Based on the forthcoming:

Blaauw, F. J., de Vos, S., Wanders, R. B. K., de Jonge, P., Aiello, M., Penninx, B., Wardenaar, K., Emerencia, A. C., (2017). Applying machine learning to patient self-report data for predicting adverse depression outcomes. *In preparation*.

Chapter 7

Machine Learning for Precision Medicine in Psychopathology Research

Depression affects 14.9 % to 19 % of all people during their lifetime (Bijl et al., 1998; Bromet et al., 2011; Kessler et al., 2011) and is a substantial public health problem, causing tremendous human suffering and costs to society. Therefore, improving treatment and early detection of depression is an absolute priority. However, despite numerous investments, progress in depression research has stagnated: we still know very little about the underlying mechanisms, and in practice clinicians struggle to determine a patient's prognosis and optimal treatment (Kapur et al., 2012; Whooley, 2014). As such, prediction of above-threshold depressive symptomatology has so far proved to be difficult. Some general risk factors of unfavorable course or outcomes have been identified, such as depression severity (Plaisier et al., 2010), trauma (Stevens et al., 2013), personality (Wardenaar, Conradi, Bos, & de Jonge, 2014), comorbidity (Wardenaar, van Loo, et al., 2014), or genetics (Hyde et al., 2016). In addition, protective factors such as social support (Lara, Leader, & Klein, 1997), coping skills (Kuehner & Huffziger, 2012), and personality (Wardenaar, Conradi, et al., 2014) have been identified. However, current models and guidelines lack the specificity to differentiate between patients with different prognostic risk profiles, which makes them of limited use for clinicians (e.g., Galfalvy, Oquendo, & Mann, 2008; Hetrick, Simmons, Thompson, & Parker, 2011; Kuiper, McLean, Fritz, Lampe, & Malhi, 2013; Perlis, 2014).

One likely reason for the stagnation in the development of prediction models so far is that prognostic studies have so far mostly relied on the use of traditional statistics, using significance testing to evaluate the predictive effect of individual predictors. Apart from well-documented problems with traditional null-hypothesis testing (e.g., Aarts, Winkens, & van Den Akker, 2012; Cox, 1958), a more general conceptual problem with this approach is that it is focused on testing prognostic effects rather than on optimizing prediction. The latter is hard to do with traditional statistics and requires a different approach rooted in statistical learning. In mathematical statistics and computational science, many techniques have been developed that can estimate optimized prediction models. By using learning algorithms, such techniques can identify the model configuration with the smallest outcome classification error (for dichotomous outcomes) or the smallest discrepancy between estimated and observed outcome values (for continuous outcomes). Furthermore, such techniques can be evaluated and selected such that they perform optimally on new, unseen data, and as such generalize well to future data. Interestingly, many of such supervised machine learning techniques allow for regularization and thus enable the inclusion of large quantities of predictors, making them an ideal match for the large datasets that are increasingly becoming available. Moreover, regularization allows for the analysis of datasets that contain more predictors than observations. Machine learning is a promising field for the development of more accurate and useful prediction models.

Some previous work has been conducted in the field of depression research using machine learning to estimate prediction models. For instance, studies have looked at prediction of treatment outcome (e.g., Andreescu et al., 2008; Jain et al., 2013), risk of suicide (e.g., Baca-García et al., 2007; Kessler et al., 2015; Seemüller et al., 2009), hospitalization (Baca-García et al., 2006), mental health service use (Cairney et al., 2014), and treatment resistance (Perlis, 2013). However, these studies used very particular samples (e.g., Kessler et al., 2015, used the army Star-D data set, which only consists of (ex-)military), used few predictors and outcomes, and each only used one particular machine learning technique, for example, tree-based models. This makes it hard to gain a general idea of the added value of machine learning techniques and the comparative usefulness of different machine learning strategies in the specific field of depression research. A systematic investigation and comparison of different machine learning techniques to estimate prediction models in depression is currently lacking, making it hard for researchers to make informed choices about which techniques to use. In addition, different machine learning approaches yield different kinds of models (e.g., additive vs. multiplicative), each with different implications for the way the risk of an outcome is calculated.

To fill this knowledge gap, the goal of this study is to evaluate the usefulness of a range of machine learning algorithms in developing clinically useful prediction models for adverse depression outcomes. To accomplish this, a range of machine learning techniques (e.g., classification trees, random forests, support vector machines, naïve Bayesian classifiers, and ensemble techniques) and more traditional statistical methods (e.g., logistic regression) are used to generate optimized prediction models for providing dichotomous outcomes (output). Machine learning is a well suited technology for creating such classifiers, and has the potential to provide a new insight into depression and the prediction thereof. Our machine learning classifiers are based on a large pool of clinically useful baseline input features. We select this pool of clinically relevant baseline features in a generic screening step, in which a subset of the most influential features is selected prior to *training* the machine learning algorithms. The notion of 'training an algorithm' is used to refer to the step where we use the data to fit the parameters in the machine learning algorithms.

Next, we evaluate the ability of the classifiers to correctly classify patients with regard to our outcome, and predictive performance will be compared across models using data of a follow up study. Data from this follow up study is only used as output, and the machine learning algorithms are trained only on features available at baseline, justifying the term 'prediction'. The machine learning algorithms will be evaluated on their ability to accurately predict whether an individual is expected to reach above clinical threshold depressive symptomatology at follow up (according to the Inventory of Depressive Symptomatology [IDS; A. Rush et al., 2003; A. J. Rush et al., 2006]).

7.1 Methods

The machine learning classifiers in this study are based on the data of the Nederlandse Studie naar Depressie en Angst (NESDA). NESDA is a longitudinal cohort study that focuses on the long-term course of depression and anxiety disorders in the Netherlands (Penninx et al., 2008). The NESDA data set used in the present work consists of a baseline study and a follow-up study two years later, using the same or comparable measurement instruments. The data set comprises various tools to measure depression and anxiety, such as measures from the Composite International Depression Interview (CIDI; World Health Organization & Others, 1993), IDS, and the Mood and Anxiety Symptom Questionnaire (MASQ; Wardenaar et al., 2010). Furthermore, it contains self-report data about somatic complaints. Lastly, several demographical features per participant are available. The complete list of features used as input is provided in Table 7.1. The full list of questions for each instrument are listed in Table C.1 in Appendix C.

One of the goals of the present work is to derive machine learning based classifiers that could be used in clinical practice. As such, we used only features that are (i) easy to collect in clinical practice (e.g., self-report questions, demographical information, etc.), (ii) available at baseline, and (iii) were completed by most participants. This resulted in a total of 128 features.

We performed a feature selection step to reduce the data set to a subset contain-

Table 7.1: All questionnaires and other data sources used in the feature selection module, for a specific list of the used features / questions see Appendix C.

Instrument[a]	**Q**[b]	**Description**	**Reference**
Demographics	1 to 12	Demographic data	N/A
Soft and hard drugs	13	Drug usage	N/A
Alcohol Use Disorder Identification Test (AUDIT)	14, 15	Diagnosis alcohol disorder / abuse	World Health Organization and Others (1993)
MASQ	16 to 18	Mood and anxiety	Wardenaar et al. (2010)
Mood Disorder Questionnaire (MDQ) Bipolar symptoms	19	Bipolar disorders	Shahid, Wilkinson, Marcu, and Shapiro (2011)
VierDimensionale KlachtenLijst (4DKL)	20 to 37	General somatic and psychological complaints	Terluin (1996)
4DKL	38 to 40	Physical complaints	Terluin (1996)
IDS	41 to 67	Depressive Symptomatology	A. J. Rush, Carmody, and Reimitz (2000)
Beck Anxiety Inventory (BAI)	68 to 71	Anxiety	Spielberger, Gorsuch, Lushene, and Vagg (1983)
Neuroticism-Extraversion-Openness Five-Factor Inventory (NEO-FFI)	72 to 92	Personality	Costa and McCrae (1992)
Chronic diseases / conditions	93, 94	Chronic diseases	N/A
CIDI Depression	95 to 106	depression diagnosis	World Health Organization and Others (1993)
CIDI Anxiety	107 to 128	anxiety disorder diagnosis	World Health Organization and Others (1993)

Note:
[a] We used a combination of raw questionnaire items and computed, derived variables, such as sum scores and average scores.
[b] Question id, correspond to the values used in Table C.1 on page 221.

ing its twenty most predictive variables. Feature selection can improve the prediction performance and the training speed of our algorithms (Guyon & Elisseeff, 2003). Furthermore, reducing the number of features also reduces the number of questions a patient needs to answer during a clinical interview. The set of features actually used as input for the machine learning algorithms (i.e., the twenty features remaining after feature selection) is provided in Table 7.2. For the analysis we converted the categorical questions in the questionnaires to binary dummy variables.

The outcome variable we used is a construct we call 'above threshold clinically depressive symptoms'. Depressive symptoms in this case are collected and evaluated using the IDS questionnaire. We used the IDS as it measures all depression criteria and symptom domains as laid out by the Diagnostic and Statistical Manual of Mental Disorders (DSM). We used a threshold of 'at least moderate depressive symptoms,' which translates to an IDS score of > 25 (A. Rush et al., 2003; van Borkulo et al., 2015). This dichotomization makes the outcome binary, and the outcome is 'one' when a participant reports clinically relevant / above-threshold levels of depressive symptoms at follow-up and 'zero' otherwise.

Table 7.2: Overview of the features selected using the elastic net feature selection.

	Instrument	Feature	Coefficient	Type
1	IDS	I see myself as equally worthwhile and deserving as other people	−0.80	Dichotomous
2	IDS	It takes me several seconds to respond to most questions and I'm sure my thinking is slowed	0.69	Dichotomous
3	NEO-FFI	Neuroticism (anxiety)	0.66	Discrete
4	NEO-FFI	Extraversion (total score)	−0.52	Discrete
5	IDS	I never take longer than 30 minutes to fall asleep	−0.46	Dichotomous
6	IDS	There is no change in my usual appetite	−0.45	Dichotomous
7	NEO-FFI	Openness (unconventionality)	−0.44	Discrete
8	4DKL	Somatization (trychotomization)	0.43	Discrete
9	IDS	I awaken more than once a night and stay awake for 20 minutes or more, more than half the time	0.42	Dichotomous
10	IDS	I rarely get a feeling of pleasure from any activity	0.38	Dichotomous
11	NEO-FFI	Conscientiousness (orderliness)	−0.37	Discrete
12	IDS	I enjoy pleasurable activities just as much as usual	−0.34	Dichotomous
13	NEO-FFI	Openness (aesthetic interest)	0.33	Discrete
14	4DKL	Somatization score	0.32	Discrete
15	N/A	Number of chronic diseases1	−0.30	Dichotomous
16	IDS	I feel anxious (tense) more than half the time	0.27	Dichotomous
17	NEO-FFI	Agreeableness (nonantagonastic orientation)	−0.24	Discrete
18	MASQ	Positive affect score	−0.24	Discrete
19	NEO-FFI	Extraversion (positive affect)	−0.22	Discrete
20	MDQ	Total score	0.21	Discrete

Note: The coefficient column denotes the coefficients as retrieved using the elastic net.

The baseline data set consisted of 2 981 'healthy' and clinically depressed subjects aged (at baseline) between 18 to 65 (median = 43, mean = 41.9, standard deviation [SD] = 13.1). Of the participants, 66.4 % was female. From the total set of 2 981 individuals, 87.1 % (2 596 people) participated in the follow-up study, of which 4.9 % completed the questionnaires needed for our outcome variable. We only considered complete cases, that is, all patients that did not have a follow-up measurement or had missing data in any of the other variables were excluded from the set. This selection step resulted in a final data set of 2 174 individuals.

7.1.1 The Machine Learning Procedure

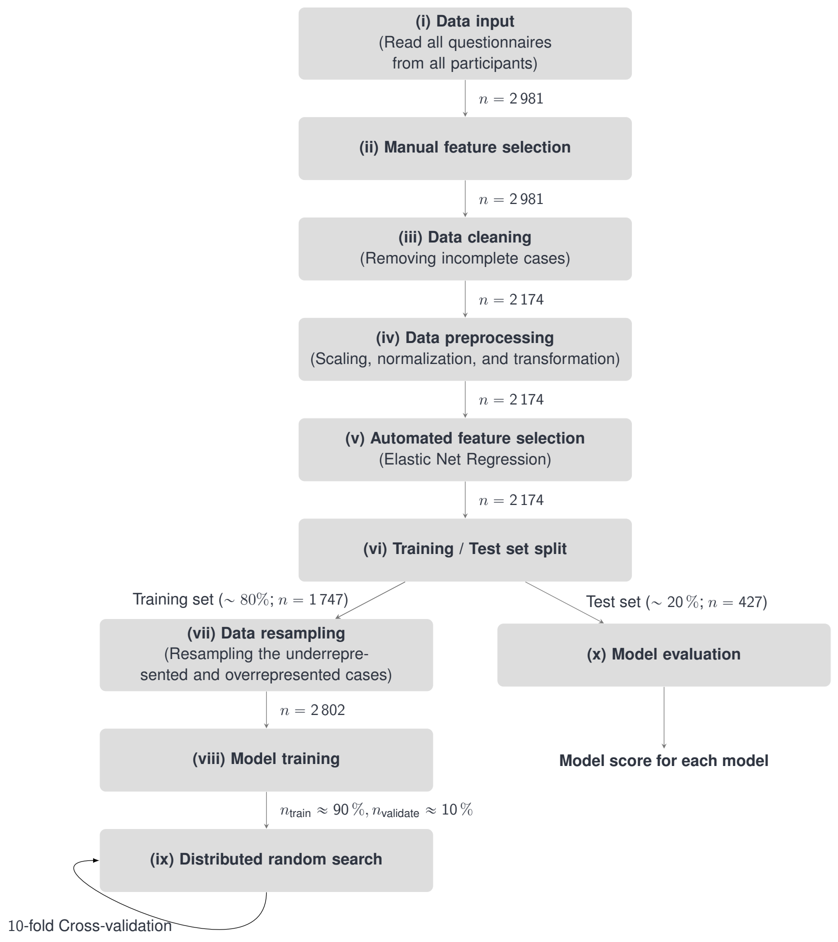

(i) Data input
(Read all questionnaires from all participants)
$n = 2\,981$
(ii) Manual feature selection
$n = 2\,981$
(iii) Data cleaning
(Removing incomplete cases)
$n = 2\,174$
(iv) Data preprocessing
(Scaling, normalization, and transformation)
$n = 2\,174$
(v) Automated feature selection
(Elastic Net Regression)
$n = 2\,174$
(vi) Training / Test set split
Training set ($\sim 80\%$; $n = 1\,747$)
Test set ($\sim 20\%$; $n = 427$)
(vii) Data resampling
(Resampling the underrepresented and overrepresented cases)
$n = 2\,802$
(viii) Model training
$n_{\text{train}} \approx 90\%, n_{\text{validate}} \approx 10\%$
(ix) Distributed random search
10-fold Cross-validation
(x) Model evaluation
Model score for each model

Figure 7.1: The used machine learning pipeline.

The machine learning procedure we applied was as follows (following the order depicted in Figure 7.1). In the first two steps, we read the data from the NESDA questionnaires. First (Step (i)) the data for each questionnaire was read from the SPSS files supplied by NESDA. Then (Step (ii)), we manually selected a subset of all of the questionnaires and questionnaire items that were considered relatively easy to collect clinically, and were relevant for the current study. A questionnaire was considered 'easy to collect clinically' when its questions could be answered instantly by the participant without any further (medical) testing. Furthermore, for the input features we only selected the questionnaire items that were available at baseline. The outcome was the only variable selected from a follow-up questionnaire. In the next step (Step (iii)), the data set was cleaned. All participants that had missing data are removed in this step, and only complete cases were used to fit the machine learning models. All data from the relevant questionnaires was collected and aggregated features were calculated (i.e., severity measures and sum scores).

In Step (iv), we performed data preprocessing. We preprocessed all variables by scaling and normalizing them. We converted our categorical variables into binary dummy variables by using a *one-hot encoding* procedure (e.g., Harris & Harris, 2012, p. 123). With one-hot encoding, a number of binary variables is created, one for each category in the categorical variable. For example, a categorical variable with three categories is encoded using three new variables, and when a subject belongs to a certain category, the corresponding one-hot variable entry lists a one, and a zero if it belongs to a different category.

In Step (v), we performed screening / feature selection to reduce the number of features used in the machine learning analysis. From the initial set of features, a subset was selected that will be used in the analysis. These features were selected using an elastic net regression (inspired by the work of Chekroud et al., 2016). Fitting the elastic net model was done using all scaled raw and converted variables as input and the variable to predict as output. Elastic net regression takes care of penalizing small coefficients and causes only the most predictive features to remain (based on the absolute value of the coefficient). From these features, we selected the top-twenty features that best predicted the outcome.

In Step (vi), we split the complete data set into two subsets: a training set and a test set. The used training set contained approximately $80\,\%$ of the data, with the remaining $20\,\%$ contained in the test set. We used an $80\,\%$ training set to have enough data to train the algorithms, whilst still having a large number of observations to test the algorithms. The two sets were created by sampling $2\,174$ values from a binomial distribution with a probability of 0.2 of being one (i.e., belonging to the test-set). We performed this sample split procedure in order to evaluate the performance of the algorithms on an out-of-sample part of the data.

After splitting our data into a test and training set, we applied data resampling on the training set in Step (vii). In this resampling step, we increased the observations that had a positive output (i.e., had the label '1'), and reduced the observations that had a negative output (i.e., had the label '0'). This step was needed because of the imbalanced nature of our data (i.e., about $7.4\,\%$ of the samples in the data set were labeled 'clinically depressed'; had label '1'). We will elaborate on this resampling step later (in Section 7.1.3).

In Step (viii), we performed the actual training of the machine learning classifiers. We used the following algorithms: (A) Decision Tree, (B) Stochastic Gradient Descent, (C) Random Forest, (D) Constant Dummy, (E) Random Dummy, (F) Support Vector Machine, (G) Gradient Boosting, (H) Logistic Regression, and (I) Bernoulli Naive Bayes. Two dummy algorithms were included as a baseline, one which always predicted the value zero, and one which performed a random classification. We adhered to a two-step procedure for training the algorithms. Besides training the algorithms to learn the parameters (or coefficients) that were used for prediction, we also implemented a data-adaptive approach for optimizing the so-called *hyperparameters* (or tuning-parameters, as performed in Step (ix)). Hyperparameters are parameters that are not optimized when training an algorithm, but serve as mere knobs to tune the algorithm itself (e.g., decision boundaries or regularization parameters can be considered hyperparameters). By training an algorithm with different combinations of hyperparameters, we can data-adaptively optimize these parameters as well.

In Step (ix), each algorithm was trained on the training set and internal validation was performed by means of 10-fold cross-validation (CV), while performing a random search procedure to optimize the hyperparameters. In hyperparameter optimization process, different hyperparameter configurations are evaluated for each of the machine learning algorithms. As most parameter spaces have infinitely many parameter options testing the whole space is impossible and a subset of parameters needs to be selected. Random search is a method in which a hyperparameter value is randomly drawn from a probability distribution that can be specified separately for each of the hyperparameters (Bergstra & Bengio, 2012). One specifies a number of iterations and draws a set of hyperparameters from their corresponding distributions in each iteration. We used the random search approach as an alternative to the traditional grid search procedure (in which a grid of hyperparameters is tested exhaustively) to be more flexible and efficient in the hyperparameter selection procedure (the random selection procedure is further elaborated in Section 7.1.2). For each algorithm we stored the hyperparameter configurations that best performed on the cross-validated training set.

Finally (Step (x)), we used the test set to evaluate each of the algorithms in order to see how well they perform on and generalize to out of sample data. From this evaluation step, several model scores are derived for evaluating the models.

We developed this pipeline / machine learning algorithm training procedure as open-source software[1]. After providing the set of questionnaires to use and performing a manual feature selection step (Step (ii)), the application performs several

[1]Source available at `https://github.com/compsy/machine-learning-depression`.

steps relevant for fitting the machine learning models automatically (e.g., data reading, cleaning and preprocessing, feature selection, and algorithm training and evaluation). An overview of the whole procedure applied by the software is depicted in Figure 7.1. This software is currently focused towards data retrieved from the NESDA data set (that is, it provides handles to retrieve data automatically from the provided data sets), but it could easily be generalized to perform the same analysis on different data sets.

7.1.2 Random Hyperparameter Search Procedure

Hyperparameter search is a method to optimize the set of hyperparameters (or tuning-parameters) of machine learning algorithms. First, a set of parameters is defined containing all different combinations of a subset of a hyperparameter space for each hyperparameter. This space can be either a continuous distribution or a discrete set of options or integers. Every parameter combination is used to train an algorithm and is evaluated using CV. This means that the number of hyperparameter configurations to test grows exponentially. For instance, if a machine learning algorithm would have only a single hyperparameter $A \in \mathcal{A} = \{1, 2, 3, 4, 5\}$, this results in five different configurations. However, if the algorithm also has a hyperparameter $B \in \mathcal{B}$ and $C \in \mathcal{C}$, each also of length five, the number grows exponentially to 5^3 different configurations. Since k-fold CV is used to evaluate the different hyperparameter configurations, the number of evaluations to perform equals $k \cdot \prod_{h=0}^{H} m_h$, where k is the number of folds in the k-fold CV, h is the number of hyperparameters, and m_h the number of values to test for hyperparameter h.

To be able to test a relatively large number of hyperparameters, we designed the application in such a way that it allows for computational scaling in both a vertical direction (CPU speed) and horizontal direction (parallelism and distributed computing). We implemented the random hyperparameter search using a *MapReduce approach*. MapReduce is a well-known programming model to work with large amounts of data or with computationally intensive tasks (J. Dean & Ghemawat, 2008). By distributing these calculations over various computational nodes (mapping), and combining the results at the end (reducing), calculations can be performed in a highly distributed and parallelized environment.

Our MapReduce procedure is as follows. First, in the mapping phase, each instance of our application performs 100 iterations of random search for each algorithm (i.e., the 'worker' nodes). In each iteration, a value for each hyperparameter is drawn from a pre-specified distribution (continuous or discrete) of possible hyperparameter values. Then, the algorithms are trained and evaluated using 10-fold CV and their performance is stored. After the application has finished training these algorithms using the 100 iterations of random search, the application selects the best configuration and uploads this configuration and fitted classifier to a centralized storage service. This centralized storage service could be any network attached storage solution. For example, in the present work we used the Amazon Simple Storage Service (S3) as centralized storage solution because of its ease to use and global availability. After all instances of the application have uploaded their optimal configuration, a separate evaluation instance is started to retrieve the different configurations from the centralized storage solution and to evaluate them (i.e., the 'evaluator' node). In this reducing step, all candidate optimal configurations are compared, and a single optimal configuration for each algorithm is selected. This configuration is then used to assess the performance on the testing set (Step (x) in Figure 7.1). This way of distributed computing has the advantage that no specialized hardware is required to run the application. Any computer running the correct Python and R versions, and that has the correct libraries installed, can run the implementation and as such contribute to the computation. A schematic of this procedure is provided in Figure 7.2.

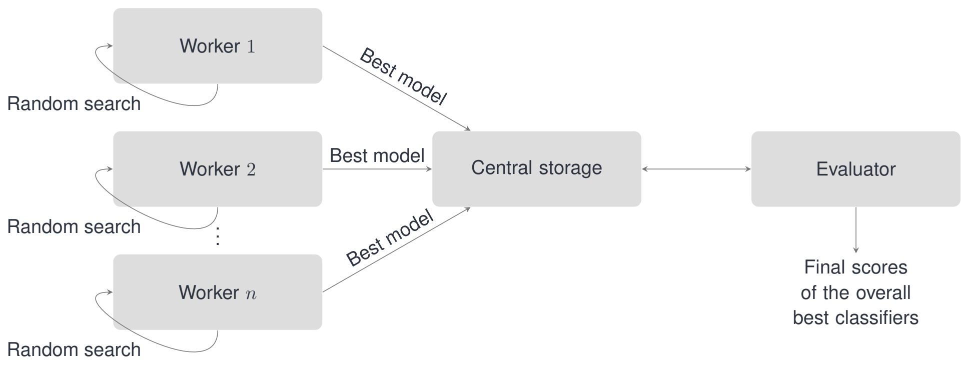


Figure 7.2: MapReduce procedure for finding the best performing classifiers.

7.1.3 Synthetic Minority Over-sampling

Because of the highly imbalanced data set (i.e., less than 8 % of the participants is classified to be clinically depressed at followup), we performed a data resampling step. In this resampling step, we performed a combination of *over*sampling and *under*sampling on the training set. In oversampling, the underrepresented class (chronically depressed participants) are resampled, introducing new instances of this minority class. Undersampling is the opposite, and removes cases from the majority class (the 'healthy' individuals). The combination of both oversampling and undersampling causes the training set to be approximately balanced with both positive and negative outcomes (Kuhn & Johnson, 2013). Note that we only performed this resampling step on the training part of the data set, and not on the test set. This way, the test set remains a reliable out-of-sample set to evaluate our classifiers on.

To perform the resampling step, we applied the Synthetic Minority Over-sampling Technique (SMOTE; Chawla, Bowyer, Hall, & Kegelmeyer, 2011) in combination with the Edited Nearest Neighbors (ENN; Wilson, 1972) technique. SMOTE introduces synthetic observations in the data based on a number of nearest neighbors to create that observation. The ENN technique reduces the majority by only using the neighbors that contribute to the estimation of a decision boundary. Before this resampling step, the training data had 7.6 % positive outcomes, after resampling this was better balanced, and was approximately 57.3 %.

7.1.4 Performance Measures

To evaluate the performance of our learners, we used five different performance measures (and a combined average of each of them). Firstly, we used the *F1-score*, which is defined as the harmonic mean between precision and recall

$$\text{F1-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

The F1-score takes its values between zero and one, one being perfect precision and recall. Secondly, we used the *Accuracy*. Accuracy measures the ratio between true positives (TPs), the cases classified as true and actually being true (in our case, the cases classified with above clinical levels of depression that actually ended up with above clinical levels of depression), true negatives (TNs), the cases classified as false and actually being false (in our case, the cases classified as sub clinical threshold levels of depression that indeed did not experience clinical levels of depression), and the total number of predictions. As such, it is defined as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

Similar to the F1-score, accuracy ranges between zero and one, one being a perfect accuracy. Although the accuracy metric is known to be inaccurate on class-imbalanced data sets (the accuracy paradox; Valverde-Albacete & Peláez-Moreno, 2014), we chose to include it as the use of the accuracy measure is still widespread. Thirdly, we used the Area Under the receiver operating characteristic (ROC) curve. The ROC is a method to visualize the performance of a classifier, based on the trade-off between the true and false positive rates. The curve itself is generated by iterating over different cut-off values / thresholds for the classifier to predict the positive versus the negative class. The area under the curve (AUC) is a scalar parameter of this ROC metric, representing the total area covered by this ROC curve (Fawcett, 2006). The AUC has the advantage that it is insensitive to class imbalance, as it only relies on the ratios of true and false positives. Similar to the earlier metrics, the AUC is valued between zero and one, one being the optimal AUC. Fourthly we calculated the *geometric mean* for each of the classifiers. The geometric mean maximizes the accuracy of both the positive and negative class, while keeping them balanced. It is implemented as

$$\text{Geometric mean} = \sqrt{\frac{\text{TP}}{(\text{TP} + \text{FN})} \times \frac{\text{TN}}{(\text{TN} + \text{FP})}}.$$

Also the geometric mean is a measure ranged between zero and one, one being the optimal geometric mean. Lastly, we implemented Cohen's Kappa (or κ; Cohen, 1960). This metric was originally created to quantify the level of agreement between two independent judges observing a phenomenon (Ben-David, 2007). In our case, one of these judges is represented by the classifier, and the other by the observed truth. It is implemented as

$$\text{Kappa score} = \frac{p_o - p_e}{1 - p_e},$$

where p_o is the empirical proportion of outcomes in which the observed classes equaled the predicted classes, and p_e is the prior proportion of outcomes for which agreement is expected by chance (Cohen, 1960). In this case, p_e is estimated by from the class labels. This performance measure ranges from minus one, meaning complete dissimilarity between predicted and observed classes, through zero, meaning random classification, to one, meaning a complete agreement.

7.1.5 Application and Implementation Details

The complete machine learning process consisted of several steps and encompassed a number of applications and software packages. Firstly, we used a number of applications to investigate the data set and to generate several summary statistics about the variables available. We used Tableau (Version 9.0.2; Tableau Software, 2018) to perform an initial inspection and WEKA (Version 3.8.0; Hall et al., 2009) to generate rudimentary machine learning classifiers. After this initial inspection, we built, trained, and evaluated the actual machine learning classifiers using the programming languages R (R Development Core Team, 2008) and Python (Version 3.6; Python Software Foundation, 2018). R is a programming language that mainly focuses on statistical computation. Python is a general purpose programming language which has gained popularity in the data science community over the past years. A byproduct of this popularity is that a large number of libraries designed for scientific purposes are available.

For both R and Python, we used several packages to aid us in the development of our application. All packages are depicted in Figure 7.3 and elaborated next. Data were first imported using the 'read.spss' R-function from the R 'Foreign' package (Version 0.8-66; R Development Core Team et al., 2017). This R functionality was exposed to the Python code using the 'rpy2' Python library (Version 2.8.2; Gautier & Rpy2 contributors, 2018). Rpy2 is a Python library that enables developers to interface with R functions from Python. All of the machine learning classifiers were created in Python. In order to perform the actual analysis we used several Python libraries. The machine learning algorithms we used were created using a Python library named Scikit-learn (Version 0.18; Pedregosa et al., 2012). The Scikit-learn library provides implementations of several machine learning libraries, and provides tools useful when training machine learning algorithms, for example feature selection, data transformation, CV, and classifier evaluation. For balancing our data set (the resampling procedure, Step (vii) in Figure 7.1), we used various tools from the Imbalanced-learn package (Version 0.3.0; Lemaitre, Nogueira, & Aridas, 2016). For the computation of basic descriptive information, probabilistic sampling, and data structures, we used the Numpy package (Version 1.11.1; NumPy developers, 2017), the Pandas package (Version 0.18; Augspurger et al., 2018), and the Scipy package (Version 0.17; SciPy developers, 2018). We used the Boto3 package (Version 1.4.7; Amazon.com Inc., 2014) to interact with Amazon Web Services, and lastly we used Matplotlib (Version 1.5; Hunter, Dale, Firing, Droettboom, & Matplotlib development team, 2017) for visualizing the results. An overview of all used packages is provided in Figure 7.3.

7.2 Results

Table 7.2 shows several descriptive statistics of the features included in fitting the machine learning classifiers. We selected this subset of features with elastic net regression ($\alpha = 0.01$, l1-ratio $= 0.05$, $\epsilon = 0.1$). Using these features, we trained each of the machine learning algorithms. The names of the algorithms appear codified in Figure 7.4 and Figure 7.5. For this we use codes (A) for Decision Tree, (B) for Stochastic Gradient Descent, (C) for Random Forest, (D) for Constant Dummy, (E) for Random Dummy, (F) for Support Vector Machine, (G) for Gradient Boosting, (H) for Logistic Regression, and (I) for Bernoulli Naive Bayes.

We present five different performance measures for each of the best performing machine learning classifiers in Figure 7.4. Figure 7.4a presents the ROC curves for

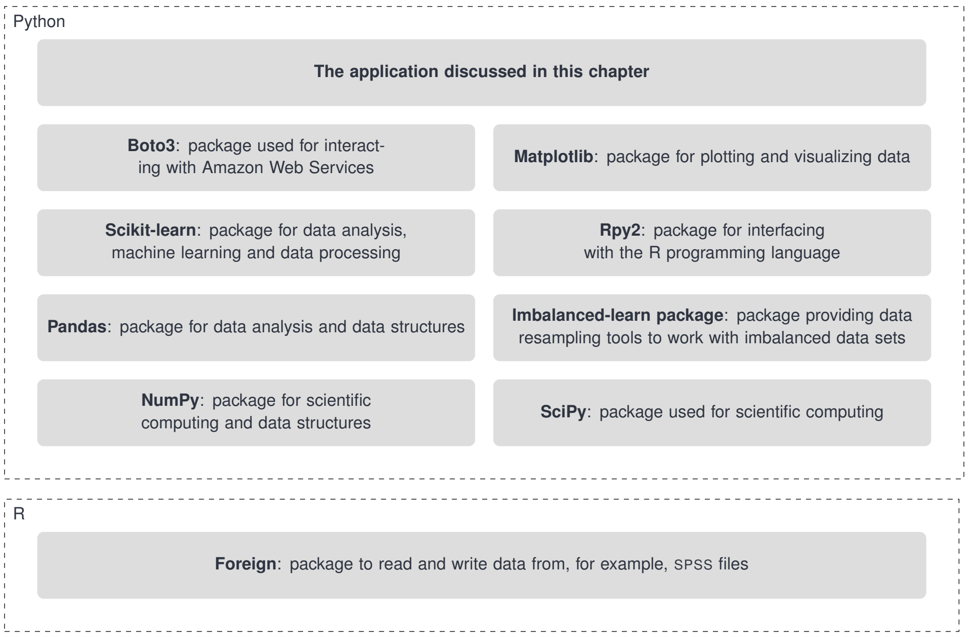
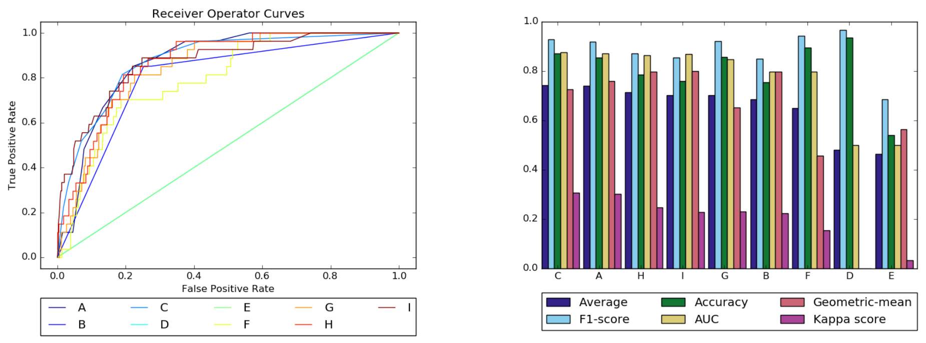

Python
The application discussed in this chapter
Boto3: package used for interacting with Amazon Web Services
Matplotlib: package for plotting and visualizing data
Scikit-learn: package for data analysis, machine learning and data processing
Rpy2: package for interfacing with the R programming language
Pandas: package for data analysis and data structures
Imbalanced-learn package: package providing data resampling tools to work with imbalanced data sets
NumPy: package for scientific computing and data structures
SciPy: package used for scientific computing
R
Foreign: package to read and write data from, for example, SPSS files

Figure 7.3: Overview of the used components and packages.



(a) Receiver operating characteristic curves.

(b) Performance measure for each classifier.

Figure 7.4: Various measures showing the performance of each machine learning algorithm.

the TP ratio. These curves show the TPs (the y-axis) versus the FPs (x-axis) whilst shifting their decision boundary. The different performance measures are presented in Figure 7.4b and Table 7.3. In this figure and table, we present five performance measures plus their average, that is, the F1-score, the accuracy, the AUC, the Geometric mean and the Kappa score. For each performance measure, a higher score corresponds to a better performing algorithm. We selected a number of metrics, as our skewed test set can influence some of their scores (Jeni, Cohn, & De La Torre, 2013).

Because of the skewed distribution of our classification labels, the constant dummy algorithm receives a relatively high accuracy and F1-score. That is, if this dummy algorithm only predicts a person to not become clinically depressed, and only 5 % of the test set becomes clinically depressed, its accuracy score will be 0.95. This biased prediction becomes visible in the other measures that take false positives and false negatives into account.

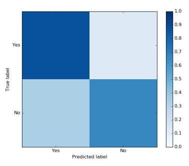
Table 7.3: Table showing the performance of each machine learning algorithm.

Algorithm	**Average**	**F1-score**	**Accuracy**	**AUC**	**Geometric mean**	**Kappa score**
Random Forest	0.742	0.928	0.871	0.876	0.726	0.307
Decision Tree	0.742	0.918	0.855	0.871	0.760	0.304
Logistic Regression	0.713	0.872	0.785	0.864	0.798	0.247
Bernoulli Naive Bayes	0.702	0.854	0.759	0.870	0.801	0.229
Gradient Boosting	0.702	0.920	0.857	0.847	0.652	0.232
Stochastic Gradient Descent	0.685	0.851	0.754	0.798	0.798	0.224
Support Vector Machine	0.649	0.944	0.895	0.797	0.457	0.154
Constant Dummy	0.481	0.967	0.937	0.500	0.000	0.000
Random Dummy	0.465	0.687	0.541	0.500	0.564	0.032

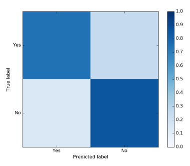
In Figure 7.5, we present the normalized confusion matrices for each of the algorithms. These confusion matrices depict the quality of the prediction in terms of true positives (upper left), false negatives (upper right), false positives (bottom left), and true negatives (bottom right). A darker color corresponds to a more frequent prediction.

7.3 Discussion and Concluding Remarks

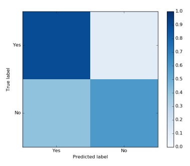
We demonstrated the implementation of a flexible and data-adaptive machine learning approach to create classifiers of above clinical threshold levels of depression based on data from the Dutch cohort study NESDA. We showed that in this particular data set (based on the average of all performance measures), the best performing algorithm was the Random Forest algorithm (average score = 0.742, accuracy = 0.871, AUC = 0.876, Geometric mean = 0.726, and Kappa = 0.307). Note that although the Random Forest algorithm performed best with respect to the average score, many other learners achieved a very similar performance.



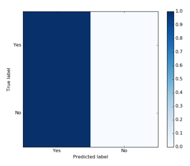
(a) Decision Tree.



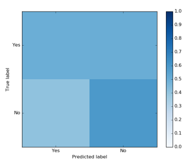
(b) Stochastic Gradient Descent.



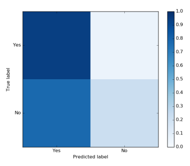
(c) Random Forest.



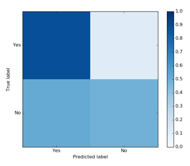
(d) Constant Dummy.



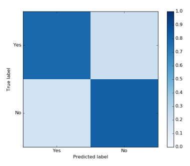
(e) Random Dummy.



(f) Support Vector Machine.



(g) Gradient Boosting.



(h) Logistic Regression.

Figure 7.5: Normalized confusion matrices for all classifiers. The vertical axis shows the true label, the horizontal axis shows the predicted label.

Our goal with this study was to show a possible application and usefulness of a flexible machine learning approach on features that could be easily acquired in clinical practice. The used features were presented in Table 7.2, and mostly comprised features collected from self-report. As such, collection of these features in clinical practice is relatively simple, and could give the clinician an early and accurate prediction of ones future risk on above-threshold levels of depression.

Currently our classifiers provide their users with a point estimate describing whether a person is expected to reach above clinical threshold levels of depression or not. Although such an estimate is useful, the lack of confidence intervals for these estimates might hinder its adoption in clinical practice. This gap between statistical inference and machine learning can be closed by applying a targeted learning approach (van der Laan & Rose, 2011). With targeted learning, one targets the initial estimators towards a specific question of interest by applying techniques such as Targeted Minimum Loss Estimation (TMLE). This procedure can improve the quality of our classifiers and, moreover, provide confidence intervals for the estimates, which in turn allow for significance and hypothesis testing (van der Laan, 2010; van der Laan & Rose, 2011). We explore this direction in Chapter 8.