# Life Insurance – Age of Death Calculator using Machine Learning

Big data is becoming more and more freely available. However, big data sources are rarely used to its full potential. A vision and good data analysis is necessary to generate a message that benefits society.

After inspecting the raw mortality data of 2012, it became apparent that this data source has an abundance of information. The trick is to transform the data in a way that will be beneficial to a specified group of people. After brainstorming, I decided that the best use of the data would be to use the data source as a training dataset to train a machine learning model, with the goal of predicting at what age a person is most likely of dying, given certain inputs. It was decided that the final product should be used by life insurance consultants in South Africa, so that they can determine whether a specific person would be worth insuring - in other words, whether a person will live long enough to contribute a sufficient amount to the life insurance company, before dying.

Each variable in the data source is represented by a specific code. Because I knew what I wanted my end product to look like, I knew that these numerical codes would have to be replaced by the correct variable names if possible. For this reason, I decided that my first step would be to import the PDF document, containing the descriptions of the codes, so that I can later convert the codes to descriptions and use it as inputs for my final product.

I identified a few variables in the data source that I thought would have an impact, yet not obvious, on the age of death. I then filtered the data source for these specific variables. The variables that I decided on are gender, smoking status, residential province, marital status, education, occupation and illness.

The dataset had to be cleaned and filtered before the machine learning training could take place. After eliminating all unknown variable codes and factorising the variables where necessary, the training was ready to begin.

Because of the speed of the final product has to be accounted for, it was decided that the training dataset would only consist of 2500 rows. I made sure that every possible combination of variables would be present in the training dataset.

I used the ‘treebag’ machine learning algorithm, present in the ‘caret’ package to train the data. In the trainControl function, I set the class probabilities to true, so that a probability would be generated for each possible age of death. The probabilities would later be summed to generate final probabilities for defined age categories. It makes more sense to have an overall probability distribution, opposed to one result giving the probable age of death (even though the distribution might be very evenly spread).

I then loaded the trained model and necessary variables into a new project to develop a Shiny application. The Shiny app consists of two general functions, the UI and server functions. In the UI function I defined all my inputs, and in the server function the probability distribution graph is generated.

For the inputs of the application, I decided that it should be quick and easy to use by any type of consultant. For that reason, I implemented images as buttons to make the selection of certain inputs easier. Most of the buttons were fairly straightforward to program. The provincial map, for selecting province of residence was programmed by using the mouse click coordinates to determine on which province the user clicked. The inputs with more options were simply placed in dropdown lists, so that the app would not be too cluttered.

When changing the input, the server function would run and a new output graph would be generated. The machine learning trained model is used to generate the data frame used for the graph. The graph shows the probability distribution of age of death. The first most likely age category of death is highlighted in red to make the result easy to understand. To make it even easier for the user, a narrative is generated at the bottom of the screen explaining the inputs selected and the age range the person is most likely to die in. Unfortunately, while importing the PDF document, some of the rows jumped around resulting in some of the illness codes not being aligned to their descriptions. For this reason it was decided to keep the illness codes as input, and provide an html link in the narrative to the document where the definitions of the illness codes are saved.

I believe that the Shiny app is very practical, and easy to understand. It is not necessary to over-complicate results. The user should be able to understand what is going on in the visualisation. Simplicity is key.