mikropml: User-Friendly R Package for Supervised Machine Learning Pipelines

Begüm D. Topçuoğlu, Zena Lapp, Kelly L. Sovacool, Evan Snitkin, Jenna Wiens, Patrick D. Schloss

2021-01-22

Summary

Machine learning (ML) for classification and prediction based on a set of features is used to make decisions in healthcare, economics, criminal justice and more. However, implementing an ML pipeline including preprocessing, model selection, and evaluation can be time-consuming, confusing, and difficult. Here, we present `mikropml` (prononced "meek-ROPE em el"), an easy-to-use R package that implements ML pipelines using regression, support vector machines, decision trees, random forest, or gradient-boosted trees. The package is available on GitHub, CRAN, and conda.

Statement of need

Most applications of machine learning (ML) require reproducible steps for data pre-processing, cross-validation, testing, model evaluation, and often interpretation of why the model makes particular predictions. Performing these steps is important, as failure to implement them can result in incorrect and misleading results (Teschendorff 2019; Wiens et al. 2019).

Supervised ML is widely used to recognize patterns in large datasets and to make predictions about outcomes of interest. Several packages including `caret` (Kuhn 2008) and `tidymodels` (Kuhn, Wickham, and RStudio 2020) in R, `scikitlearn` (Pedregosa et al. 2011) in Python, and the H2O `autoML` platform (H2O.ai 2020) allow scientists to train ML models with a variety of algorithms. While these packages provide the tools necessary for each ML step, they do not implement a complete ML pipeline according to good practices in the literature. This makes it difficult for practitioners new to ML to easily begin to perform ML analyses.

To enable a broader range of researchers to apply ML to their problem domains, we created `mikropml`, an easy-to-use R package (R Core Team 2020) that implements the ML pipeline created by Topçuoğlu *et al.* (Topçuoğlu et al. 2020) in a single function that returns a trained model, model performance metrics and feature importance. `mikropml` leverages the `caret` package to support several ML algorithms: linear regression, logistic regression, support vector machines with a radial basis kernel, decision trees, random forest, and gradient boosted trees. It incorporates good practices in ML training, testing, and model evaluation (Topçuoğlu et al. 2020; Teschendorff 2019). Furthermore, it provides data preprocessing steps based on the FIDDLE (FlexIble Data-Driven pipeLinE) framework outlined in Tang *et al.* (Tang et al. 2020) and post-training permutation importance steps to estimate the importance of each feature in the models trained (Breiman 2001; Fisher, Rudin, and Dominici 2018).

`mikropml` can be used as a starting point in the application of ML to datasets from many different fields. It has already been applied to microbiome data to categorize patients with colorectal cancer (Topçuoğlu et al. 2020), to identify differences in genomic and clinical features associated with bacterial infections (Lapp et al. 2020), and to predict gender-based biases in academic publishing (Hagan et al. 2020).

mikropml package

The mikropml package includes functionality to preprocess the data, train ML models, evaluate model performance, and quantify feature importance (Figure 1). We also provide vignettes and an example Snakemake workflow (Köster and Rahmann 2012) to showcase how to run an ideal ML pipeline with multiple different train/test data splits. The results can be visualized using helper functions that use ggplot2 (Wickham 2016).

While mikropml allows users to get started quickly and facilitates reproducibility, it is not a replacement for understanding the ML workflow which is still necessary when interpreting results (Pollard et al. 2019). To facilitate understanding and enable one to tailor the code to their application, we have heavily commented the code and have provided supporting documentation which can be read online.

Preprocessing data

We provide the function preprocess_data() to preprocess features using several different functions from the caret package. preprocess_data() takes continuous and categorical data, re-factors categorical data into binary features, and provides options to normalize continuous data, remove features with near-zero variance, and keep only one instance of perfectly correlated features. We set the default options based on those implemented in FIDDLE (Tang et al. 2020). More details on how to use preprocess_data() can be found in the accompanying vignette.

Running ML

The main function in mikropml, run_ml(), minimally takes in the model choice and a data frame with an outcome column and feature columns. For model choice, mikropml currently supports logistic and linear regression (glmnet: Friedman, Hastie, and Tibshirani 2010), support vector machines with a radial basis kernel (kernlab: Karatzoglou et al. 2004), decision trees (rpart: Therneau et al. 2019), random forest (randomForest: Liaw and Wiener 2002), and gradient-boosted trees (xgboost: Chen et al. 2020). run_ml() randomly splits the data into train and test sets while maintaining the distribution of the outcomes found in the full dataset. It also provides the option to split the data into train and test sets based on categorical variables (e.g. batch, geographic location, etc.). mikropml uses the caret package (Kuhn 2008) to train and evaluate the models, and optionally quantifies feature importance. The output includes the best model built based on tuning hyperparameters in an internal and repeated cross-validation step, model evaluation metrics, and optional feature importances. Feature importances are calculated using a permutation test, which breaks the relationship between the feature and the true outcome in the test data, and measures the change in model performance. This provides an intuitive metric of how individual features influence model performance and is comparable across model types, which is particularly useful for model interpretation (Topçuoğlu et al. 2020). Our introductory vignette contains a comprehensive tutorial on how to use run_ml().

Ideal workflow for running mikropml with many different train/test splits

To investigate the variation in model performance depending on the train and test set used (Topçuoğlu et al. 2020; Lapp et al. 2020), we provide examples of how to run_ml() many times with different train/test splits and how to get summary information about model performance on a local computer or on a high-performance computing cluster using a Snakemake workflow.

Tuning & visualization

One particularly important aspect of ML is hyperparameter tuning. We provide a reasonable range of default hyperparameters for each model type. However practitioners should explore whether that range is appropriate for their data, or if they should customize the hyperparameter range. Therefore, we provide a function plot_hp_performance() to plot the cross-validation performance metric of a single model or models built using different train/test splits. This helps evaluate if the hyperparameter range is being searched exhaustively and allows the user to pick the ideal set. We also provide summary plots of test performance metrics for the

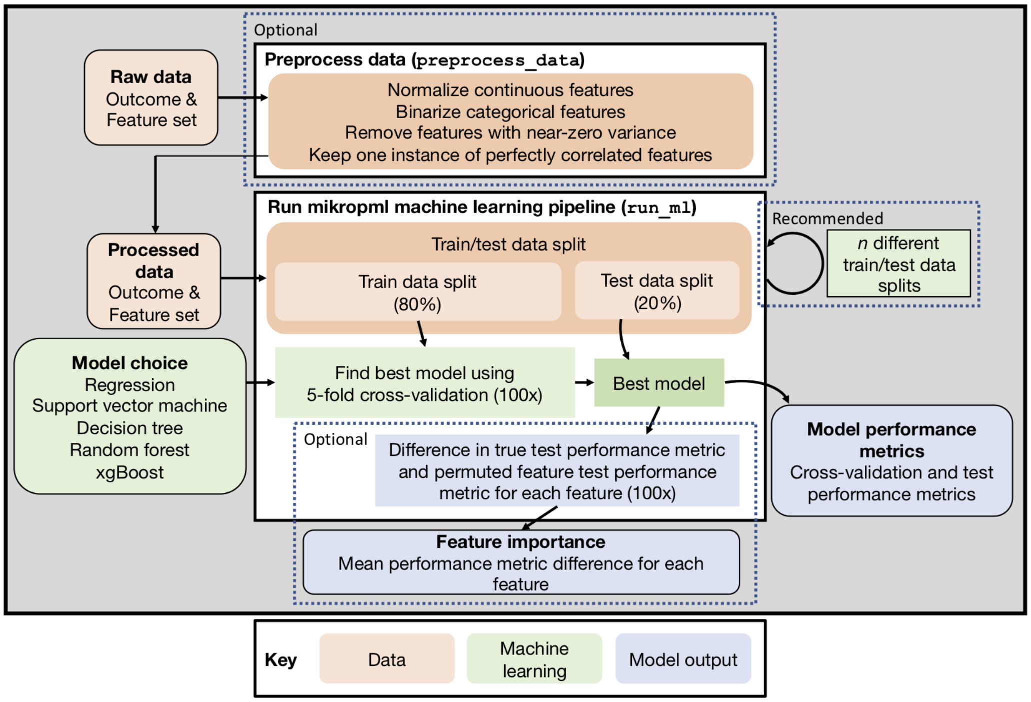
Optional
Preprocess data (preprocess_data)
Raw data
Outcome &
Feature set
Normalize continuous features
Binarize categorical features
Remove features with near-zero variance
Keep one instance of perfectly correlated features
Run mikropml machine learning pipeline (run_ml)
Recommended
n different
train/test data
splits
Processed
data
Outcome &
Feature set
Train/test data split
Train data split
(80%)
Test data split
(20%)
Model choice
Regression
Support vector machine
Decision tree
Random forest
xgBoost
Find best model using
5-fold cross-validation (100x)
Best model
Model performance
metrics
Cross-validation and test
performance metrics
Optional
Difference in true test performance metric
and permuted feature test performance
metric for each feature (100x)
Feature importance
Mean performance metric difference for each
feature
Key
Data
Machine
learning
Model output

Figure 1: mikropml pipeline

many train/test splits with different models using `plot_model_performance()`. Examples are described in the accompanying vignette on hyperparameter tuning.

Dependencies

mikropml is written in R (R Core Team 2020) and depends on several packages: `dplyr` (Wickham et al. 2020), `rlang` (Henry, Wickham, and RStudio 2020) and `caret` (Kuhn 2008). The ML algorithms supported by `mikropml` require: `glmnet` (Friedman, Hastie, and Tibshirani 2010), `e1071` (Meyer et al. 2020), and `MLmetrics` (Yan 2016) for logistic regression, `rpart2` (Therneau et al. 2019) for decision trees, `randomForest` (Liaw and Wiener 2002) for random forest, `xgboost` (Chen et al. 2020) for xgboost, and `kernlab` (Karatzoglou et al. 2004) for support vector machines. We also allow for parallelization of cross-validation and other steps using the `foreach`, `doFuture`, `future.apply`, and `future` packages (Bengtsson and Team 2020). Finally, we use `ggplot2` for plotting (Wickham 2016).

Acknowledgments

We thank members of the Schloss Lab who participated in code clubs related to the initial development of the pipeline, made documentation improvements, and provided general feedback.

Funding

Salary support for PDS came from NIH grant 1R01CA215574. KLS received support from the NIH Training Program in Bioinformatics (T32 GM070449). ZL received support from the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1256260. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Author contributions

BDT, ZL, and KLS contributed equally. Author order among the co-first authors was determined by time since joining the project.

BDT, ZL, and KLS conceptualized the study and wrote the code. KLS structured the code in R package form. BDT, ZL, JW, and PDS developed methodology. PDS, ES, and JW supervised the project. BDT, ZL, and KLS wrote the original draft. All authors reviewed and edited the manuscript.

Conflicts of interest

None.

References

Bengtsson, Henrik, and R Core Team. 2020. "Future.Apply: Apply Function to Elements in Parallel Using Futures," July.

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

Chen, Tianqi, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. 2020. "Xgboost: Extreme Gradient Boosting," June.

Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. 2018. "All Models Are Wrong, but Many Are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously."

Friedman, Jerome H., Trevor Hastie, and Rob Tibshirani. 2010. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33 (1): 1–22. https://doi.org/10.18637/jss.v033.i01.

H2O.ai. 2020. *H2O: Scalable Machine Learning Platform.* Manual.

Hagan, Ada K., Begüm D. Topçuoğlu, Mia E. Gregory, Hazel A. Barton, and Patrick D. Schloss. 2020. "Women Are Underrepresented and Receive Differential Outcomes at ASM Journals: A Six-Year Retrospective Analysis." *mBio* 11 (6). https://doi.org/10.1128/mBio.01680-20.

Henry, Lionel, Hadley Wickham, and RStudio. 2020. "Rlang: Functions for Base Types and Core R and 'Tidyverse' Features," July.

Karatzoglou, Alexandros, Alexandros Smola, Kurt Hornik, and Achim Zeileis. 2004. "Kernlab - an S4 Package for Kernel Methods in R." *Journal of Statistical Software* 11 (1): 1–20. https://doi.org/10.18637/jss.v011.i09.

Köster, Johannes, and Sven Rahmann. 2012. "Snakemakea Scalable Bioinformatics Workflow Engine." *Bioinformatics* 28 (19): 2520–2. https://doi.org/10.1093/bioinformatics/bts480.

Kuhn, Max. 2008. "Building Predictive Models in R Using the Caret Package." *Journal of Statistical Software* 28 (1): 1–26. https://doi.org/10.18637/jss.v028.i05.

Kuhn, Max, Hadley Wickham, and RStudio. 2020. "Tidymodels: Easily Install and Load the 'Tidymodels' Packages," July.

Lapp, Zena, Jennifer Han, Jenna Wiens, Ellie JC Goldstein, Ebbing Lautenbach, and Evan Snitkin. 2020. "Machine Learning Models to Identify Patient and Microbial Genetic Factors Associated with Carbapenem-Resistant Klebsiella Pneumoniae Infection." *medRxiv*, July, 2020.07.06.20147306. https://doi.org/10.1101/2020.07.06.20147306.

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by randomForest" 2: 5.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, Friedrich Leisch, Chih-Chung Chang (libsvm C++-code), and Chih-Chen Lin (libsvm C++-code). 2020. "E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien."

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12 (85): 2825–30.

Pollard, Tom J., Irene Chen, Jenna Wiens, Steven Horng, Danny Wong, Marzyeh Ghassemi, Heather Mattie, Emily Lindemer, and Trishan Panch. 2019. "Turning the Crank for Machine Learning: Ease, at What Expense?" *The Lancet Digital Health* 1 (5): e198–e199. https://doi.org/10.1016/S2589-7500(19)30112-8.

R Core Team. 2020. "R: A Language and Environment for Statistical Computing."

Tang, Shengpu, Parmida Davarmanesh, Yanmeng Song, Danai Koutra, Michael W. Sjoding, and Jenna Wiens. 2020. "Democratizing EHR Analyses with FIDDLE: A Flexible Data-Driven Preprocessing Pipeline for Structured Clinical Data." *J Am Med Inform Assoc*, October. https://doi.org/10.1093/jamia/ocaa139.

Teschendorff, Andrew E. 2019. "Avoiding Common Pitfalls in Machine Learning Omic Data Science." *Nature Materials* 18 (5): 422–27. https://doi.org/10.1038/s41563-018-0241-z.

Therneau, Terry, Beth Atkinson, Brian Ripley (producer of the initial R. port, and maintainer 1999-2017). 2019. "Rpart: Recursive Partitioning and Regression Trees," April.

Topçuoğlu, Begüm D., Nicholas A. Lesniak, Mack T. Ruffin, Jenna Wiens, and Patrick D. Schloss. 2020. "A Framework for Effective Application of Machine Learning to Microbiome-Based Classification Problems." *mBio* 11 (3). https://doi.org/10.1128/mBio.00434-20.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis.* Use R! Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-24277-4.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and RStudio. 2020. "Dplyr: A Grammar of Data Manipulation," August.

Wiens, Jenna, Suchi Saria, Mark Sendak, Marzyeh Ghassemi, Vincent X. Liu, Finale Doshi-Velez, Kenneth Jung, et al. 2019. "Do No Harm: A Roadmap for Responsible Machine Learning for Health Care." *Nat. Med.* 25 (9): 1337–40. https://doi.org/10.1038/s41591-019-0548-6.

Yan, Yachen. 2016. "MLmetrics: Machine Learning Evaluation Metrics."