**Supplementary Information on Methodology**

In this section, we provide more technical details on how our three models are estimated.

*Regularized logistic regression*: In L1 regularized logistic regression, the key parameter to tune is λ. The parameter λ is the coefficient weighting the regularization term in the estimation procedure. It effectively controls the degree of shrinkage and sparsity which is a coefficient weighting the regularization term, with larger values inducing more sparsity and shrinkage.

This parameter is typically tuned using *k*-fold cross-validation. Cross-validation is a technique used to tune hyperparameters of machine learning models, such as λ for L1 regularized logistic regression. In *k*-fold cross-validation, the training data set is divided into *k* equally sized subsets (“folds”). The overall procedure then works as follows:

1. Select one of the *k* folds to serve as the test set, and the remaining *k* – 1 folds to serve as the training set.
2. Use the *k* – 1 folds to estimate the model for different values of the hyperparameter. (In the case of regularized logistic regression, we would estimate the regularized logistic regression model for different values of λ.)
3. For each value of the hyperparameter, compute the corresponding model’s performance on the test set.

Steps 1 – 3 are repeated *k* times, with each of the *k* folds serving as the test set. For each value of the hyperparameter, the test set performance is averaged over the *k* held out folds. The final value of the hyperparameter is simply the value that gives the best performance averaged over the *k* folds*.*

In our analysis, we tuned the regularization parameter λ using *k*-fold cross validation with *k* = 5 folds. We estimated all regularized logistic regression models using the *glmnet* package in R.1

*Random forest*: In a random forest, each classification tree is randomized by building it from a bootstrapped sample of the training data set and by restricting the set of features that may be selected for splitting to a random sample of all features.2 The rationale for this randomization is two-fold. First, it is well-known in machine learning that individual classification trees can have poor predictive performance, and are often inferior to logistic regression.3 By taking bootstrap samples of the original data set, growing a classification tree on each such bootstrapped sample, and aggregating the trees, one can improve the accuracy significantly.4 Second, by restricting the tree induction procedure to consider random samples of the features, one can reduce the correlation between the trees in the forest, which also helps to improve the accuracy.5

The main parameters in the random forest model are *ntree*, the number of trees to grow; *mtry*, the size of the random sample of features selected for splitting; and *nodesize*, the minimum number of observations in each leaf of each tree. It is common to set *ntree* to 500 trees, *mtry* to the square root of the number of features (for classification), and *nodesize* to 1 (for classification). For *ntree*, larger values are preferred so as to reduce the random variation in the predictions. For *mtry* and *nodesize*, it is known that tuning these parameters typically has a minimal effect on performance, and that the default values above are often near-optimal.6,7 We thus use the default values above. We estimated all random forest models using the *ranger* package in R.8

*Gradient boosted trees*: We estimated all gradient boosted tree models using the *xgboost* package in R.9 In *xgboost*, the key parameters that need to be tuned are *eta*, the learning rate, and *depth*, which is the maximum allowable depth of each tree. We set the number of iterations *nrounds* to 100. To tune *eta* and *depth*, we applied *k*-fold cross validation on the training set with *k* = 5 folds. We tested the range {0.001, 0.01, 0.02, 0.05, 0.1} for *eta* and the range {2,3,4,5} for *depth*.

**Supplemental References**

1. Friedman J, Hastie T, Tibshirani R. glmnet: Lasso and elastic-net regularized generalized linear models. R Packag version 2009;1.

2. Breiman L. Random Forests. Mach Learn 2001;45:5–32.

3. James G, Witten D, Hastie T, Tibshirani R. Statistical Learning. In: Springer, 2013:15–57.

4. Breiman L. Bagging predictors. Mach Learn 1996;24:123–40.

5. Ho TK. The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 1998;20:832–44.

6. Tang GH, Rabie ABM, Hägg U. Classification and regression by randomForest. R News 2004;83:434–8.

7. Biau G, Scornet E. A random forest guided tour. Test 2016;25:197–227.

8. Wright MN, Ziegler A. ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. 2015.

9. Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDDD International Conference on Knowledge Discovery and Data Mining., 2016:785–94.