Supervised machine learning methods in psychology:

A practical introduction with annotated R code

Author Note

Hannes Rosenbusch, Department of Social Psychology, Tilburg University.

Felix Soldner, Department of Security and Crime Science, University College London.

Anthony M. Evans, Department of Social Psychology, Tilburg University.

Marcel Zeelenberg, Department of Social Psychology, Tilburg University, & Department of Marketing, VU Amsterdam.

Correspondence concerning this article should be addressed to Hannes Rosenbusch, Department of Social Psychology, Tilburg University, 5000 LE Tilburg, The Netherlands. Email: h.rosenbusch@uvt.nl

Abstract

Machine learning methods for pattern detection and prediction are increasingly prevalent in psychological research. We provide a comprehensive overview of machine learning, its applications, and how to implement models for research. We review fundamental concepts of machine learning, such as prediction accuracy and out-of-sample evaluation, and summarize four standard prediction algorithms: linear regressions, ridge regressions, decision trees, and random forests (plus k-nearest neighbors, Naïve Bayes classifiers, and support vector machines in the supplementary material). This selection provides a set of powerful models that are implemented regularly in machine learning projects. We demonstrate each method with examples and annotated R code, and discuss best practices for determining sample sizes; comparing model performances; tuning prediction models; preregistering prediction models; and reporting results. Finally, we discuss the value of machine learning methods in maintaining psychology's status as a predictive science.

Keywords: machine learning, statistics, prediction, data science, research methods

Supervised machine learning methods in psychology:

A practical introduction with annotated R code

Psychologists are increasingly interested in accurately predicting real-world phenomena (e.g., Park et al., 2015; Plonsky, Erev, Hazan, & Tennenholtz, 2017; Joel, Eastwick, & Finkel, 2017, Wang & Kosinski, 2018). In the process of shifting from pure explanation to accurate prediction, psychologists have adopted powerful computational techniques from the field of machine learning (see Yarkoni & Westfall, 2017). The current work introduces machine learning as a collection of methods and tools that can boost the predictive accuracy of psychological research. The goals of this paper are to review fundamental concepts of machine learning, discuss its relationship with standard psychological methods, and give concrete guidelines to implement machine learning projects in R. The example analyses in the tutorial sections can be replicated with data and scripts provided in the supplementary materials. To conclude, we provide recommendations for best practices and warn of common dangers when implementing machine learning techniques.

Machine Learning and Prediction Accuracy

Machine learning is “a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data” (Murphy, 2012, p. 1). Psychologists already have the tools to detect patterns in data; most commonly least-squares regression techniques. When developing and testing theories, such patterns (represented as regression coefficients) are examined for statistical significance to ascertain the effect of the predictor on the outcome variable. The remaining element of machine learning, predicting future data, is becoming increasingly important to psychologists (Alharthi, Guthier, & El Saddik, 2018; Plonsky et al., 2017; Walsh, Ribeiro, & Franklin, 2017).

When researchers’ primary goal is making accurate predictions, they can again use statistical models to link predictor variables to outcome variables. This search for accurate models lies at the heart of machine learning. Now, the primary metrics of interest are no longer model coefficients, but the accuracy of the model’s predictions (i.e., how close predictions are to true values). Importantly, model accuracy must be evaluated using new data. Said differently, machine

learning models are fit on one data set (the 'training set'), and predictions are made and evaluated using a new data set (the 'test set'). This out-of-sample testing avoids overestimating a model's accuracy, as models are generally overfitted to the training sample.

Testing out-of-sample prediction accuracy is sufficient to turn traditional regression analyses into machine learning. Consider classic work finding that insufficient sleep is associated with suicidal intentions (Ribeiro et al., 2012). This work, based on inferential tests, does not tell us how accurately sleep predicts suicidal intentions. Is the model accurate enough to implement alert systems based on sleep quality? An overlapping group of researchers published a machine learning study, focused on prediction accuracy, in which they predicted the risk of suicide attempts using an array of psychological variables (Walsh, Ribeiro, & Franklin, 2017). They did not, however, include sleep quality as a predictor. The potential contribution of sleep quality in a model for accurately predicting suicide attempts, therefore, remains unquantified.

Psychology and machine learning come together whenever the research question is "How well does x predict y?" Most psychologists use linear regression to quantify achievable prediction accuracy (e.g., Rimfeld, Kovas, Dale, & Plomin, 2016; Hassan, Shiu, & Shaw, 2016). However, machine learning offers a wide range of alternative models which usually lead to accuracy improvements (e.g., Park et al., 2015; Plonsky et al., 2017; Joel, Eastwick, & Finkel, 2017, Wang & Kosinski, 2018). The cited projects can be labeled applied machine learning and differ from most psychological work in three ways:

1. A focus on prediction accuracy.
2. Measures of prediction accuracy for *new* samples.
3. Use of prediction models that, unlike typical linear regressions, are manually tuned to better fit the specific problem at hand (see the section 'Hyperparameter tuning').

In the following sections, we describe the concepts of prediction accuracy and out-of-sample evaluation. Subsequently, we introduce four of the most common machine learning models (plus three in the supplementary material) and provide sample R code to apply, tune, and evaluate such prediction models in psychological research.

Prediction Accuracy

The central premise of machine learning is to use statistical models to make (accurate) predictions. In the following section, we describe the most relevant metrics for assessing accuracy.

Continuous outcome variable. If the predicted outcome is continuous, there are multiple approaches to assess accuracy. One of the most common metrics is R^2, defined as the proportion of variance accounted for by the model. Higher R^2 values signify higher accuracy. When the residual variance (i.e., "sum of squared residuals" in the formula below) is zero, the model makes perfect predictions and R^2 = 1. If the summed residuals are equal to the total variance (in the denominator), the model is useless, predicting the mean is equally accurate, and R^2 = 0.

$$R^2 = 1 - \frac{\sum of\ Squared\ Residuals}{\sum of\ Squared\ Deviations ¿\ the\ Mean ¿}$$

Equivalently, researchers can measure model accuracy by correlating predicted and observed scores (Youyou, Kosinski, & Stillwell, 2015). Other metrics focus on the average residual size (instead of residual proportion as in R^2) with smaller residuals signifying higher prediction accuracy. Most common are the root mean square error (RMSE; residuals are squared before they are added up) and the mean absolute error (MAE; negative signs are removed before residuals are added up).

$$RMSE ¿ \sqrt{\frac{1}{n} ¿ (y_i - \hat{y}_i)^2}$$

$$MAE = \frac{1}{n} * \sum_{i=1}^{n} ¿\ y_i - \hat{y}_i \vee ¿$$

The MAE-metric weighs each prediction error equally, whereas the RMSE gives more weight to large errors (due to the squaring). This makes the MAE metric easy to interpret, while the RMSE is more relevant when large mispredictions are disproportionately costly. As a result, evaluating either metric requires familiarity with the scale of the outcome variable.

Categorical outcome variable. If the outcome is categorical, there are again multiple options for assessing model accuracy. Many measures can be extracted from the so-called

confusion matrix, which shows a cross-tabulation of predicted and observed values (see Table 1 for an example).

Table 1. *Example of a confusion matrix*

	Predicted value: positive	Predicted value: negative
True value: positive	10 (true positive)	5 (false negative)
True value: negative	8 (false positive)	20 (true negative)

Note. Numbers are counts.

The most prominent measure of accuracy is called the accuracy score (or just 'accuracy') and is defined as the number of correct predictions divided by the total number of predictions (i.e., percentage of correct predictions):

$$Accuracy\ score = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives}$$

However, sometimes researchers have reason to focus on either positive or negative cases. Thus, more refined quantifications of accuracy pertain to the number of false and true positives and negatives and are known as the sensitivity (also called 'recall') and specificity. Sensitivity signifies how many of the positive measurements (e.g., 'behavior occurred', or 'disorder is present') were predicted to be positive cases, whereas specificity describes how many of the negative measurements (e.g., 'behavior did not occur', or 'disorder was not present') were predicted to be negative cases (cf. signal detection theory; Macmillan, 2002). A model's 'precision' refers to the number of true positives divided by all positive predictions.

$$Sensitivity = Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives}$$

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Thus, if researchers want to ensure that their model is predicting "positive" cases (e.g., predicting cases of a psychological disorder), they might want to focus on the model's sensitivity. If false positives are to be avoided, as in forensic DNA tests, then the model's precision becomes

more important. When priorities are not one-sided, the harmonic mean between precision and recall can be used (i.e., the F1 score; Scherer, Stratou, Gratch, & Morency, 2013).

$$F1\,score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * True\ Positives}{2 * True\ Positives + False\ Positives + False\ Negatives}$$

Given the many alternatives, it is always preferable to report multiple accuracy metrics (e.g., by plotting false positive against true negative rates). Focusing on one individual metric can distort the reader's impression of accuracy and hinders comparisons between projects (e.g., Akosa, 2017).

Evaluations of prediction accuracy. Interpreting prediction accuracy metrics (e.g., as high, low, or medium accuracy) is difficult. Interpretation depends on the context and requires domain expertise. There are different ways of evaluating accuracy, and most involve comparisons with a reference point. For psychologists, this is often the reliability of the measurement of the outcome variable, which poses the maximum accuracy. When the outcome's test-retest reliability is $r = .7$, prediction cannot exceed this ceiling without predicting random error variance. This upper limit entails, for instance, that demographic info (which can be measured with perfect reliability) can often be predicted more accurately than noisy personality scores (Kosinski, Bachrach, Kohli, Stillwell, & Graepel, 2014). Models with accuracies that come close to the outcome's reliability are, therefore, very successful.

Baseline (or null) models are another common standard for comparison: Simple baseline accuracies for continuous variables might be provided by the model $\hat{y} = \text{mean}(y)$, and for categorical variables by $\hat{y} = \text{mode}(y)$. If a new model cannot predict y substantially better than such baseline models, the performance of the new model can be evaluated as poor. Imagine having a model predicting relapse rates among drug addicts with an accuracy of 80.3%. This accuracy is not impressive if only 20% of investigated people relapse. Always predicting the mode ('no relapse') would already lead to an accuracy of 80% and the new model barely improves on that. Still, if there are no better alternatives (e.g., human judgment), an increase of 0.3% accuracy can amount to saving many lives. That is to say, context matters for evaluating accuracies. For example, Sumner, Byers, Boochever, and Park (2012) found that language-based models can marginally predict

negative personality from Twitter, but are incapable of predicting whether someone lies in the "interesting" tails of the distribution.

Lastly, it is common to consider prior prediction attempts. Such reference points are commonly discussed in the machine learning literature, where the history of classic prediction challenges receives much interest (e.g., identifying words from speech, handwritten symbols from pictures, or tumors on scans). If a model can improve historical accuracies, potentially including predictions made by human raters (Youyou, Kosinski, & Stillwell, 2015), it is argued to be relatively accurate. While psychological research has yet to develop a set of classic prediction problems, there are many challenges that would benefit from an ongoing competition and bookkeeping. Examples include the behavior of people in economic games (e.g., Plonsky et al., 2017), health trajectories (e.g., Chekroud et al., 2016), or dispositional traits (e.g., Bachrach, Graepel, Kohli, Kosinski, & Stillwell, 2014).

Out-of-sample evaluation

Another essential feature of machine learning is that accuracies are not tested using the same sample that was used to fit the prediction model. Accuracy metrics derived from the model-fitting sample are biased due to 'overfitting,' meaning that parameters in the prediction model not only reflect the true relationships between predictors and outcome but also unique sample characteristics. In other words, prediction models are optimized to predict an outcome from predictor variables *in the present sample*. This problem is intuitive when compared with the problem of complex experimental designs (e.g., 3×3×4 experiments) implemented with small samples of participants. Statistical patterns in such a design are unlikely to generalize; similarly, prediction accuracy extracted from the model-fitting sample (i.e., the training accuracy) overestimates the accuracy achieved on new samples (i.e., the testing accuracy), especially for complex models.

Testing accuracy is informative because it quantifies the accuracy that can be expected when applying the model to new data. The ideal way to test accuracy is to fit the model on one sample (training set) and evaluate the model on a different sample (test set). We discuss two of the most common approaches to split data into training and test sets in machine learning:

Train-test-split (Hold-out method). The simplest way to generate separate training and testing samples is to collect one big sample and then to randomly split it. An alternative (and arguably even better) method is to collect one sample for training, fit the model on this sample, preregister the model, and then collect the new sample for the test set (cf., Brandt, 2017). We provide an example for preregistering a machine learning model in the supplementary materials, which can be used as a template.

When splitting the original sample, how much data should be used for model fitting and how much should be used to quantify the accuracy? Commonly, the training set encompasses 60%-80% of the data, while the test set has 40-20% (e.g., Ng, 2018). Considering the purpose of the test set is helpful when making cut-off decisions. That is, how many predictions does the researcher want to see before judging the model? A small test set can already inform the researcher whether the model is close to perfect or close to useless. If more precise evaluations are needed, then the test set needs to be larger. Another helpful heuristic is that the test set should be representative of the target population, which limits how small it can be. For more detailed considerations see the section "Sample sizes in machine learning" below.

K-fold cross-validation. Randomly splitting data into a training and test set bears risks, especially if the original sample is relatively small. More precisely, chance affects the transferability of the prediction model (and the computed test accuracy). A process called k-fold cross-validation offers a partial solution: this process involves *repeatedly* splitting the data, fitting the model, and testing it on new data. This process involves every observation as part of the training data and the test data, respectively, at some point in the process.

In k-fold cross-validation, the data is split into k (often $k = 10$; Kuhn & Johnson, 2013) equally-sized subsets called folds. Subsequently, k rounds of model fitting, test set prediction, and accuracy evaluation are conducted (see Figure 1). In the first round, the model is fit on the first 9 subsets, the model is used to make predictions for the 10^{th} subset. In the second round, the model is fit on subset 1 through 8 plus subset 10 and evaluated on subset 9, etc. The 10 individual accuracy quantifications obtained from this procedure are averaged to give an estimation of the model's

overall accuracy. This procedure mitigates the danger of chance affecting test accuracy; permits researchers to use relatively more of their data for training the model; and highlights variability in the model's accuracy.

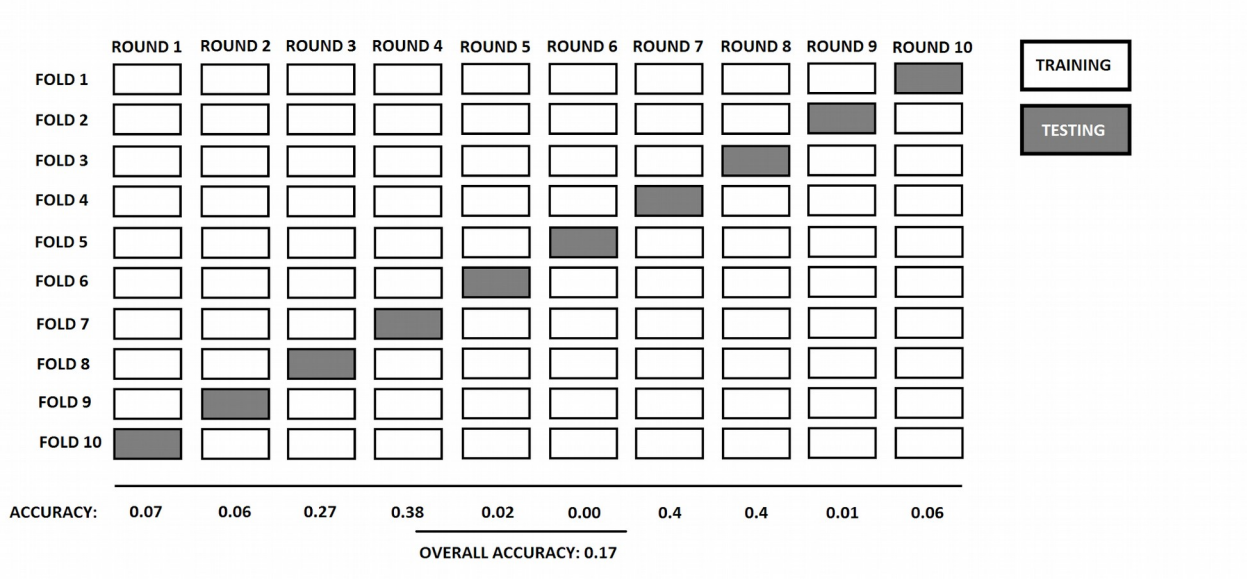


Figure 1. The process of cross-validation, including random splitting of the dataset into *k* (here 10) folds, fitting of the model on *k*-1 folds and computing the accuracy achieved on the held-out fold. After *k* rounds of training and testing, the average accuracy can be computed alongside the variability of the achieved accuracy.

Sample sizes in machine learning. In machine learning, samples sizes range from hundreds to millions. Complicated models with many coefficients might even benefit from more observations (He, Zhang, Ren, & Sun, 2016), while other models can be fit using fewer than a thousand cases (Golbeck, Robles, & Turner, 2011). An accessible description of factors affecting required sample sizes in machine learning is provided by Raudys and Jain (1991). However, there are no exact guidelines for researchers. Here we present four strategies to help determine appropriate sample sizes:

1. Familiarize yourself with research projects in the past that are similar to yours and make an overview containing each project's a) predictor variables, b) outcome variable, c) prediction model(s), d) sample size, e) achieved prediction accuracy. Projects with similar predictors, outcomes, and models usually indicate how accurate models will be

given specific sample sizes. We provide a short example in Table 2.

Table 2. *Example papers for predicting extraversion from Facebook material*

Paper	Predictor	Outcome	Model	Accuracy	Sample size
Golbeck, Robles, & Turner, 2011	Facebook likes, language, demographic, friends	Extraversion	Gaussian processes	*r* = .05	167
Kosinski et al., 2014	Facebook friends, groups, likes, network density, photo tags, statuses	Extraversion	Linear regression	*r* = .31	16,900
Schwartz et al, 2013	Facebook language	Extraversion	Ridge regression	*r* = .38	75,000

2. Look for similar data published online. In machine learning, it is quite common to publish data sets in order to stimulate prediction competition and knowledge exchange. Acquiring a data set allows you to plot achieved accuracies of specific models against the sample size used for training the model (i.e., learning curves). The saturation point of such curves can give you a good reference for how much data you might need (Figueroa, Zeng-Treitler, Kandula, & Ngo, 2012).
3. Simulate a realistic data set based on prior knowledge on variable distributions. Multiple packages in R are available to generate data sets with prespecified covariance structures (e.g., Goldfeld, 2018). Examining how the accuracy of simulated prediction models changes with different sample sizes (see Figure 2), can help in estimating realistic accuracy levels that can be reached with different sample sizes. Note that this

approach requires a-priori assumptions about the data and is less feasible when there are many predictors.

4. Collect initial data to assess model requirements. Researchers can diagnose a lack of training data by investigating how the achieved testing accuracy compares to the training accuracy. If a model's training accuracy is much higher than its testing accuracy, the model is too complex for the size of the training set and therefore it is severely overfitted to the training data (i.e., a high variance problem). In such instances, it is reasonable to collect more data (or reduce model complexity). If both training and testing accuracy are poor, researchers can diagnose that their model is underfitted (i.e., a high bias problem) and needs a higher degree of complexity or better predictors.

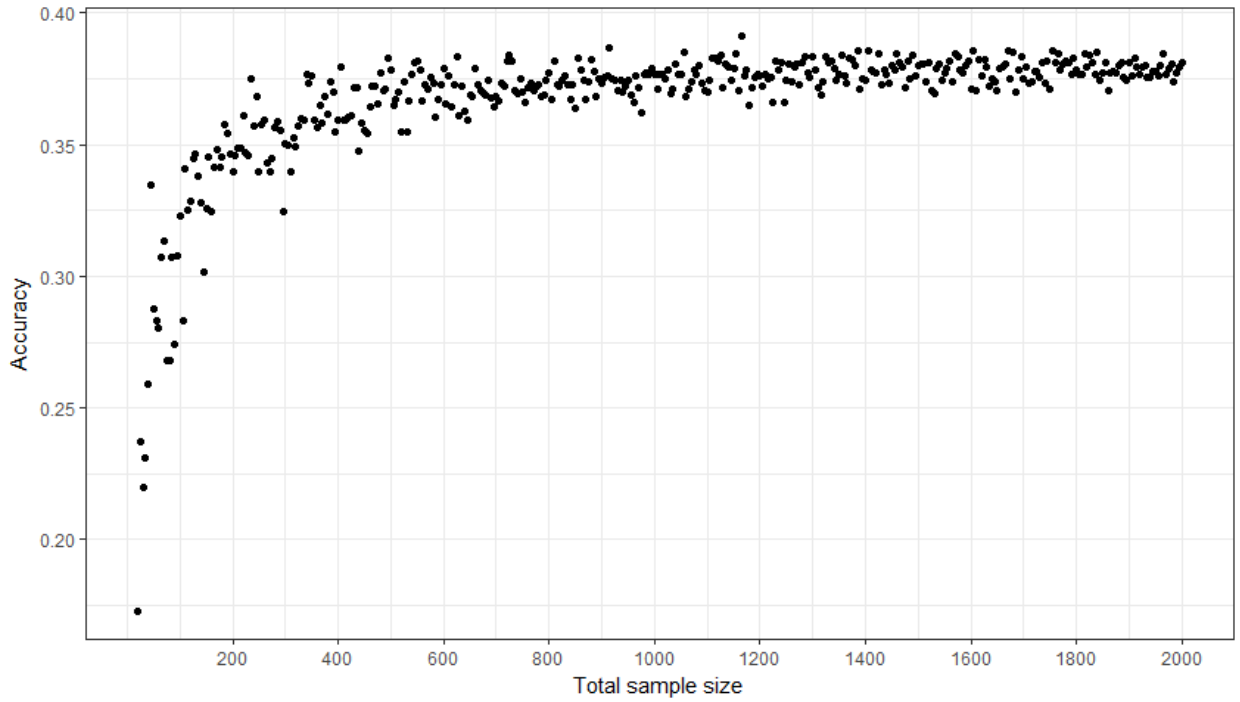


Figure 2. The learning curve of a linear regression model based on simulated data with 6 predictor variables (inter-correlations between .15 and .7; see supplementary materials for code). Accuracy is computed as the correlation between true and predicted values. The learning curve suggests that around 600 observations are needed for approaching the maximal prediction accuracy with this model.

In sum, machine learning projects split data into separate training and testing sets. As the

shape of the best prediction model is determined based on the data, it is more difficult and less common to preregister required sample sizes. However, machine learning requires extra data for out-of-sample evaluation and prediction models often include many predictors/coefficients; as a result, required sample sizes are typically large. The lack of sufficient data can be diagnosed if the testing accuracy of a model is substantially worse than its in-sample training accuracy. Such instances of overfitting can be dealt with by collecting more data or reducing model complexity (e.g., changing from non-linear to linear models).

The following section introduces popular machine learning models, including how they can be implemented in R code, and "tuned" to improve accuracy.

Implementations of Common Models in Machine Learning

In our examples, we use models to predict regional happiness in the USA (BRFSS, 2005-2010; see supplementary materials). The example code will predict regional (i.e., county-level) happiness of US citizens as closely as possible, but the code examples are transferable to other prediction problems. We utilize this dataset because it is known among psychologists, publicly available, and has geographic markers that enable merging in a wide range of additional predictor variables. Throughout the code sections, we utilize the R package *caret* (Kuhn, 2015), which includes a large and standardized selection of prediction models.

The basics: regression, prediction accuracy, and out-of-sample validation

Standard linear and logistic regressions are often not expected to constitute machine learning. However, these traditional techniques can challenge the accuracy of more complex models (Lučić, & Trinajstić, 1999). We demonstrate a typical implementation of a linear regression model in machine learning. Our outcome of interest is regional happiness and our predictor variable is the relative amount of people drinking alcohol (measured at the regional level) (e.g., Bellos et al., 2013). In other words, can regional alcohol consumption be used to predict regional differences in happiness?

```
> #We work with a data frame containing 2546 rows (US counties) and two columns.
> dim(data)
        [1] 2546    2

> colnames(data)
        [1] "alcohol_use" "happiness"

> #We randomly select 75% of the row numbers of the data.
> indices <- createDataPartition(data$happiness, p = 0.75, list = FALSE)

> #We include the selected rows in our training set (i.e. the data that we fit the model on).
> trainingset <- data[indices,]

> #We include the remaining rows in our test set (i.e. the data used to evaluate the model
accuracy).
> testset <- data[-indices,]

> #We fit the model with caret's train function.
> #This function can be used to fit a large selection of machine learning models.
> #Here we chose the model "lm", which specifies a linear regression model.
> predictionmodel <- train(happiness ~ alcohol_use, data = trainingset, method = 'lm')

> #We evaluate the prediction accuracy on the test set with caret's R2 function.
> predictions <- predict(predictionmodel, testset)
> R2(predictions, testset$happiness)
        [1] 0.114
```

The linear regression model based on county-level alcohol consumption predicted 11.4% of the variance in regional happiness[1]. However, it is worth checking if the randomness of our split into training and test set affected the observed accuracy. To investigate such chance effects, we systematically repeat the splitting, fitting, and evaluating process through 10-fold cross-validation.

```
> #We prespecify that the following cross-validation should have k=10 splits.
> cross_validation <- caret::trainControl(method="cv", number=10)

> #We again train the model.
> #We do not need to explicitly declare training and test set.
> #caret automatically implements the cross-validation.
> predictionmodel <- train(happiness ~ alcohol_use, data = data, method = 'lm', trControl =
cross_validation)
```

[1] The accuracy is substantially higher when excluding countries where predictor and outcome only have a handful of measurements.

```
> #We obtain a list of achieved accuracies obtained for each split.
> predictionmodel$resample
   RMSE  Rsquared  MAE  Resample
1  0.070 0.160     0.052 Fold01
2  0.068 0.169     0.050 Fold02
3  0.082 0.103     0.055 Fold03
4  0.073 0.057     0.053 Fold04
5  0.071 0.071     0.051 Fold05
6  0.066 0.177     0.048 Fold06
7  0.085 0.072     0.052 Fold07
8  0.078 0.036     0.053 Fold08
9  0.076 0.102     0.053 Fold09
10 0.070 0.137     0.052 Fold10

> #We obtain the overall (average) result.
> predictionmodel$results
  RMSE  R2    MAE   RMSESD  R2SD  MAESD
  0.074 0.108 0.052 0.006   0.050 0.002
```

The result of the cross-validation procedure is in line with our initial training and test split. However, there is some variability in accuracy estimates. In one of the 10 subsets, the model only explains 4% of the variance (Fold 08) whereas in another one it explains 18% (Fold 06) of the variance. Being able to observe such variability is one reason why it is preferable to conduct cross-validation instead of a train-test split, as the latter should only be applied with large sample sizes (more than 1000 participants in the test data). The disadvantage of cross-validation lies in an increased need for computational resources, which is however unlikely to be an issue with sample sizes under 100,000 observations.

Ridge regression

In standard linear regression models, the individual beta coefficients are optimized to reduce the residual sum of squares of the outcome variable. This optimization should improve accuracy. However, it also guarantees that the coefficients are perfectly fitted for the present sample, which might diminish generalizability to other samples. Coefficients for each predictor variable are specified exactly to the last decimal and even predictors that are not truly related to the outcome variable virtually always have a non-zero coefficient because they *happen* to explain a small part of residual

variance (Lever, Krzywinski, & Altman, 2016). The result is an overfitted model which will not perform well with new samples. This problem becomes exacerbated if the number of predictor variables is high in relation to the number of observations, and if predictor variables are correlated (Askin, 1982). In such cases, the high-dimensional prediction model can explain a large proportion of variance in the current sample. However, the resulting model may be too complex for the sample size, and as a result the model achieves poor accuracies on new samples (Dana & Dawes, 2004). The need for much more data for little increases in complexity is referred to as the 'curse of dimensionality.'

A potential solution is to use an alternative procedure that simultaneously minimizes both the residual variance *and* the complexity of the model. Ridge regression models accomplish these goals by biasing individual regression coefficients towards zero. Hence, ridge regression models give more weight to the intercept, making predictions less variable and decreasing the magnitude of prediction errors which emerge from poor model transferability. This simplified model has less noise and can be more generalizable.

Statistically, ridge regression suppresses beta coefficients by changing the optimization criterion of the standard regression model. In addition to minimizing residual variance, the model attempts to minimize the sum of squared beta coefficients. The result is a tradeoff between residual reduction and complexity reduction, which is expressed in the cost function to be minimized:

$$Cost\ Linear\ Regression = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$Cost\ Ridge\ Regression = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda * \sum b^2$$

The ridge regression cost function has a parameter λ (lambda) that is absent in standard linear regression. Lambda is an example of a *hyper*parameter (see next section), that needs to be manually tuned using data. Lambda quantifies how strongly regression weights should be suppressed, thereby dictating the tradeoff between error reduction and model simplicity. Higher values of lambda lead to a stronger penalization of regression weights and therefore a more

restricted/conservative model. It is standard practice to test a range of values for lambda (either manually or automatically) and choose the value that gives the best accuracy for new cases, but not yet the final test set (Claesen & De Moor, 2015). Next, we describe the general practice of hyperparameter tuning. Subsequently, we demonstrate how the specific hyperparameter lambda can be tuned for ridge regression models.

Hyperparameter tuning. For linear or logistic regression, researchers can fit the model without having to manually specify further parameters. That is to say, if the data are the same, two researchers fitting a regression model should usually obtain the same result. Machine learning techniques (like ridge regression) are often tuned with hyperparameters, which are not automatically tuned by the data. Hyperparameters determine stable model characteristics *before* the model is fit to a dataset. There are some parallels in traditional psychological methods. For example, in factor analyses, researchers can specify model structure *a priori* (e.g., four factors should be extracted).

Hyperparameter settings are used to increase prediction accuracy. Often, hyperparameters determine model complexity (vs. parsimony), or the procedure with which a model incorporates new data and adjusts its coefficients. Most prediction models come with a small set of hyperparameters that are straightforward to tune. Tuning involves testing a range of possible values and selecting the value with which the model achieves the highest out-of-sample accuracy. If multiple hyperparameters are tuned simultaneously, researchers typically test ranges for each hyperparameter, and try out different combinations (i.e., the ‘value range search’ becomes a ‘values grid search’).

Importantly, the prediction model with the best hyperparameters is ultimately evaluated again on new data. Most commonly, all models are fit on sample A, and the final model is selected based on its accuracy achieved on sample B. The accuracies achieved on sample B might still overestimate a model’s true accuracy, because we select the best model, out of a potentially wide range of models. Said differently, we might “manually” overfit to sample B during hyperparameter selection. Thus, after fitting the models on sample A (the training set), and selecting the best performing model on sample B (the development set), researchers apply the final model to a new

sample C (the test set). Notice, that we only have to introduce the development set (sample B), if we tune model hyperparameters. In the case of linear regression above, for instance, we only had one model, allowing us to directly quantify prediction accuracy on the test set (see Figure 3).

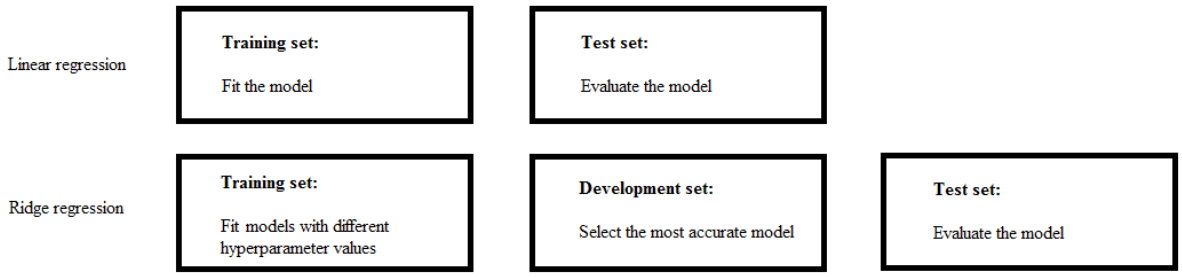


Figure 3. The difference in training and test split between OLS regression and models with tunable hyperparameters.

Next, we demonstrate how to efficiently split our data into training, development, and test samples.

Example ridge regression. Ridge regression is most useful if researchers have a large number of (intercorrelated) predictors relative to their sample size. An example case is to predict psychological phenomena based on language. A regression model that predicts an outcome based on language often has as many predictor variables as there were unique words in the dataset. Further, many words are highly correlated in their usage leading to collinearity and unstable model coefficients. These conditions make ridge regression a sensible choice (Schwartz et al, 2013; Zhang & Oles, 2001).

We demonstrate how to implement ridge regression by using regional Twitter language to predict regional happiness (Wang, Kosinski, Stillwell, & Rust, 2014). Each predictor variable pertains to the relative frequency with which a specific word is used by Twitter users in the target region. In other words, we ask if regional differences in word use on Twitter can be used to predict regional

differences in happiness.

Implementing the ridge regression model is similar to linear regression. However, there are three important differences. First, we implement cross-validation to split the data into training and development set, while withholding a final test set. Second, we tune the hyperparameter lambda during cross-validation[2]. Third, when training the model, we additionally specify that predictors should be standardized (i.e., centered and scaled) before the model is fit. Given that we penalize high beta coefficients, all predictors need to be on the same scale.

```
> #Split off the test set from the training and development data.
> indices <- caret::createDataPartition(data$happiness, p = 0.75, list=FALSE)
> train_dev_data <- data[indices,]
> testdata <- data[-indices,]

> #We implement 10 fold cross-validation (same as above).
> cross_validation <- trainControl(method="cv", number=10)

> #Set possible values for hyperparameter lambda.
> tuning <- expand.grid(alpha = 0, lambda = c(0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4))

> #We train the model (including preprocessing).
> #The cross-validation is repeated for each value that the hyperparameter can take on.
> predictionmodel <- train(happiness ~ ., data = train_dev_data, method = 'glmnet', preProcess =
c("center", "scale"), tuneGrid = tuning , trControl = cross_validation)
        Aggregating results
        Selecting tuning parameters
        Fitting alpha = 0, lambda = 1.6 on full training set

> #Get the results.
> predictionmodel$results
     lambda  RMSE   R2        MAE       RMSESD  R2SD      MAESD
1    0.1     0.067  0.104     0.049     0.005   0.052     0.003
2    0.2     0.066  0.106     0.048     0.005   0.050     0.002
3    0.4     0.065  0.111     0.048     0.004   0.047     0.002
4    0.8     0.065  0.114     0.047     0.004   0.044     0.002
```

[2] A further variable appearing in the code is called ‘alpha’, which is not typically part of ridge regression. Here, we set alpha to zero, which simply tells the more general ‘glmnet’ model that we want to compute a ridge regression model. An alternative approach would be to directly set the method argument to ‘ridge’ instead of ‘glmnet’, which would allow us to leave out the alpha specification. However, this method appears to take more computational resources based on our test runs, which might reflect differences in the methods’ back-end implementation.

```
      5   1.6     0.065   0.115     0.047     0.004     0.042     0.002
      6   3.2     0.065   0.111     0.047     0.004     0.041     0.002
      7   6.4     0.065   0.101     0.048     0.004     0.039     0.002

> #Make predictions with best model and evaluate accuracy
> test_predictions <- predict(predictionmodel, testdata)
> R2(test_predictions, testdata$happiness)
      [1] 0.122
```

The most accurate model has a lambda value of 1.6. Notice that caret automatically selects this value and uses it to fit the final model on all the non-test data (the training and development sets). Subsequently, we get an R^2 value for the test data which is slightly higher than in the training/tuning phase. The reason is likely that the final model was fit on all training and development observations, whereas before, we used a cross-validation approach and therefore 10% less data for model fitting. A traditional OLS regression without penalization achieves an accuracy of R^2 = .11 using the same predictors, which is just marginally worse. However, some implementations of linear regression in R throw warnings and give worse results when the number of predictors is much higher than the number of observations. Example implementations of ridge regression in psychological research are provided by a range of authors (Eichstaedt et al., 2015; Ghandeharioun et al., 2017; Dana & Dawes, 2004).

There are not set rules for which and how many values should be tried out for lambda. Generally, it makes sense to try out a wide range of values as long as the best solution is later evaluated on new data. A popular and statistically efficient method to choose values for hyperparameters like lambda is to pre-specify a range of values and let the computer determine a random selection of tested hyperparameter values within that range (Bergstra & Bengio, 2012). Alternatively, the caret package can choose a default range and values within that range for the researcher (we demonstrate this approach below).

Decision trees and random forests

Decision trees are another popular predictive algorithm, and they are especially popular in cases where researchers are interested in easily readable and comprehensible prediction rules (as

in the fictitious example in Figure 4).

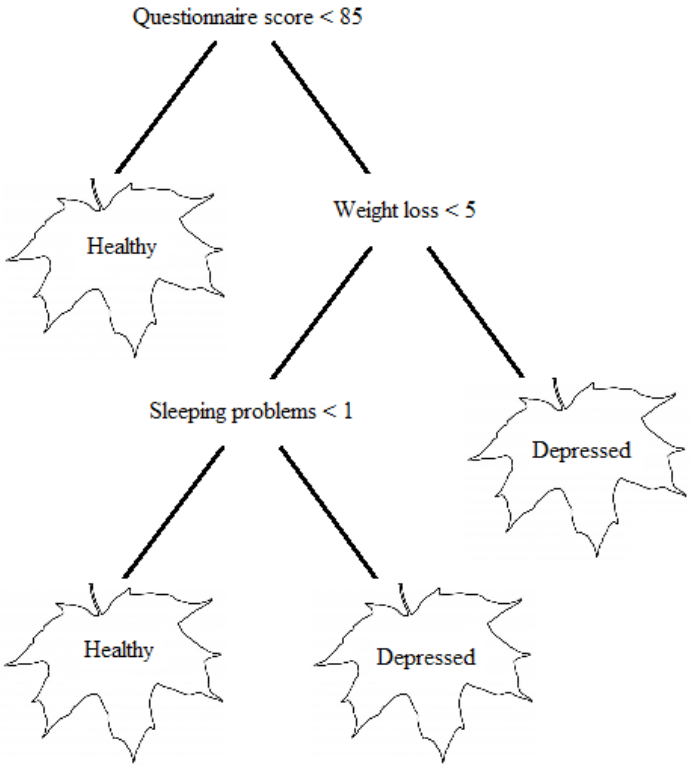


Figure 4. Fictitious example of a decision tree model to diagnose depression. Each observation has scores on the predictor variables questionnaire score, weight loss, and sleeping problems. The final 'leaves' at the bottom of the tree give the model's prediction for new cases.

Decision trees split observations into increasingly homogeneous subgroups based on binary questions (e.g., "Does this observation score lower than 85 on the questionnaire?"). Accordingly, the decision tree consists of a sequence of questions used to subcategorize the data set. Individual observations are funneled through the tree, and at each crossway of branches, the observation is either guided to the left or right depending on whether the answer to the specific binary question is yes or no. Predictions for a categorical outcome variable are made by assigning the value that was most common on this specific leaf in the training data ("majority vote"). For continuous variables, the predicted value is usually the mean of all training observations that landed on this leaf.

The sequence of questions, as well as the optimal cut-off values within each branching fork, are determined when fitting the tree model to the training data. To increase prediction accuracy, it is

favorable to have a tree model which leads the same outcome values to land on the same leaves and dissimilar values on different leaves. Said differently, if a leaf only includes observations that have the same outcome value (homogenous), we can expect new cases landing on this leaf to also have that (or a very similar) value. Conversely, if a leaf includes observations that have a range of very different values (heterogeneous), we can be less certain about predictions for new cases landing on this leaf. Thus, the statistical criterion which is minimized when fitting a tree model is the heterogeneity of measurements in each of the final leaf nodes. This measure of heterogeneity is often an entropy score. For the hypothetical example tree in Figure 4 the formula for a single leaf's entropy is:[3]

$$Entropy_i = \frac{-depressed_i}{all\,cases_i} * \log\left(\frac{depressed_i}{all\,cases_i}\right) - \frac{healthy_i}{all\,cases_i} * \log\left(\frac{healthy_i}{all\,cases_i}\right)$$

The entropy of the overall model is a weighted sum of the leaves' entropies. The sequence (i.e., order) of questions in the tree is determined by the information gain, which quantifies the decrease in entropy after an additional data split. At each new node, the question which provides the highest information gain is selected and appended to the tree.

Researchers implementing the decision tree algorithm often tune one hyperparameter to determine the maximal size of the tree. Very large trees with many questions and splits are more successful in minimizing entropy and prediction residuals in the training data. However, they run an increased risk of overfitting. Imagine, for instance, a tree which keeps adding data splits until each leaf only consists of a single training case. Such a tree would have perfect accuracy on the training set, but it would be very specific for the sample, therefore entailing poor transferability to new samples. Manually setting the maximal size of the decision tree allows researchers to control this trade-off. For instance, it is possible to select a maximum number of leaves as the according hyperparameter. Here, we give an example of tuning the slightly more sophisticated hyperparameter *cp* (the complexity parameter), which determines how high the minimal increase in R^2 has to be for a new branch to be drawn. If there are no predictor variables that can attain this value, the branch will

[3] Subscript *i* refers to the *i*-th leaf. All variables in the formula are counts (e.g., $depressed_i$ = number of depressed training cases on leaf *i*).

not be drawn and the tree ends with a leaf at this point. The final value for the hyperparameter will again be selected by trying out a range of values and choosing the value that gives the highest accuracy on the development data.

Example decision tree. In the code example, we predict regional happiness based on common census variables, including regional levels of education, gender ratio, median age, unemployment, ratios of white and black people, proportion of democratic and republican voters, and population size. To show an alternative method of tuning hyperparameters, we do not explicitly set the values for *cp* in the code example, but let caret pick values at random. The number of values to be generated can be set through the argument "tuneLength". We repeat the steps of splitting off test data, fitting models on training data, and selecting the best model on the development data 20 times, so we can estimate the variability in our accuracy evaluation. Alternatively, researchers can implement a nested cross-validation (see Kuhn & Johnson, n.d.).

```
> #create an empty vector to store the accuracies
> achieved_accuracies <- c()

> #repeat splitting, fitting, and evaluating 20 times
> for(i in 1:20){

        #data splitting
        indices <- caret::createDataPartition(data$happiness, p = 0.75, list=FALSE)
        train_dev_data <- data[indices,]
        testdata <- data[-indices,]

        #set up cross-validation
        cross_validation <- trainControl(method="cv", number=10)

        #train models with 8 different hyperparameters
        predictionmodel <- train(happiness ~ ., data = train_dev_data, method = 'rpart', tuneLength
        = 8,  trControl = cross_validation)

        #use best model to make predictions on test set
        test_predictions <- predict(predictionmodel, testdata)
        test_result <- R2(test_predictions, testdata$happiness)

        #append the achieved accuracy (20 in total)
        achieved_accuracies <- c(achieved_accuracies, test_result)
}
```

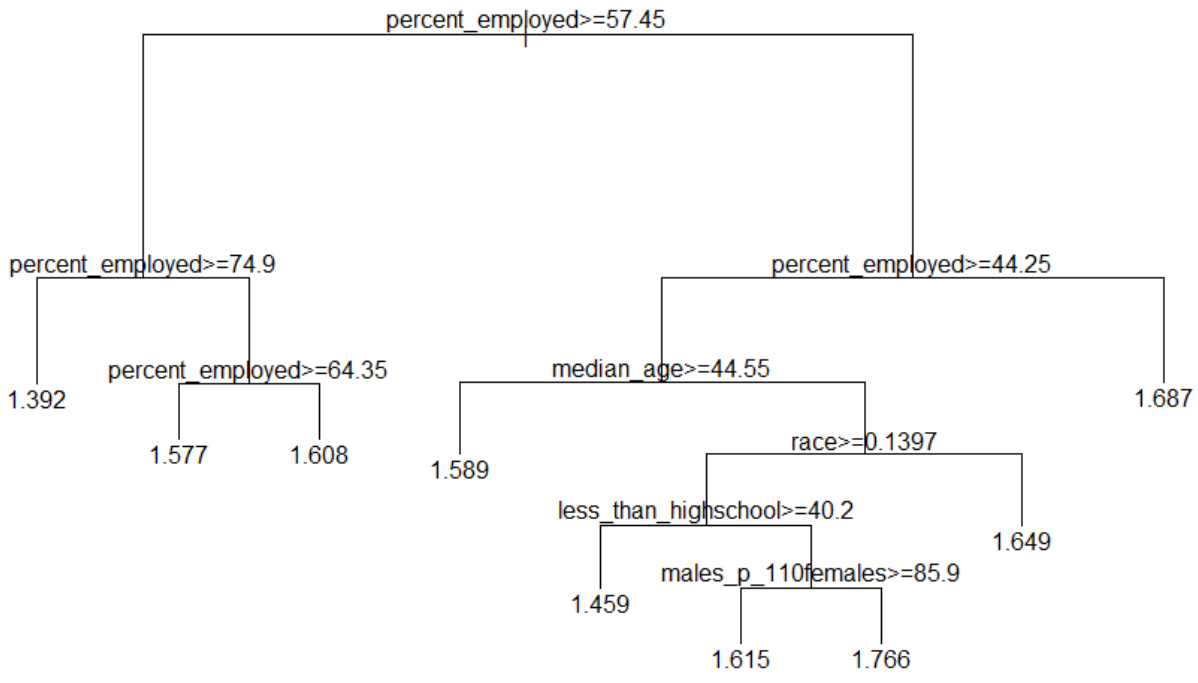
```
There were 20 warnings (use warnings() to see them)

> #inspect results across 20 iterations
> summary(achieved_accuracies)
      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
     0.023   0.088      0.117       0.109  0.126        0.202
```

There are warnings because caret sometimes sets the *cp* parameter so high that no binary split of the data can achieve the desired increase in R^2 (resulting in an empty decision tree). As shown, the best models achieved an average accuracy of R^2 = .11. Decision tree models can also be visualized in tree form in the caret package.

```
> plot(predictionmodel$finalModel)

> text(predictionmodel$finalModel)
```



Note. The split into counties with high and low employment rates is done first (at the top of the tree) as it is the most informative (helps the most to distinguish relatively happy and relatively sad counties). After this initial split, the employment variable is still the most informative as it is used again for splitting etc.

As depicted in the graph, most predictor variables are excluded as they do not enable improvements higher than the *cp* hyperparameter dictates.

Random forest models. While the decision tree algorithm is well-known, its prediction

accuracy can often be improved through models that build on its basic structure. The most prominent extension are random forest models, which consist of many different decision trees (therefore 'model ensembles'). Predictions from the individual trees within the random forest are averaged (continuous outcome) or the most frequent prediction is selected (categorical outcome). Why are the individual trees in the forest different from each other? For building each decision tree, a random (bootstrapped) sample of available observations is selected, and a random subset of available predictor variables is considered for each split. Having many different decision trees based on slightly different sets of observations and predictor variables minimizes the biases of individual trees and reduces overfitting. Thus, it is often not necessary to limit the size of the individual trees in the random forest (Breiman, 2001).

There are two primary hyperparameters to set when fitting a random forest model. First, the number of decision trees has to be predetermined. As higher numbers of trees do not lead to overfitting and allow for better predictions, it is common to set this hyperparameter to a high number like 500 (Oshiro, Perez, & Baranauskas, 2012). Increasing the number of trees does not have disadvantages, apart from increased computational costs. Second, the number of random predictor variables that get considered at each split must be set (hyperparameter 'mtry' below). Including more variables at any stage can lead to either overfitting or better predictions, which means that this value must be tuned carefully. It's common to select the square root of the number of available predictors and systematically investigate how higher and lower values affect the performance.

Example random forest. In the code example, we predict regional happiness using the predictor variables introduced in the decision tree example. Before, we used the outer loop only to quantify prediction accuracy (and the inner loop to select the best hyperparameter). Now, we also keep track of the best hyperparameter chosen in each of the 20 iterations. Additionally, we compare the accuracies of the random forest model and the decision tree model through a paired samples *t*-test (Huang, Lu, & Ling, 2003). Inferential tests to compare model performance are common in machine learning (Salzberg, 1997), but of smaller importance than assessing the practical significance of accuracy differences.

```
> #create empty vectors to store the accuracies and best hyperparameters across iterations
> achieved_accuracies2 = c()
> best_mtry = c()

> #20 iterations
> for(i in 1:20){

        #split data
        indices <- caret::createDataPartition(data$happiness, p = 0.75, list=FALSE)
        train_dev_data <- data[indices,]
        testdata <- data[-indices,]

        #set up cross-validation
        cross_validation = trainControl(method="cv", number=10)

        #set up hyperparameter
        tuning = expand.grid(mtry = c(1,2,3,5,7,9))

        #train models
        predictionmodel = train(happiness ~ ., data = train_dev_data, method = 'rf', tuneGrid =
        tuning, trControl = cross_validation)

        #make and evaluate predictions with best model
        test_predictions = predict(predictionmodel, testdata)
        test_result = R2(test_predictions, testdata$happiness)

        #append achieved accuracy
        achieved_accuracies2 = c(achieved_accuracies2, test_result)

        #append best hyperparameter
        best_mtry = c(best_mtry, predictionmodel$bestTune$mtry)
}

> #inspect accuracies and hyperparameters across iterations
> summary(achieved_accuracies2)
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
         0.112    0.157     0.177     0.179     0.194     0.264
> table(best_mtry)
        best_mtry
         2  3  5  7
         2 12  5  1

> #compare accuracies with decision tree results
> t.test(achieved_accuracies, achieved_accuracies2, paired = TRUE, alternative = "two.sided")
            Paired t-test
```

```
data:  achieved_accuracies and achieved_accuracies2
t = -6.1189, df = 19, p-value = 6.982e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.094 -0.046
sample estimates:
mean of the differences
        -0.070
```

The most accurate model (based on the inner cross-validation) most often considers three randomly selected predictor variables at each split. This model improves the accuracy achieved previously with the decision tree model by a substantial margin. On the 20 iterations, the random forest provides an average increase of 7% in explained variance, which is an increase of 57% relative to the decision tree models. Inferential tests comparing the sets of achieved accuracies confirm that the superiority of the random forest models is not only practically, but also statistically, significant ($t(19) = 6.119$, $p < .001$).

Yet, the final random forest model is not easily interpretable, as it consists of 500 distinct trees. A traditional OLS regression model based on the same predictors achieved an average accuracy of $R^2 = .11$ on the test data. Example implementations of decision trees and forests are provided by a range of authors in various subdisciplines of psychology (Piper, Loh, Smith, Japuntich, & Baker, 2011; Joel, Eastwick, & Finkel, 2017; Plonsky et al., 2017).

In the last sections, we discussed models based on either linear combinations or dichotomizations of predictor variables. In the supplementary material, we provide guidance for alternative modeling options based on point distances, Bayes' rule, and predictor space transformations.

Discussion

Traditional psychological research aims to establish causal effects of the predictor variables on the outcome variables, whereas machine learning projects aim to achieve the maximal (unbiased) accuracy when predicting the outcome variable. Still, intentions of researchers in both disciplines often converge. Psychologists are also interested in the question how well A predicts B,

and therefore the number of papers focusing on prediction accuracies is growing.

Our code examples demonstrate that traditional regression models are frequently less accurate than alternative models and that such alternatives, therefore, should be considered when making predictions. By applying the methods and tools of machine learning, psychology can maintain its aspiration to serve society as a predictive science (Yarkoni & Westfall, 2017).

Limitations of machine learning

Critics of machine learning models have argued that they are black-boxy, a-theoretical, or too complex to allow human interpretation (Krause, Perer, & Ng, 2016). While this is true for some of the most sophisticated models, it applies less often to the most commonly used models (Hindman, 2015). In fact, machine learning includes a range of clustering methods ('unsupervised learning'), which allow for the detection of theoretically meaningful patterns in psychological data (Jordan & Mitchell, 2015). Experimental psychologists can use machine learning techniques to explore differences between experimental conditions (Koul, Becchio, & Cavallo, 2018), and personality researchers can use them to assess the validity of psychological constructs (Bleidorn & Hopwood, 2018). Thus, while we concentrate on prediction accuracy in machine learning, it also has much to offer for theory-driven research.

Similarly, it is sometimes argued that machine learning methods need too much data and their application is unrealistic for many areas of psychology. Again, the required sample size depends on the context and the model that researchers utilize. When using image data, for instance, models tend to be very complex, and one might need a million images for a categorization task. Conversely, when only two or three predictor variables are available, say in an anonymous online survey, a linear regression model will usually require less than 1,000 observations to reach maximum performance.

Of course, the field of machine learning also has methodological struggles: Some issues, such as insufficient data sharing, reporting, and replication are akin to challenges in psychological research (Hutson, 2018). Others are more characteristic of machine learning, such as ethical problems with deploying socially biased models (Veale & Binns, 2017) or unequal distribution of

computational resources (Amolo, 2018).

Best practices for machine learning projects

With a new set of methods and tools, there come new mistakes that can be made. Therefore, we discuss five issues that authors and reviewers should pay attention to in psychological machine learning research.

First, prediction accuracy is accompanied with a degree of uncertainty. While a single split into training and test set gives exact numbers of achieved accuracy, the next split usually shows a different accuracy. This instability is especially large when the sample size is relatively small and the model includes many coefficients. Repeating the splitting process usually leads to a more reliable estimation. However, the estimate remains uncertain and quantifications of uncertainty (e.g., through confidence intervals) are needed.

Second, the data left-out for final model testing should not inform the model (i.e., there should be no 'information leak'). Common dangers are to preselect predictor variables based on their correlation with the outcome *before* splitting the data into training and test sets. Such practices lead to an artificially inflated measure of accuracy. To avoid such biases, all model characteristics should be selected before the model is evaluated on the final testing data. This should be done, for instance, by selecting the best predictors, hyperparameters, and models on dedicated development sets. An optimal procedure is to pre-register the final model in a public repository, collect new data, and report the accuracy that the pre-registered model achieved on the new data.

Third, authors should not selectively report accuracy metrics. For example, models sometimes have seemingly small mean absolute errors, whereas the proportion of explained variance is negligible. This occurs when the outcome variable is tightly clustered around the mean value and making predictions with small errors is therefore easy. Reporting complementary metrics and baseline accuracies prevents misinterpretations.

Fourth, when comparing the predictive value of two competing sets of predictor variables, multiple models should be considered. It is likely that the predictor sets are not equally compatible with all available models. Thus, fitting two regression models and finding that one predictor set leads

to higher accuracy does not imply that this set is always more useful; it merely demonstrates that the set is more useful when using linear regression. Results may differ when using other, potentially more accurate models. Relatedly, practical considerations might steer model selection. A deep neural network which requires immense computational resources might not be the optimal choice if less costly models perform almost as well.

Fifth, common questionable research practices can be exacerbated in machine learning projects. For instance, machine learning models are often preceded by multiple preprocessing steps (e.g., standardizing predictors for ridge regression). Transparency is needed to evaluate and replicate prediction accuracies. One of the most powerful techniques to guard oneself against many of the listed mistakes is to follow the necessities of open science. For machine learning projects, this includes pre-registering and publishing predictions models (through data and code) and making them openly available for review and replication.

Additional topics and readings

The current article provides guidance for psychologists interested in prediction-based research and machine learning. Topics that were either only briefly or not at all discussed include alternative cross-validation techniques (Kohavi, 1995), boosting (Dietterich, 2000), reinforcement learning (Sutton & Barto, 2018), deep learning (LeCun, Bengio, & Hinton, 2015), and advanced data preprocessing/acquisition steps (e.g., for unbalanced data; Chawla, Japkowicz, & Kotcz, 2004). These topics warrant a review of their own, but psychologists interested in applying machine learning themselves certainly benefit from familiarizing themselves with these concepts.

Conclusion

We provide researchers in psychology with concrete guidance on implementing and reviewing machine learning techniques. We highlighted machine learning's focus on generating accurate prediction models. Further, we introduced the main metrics to quantify prediction accuracy, as well as different strategies to evaluate these accuracies on new data. Relatedly, we described the practice of tuning machine learning models through hyperparameters, and selecting the best hyperparameter settings on dedicated development sets, before quantifying the final model's

achieved accuracy. Further, we introduced some of the most common machine learning models, alongside annotated implementations in R code. Finally, we discussed some dangers and questionable practices for implementing machine learning models in psychological research. Together, we hope that the current review and tutorial sections will facilitate research aiming to predict psychological phenomena.

References

Akosa, J. S. (2017). Predictive accuracy: A misleading performance measure for highly imbalanced data. *Proceedings of the SAS Global Forum*.

Alharthi, R., Guthier, B., & El Saddik, A. (2018). Recognizing human needs during critical events using machine learning powered psychology-based framework. *IEEE Access*, *6*, 58737-58753.

Amolo, G. O. (2018). The growth of high-performance computing in Africa. *Computing in Science & Engineering*, *20*, 21-24.

Askin, R. G. (1982). Multicollinearity in regression: Review and examples. *Journal of Forecasting*, *1*, 281-292.

Bachrach, Y., Graepel, T., Kohli, P., Kosinski, M., & Stillwell, D. (2014). Your digital image: Factors behind demographic and psychometric predictions from social network profiles. *13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014*.

Bellos, S., Skapinakis, P., Rai, D., Zitko, P., Araya, R., Lewis, G., ... & Mavreas, V. (2013). Cross-cultural patterns of the association between varying levels of alcohol consumption and the common mental disorders of depression and anxiety: Secondary analysis of the WHO Collaborative Study on Psychological Problems in General Health Care. *Drug and Alcohol Dependence*, *133*, 825-831.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281-305.

Bleidorn, W., & Hopwood, C. J. (2018). Using machine learning to advance personality assessment and theory. *Personality and Social Psychology Review*, https://doi.org/10.1177/1088868318772990

Brandt, M. J. (2017). Predicting ideological prejudice. *Psychological Science*, *28*, 713-722.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5-32.

BRFSS (2005-2010). *SMART Data and Documentation*. Retrieved from https://www.cdc.gov/brfss/smart/Smart_data.htm

Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Special issue on learning from imbalanced data

sets. *ACM Sigkdd Explorations Newsletter*, *6*, 1-6.

Chekroud, A. M., Zotti, R. J., Shehzad, Z., Gueorguieva, R., Johnson, M. K., Trivedi, M. H., ... & Corlett, P. R. (2016). Cross-trial prediction of treatment outcome in depression: a machine learning approach. *The Lancet Psychiatry*, *3*, 243-250.

Claesen, M., & De Moor, B. (2015). *Hyperparameter search in machine learning.* Retrieved from https://arxiv.org/pdf/1502.02127v2.pdf

Dana, J., & Dawes, R. M. (2004). The superiority of simple alternatives to regression for social science predictions. *Journal of Educational and Behavioral Statistics*, *29*, 317-331.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*, 139-157.

Eichstaedt, J. C., Schwartz, H. A., Kern, M. L., Park, G., Labarthe, D. R., Merchant, R. M., … Seligman, M. E. P. (2015). Psychological language on Twitter predicts county-level heart disease mortality. *Psychological Science*, *26*, 159–169.

Figueroa, R. L., Zeng-Treitler, Q., Kandula, S., & Ngo, L. H. (2012). Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, *12*, 8-18.

Ghandeharioun, A., Fedor, S., Sangermano, L., Ionescu, D., Alpert, J., Dale, C., … Picard, R. (2017). Objective assessment of depressive symptoms with machine learning and wearable sensors data. In *Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*.

Golbeck, J., Robles, C., & Turner, K. (2011, May). Predicting personality with social media. In *CHI'11 extended abstracts on human factors in computing systems* (pp. 253-262). ACM.

Goldfeld, K. (2018). *simstudy: Simulation of Study Data*. R package version 0.1.10. https://CRAN.R-project.org/package=simstudy

Hassan, L. M., Shiu, E., & Shaw, D. (2016). Who says there is an intention–behaviour gap? Assessing the empirical evidence of an intention–behaviour gap in ethical consumption. *Journal of Business Ethics*, *136*, 219-236.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In

Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Hindman, M. (2015). Building better models: Prediction, replication, and machine learning in the social sciences. *ANNALS of the American Academy of Political and Social Science*, *659*, 48-62.

Huang, J., Lu, J., & Ling, C. X. (2003, November). Comparing naive Bayes, decision trees, and SVM with AUC and accuracy. In *Null* (p. 553). IEEE.

Hutson, M. (2018). Artificial intelligence faces reproducibility crisis. *Science*, *359*(6377), 725-726.

Joel, S., Eastwick, P. W., & Finkel, E. J. (2017). Is romantic desire predictable? Machine learning applied to initial romantic attraction. *Psychological Science*, *28*, 1478–1489.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255-260.

Kohavi R (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, (pp. 1137–1143).

Kosinski, M., Bachrach, Y., Kohli, P., Stillwell, D., & Graepel, T. (2014). Manifestations of user personality in website choice and behaviour on online social networks. *Machine Learning*, *95*, 357-380.

Koul, A., Becchio, C., & Cavallo, A. (2018). PredPsych: A toolbox for predictive machine learning-based approach in experimental psychology research. *Behavior Research Methods, 50*, 1-16.

Krause, J., Perer, A., & Ng, K. (2016). Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 5686-5697).

Kuhn, M. (2015). A short introduction to the caret package. *R Foundation for Statistical Computing*.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling* (Vol. 26). New York: Springer.

Kuhn, M., & Johnson, K. (n.d.). *Nested resampling with rsample.* Retrieved from http://appliedpredictivemodeling.com/blog/2017/9/2/njdc83d01pzysvvlgik02t5qnaljnd

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436-444.

Lever, J., Krzywinski, M., & Altman, N. (2016). Points of significance: Model selection and overfitting.

Nature Methods, *13*, 703-704.

Lučić, B., & Trinajstić, N. (1999). Multivariate regression outperforms several robust architectures of neural networks in QSAR modeling. *Journal of Chemical Information and Computer Sciences*, *39*, 121-132.

Macmillan, N. A. (2002). Signal detection theory. In H. Pashler & J. Wixted (Eds.), *Stevens' handbook of experimental psychology: Methodology in experimental psychology* (pp. 43-90). Hoboken, NJ: John Wiley & Sons Inc.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective.* Cambridge, MA: MIT Press.

Ng, A. (2018). Size of the dev and test sets. Retrieved from https://www.coursera.org/lecture/machine-learning-projects/size-of-the-dev-and-test-sets-HOby4

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How many trees in a random forest?. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition* (pp. 154-168). Springer: Berlin, Heidelberg.

Park, G., Schwartz, H., Eichstaedt, J. C., Kern, M. L., Kosinski, M., Stillwell, D. J., … Seligman, M. E. P. (2015). Automatic personality assessment through social media language. *Journal of Personality and Social Psychology*, *108*, 934–952.

Piper, M. E., Loh, W. Y., Smith, S. S., Japuntich, S. J., & Baker, T. B. (2011). Using decision tree analysis to identify risk factors for relapse to smoking. *Substance Use & Misuse*, *46*, 492-510.

Plonsky, O., Erev, I., Hazan, T., & Tennenholtz, M. (2017). Psychological Forest: Predicting Human Behavior. In *The Proceedings of the Thirty-first AAAI Conference on Artificial Intelligence (AAAI-17), 656-662*

Raudys, S. J., & Jain, A. K. (1991). Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 13*, 252-264.

Ribeiro, J. D., Pease, J. L., Gutierrez, P. M., Silva, C., Bernert, R. A., Rudd, M. D., & Joiner Jr, T. E. (2012). Sleep problems outperform depression and hopelessness as cross-sectional and

longitudinal predictors of suicidal ideation and behavior in young adults in the military. *Journal of Affective Disorders*, *136*, 743-750.

Rimfeld, K., Kovas, Y., Dale, P. S., & Plomin, R. (2016). True grit and genetics: Predicting academic achievement from personality. *Journal of Personality and Social Psychology, 111*, 780-789.

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, *1*, 317-328.

Scherer, S., Stratou, G., Gratch, J., & Morency, L. P. (2013). Investigating voice quality as a speaker-independent indicator of depression and PTSD. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*.

Schwartz, H. A., Eichstaedt, J. C., Kern, M. L., Dziurzynski, L., Ramones, S. M., Agrawal, M., ... & Ungar, L. H. (2013). Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS One*, *8*(9), e73791.

Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, *22*, 1359–1366.

Sumner, C., Byers, A., Boochever, R., & Park, G. J. (2012). Predicting dark triad personality traits from twitter usage and a linguistic analysis of tweets. In *Proceedings - 2012 11th International Conference on Machine Learning and Applications, ICMLA 2012* (Vol. 2, pp. 386–393).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Boston, MA: MIT Press.

Veale, M., & Binns, R. (2017). Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data. *Big Data & Society*, *4*, 2053951717743530.

Walsh, C. G., Ribeiro, J. D., & Franklin, J. C. (2017). Predicting risk of suicide attempts over time through machine learning. *Clinical Psychological Science*, *5*, 457-469.

Wang, N., Kosinski, M., Stillwell, D. J., & Rust, J. (2014). Can well-being be measured using Facebook status updates? Validation of Facebook's gross national happiness index. *Social Indicators Research*, *115*, 483-491.

Wang, Y., & Kosinski, M. (2018). Deep neural networks are more accurate than humans at detecting sexual orientation from facial images. *Journal of Personality and Social Psychology*, *114*, 246–257.

Yarkoni, T., & Westfall, J. (2017). Choosing prediction over explanation in psychology: Lessons from machine learning. *Perspectives on Psychological Science*, *12*, 1100–1122.

Youyou, W., Kosinski, M., & Stillwell, D. (2015). Computer-based personality judgments are more accurate than those made by humans. *Proceedings of the National Academy of Sciences, 112*, 1036-1040

Zhang, T., & Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval, 4*, 5-31.