Machine learning and prediction in psychological assessment: Some promises and pittfalls

Marjolein Fokkema [1]

Dragos Iliescu [2]

Samuel Greiff [3]

Matthias Ziegler [4]

1 Institute of Psychology, University Leiden, The Netherlands
2 Faculty of Psychology and Educational Sciences, University of Bucharest, Romania
3 Institute of Cognitive Science and Assessment (COSA), University of Luxembourg, Luxembourg
4 Department of Psychology, Humboldt University Berlin, Germany

"When we raise money it's AI, when we hire it's machine learning,

and when we do the work it's logistic regression."

(Tweet by bio-statistician Daniella Witten; original author unknown)

Abstract

Modern prediction methods from machine learning (ML) and artificial intelligence (AI) are becoming increasingly popular, also in the field of psychological assessment. These methods provide unprecedented flexibility for modeling large numbers of predictor variables, and non-linear associations between predictors and response. In this paper, we aim to take a look at what these methods may contribute for the assessment of criterion validity, and what their possible drawbacks are. We apply a range of modern statistical prediction methods to a dataset for predicting the university major completed, based on the subscales and items of a scale for vocational preferences. The results indicate that logistic regression combined with regularization performs strikingly well already in terms of predictive accuracy. More sophisticated techniques for incorporating non-linearities can further contribute predictive accuracy and validity, but often marginally.

Introduction

Machine learning (ML) and artificial intelligence (AI) are by now familiar buzzwords in many fields of empirical research, including psychology. In the field of psychological assessment, interest and application of these methods is also increasing. We believe ML and AI have the potential to contribute to our field, but the buzz around these methods can be reminiscent of the tale of the emperor's new clothes. We believe when it comes to ML and AI, the emperor is in fact wearing clothes, but they are often not so new. Many of the techniques presented as machine learning (e.g., cross validation, regularization, ensembling) have long been known and fruitfully applied in statistics, psychometrics and psychological assessment.

In the current paper, we look at how several modern methods from statistics, ML and AI may contribute to our field, and what their limitations are. Note, we will use the term statistical learning to refer to both traditional and more recent (sometimes referred to as ML or AI) tools for data analysis. As already suggested by the motto at the start of this paper, there is no consensus on whether specific methods are statistical, AI or ML, so we avoid making the distinction altogether. We focus instead on the aim shared by all these methods: Learning from data. We focus on methods for prediction of a response (dependent, criterion) variable, often referred to as supervised learning methods. Thus, unsupervised learning methods (e.g., factor analysis, clustering, correlation networks, topic models from natural language processing) are outside the scope of the current paper.

Recent shifts in statistical learning

Modern developments in statistical learning methodology have yielded two main shifts:

1. Increased focus on prediction.

2. Increased flexibility: Modern methods allow for capturing non-linear associations and/or modeling large numbers of predictors.

We believe that the first shift is highly beneficial for our field, because prediction of behavior is one of the core tasks of psychological assessment. Accurate evaluation of predictive accuracy is needed to provide evidence for the validity of test score interpretations, but also when more complex decision systems are developed for data-driven decision making. Traditionally, the field of psychology at large has been mostly interested in *explanation*, or developing and testing theories of human behavior. This has sometimes led researchers to overlook *prediction*, perhaps because their main aim was to explain behavior. A theory, however, can only explain real-world phenomena to the extent that it can accurately predict them (Yarkoni & Westfall, 2017).

The traditional focus on explanation may have motivated researchers to compute effect sizes (e.g., R^2, Cohen's *d*) on *training* data; that is, using observations that were also used to fit the model. This leads to overly optimistic effect size estimates. More realistic effect sizes can be obtained, for example, through cross validation: By computing effect sizes on a sample of observations *not* used for fitting the model (de Rooij & Weeda, 2020). It is interesting to note that cross validation has been discussed in the field of assessment for almost a century (Larson, 1931; Mosier, 1951), but its use has become more common only in recent years.

We believe that the second shift, towards flexibility, brings both promises and pitfalls for our field. Promises, because few if any real-world phenomena behave in a purely linear and additive fashion. Pitfalls, because assumptions of linearity and additivity (i.e., no interactions) are very powerful when it comes to inference and interpretation, even if they are known to be only partialy true. This means that the often one-sided focus on maximizing predictive accuracy

in AI and ML are of limited value when it comes to understanding and explaining behavior, and the role of these methods is, at best, in hypotheses generation.

Of note, unrestricted flexibility leads to overfitting and poorly generalizable results. In statistical learning, this has been formalized in the *bias-variance trade-off*. Informally, this trade-off states that the more flexible a model is allowed to approximate any possible shape of association between predictors and response (i.e., the lower the *bias*), the worse the model will generalize to new samples from the same population (i.e., the higher the *variance*). To obtain optimally generalizable results for a given data problem and sample (size), bias and variance should thus be carefully balanced through choosing an appropriate model-fitting procedure.

Bias can be increased and variance reduced in various ways, including:

- Limiting the complexity of the functional form (e.g., model only linear associations; model only main effects);
- Limiting the number of potential predictors used (e.g., include only few predictors; use sum or factor scores instead of item scores as predictors);
- Regularized estimation procedures (e.g, lasso, ridge, or elastic net regression; use of Bayesian priors);
- Ensembling (e.g., in psychometrics, multiple items are often aggregatec into subscale or factor scores; in ML predictions of so-called base learners are often aggregated into the predictions of an ensemble).

If the bias is well-chosen and realistic, generalizability of the fitted model will be improved. In other words: we can buy predictive power by making realistic assumptions. If the bias is not well chosen, predictive accuracy and generalizability will obviously suffer.

Empirical example

We aim to illustrate and compare the use of a range of statistical learning techniques through a data-analytic example. We focus on a predictive validity question: To what extent do the item and subscale scores on a measure of vocational preferences predict the type of university major completed? Note, we will not focus on substantive aspects of this prediction problem, but we use it to illustrate more general principles of flexibility, overfitting and interpretability in predictive modeling in assessment. In test development, providing evidence for criterion validity of the scores is vital as it often is used by practitioners to choose between existing tests. Thus, establishing test-criterion related evidence is a fundamental part of test construction. Therefore, it seems obvious that the potential of statistical learning procedures should come to bear here.

Readers interested in replicating our analyses will find our annotated code and results in the ESM.

Method

Dataset

We use a dataset from the Open Psychometrics Project (https://openpsychometrics.org/_rawdata/). Data were collected through their website from 2015 to 2018. Respondents answered items on vocational preferences, personality and

sociodemographic characteristics. The sample likely does not represent a random sample from a well-defined population, which would normally be required for evaluating a test's validity.

We investigate predictive validity of the RIASEC vocational preferences scales (Liao et al., 2008). The RIASEC uses six occupational categories from Holland's Occupational Themes (Holland, 1959) theory: Realistic (R), Investigative (I), Artistic (A), Social (S), Enterprising (E), and Conventional (C). There are 8 items for each category, each describing a task (e.g., R6: "Fix a broken faucet" or I2: "Study animal behavior"), to which respondents answer on a 1-5 scale, with 1=Dislike, 3=Neutral, 5=Enjoy. The items are presented in Appendix A. The research question from an assessment perspective is whether the RIASEC scores can be used to predict the university major completed. Such evidence could support the use of the scale in applied settings; moreover, the results could inform decision rules.

From the full dataset, we selected participants who completed at least a university degree, yielding a sample of N = 55,593 observations. As the criterion we take a binary variable, indicating whether respondents majored in Psychology (19.42%), or in a different topic (80.58%). Further descriptive statistics of the sample are presented in Appendix B.

Model fitting and evaluation

We fitted a range of traditional and more recent (ML/AI) methods to model the relation between the RIASEC scores and the criterion. This will show the magnitude of differences in performance such algorithms typically yield. Also, it exemplifies the researcher degrees of freedom in such cases and it is thus important to use separate data for fitting and evaluation of the models.

We separated the data into 75% training observations and 25% test observations. Our training sample thus consists of 41,694 respondents, of which 19.46% majored in psychology. Our test sample consisted of 13,899 respondents, of which 19.3% majored in psychology. Other train and test sample sizes may sometimes be preferred, or k-fold CV. Considering the current sample size, however, we do not expect the results to be very sensitive to this choice.

All analyses were performed in **R** (version 4.1.0, R Core Team, 2021). We tuned the model-fitting parameters for all models using resampling and cross validation (CV) on the training observations. We did *not* tune the parameters of the generalized additive models (GAMs), because we expected the defaults to work well out of the box. The specific packages used, as well as the code and results of tuning and fitting the models are provided in the ESM.

We evaluated predictive accuracy of the fitted models by computing the Brier score on test observations. The use of accuracy measures derived from the confusion matrix of actual and predicted classes, like the misclassification error, sensitivity (or recall), positive predictive value (or precision) are pervasive in the machine learning literature. However, these measures disregard the quality of predicted probabilities from a fitted model and we therefore recommend against their use for evaluating predictive accuracy. Methods for predicting a binary outcome should not only provide a predicted class, but also a predicted probability to quantify the uncertainty of the classification. To evaluate performance, the quality of this probability forecast should thus be evaluated (Gneiting & Raftery, 2007).

The Brier score is the mean squared error of the predicted probabilities:

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{p}_i)^2$$

Where y_i is the observed outcome for observation i, taking a value of 0 or 1; $\hat{p}_i$ is the model's predicted probability. We computed Brier scores on training as well as on test observations; thus N can be taken to be the training or the test sample size. A Brier score equal to the variance of y indicates performance no better than chance (in the current dataset, the variance was 0.1946 for training and 0.193 for test data). To obtain a pseudo-R^2 measure, we take 1 minus the Brier score divided by the variance of y, which takes values between 0 (indicating performance no better than chance) and 1 (indicating perfect accuracy).

Results

Considering the two shifts in predictive modeling discussed in the Introduction, we fitted all models twice: Once using subscale scores, once using item scores. This allows us to evaluate whether our conclusions generalize between the two approaches, and to gauge the effect of having a larger pool of predictor variables (which are likely more noisy but possibly more informative of the criterion).

(penalized) Logistic regression

Our benchmark traditional method is an additive generalized linear model (GLM): Logistic regression. If CV results indicated predictive accuracy could be improved by application of a lasso or ridge penalty, we applied it. For prediction with subscale scores, no penalization was found to be optimal. The estimated coefficients for the subscale scores are presented in Figure 3; as expected with the currently large sample size, all subscale scores obtained p-values $< .001$. The strongest effect was a positive effect from the Social preferences scale, and the weakest effect was a negative effect from the Conventional preferences scale.

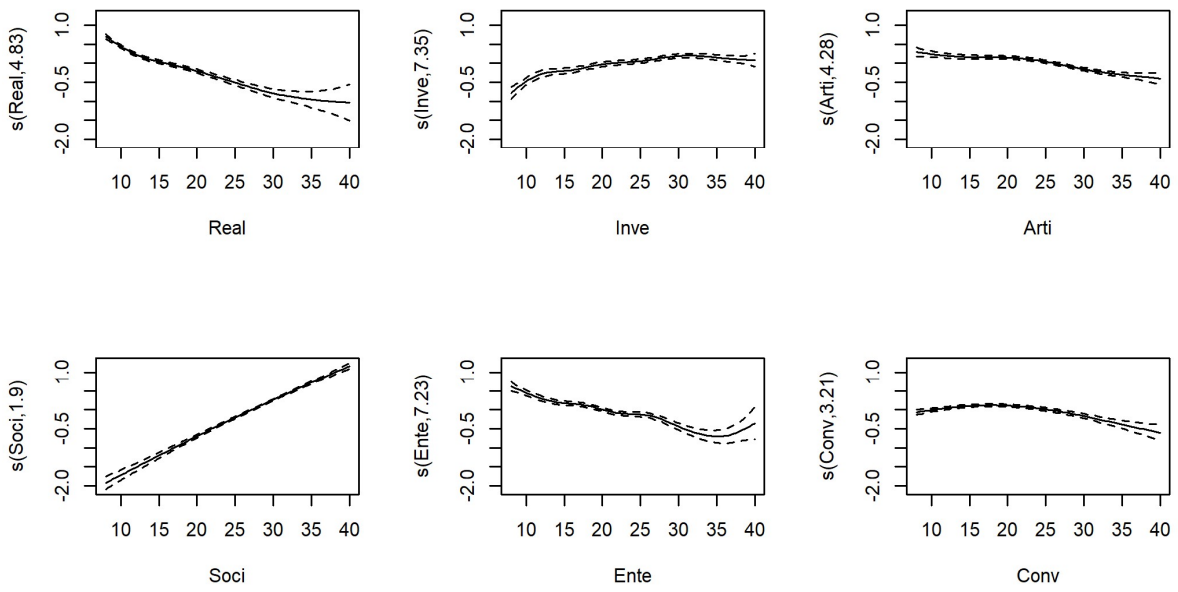
For prediction with item scores, CV indicated optimal performance for a small but non-zero value of the lasso penalty. With an increasing number of predictor variables, this beneficial effect of penalization (or regularization) is generally expected. The resulting item-level coefficients are depicted in Figure C1 (ESM). The item coefficients indicate similar relevance of the subscales as the previous analysis, but provide a more finegrained view of individual item's contributions.

Generalized additive model

Next we fitted generalized additive models (GAMs) with smoothing splines. Smoothing splines allow for flexibly approximating non-linear shapes of association between predictor and response. At the same time, overfitting is prevented by penalizing the wigglyness of the fitted curves. The splines provide a flexible but smooth approximation to the observed datapoints, while the additive structure provides ease of interpretability because the estimated effects are *conditional* (i.e., keeping the values of all remaining predictors fixed). Bringmann et al. (2017) provide a more detailed introduction to GAMs aimed at psychologists.

Figure 1

Fitted smoothing spline curves for each of the RIASEC subscales



Note. Values on the y-axis reflect the effect on the log-odds of having completed a university major in psychology.

The splines fitted to the subscale scores are presented in Figure 1. Similar to the GLM, we see positive effects of the Social and Investigative subscales, and negative effects of the Realistic, Artistic, Enterprising and Conventional subscales. The Social preferences subscale shows a near-linear effect, while the other subscales' effects clearly exhibit some stronger non-linearity. An advantage of GAMs is that they allow for inference: they provide χ^2 tests to

evaluate the significance of the effect of each predictor variable. As expected with the current large sample size, all subscale scores obtained p-values $< .001$.

For the GAM fitted using item scores, we also applied penalization, as this was expected to be beneficial for prediction, as similarly observed in the GLM. We do not depict the fitted curves for space considerations here, but Figures 3 and C1 (ESM) show the χ^2 values per subscale and per item, respectively. The figures indicate very similar effects of the predictors, between the (penalized) GLMs and GAMs.

We now leave the realm of additive models, and set about fitting models that allow for capturing interaction effects:

Decision tree

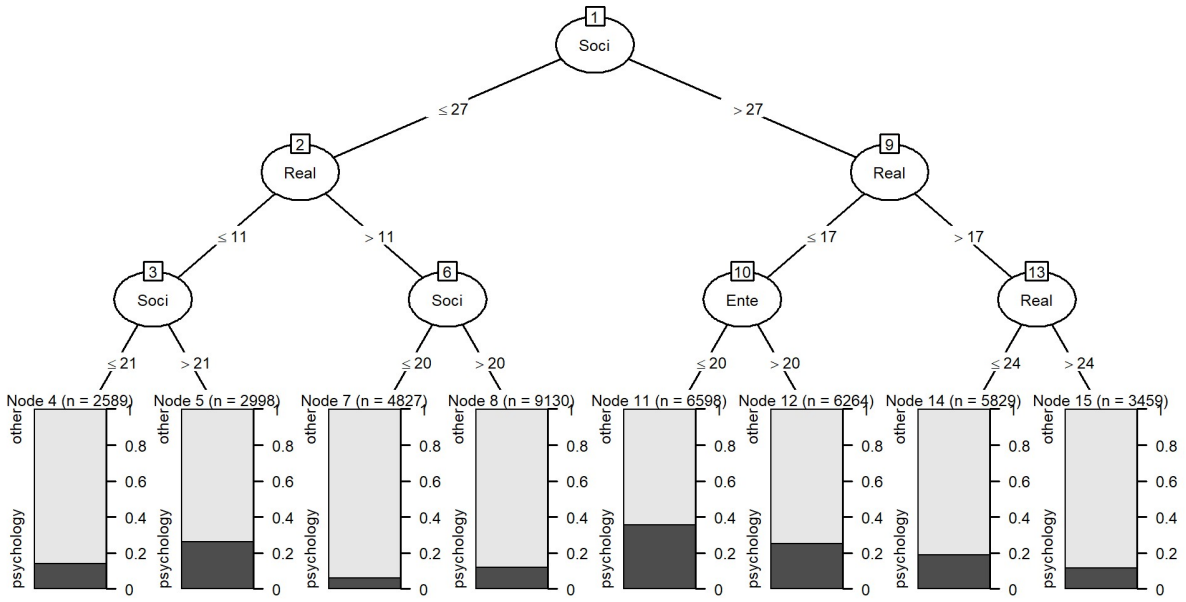
We fit a single decision tree using the conditional inference tree algorithm (Hothorn et al., 2006). This algorithm eliminates the variable selection bias present in many other decision-tree algorithms. Decision-tree methods and variable selection bias are discussed in more detail by Strobl et al. (2009), who provide a comprehensive introduction aimed at psychologists. According to our CV results with the subscales as predictors, a tree depth of seven was optimal, yielding a tree with $2^6 = 128$ terminal nodes. Thus, for this data problem, the most accurate tree is surely not the most interpretable. The predictor variables selected by the trees are depicted in Figures 3 and C1 (ESM).

For illustration, Figure 2 shows the decision tree fitted to the subscale scores, pruned to a depth of three. The Social, Realistic and Enterprising preferences subscales were used in the first splits of the tree. The bars in the terminal nodes depict the proportion of participants within each node that majored in psychology. Thus, the Social subscale shows a positive effect, and the

Realistic and Enterprising subscales show a negative effect. With regards to possible interactions, note that split number 10 suggests that the Enterprising subscale appears relevant only for higher values of the Social, and lower values of the Realistic subscales. However, such a split may also reflect additive effects combined with multicollinearity. Although decision trees can *capture* interaction effects, they cannot be straightforwardly used to statistically test their significance; a disadvantage shared by virtually all flexible ML and AI techniques.

Figure 2

Conditional inference tree pruned to a depth of three



Although decision trees are easy to interpret, they suffer more strongly from instability than GLMs and GAMs. With instability, we mean that a small change in the training data can lead to large changes in the resulting model. The cause of this instability partly lies in the rather rough cuts made in the tree. Tree ensembling methods capitalize on this instability. They derive a large number of learners (e.g., trees), each fitted on different versions of the training dataset. Different versions of the training data can be generated, for example, by taking bootstrap samples from the training data, a method also known as bagging. More powerful tree ensembling methods are random forests and boosting. Introductions about tree ensemble methods aimed at psychologists can be found in Strobl et al. (2009) and Miller et al. (2016).

Gradient boosted tree ensemble

The first tree ensemble method we apply to the data is a gradient boosted ensemble. Boosting uses sequential fitting of so-called weak learners to create a strong learner. Weak learners are simple models, that provide predictive accuracy (slightly) better than chance. When boosting trees, we use weak learners in the form of small trees, with only a few splits. Sequential learning means that each consecutive tree is adjusted for the predictions of previous trees. In effect, observations that were well (badly) predicted by previous trees receive less (more) weight when fitting the next tree.

A disadvantage of decision tree ensembles is their black box nature: While individual trees are generally easy to interpret, an ensemble of trees is impossible for humans to grasp. Therefore, so-called variable importance measures have been developed for interpretation of tree ensembles, which aim to quantify the effect of predictor variable on the predictions of the ensemble. In this paper, we use the permutation importances proposed by Breiman (2001). These quantify how much an ensemble's predictive accuracy would be reduced, if the values of each of

the predictor variables are randomly shuffled. The variable importances of the fitted gradient boosting ensembles are depicted in Figures 3 and C1 (ESM).

Importance measures provide a useful ranking of the contributions of each predictor to the ensemble's predictions, but should be interpreted with care. They should not be used to judge the significance of the effect of predictors; tree ensembles can easily include predictors in the model which in fact have no effect on the outcome. Furthermore, there are many ways to compute variable importance measures, which each may yield different conclusions, especially when predictors are correlated (Nicodemus et al., 2010; Nicodemus, 2011; Strobl et al., 2007, 2008). Especially with correlated predictors, permuting the values of predictor variables may lead to unrealistic data patterns. These issues illustrate the interpretability problems which come along with complex prediction methods such as tree ensembles, support vector machines and (deep) neural networks.

Random forest

Another popular decision-tree ensembling method are random forests (Breiman, 2001). Like boosted tree ensembles, random forests fit a large number of decision trees. The ensemble's predictions are simply the average over the predictions of the individual trees. Random forests do not employ sequential learning: each tree is fitted without adjusting for predictions of the other trees in the ensemble. Unlike boosting, random forests employ trees with many splits: in the original algorithm of (Breiman, 2001), trees were grown as large as possible. Later studies, however, have shown that large trees can lead to unstable results when there are many correlated predictors that are at best weakly correlated to the response (Segal, 2004). It is thus beneficial to grow large, but not too large trees.

The most characteristic feature of random forests is how it selects variables for splitting: A random sample of *mtry* candidate predictor variables is considered for every split in every tree. From this set of predictor variables, the best splitting variable and value is selected. Without random selection of variables, each tree of the ensemble would likely use the same set of relatively strong predictors, and thus be very similar. Averaging over many very similar trees is unlikely to improve predictive accuracy. Thus, the randomization makes the trees more dissimilar, which likely improves performance of the ensemble.

The variable importances of the fitted random forests are depicted in Figures 3 and C1 (ESM).

Prediction rule ensembling

Prediction rule ensembles (PRE) aim to strike a balance between the high predictive accuracy of decision tree ensembles, and the ease of interpretability of single decision trees and GLMs (Fokkema, 2020; Fokkema & Strobl, 2020). The method fits a boosted decision tree ensemble to the training dataset, and takes every node from every tree as a rule. For example, membership of Node 2 in the tree in Figure 2 can be coded using a single condition: *Social* $\leq$ *27*. Membership of Node 14 involves multiple conditions: *Social* > *27 & Realistic* > *17 & Realistic* $\leq$ *24*. Each of these nodes can be seen as a dummy-coded rule, which takes a value of 1 if the conditions apply, and 0 if not.

PRE applies lasso regression on a dataset consisting of both these rules and the original predictor variables. As such, it combines the strengths of penalized regression and tree ensembles. Although the boosted decision tree ensemble will initially contribute a large number of nodes (rules), use of lasso regression will give many of these rules a weight of zero, which

removes them from the final ensemble. As such, PRE provides a sparse and interpretable final model.

The PRE we fitted using the subscale scores consisted of 48 rules, providing a great simplification compared to the > 500 trees of the boosted ensemble and random forest. Note that the current dataset is exceptionally large, which tends to result in longer rule lists when only predictive accuracy is optimized, because very large samples allow for capturing highly nuanced effects. In Table 1, the six most important rules are shown.

Table 1

Six most important rules in the prediction rule ensemble

Description	**Coefficient**
Soci > 27 & Ente <= 31 & Conv <= 30	0.182
Soci > 23 & Ente <= 29 & Real <= 24	0.181
Real > 10 & Soci <= 35	-0.175
Real <= 22 & Soci > 19 & Inve > 18	0.138
Inve > 10 & Real <= 13	0.120
Conv <= 23 & Arti <= 29 & Soci > 21	0.112

Note that each rule has obtained an estimated coefficient, which are simply logistic regression coefficients: They reflect the expected increase in log-odds if the conditions of the rule apply. PRE also provides variable importance measures, which are presented for the fitted ensembles in Figures 3 and C1 (ESM). An introduction to PRE aimed at psychologists is provided in Fokkema & Strobl (2020).

k Nearest neighbours

A prime example of a highly flexible method, perhaps the most non-parametric method of all, is the method of *k*-nearest neighbours (kNN). In fact, kNN does not even fit a model; it merely remembers the training observations. To compute predictions for new observations, kNN computes the distance of a new observation to all training observations, in order to find the *k* nearest ones (the neighbours). It then takes the mean of the response variable over these *k* neighbours as the predicted value. This provides the greatest possible flexibility of all prediction methods, as it does not impose *any* a-priori restriction on the shape of association between predictors and response. This flexibility is both the strength and weakness of kNN: with increasing numbers of of predictor variables, the performance of kNN worsens fast. Only in lower dimensions is the great flexibility of kNN beneficial.

kNN has only a single tuning parameter: *k*. With larger values of *k*, the predicted value for a new observation averages over a larger number of observations (neighbours). Thus, higher values of *k* yield lower variance, but higher bias. Furthermore, because kNN is a fully distance-based method, in which all variables obtain the same weight of 1, the method does not provide *any* measure of effect of individual variables, and we thus do not plot variable contributions for kNN here.

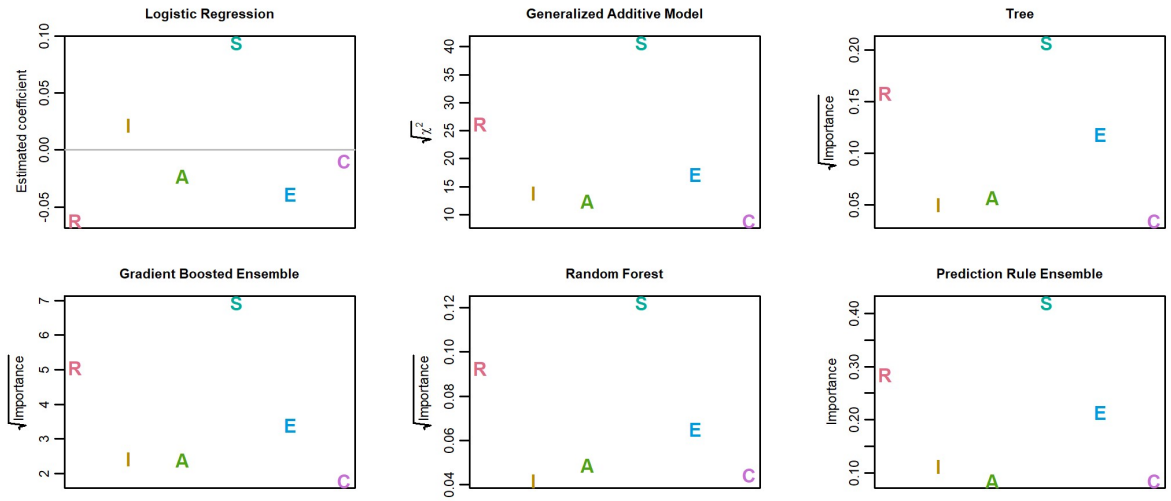
Model comparisons

Variable contributions

Figure 3 depicts the variable contributions in the models fitted using RIASEC subscale scores. Note that the coefficients of the logistic regression reflect both direction and strength of the effects. For the other models, the variable contributions only reflect the strength of the variables' effects. Figure 3 shows similar variable contributions for all methods: The Social preferences are most important for predicting university major completed, followed by Realistic, followed by Enterprising preferences, while the Conventional and Artistic subscales contribute least. The variable contributions for models fitted using the item scores as predictors yielded similar conclusions and are provided and discussed in Figure C1 (ESM).

Figure 3

Variable contributions for each of the models fitted using RIASEC subscale scores as predictors



Note. Coefficients in the logistic regression and importance measures of the prediction rule ensemble are on the scale of standard deviations. Importance measures for the other methods are on the scale of variances; for those methods, the square roots are plotted.

In Figure 4, pseudo-R^2 values on train and test data are depicted with confidence intervals. Note that the confidence intervals for test data are systematically wider than for training data, but this is mostly due to the much larger number of training observations.

The left panel of Figure 4 shows that with the subscale score, the best test set performance was obtained with the boosted tree ensemble, and very closely followed by the generalized additive model, prediction rule ensemble, random forest, k nearest neighbours, logistic regression, and finally the decision tree. This latter result is rather unsurprising: a single decision tree is generally expected to have somewhat lower predictive accuracy, but they often 'win' in terms of interpretability, which can be observed in Figure 4, which shows that the decision tree uses only about half of the items for prediction. The boosted tree ensemble performing best is also not very surprising, giving its top-ranking performance in forecasting competitions.

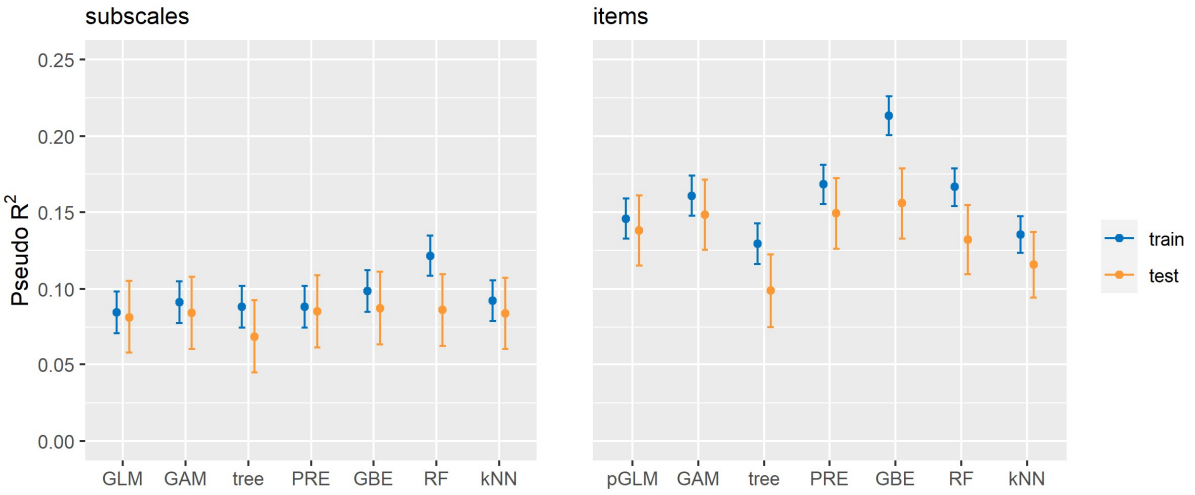
From the left panel in Figure 4, we obtain the following take-aways:

1. On the test data, none of the methods performs significantly worse or better than any of the other methods.
2. The difference between training and test performance increases with increasing flexibility. The methods that incorporate linear main effects (logistic regression, GAM, PRE) show the smallest difference in performance between training and test data. These methods thus appear least likely to overfit.
3. The more flexible methods (single tree, kNN, boosted ensemble, random forest) show greater susceptibility to overfitting.

The subscale scores did not provide strong predictive power, with R^2 indicative of a moderate effect. Using item scores as predictors yielded a substantial (about 50%) increase in variance explained. Again, best performance on the test data was obtained with the boosted tree ensemble. This time, it was followed by the prediction rule ensemble, then the generalized additive model, logistic regression, random forest, *k* nearest neighbours, and finally the decision tree.

Figure 4

Predictive accuracy on train and test observations for each of the models fitted on subscale scores (left panel) and items scores (right panel)



Note. (p)GLM = (penalized) logistic regression; GAM = generalized additive model with smoothing splines; PRE = prediction rule ensemble; GBE = gradient boosted tree ensemble; RF = random forest; kNN = k nearest neighbours.

From the right panel in Figure 4, we can add to our earlier take-aways:

4. With a larger number of predictors, differences in performance between the methods become more pronounced, but none of the more sophisticated methods significantly (or substantially) outperforms the GLM with lasso penalty.

5. With a larger number of predictors, the difference in performance between training and test data becomes more pronounced. Higher dimensionality creates more opportunity for overfitting, even though all methods feature powerful built-in overfitting control.

Discussion

Our conclusions can be succinctly summarized as: Logistic regression is hard to beat. Linear main effects models (i.e., (penalized) GLMs) tend to capture most of the explainable variance. This finding corresponds to a range of previous studies noting a lack of (substantial or significant) benefit of sophisticated machine learning methods over (penalized) regression, in prediction problems from psychology and medicine [e.g., Elleman et al. (2020); Littlefield et al. (2021); ChriyJie19; Gravesteijn et al. (2020); Nusinovici et al. (2020); Lynam et al. (2020)].

Sophisticated methods can only improve upon linear main-effects models by capturing more nuanced non-linearities and interactions. Almost by definition, these effects are of smaller size. Capturing these smaller, more nuanced effects comes at the price of an increased tendency to overfit. To reliably approximate small effects, much larger sample sizes are needed. Even if sophisticated methods outperform simpler methods like logistic regression in terms of predictive

accuracy on test data, their tendency to overfit and their black-box nature may make them less suited for increasing scientific understanding, and/or making influential decisions about individuals (e.g., clinical or selection settings).

Perhaps GAMs and PREs may provide the most steady improvement on (penalized) GLMs. They are essentially GLMs with added flexibility for capturing non-linearities, but provide robust overfitting control and also retain interpretability. Especially GAMs may provide the 'best of both worlds': They provide the flexibility of modern statistical learning, robust overfitting control and allow for performing statistical inference. Most flexible machine-learning methods especially fall short in terms of the latter, which limits their use for increasing scientific understanding and theory development.

Our finding that item scores can provide better predictive accuracy than subscale scores corresponds to previous studies (e.g., Seeboth & Mõttus, 2018; Stewart et al., 2021). As also noted by Yarkoni (2020), a large number of item scores will outperform any predictive model fitted on subscale scores, given a large enough sample size. At the same time, a handful of subscale scores is easier to interpret and use than hundreds of personality items. Also, with smaller samples (e.g., $N = 300$ or 500), including prior knowledge about the subscale structure, through the use of subscale or factor scores, may likely improve predictive accuracy (de Rooij et al., under review).

Big-data applications involving, for example, image-, video- and text-based analytics may exhibit stronger patterns of non-linearity and interaction than the analytic example presented here. More sophisticated methods like deep neural networks may even be called for in such applications. However, similar rules of sampling and statistics apply in such applications: The more nuanced the patterns that we want to capture, the larger the sample sizes required. Sample

size requirements for artificial neural networks by far exceed the sample sizes common in our field (e.g., Alwosheel et al., 2018). There is no doubt that image, text, audio, video and sensor-based data (will) provide novel ways of assessing psychological traits (Boyd et al., 2020; Gillan & Rutledge, 2021). Their relatively unobtrusiveness opens up new avenues for assessment, but the black-box nature of algorithms that can capture complex non-linear effects also brings ethical risks (Boyd et al., 2020; Rudin, 2019).

The focus on predictive accuracy brought about by recent statistical, ML and AI methods is beneficial for the field of assessment. We should, however, guard against a blind focus on maximizing predictive accuracy on test observations, as this disregards two important issues:

- Data points analysed in, for example, research settings or forecasting competitions may likely differ from the data points that the predictive model will be applied to in practice. These differences may be subtle in relatively closed, low-stakes systems, like online recommender systems. Much psychological assessment is, however, focused on offline, out-of-lab human behavior, often with high stakes. Generalizing research findings to the real world remains difficult; external validity has not become irrelevant all of a sudden. Gains in predictive accuracy in controlled research settings may be swamped by practical aspects of data problems, like population drift, measurement error, ethics, interpretability, and data-collection costs (Efron, 2020; Fokkema et al., 2015; Hand, 2006; Luijken et al., 2019; Rauthmann, 2020).
- From both an ethical and scientific perspective, validity has become more (not less!) important with newer and bigger data sources. A blind focus on predictive validity leads to black-box assessment procedures with limited content, internal and construct validity. For opening the black box, there is an important role for the field of psychological

assessment and psychometrics. Not only by applying our existing theory, evidence and methods, but also by continually improving, adopting and developing them (Alexander III et al., 2020; Bleidorn & Hopwood, 2019; Iliescu & Greiff, 2019; Tay et al., 2020).

Finally, although modern statistical prediction methods have certainly improved our ability to predict, attribution and interpretation have not become easier. Attribution (assigning significance to individual predictors) requires strong individual predictors and large sample sizes (Efron, 2020). This task only becomes more difficult when datasets contain increasing numbers of predictors with modest effects. The task also becomes more difficult with methods that can capture increasingly nuanced non-linear and interaction effects. A range of interpretation tools for black box models have been proposed (e.g., variable importances, LIME, Shapley values, SHAP). However, the accuracy of their explanations cannot be quantified (Carvalho et al., 2019; Ross et al., 2017), and their inner workings pose another black box to most users, resulting in misinterpretation and misuse (Kaur et al., 2020; Kumar et al., 2020; Rudin, 2019; Waa et al., 2021). With large numbers of predictors, fitted models become inherently difficult to interpret and black-box interpretation tools are unlikely to help with this. Thus, while flexible models might help to inform theory building, their use for making decisions in assessment procedures aimed at individuals is currently limited.

References

Alexander III, L., Mulfinger, E., & Oswald, F. L. (2020). Using big data and machine learning in personality measurement: Opportunities and challenges. *European Journal of Personality*, *34*(5), 632–648. https://doi.org/10.1002/per.2305

Alwosheel, A., Cranenburgh, S. van, & Chorus, C. G. (2018). Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis. *Journal of Choice Modelling*, *28*, 167–182. https://doi.org/10.1016/j.jocm.2018.07.002

Bleidorn, W., & Hopwood, C. J. (2019). Using machine learning to advance personality assessment and theory. *Personality and Social Psychology Review*, *23*(2), 190–203. https://doi.org/10.1177/1088868318772990

Boyd, R. L., Pasca, P., & Lanning, K. (2020). The personality panorama: Conceptualizing personality through big behavioural data. *European Journal of Personality*, *34*(5), 599–612. https://doi.org/10.1002/per.2254

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Bringmann, L. F., Hamaker, E. L., Vigo, D. E., Aubert, A., Borsboom, D., & Tuerlinckx, F. (2017). Changing dynamics: Time-varying autoregressive models using generalized additive modeling. *Psychological Methods*, *22*(3), 409. https://doi.org/10.1037/met0000085

Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, *8*(8), 832. https://doi.org/10.3390/electronics8080832

de Rooij, M., Karch, J. D., Fokkema, M., Bakk, Z., Pratiwi, B. C., & Kelderman, H. (under review). *SEM-based out-of-sample predictions*.

de Rooij, M. de, & Weeda, W. (2020). Cross-validation: A method every psychologist should know. *Advances in Methods and Practices in Psychological Science*, *3*(2), 248–263. https://doi.org/10.1177/2515245919898466

Efron, B. (2020). Prediction, estimation, and attribution. *International Statistical Review*, *88*, S28–S59. https://doi.org/10.1111/insr.12409

Elleman, L. G., McDougald, S. K., Condon, D. M., & Revelle, W. (2020). That takes the BISCUIT: Predictive accuracy and parsimony of four statistical learning techniques in personality data, with data missingness conditions. *European Journal of Psychological Assessment*, *36*(6), 948. http://dx.doi.org/10.1027/1015-5759/a000590

Fokkema, M. (2020). Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software*, *92*(1), 1–30. https://doi.org/10.18637/jss.v092.i12

Fokkema, M., Smits, N., Kelderman, H., & Penninx, B. W. (2015). Connecting clinical and actuarial prediction with rule-based methods. *Psychological Assessment*, *27*(2), 636. https://doi.org/10.1037/pas0000072

Fokkema, M., & Strobl, C. (2020). Fitting prediction rule ensembles to psychological research data: An introduction and tutorial. *Psychological Method, 25*(5), 636–652. https://doi.org/10.1037/met0000256

Gillan, C. M., & Rutledge, R. B. (2021). Smartphones and the neuroscience of mental health. *Annual Review of Neuroscience*, *44*. https://doi.org/10.1146/annurev-neuro-101220-014053

Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, *102*(477), 359–378. https://doi.org/10.1198/016214506000001437

Gravesteijn, B. Y., Nieboer, D., Ercole, A., Lingsma, H. F., Nelson, D., Van Calster, B., Steyerberg, E. W., Åkerlund, C., Amrein, K., Andelic, N., & others. (2020). Machine learning algorithms performed no better than regression models for prognostication in traumatic brain injury. *Journal of Clinical Epidemiology*, *122*, 95–107. https://doi.org/10.1016/j.jclinepi.2020.03.005

Hand, D. J. (2006). Classifier technology and the illusion of progress. *Statistical Science*, *21*(1), 1–14. https://doi.org/10.1214/088342306000000060

Holland, J. L. (1959). A theory of vocational choice. *Journal of Counseling Psychology*, *6*(1), 35. https://doi.org/10.1037/h0040767

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, *15*(3), 651–674. https://doi.org/10.1198/106186006X133933

Iliescu, D., & Greiff, S. (2019). The impact of technology on psychological testing in practice and policy. *European Journal of Psychological Assessment*, *35*(2), 151–155. https://doi.org/10.1027/1015-5759/a000532

Kaur, H., Nori, H., Jenkins, S., Caruana, R., Wallach, H., & Wortman Vaughan, J. (2020). Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–14. https://doi.org/10.1145/3313831.3376219

Kumar, I. E., Venkatasubramanian, S., Scheidegger, C., & Friedler, S. (2020). Problems with Shapley-value-based explanations as feature importance measures. *Proceedings of the 37th International Conference on Machine Learning*, 5491–5500. Available from https://proceedings.mlr.press/v119/kumar20e.html

Larson, S. C. (1931). The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, *22*(1), 45. https://doi.org/10.1037/h0072400

Liao, H.-Y., Armstrong, P. I., & Rounds, J. (2008). Development and initial validation of public domain basic interest markers. *Journal of Vocational Behavior*, *73*(1), 159–183. https://doi.org/10.1016/j.jvb.2007.12.002

Littlefield, A. K., Cooke, J. T., Bagge, C. L., Glenn, C. R., Kleiman, E. M., Jacobucci, R., Millner, A. J., & Steinley, D. (2021). Machine learning to classify suicidal thoughts and behaviors: Implementation within the common data elements used by the military suicide research consortium. *Clinical Psychological Science*, *9*(3), 467–481. https://doi.org/10.1177/2167702620961067

Luijken, K., Groenwold, R. H., Van Calster, B., Steyerberg, E. W., & Smeden, M. van. (2019). Impact of predictor measurement heterogeneity across settings on the performance of prediction models: A measurement error perspective. *Statistics in Medicine*, *38*(18), 3444–3459. https://doi.org/10.1002/sim.8183

Lynam, A. L., Dennis, J. M., Owen, K. R., Oram, R. A., Jones, A. G., Shields, B. M., & Ferrat, L. A. (2020). Logistic regression has similar performance to optimised machine learning algorithms in a clinical setting: Application to the discrimination between type 1 and type 2 diabetes in young adults. *Diagnostic and Prognostic Research*, *4*, 1–10. https://doi.org/10.1186/s41512-020-00075-2

Miller, P. J., Lubke, G. H., McArtor, D. B., & Bergeman, C. (2016). Finding structure in data using multivariate tree boosting. *Psychological Methods*, *21*(4), 583. https://doi.org/10.1037/met0000087

Mosier, C. I. (1951). I. Problems and designs of cross-validation 1. *Educational and Psychological Measurement*, *11*(1), 5–11. https://doi.org/10.1177/001316445101100101

Nicodemus, K. K. (2011). On the stability and ranking of predictors from random forest variable importance measures. *Briefings in Bioinformatics*, *12*(4), 369–373. https://doi.org/10.1093/bib/bbr016

Nicodemus, K. K., Malley, J. D., Strobl, C., & Ziegler, A. (2010). The behaviour of random forest permutation-based variable importance measures under predictor correlation. *BMC Bioinformatics*, *11*(1), 1–13. https://doi.org/10.1186/1471-2105-11-110

Nusinovici, S., Tham, Y. C., Yan, M. Y. C., Ting, D. S. W., Li, J., Sabanayagam, C., Wong, T. Y., & Cheng, C.-Y. (2020). Logistic regression was as good as machine learning for

predicting major chronic diseases. *Journal of Clinical Epidemiology*, *122*, 56–69. https://doi.org/10.1016/j.jclinepi.2020.03.002

R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Rauthmann, J. F. (2020). A (more) behavioural science of personality in the age of multi–modal sensing, big data, machine learning, and artificial intelligence. In *European Journal of Personality* (Vol. 34, pp. 593–598). SAGE Publications Sage UK: London, England. https://doi.org/10.1002/per.2310

Ross, A. S., Hughes, M. C., & Doshi-Velez, F. (2017). Right for the right reasons: Training differentiable models by constraining their explanations. *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2662–2670. https://doi.org/10.48550/arXiv.1703.03717

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x

Seeboth, A., & Mõttus, R. (2018). Successful explanations start with accurate descriptions: Questionnaire items as personality markers for more accurate predictions. *European Journal of Personality*, *32*(3), 186–201. https://doi.org/10.1002/per.2147

Segal, M. R. (2004). Machine learning benchmarks and random forest regression. *eScholarship Repository, University of California*. http://repositories.cdlib.org/cbmb/bench_rf_regn

Stewart, R. D., Mõttus, R., Seeboth, A., Soto, C. J., & Johnson, W. (2021). The finer details? The predictability of life outcomes from big five domains, facets, and nuances. *Journal of Personality*. https://doi.org/10.1111/jopy.12660

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for random forests. *BMC Bioinformatics*, *9*(1), 1–11. https://doi.org/10.1186/1471-2105-9-307

Strobl, C., Boulesteix, A.-L., Zeileis, A., & Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, *8*(1), 1–21. https://doi.org/10.1186/1471-2105-8-25

Strobl, C., Malley, J., & Tutz, G. (2009). An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological Methods*, *14*(4), 323. https://doi.org/10.1037/a0016973

Tay, L., Woo, S. E., Hickman, L., & Saef, R. M. (2020). Psychometric and validity issues in machine learning approaches to personality assessment: A focus on social media text mining. *European Journal of Personality*, *34*(5), 826–844. https://doi.org/10.1002/per.2290

Waa, J. van der, Nieuwburg, E., Cremers, A., & Neerincx, M. (2021). Evaluating XAI: A comparison of rule-based and example-based explanations. *Artificial Intelligence*, *291*, 103404. https://doi.org/10.1016/j.artint.2020.103404

Yarkoni, T. (2020). Implicit realism impedes progress in psychology: Comment on fried (2020). *Psychological Inquiry*, *31*(4), 326–333. https://doi.org/10.1080/1047840X.2020.1853478

Yarkoni, T., & Westfall, J. (2017). Choosing prediction over explanation in psychology: Lessons from machine learning. *Perspectives on Psychological Science*, *12*(6), 1100–1122. https://doi.org/10.1177/1745691617693393

Appendix A

RIASEC Items

The following items were rated on a 1-5 scale of how much they would like to perform that task, with the labels 1=Dislike, 3=Neutral, 5=Enjoy:

R1 Test the quality of parts before shipment

R2 Lay brick or tile

R3 Work on an offshore oil-drilling rig

R4 Assemble electronic parts

R5 Operate a grinding machine in a factory

R6 Fix a broken faucet

R7 Assemble products in a factory

R8 Install flooring in houses

I1 Study the structure of the human body

I2 Study animal behavior

I3 Do research on plants or animals

I4 Develop a new medical treatment or procedure

I5 Conduct biological research

I6 Study whales and other types of marine life

I7 Work in a biology lab

I8 Make a map of the bottom of an ocean

A1 Conduct a musical choir

A2 Direct a play

A3 Design artwork for magazines

A4 Write a song

A5 Write books or plays

A6 Play a musical instrument

A7 Perform stunts for a movie or television show

A8 Design sets for plays

S1 Give career guidance to people

S2 Do volunteer work at a non-profit organization

S3 Help people who have problems with drugs or alcohol

S4 Teach an individual an exercise routine

S5 Help people with family-related problems

S6 Supervise the activities of children at a camp

S7 Teach children how to read

S8 Help elderly people with their daily activities

E1 Sell restaurant franchises to individuals

E2 Sell merchandise at a department store

E3 Manage the operations of a hotel

E4 Operate a beauty salon or barber shop

E5 Manage a department within a large company

E6 Manage a clothing store

E7 Sell houses

E8 Run a toy store

C1 Generate the monthly payroll checks for an office

C2 Inventory supplies using a hand-held computer

C3 Use a computer program to generate customer bills

C4 Maintain employee records

C5 Compute and record statistical and other numerical data

C6 Operate a calculator

C7 Handle customers' bank transactions

C8 Keep shipping and receiving records

Appendix B

Uni- and Bivariate Sample Descriptives

The total sample consisted of 55,593 participants. Mean age was 33.06 (SD = 11.68). With respect to gender, 67% of participants reported female, 32% reported male, 1% reported "other." With respect to marital status, 58% of participants reported being never married, 33% reported being currently married, 8% reported being previously married. With respect to type of area lived in when a child, 22% of participants reported rural, 37% reported suburban, 40% reported urban. With respect to language, 66% of participants reported that English is their native language, 34% reported that English is not their native language. With respect to race, 20% reported Asian, 1% reported Arab, 7% reported Black, 61% reported White, Native American, or Indigenous Australian (note that these three options were merged due to a coding mistake), and 1% did not report race.

Figure B1

Univariate distributions of the RIASEC subscale scores ($N = 55{,}593$)

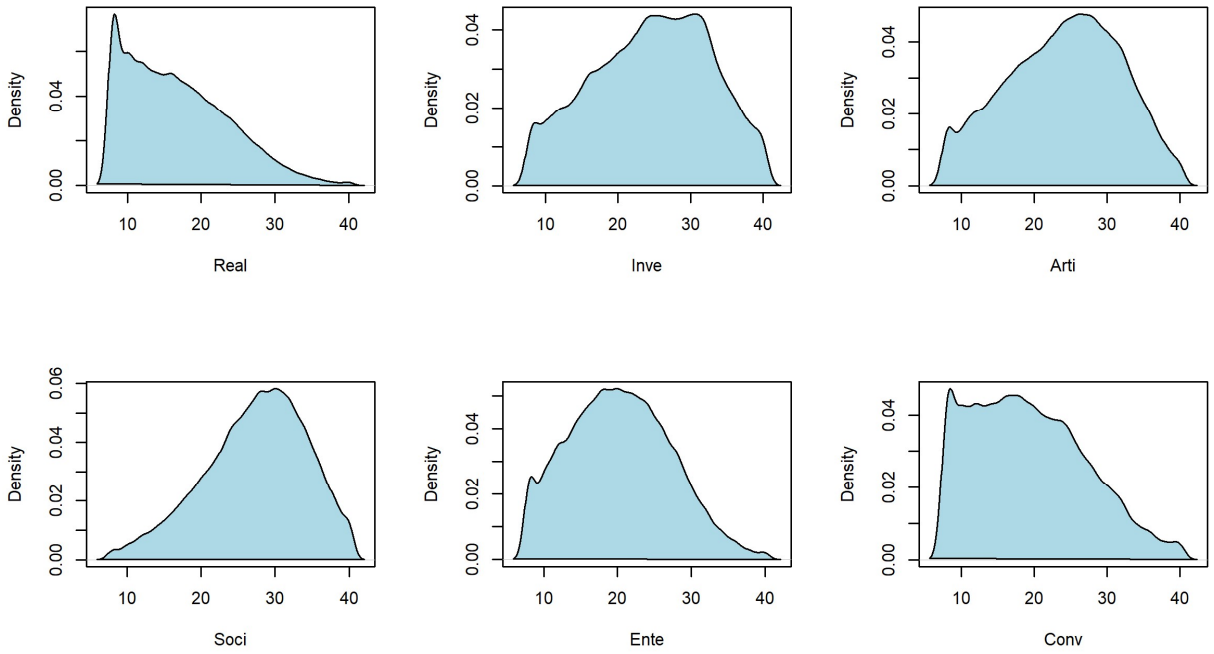


Table B1

Pearson correlations between RIASEC subscale scores ($N = 55{,}593$)

	Real	**Inve**	**Arti**	**Soci**	**Ente**	**Conv**
Real	1.000	0.332	0.182	0.049	0.305	0.460
Inve	0.332	1.000	0.329	0.141	0.016	0.065
Arti	0.182	0.329	1.000	0.290	0.253	-0.056
Soci	0.049	0.141	0.290	1.000	0.356	0.124
Ente	0.305	0.016	0.253	0.356	1.000	0.464
Conv	0.460	0.065	-0.056	0.124	0.464	1.000

Appendix C

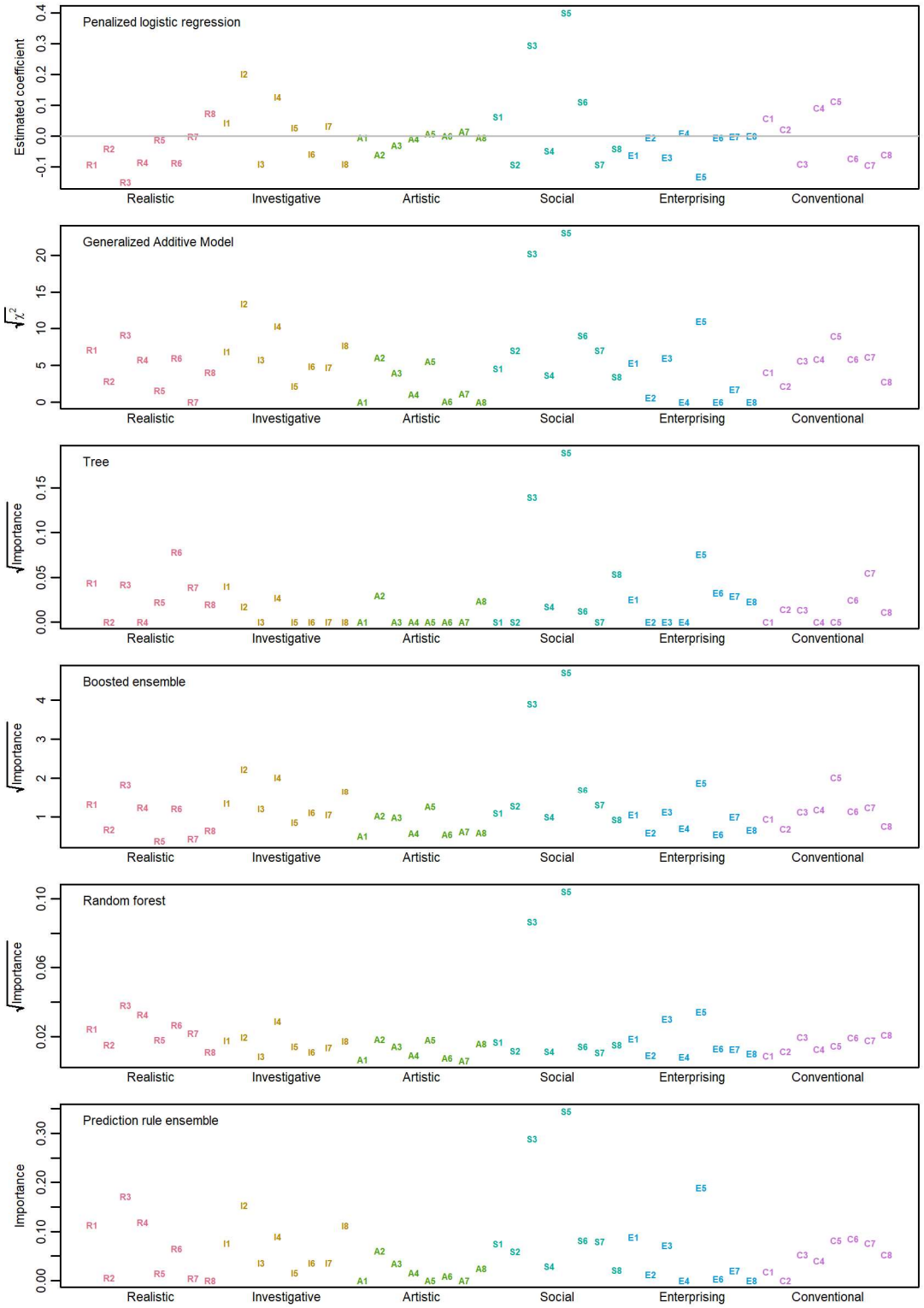
Variable contributions from the item-level predictive models

In Figure C1, the variable contributions are depicted for the models fitted using the RIASEC item scores as predictors. The plots show a similar, but more nuanced view, compared to the variable contributions in the models fitted using the subscale scores (Figure 3, main paper). Again, the Social preferences scale contains the strongest predictors of university major completed, followed by the Realistic subscale, then followed by the Enterprising, Investigative and Conventional preferences scales. The item-level variable contributions do provide a more nuanced view. For example, in the analyses using the item scores as predictors, the Investigative subscale does seem to contribute more strongly compared to the analyses based on the subscales.

All methods found items S5 (“I would like to help people with family-related problems”) and S3 (“I would like to help people who have problems with drugs or alcohol”) and to contribute most. For the penalized GLM, the GAM and the boosted ensemble, this was followed by I2 (“I would like to study animal behavior”). For the single tree, the third most important predictor was item R6 (“I would like to fix a broken faucet”). For the prediction rule ensemble and the random forest, the third and fourth most important predictors were items R3 (“I would like to work on an offshore oil-drilling rig”) and E5 (“I would like to manage a department within a large company”).

Figure C1

Variable contributions for each of the models fitted using RIASEC item scores as predictors



Appendix D
Replication scripts: Models fitted in main paper

Introduction

In this Appendix, we present scripts for replicating the results from the main paper. In the Method section, packages used are listed and data preparation is shown. In the Results section, we show how each of the models in the main document were fit, and how the figures and tables can be replicated. In sections Appendix B and C, we show how results, tables and figures from the appendices can be replicated. Finally, the last section provides the references.

Method

All analyses were performed in **R** (version 4.1.0, R Core Team, 2021). We fitted (penalized) logistic regression models using **R** package **glmnet** (version 4.1.3, Friedman et al., 2010); generalized additive models (GAMs) with smoothing splines using package **mgcv** (version 1.8.35, Wood, 2017); conditional inference trees using package **partykit** (version 1.2.15, Hothorn et al., 2006); gradient boosted tree ensembles using package **gbm** (version 2.1.8, Greenwell et al., 2020); random forests using package **ranger** (version 0.13.1, Wright & Ziegler, 2017); prediction rule ensembles using package **pre** (version 1.0.3, Fokkema, 2020); k nearest neighbours using package **class** (version 7.3.19, Venables & Ripley, 2002).

Appendix E shows our code and results for tuning the parameter settings using cross validation. All parameter settings employed here are based on those results, for most methods. We did not tune the parameters of the GAMs with smoothing splines, because we expected the defaults to work well out of the box. We simply employed the defaults of thin-plate regression splines and generalized cross validation method to fit the smoothing splines. This provides built-in regularization of the wigglyness, without the need for choosing an optimal value for penalty or tuning parameters. For smaller samples, restricted maximum likelihood (REML) would be preferred. For the analyses using the items as predictors, we set the number of basis functions of the thin-plate regression splines to four, instead of the default of eight, because the item responses have only five possible values; increasing the number of basis functions would yield an unidentified model. Furthermore, because of the larger number of predictors, and the benefit of penalization observed in the logistic regression, we also applied a penalty to the linear part of each smoothing spline function, so that some items can be completely eliminated from the model.

Data preparation

```
data <- read.delim("data.csv", header = TRUE)
```

```r
## Items should be scored 1-5, 0s are missings
data[ , 1:48][sapply(data[ , 1:48], function(x) x == 0)] <- NA
data <- data[complete.cases(data[ , 1:48]), ]

## Select only university students
data <- data[data$education >= 3, ]
# table(data$major)
psych_ids <- rowSums(sapply(c("psych", "psyhcology", "psycotherapy", "couns",
                              "behavior", "behaviour", "neuro"),
                            function(x) grepl(x, data$major, ignore.case = TRUE)))
anim_ids <- grepl("anim", data$major, ignore.case = TRUE) ## exclude animal psych
data$major <- factor(ifelse(psych_ids > 0, "psychology", "other"))
data$major[anim_ids > 0 & psych_ids > 0] <- "other"

set.seed(42)
test_ids <- sample(1:nrow(data), ceiling(nrow(data)/4))
train_ids <- which(!1:nrow(data) %in% test_ids)
train_y <- as.numeric(data$major)[train_ids] - 1
test_y <- as.numeric(data$major)[test_ids] - 1

data$Real <- rowSums(data[ , paste0("R", 1:8)])
data$Inve <- rowSums(data[ , paste0("I", 1:8)])
data$Arti <- rowSums(data[ , paste0("A", 1:8)])
data$Soci <- rowSums(data[ , paste0("S", 1:8)])
data$Ente <- rowSums(data[ , paste0("E", 1:8)])
data$Conv <- rowSums(data[ , paste0("C", 1:8)])
```

Sample size:

```r
nrow(data)
```

```
## [1] 55593
```

Proportions of observations choosing psychology (not) as a major:

```r
prop.table(table(data$major)) ## total dataset
```

```
##
##       other psychology
##   0.8057849   0.1942151
```

```r
prop.table(table(data$major[train_ids])) ## train dataset
```

```
##
##       other psychology
##   0.8053677   0.1946323
```

```
prop.table(table(data$major[test_ids])) ## test dataset
```

```
##
##      other psychology
##  0.8070365  0.1929635
```

Results

```
varnames_i <- paste0(rep(c("R", "I", "A", "S", "E", "C"), each = 8), 1:8)
varnames_s <- c("R", "I", "A", "S", "E", "C")
```

(Penalized) Logistic Regression

```
glmod_s <- glm(major ~ Real + Inve + Arti + Soci + Ente + Conv,
               data = data[train_ids , ], family = "binomial")
#summary(glmod_s)
glm_preds_train_s <- predict(glmod_s, newdata = data[train_ids, ], type = "response")
glm_preds_test_s <- predict(glmod_s, newdata = data[test_ids, ], type = "response")
```

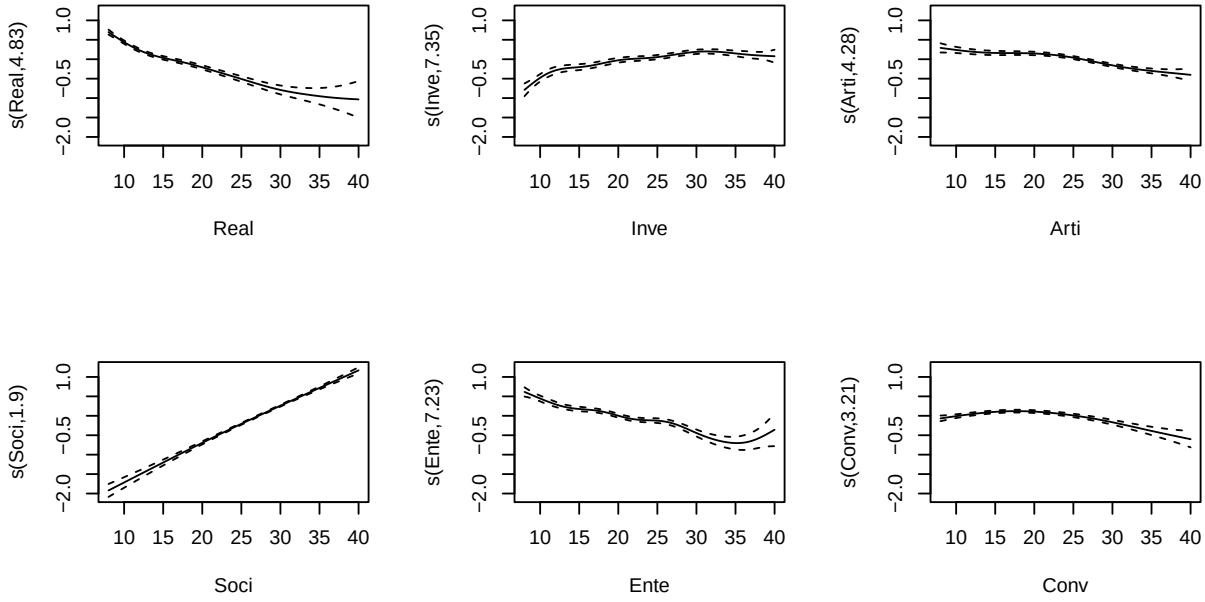
```
library("glmnet")
X <- as.matrix(data[train_ids, varnames_i])
set.seed(42)
glmod_i <- glmnet(X, train_y, family = "binomial", alpha = 1, lambda = 0.0003568404)
glm_preds_train_i <- predict(glmod_i, newx = X, type = "response")
glm_preds_test_i <- predict(glmod_i, newx = as.matrix(data[test_ids, varnames_i]),
                            type = "response")
```

Generalized Additive Model

```
library("mgcv")
gamod_s <- gam(major ~ s(Real) + s(Inve) + s(Arti) + s(Soci) + s(Ente) + s(Conv),
               data = data[train_ids , ], family = "binomial")
```

Figure 1

```
par(mfrow = c(2, 3))
plot(gamod_s)
```




```
gam_preds_train_s <- predict(gamod_s, newdata = data[train_ids, ], type = "response")
gam_preds_test_s <- predict(gamod_s, newdata = data[test_ids, ], type = "response")
```

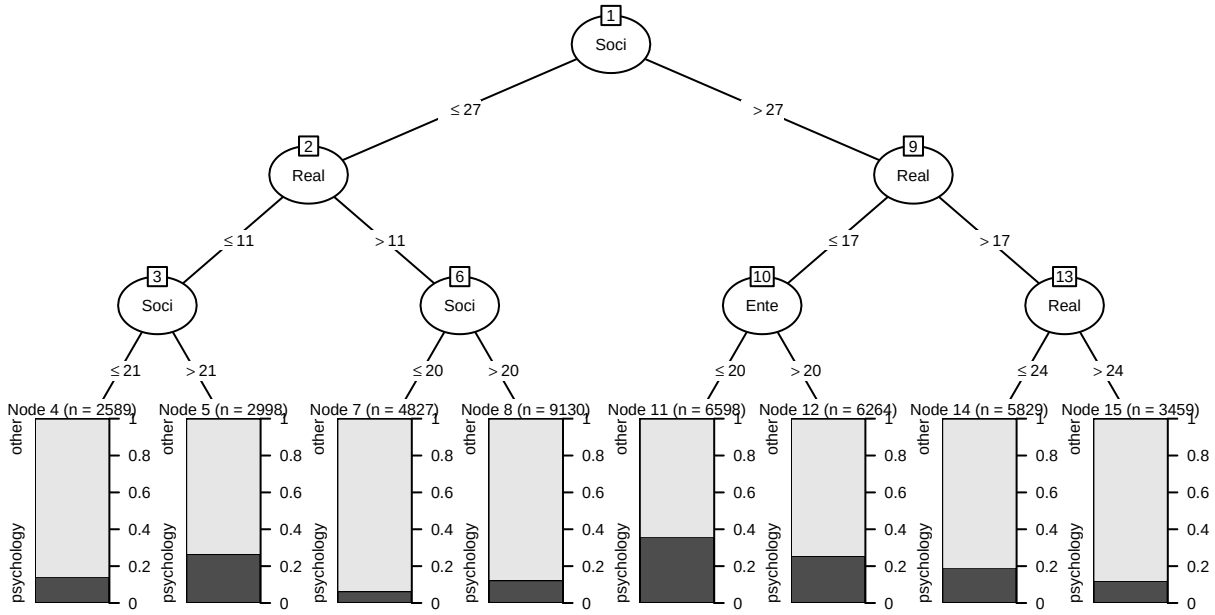
```
gamod_i <- gam(gam_form, data = data[train_ids , ], family = "binomial",
               select = TRUE)
```

```
gam_preds_train_i <- predict(gamod_i, newdata = data[train_ids, ], type = "response")
gam_preds_test_i <- predict(gamod_i, newdata = data[test_ids, ], type = "response")
```

Decision Tree

Figure 2

```
library("partykit")
ct_s <- ctree(major ~ Real + Inve + Arti + Soci + Ente + Conv, data = data[train_ids , ],
              maxdepth = 7)
ct3 <- ctree(major ~ Real + Inve + Arti + Soci + Ente + Conv, data = data[train_ids , ],
             maxdepth = 3)
plot(ct3, gp = gpar(cex = .5), ip_args = list(pval = FALSE))
```




```
ct_preds_train_s <- predict(ct_s, type = "prob")[ , "psychology"]
ct_preds_test_s <- predict(ct_s, newdata = data[test_ids , ], type = "prob")[ , "psychology"]
```

```
ct_form <- formula(paste("major ~", paste(varnames_i, collapse = "+")))
ct <- ctree(ct_form, data = data[train_ids , ], maxdepth = 9)
ct_preds_train_i <- predict(ct, type = "prob")[ , "psychology"]
ct_preds_test_i <- predict(ct, newdata = data[test_ids , ], type = "prob")[ , "psychology"]
```

Gradient boosted tree ensemble

```
library("gbm")
set.seed(42)
gb_s <- gbm(I(as.numeric(major)-1) ~ Real + Inve + Arti + Soci + Ente + Conv,
            n.trees = 1100, interaction.depth = 3L, shrinkage = 0.01,
            data = data[train_ids , ])
```

```
gb_preds_train_s <- predict(gb_s, newdata = data[train_ids, ], type = "response")
gb_preds_test_s <- predict(gb_s, newdata = data[test_ids, ], type = "response")
```

```
library("gbm")
set.seed(42)
gbm_form <- formula(paste("I(as.numeric(major)-1) ~ ",
                          paste(paste0(rep(c("R", "I", "A", "S", "E", "C"),
                                           each = 8), 1:8), collapse = "+")))
gb_i <- gbm(gbm_form, n.trees = 3500, interaction.depth = 5L, shrinkage = 0.01,
            data = data[train_ids , ])
sum_i <- summary(gb_i, plotit = FALSE, method = permutation.test.gbm)
```

```
gb_preds_train_i <- predict(gb_i, newdata = data[train_ids, ], type = "response")
gb_preds_test_i <- predict(gb_i, newdata = data[test_ids, ], type = "response")
```

Random Forest

```
library("ranger")
set.seed(42)
rf_s <- ranger(major ~ Real + Inve + Arti + Soci + Ente + Conv, data = data[train_ids , ],
                   probability = TRUE, mtry = 3L, min.node.size = 500,
             importance = "permutation")
```

```
rf_preds_train_s <- predict(rf_s, data = data[train_ids, ])$predictions[ , "psychology"]
rf_preds_test_s <- predict(rf_s, data = data[test_ids, ])$predictions[ , "psychology"]
```

```
set.seed(42)
varnames <- paste0(rep(c("R", "I", "A", "S", "E", "C"), each = 8), 1:8)
rf_form <- formula(paste("major ~", paste(varnames, collapse = "+")))
rf_i <- ranger(rf_form, data = data[train_ids , ], probability = TRUE,
             mtry = 10L, min.node.size = 500, importance = "permutation")
```

```
rf_preds_train_i <- predict(rf_i, data = data[train_ids, ])$predictions[ , "psychology"]
rf_preds_test_i <- predict(rf_i, data = data[test_ids, ])$predictions[ , "psychology"]
```

Prediction Rule Ensembling

Table 1

```
pr_preds_train_s <- predict(pr_s, type = "response")
pr_preds_test_s <- predict(pr_s, newdata = data[test_ids , ], type = "response")
imps <- pre::importance(pr_s, plot=FALSE)
varimps_pre <- imps$varimps
imps <- imps$baseimps[1:6, c("description", "coefficient")]
colnames(imps) <- c("Description", "Coefficient")
imps$Coefficient <- round(imps$Coefficient, digits = 3)
kable(imps, row.names = FALSE, align = c("l", "c"))
```

Description	Coefficient
Soci > 27 & Ente <= 31 & Conv <= 30	0.182
Soci > 23 & Ente <= 29 & Real <= 24	0.181
Real > 10 & Soci <= 35	-0.175
Real <= 22 & Soci > 19 & Inve > 18	0.138
Inve > 10 & Real <= 13	0.120

Description	Coefficient
Conv <= 23 & Arti <= 29 & Soci > 21	0.112

```
pr_preds_train_i <- predict(pr_i, type = "response")
pr_preds_test_i <- predict(pr_i, newdata = data[test_ids , ], type = "response")
```

k Nearest Neighbours

```
library("class")

## Model for training predictions
knn_mod <- knn(train = data[train_ids , varnames_s],
               test = data[train_ids , varnames_s],
               cl = as.factor(data[train_ids, "major"]),
               k = 300, use.all = TRUE, prob = TRUE)
## Need to obtain predicted probability for second class
knn_preds_train_s <- ifelse(
  knn_mod == "psychology", attr(knn_mod, "prob"), 1 - attr(knn_mod, "prob"))
```

```
## Model for testing predictions
knn_mod <- knn(train = data[train_ids , varnames_s],
               test = data[test_ids , varnames_s],
               cl = as.factor(data[train_ids, "major"]),
               k = 300, use.all = TRUE, prob = TRUE)
knn_preds_test_s <- ifelse(
  knn_mod == "psychology", attr(knn_mod, "prob"), 1 - attr(knn_mod, "prob"))
```

```
## Model for training predictions
knn_mod <- knn(train = data[train_ids , varnames_i],
               test = data[train_ids , varnames_i],
               cl = as.factor(data[train_ids, "major"]),
               k = 100, use.all = TRUE, prob = TRUE)
## Need to obtain predicted probability for second class
knn_preds_train_i <- ifelse(
  knn_mod == "psychology", attr(knn_mod, "prob"), 1 - attr(knn_mod, "prob"))
```

```
## Model for testing predictions
knn_mod <- knn(train = data[train_ids , varnames_i],
               test = data[test_ids , varnames_i],
               cl = as.factor(data[train_ids, "major"]),
               k = 100, use.all = TRUE, prob = TRUE)
knn_preds_test <- ifelse(
  knn_mod == "psychology", attr(knn_mod, "prob"), 1 - attr(knn_mod, "prob"))
```

Model comparisons

Figure 3

```
par(mfrow = c(2, 3))
par(mar = c(1, 4, 2, 1), mgp = c(1.5, .5, 0), tck = -0.05)
library("colorspace")

## Logistic regression
plot(coef(glmod_s)[-1], xaxt = "n", ylab = "Estimated coefficient",
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ",
     main = "Logistic Regression", cex.main = .7)
text(coef(glmod_s)[-1], labels = varnames_s, cex = 1,
     col = rep(qualitative_hcl(6)), font = 2)
abline(0, 0, col = "grey")

## Generalized additive model
sum <- summary(gamod_s)
plot(sqrt(sum$chi.sq), xaxt = "n", ylab = expression(sqrt(chi^2)),
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ",
     main = "Generalized Additive Model", cex.main = .7)
text(sqrt(sum$chi.sq), labels = varnames_s, cex = 1,
     col = rep(qualitative_hcl(6)), font = 2)

## Conditional inference tree
ct6 <- cforest(major ~ Real + Inve + Arti + Soci + Ente + Conv,
               data = data[train_ids , ], ntree = 1L, mtry = 6,
               perturb = list(replace = FALSE, fraction = 1L),
               control = ctree_control(maxdepth = 6))
imps <- varimp(gettree(ct6), risk = "loglik")
imps <- imps[c("Real", "Inve", "Arti", "Soci", "Ente", "Conv")]
plot(sqrt(imps), xaxt = "n", ylab = expression(sqrt(Importance)),
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ",
     main = "Tree", cex.main = .7)
text(sqrt(imps), labels = varnames_s, cex = 1,
     col = rep(qualitative_hcl(6)), font = 2)

## Gradient boosted ensemble
sum <- summary(gb_s, plotit = FALSE, method = permutation.test.gbm)
imps <- sum$rel.inf
names(imps) <- sum$var
imps <- imps[c("Real", "Inve", "Arti", "Soci", "Ente", "Conv")]
plot(sqrt(imps), xaxt = "n", ylab = expression(sqrt(Importance)),
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ",
```

```
     main = "Gradient Boosted Ensemble", cex.main = .7)
text(sqrt(imps), labels = varnames_s, cex = 1,
     col = rep(qualitative_hcl(6)), font = 2)

## Random forest
library("ranger")
load(file = "RF_subscales.Rda")
imps <- ranger::importance(rf_s)
plot(sqrt(imps), xaxt = "n", ylab = expression(sqrt(Importance)),
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ",
     main = "Random Forest", cex.main = .7)
text(sqrt(imps), labels = varnames_s, cex = 1,
     col = rep(qualitative_hcl(6)), font = 2)

## Prediction rule ensemble
imps <- varimps_pre$imp
names(imps) <- varimps_pre$varname
imps <- imps[c("Real", "Inve", "Arti", "Soci", "Ente", "Conv")]
plot(imps, xaxt = "n", ylab = "Importance",
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ",
     cex.main = .7, main = "Prediction Rule Ensemble")
text(imps, labels = varnames_s, cex = 1,
     col = rep(qualitative_hcl(6)), font = 2)
```

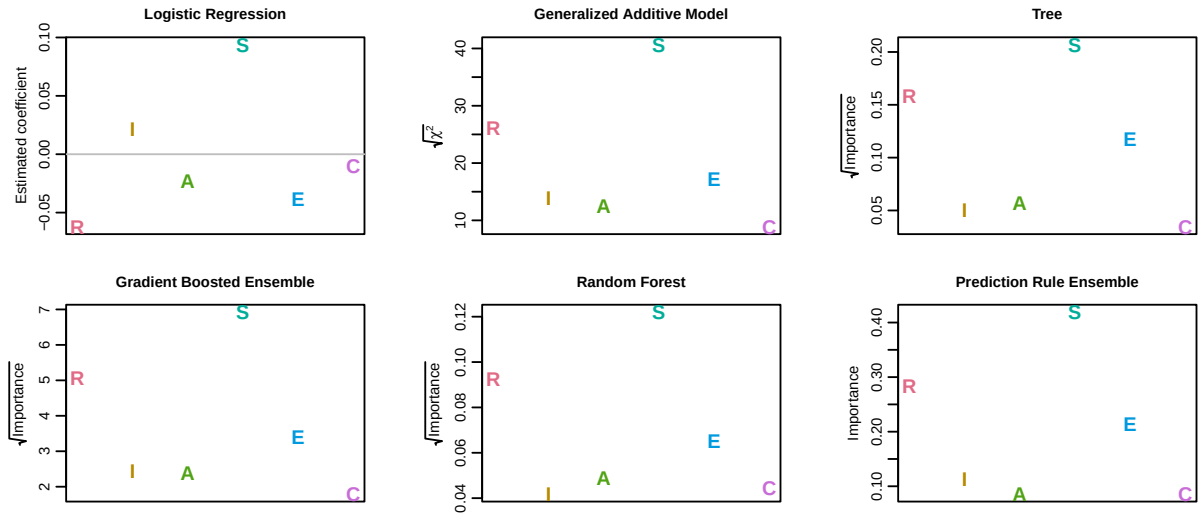
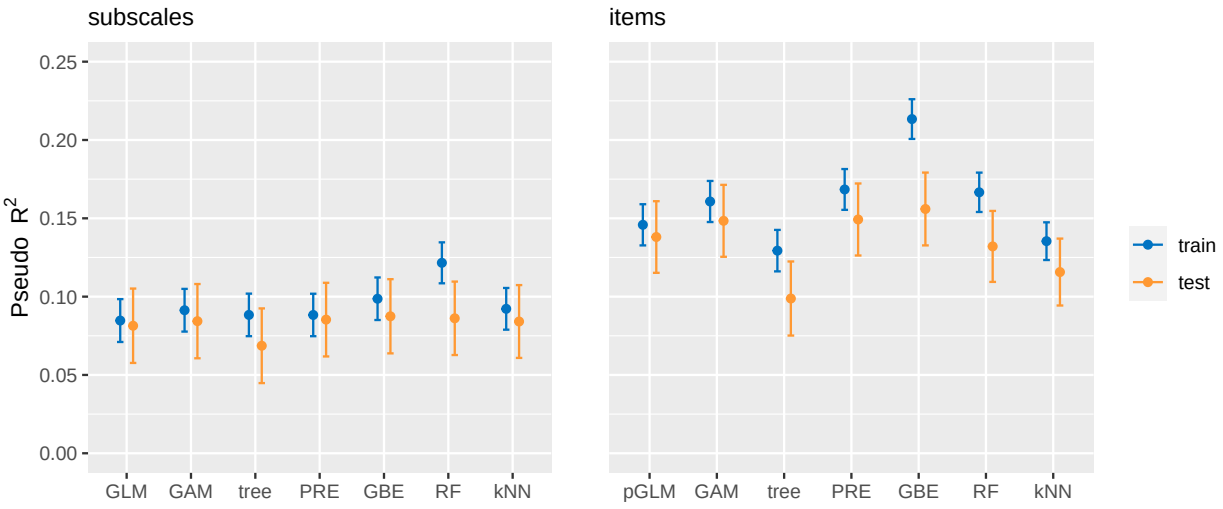


Figure 5

```
## Gather results for prediction with subscales
preds_train <- data.frame(bm_preds_train = rep(mean(train_y), times = length(train_ids)),
                          glm_preds_train_s, gam_preds_train_s,
```

```
                          ct_preds_train_s, pr_preds_train_s, gb_preds_train_s,
                          rf_preds_train_s, knn_preds_train_s)
preds_test <- data.frame(bm_preds_test = rep(mean(train_y), times = length(test_ids)),
                         glm_preds_test_s, gam_preds_test_s,
                         ct_preds_test_s, pr_preds_test_s, gb_preds_test_s,
                         rf_preds_test_s, knn_preds_test_s)
results <- data.frame(
  Brier_train = sapply(preds_train, function(x) mean((train_y - x)^2)),
  Brier_test = sapply(preds_test, function(x) mean((test_y - x)^2)))
results$R2_train <- 1 - results$Brier_train / results$Brier_train[1]
results$R2_test <- 1 - results$Brier_test / results$Brier_test[1]

tmp <- data.frame(
  R2 = c(results$R2_train[-1], results$R2_test[-1]),
  SE = c(sapply((preds_train[,-1] - train_y)^2,
                function(x) sd(x)/(sqrt(nrow(preds_train))*results$Brier_train[1])),
         sapply((preds_test[,-1] - test_y)^2,
                function(x) sd(x)/(sqrt(nrow(preds_test))*results$Brier_test[1]))),
  data = rep(c("train", "test"), each = nrow(results)-1),
  method = rep(c("GLM", "GAM", "tree", "PRE", "GBE", "RF", "kNN"), times = 2))
tmp$data <- factor(tmp$data, levels = c("train", "test"))
tmp$method <- factor(tmp$method, levels = c("GLM", "GAM", "tree", "PRE", "GBE", "RF", "kNN"))

## Plot results
library("ggplot2")
p_subscales <- ggplot(tmp, aes(x = method, y = R2)) +
  geom_point(aes(color = data, fill = data),
             stat = "identity", position = position_dodge(0.4)) +
  ggtitle("subscales") + xlab("") + ylab(expression("Pseudo "~R^2)) +
  theme(legend.position = "none", plot.title = element_text(size = 11)) +
  scale_color_manual(values = c("#0073C2FF", "#ff9933")) + ylim(0, 0.25) +
  scale_fill_manual(values = c("#0073C2FF", "#ff9933")) +
  geom_errorbar(aes(color=data, ymin = R2-1.96*SE, ymax = R2+1.96*SE),
                width = .2, position = position_dodge(0.4))

## Gather results for prediction with items
preds_train <- data.frame(bm_preds_train = rep(mean(train_y), times = length(train_ids)),
                          glm_preds_train_i, gam_preds_train_i,
                          ct_preds_train_i, pr_preds_train_i, gb_preds_train_i,
                          rf_preds_train_i, knn_preds_train_i)
preds_test <- data.frame(bm_preds_test = rep(mean(train_y), times = length(test_ids)),
                          glm_preds_test_i, gam_preds_test_i,
                          ct_preds_test_i, pr_preds_test_i, gb_preds_test_i,
```

```
                        rf_preds_test_i, knn_preds_test_i)
results <- data.frame(
  Brier_train = sapply(preds_train, function(x) mean((train_y - x)^2)),
  Brier_test = sapply(preds_test, function(x) mean((test_y - x)^2)))
results$R2_train <- 1 - results$Brier_train / results$Brier_train[1]
results$R2_test <- 1 - results$Brier_test / results$Brier_test[1]

tmp <- data.frame(
  R2 = c(results$R2_train[-1], results$R2_test[-1]),
  SE = c(sapply((preds_train[,-1] - train_y)^2,
                function(x) sd(x)/(sqrt(nrow(preds_train))*results$Brier_train[1])),
         sapply((preds_test[,-1] - test_y)^2,
                function(x) sd(x)/(sqrt(nrow(preds_test))*results$Brier_test[1]))),
  data = rep(c("train", "test"), each = nrow(results)-1),
  method = rep(c("pGLM", "GAM", "tree", "PRE", "GBE", "RF", "kNN"), times = 2))
tmp$data <- factor(tmp$data, levels = c("train", "test"))
tmp$method <- factor(tmp$method, levels = c("pGLM", "GAM", "tree", "PRE", "GBE", "RF", "kNN"))

## Plot results
p_items <- ggplot(tmp, aes(x = method, y = R2)) +
  geom_point(aes(color = data, fill = data),
    stat = "identity", position = position_dodge(0.4)) + xlab("") + ylab(" ") +
  theme(axis.text.y=element_blank(), axis.ticks.y = element_blank(),
        plot.title = element_text(size = 11), legend.title = element_blank()) +
  ggtitle("items") + scale_color_manual(values = c("#0073C2FF", "#ff9933")) +
  scale_fill_manual(values = c("#0073C2FF", "#ff9933")) + ylim(0, .25) +
  geom_errorbar(aes(color=data, ymin = R2-1.96*SE, ymax = R2+1.96*SE),
                width = .2, position = position_dodge(0.4))

library("gridExtra")
grid.arrange(p_subscales, p_items, ncol = 2, widths = c(6.75, 8))
```



Appendix B

```
data$age[data$age > 103] <- NA ## remove impossible ages
prop.table(table(data$gender)) ## 0=missing, 1=Male, 2=Female, 3=Other
```

```
##
##           0           1           2           3
## 0.001151224 0.320382062 0.673160290 0.005306423
```

```
prop.table(table(data$married)) ## 0=missing, 1,2,3=Never, currently, previously married
```

```
##
##           0           1           2           3
## 0.005648193 0.582717249 0.328242764 0.083391794
```

```
prop.table(table(data$urban)) ## 0=missing, 1=Rural, 2=Suburban, 3=Urban
```

```
##
##           0           1           2           3
## 0.007087223 0.217419459 0.372474952 0.403018366
```

```
prop.table(table(data$race)) ## 0=missing, 1=Asian, 2=Arab, 3=Black,
```

```
##
##           0           1           2           3           4           5
## 0.01404853 0.19833432 0.01329304 0.07436188 0.60887162 0.09109060
```

```
## 4=Indigenous Australian/Native American/White, 5=Other
## (A coding mistake resulted in aggregating options in category 4)
prop.table(table(data$engnat)) ## 0=missing, 1=Yes, 2=No
```

```
##
```

```
##           0           1           2
## 0.002248485 0.659813286 0.337938230
```

Figure B1

```
par(mfrow = c(2, 3))
for (i in c("Real", "Inve", "Arti", "Soci" , "Ente", "Conv")) {
  dens <- density(data[ , i])
  plot(dens, main = "", xlab = i)
  polygon(dens, col="lightblue", border="black")
}
```

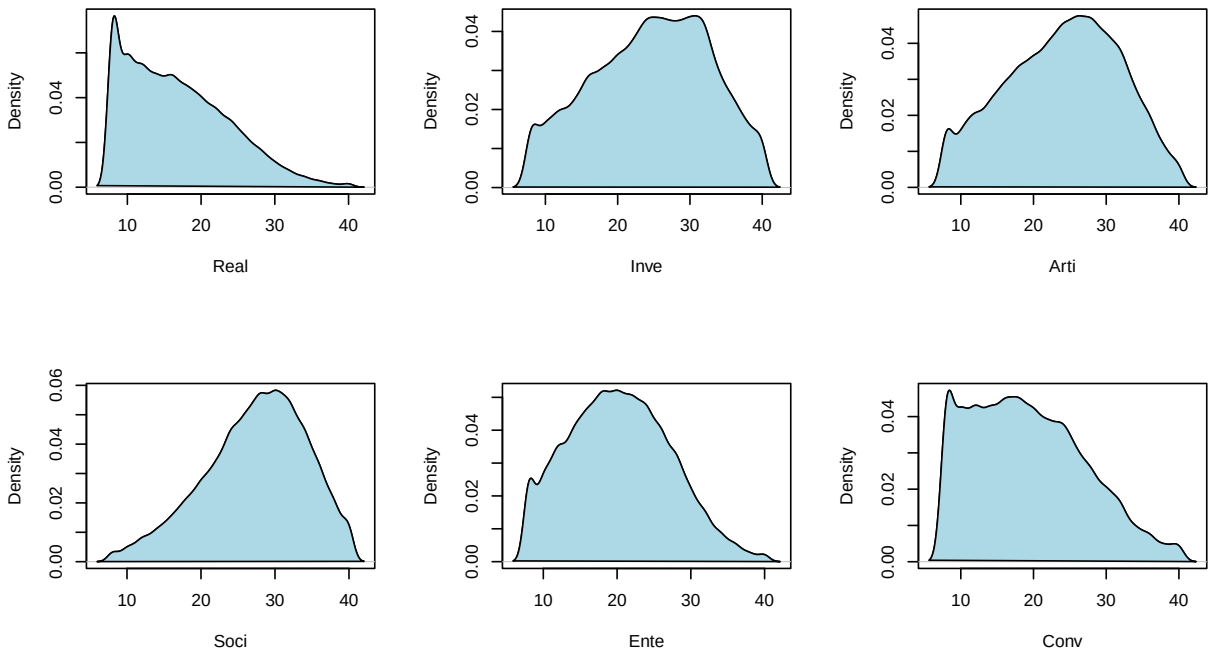


Table B1

```
tab <- round(cor(data[ , c("Real", "Inve", "Arti", "Soci" , "Ente", "Conv")]),
             digits = 3L)
kable(tab)
```

	Real	Inve	Arti	Soci	Ente	Conv
Real	1.000	0.332	0.182	0.049	0.305	0.460
Inve	0.332	1.000	0.329	0.141	0.016	0.065
Arti	0.182	0.329	1.000	0.290	0.253	-0.056
Soci	0.049	0.141	0.290	1.000	0.356	0.124
Ente	0.305	0.016	0.253	0.356	1.000	0.464

	Real	Inve	Arti	Soci	Ente	Conv
Conv	0.460	0.065	-0.056	0.124	0.464	1.000

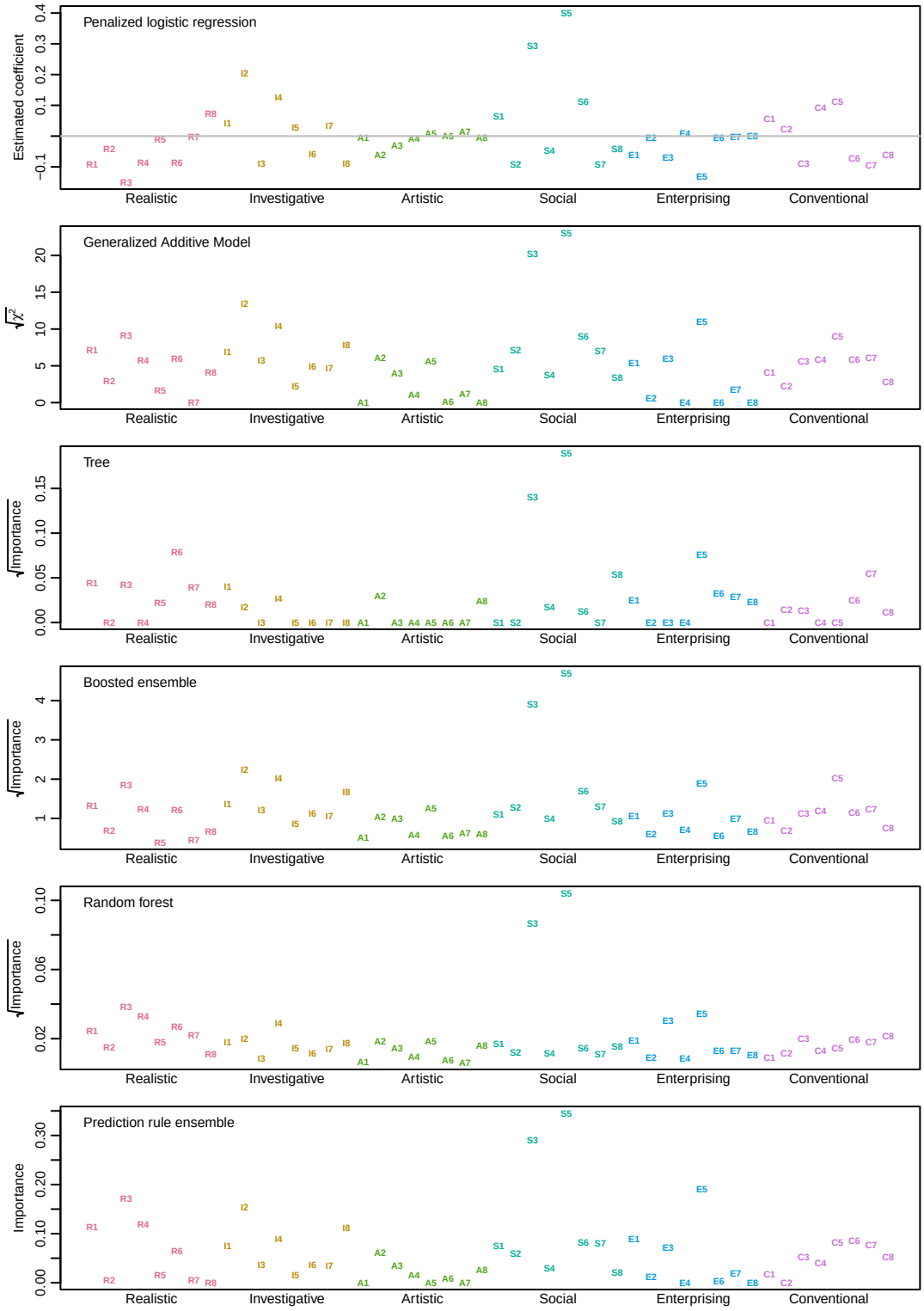
Appendix C

Figure C1

```
par(mar = c(1.5, 4, 0.2, 2), mgp = c(1.5, .5, 0), tck = -0.05)
par(mfrow = c(6, 1))

library("glmnet")
plot(coef(glmod_i)[-1], xaxt = "n", ylab = "Estimated coefficient",
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ",
     main = " ", cex.main = .7)
text(coef(glmod_i)[-1], labels = varnames_i, cex = .5,
     col = rep(qualitative_hcl(6), each = 8), font = 2)
abline(0, 0, col = "grey")
legend("topleft", legend = "Penalized logistic regression", cex = .7, bty = "n")
axis(1, 4.5 + c(0:5)*8, tick = FALSE, padj = -1.5,
     labels = c("Realistic", "Investigative", "Artistic", "Social" ,
                              "Enterprising", "Conventional"),
     cex.axis = .7)

library("mgcv")
load(file = "GAM_items.Rda")
sum <- summary(gamod_i)
plot(sqrt(sum$chi.sq), xaxt = "n", ylab = expression(sqrt(chi^2)),
     main = " ", cex.main = .7,
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ")
text(sqrt(sum$chi.sq), labels = varnames_i, cex = .5,
     col = rep(qualitative_hcl(6), each = 8), font = 2)
legend("topleft", legend = "Generalized Additive Model", cex = .7, bty = "n")
axis(1, 4.5 + c(0:5)*8, tick = FALSE, padj = -1.5,
     labels = c("Realistic", "Investigative", "Artistic", "Social" ,
                              "Enterprising", "Conventional"),
     cex.axis = .7)

ct6 <- cforest(ct_form,
               data = data[train_ids , ], ntree = 1L, mtry = 6,
               perturb = list(replace = FALSE, fraction = 1L),
               control = ctree_control(maxdepth = 6))
imps <- varimp(gettree(ct6), risk = "loglik")
```

```
imp_names <- names(imps)
imps <- c(imps, rep(0, times = 48 - length(imps)))
names(imps) <- c(imp_names, varnames_i[!varnames_i %in% imp_names])
plot(sqrt(imps[varnames_i]), xaxt = "n", ylab = expression(sqrt(Importance)),
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ",
     main = " ", cex.main = .7)
text(sqrt(imps[varnames_i]), labels = varnames_i, cex = .5,
     col = rep(qualitative_hcl(6), each = 8), font = 2)
legend("topleft", legend = "Tree", cex = .7, bty = "n")
axis(1, 4.5 + c(0:5)*8, tick = FALSE, padj = -1.5,
     labels = c("Realistic", "Investigative", "Artistic", "Social" ,
                              "Enterprising", "Conventional"),
     cex.axis = .7)

load(file = "gb_i_summary.Rda")
imps <- sum_i[match(varnames_i, sum_i$var), ]
plot(sqrt(imps$rel.inf), xaxt = "n", main = " ",
     ylab = expression(sqrt(Importance)), cex.main = .7,
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ")
text(sqrt(imps$rel.inf), labels = imps$var, cex = .5,
     col = rep(qualitative_hcl(6), each = 8), font = 2)
legend("topleft", legend = "Boosted ensemble", cex = .7, bty = "n")
axis(1, 4.5 + c(0:5)*8, tick = FALSE, padj = -1.5,
     labels = c("Realistic", "Investigative", "Artistic", "Social" ,
                              "Enterprising", "Conventional"),
     cex.axis = .7)

imps <- ranger::importance(rf_i)
plot(sqrt(imps), xaxt = "n", main = " ",
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ",
     ylab = expression(sqrt(Importance)), cex.main = .7)
text(sqrt(imps), labels = names(imps), cex = .5,
     col = rep(qualitative_hcl(6), each = 8), font = 2)
legend("topleft", legend = "Random forest", cex = .7, bty = "n")
axis(1, 4.5 + c(0:5)*8, tick = FALSE, padj = -1.5,
     labels = c("Realistic", "Investigative", "Artistic", "Social" ,
                              "Enterprising", "Conventional"),
     cex.axis = .7)

imps <- pre::importance(pr_i, cex.axis = .7, plot = FALSE)$varimps
zero_vars <- varnames_i[!varnames_i %in% imps[ , 1]]
imps <- rbind(imps, data.frame(varname = zero_vars,
                               imp = rep(0, times = length(zero_vars))))
```

```
imps <- imps[match(varnames_i, imps$varname), ]
plot(imps$imp, xaxt = "n", main = " ",
     ylab = "Importance",
     col = "white", cex.lab = .7, cex.axis = .7, xlab = " ", cex.main = .7)
text(imps$imp, labels = imps$varname, cex = .5,
     col = rep(qualitative_hcl(6), each = 8), font = 2)
legend("topleft", legend = "Prediction rule ensemble", cex = .7, bty = "n")
axis(1, 4.5 + c(0:5)*8, tick = FALSE, padj = -1.5,
     labels = c("Realistic", "Investigative", "Artistic", "Social" ,
                              "Enterprising", "Conventional"),
     cex.axis = .7)
```



References

Fokkema, M. (2020). Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software*,

92(1), 1–30.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1.

Greenwell, B., Boehmke, B., Cunningham, J., & Developers, G. (2020). *Gbm: Generalized boosted regression models*. https://CRAN.R-project.org/package=gbm

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, *15*(3), 651–674. https://doi.org/10.1198/106186006X133933

R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with S* (4th Edition). Springer. https://www.stats.ox.ac.uk/pub/MASS4/

Wood, S. N. (2017). *Generalized additive models: An introduction with R*. CRC press.

Wright, M. N., & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, *77*(1), 1–17. https://doi.org/10.18637/jss.v077.i01

Appendix E
Replication scripts: Parameter tuning

Introduction

All analyses were performed in **R** (version 4.1.0, R Core Team, 2021). We fitted (penalized) logistic regression models using **R** package **glmnet** (version 4.1.3, Friedman et al., 2010); generalized additive models (GAMs) with smoothing splines using package **mgcv** (version 1.8.35, Wood, 2017); conditional inference trees using package **partykit** (version 1.2.15, Hothorn et al., 2006); gradient boosted tree ensembles using package **gbm** (version 2.1.8, Greenwell et al., 2020); random forests using package **ranger** (version 0.13.1, Wright & Ziegler, 2017); prediction rule ensembles using package **pre** (version 1.0.3, Fokkema, 2020); k nearest neighbours using package **class** (version 7.3.19, Venables & Ripley, 2002).

We tuned the model-fitting parameters for all models using resampling and cross validation (CV) on the training data. For tuning the parameters of random forests, boosted tree ensembles and prediction rule ensembles, we used package **caret** (version 6.0.89, Kuhn, 2021). For tuning the parameters of conditional inference trees and k nearest neighbours, we wrote custom code; we tuned the penalized regression models using function `cv.glmnet` from package **glmnet**. We did not tune the parameters of the GAMs with smoothing splines, because we expected the defaults to work well out of the box.

The remainder of this document is structured as follows: The next section (Data preparation) provides the code used for data preparation. In the subsequent section (Cross validation of parameter settings), we provide code and output of the cross validation of model parameters. In the final two sections we provide version information about R and all packages used, and list the references.

Data preparation

The data can be downloaded as a .csv file (contained in a .zip file) from https://openpsychometrics.org/_rawdata/, or more specifically: http://openpsychometrics.org/_rawdata/RIASEC_data12Dec2018.zip.

```
data <- read.delim("data.csv", header = TRUE)
```

Items are scored 1-5, thus 0s are assumed to be missing values:

```
data[ , 1:48][sapply(data[ , 1:48], function(x) x == 0)] <- NA
data <- data[complete.cases(data[ , 1:48]), ]
```

We select participants who completed a university degree only:

```
data <- data[data$education >= 3, ]
```

The variable `major` contains the answer to the question: “If you attended a university, what was your major

(e.g. psychology, English, civil engineering)?" We code it as a binary factor, indicating whether the respondent did take psychology as a major, or not. The variable contains several typos, which we take into account when constructing the binary factor:

```
psych_ids <- rowSums(sapply(c("psych", "psyhcology", "psycotherapy", "couns",
                              "behavior", "behaviour", "neuro"),
                            function(x) grepl(x, data$major, ignore.case = TRUE)))
anim_ids <- grepl("anim", data$major, ignore.case = TRUE) ## exclude animal psych
data$major <- factor(ifelse(psych_ids > 0, "psychology", "other"))
data$major[anim_ids > 0 & psych_ids > 0] <- "other"
```

We create identifiers to separate the dataset into 75% training and 25% test observations:

```
set.seed(42)
test_ids <- sample(1:nrow(data), ceiling(nrow(data)/4))
train_ids <- which(!1:nrow(data) %in% test_ids)
```

We create 0-1 coded versions of the response variable (for computing Brier scores):

```
train_y <- as.numeric(data$major)[train_ids] - 1
test_y <- as.numeric(data$major)[test_ids] - 1
```

Finally, we compute RIASEC scale scores by summing the item responses:

```
data$Real <- rowSums(data[ , paste0("R", 1:8)])
data$Inve <- rowSums(data[ , paste0("I", 1:8)])
data$Arti <- rowSums(data[ , paste0("A", 1:8)])
data$Soci <- rowSums(data[ , paste0("S", 1:8)])
data$Ente <- rowSums(data[ , paste0("E", 1:8)])
data$Conv <- rowSums(data[ , paste0("C", 1:8)])
```

Parameter Tuning

(Penalized) Logistic Regression

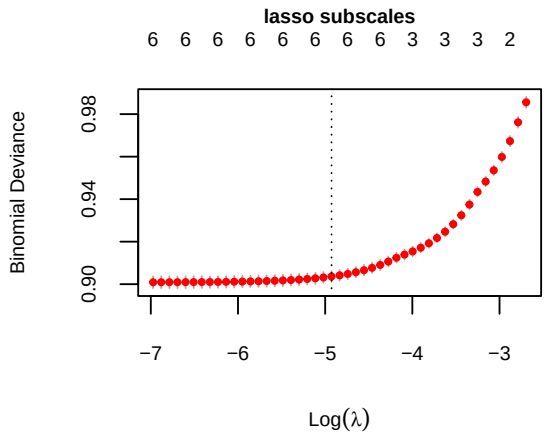
```
library("glmnet")

## Lasso scale scores
varnames <- c("Real", "Inve", "Arti", "Soci", "Ente", "Conv")
X <- as.matrix(data[train_ids, varnames])
set.seed(42)
l1 <- cv.glmnet(X, train_y, alpha = 1, family = "binomial")
lambda_l1 <- l1$lambda
## cv.glmnet() does not include lambda=0 by default, so need to include manually
set.seed(42)
l1 <- cv.glmnet(X, train_y, alpha = 1, family = "binomial",
```

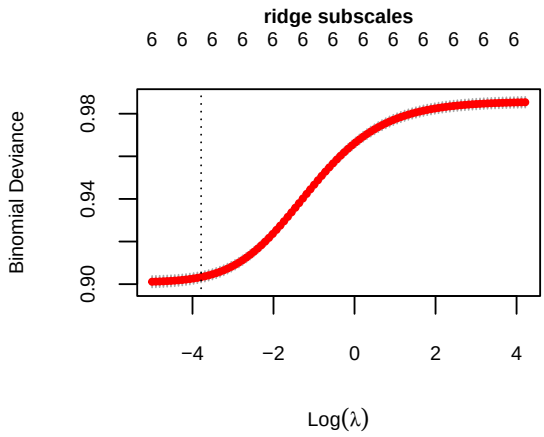
```
                  lambda = c(lambda_l1, 0))

## Ridge scale scores
set.seed(42)
l2 <- cv.glmnet(X, train_y, alpha = 0, family = "binomial")
lambda_l2 <- l2$lambda
set.seed(42)
l2 <- cv.glmnet(X, train_y, alpha = 0, lambda = c(lambda_l2, 0), family = "binomial")
```

For plotting the results with the adjusted penalty parameter path, we need a slightly adjusted plotting function:

```
plot.cv.glmnet <- function (x, sign.lambda = 1, cex = .7, main = "") {
  cvobj = x
  xlab = expression(Log(lambda))
  if (sign.lambda < 0)
    xlab = paste("-", xlab, sep = "")
  plot.args = list(x = sign.lambda * log(cvobj$lambda), y = cvobj$cvm,
                   ylim = range(cvobj$cvup, cvobj$cvlo), xlab = xlab, ylab = cvobj$name,
                   type = "n", cex = cex, cex.lab = cex, cex.main = cex, cex.axis = cex,
                   main = main)
  do.call("plot", plot.args)
  glmnet:::error.bars(sign.lambda * log(cvobj$lambda), cvobj$cvup, cvobj$cvlo,
                      width = 0.01, col = "darkgrey", cex = cex)
  points(sign.lambda * log(cvobj$lambda), cvobj$cvm, pch = 20,
         col = "red", cex = cex)
  axis(side = 3, at = sign.lambda * log(cvobj$lambda), labels = paste(cvobj$nz),
       tick = FALSE, line = 0, cex.axis = cex)
  abline(v = sign.lambda * log(cvobj$lambda.min), lty = 3)
  abline(v = sign.lambda * log(cvobj$lambda.1se), lty = 3)
  invisible()
}
```

```
## Plot and print results
par(mfrow = c(1, 2))
plot(l1, cex = .7, main = "lasso subscales")
plot(l2, cex = .7, main = "ridge subscales")
```






```
l1$lambda.min
```

```
## [1] 0
```

```
l1$lambda.1se
```

```
## [1] 0.007250111
```

```
l2$lambda.min
```

```
## [1] 0
```

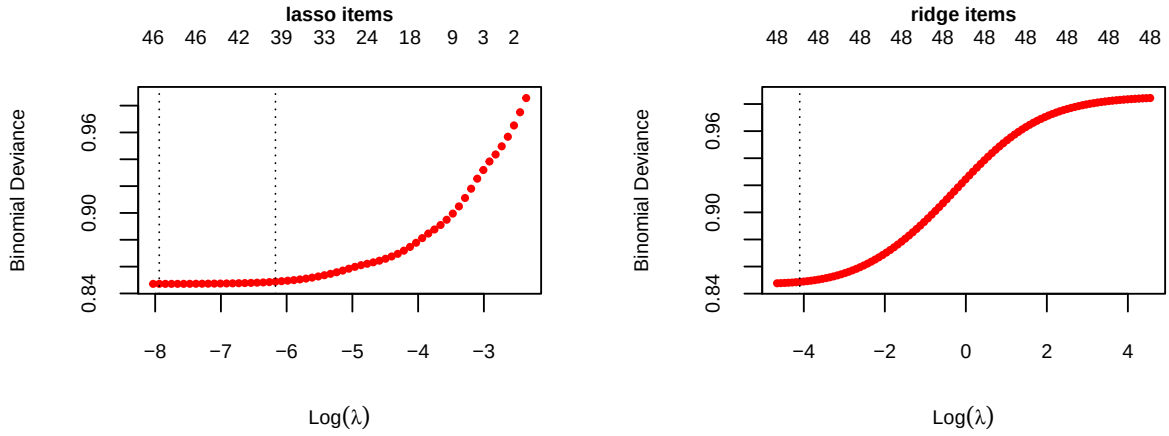
```
l2$lambda.1se
```

```
## [1] 0.02266179
```

```
## Items
varnames <- paste0(rep(c("R", "I", "A", "S", "E", "C"), each = 8), 1:8)
X <- as.matrix(data[train_ids, varnames])

## Lasso
set.seed(42)
l1 <- cv.glmnet(X, train_y, alpha = 1, family = "binomial")
lambda_l1 <- l1$lambda
set.seed(42)
l1 <- cv.glmnet(X, train_y, alpha = 1, family = "binomial",
                lambda = c(lambda_l1, 0))

## Ridge
set.seed(42)
l2 <- cv.glmnet(X, train_y, alpha = 0, family = "binomial")
lambda_l2 <- l2$lambda
set.seed(42)
```

```
l2 <- cv.glmnet(X, train_y, alpha = 0, lambda = c(lambda_l2, 0), family = "binomial")
```

```
## Plot and print results
par(mfrow = c(1, 2))
plot(l1, cex = .7, main = "lasso items")
plot(l2, cex = .7, main = "ridge items")
```




```
l1$lambda.min
```

```
## [1] 0.0003568404
```

```
l1$lambda.1se
```

```
## [1] 0.002090022
```

```
l2$lambda.min
```

```
## [1] 0
```

```
l2$lambda.1se
```

```
## [1] 0.01656306
```

```
l1$cvm[which(l1$lambda.min == l1$lambda)]
```

```
## [1] 0.8471471
```

```
l2$cvm[which(l2$lambda.min == l2$lambda)]
```

```
## [1] 0.8472255
```

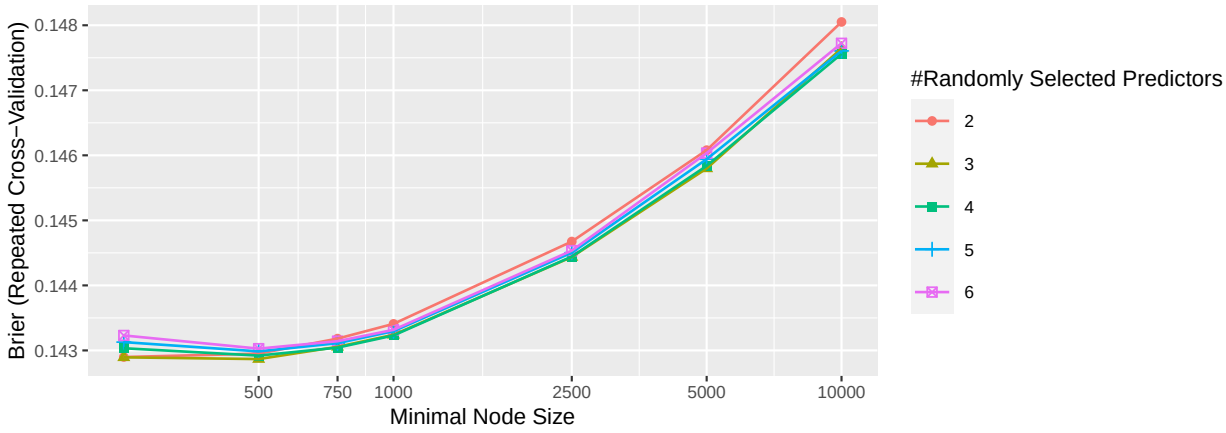
Random Forest

Parameters for gradient boosting, random forests and prediction rule ensembles were tuned using package **caret**.

```
## Load library, set up custom functions
library("caret")
library("ggplot2")
BigSummary <- function (data, lev = NULL, model = NULL) {
  brscore <- try(mean((data[, lev[2]] - ifelse(data$obs == lev[2], 1, 0)) ^ 2),
                 silent = TRUE)
  rocObject <- try(pROC::roc(ifelse(data$obs == lev[2], 1, 0), data[, lev[2]],
                             direction = "<", quiet = TRUE), silent = TRUE)
  if (inherits(brscore, "try-error")) brscore <- NA
  rocAUC <- if (inherits(rocObject, "try-error")) {
    NA
  } else {
    rocObject$auc
  }
  tmp <- unlist(e1071::classAgreement(table(data$obs,
                                            data$pred)))[c("diag", "kappa")]
  out <- c(Acc = tmp[[1]],
           Kappa = tmp[[2]],
           AUCROC = rocAUC,
           Brier = brscore)
  out
}
fitControl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 1,
                           ## Estimate class probabilities:
                           classProbs = TRUE,
                           ## Evaluate performance using
                           ## the following function:
                           summaryFunction = BigSummary,
                           verboseIter = TRUE)
```

```
## Subscales
rfGrid <-  expand.grid(mtry = c(2:6),
                       min.node.size = c(10000, 5000, 2500, 1000, 750, 500),
                       splitrule = "gini")
set.seed(825)
rfFit <- train(major ~ Real + Inve + Arti + Soci + Ente + Conv,
               data = data[train_ids, ], method = "ranger", trControl = fitControl,
               tuneGrid = rfGrid, metric = "Brier", maximize = FALSE)
```

```
## Print and plot results
ggplot(rfFit) + scale_x_continuous(trans="log") + theme_gray(base_size=9) +
  scale_x_continuous(trans = "log", breaks = c(10000, 5000, 2500, 1000, 750, 500))
```

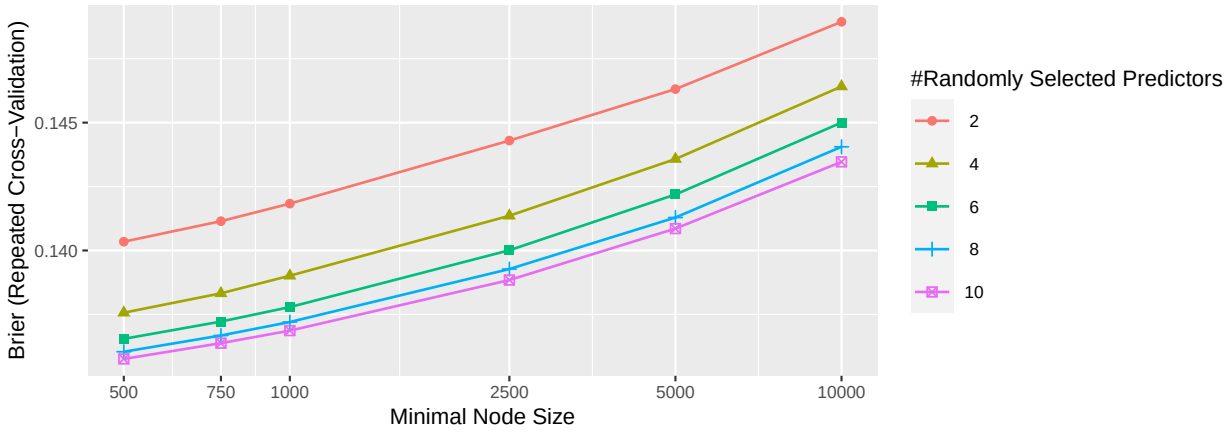



```
rfFit$bestTune
```

```
##   mtry splitrule min.node.size
## 9    3      gini           500
```

```
## Items
rfGrid_i <-  expand.grid(mtry = 2*(1:5),
                         min.node.size = c(10000, 5000, 2500, 1000, 750, 500),
                         splitrule = "gini")
x <- data[train_ids, paste0(rep(c("R", "I", "A", "S", "E", "C"), each = 8), 1:8)]
y <- data$major[train_ids]
set.seed(825)
rfFit_i <- train(x = x, y = y, method = "ranger", trControl = fitControl,
                 tuneGrid = rfGrid_i, metric = "Brier", maximize = FALSE)
```

```
## Print and plot results
ggplot(rfFit_i) + scale_x_continuous(trans="log") + theme_gray(base_size=9) +
  scale_x_continuous(trans = "log", breaks = c(10000, 5000, 2500, 1000, 750, 500))
```




```
rfFit_i$bestTune
```

```
##    mtry splitrule min.node.size
```

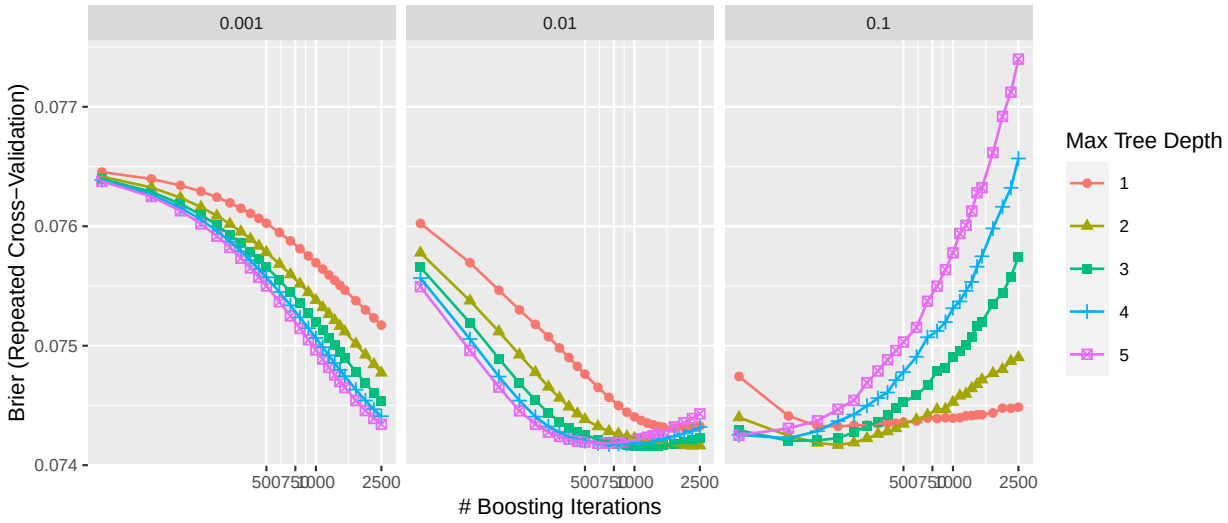
```
## 25    10       gini            500
```

Gradient Boosted Trees

Gradient boosting is one of the top prediction approaches. Many forecasting competitions have been won by using gradient boosting. To obtain good performance with boosting, careful tuning of the model-fitting parameters is necessary. The most important parameters are tree size, the number of trees in the ensemble and the shrinkage or learning rate. Tree size (or interaction depth) determines the highest degree of interactions that can be captured by the tree in the ensembles. The learning rate (or shrinkage) parameter reflects the weight that is attributed to the predictions of each previous tree, when fitting the current tree.

```
## Subscales
gbmGrid <-  expand.grid(interaction.depth = 1:5,
                        n.trees = c((1:10)*50, 500+(1:10)*100, 1500+(1:4)*250),
                        shrinkage = c(0.001, 0.01, 0.1),
                        n.minobsinnode = 20)
set.seed(825)
gbmFit <- train(major ~ Real + Inve + Arti + Soci + Ente + Conv,
                data = data[train_ids, ],
                 method = "gbm",
                 trControl = fitControl,
                 tuneGrid = gbmGrid,
                 metric = "Brier",
                 maximize = FALSE)
```

```
## Print and plot results
ggplot(gbmFit, size = 2) + theme_gray(base_size=9) +
  scale_x_continuous(trans = "log", breaks = c(10000, 5000, 2500, 1000, 750, 500))
```



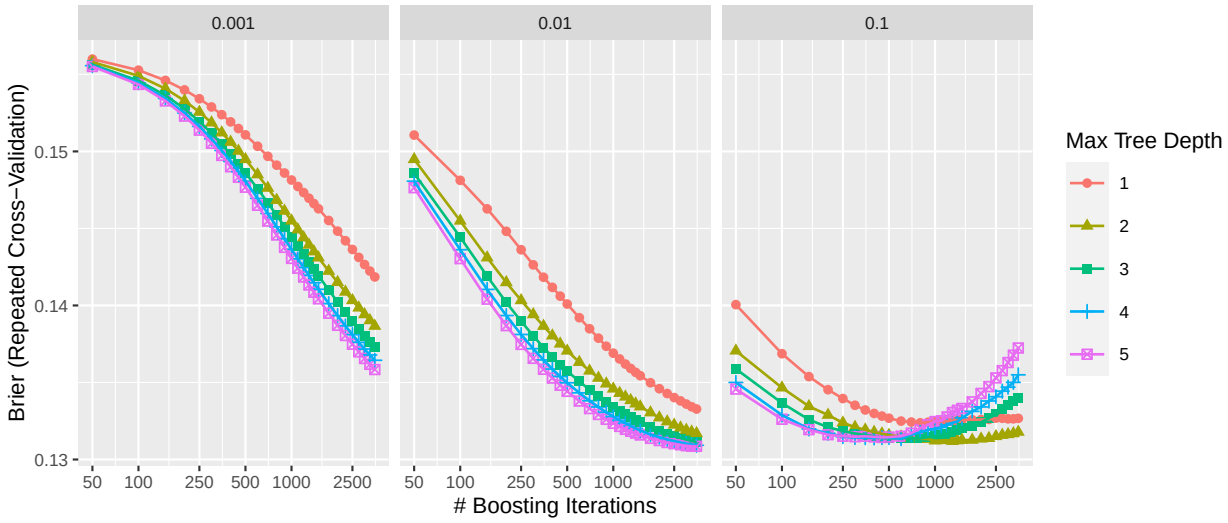

```
gbmFit$bestTune
```

```
##     n.trees interaction.depth shrinkage n.minobsinnode
```

```
## 184    1100                 3       0.01             20
```

```
## Items
x <- data[train_ids, paste0(rep(c("R", "I", "A", "S", "E", "C"), each = 8), 1:8)]
y <- data$major[train_ids]
gbmGrid_i <-  expand.grid(interaction.depth = 1:5,
                          n.trees = c((1:10)*50, 500+(1:10)*100, 1500+(1:8)*250),
                          shrinkage = c(0.001, 0.01, 0.1),
                          n.minobsinnode = 20)
set.seed(825)
gbmFit_i <- train(x = x, y = y,
                  method = "gbm",
                  trControl = fitControl,
                  tuneGrid = gbmGrid_i,
                  metric = "Brier",
                  maximize = FALSE)
```

```
## Print and plot results
ggplot(gbmFit_i) + theme_gray(base_size=9) +
  scale_x_continuous(trans = "log", breaks = c(50, 100, 250, 500, 1000, 2500))
```




```
gbmFit_i$bestTune
```

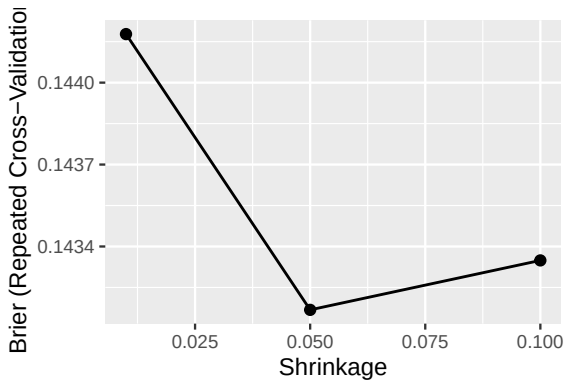
```
##     n.trees interaction.depth shrinkage n.minobsinnode
## 280    3500                 5       0.01             20
```

Prediction Rule Ensembling

Fitting prediction rule ensembles is computationally quite demanding. We therefore test only a small range of tuning parameters. For most tuning parameters, we expect the defaults to work well, but tuning the learning rate may likely improve predictive performance.

```
## Subscales
preGrid <- getModelInfo("pre")[[1]]$grid(
  learnrate = c(.01, .05, .1))
set.seed(825)
preFit <- train(major ~ Real + Inve + Arti + Soci + Ente + Conv,
                data = data[train_ids, ],
                method = "pre",
                trControl = fitControl,
                tuneGrid = preGrid,
                metric = "Brier",
                maximize = FALSE)
```

```
## Print and plot results
ggplot(preFit) + theme_gray(base_size=9)
```

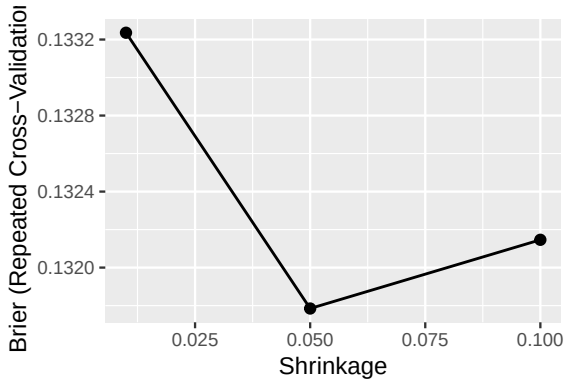



```
preFit$bestTune
```

```
##   sampfrac maxdepth learnrate mtry use.grad penalty.par.val
## 2      0.5        3      0.05  Inf     TRUE      lambda.1se
```

```
## items
varnames <- paste0(rep(c("R", "I", "A", "S", "E", "C"), each = 8), 1:8)
pr_form <- formula(paste("major ~", paste(varnames, collapse = "+")))
set.seed(825)
preFit_i <- train(pr_form,
                  data = data[train_ids, ],
                  method = "pre",
                  trControl = fitControl,
                  tuneGrid = preGrid,
                  metric = "Brier",
                  maximize = FALSE)
```

```
## Print and plot results
ggplot(preFit_i) + theme_gray(base_size=9)
```




```
preFit_i$bestTune
```

```
##   sampfrac maxdepth learnrate mtry use.grad penalty.par.val
## 2      0.5        3      0.05  Inf     TRUE      lambda.1se
```

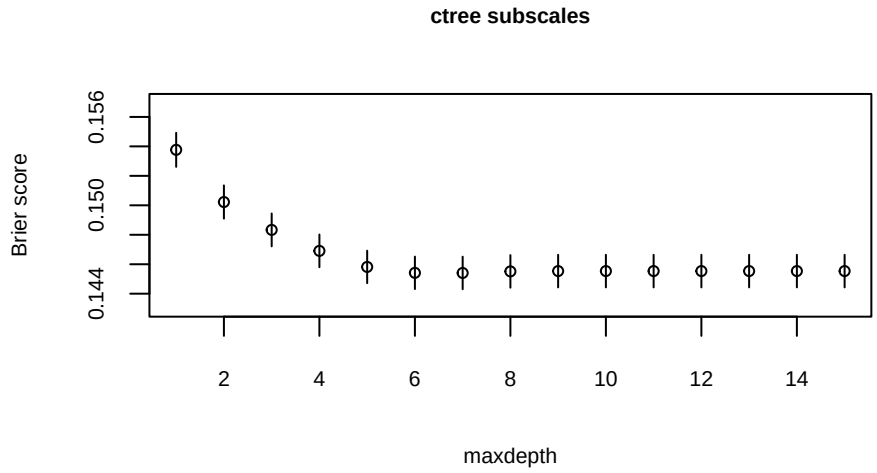
Conditional Inference Tree

For conditional inference trees and k nearest neighbours, we wrote custom code for tuning the parameters:

```
library("partykit")
dat <- data[train_ids, ]

## Subscales
varnames <- c("Real", "Inve", "Arti", "Soci", "Ente", "Conv")
ct_form <- formula(paste("major ~", paste(varnames, collapse = "+")))
set.seed(42)
fold_ids <- sample(1:10, size = nrow(dat), replace = TRUE)
ct_preds <- data.frame(matrix(rep(NA, times = nrow(dat)*15), nrow = nrow(dat)))
names(ct_preds) <- paste0("m", 1:15)

set.seed(43)
for (i in 1:10) {
  cat("Fold", i, ". ")
  for (j in 1:15) {
    ct <- ctree(ct_form, data = dat[fold_ids != i, ], maxdepth = j)
    ct_preds[fold_ids == i, paste0("m", j)] <- predict(
      ct, type = "prob", newdata = dat[fold_ids == i, ])[ , "psychology"]
  }
}
```

```
## Print and plot results
br_ct <- sapply(ct_preds, function(x) mean((x - train_y)^2))
br_ct_se <- sapply(ct_preds, function(x) sd((x - train_y)^2)/sqrt(length(train_ids)))
plot(br_ct, xlab = "maxdepth", ylab = "Brier score", main = "ctree subscales",
     ylim = c(0.143, 0.157), cex = .7, cex.axis = .7, cex.main = .7, cex.lab = .7)
```

```
arrows(x0 = 1:15, y0 = br_ct - br_ct_se, y1 = br_ct + br_ct_se, length = 0)
```



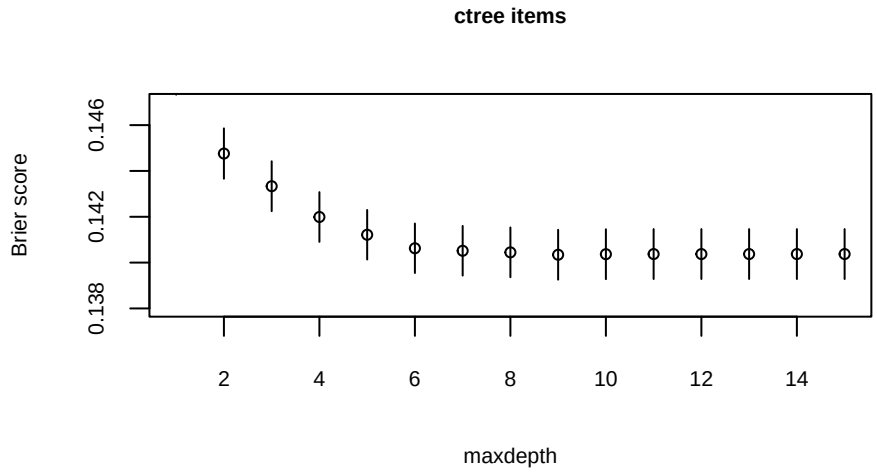

```
which(br_ct == min(br_ct))
```

```
## m7
##  7
```

```
## Items
varnames <- paste0(rep(c("R", "I", "A", "S", "E", "C"), each = 8), 1:8)
ct_form <- formula(paste("major ~", paste(varnames, collapse = "+")))
set.seed(42)
fold_ids <- sample(1:10, size = nrow(dat), replace = TRUE)
ct_preds <- data.frame(matrix(rep(NA, times = nrow(dat)*15), nrow = nrow(dat)))
names(ct_preds) <- paste0("m", 1:15)

set.seed(43)
for (i in 1:10) {
  cat("Fold", i, ". ")
  for (j in 1:15) {
    ct <- ctree(ct_form, data = dat[fold_ids != i, ], maxdepth = j)
    ct_preds[fold_ids == i, paste0("m", j)] <- predict(
      ct, type = "prob", newdata = dat[fold_ids == i, ])[ , "psychology"]
  }
}
```

```
## Print and plot results
br_ct <- sapply(ct_preds, function(x) mean((x - train_y)^2))
br_ct_se <- sapply(ct_preds, function(x) sd((x - train_y)^2)/sqrt(length(train_ids)))
plot(br_ct, xlab = "maxdepth", ylab = "Brier score", main = "ctree items",
     ylim = c(0.138, 0.147), cex = .7, cex.axis = .7, cex.main = .7, cex.lab = .7)
```

```
arrows(x0 = 1:15, y0 = br_ct - br_ct_se, y1 = br_ct + br_ct_se, length = 0)
```




```
which(br_ct == min(br_ct))
```

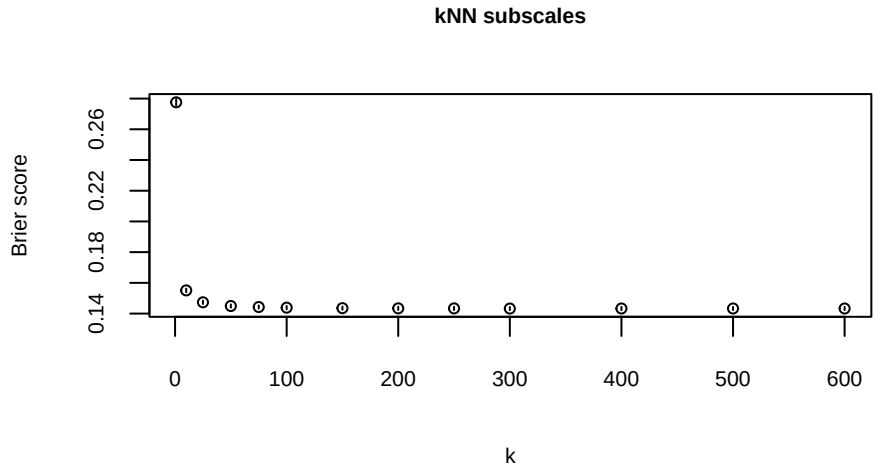
```
## m9
##  9
```

k Nearest Neighbours

```
library("class")

## Subscales
varnames <- c("Real", "Inve", "Arti", "Soci", "Ente", "Conv")
set.seed(42)
fold_ids <- sample(1:10, size = nrow(dat), replace = TRUE)
names(ct_preds) <- paste0("m", 1:15)
k <- c(1L, 10L, 25L, 50L, 75L, 100L, 150L, 200L, 250L, 300L, 400L, 500L, 600L)
knn_preds <- data.frame(matrix(rep(NA, times = nrow(dat)*length(k)),
                               nrow = nrow(dat)))
names(knn_preds) <- as.character(k)
set.seed(43)
for (i in 1:10) {
  cat("Fold", i, ". ")
  for (j in k) {
    try(
      knn_mod <- knn(dat[fold_ids != i, varnames], dat[fold_ids == i, varnames],
          cl = as.factor(dat[fold_ids != i, "major"]),
          k = j, use.all = TRUE, prob = TRUE)
    )
```

```
    ## Need to obtain predicted probability for second class
    knn_preds[fold_ids == i, as.character(j)] <- ifelse(
      knn_mod == "psychology", attr(knn_mod, "prob"), 1 - attr(knn_mod, "prob"))
  }
}
```

```
## Print and plot results
br_knn <- sapply(knn_preds, function(x) mean((x - train_y)^2))
br_knn_se <- sapply(knn_preds, function(x) sd((x - train_y)^2)/sqrt(length(train_ids)))
plot(k, br_knn, main = "kNN subscales", ylab = "Brier score",
     cex = .7, cex.axis = .7, cex.main = .7, cex.lab = .7)
arrows(x0 = k, y0 = br_knn - br_knn_se, y1 = br_knn + br_knn_se, length = 0)
```



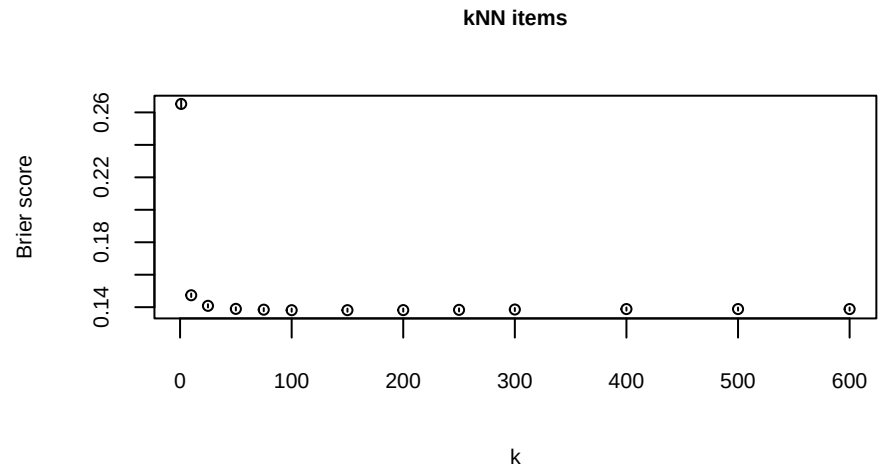

```
which(br_knn == min(br_knn))
```

```
## 300
##  10
```

```
## Items
varnames <- paste0(rep(c("R", "I", "A", "S", "E", "C"), each = 8), 1:8)
set.seed(42)
fold_ids <- sample(1:10, size = nrow(dat), replace = TRUE)
k <- c(1L, 10L, 25L, 50L, 75L, 100L, 150L, 200L, 250L, 300L, 400L, 500L, 600L)
knn_preds <- data.frame(matrix(rep(NA, times = nrow(dat)*length(k)),
                               nrow = nrow(dat)))
names(knn_preds) <- as.character(k)

set.seed(43)
for (i in 1:10) {
  cat("Fold", i, ". ")
```

```
  for (j in k) {
    try(
      knn_mod <- knn(dat[fold_ids != i, varnames], dat[fold_ids == i, varnames],
                     cl = as.factor(dat[fold_ids != i, "major"]),
                     k = j, use.all = TRUE, prob = TRUE)
    )
    ## Need to obtain predicted probability for second class
    knn_preds[fold_ids == i, as.character(j)] <- ifelse(
      knn_mod == "psychology", attr(knn_mod, "prob"), 1 - attr(knn_mod, "prob"))
  }
}
```

```
## Print and plot results
br_knn <- sapply(knn_preds, function(x) mean((x - train_y)^2))
br_knn_se <- sapply(knn_preds, function(x) sd((x - train_y)^2)/sqrt(length(train_ids)))
plot(k, br_knn, main = "kNN items", ylab = "Brier score",
     cex = .7, cex.axis = .7, cex.main = .7, cex.lab = .7)
arrows(x0 = k, y0 = br_knn - br_knn_se, y1 = br_knn + br_knn_se, length = 0)
```




```
which(br_knn == min(br_knn))
```

```
## 100
##   6
```

R Version and Package Info

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
```

```
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Dutch_Netherlands.1252  LC_CTYPE=Dutch_Netherlands.1252
## [3] LC_MONETARY=Dutch_Netherlands.1252 LC_NUMERIC=C
## [5] LC_TIME=Dutch_Netherlands.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] caret_6.0-89    lattice_0.20-44 ggplot2_3.3.5   glmnet_4.1-3
## [5] Matrix_1.3-4
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.7           lubridate_1.7.10     listenv_0.8.0
##  [4] class_7.3-19         assertthat_0.2.1     digest_0.6.27
##  [7] ipred_0.9-12         foreach_1.5.1        utf8_1.2.1
## [10] parallelly_1.28.1    R6_2.5.0             plyr_1.8.6
## [13] stats4_4.1.0         evaluate_0.14        highr_0.9
## [16] pillar_1.6.1         rlang_0.4.11         data.table_1.14.0
## [19] rpart_4.1.16         rmarkdown_2.11       labeling_0.4.2
## [22] splines_4.1.0        gower_0.2.2          stringr_1.4.0
## [25] munsell_0.5.0        compiler_4.1.0       xfun_0.29
## [28] pkgconfig_2.0.3      shape_1.4.6          globals_0.14.0
## [31] htmltools_0.5.1.1    nnet_7.3-16          tidyselect_1.1.1
## [34] tibble_3.1.2         prodlim_2019.11.13   codetools_0.2-18
## [37] fansi_0.5.0          future_1.22.1        crayon_1.4.1
## [40] dplyr_1.0.7          withr_2.4.2          ModelMetrics_1.2.2.2
## [43] MASS_7.3-54          recipes_0.1.17       grid_4.1.0
## [46] nlme_3.1-152         gtable_0.3.0         lifecycle_1.0.0
## [49] DBI_1.1.1            magrittr_2.0.1       pROC_1.18.0
## [52] scales_1.1.1         future.apply_1.8.1   stringi_1.6.2
## [55] farver_2.1.0         reshape2_1.4.4       timeDate_3043.102
## [58] ellipsis_0.3.2       generics_0.1.0       vctrs_0.3.8
## [61] lava_1.6.10          iterators_1.0.13     tools_4.1.0
## [64] glue_1.4.2           purrr_0.3.4          parallel_4.1.0
## [67] survival_3.2-11      yaml_2.2.1           colorspace_2.0-2
## [70] knitr_1.37
```

References

Fokkema, M. (2020). Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software*, *92*(1), 1–30.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1.

Greenwell, B., Boehmke, B., Cunningham, J., & Developers, G. (2020). *Gbm: Generalized boosted regression models.* https://CRAN.R-project.org/package=gbm

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, *15*(3), 651–674. https://doi.org/10.1198/106186006X133933

Kuhn, M. (2021). *Caret: Classification and regression training.* https://CRAN.R-project.org/package=caret

R Core Team. (2021). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing. https://www.R-project.org/

Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with S* (4th Edition). Springer. https://www.stats.ox.ac.uk/pub/MASS4/

Wood, S. N. (2017). *Generalized additive models: An introduction with R.* CRC press.

Wright, M. N., & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, *77*(1), 1–17. https://doi.org/10.18637/jss.v077.i01