Final Paper
Neural Net - The ultimate Answer to all Machine Learning Problems? Comparing different ML Approaches on Data on Voluntary Action

Larissa Haas
Mannheim University
Advanced Quantitative Methods
FSS 2017
MatrNr: 1417669
larhaas@mail.uni-mannheim.de

Abstract

There are people who participate in voluntary organizations, work for clubs and churches, and there are people who don't. Various researchers try to understand where this different behavior comes from and which characteristics foster or hinder the opportunity to participate. Since this distinct question is a classical yes or no question, published studies often come up with Logistic Regressions to investigate this relationship. However, the field of Machine Learning offers a broad range of alternative approaches which can handle binary data for prediction and analysis.

The goal of this paper is to have a look at data from the European Social Survey (ESS) and to use selected Machine Learning options for binary classification, such as Random Forests, SVMs (Support Vector Machines), Neural Nets, as well as the classical Logistic Regression, and to compare the models based on their predictive power on voluntary action. Based on this comparison and the best resulting approach, it should be possible to state also the predictive importance of the variables.

Contents

1 Introduction

The buzzwords "Machine Learning", "Neural Networks" and "Artifical Intelligence" hover through the air wherever people are working with large sets of data, also called Big Data. Such approaches are already in use for friendship recommendations on Facebook, for improving the quality of your smartphone pictures or for reducing waste by regulating product orders of supermarkets. They improve predictions wherever there is enough data to "learn" from. But this trend just starts to be relevant for other scientific fields, for example in political science. Even though there are tons of records where people were asked what they think, how they behave, what they expect, only one set of machine learning (ML) algorithms seems to be used: the regression.

The goal of this paper is to broaden the view on ML approaches existing beyond regression, for example, Random Forests, Support Vector Machines, and Neural Nets, and to compare these algorithms in terms of their predictive power on a genuine political science topic: participation in voluntary organizations. Therefore I will have a look at data from the European Social Survey (ESS) from round 1 and 6 because they both include a battery of survey questions about voluntary participation. Based on the theoretical foundations extracted in chapter 1 and the theoretical explanation of different ML approaches in chapter 2, I will select and prepare variables (as described in chapter 3) and feed them into ML algorithms. The results and their implications will be interpreted in chapter 4. In the end, advantages (and disadvantages) of the use of machine learning approaches in the field of political science should be clear as well as why it should be worth for a political scientist to have a look at these approaches.

2 Brief Theoretical Overview

In the broad field of political science, there are subfields with concepts that can be measured quite well, and there are others, like social participation. For a long time, this subfield was only viewed as an attachment for analyzing political behavior, not realizing that this participation is a distinct field of study on its own (Gabriel and Völkl, 2008, p. 269). With the establishment of the term "civil society" the literature begins to work on both fields of social and political participation. As van Deth states, "the distinction between social and political involvement of citizens does [...] rest [...] on the primary goals of the groups involved." (van Deth, 1997, p. 3). When we have a look at voluntary organizations that are part of the social participation sphere and of the civil society at the same moment, we may assume similar mechanisms as for political participation, only assuming different goals.

To localize the term "activity in voluntary associations" more clearly, we can look at Dekkers and van den Broeks definition, which describes

> "*voluntariness* as the guiding principle of civil society and *associations* as its dominant collective actors. A prerequisite for partaking in civil society is *commitment*: the willingness to bind oneself to a common course and to take responsibilities" (original emphasis) (Dekker and van den Broek, 1998, p. 13)

This willingness was questioned in parts of the literature, especially in the American political science sphere, where Putnam stated that Americans were "Bowling alone" instead of meeting in sports clubs as they used to have in the past. For Europe on the other hand, no evidence for such behavior could be found (Gabriel and Völkl, 2008, p. 279), (Dekker and van den Broek, 1998, p. 35).

Sports clubs, churches, interest groups, etc. (see (van Deth, 1997, p. 1) for a more detailed list) stand in the focus of the literature, because on the one hand these associations fill gaps where the state fails to help (e.g. food sharing, charity organizations), on the other hand the participating citizens learn values and ways of behavior that are important for the society in total (for more examples see (van Deth and Maloney, 2015, p. 826)). Some scholars even see associations as "schools of democracy", where "they can promote positive feelings towards other social and political institutions [...], towards other individuals [...], and towards oneself as a politically capable actor" (Morales and Geurts, 2007, p. 135). It is not quite clear, though, if the domain of active citizens has to be an officially organized group of people or if work for informal groups of citizens counts as "voluntary activity", too (Gabriel and Völkl, 2008, p. 270).

As many different notions of the concept "voluntary association" appear in the literature, as many influence factors we see, that have an impact on the willingness to participate. From narrowly defined lists of factors to broadly defined concept groups (e.g. (Zukin et al., 2006, p. 124); (Badescu and Neller, 2007, p. 159)), there is one model that seems to concentrate all possible influences in three simple questions: Are they able to participate? Do they want to participate? Did somebody ask them to participate? (Dalton, 2014, p. 64). This split in "resources", "engagement" and "recruitment" (Verba et al., 1995, p. 269) results in the so-called "civic voluntarism model", which defines participation especially in terms of individual influence factors what seems to be more appropriate in our individualized societies than an analysis in terms of social class or other social groups (Gabriel and Völkl, 2008, p. 288).

Even if the participation itself and the positive results of participation are so highly relevant for democracies and societies, the type and frequency of this social participation are measured not quite well (van Deth and Maloney, 2015, p. 831). Most data that is available comes from surveys, where they ask about participation and associational involvement. But every country has an own understanding of "associations" and "involvement" or "participation", sometimes different lists of example organizations where provided, or the question wording was not clear enough (van Deth and Maloney, 2015, p. 832). Additionally, people often don't recognize their social participation when they are asked within a survey, because they don't know how the concept is defined and that, for example, their activity in food sharing would count as well as being active as a youth coach at the soccer club. These factors, most often, lead to an underestimation of participation, but this "is not constant across countries, since some nations [...] show a higher proportion of citizens who are involved in associations without being members" (Morales and Geurts, 2007, p. 137). In addition, in most of the European countries there is no organizational register (or it is not up to date), and the associations do not count how many people are organized within their structure.

This lack of "good" data leads to the problem that we must work very effectively with the data sets we have at hand. And as regression seems to be a multi-purpose tool in empirical political science, it's time to look at this data from a different point of view.

3 Explanation of Approaches

In the following section, I want to briefly describe the four different methods that I will use later in the comparison. First, these methods are all together supervised algorithms. Supervised (in contrast to unsupervised) means, that we give the "right" solution with our data. Besides the known Logistic Regression, I want to introduce the algorithms "Random Forest", "Support Vector Machines" and "Neural Nets". Of course, this section can't serve the function of a total and exhaustive explanation of all details of these approaches, there are other very good books and articles which do this better and deeper as I ever could do it in this paper. The following section should rather give hints on the basic mechanisms at work, what to be aware of before using these approaches, what to expect and which advantages and disadvantages we may expect from the resulting models.

3.1 Logistic Regression

The first approach for predicting voluntary action and analyzing the factors that lead to a participation would be the "classical" Logistic Regression. There are various articles in the respective literature (Badescu and Neller, 2007; Dekker and van den Broek, 1998; Armingeon,2007) using this Logistic Regression on this topic, so it is known and widely accepted. In this case, I will use the Logistic Regression specified as follows:

Stochastic Component:

$$\begin{aligned} Y_i \sim Y_{Bern}(Y_i|\pi_i) &= \pi_i^{y_i}(1-\pi_i)^{1-y_i} \\ &= \begin{cases} \pi_i & \text{for } y_i = 1 \\ 1-\pi_i & \text{for } y_i = 0 \end{cases} \end{aligned}$$

Systematic Component:

$$\begin{aligned} Pr(Y_i = 1) &= \pi_i \\ &= g(X_i\beta) \end{aligned}$$

Where g() is the cumulative standard logistic function, which leads me to the according log-likelihood function, which is optimized in respect to the beta coefficients.

$$\begin{aligned} Pr(Y_i = 1) &= \pi_i \\ &= \Lambda(\beta_0 + \beta_1 x_i) \\ &= \frac{1}{1+e^{-X_i\beta}} \end{aligned}$$

$$
\begin{aligned}
lnL(\pi|y) = &\sum_{i=1}^{n}(y_i * ln(\pi_i) \\
&+ (1-y_i) * ln(1-\pi_i)) \\
lnL(\beta|y) = &\sum_{i=1}^{n}(y_i * ln(\frac{1}{1+e^{-X_i\beta}}) \\
&+ (1-y_i) * ln(1-\frac{1}{1+e^{-X_i\beta}}))
\end{aligned}
$$

The results of this function are optimized betas, which are coefficients used to calculate the predictions of my model. The advantage of this method is that I get significance values together with the coefficients, so I can see how certain or uncertain I am about the effects I measured. Additionally, I can interpret the sign of the coefficients, which is the direction of the effect. The value of the coefficients is harder to interpret, mathematically spoken they express the log odds that with an increase in this variable the result will be a success (i.e. resulting in the 1-class of my 2-class-scenario). Although the results seem to be interpretable, too large tables with different models, coefficients, standard errors and stars now and then appear to be daunting for the average reader. Therefore, authors try to "simplify" their regression results which lead to confusion, because now numbers are replaced with pluses (Gabriel and Völkl, 2008), are printed in bold (Zukin et al., 2006) or represented with differently styled arrows (Dalton, 2014). This may enhance the understandability in the short term, but it also leads to ambiguity and decreasing comparability across the studies.

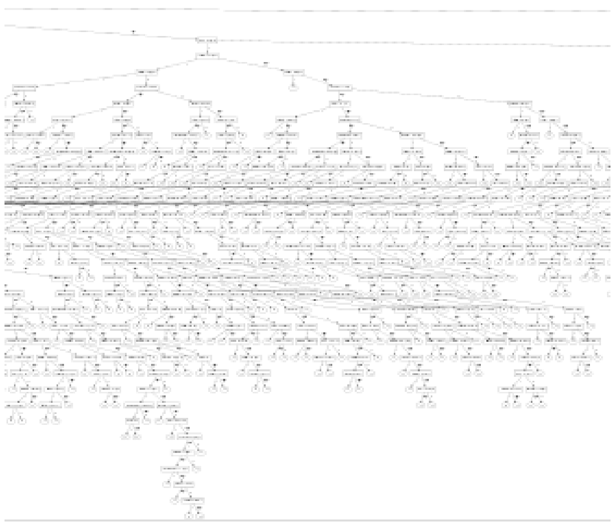
Another big disadvantage of Logistic Regression is the absent possibility to specify the model besides the data. For Logistic Regression there exists only one model for one data set (the various possibilities of variable selection and interaction terms not included), the only adjustment is the assignment of the threshold for deciding on 1 and 0 at the end. In the following subsections, I will show that this lacks flexibility in comparison to the other approaches.

3.2 Random Forest

The approach called "Random Forest" is, as the name already shows it, an ensemble of decision trees. A decision tree grows from top to bottom. It "guides" the data through its branches and splits the branches by specific criteria to make the split data purer (in the distribution of target classes). When applied to regression (rather than to classification tasks), a so-called "Regression Tree" is built. In contrast to the classical decision tree, here the leaves do not decide on a predicted class but on a regression function applied to the data that came through the according branches. A Random Forest now first samples from records and variables and feeds these samples into various regression trees. The results are then combined to the "best" possible result over all trees (Dangeti, 2017, p. 111). This method is called "Ensemble" and it inherits characteristics of the "Bagging" approach.

There are two crucial options to specify when building a Random Forest: the number of trees and the maximum depth of each tree. The depth describes the number of splits the tree can make from trunk to leaves. The more splits I allow, the more fine-grained is the result at the leaves, but also the more decisions are to make.

Figure 1: 1% of a Random Forest Tree (1 out of 400)



An advantage of Random Forests is their characteristic to combine the best of Logistic Regression and decision trees:

> "Logistic regression has very high bias and low variance technique; on the other hand, decision trees have high variance and low bias, which makes decision trees unstable. By averaging decision trees, we will minimize the variance

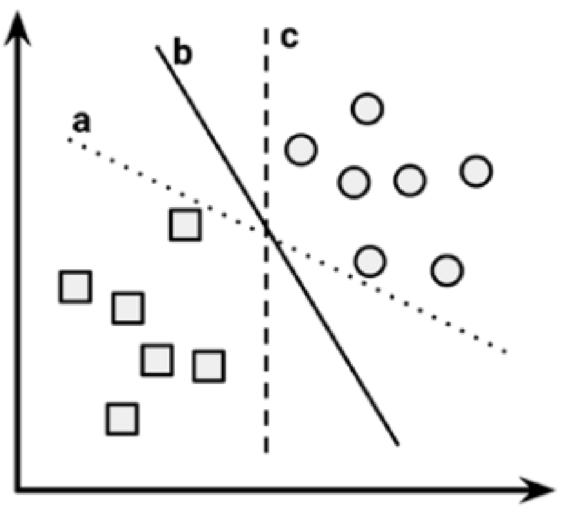
component the of model, which makes approximate nearest to an ideal model." (Dangeti, 2017, p. 111)

Unfortunately, there can no significances of variables be assigned, and even if decision trees themselves are highly intuitive because they can be plotted and interpreted by humans, Random Forests lack interpretability because of their shire size and depth (Dangeti, 2017, p. 113). Just to give an impression of the dimensions I am talking about: Figure 1 shows exact 1 percent of one tree out of 400 trees within a Random Forest (with depth maximum at 30).

3.3 Support Vector Machines

We can see a Support Vector Machine (SVM) as the best possible line (or hyperplane in more than two dimensions) to separate data. The goal is to make this separated data on both sides of the line as homogeneous as possible (Lantz, 2015, p. 239).

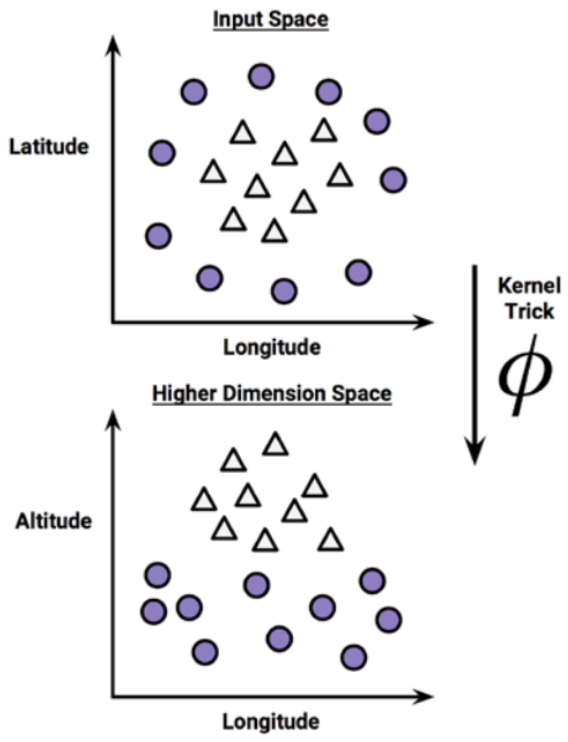
Figure 2: The basic principle how a Support Vector Machine works (Lantz, 2015, p. 241)



To understand the admittedly complex mechanism of SVMs, I will briefly introduce three of the core concepts:

- **Maximum margin**: In figure 2, we can see three lines, equally able to separate between circles and squares. But only line b has the largest distance to the points and because "it is likely that the line that leads to the greatest separation will generalize the best to the future data" (Lantz, 2015, p. 241), it is considered as the best separation.
- **Support vectors**: The lines or hyperplanes are not calculated between all available data points, this would lead to an exceptionally high computational effort. Only the data points closest to the hyperplane are taken into consideration to define the separation. "This is a key feature of SVMs; the support vectors provide a very compact way to store a classification model, even if the number of features is extremely large." (Lantz, 2015, p. 242)

Figure 3: How the Kernel Trick transforms the view on the Data (Lantz, 2015, p. 246)



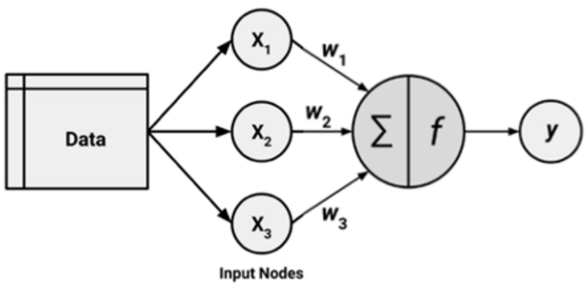
- **Kernel trick**: In practice, most data points are not separable by a line or a hyperplane, SVMs use various computational tricks to modify the data. They assume that a separation of classes is possible when the data is transformed to another dimension. Think of a map with an island pictured on it (Figure 3). There is no line that can separate island from water. But when you add a third dimension and examine the island again, you see that the water level cuts island from ocean. Kernel functions can be of different nature, there exist for example linear, polynomial or sigmoid kernels (Lantz, 2015, p. 247f.).

These flexibilities in building a model according to your data make SVMs especially powerful tools. Often, they achieve the best results in predicting when compared with other methods, assumed they get enough data to train on. Disadvantages are their complex computational effort caused by the high dimensionality of data as well as the kernel transformations. Additionally, their resulting models cannot be interpreted by humans. I don't get any significances, coefficients or importance distributions for the variables, so I cannot conclude anything substantial from the model.

3.4 Neural Net

A Neural Net is a machine learning approach that can be used for both, classification and regression problems. Just like a biological brain, it connects inputs with "signals" or weights via hidden nodes to outputs. Within these nodes, activation functions "decide" which information passes the node, and which gets "forgotten". The output is (just like for every other considered approach) the prediction whether a person is active in voluntary organizations or not.

Figure 4: Parts of a Neural Network (Lantz, 2015, p. 222ff.)



As already mentioned, each node consists of an activation function, which is fed with some input (the values in the first layer or some transmitted "signals" in the hidden layers) and generates some output. These nodes, or neurons as they are called, can be placed in different patterns or structures within the Neural Net. Basically, I am estimating various regressions in a hierarchized way:

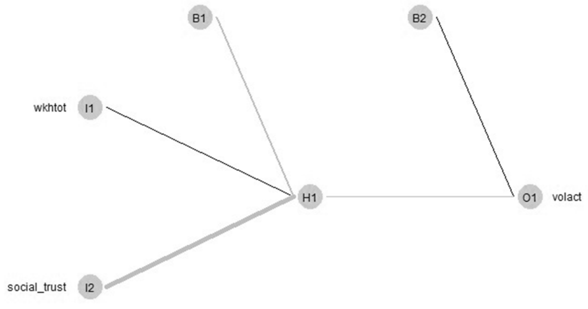
$$
\begin{aligned}
Pr(Y_i = 1) &= \pi_i \\
&= \frac{1}{1 + e^{-X_i\beta}} \\
&= logit(X_i\beta) \\
&= logit(linear(X_i\beta))
\end{aligned}
$$

This formula shows how a linear regression is encapsulated inside of the Logistic Regression function. I could take a step further and add another layer, the result would look like this:

$$
\begin{aligned}
Pr(Y_i = 1) &= \pi_i \\
&= logit(linear(logit(linear(X_i))))
\end{aligned}
$$

In fact, this is a simple "single hidden layer" network. If I had two independent variables feeding in the network, I could visualize the dependencies like in Figure 5. The figure shows also the error weights (marked here with "B").

Figure 5: "Single hidden layer" network



Of course, I don't want to build a Neural Net because of one single hidden layer, but it takes some of the "black box" characteristics away. In fact, I am calculating somehow chained functions (Logistic Regression functions for the case above), until only one neuron is left and gives us the result.

There are two crucial parameters that form the basic characteristic of a Neural Network. I will briefly explain what these important parameters are and what they basically do when I use them within our models.

- **Activation function**: As already mentioned above, the activation function decides which neurons are activated and should send signals through their connections. One typical function we already know is the sigmoid function, which is a special case of the logistic function. Other widely used activation functions are the hyperbolic tangent or the Rectifier (or Relu for Rectified Linear Unit (Dangeti, 2017, p. 243)), which can be mathematically described as $f(x) = max(0, x)$.
- **Network architecture**: There are quite a lot ways of structuring a Neural Net. The only

fixed points are the input nodes for each variable and the output node for the prediction results. In between everything is possible: several layers of hidden nodes, deeper (more layers), or wider (more nodes within one layer), fully connected or partly connected, with dropout ratios between the nodes and so on. There is no rule which combination of all these factors fits best to your data, but the structure is crucial for building the "best" model instead of building a "good" model.

Besides these two, Neural Networks have many more parameters, that can be applied and tuned. For example the H2O deep learning function, I will use for my Neural Net models later on, has in total 90 options (mandatory and optional) to alter.[1] This results in the apparently unsolvable task to test all possible combinations of parameters to find the best working solution. Usually "grid searches" are used to complete this task. They run through the specified options systematically and rank the resulting models based on some criteria. Unfortunately, grid searches are computationally highly intensive so that I cannot take this method into account for my recorded solutions.

4 Data Preparation

4.1 Data Set and Chosen Variables

For this paper, I worked with data from the European Social Survey (short ESS). I used data from two rounds, namely round 1 and 6 (ESS, 2002; ESS, 2012). These two rounds include both questions about activity in voluntary organizations. In ESS round 1 the question was asked whether the respondent did the following this in the past 12 months:

- Is he/she a member of an organization?
- Did he/she participate within the organization?
- Did he/she donate money to a organization?
- Did he/she voluntarily work for a organization? [2]

Possible organizations were listed, where these actions could have taken place, for example culture, sports or social clubs. I recoded these variables into one binary variable, being 1 if the respondent answered "yes" to the second or the forth question. Simply donating money or being member of an organization is not enough to count as "being active".

For ESS round 6 the question was different, but captured the same meaning. Again it was asked about the activity for voluntary organizations within the last 12 months, the options to select ranged from "at least once a week" to "less often" than once every six months [3]. I recoded this variable into a binary variable, where all positive answers were transformed into a 1 and "never" into a 0. We have to keep in mind that these two questions are principally different, but I decided to go with data from both years as it is known that ML algorithms need a lot of training data to work well. By using both rounds as one data set I am able to double the number of observations I can use to feed in the algorithms. To make sure that this is not a disadvantage for the predicting, I will also calculate models with the two rounds separated in two data sets.

The independent variables were selected based on the "civic voluntarism model" by (Verba et al., 1995) and were guided by the work of (Badescu and Neller, 2007). A detailed explanation of the variables and comments on the pre-processing can be seen in the appendix 2. Here I just want to list the variables based on their belonging to the three levels of "civic voluntarism":

- **Resources**: gender, age, country, years in education, living area, marital status, children, being a houseperson, weekly work hours
- **Engagement**: tolerance, self realization, solidarity, political interest, trust in executive, trust in legislative, trust in European Parliament, satisfaction with democracy, political TV consumption, total TV consumption
- **Recruitment**: attending church, meet social contacts

By merging the two rounds together I added a binary variable showing to which round the record

[1] H2O Documentation: `http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/deep-learning.html`

[2] ESS1 Main Questionnaire: `http://www.europeansocialsurvey.org/docs/round1/fieldwork/source/ESS1_source_main_questionnaire.pdf`

[3] ESS6 Main Questionnaire: `http://www.europeansocialsurvey.org/docs/round6/fieldwork/source/ESS6_source_main_questionnaire.pdf`

belongs (this should also work against the error based on taking the two rounds as one data set).

The ESS recommends to include population and design weight variables in the model [4] to make the estimations representative for the population of one country or the countries within the EU. As it is difficult to include these weight variables to the same impact in all models, I will not include them in the analysis. As my goal is in the first instance to evaluate the models on their quality in prediction, it would be a subsequent step to make the results representative for the population. Therefore we have to keep in mind (especially for the Logistic Regression, as I don't get any interpretable results from the other models), that the results from the different approaches will be not representative neither for one country nor for the countries compared within the EU.

4.2 Pre-Processing

Missing Values

Unfortunately, the data was created with a lot of empty spaces in the matrix. Therefore, I used the default approach, which can also be applied within each algorithm itself, and replaced the missing spaces with the average value of that respective variable, or the most frequent value when it was a binary variable. This may lead to an underestimation of the variance in the variable, so this must be remembered for the Logistic Regression when we interpret the significance of the calculated coefficients.

Normalization

Especially Neural Networks are very sensitive to data measured on different scales. The activation functions are sensitive around +/- 1, so it is useful to re-scale the included variables.

> ”By restricting the range of input values, the activation function will have action across the entire range, preventing large-valued features such as household income from dominating small-valued features such as the number of children in the household. A side benefit is that the model may also be faster to train since the algorithm can iterate more quickly through the actionable range of input values.” (Lantz, 2015, p. 225)

[4]Weighting European Social Survey Data: `http://www.europeansocialsurvey.org/docs/methodology/ESS_weighting_data_1.pdf`

I therefore scaled the data (where it was not categorical or binomial) to a mean of 0 and a standard deviation of 1. These re-scaled values will be used for all the approaches, again to make the results comparable. The other approaches should not be influenced negatively by normalized values, on the contrary, this step has the tendency to make the prediction better.

There is one problem with normalizing the variable values: We have to keep it in mind when we want to interpret the Logistic Regression coefficients. I will remember the original distribution of the data, such as mean, highest and lowest values and so on, so I am able to interprete the scaled results with respect to the original ones.

For the categorical country variable I created dummy variables, which are included in an extra model to test their influence (they easily doubled the number of my independent variables, so I had to include them separately).

Splitting in Training and Test Data Sets

To avoid that an algorithm learns instances of the data it should predict later, I clearly separated my data into a training and a test data set. I randomly selected n/10 indices from my data set and excluded them as ”test data” from the rest, the ”training data”. The algorithms are then able to learn on the training data and to apply their ”knowledge” about the models on the unseen test data set. This separation is very important as this is the only way to proof the validity of the model and to prevent over-fitting on the training data (which would be the case if the algorithm knows the data ”by heart” and performs way too good but is not able to deal with unseen data).

Balancing

Because the not voluntary active citizens where nearly twice as much as the participating citizenship (65% : 35%), I created an additional data set for training, which had equally numbered target variable levels. Sometimes this facilitates the work of the algorithm. I used oversampling, so records of the underrepresented category of voluntary activity were duplicated. These artificially added voluntary active citizens to should equalize the relationship between the two categories.

5 Comparison of Results

5.1 Evaluation Metrics

To compare the different models and evaluate the best model for each approach, I used two metrics, each one capturing a different concept within the scoring process. I think the measurement **accuracy**, also known as “Percent Correctly Predicted” (PCP), can be captured most intuitively compared to the other two. You can just interpret it as the percent of correctly predicted cases. In mathematical terms it looks like this:

$$Accuracy = \frac{(tp + tn)}{(tp + fp + tn + fn)}$$

Where tp (= true positives), tn (= true negatives), fp (= false positives) and fn (= false negatives) are the cells of a confusion matrix (see table 1).

Table 1: Confusion Matrix

		truth	
		0	1
prediction	0	tn	fn
	1	fp	tp

Secondly, I will have a look at the **F1 measure**, which combines two famous measurements called recall and precision. Precision captures the error rate in the predictions, how often I am wrong when I predict a 1. Recall measures how many of the true cases I can predict correctly.

$$F1 = \frac{(2 * Precision * Recall)}{(Precision + Recall)}$$
$$Precision = \frac{tp}{(tp + fp)}$$
$$Recall = \frac{tp}{(tp + fn)}$$

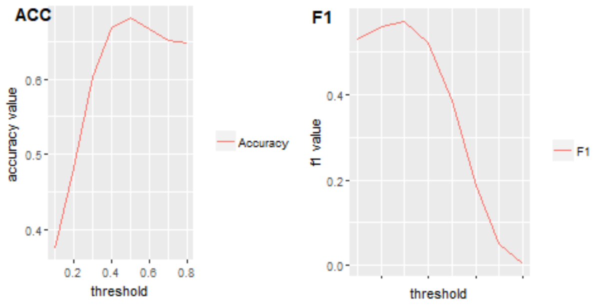
The two parts are weighted equally, which is the reason why it is called F1 (there are measurements like F0.5 and F2 existing, each weighing one of the components more). The algorithms of the Neural Net and Random Forest optimize according to F1, because it is regarded as the ”broadest” measurement. By focusing, for example, only on accuracy, I could easily be distracted because with a highly unbalanced dependent variable I gain a high accuracy measurement, even if I am just predicting the most often occurring class (and doing no machine learning at all).

In combination, these three measurements should give us a good notion of which model could be evaluated as the “best”.

5.2 Evaluation of Results

To evaluate the results from the four established approaches, we must keep our baseline in mind. The distribution of voluntarily participating citizens in the data is 35 percent against 65 percent not participating. This implies that with a model, that just predicts the modal category (i.e. the most often occurring category), I achieve an accuracy of 65 percent. The goal of the models is to top this baseline results. As already said, I will compare the results based on three different evaluation metrics. The given results in the following paragraphs are the “maximized” results, i.e. the reported metrics were observed with a threshold optimized especially for them.

Figure 6: Accuracy and F1 for Logistic Regression on different Thresholds



At first, the results from the **Logistic Regression** appeared pretty bad in comparison to the other used approaches. But then I realized that I did not optimize the threshold, where the decision was made which record should count as 1 or as 0. Figure 6 shows that it matters where I put the threshold. For an optimal model I would take a value inbetween the maximum of accuracy and F1.

When I now compare the Logistic Regression results to the other results in figure 7 and 8, the values show that it is not that far away from the others. In terms of F1, the other models score slightly better, but in terms of accuracy they are all equally good.

The best model for **Random Forest** was a forest with 400 trees and a depth of 30. It scores highest in both, accuracy and F1, at the basic model. All

Figure 7: Accuracy Comparison

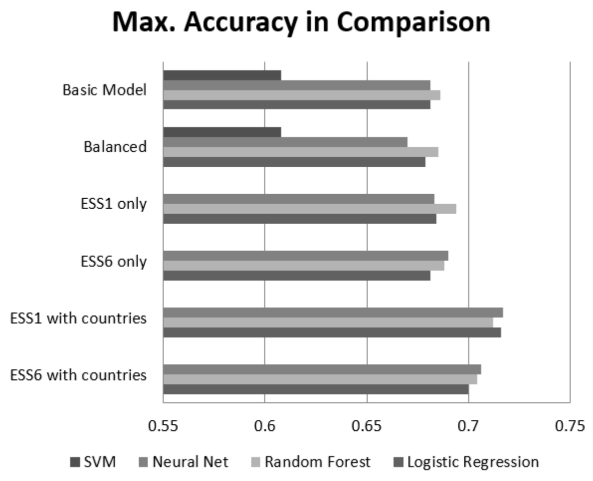


Figure 8: F1 Comparison

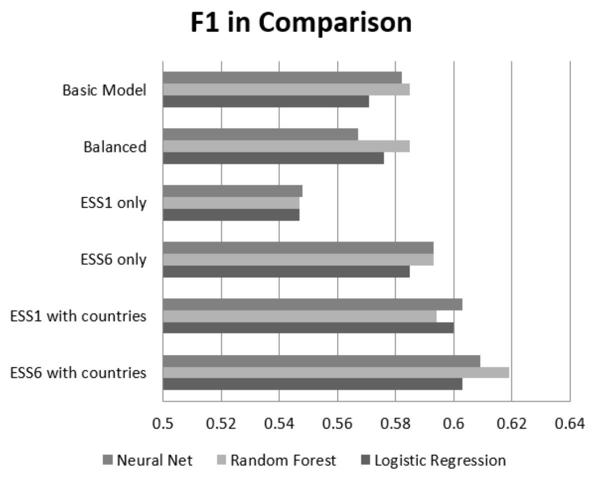
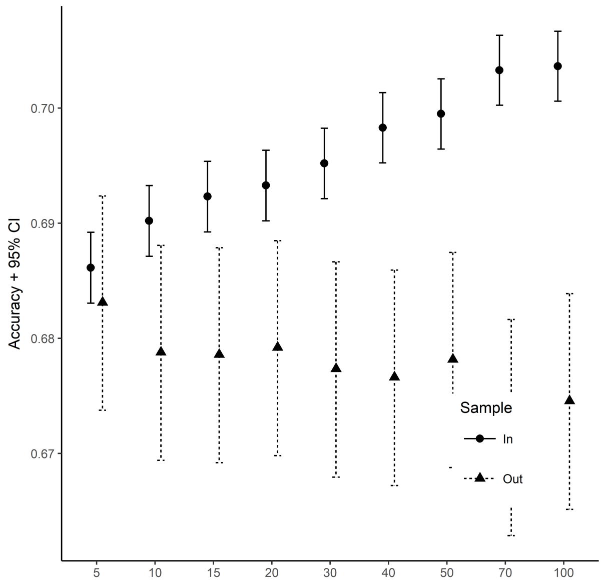


Figure 9: Parameter Tuning: Adding more Neurons to the Hidden Layer



the Random Forests I built were very stable and did not tend to over-fit, as the accuracy measured on in- and out-of-sample was equally high.

Tuning the **Neural Net** models was really tricky because no parameter change seemed to help. Figure 9 shows how adding more and more neurons to the hidden layer makes the in-sample accuracy better and better - but the out-of-sample accuracy actually decreases. Clearly, this figure was not created with the powerful H2O-algorithm, but it shows a crucial part of the story: Tuning the parameters did not help.

I could have done a grid search, but to do so I needed more (and longer) server power. The best result was achieved with a 3-layer network with 200 neurons in the first and the second layer and 100 neurons in the third layer. As activation function I used the ”rectifier” function. I also tried tuning the epoch number (usually the higher the better), but it didn't make any difference, so I stayed with 10, a relative low number compared to models in other domains. The results of this Neural Net (Basic Model) were not the highest, but also not the worst compared to the other approaches. The accuracy value is about 68 percent, which is higher than the 65 percent baseline, but only 3 percent.

I ran the **Support Vector Machine** models only on my R-studio server capacity, because they took too long to calculate locally on my laptop. The in-sample accuracy was pretty good, at 70 percent by first guess, but the out-of-sample accuracy was worse than the baseline. Unfortunately my server time ran up before I could test all data sets and do some extensive grid search, therefore some values in figure 7 and all in figure 8 are missing.

Besides the basic model, I mentioned in the sections above several aspects in the data which I wanted to control. The results are already pictured in the figures 7 and 8, but they will be also analyzed in the following paragraphs.

Does the (unequal) distribution of categories have an influence on the prediction?

As figure 7 shows, in terms of accuracy, the results of the models trained with a ”balanced” data set do not differ from the results from an ”unbalanced” data set. When we compare the F1 values in figure 8 we see a slight improvement for the ”balanced” data set for the Logistic Regression. For the Neural Net, on the other hand, the F1-value was reduced. This shows that balancing the data can improve your results, but it really depends on your data and your model.

Does the merging of data sets influence the prediction?

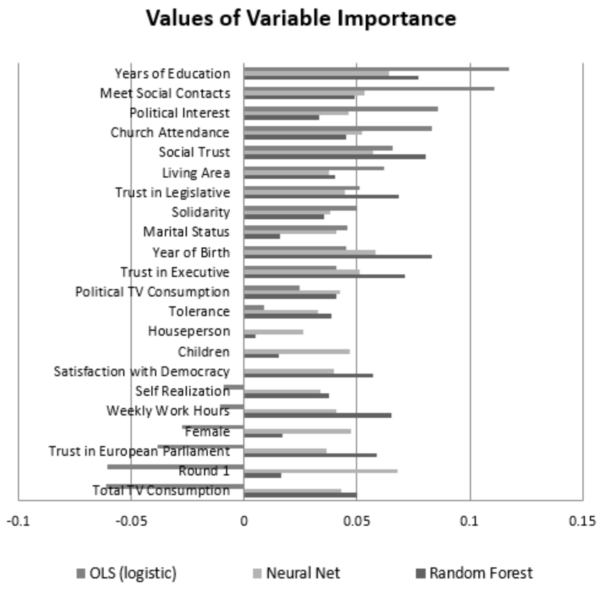
In terms of accuracy there are no major changes when I built models for the rounds separated in comparison to the merged data model. In terms of F1, ESS round 1 scores not over 0.55 (worse than for the merged data model), whereas ESS round 6 nearly reaches 0.60. In this case we can see the big difference between accuracy and F1. For accuracy the models look equally good, but not for F1. There has to be some potential for improvement.

Does the inclusion of the country variable can improve the results?

Yes, it does. All three observed models score higher when the country dummies are included (compared to the "ESS only" models). both for accuracy and F1. With this data I observed the overall best scoring models: For accuracy it is the Neural Net (on ESS1 data), for F1 it is the Random Forest (on ESS6 data).

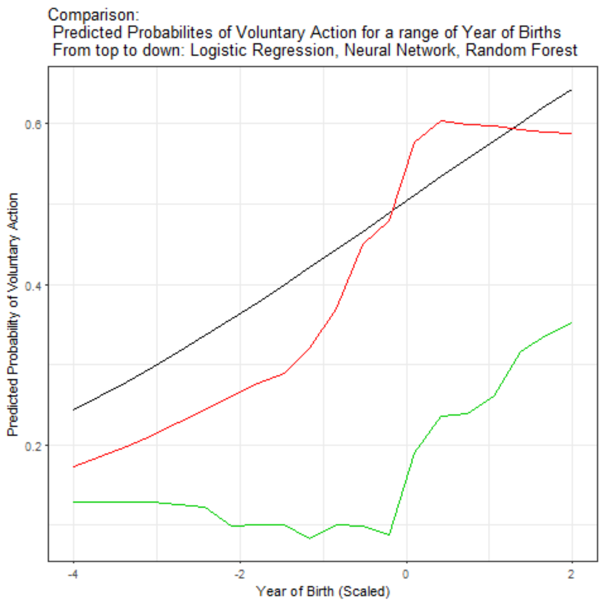
5.3 Evaluation of Variables

Figure 10: Variable Importance in Comparison



In the end, I want to compare the models concerning the variable importances. Maybe I can get a better image of the factors influencing voluntary activity, when I take all three basic models together. For the Logistic Regression, I summed up the absolute values of all significant (p ¿ 0.05) coefficients (to assign predictive "power" to them, as they are all scaled). Then I calculated relative importances [5] which are comparable to the importances I get from the H2O Random Forest and Neural Net algorithms. Table 10 shows these values compared to each other. In all three approaches together, there are in total three variables with an importance value over average: social trust, meet social contacts, and years in education. In general, the "social" variables tend to be more important than, for example, status characteristics as the marital status, the existence of children or being female. But, for the data set at hand, I cannot claim that the importances follow patterns described in the literature. There is no separation between "resource" and "engagement" variables, as some of them seem important while others don't. Only the "recruitment" part (with "meet social contacts" and "church attendance") is in complete regarded as important for the three approaches.

Figure 11: Predicted Probabilities of Voluntary Action for a range of Year of Births



One last comparison of the three approaches concerns the marginal effects. Which influence has one variable on the outcome, assuming the other variables do not change? Figure 11 shows clearly where the differences between Logistic Regression, Neural Networks and Random Forests are. The line of the Logistic Regression is a

[5] *relative* means: Their absolute values sum up to 1

straight monotone increasing line. The Neural Net predicts lower participation probabilities for smaller birth years (older respondents), the probabilities increase the younger the respondent gets until, roughly around 0, the curve starts to decrease. 0, the mean of the scaled birth-year-variable, stands for the birth year 1961, therefore the ”average aged” respondents are between 41 (ESS round 1) and 51 (ESS round 6) years old. The oldest person in the data is born 1893, while the youngest is born in 1998. For both, Neural Net and Random Forest this means that we start with a relatively high probability of voluntary activity when we are young. Then we go to work, have a family, and at some point the probability starts to decrease. For Neural Net slower and steadier, for Random Forest drastically and with a positive tendency towards the older age.

6 Conclusion

From the last figure I can conclude that Neural Nets and Random Forests can model different influences at different data values better and more precise than a simple Logistic Regression could do. Nevertheless, the prediction results of the Logistic Regression are not as bad as I expected them to be at the beginning. Despite the precise modelling potential of Neural Nets and Random Forests, these two models could not take advantage of their characteristics: No parameter tuning helped to improve the results.

Apart from that, Support Vector Machines could not convince at all because they needed more time on the server to run than a Neural Net on my laptop and could not achieve equally high results with default settings.

But these findings provide valuable insights: When I cannot improve my model, I have to go back to the data and start with the pre-processing again. We have seen that the inclusion of other variables can make our models work better than before. Possible additional strategies could be

- a better missing value replacement,
- automated parameter tuning and feature subset selection algorithms,
- feature engineering, or
- a different variable selection.

All in all, leaving the comfort zone of the ”known” regression could be helpful for every scientist working with an huge amount of data. Depending on the data at hand, other algorithms like Random Forests or Neural Nets could provide comparable and even better results, and are still interpretable at the same moment (see for example figure 11). There hasn’t to be any fear when working with these unusual approaches, of course they all have their own prerequisites and assumptions, but as long as you keep them in mind, with suitable data they can be very powerful tools.

7 Appendix

7.1 R Code

The code used for this paper was written with the help of the tutorial held by Marcel Neunhoeffer (provided by the chair of Quantitative Methods in the Social Sciences) to support the lecture Advanced Quantitative Methods, as well as with the provided R code by (Lantz, 2015; Cook, 2017; Dr. Wiley, 2016).

```r
######################################################################

# Local R-File
# Final Submission for AQM
# Larissa Haas

######################################################################

# Note: The plots and long running code snippets are
#      commented out. Please consider if it is really
#    necessary to run them, as this might take some while.

# Eventually, I do a lot of plotting in the code, but I can't
# show everything in my paper. I didn't want to delete it, either.
# I hope the code is nevertheless interesting.

######################################################################
# Setting up the Working Environment
######################################################################
library(neuralnet)
library(devtools)
library(stargazer)
library(RCurl)
library(jsonlite)
library(caret)
library(ggplot2)
library(e1071)
library(h2o)
library(statmod)
library(MASS)
library(corrplot)
library(data.table)
library(ROSE)
library(reshape2)
library(ggpubr)

F1 <- function(table){
   tp <- table[2,2]
   tn <- table[1,1]
   fp <- table[2,1]
   fn <- table[1,2]
   precision <- tp / (tp + fp)
   recall <- tp / (tp + fn)
   result <- (2 * precision * recall) / (precision + recall)
   return(result)
}

ACC <- function(table){
   tp <- table[2,2]
   tn <- table[1,1]
   fp <- table[2,1]
   fn <- table[1,2]
   acc <- (tn + tp) / (tn + tp + fn + fp)
   return(acc)
}

ll_logit <- function(theta, y, X) {
  beta <- theta[1:ncol(X)]
  mu <- X %*% beta
  p <- 1/(1 + exp(-mu))
  ll <- y * log(p) + (1 - y) * log(1 - p)
  ll <- sum(ll)
  return(ll)
}

###################################################################
# Loading and Pre-Processing the Data
###################################################################

ess1 <-
    read.table("C:/Users/laris/Desktop/MMDS/Semester3-FSS2018/Advanced-Quantitative-Methods/paper/
    header=TRUE, sep=",")

ess6 <-
    read.table("C:/Users/laris/Desktop/MMDS/Semester3-FSS2018/Advanced-Quantitative-Methods/paper/
    header=TRUE, sep=",")

ess1$round1 = 1
ess6$round1 = 0

data <- rbind(ess1, ess6)

dim(data)

data.complete <- data

for(i in c(4,5,6,10,11,12,13,14,15,16,17,18,19,20,21,22,23)){
  data.complete[is.na(data.complete[,i]), i] <- mean(data.complete[,i], na.rm =
      TRUE)
}

data.complete[is.na(data.complete[,3]), 3] <- 1
data.complete[is.na(data.complete[,7]), 7] <- 1
data.complete[is.na(data.complete[,8]), 8] <- 0
data.complete[is.na(data.complete[,9]), 9] <- 0

data.scaled <- cbind(data.complete[c(1, 3, 7, 8, 9, 25)], scale(data.complete[-c(1,
    2, 3, 7, 8, 9, 24, 25)]))

# This part can be used to write and to load
# the already pre-processed data

#write.csv(data.complete, file = "ess-complete.csv")
#write.csv(data.scaled, file = "ess-scaled.csv")

#data.scaled <- read.table("ess-scaled.csv", header=TRUE, sep=",")
#data.scaled <- data.scaled[,-1]

# Splitting in Train and Test Data

smp_size <- floor(0.9 * nrow(data.scaled))
set.seed(123)
train_ind <- sample(seq_len(nrow(data.scaled)), size = smp_size)
train <- data.scaled[train_ind,]
test <- data.scaled[-train_ind,]

# Balancing the Data

train.bal <- ovun.sample(volact ~ ., data = train, method = "over", N = 114264)$data

# Defining X and y for later use

train.X <- train[, -1]
train.y <- train[, 1]
train.y <- factor(train.y, levels = 0:1)
train.X.bal <- train[, -1]
train.y.bal <- train[, 1]
train.y.bal <- factor(train.y, levels = 0:1)
test.X <- test[, -1]
test.y <- test[, 1]
test.y <- factor(test.y, levels = 0:1)

##################################################################
# First Step into Neural Nets
##################################################################

# In this section I want to replicate the code we worked on
# in the tutorial. While doing this, necessary functions are
# defined. But most of the code is just "getting warm" wit NNs.

##################################################################

col <- test$volact
col[test$volact == 1] <- adjustcolor("orange", alpha = 0.3)
col[test$volact == 0] <- adjustcolor("blue", alpha = 0.3)

#plot(test$wkhtot, test$social_trust, col = col, pch = 19,
#    main = "True Relationship \n (scaled values)",
#    bty = "n", las = 1,
#    xlab = "Total Work Hours (per Week)", ylab = "Social Trust")

y <- test$volact
X <- cbind(1, test$wkhtot, test$social_trust)

startvals <- c(0, 0, 0)

res <- optim(par = startvals,fn = ll_logit, y = y, X = X,
           control = list(fnscale = -1),
           method = "BFGS"
           )

mu <- X %*% res$par

p <- 1/(1 + exp(-mu))

y_hat <- rbinom(nrow(p), 1, p)
col <- y_hat
col[y_hat == 1] <- adjustcolor("orange", alpha = 0.3)
col[y_hat == 0] <- adjustcolor("blue", alpha = 0.3)

#plot(test$wkhtot, test$social_trust, col = col, pch = 19,
#     main = "True Relationship",
#    bty = "n", las = 1,
#    xlab = "x1", ylab = "x2")

logit_pred <- table(y_hat, test$volact)
#logit_pred

ll_simple_nn <- function(theta, y, X){

 gamma <- theta[1:4]
 beta_neuron1 <- theta[5:7]
 beta_neuron2 <- theta[8:10]
 beta_neuron3 <- theta[11:13]

 mu_neuron1 <- X %*% beta_neuron1
 mu_neuron2 <- X %*% beta_neuron2
 mu_neuron3 <- X %*% beta_neuron3

 logitResponse <- function(mu) 1 / (1+exp(-mu))

 p_neuron1 <- logitResponse(mu_neuron1)
 p_neuron2 <- logitResponse(mu_neuron2)
 p_neuron3 <- logitResponse(mu_neuron3)

 Z <- cbind(1, p_neuron1, p_neuron2, p_neuron3)

```

```
  mu <- Z %*% gamma

  p <- logitResponse(mu)

  ll <- y * log(p) + (1 - y) * log(1 - p)

  ll <- sum(ll)
  return(ll)
}

# initial values
startvals <- rnorm(13)

ll_simple_nn(startvals, y, X)

# optimize
resNN <- optim(par = startvals, fn = ll_simple_nn, y = y, X = X,
          control = list(fnscale = -1),
          hessian = F,
          method = "BFGS"
          )

#resNN$par

gammaEst <- resNN$par[1:4]
beta_neuron1Est <- resNN$par[5:7]
beta_neuron2Est <- resNN$par[8:10]
beta_neuron3Est <- resNN$par[11:13]

mu_neuron1Est <- X %*% beta_neuron1Est
mu_neuron2Est <- X %*% beta_neuron2Est
mu_neuron3Est <- X %*% beta_neuron3Est

logitResponse <- function(mu) 1/(1+exp(-mu))

p_neuron1Est <- logitResponse(mu_neuron1Est)
p_neuron2Est <- logitResponse(mu_neuron2Est)
p_neuron3Est <- logitResponse(mu_neuron3Est)

Z <- cbind(1, p_neuron1Est, p_neuron2Est, p_neuron3Est )

mu <- Z %*% gammaEst

p <- logitResponse(mu)


y_hat <- rbinom(nrow(p),1,p)
col <- y_hat

col[y_hat == 1] <- adjustcolor("orange", alpha = 0.3)
col[y_hat == 0] <- adjustcolor("blue", alpha = 0.3)

#plot(X[, 2], X[, 3], col = col, pch = 19,
#    main = "Predicted Values from a Neural Net",
#    bty = "n", las = 1,
#    xlab = "x1", ylab = "x2")

nn_pred <- table(y_hat, test$volact)
#nn_pred

m <- neuralnet(volact ~ wkhtot + social_trust,
          train, hidden = 1)

p <- compute(m, test[,c(11, 14)])

predictions <- p$net.result

result_test <- rbinom(nrow(predictions),1,predictions)
result3 <- ifelse(predictions >= 0.3, 1, 0)
result2 <- ifelse(predictions >= 0.2, 1, 0)
```

```
result4 <- ifelse(predictions >= 0.4, 1, 0)

print(cor(result_test, test$volact))

prenn_pred <- table(result_test, test$volact)
#prenn_pred

#png("basic_nn.png", width = 800, height = 600)
#plot.nnet(m)
#dev.off()

# Comparison of the PRE-defined NN, our custom NN and the
# logistic regression (based on two input variables).

F1(nn_pred)
F1(logit_pred)
F1(prenn_pred)
F1(table(result2, test$volact))
F1(table(result3, test$volact))
F1(table(result4, test$volact))

#################################################################
# Logistic Regression: Calculated on the "Basic Model" Data
#################################################################

y <- train$volact
X <- as.matrix(cbind(1, train[,-1]))

startvals <- c(rep(0, 23))
res <- optim(par = startvals,fn = ll_logit, y = y, X = X,
          control = list(fnscale = -1),
          method = "BFGS"
          )

mu <- as.matrix(cbind(1, test[,-1])) %*% res$par
p <- 1/(1 + exp(-mu))

# The following code snippet will be repeated more often, because
# I can have a look at different thresholds for evaluating the
# logistic regression results. The output will always be:
# 1. MAX Accuracy
# 2. MAX F1 measure

y_hat1 <- ifelse(p > 0.1, 1, 0)
y_hat2 <- ifelse(p > 0.2, 1, 0)
y_hat3 <- ifelse(p > 0.3, 1, 0)
y_hat4 <- ifelse(p > 0.4, 1, 0)
y_hat5 <- ifelse(p > 0.5, 1, 0)
y_hat6 <- ifelse(p > 0.6, 1, 0)
y_hat7 <- ifelse(p > 0.7, 1, 0)
y_hat8 <- ifelse(p > 0.8, 1, 0)
y_hat9 <- ifelse(p > 0.9, 1, 0)

threshold <- cbind(seq(0.1, 0.8, length.out = 8),

c(ACC(table(y_hat1, test$volact)),
ACC(table(y_hat2, test$volact)),
ACC(table(y_hat3, test$volact)),
ACC(table(y_hat4, test$volact)),
ACC(table(y_hat5, test$volact)),
ACC(table(y_hat6, test$volact)),
ACC(table(y_hat7, test$volact)),
ACC(table(y_hat8, test$volact))),
#ACC(table(y_hat9, test$volact))),

c(F1(table(y_hat1, test$volact)),
F1(table(y_hat2, test$volact)),
F1(table(y_hat3, test$volact)),
F1(table(y_hat4, test$volact)),
F1(table(y_hat5, test$volact)),
F1(table(y_hat6, test$volact)),
F1(table(y_hat7, test$volact)),
F1(table(y_hat8, test$volact))))
#F1(table(y_hat9, test$volact))))

print("MAX Accuracy for BASIC Logistic Regression")
max(threshold[,2])
print("MAX F1 for BASIC Logistic Regression")
max(threshold[,3])

# The following plot shows the distribution of Accuracy and
# F1 for the different threshold levels. A threshold with both
# values high would be the best!

#acc <-
    ggplot(data.frame(threshold),aes(threshold[,1],threshold[,2]))+geom_line(aes(color="Accuracy")
#   labs(color=" ") +
#   ylab("accuracy value") + xlab("threshold")
#f1 <-
    ggplot(data.frame(threshold),aes(threshold[,1],threshold[,4]))+geom_line(aes(color="F1"))+
#   labs(color=" ") +
#   ylab("f1 value") + xlab("threshold")
#png("evaluation_log_threshold_unbal.png")
#myplot <- ggarrange(acc, f1 + rremove("x.text"),
#        labels = c("ACC", "F1"),
#        ncol = 2, nrow = 1)
#print(myplot)
#dev.off()
#print(myplot)

#################################################################
# Logistic Regression: Calculated on Balanced Data
#################################################################

y.bal <- train.bal$volact
X.bal <- as.matrix(cbind(1, train.bal[,-1]))

startvals <- c(rep(0, 23))
res <- optim(par = startvals,fn = ll_logit, y = y.bal, X = X.bal,
             control = list(fnscale = -1), hessian = TRUE,
             method = "BFGS"
             )

mu <- as.matrix(cbind(1, test[,-1])) %*% res$par

p <- 1/(1 + exp(-mu))

y_hat1 <- ifelse(p > 0.1, 1, 0)
y_hat2 <- ifelse(p > 0.2, 1, 0)
y_hat3 <- ifelse(p > 0.3, 1, 0)
y_hat4 <- ifelse(p > 0.4, 1, 0)
y_hat5 <- ifelse(p > 0.5, 1, 0)
y_hat6 <- ifelse(p > 0.6, 1, 0)
y_hat7 <- ifelse(p > 0.7, 1, 0)
y_hat8 <- ifelse(p > 0.8, 1, 0)
y_hat9 <- ifelse(p > 0.9, 1, 0)

threshold <- cbind(seq(0.1, 0.9, length.out = 9),

c(ACC(table(y_hat1, test$volact)),
ACC(table(y_hat2, test$volact)),
ACC(table(y_hat3, test$volact)),
ACC(table(y_hat4, test$volact)),
ACC(table(y_hat5, test$volact)),
ACC(table(y_hat6, test$volact)),
ACC(table(y_hat7, test$volact)),
ACC(table(y_hat8, test$volact)),
ACC(table(y_hat9, test$volact))),

c(F1(table(y_hat1, test$volact)),
F1(table(y_hat2, test$volact)),
F1(table(y_hat3, test$volact)),
F1(table(y_hat4, test$volact)),
F1(table(y_hat5, test$volact)),
F1(table(y_hat6, test$volact)),
F1(table(y_hat7, test$volact)),
F1(table(y_hat8, test$volact)),
F1(table(y_hat9, test$volact))))

print("MAX Accuracy for Balanced Logistic Regression")
max(threshold[,2])
print("MAX F1 for Balanced Logistic Regression")
max(threshold[,3])

#acc <- ggplot(data.frame(threshold),aes(threshold[,1],threshold[,2]))+
#  geom_line(aes(color="Accuracy"))+
#   labs(color=" ") +
#   ylab("accuracy value") + xlab("threshold")
#f1 <- ggplot(data.frame(threshold),aes(threshold[,1],threshold[,4]))+
#  geom_line(aes(color="F1"))+
#   labs(color=" ") +
#   ylab("f1 value") + xlab("threshold")
#png("evaluation_log_threshold_bal.png")
#myplot <- ggarrange(acc, f1 + rremove("x.text"),
#        labels = c("ACC", "F1"),
#        ncol = 2, nrow = 1)
#print(myplot)
#dev.off()
#print(myplot)

##################################################################
# Logistic Regression: Tested with Pre-Defined Function
##################################################################

ols_model <- glm(y ~ X, family=binomial(link='logit'))

#summary(ols_model)

##################################################################
# Logistic Regression: Preparing Plot for Marginal Effects later
##################################################################

range <- seq(-4, 2, length.out = 20)
sel <- 7
plot.data <- matrix(NA, nrow = length(range), ncol = ncol(test[,-1]))
for(i in 1:ncol(test[,-1])){
   plot.data[,i] <- mean(test[,i])
}
for(i in 1:length(range)){
   plot.data[i, sel] <- range[i]
}

mu <- as.matrix(cbind(1, plot.data)) %*% res$par
p <- 1/(1 + exp(-mu))

age.volact <- data.frame(cbind(range, p))

##################################################################
# Neural Net: Plotting Nets with Neuralnet-Library
##################################################################

#m <- neuralnet(volact ~ sclmeet,
#          train, hidden = 1)

#p <- compute(m, test[,13])

#predictions <- p$net.result

#print(cor(predictions, test$volact))

```

```r
#result <- ifelse(predictions >= 0.3, 1, 0)
#ACC(table(result, test$volact))
#F1(table(result, test$volact))

#plot.nnet(m)

#m2 <- neuralnet(volact ~ sclmeet + social_trust + tolerance +
#         self_realisation + solidarity,
#            train, hidden = 2)

#p2 <- compute(m2, test[,c(13, 14, 15, 16, 17)])

#predictions2 <- p2$net.result

#print(cor(predictions2, test$volact))

#result3 <- ifelse(predictions2 >= 0.3, 1, 0)
#result2 <- ifelse(predictions2 >= 0.2, 1, 0)
#result4 <- ifelse(predictions2 >= 0.4, 1, 0)

#ACC(table(result4, test$volact))
#F1(table(result4, test$volact))

#png("more-complex_nn.png", width = 800, height = 600)
#plot.nnet(m2)
#dev.off()

#m3 <- neuralnet(volact ~ round1 + female + yrbrn + eduyrs + domicil + married +
#            children + houseperson + wkhtot + church + sclmeet + social_trust +
#            tolerance + self_realisation + solidarity + tvpol + tvtot +
#            political_interest + trust_exe + trust_leg + trstep + stfdem,
#            train, hidden = 1)

#p3 <- compute(m3, test[,-c(1, 2, 25)])

#predictions3 <- p3$net.result

#print(cor(predictions3, test$volact))

#library(nnet)
#png("total-input_nn.png", width = 800, height = 600)
#plot.nnet(m3)
#dev.off()

##################################################################
# Neural Net: Models for Plotting Tuning Differences (Part 1)
##################################################################

# For this plot, I am tuning the decay rate, which is an
# additional weight parameter influencing the signals between
# the neurons.

#aus: R deep learning essentials
#m20.0 <- train(x = train.X, y = train.y,
#        method = "nnet",
#        tuneGrid = expand.grid(
#        .size = c(20),
#        .decay = 0),
#        trControl= trainControl(method = "none"),
#        MaxNWts = 1000,
#        maxit = 100)
#m20.1 <- train(x = train.X, y = train.y,
#        method = "nnet",
#        tuneGrid = expand.grid(
#        .size = c(20),
#        .decay = 0.1),
#        trControl= trainControl(method = "none"),
#        MaxNWts = 1000,
#        maxit = 100)
#m20.2 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 0.2),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 1000,
#         maxit = 100)
#m20.3 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 0.3),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 1000,
#         maxit = 100)
#m20.4 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 0.4),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 1000,
#         maxit = 100)
#m20.5 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 0.5),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 1000,
#         maxit = 100)
#m20.6 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 0.6),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 1000,
#         maxit = 100)
#m20.7 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 0.7),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 1000,
#         maxit = 100)
#m20.8 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 0.8),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 1000,
#         maxit = 100)
#m20.9 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 0.9),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 1000,
#         maxit = 100)
#m20.10 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 1),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 1000,
#        maxit = 100)

#yhat20.0 <- predict(m20.0)
#yhat20.1 <- predict(m20.1)
#yhat20.2 <- predict(m20.2)
#yhat20.3 <- predict(m20.3)
#yhat20.4 <- predict(m20.4)
#yhat20.5 <- predict(m20.5)
#yhat20.6 <- predict(m20.6)
#yhat20.7 <- predict(m20.7)
#yhat20.8 <- predict(m20.8)
#yhat20.9 <- predict(m20.9)
#yhat20.10 <- predict(m20.10)
#yhat_unseen20.0 <- predict(m20.0, as.matrix(test.X))
#yhat_unseen20.1 <- predict(m20.1, as.matrix(test.X))
#yhat_unseen20.2 <- predict(m20.2, as.matrix(test.X))
#yhat_unseen20.3 <- predict(m20.3, as.matrix(test.X))
#yhat_unseen20.4 <- predict(m20.4, as.matrix(test.X))
#yhat_unseen20.5 <- predict(m20.5, as.matrix(test.X))
#yhat_unseen20.6 <- predict(m20.6, as.matrix(test.X))
#yhat_unseen20.7 <- predict(m20.7, as.matrix(test.X))
#yhat_unseen20.8 <- predict(m20.8, as.matrix(test.X))
#yhat_unseen20.9 <- predict(m20.9, as.matrix(test.X))
#yhat_unseen20.10 <- predict(m20.10, as.matrix(test.X))

#measures <- c("AccuracyNull", "Accuracy", "AccuracyLower", "AccuracyUpper")

#n20.0.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.0))
#n20.1.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.1))
#n20.2.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.2))
#n20.3.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.3))
#n20.4.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.4))
#n20.5.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.5))
#n20.6.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.6))
#n20.7.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.7))
#n20.8.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.8))
#n20.9.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.9))
#n20.10.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20.10))
#n20.0.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.0))
#n20.1.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.1))
#n20.2.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.2))
#n20.3.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.3))
#n20.4.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.4))
#n20.5.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.5))
#n20.6.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.6))
#n20.7.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.7))
#n20.8.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.8))
#n20.9.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.9))
#n20.10.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20.10))

#shrinkage <- rbind(
#  cbind(Size = 20.0, Sample = "In",
    as.data.frame(t(n20.0.insample$overall[measures]))),
#  cbind(Size = 20.0, Sample = "Out",
    as.data.frame(t(n20.0.outsample$overall[measures]))),
#  cbind(Size = 20.1, Sample = "In",
    as.data.frame(t(n20.1.insample$overall[measures]))),
#  cbind(Size = 20.1, Sample = "Out",
    as.data.frame(t(n20.1.outsample$overall[measures]))),
#  cbind(Size = 20.2, Sample = "In",
    as.data.frame(t(n20.2.insample$overall[measures]))),
#  cbind(Size = 20.2, Sample = "Out",
    as.data.frame(t(n20.2.outsample$overall[measures]))),
#  cbind(Size = 20.3, Sample = "In",
    as.data.frame(t(n20.3.insample$overall[measures]))),
#  cbind(Size = 20.3, Sample = "Out",
    as.data.frame(t(n20.3.outsample$overall[measures]))),
#  cbind(Size = 20.4, Sample = "In",
    as.data.frame(t(n20.4.insample$overall[measures]))),
#  cbind(Size = 20.4, Sample = "Out",
     as.data.frame(t(n20.4.outsample$overall[measures]))),
#  cbind(Size = 20.5, Sample = "In",
     as.data.frame(t(n20.5.insample$overall[measures]))),
#  cbind(Size = 20.5, Sample = "Out",
     as.data.frame(t(n20.5.outsample$overall[measures]))),
#  cbind(Size = 20.6, Sample = "In",
     as.data.frame(t(n20.6.insample$overall[measures]))),
#  cbind(Size = 20.6, Sample = "Out",
     as.data.frame(t(n20.6.outsample$overall[measures]))),
#  cbind(Size = 20.7, Sample = "In",
     as.data.frame(t(n20.7.insample$overall[measures]))),
#  cbind(Size = 20.7, Sample = "Out",
     as.data.frame(t(n20.7.outsample$overall[measures]))),
#  cbind(Size = 20.8, Sample = "In",
     as.data.frame(t(n20.8.insample$overall[measures]))),
#  cbind(Size = 20.8, Sample = "Out",
     as.data.frame(t(n20.8.outsample$overall[measures]))),
#  cbind(Size = 20.9, Sample = "In",
     as.data.frame(t(n20.9.insample$overall[measures]))),
#  cbind(Size = 20.9, Sample = "Out",
     as.data.frame(t(n20.9.outsample$overall[measures]))),
#  cbind(Size = 20.99, Sample = "In",
     as.data.frame(t(n20.10.insample$overall[measures]))),
#  cbind(Size = 20.99, Sample = "Out",
     as.data.frame(t(n20.10.outsample$overall[measures])))
#  )
#shrinkage$Pkg <- rep(c("In", "Out"), 1)

#dodge <- position_dodge(width=0.4)

#p.shrinkage <- ggplot(shrinkage, aes(interaction(Size, sep = " : "), Accuracy,
#                 ymin = AccuracyLower, ymax = AccuracyUpper,
#                 shape = Sample, linetype = Sample)) +
#  geom_point(size = 2.5, position = dodge) +
#  geom_errorbar(width = .25, position = dodge) +
#  xlab("") + ylab("Accuracy + 95% CI") +
#  theme_classic() +
#  theme(legend.key.size = unit(1, "cm"), legend.position = c(.8, .2))

#png("test_20.png",
#   width = 6, height = 6, units = "in", res = 600)
#  print(p.shrinkage)
#dev.off()

#################################################################
# Neural Net: Models for Plotting Tuning Differences (Part 2)
#################################################################

# This time I am tuning the hidden layer size, ranging from 5
# hidden neurons to 100.

#m5 <- train(x = train.X, y = train.y,
#        method = "nnet",
#        tuneGrid = expand.grid(
#        .size = c(5),
#        .decay = 0),
#        trControl= trainControl(method = "none"),
#        MaxNWts = 2000,
#        maxit = 100)
#m10 <- train(x = train.X, y = train.y,
#        method = "nnet",
#        tuneGrid = expand.grid(
#        .size = c(10),
#        .decay = 0),
#        trControl= trainControl(method = "none"),
#        MaxNWts = 2000,
#        maxit = 100)
#m15 <- train(x = train.X, y = train.y,
#        method = "nnet",
#        tuneGrid = expand.grid(
#         .size = c(15),
#         .decay = 0),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 2000,
#         maxit = 100)
#m20 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(20),
#         .decay = 0),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 2000,
#         maxit = 100)
#m30 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(30),
#         .decay = 0),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 2000,
#         maxit = 100)
#m40 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(40),
#         .decay = 0),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 2000,
#         maxit = 100)
#m50 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(50),
#         .decay = 0),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 2000,
#         maxit = 100)
#m70 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(70),
#         .decay = 0),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 2000,
#         maxit = 100)
#m100 <- train(x = train.X, y = train.y,
#         method = "nnet",
#         tuneGrid = expand.grid(
#         .size = c(100),
#         .decay = 0),
#         trControl= trainControl(method = "none"),
#         MaxNWts = 20000,
#         maxit = 100)

#yhat5 <- predict(m5)
#yhat10 <- predict(m10)
#yhat15 <- predict(m15)
#yhat20 <- predict(m20)
#yhat30 <- predict(m30)
#yhat40 <- predict(m40)
#yhat50 <- predict(m50)
#yhat70 <- predict(m70)
#yhat100 <- predict(m100)

#yhat_unseen5 <- predict(m5, as.matrix(test.X))
#yhat_unseen10 <- predict(m10, as.matrix(test.X))
#yhat_unseen15 <- predict(m15, as.matrix(test.X))
#yhat_unseen20 <- predict(m20, as.matrix(test.X))
#yhat_unseen30 <- predict(m30, as.matrix(test.X))
#yhat_unseen40 <- predict(m40, as.matrix(test.X))
```

```
#yhat_unseen50 <- predict(m50, as.matrix(test.X))
#yhat_unseen70 <- predict(m70, as.matrix(test.X))
#yhat_unseen100 <- predict(m100, as.matrix(test.X))

#measures <- c("AccuracyNull", "Accuracy", "AccuracyLower", "AccuracyUpper")

#n5.insample <- caret::confusionMatrix(xtabs(~train.y + yhat5))
#n10.insample <- caret::confusionMatrix(xtabs(~train.y + yhat10))
#n15.insample <- caret::confusionMatrix(xtabs(~train.y + yhat15))
#n20.insample <- caret::confusionMatrix(xtabs(~train.y + yhat20))
#n30.insample <- caret::confusionMatrix(xtabs(~train.y + yhat30))
#n40.insample <- caret::confusionMatrix(xtabs(~train.y + yhat40))
#n50.insample <- caret::confusionMatrix(xtabs(~train.y + yhat50))
#n70.insample <- caret::confusionMatrix(xtabs(~train.y + yhat70))
#n100.insample <- caret::confusionMatrix(xtabs(~train.y + yhat100))

#n5.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen5))
#n10.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen10))
#n15.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen15))
#n20.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen20))
#n30.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen30))
#n40.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen40))
#n50.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen50))
#n70.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen70))
#n100.outsample <- caret::confusionMatrix(xtabs(~test.y + yhat_unseen100))

#shrinkage <- rbind(
#  cbind(Size = 5, Sample = "In", as.data.frame(t(n5.insample$overall[measures]))),
#  cbind(Size = 5, Sample = "Out", as.data.frame(t(n5.outsample$overall[measures]))),
#  cbind(Size = 10, Sample = "In", as.data.frame(t(n10.insample$overall[measures]))),
#  cbind(Size = 10, Sample = "Out",
    as.data.frame(t(n10.outsample$overall[measures]))),
#  cbind(Size = 15, Sample = "In", as.data.frame(t(n15.insample$overall[measures]))),
#  cbind(Size = 15, Sample = "Out",
    as.data.frame(t(n15.outsample$overall[measures]))),
#  cbind(Size = 20, Sample = "In", as.data.frame(t(n20.insample$overall[measures]))),
#  cbind(Size = 20, Sample = "Out",
    as.data.frame(t(n20.outsample$overall[measures]))),
#  cbind(Size = 30, Sample = "In", as.data.frame(t(n30.insample$overall[measures]))),
#  cbind(Size = 30, Sample = "Out",
    as.data.frame(t(n30.outsample$overall[measures]))),
#  cbind(Size = 40, Sample = "In", as.data.frame(t(n40.insample$overall[measures]))),
#  cbind(Size = 40, Sample = "Out",
    as.data.frame(t(n40.outsample$overall[measures]))),
#  cbind(Size = 50, Sample = "In", as.data.frame(t(n50.insample$overall[measures]))),
#  cbind(Size = 50, Sample = "Out",
    as.data.frame(t(n50.outsample$overall[measures]))),
#  cbind(Size = 70, Sample = "In", as.data.frame(t(n70.insample$overall[measures]))),
#  cbind(Size = 70, Sample = "Out",
    as.data.frame(t(n70.outsample$overall[measures]))),
#  cbind(Size = 100, Sample = "In",
    as.data.frame(t(n100.insample$overall[measures]))),
#  cbind(Size = 100, Sample = "Out",
    as.data.frame(t(n100.outsample$overall[measures])))
#  )
#shrinkage$Pkg <- rep(c("In", "Out"), 1)

#dodge <- position_dodge(width=0.4)

#p.shrinkage <- ggplot(shrinkage, aes(interaction(Size, sep = " : "), Accuracy,
#                 ymin = AccuracyLower, ymax = AccuracyUpper,
#                 shape = Sample, linetype = Sample)) +
#  geom_point(size = 2.5, position = dodge) +
#  geom_errorbar(width = .25, position = dodge) +
#  xlab("") + ylab("Accuracy + 95% CI") +
#  theme_classic() +
#  theme(legend.key.size = unit(1, "cm"), legend.position = c(.8, .2))

#png("test_5150.png",
#   width = 6, height = 6, units = "in", res = 600)
#  print(p.shrinkage)
#dev.off()

##################################################################
# Neural Net: H2O Models
##################################################################

# Initializing the H2O-cluster
c1 <- h2o.init()

# Setting up the data for H2O
train$volact <- as.factor(train$volact)
test$volact <- as.factor(test$volact)

h2o.train <- as.h2o(train)
h2o.test <- as.h2o(test)

train.bal$volact <- as.factor(train.bal$volact)
h2o.train.bal <- as.h2o(train.bal)

xnames <- colnames(train[,-1])

##################################################################
# Neural Net: H2O Models for Plotting Tuning Differences (Part 3)
##################################################################

# Now I am testing different depths of hidden layers, starting
# with one layer and going until 4 layers. The error
# distribution is going to be plotted.

#m2a <- h2o.deeplearning(
#  x = xnames,
#  #y = "Outcome",
#  training_frame= h2o.train,
#  validation_frame = h2o.test,
#  activation = "Tanh",
#  autoencoder = TRUE,
#  hidden = c(20),
#  epochs = 10,
#  sparsity_beta = 0,
#  l1 = 0,
#  l2 = 0
#)

#m2b <- h2o.deeplearning(
#  x = xnames,
#  #y = "Outcome",
#  training_frame= h2o.train,
#  validation_frame = h2o.test,
#  activation = "Tanh",
#  autoencoder = TRUE,
#  hidden = c(20, 15),
#  epochs = 10,
#  sparsity_beta = 0,
#  #hidden_pout_ratios = c(.3),
#  l1 = 0,
#  l2 = 0
#)

#m2c <- h2o.deeplearning(
#  x = xnames,
#  #y = "Outcome",
#  training_frame= h2o.train,
#  validation_frame = h2o.test,
#  activation = "Tanh",
#  autoencoder = TRUE,
#  hidden = c(20, 15, 10),
#  epochs = 10,
#  sparsity_beta = 0,
#  l1 = 0,
#  l2 = 0
#)

#m2d <- h2o.deeplearning(
#  x = xnames,
#  #y = "Outcome",
#  training_frame= h2o.train,
#  validation_frame = h2o.test,
#  activation = "Tanh",
#  autoencoder = TRUE,
#  hidden = c(20, 15, 10, 5),
#  epochs = 10,
#  sparsity_beta = 0,
#  #hi2den_dropout_ratios = c(.3),
#  l1 = 0,
#  l2 = 0
#)

#summary(m2a)
#summary(m2b)
#summary(m2c)
#summary(m2d)

#error1 <- as.data.frame(h2o.anomaly(m2a, h2o.train))
#error2a <- as.data.frame(h2o.anomaly(m2b, h2o.train))
#error2b <- as.data.frame(h2o.anomaly(m2c, h2o.train))
#error2c <- as.data.frame(h2o.anomaly(m2d, h2o.train))

#error <- as.data.table(rbind(
#  cbind.data.frame(Model = "2a", error1),
#  cbind.data.frame(Model = "2b", error2a),
#  cbind.data.frame(Model = "2c", error2b),
#  cbind.data.frame(Model = "2d", error2c)))

#percentile <- error[, .(
#  Percentile = quantile(Reconstruction.MSE, probs = .95)
#), by = Model]

#p1 <- ggplot(error, aes(Reconstruction.MSE)) +
#  geom_histogram(binwidth = .001, fill = "grey50") +
#  geom_vline(aes(xintercept = Percentile), data = percentile, linetype = 2) +
#  theme_bw() +
#  facet_wrap(~Model)
#print(p1)

#png("error_1.png",
#   width = 5.5, height = 5.5, units = "in", res = 600)
#print(p1)
#dev.off()

##################################################################
# Neural Net: Calculated on the "Basic Model" Data
##################################################################

mt3 <- h2o.deeplearning(
  x = xnames,
  y = "volact",
  training_frame = h2o.train,
  validation_frame = h2o.test,
  activation = "Rectifier",
  hidden = c(200, 200, 100),
  epochs = 10,
  rate = .005,
  loss = "CrossEntropy",
  #input_dropout_ratio = .2,
  #hidden_dropout_ratios = c(.5, .3, .1),
  export_weights_and_biases = TRUE
)
summary(mt3)

```

```
mt3_pred <- h2o.predict(mt3, h2o.test)
ACC(table(as.vector(mt3_pred$predict), as.vector(h2o.test$volact)))
F1(table(as.vector(mt3_pred$predict), as.vector(h2o.test$volact)))

# These variable importances are used for the diagramm.
#h2o.varimp(mt3)

##################################################################
# Neural Net: Calculated on Balanced Data
##################################################################

mtbal <- h2o.deeplearning(
  x = xnames,
  y = "volact",
  training_frame = h2o.train.bal,
  validation_frame = h2o.test,
  activation = "Rectifier",
  hidden = c(200, 200, 100),
  epochs = 10,
  rate = .005,
  loss = "CrossEntropy",
  #input_dropout_ratio = .2,
  #hidden_dropout_ratios = c(.5, .3, .1),
  export_weights_and_biases = TRUE
)
summary(mtbal)

mtbal_pred <- h2o.predict(mtbal, h2o.test)
ACC(table(as.vector(mtbal_pred$predict), as.vector(h2o.test$volact)))
F1(table(as.vector(mtbal_pred$predict), as.vector(h2o.test$volact)))

#h2o.varimp(mtbal)

##################################################################
# Neural Net: Plot a Heatmap of the First Layer Weights
##################################################################

## weights for mapping from inputs to hidden layer 1 neurons
#w1 <- as.matrix(h2o.weights(mtbal, 1))

## plot heatmap of the weights
#tmp <- as.data.frame(t(w1))
#tmp$Row <- 1:nrow(tmp)
#tmp <- melt(tmp, id.vars = c("Row"))

#p.heat <- ggplot(tmp,
#      aes(variable, Row, fill = value)) +
#  geom_tile() +
#  scale_fill_gradientn(colours = c("black", "white", "blue")) +
#  theme_classic() +
#  theme(axis.text = element_blank()) +
#  xlab("Hidden Neuron") +
#  ylab("Input Variable") +
#  ggtitle("Heatmap of Weights for Layer 1")
#print(p.heat)

#png("heatmap_layer1.png",
#   width = 5.5, height = 7.5, units = "in", res = 600)
#print(p.heat)
#dev.off()

##################################################################
# Neural Net: Preparing Plot for Marginal Effects later
##################################################################

h2o.plot <- as.h2o(plot.data)

mt3_pred <- h2o.predict(mt3, h2o.plot)

##################################################################
```

```
# Random Forest: Calculated on the "Basic Model" Data
###################################################################

rf2 <- h2o.randomForest(
 training_frame = h2o.train,
 validation_frame = h2o.test,
 x = xnames,
 y = "volact",
 model_id = "forest2",
 ntrees = 400,
 max_depth = 30,
 stopping_rounds = 0,
 score_each_iteration = F,
 seed = 1000000)
summary(rf2)

# These variable importances are used for the diagramm.
# h2o.varimp(rf2)

rf2_pred <- h2o.predict(rf2, h2o.test)
ACC(table(as.vector(rf2_pred$predict), as.vector(h2o.test$volact)))
F1(table(as.vector(rf2_pred$predict), as.vector(h2o.test$volact)))

###################################################################
# Random Forest: Calculated on Balanced Data
###################################################################

rfbal <- h2o.randomForest(
 training_frame = h2o.train.bal,
 validation_frame = h2o.test,
 x = xnames,
 y = "volact",
 model_id = "forestbal",
 ntrees = 400,
 max_depth = 30,
 stopping_rounds = 0,
 score_each_iteration = F,
 seed = 1000000)
summary(rfbal)

rfbal_pred <- h2o.predict(rfbal, h2o.test)
ACC(table(as.vector(rfbal_pred$predict), as.vector(h2o.test$volact)))
F1(table(as.vector(rfbal_pred$predict), as.vector(h2o.test$volact)))


###################################################################
# Random Forest: Preparing Plot for Marginal Effects
###################################################################

rf2_pred <- h2o.predict(rf2, h2o.plot)

# Condensing the prepared results into one dataframe
# and plotting it.

age.volact <- data.frame(cbind(as.matrix(age.volact),
   as.vector(mt3_pred$p1), as.vector(rf2_pred$p1)))
colnames(age.volact) <- c("range", "1", "2", "3")
melted <- melt(age.volact, id.vars="range")

#png("age-volact.png")
#myplot <- ggplot(data=melted, aes(x=range, y=value, group=variable)) +
#  geom_line(col = melted$variable) +
#   ylab("Predicted Probability of Voluntary Action") +
#   xlab("Year of Birth (Scaled)") +
#   ggtitle("Comparison: \n Predicted Probabilites of Voluntary Action for
#  a range of Year of Births \n From top to down: Logistic Regression, Neural
    Network, Random Forest") +
#   theme_bw()
#print(myplot)
#dev.off()
```

```
#print(myplot)

####################################################################
####################################################################

####################################################################
# Preparing the Data for the Separate Models
####################################################################

ess1 <-
    read.table("C:/Users/laris/Desktop/MMDS/Semester3-FSS2018/Advanced-Quantitative-Methods/paper/
    header=TRUE, sep=",")

ess6 <-
    read.table("C:/Users/laris/Desktop/MMDS/Semester3-FSS2018/Advanced-Quantitative-Methods/paper/
    header=TRUE, sep=",")

for(i in c(4,5,6,10,11,12,13,14,15,16,17,18,19,20,21,22,23)){
   ess1[is.na(ess1[,i]), i] <- mean(ess1[,i], na.rm = TRUE)
}

ess1[is.na(ess1[,3]), 3] <- 1
ess1[is.na(ess1[,7]), 7] <- 1
ess1[is.na(ess1[,8]), 8] <- 0
ess1[is.na(ess1[,9]), 9] <- 0

for(i in c(4,5,6,10,11,12,13,14,15,16,17,18,19,20,21,22,23)){
   ess6[is.na(ess6[,i]), i] <- mean(ess6[,i], na.rm = TRUE)
}

ess6[is.na(ess6[,3]), 3] <- 1
ess6[is.na(ess6[,7]), 7] <- 1
ess6[is.na(ess6[,8]), 8] <- 0
ess6[is.na(ess6[,9]), 9] <- 0

ess1.scaled <- cbind(ess1[c(1, 3, 7, 8, 9)], scale(ess1[-c(1, 2, 3, 7, 8, 9, 24)]))

ess1c.scaled <- cbind(ess1[c(1, 2, 3, 7, 8, 9)], scale(ess1[-c(1, 2, 3, 7, 8, 9,
    24)]))

ess6.scaled <- cbind(ess6[c(1, 3, 7, 8, 9)], scale(ess6[-c(1, 2, 3, 7, 8, 9, 24)]))

ess6c.scaled <- cbind(ess6[c(1, 2, 3, 7, 8, 9)], scale(ess6[-c(1, 2, 3, 7, 8, 9,
    24)]))

for(t in unique(ess1c.scaled$cntry)) {
   ess1c.scaled[paste("cntry",t,sep="")] <- ifelse(ess1c.scaled$cntry==t,1,0)
}

ess1c.scaled <- ess1c.scaled[,-2]

for(t in unique(ess6c.scaled$cntry)) {
   ess6c.scaled[paste("cntry",t,sep="")] <- ifelse(ess6c.scaled$cntry==t,1,0)
}

ess6c.scaled <- ess6c.scaled[,-2]

# Same as above: this can be un-commented when the data sets are
# already there and just have to be written and loaded.

#write.csv(ess1.scaled, file = "ess1-scaled.csv")
#write.csv(ess1c.scaled, file = "ess1-scaled-cntry.csv")

#write.csv(ess6.scaled, file = "ess6-scaled.csv")
#write.csv(ess6c.scaled, file = "ess6-scaled-cntry.csv")

#ess1.scaled <- read.table("ess1-scaled.csv", header=TRUE, sep=",")
#ess6.scaled <- read.table("ess6-scaled.csv", header=TRUE, sep=",")
#ess1c.scaled <- read.table("ess1-scaled-cntry.csv", header=TRUE, sep=",")
#ess6c.scaled <- read.table("ess6-scaled-cntry.csv", header=TRUE, sep=",")
```

```

#ess1.scaled <- ess1.scaled[,-1]
#ess6.scaled <- ess6.scaled[,-1]

#ess1c.scaled <- ess1c.scaled[,-1]
#ess6c.scaled <- ess6c.scaled[,-1]

# Again: splitting into training and test set

set.seed(123)
smp_size <- floor(0.9 * nrow(ess1.scaled))
train_ind <- sample(seq_len(nrow(ess1.scaled)), size = smp_size)
train1 <- ess1.scaled[train_ind,]
test1 <- ess1.scaled[-train_ind,]
train1c <- ess1c.scaled[train_ind,]
test1c <- ess1c.scaled[-train_ind,]

set.seed(123)
smp_size <- floor(0.9 * nrow(ess6.scaled))
train_ind <- sample(seq_len(nrow(ess6.scaled)), size = smp_size)
train6 <- ess6.scaled[train_ind,]
test6 <- ess6.scaled[-train_ind,]
train6c <- ess6c.scaled[train_ind,]
test6c <- ess6c.scaled[-train_ind,]

#################################################################
# Logistic Regression: Calculated on two years separately
#################################################################

y1 <- train1$volact
X1 <- as.matrix(train1[,-1])
ols_model1 <- glm(y1 ~ X1, family=binomial(link='logit'))
X1 <- as.matrix(test1[,-1])
p_ols1 <- predict(ols_model1, data.frame(X1), type = "response")

#table(p_ols1, test1$volact)
#ACC(table(p_ols1, test1$volact))
#F1(table(p_ols1, test1$volact))

y6 <- train6$volact
X6 <- as.matrix(train6[,-1])
ols_model6 <- glm(y6 ~ X6, family=binomial(link='logit'))
X6 <- as.matrix(test6[,-1])
p_ols6 <- predict(ols_model6, data.frame(X6), type = "response")

#table(y_hat6, test6$volact)
#ACC(table(y_hat6, test6$volact))
#F1(table(y_hat6, test6$volact))

p_ols11 <- ifelse(p_ols1 > 0.1, 1, 0)
p_ols12 <- ifelse(p_ols1 > 0.2, 1, 0)
p_ols13 <- ifelse(p_ols1 > 0.3, 1, 0)
p_ols14 <- ifelse(p_ols1 > 0.4, 1, 0)
p_ols15 <- ifelse(p_ols1 > 0.5, 1, 0)
p_ols16 <- ifelse(p_ols1 > 0.6, 1, 0)
p_ols17 <- ifelse(p_ols1 > 0.7, 1, 0)
#p_ols18 <- ifelse(p_ols1 > 0.8, 1, 0)
#p_ols19 <- ifelse(p_ols1 > 0.9, 1, 0)

threshold <- cbind(seq(0.1, 0.7, length.out = 7),

c(ACC(table(p_ols11, test1$volact)),
ACC(table(p_ols12, test1$volact)),
ACC(table(p_ols13, test1$volact)),
ACC(table(p_ols14, test1$volact)),
ACC(table(p_ols15, test1$volact)),
ACC(table(p_ols16, test1$volact)),
ACC(table(p_ols17, test1$volact))),
#ACC(table(p_ols18, test1$volact)),
#ACC(table(p_ols19, test1$volact))),

c(F1(table(p_ols11, test1$volact)),
F1(table(p_ols12, test1$volact)),
F1(table(p_ols13, test1$volact)),
F1(table(p_ols14, test1$volact)),
F1(table(p_ols15, test1$volact)),
F1(table(p_ols16, test1$volact)),
F1(table(p_ols17, test1$volact))))
#F1(table(p_ols18, test1$volact))))
#F1(table(p_ols19, test1$volact))))

print("MAX Accuracy for ESS1 Logistic Regression")
max(threshold[,2])
print("MAX F1 for ESS1 Logistic Regression")
max(threshold[,3])

p_ols61 <- ifelse(p_ols6 > 0.1, 1, 0)
p_ols62 <- ifelse(p_ols6 > 0.2, 1, 0)
p_ols63 <- ifelse(p_ols6 > 0.3, 1, 0)
p_ols64 <- ifelse(p_ols6 > 0.4, 1, 0)
p_ols65 <- ifelse(p_ols6 > 0.5, 1, 0)
p_ols66 <- ifelse(p_ols6 > 0.6, 1, 0)
p_ols67 <- ifelse(p_ols6 > 0.7, 1, 0)
p_ols68 <- ifelse(p_ols6 > 0.8, 1, 0)
#p_ols69 <- ifelse(p_ols6 > 0.9, 1, 0)

threshold <- cbind(seq(0.1, 0.8, length.out = 8),

c(ACC(table(p_ols61, test6$volact)),
ACC(table(p_ols62, test6$volact)),
ACC(table(p_ols63, test6$volact)),
ACC(table(p_ols64, test6$volact)),
ACC(table(p_ols65, test6$volact)),
ACC(table(p_ols66, test6$volact)),
ACC(table(p_ols67, test6$volact)),
ACC(table(p_ols68, test6$volact))),
#ACC(table(p_ols69, test6$volact))),

c(F1(table(p_ols61, test6$volact)),
F1(table(p_ols62, test6$volact)),
F1(table(p_ols63, test6$volact)),
F1(table(p_ols64, test6$volact)),
F1(table(p_ols65, test6$volact)),
F1(table(p_ols66, test6$volact)),
F1(table(p_ols67, test6$volact)),
F1(table(p_ols68, test6$volact))))
#F1(table(p_ols69, test6$volact))))

print("MAX Accuracy for ESS6 Logistic Regression")
max(threshold[,2])
print("MAX F1 for ESS6 Logistic Regression")
max(threshold[,3])

##################################################################
# Logistic Regression: Calculated with Country Dummies
##################################################################

y1c <- train1c$volact
X1c <- as.matrix(train1c[,-c(1, 44)])
ols_model1c <- glm(y1c ~ X1c, family=binomial(link='logit'))
X1c <- as.matrix(test1c[,-c(1, 44)])
p_ols1c <- predict(ols_model1c, data.frame(X1c), type = "response")

#table(p_ols1, test1$volact)
#ACC(table(p_ols1, test1$volact))
#F1(table(p_ols1, test1$volact))

y6c <- train6c$volact
X6c <- as.matrix(train6c[,-c(1, 51)])
ols_model6c <- glm(y6c ~ X6c, family=binomial(link='logit'))
```

```
X6c <- as.matrix(test6c[,-c(1, 51)])
p_ols6c <- predict(ols_model6c, data.frame(X6c), type = "response")

#table(y_hat6, test6$volact)
#ACC(table(y_hat6, test6$volact))
#F1(table(y_hat6, test6$volact))

p_ols11c <- ifelse(p_ols1c > 0.1, 1, 0)
p_ols12c <- ifelse(p_ols1c > 0.2, 1, 0)
p_ols13c <- ifelse(p_ols1c > 0.3, 1, 0)
p_ols14c <- ifelse(p_ols1c > 0.4, 1, 0)
p_ols15c <- ifelse(p_ols1c > 0.5, 1, 0)
p_ols16c <- ifelse(p_ols1c > 0.6, 1, 0)
p_ols17c <- ifelse(p_ols1c > 0.7, 1, 0)
p_ols18c <- ifelse(p_ols1c > 0.8, 1, 0)
#p_ols19c <- ifelse(p_ols1c > 0.9, 1, 0)

threshold <- cbind(seq(0.1, 0.8, length.out = 8),

c(ACC(table(p_ols11c, test1c$volact)),
ACC(table(p_ols12c, test1c$volact)),
ACC(table(p_ols13c, test1c$volact)),
ACC(table(p_ols14c, test1c$volact)),
ACC(table(p_ols15c, test1c$volact)),
ACC(table(p_ols16c, test1c$volact)),
ACC(table(p_ols17c, test1c$volact)),
ACC(table(p_ols18c, test1c$volact))),
#ACC(table(p_ols19c, test1c$volact))),

c(F1(table(p_ols11c, test1c$volact)),
F1(table(p_ols12c, test1c$volact)),
F1(table(p_ols13c, test1c$volact)),
F1(table(p_ols14c, test1c$volact)),
F1(table(p_ols15c, test1c$volact)),
F1(table(p_ols16c, test1c$volact)),
F1(table(p_ols17c, test1c$volact)),
F1(table(p_ols18c, test1c$volact))))
#F1(table(p_ols19c, test1c$volact))))

print("MAX Accuracy for ESS1 Logistic Regression with Countries")
max(threshold[,2])
print("MAX F1 for ESS1 Logistic Regression with Countries")
max(threshold[,3])

p_ols61c <- ifelse(p_ols6c > 0.1, 1, 0)
p_ols62c <- ifelse(p_ols6c > 0.2, 1, 0)
p_ols63c <- ifelse(p_ols6c > 0.3, 1, 0)
p_ols64c <- ifelse(p_ols6c > 0.4, 1, 0)
p_ols65c <- ifelse(p_ols6c > 0.5, 1, 0)
p_ols66c <- ifelse(p_ols6c > 0.6, 1, 0)
p_ols67c <- ifelse(p_ols6c > 0.7, 1, 0)
p_ols68c <- ifelse(p_ols6c > 0.8, 1, 0)
#p_ols69c <- ifelse(p_ols6c > 0.9, 1, 0)

threshold <- cbind(seq(0.1, 0.8, length.out = 8),

c(ACC(table(p_ols61c, test6c$volact)),
ACC(table(p_ols62c, test6c$volact)),
ACC(table(p_ols63c, test6c$volact)),
ACC(table(p_ols64c, test6c$volact)),
ACC(table(p_ols65c, test6c$volact)),
ACC(table(p_ols66c, test6c$volact)),
ACC(table(p_ols67c, test6c$volact)),
ACC(table(p_ols68c, test6c$volact))),
#ACC(table(p_ols69c, test6c$volact))),

c(F1(table(p_ols61c, test6c$volact)),
F1(table(p_ols62c, test6c$volact)),
F1(table(p_ols63c, test6c$volact)),
F1(table(p_ols64c, test6c$volact)),
F1(table(p_ols65c, test6c$volact)),
F1(table(p_ols66c, test6c$volact)),
F1(table(p_ols67c, test6c$volact)),
F1(table(p_ols68c, test6c$volact))))
#F1(table(p_ols69c, test6c$volact))))

print("MAX Accuracy for ESS6 Logistic Regression with Countries")
max(threshold[,2])
print("MAX F1 for ESS6 Logistic Regression with Countries")
max(threshold[,3])

##################################################################
# Neural Net: H2O Models Extended
##################################################################

# Again: transforming the data into H2O-data
train1$volact <- as.factor(train1$volact)
test1$volact <- as.factor(test1$volact)
train6$volact <- as.factor(train6$volact)
test6$volact <- as.factor(test6$volact)

h2o.train1 <- as.h2o(train1)
h2o.test1 <- as.h2o(test1)
h2o.train6 <- as.h2o(train6)
h2o.test6 <- as.h2o(test6)

train1c$volact <- as.factor(train1c$volact)
test1c$volact <- as.factor(test1c$volact)
train6c$volact <- as.factor(train6c$volact)
test6c$volact <- as.factor(test6c$volact)

h2o.train1c <- as.h2o(train1c)
h2o.test1c <- as.h2o(test1c)
h2o.train6c <- as.h2o(train6c)
h2o.test6c <- as.h2o(test6c)

xnames1 <- colnames(train1[,-1])
xnames6 <- colnames(train6[,-1])

xnames1c <- colnames(train1c[,-1])
xnames6c <- colnames(train6c[,-1])

##################################################################
# Neural Net: Calculated on two years separately
##################################################################

nn1_1 <- h2o.deeplearning(
  x = xnames1,
  y = "volact",
  training_frame = h2o.train1,
  validation_frame = h2o.test1,
  activation = "Rectifier",
  hidden = c(200, 200, 100),
  epochs = 10,
  rate = .005,
  loss = "CrossEntropy",
  export_weights_and_biases = TRUE,
  seed = 1000000
)

nn6_1 <- h2o.deeplearning(
  x = xnames6,
  y = "volact",
  training_frame = h2o.train6,
  validation_frame = h2o.test6,
  activation = "Rectifier",
  hidden = c(200, 200, 100),
  epochs = 10,
  rate = .005,
  loss = "CrossEntropy",
  export_weights_and_biases = TRUE,
  seed = 1000000
)

summary(nn1_1)

summary(nn6_1)

nn1_1_pred <- h2o.predict(nn1_1, h2o.test1)
nn6_1_pred <- h2o.predict(nn6_1, h2o.test6)
ACC(table(as.vector(nn1_1_pred$predict), test1$volact))
F1(table(as.vector(nn1_1_pred$predict), test1$volact))
ACC(table(as.vector(nn6_1_pred$predict), test6$volact))
F1(table(as.vector(nn6_1_pred$predict), test6$volact))

#################################################################
# Neural Net: Calculated with Country Dummies
#################################################################

nn1_1c <- h2o.deeplearning(
  x = xnames1c,
  y = "volact",
  training_frame = h2o.train1c,
  validation_frame = h2o.test1c,
  activation = "Rectifier",
  hidden = c(200, 200, 100),
  epochs = 10,
  rate = .005,
  loss = "CrossEntropy",
  export_weights_and_biases = TRUE,
  seed = 1000000
)

nn6_1c <- h2o.deeplearning(
  x = xnames6c,
  y = "volact",
  training_frame = h2o.train6c,
  validation_frame = h2o.test6c,
  activation = "Rectifier",
  hidden = c(200, 200, 100),
  epochs = 10,
  rate = .005,
  loss = "CrossEntropy",
  export_weights_and_biases = TRUE,
  seed = 1000000
)

summary(nn1_1c)

summary(nn6_1c)

nn1_1c_pred <- h2o.predict(nn1_1c, h2o.test1c)
nn6_1c_pred <- h2o.predict(nn6_1c, h2o.test6c)
ACC(table(as.vector(nn1_1c_pred$predict), test1c$volact))
F1(table(as.vector(nn1_1c_pred$predict), test1c$volact))
ACC(table(as.vector(nn6_1c_pred$predict), test6c$volact))
F1(table(as.vector(nn6_1c_pred$predict), test6c$volact))

#################################################################
# Random Forest: Calculated on two years separately
#################################################################

rf1_1 <- h2o.randomForest(
  training_frame = h2o.train1,
  validation_frame = h2o.test1,
  x = xnames1,
  y = "volact",
  model_id = "forest1",
  ntrees = 300,
  max_depth = 50,
  stopping_rounds = 2,
  score_each_iteration = F,
  seed = 1000000)

rf6_1 <- h2o.randomForest(
  training_frame = h2o.train6,
  validation_frame = h2o.test6,
  x = xnames6,
  y = "volact",
  model_id = "forest1",
  ntrees = 300,
  max_depth = 50,
  stopping_rounds = 2,
  score_each_iteration = F,
  seed = 1000000)

summary(rf1_1)

summary(rf6_1)

rf1_1_pred <- h2o.predict(rf1_1, h2o.test1)
rf6_1_pred <- h2o.predict(rf6_1, h2o.test6)
ACC(table(as.vector(rf1_1_pred$predict), test1$volact))
F1(table(as.vector(rf1_1_pred$predict), test1$volact))
ACC(table(as.vector(rf6_1_pred$predict), test6$volact))
F1(table(as.vector(rf6_1_pred$predict), test6$volact))

#################################################################
# Random Forest: Calculated with Country Dummies
#################################################################

rf1_1c <- h2o.randomForest(
  training_frame = h2o.train1c,
  validation_frame = h2o.test1c,
  x = xnames1c,
  y = "volact",
  model_id = "forest1",
  ntrees = 300,
  max_depth = 50,
  stopping_rounds = 2,
  score_each_iteration = F,
  seed = 1000000)

rf6_1c <- h2o.randomForest(
  training_frame = h2o.train6c,
  validation_frame = h2o.test6c,
  x = xnames6c,
  y = "volact",
  model_id = "forest1",
  ntrees = 300,
  max_depth = 50,
  stopping_rounds = 2,
  score_each_iteration = F,
  seed = 1000000)

summary(rf1_1c)

summary(rf6_1c)

rf1_1c_pred <- h2o.predict(rf1_1c, h2o.test1c)
rf6_1c_pred <- h2o.predict(rf6_1c, h2o.test6c)
ACC(table(as.vector(rf1_1c_pred$predict), test1c$volact))
F1(table(as.vector(rf1_1c_pred$predict), test1c$volact))
ACC(table(as.vector(rf6_1c_pred$predict), test6c$volact))
F1(table(as.vector(rf6_1c_pred$predict), test6c$volact))

#################################################################
#################################################################
```

Table 2: Included Variables

	Pre-pre-processing (done in Stata)	Pre-processing
Round		is binary
Female	1 if gndr == 2, 0 otherwise	is binary
Year of Birth		scaled
Years in Education		scaled
Area of Living		scaled
Marital Status	ESS1: 1 if marital == 1, 0 o. ESS6: 1 if marsts == 1 or 2 or if rshpsts == 1 or 2, 0 o.	is binary
Children	1 if chldhm == 1, otherwise 0	is binary
Houseperson	1 if mainact == 8	is binary
Weekly Work Hours		scaled
Attend Church	inverse*	scaled
Meet Social Contacts		scaled
Social Trust	(ppltrst + 1) + (pplfair + 1) + (pplhlp + 1)	scaled
Tolerance	inverse*	scaled
Self Realization	inverse*	scaled
Solidarity	inverse*	scaled
Political TV Consumption		scaled
Total TV Consumption		scaled
Political Interest	inverse*	scaled
Trust in Executive	average of trstlgl and trstplc	scaled
Trust in Legislative	average of trstplt and trstprl	scaled
Trust in European Parliament		scaled
Satisfaction with Democracy		scaled
Country		turned into dummy variables
Voluntary Active	ESS1: 1 if one of the asked participation opportunities was answered with yes, 0 o. ESS6: 1 if the respondent answered at least with ”less often” than every 6 months (answers 1-5), 0 o.	is binary
Comment	all variables were cleaned: ”refusal”, ”don’t know”, ”not applicable” etc. set to missing *inverse: turned the scale to make it more intuitive to interprete	if binary: missings replaced with modal if scaled: missings replaced with mean (before scaling)

References

[Armingeon, 2007] K. Armingeon. 2007. Political participation and associational involvement. In J. W. van Deth, editor, Citizenship and Involvement in European Democracies: A Comparative Analysis, chapter 14, pages 358–383. Routledge, London.

[Badescu and Neller, 2007] G. Badescu and K. Neller. 2007. Explaining associational involvement. In J. W. van Deth, editor, Citizenship and Involvement in European Democracies: A Comparative Analysis, chapter 7, pages 158–187. Routledge, London.

[Cook, 2017] D. Cook. 2017. Practical Machine Learning with H2O. O'Reilly Media, Sebastopol.

[Dalton, 2014] R. J. Dalton. 2014. Citizen Politics: Public Opinion and Political Parties in Advanced Industrial Democracies. SAGE Publications, Los Angeles.

[Dangeti, 2017] P. Dangeti. 2017. Statistics for Machine Learning. Build supervised, unsupervised, and reinforcement learning models using both Python and R. Packt Publishing, Birmingham.

[Dekker and van den Broek, 1998] P. Dekker and A. van den Broek. 1998. Civil society in comparative perspective: Involvement in voluntary associations in north america and western europe. Voluntas: International Journal of Voluntary and Nonprofit Organizations, 9(1):11–38.

[Dr. Wiley, 2016] Joshua F. Dr. Wiley. 2016. R Deep Learning Essentials. Build automatic classification and prediction models using unsupervised learning. Packt Publishing, Birmingham.

[ESS, 2002] ESS. 2002. Ess round 1: European social survey round 1 data. NSD - Norwegian Centre for Research Data, Norway – Data Archive and distributor of ESS data for ESS ERIC. Data file edition 6.5.

[ESS, 2012] ESS. 2012. Ess round 6: European social survey round 6 data. NSD - Norwegian Centre for Research Data, Norway – Data Archive and distributor of ESS data for ESS ERIC. Data file edition 2.3.

[Gabriel and Völkl, 2008] O. W. Gabriel and K. Völkl. 2008. Politische und soziale partizipation. In O. W. Gabriel and S. Kropp, editors, Die EU-Staaten im Vergleich, pages 268–298. VS Verlag, Wiesbaden.

[Lantz, 2015] B. Lantz. 2015. Machine Learning with R. Second Edition. Packt Publishing, Birmingham.

[Morales and Geurts, 2007] L. Morales and P. Geurts. 2007. Associational involvement. In J. W. van Deth, editor, Citizenship and Involvement in European Democracies: A Comparative Analysis, chapter 6, pages 135–157. Routledge, London.

[van Deth and Maloney, 2015] J. W. van Deth and W. A. Maloney. 2015. Associations and associational involvement in europe. In J. M. Magone, editor, Routledge Handbook of European Politics, chapter 45, pages 826–842. Routledge, Abingdon.

[van Deth, 1997] J. W. van Deth. 1997. Private Groups and Public Life. Routledge, London.

[Verba et al., 1995] S. Verba et al. 1995. Voice and Equality: Civic Voluntarism in American Politics. Harvard University Press, Cambridge, MA.

[Zukin et al., 2006] C. Zukin et al. 2006. A New Engagement?: Political Participation, Civic Life, and the Changing American Citizen. Oxford University Press, New York.

Statement of Authorship

I hereby declare that the paper presented is my own work and that I have not called upon the help of a third party. In addition, I affirm that neither I nor anybody else has submitted this paper or parts of it to obtain credits elsewhere before. I have clearly marked and acknowledged all quotations or references that have been taken from the works of others. All secondary literature and other sources are marked and listed in the bibliography. The same applies to all charts, diagrams and illustrations as well as to all Internet resources. Moreover, I consent to my paper being electronically stored and sent anonymously in order to be checked for plagiarism. I am aware that the paper cannot be evaluated and may be graded ”failed” (”nicht ausreichend”) if the declaration is not made.

Larissa Haas
June 6, 2018