variables that captured information about household conditions, education and health of the household head. The choice of these descriptors was justified by the theoretical framework that represents our view of poverty as a multidimensional concept. Consequently we have applied the K-Means algorithm on the newly created numeric index of poverty to cluster our respondents into four groups. The last group with the lowest index values was categorized as poor.

With this new variable we could proceed with predictive modeling. Models we used belong to the group of supervised learning algorithms. As the name suggests, these algorithms require that both the predictors and the outcome are included in the model. The learning then takes place which represents the process of tuning the parameters of an algorithm. We used four methods: Binary logistic regression, Neural networks, Decision Trees and Random Forest. The main problem with all models was the low values of Sensitivity, i.e. the measure of how well the model predicts the poor among those who are poor. One possible reason might be that the predictors that we chose were not sufficient. We believe that introducing income and expense variables would result in higher levels of sensitivity and thus also accuracy. Due to the scope of this thesis we did not revise the original dataset as it would require broadening our view of poverty.

Our results show that Neural networks is the most accurate method for prediction based on our measures of accuracy. However, the trade-off for higher precision is low interpretability. The output of this algorithm does not provide information on the importance of the predictors. We imagine there are tasks where only accuracy is important. However, if we want to link our work with development practice, understanding how different factors influence poverty is crucial for taking further actions. We introduced another technique which gives information about the relative importance of predictors. However, this output has to be interpreted with caution as it provides information on how strong the predictor is in relation to other predictors.

Binary logistic regression and Decision trees are easily interpreted and provide exact connection of the predictor to the outcome, however they had low levels of accuracy. This was actually our expectation which stems from the nature of these techniques. Both logistic regression and Decision trees are methods devised for linear problems, i.e. problems where the decision boundary is linear (Ng. 2015, James et al. 2014).

Actually, Decision trees was the least performing algorithm based on accuracy. It is also very prone to overfitting. Therefore we conclude that they are not a suitable method for poverty prediction. Random forests is a technique that reduces the overfitting problem of Decision Trees. Hence, their results are more stable and accurate. Again, the trade-off is their limited interpretability. Still, unlike Neural networks, the output for Random forests provides also the importance of the predictors which could be highly appreciated in poverty problems.

To conclude, we claim that Neural networks is the algorithm with the greatest predictive ability. This was also supported when we fed the algorithms new data from Wave 2 (year of collection 1997). However, with the results we obtained, we cannot claim that we have created a proper model for poverty prediction. Some revision has to take place to introduce more or better predictors. We recommend that further research takes place in this direction.

7 Appendices

References

[1] Abdi, H. and Valentin, D. (2007): Multiple correspondence analysis. Retrieved 27-03-2016 from: https://www.researchgate.net/profile/Dominique_Valentin/publication/239542271_Multiple_Correspondence_Analysis/links/54a979900cf256bf8bb95c95.pdf

[2] Asselin, L.-M. And Anh, V.T. (2008) : Multidimensional Poverty Measurement with Multiple Correspondence Analysis. In: Kakwani, N. And Silber, J. (eds.) Quantitative Approaches to Multidimensional Poverty Measurement. Palgrave Macmillan

[3] Breiman, L. (2001): Statistical Modeling: The Two Cultures. Statistical Science. Vol. 16, No. 3, 199-231.

[4] Chambers, R. and Conway, G. (1992): Sustainable rural livelihoods: practical concepts for the 21st century. IDS Discussion Paper, 296. Brighton: IDS.

[5] Greenacre, M.(2010) : Correspondence analysis and related methods [course]. Retrieved 26-03-2016 from http://statmath.wu.ac.at/courses/CAandRelMeth/

[6] Mathiassen, A. (2008): The predictive ability of poverty models. Empirical Evidence from Uganda. 19-09-2015 retrieved from: http://www.ssb.no/a/publikasjoner/pdf/DP/dp560.pdf

[7] James, G; Witten, D., Hastie, T., Tibshirani, R. (2013): An Introduction to Statistical Learning with Applications in R. New York: Springer

[8] Ng, A. (2015) : Machine learning [course]. Lecture materials: https://www.coursera.org/learn/machine-learning. Stanford University: 2015.

[9] RAND Labor and Population: Indonesian Family Life Survey IFLS. Data retrieved from: http://www.rand.org/labor/FLS/IFLS/download.html

[10] Scoones, I. (2009): Livelihoods perspectives and rural development, Journal of Peasant Studies, 36(1), pp. 171-196.

[11] Sen, A. (1999): Development as freedom. New York: Knopf, 1999, pp. 366

[12] Shalev-Schwartz, S. and Ben-David, S. (2014): Understanding Machine Learning : From Theory to Algorithms. Cambridge University Press.

[13] Stojanovic, Miodrag, et.al (2014): Understanding sensitivity, specificity and predictive values. Retrieved 23-04-2016 from: https://www.researchgate.net/publication/278871229_Understanding_sensitivity_specificity_and_predictive_values?ev=srch_pub&_sg=L-CWV2-fv5QZuwg6zQ8h4u0Z092PDLGf1CZjUYHHq5HYEkm_aRQKqtB2qkjEz-QT.bTFXiSNHbYd8DH_1yTnXpz2rfccdfHPlIzRdnrOgDOm8S9hsZowjxYhLhOYSovhV.rD8wBQ9PbbfezfdJsztAeUeGKRU4ZTkOIs1mq-yWXwFvRMjEtHZ7tRqxRRojBsS5

[14] Vella, V., The World Bank (1997): Identification of Standards of Living and Poverty in South Africa. Retrieved 22-11-2015 from https://www.google.sk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0CC4QFjABahUKEwiR8dWF0ajIAhUFvhQKHT6wBj4&url=http%3A%2F%2Fsiteresources.worldbank.org%2FINTPGI%2FResources%2F13278_use_of_alternative_means.doc&usg=AFQjCNHZyu-XeJponNpSSlf75ynMd5A6dQ&sig2=vnpfqkaP565lFRu9Mrgruw&bvm=bv.104317490,d.d24

R Packages and methods

[15] Beck, Marcus W (2016) : nnet_plot_update.r. Retrieved 22-04-2016 from https://gist.github.com/fawda123/7471137/

[16] Everitt, B.S. and Hothorn, T. : Handbook of Statistical Analysis Using R. Retrieved 27-04-2016 from https://cran.r-project.org/web/packages/HSAUR/vignettes/Ch_logistic_regression_glm.pdf

[17] FactoMineR in R – documentation. Retrieved 1-05-2016 from https://cran.r-project.org/web/packages/FactoMineR/FactoMineR.pdf

[18] Ggplot2 in R – documentation. Retrieved 10-02-2016 from http://docs.ggplot2.org/current/

[19] K-Means Clustering in R – documentation. Retrieved 28-04-2016 from https://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html

[20] Nnet in R – documentation. Retrieved 22-04-2016 from https://cran.r-project.org/web/packages/nnet/nnet.pdf

[21] Rpart in R – documentation. Retrieved 25-04-2016 from https://cran.r-project.org/web/packages/rpart/rpart.pdf

7.1 Output from MCA in 1993

Call:

```
MCA(X = pover.mca, graph = FALSE, method = "Burt")
```

Eigenvalues

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6	Dim.7
Variance	0.075	0.012	0.009	0.008	0.007	0.007	0.005
% of var.	53.574	8.515	6.633	5.795	5.349	4.710	3.634
Cumulative % of var.	53.574	62.089	68.721	74.516	79.865	84.576	88.210

	Dim.8	Dim.9	Dim.10	Dim.11	Dim.12
Variance	0.005	0.004	0.003	0.003	0.002
% of var.	3.244	2.797	2.151	1.888	1.711
Cumulative % of var.	91.454	94.251	96.402	98.289	100.000

Individuals (the 10 first)

	Dim.1	ctr	cos2	Dim.2	ctr	cos2	Dim.3	ctr	cos2
1	-0.636	0.023	0.274	-0.233	0.008	0.037	0.442	0.032	0.133
3	-0.777	0.035	0.543	0.201	0.006	0.036	-0.082	0.001	0.006
4	-0.351	0.007	0.315	-0.072	0.001	0.013	-0.291	0.014	0.216
6	-0.564	0.018	0.362	0.042	0.000	0.002	-0.083	0.001	0.008
8	-0.138	0.001	0.018	0.029	0.000	0.001	0.138	0.003	0.018
9	-0.850	0.041	0.423	-0.074	0.001	0.003	0.443	0.032	0.115
11	-0.491	0.014	0.133	-0.452	0.029	0.113	0.891	0.129	0.439
13	-0.132	0.001	0.020	-0.309	0.014	0.111	-0.314	0.016	0.114
15	-0.305	0.005	0.056	-0.095	0.001	0.005	0.291	0.014	0.051
17	-0.361	0.007	0.142	-0.115	0.002	0.015	-0.151	0.004	0.025

Categories (the 10 first)

	Dim.1	ctr	cos2	v.test	Dim.2	ctr	cos2	v.test
grade sch	-0.174	1.852	0.283	-13.957	-0.040	0.604	0.015	-3.178
higher sch	0.496	9.253	0.729	26.498	0.133	4.218	0.053	7.133
unschooled	-0.349	2.799	0.266	-13.452	-0.113	1.850	0.028	-4.360
healthy	0.028	0.084	0.056	5.832	0.043	1.225	0.130	8.858
unhealthy	-0.189	0.564	0.056	-5.832	-0.287	8.193	0.130	-8.858
insurance no	-0.102	1.108	0.563	-21.658	-0.050	1.666	0.134	-10.587
insurance yes	0.721	7.840	0.563	21.658	0.352	11.787	0.134	10.587
surface.wtr no	0.098	0.984	0.447	18.360	-0.021	0.281	0.020	-3.912
surface.wtr yes	-0.540	5.446	0.447	-18.360	0.115	1.556	0.020	3.912
tap.wtr no	-0.104	1.104	0.477	-19.164	0.045	1.292	0.089	8.265

	Dim.3	ctr	cos2	v.test
grade sch	-0.198	19.303	0.365	-15.856
higher sch	0.107	3.514	0.034	5.746
unschooled	0.349	22.653	0.266	13.465
healthy	-0.072	4.465	0.369	-14.924
unhealthy	0.483	29.855	0.369	14.924
insurance no	-0.036	1.139	0.072	-7.726
insurance yes	0.257	8.058	0.072	7.726
surface.wtr no	-0.034	0.945	0.053	-6.329
surface.wtr yes	0.186	5.227	0.053	6.329
tap.wtr no	0.004	0.012	0.001	0.707

Categorical variables (eta2)

	Dim.1	Dim.2	Dim.3
edu	0.418	0.080	0.481
health	0.019	0.113	0.363
insurance	0.269	0.161	0.097
surface.wtr	0.193	0.022	0.065
tap.wtr	0.211	0.098	0.001
creek.toilet	0.288	0.064	0.000
own.toilet	0.480	0.017	0.003
run.sewage	0.337	0.080	0.013
garbage.col	0.419	0.124	0.000
savings	0.254	0.190	0.016

7.2 Binary Logistic Regression

```
glm(formula = pov   ., family = binomial, data = train[, c(3:ncol(train))])
```

```
Deviance Residuals:
     Min          1Q         Median        3Q          Max
 -1.8230124  -0.6849053  -0.2269555  -0.0590958  2.9770558

Coefficients:
                         Estimate     Std. Error   z value              Pr(>|z|)
        (Intercept)       1.31695304  0.24917956   5.28516    0.000000125597160705 ***
          health         -0.79441496  0.15409565  -5.15534    0.000000253175384587 ***
        electricity      -1.10989773  0.11247813  -9.86768  < 0.0000000000000000222 ***
        appliances       -0.56299005  0.11434058  -4.92380    0.000000848800063512 ***
          savings        -1.91093293  0.29649406  -6.44510    0.000000000115526574 ***
   educationelementary   -0.46285176  0.15304578  -3.02427          0.0024923 **
     educationhigher     -3.00786205  0.39783074  -7.56066    0.000000000000040104 ***
    employmentearning    -0.02397278  0.21952407  -0.10920          0.9130411
  employmentunemployed   -0.52598955  0.28934233  -1.81788          0.0690825 .
       consume meat       0.01907007  0.17456161   0.10925          0.9130078
      read.newspaper     -0.58007046  0.31325099  -1.85178          0.0640581 .
       write.letter       0.32372304  0.31247196   1.03601          0.3001990
         hardship         0.15820996  0.11954432   1.32344          0.1856885
—
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2762.3899 on 2694 degrees of freedom
Residual deviance: 2042.2696 on 2682 degrees of freedom
AIC: 2068.2696 newline

Number of Fisher Scoring iterations: 7
```

7.3 Output from Decision tree

n= 2695

```
node)   split                  n     loss  yval  (yprob)
1)      root                   2695  563   0     (0.79109461967 0.20890538033)
2)      electricity>=0.5       1911  229   0     (0.88016745160 0.11983254840) *
3)      electricity< 0.5       784   334   0     (0.57397959184 0.42602040816)
6)      educationhigher>=0.5   76    4     0     (0.94736842105 0.05263157895) *
7)      educationhigher< 0.5   708   330   0     (0.53389830508 0.46610169492)
14)     savings>=0.5           51    7     0     (0.86274509804 0.13725490196) *
15)     savings< 0.5           657   323   0     (0.50837138508 0.49162861492)
30)     health>=0.5            554   253   0     (0.54332129964 0.45667870036)
60)     appliances>=0.5        264   101   0     (0.61742424242 0.38257575758) *
61)     appliances< 0.5        290   138   1     (0.47586206897 0.52413793103)
122)    read.newspaper>=0.5    95    41    0     (0.56842105263 0.43157894737) *
123)    read.newspaper< 0.5    195   84    1     (0.43076923077 0.56923076923) *
31)     health< 0.5            103   33    1     (0.32038834951 0.67961165049) *
* denotes terminal node
```

```
> printcp(fit.tree)

Classification tree:
rpart(formula = f, data = train, method = "class")

Variables actually used in tree construction:
appliances, educationhigher, electricity, health, read.newspaper, savings

Root node error: 563/2695 = 0.20890538

n= 2695

  CP          nsplit  rel error   xerror      xstd
1 0.01642984  0       1.00000000  1.00000000  0.037485216
2 0.01000000  6       0.88632327  0.94316163  0.036676519
```

7.4 Output from Random Forest

Call:

```
randomForest(formula = f, data = traintree, importance = TRUE, mtry =
round(sqrt(13)), ntree = 100, nodesize = 1)
```

Type of random forest: classification

Number of trees: 100

No. of variables tried at each split: 4

OOB estimate of error rate: 18.81%

Confusion matrix:

	0	1	class.error
0	2002	130	0.06097560976
1	377	186	0.66962699822

7.5 Data used in 1997

Table 4: Poverty descriptors in 1997

Dimension	Variable name	Values	Frequency
Human capital	Education	grade school	3205
		higher education	1975
		unschooled	1208
	Health	healthy	5571
		unhealthy	831
	Insurance	yes	791
		no	5597
Physical capital	Surface water	yes	978
		no	5410
	Tap water	yes	1007
		no	5381
	Creek toilet	yes	1630
		no	4758
	Own toilet	yes	1993
		no	4395
	Running sewage	yes	2866
		no	3522
	Garbage collector	yes	1435
		no	4953
	Vehicles	yes	2015
		no	4373
Financial assets	Savings	yes	1477
		no	4911

Table 5: Poverty predictors

Dimension	Variable name	Levels	Frequency
Human capital	Education - elementary	1-yes	2300
		0-no	2193
	Education - higher	1-yes	1208
		0-no	3285
	Healthy	1-yes	3693
		0-no	800
	Can write letter in Bahasa	1-yes	2970
		0-no	1523
	Can read newspaper in Bahasa	1-yes	3043
		0-no	1450
Physical capital	Has electricity	1-yes	3178
		0-no	1315
	Has household appliances	1-yes	3215
		0-no	1278
	Can buy meat	1-yes	898
		0-no	3595
Financial assets	Savings	1- yes	1080
		0-no	3413
	Employed	1-yes	3794
		0-no	699
	Unemployed	1-yes	411
		0-no	4082
Household shocks	Experienced hardship	1-yes	1323
		0-no	3170

7.6 Output from MCA in 1997

Call: `MCA(X = mca97[, c(3:11)], graph = FALSE, method = "Burt")`

Eigenvalues

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6
Variance	0.117	0.015	0.014	0.011	0.010	0.008
% of var.	61.819	7.810	7.162	6.011	5.395	4.435
Cumulative % of var.	61.819	69.629	76.792	82.803	88.198	92.633

	Dim.7	Dim.8	Dim.9	Dim.10
Variance	0.005	0.004	0.004	0.002
% of var.	2.422	2.021	1.860	1.065
Cumulative % of var.	95.055	97.076	98.935	100.000

Individuals (the 10 first)

	Dim.1	ctr	cos2	Dim.2	ctr	cos2	Dim.3	ctr	cos2
1	-0.696	0.023	0.329	-0.448	0.027	0.136	-0.065	0.001	0.003
4	-0.159	0.001	0.043	-0.053	0.000	0.005	-0.393	0.022	0.264
7	-0.985	0.046	0.470	0.040	0.000	0.001	0.304	0.013	0.045
10	-0.468	0.010	0.138	-0.556	0.041	0.196	0.369	0.019	0.086
13	-0.923	0.040	0.602	-0.529	0.037	0.198	0.003	0.000	0.000
16	-0.696	0.023	0.329	-0.448	0.027	0.136	-0.065	0.001	0.003
19	-0.923	0.040	0.602	-0.529	0.037	0.198	0.003	0.000	0.000
22	-0.493	0.012	0.142	-0.061	0.000	0.002	0.504	0.035	0.149
25	-0.772	0.028	0.312	-0.059	0.000	0.002	0.541	0.041	0.153
28	-1.061	0.053	0.449	0.429	0.025	0.073	0.910	0.116	0.331

Categories (the 10 first)

	Dim.1	ctr	cos2	v.test	Dim.2	ctr	cos2	v.test
grade school	-0.195	1.812	0.285	-15.282	-0.032	0.373	0.007	-2.466
higher edu	0.506	8.154	0.703	28.139	-0.150	5.671	0.062	-8.341
unschooled	-0.429	2.934	0.280	-15.092	0.394	19.544	0.235	13.845
surface wtr 0	0.109	0.968	0.488	21.368	0.059	2.303	0.147	11.713
surface wtr 1	-0.683	6.094	0.488	-21.368	-0.374	14.494	0.147	-11.713
tap wtr 0	-0.365	6.622	0.757	-29.875	-0.090	3.171	0.046	-7.349
tap wtr 1	0.396	7.184	0.757	29.875	0.098	3.441	0.046	7.349
creek tlt 0	0.261	4.527	0.780	30.991	-0.001	0.000	0.000	-0.092
creek tlt 1	-0.596	10.329	0.780	-30.991	0.002	0.001	0.000	0.092
own tlt 0	-0.376	7.576	0.841	-33.447	0.005	0.010	0.000	0.423

	Dim.3	ctr	cos2	v.test
grade school	-0.257	27.165	0.496	-20.142
higher edu	0.196	10.539	0.105	10.889
unschooled	0.376	19.430	0.215	13.220
surface wtr 0	-0.042	1.240	0.072	-8.229
surface wtr 1	0.263	7.801	0.072	8.229
tap wtr 0	0.060	1.556	0.021	4.929
tap wtr 1	-0.065	1.688	0.021	-4.929
creek tlt 0	-0.012	0.079	0.002	-1.394
creek tlt 1	0.027	0.180	0.002	1.394
own tlt 0	-0.005	0.014	0.000	-0.487

Categorical variables (eta2)

	Dim.1	Dim.2	Dim.3
education	0.396	0.279	0.598
surface wtr	0.217	0.183	0.095
tap wtr	0.424	0.072	0.034
creek tlt	0.457	0.000	0.003
own tlt	0.532	0.000	0.000
run sweage	0.351	0.016	0.011
garbage col	0.427	0.012	0.000
health	0.013	0.389	0.105
insurance	0.257	0.139	0.201