International Epidemiological Association

International Journal of Epidemiology, 2023, 1276–1285
https://doi.org/10.1093/ije/dyad023
Advance Access Publication Date: 11 March 2023
Education Corner

OXFORD

Education Corner

Practical considerations for specifying a super learner

Rachael V Phillips ,[1]* Mark J van der Laan,[1] Hana Lee,[2] and Susan Gruber[3]

[1]Division of Biostatistics, School of Public Health, University of California at Berkeley, Berkeley, California, United States, [2]Office of Biostatistics, Center for Drug Evaluation and Research, United States Food and Drug Administration, Silver Spring, Maryland, United States and [3]Putnam Data Sciences, LLC, Cambridge, Massachusetts, United States

*Corresponding author. Division of Biostatistics, School of Public Health, University of California at Berkeley, 2121 Berkeley Way, Berkeley, California, 94704, United States. E-mail: rachaelvphillips@berkeley.edu

Received 1 April 2022; Editorial decision 16 January 2023; Accepted 23 February 2023

Abstract

Common tasks encountered in epidemiology, including disease incidence estimation and causal inference, rely on predictive modelling. Constructing a predictive model can be thought of as learning a prediction function (a function that takes as input covariate data and outputs a predicted value). Many strategies for learning prediction functions from data (learners) are available, from parametric regressions to machine learning algorithms. It can be challenging to choose a learner, as it is impossible to know in advance which one is the most suitable for a particular dataset and prediction task. The super learner (SL) is an algorithm that alleviates concerns over selecting the one 'right' learner by providing the freedom to consider many, such as those recommended by collaborators, used in related research or specified by subject-matter experts. Also known as stacking, SL is an entirely prespecified and flexible approach for predictive modelling. To ensure the SL is well specified for learning the desired prediction function, the analyst does need to make a few important choices. In this educational article, we provide step-by-step guidelines for making these decisions, walking the reader through each of them and providing intuition along the way. In doing so, we aim to empower the analyst to tailor the SL specification to their prediction task, thereby ensuring their SL performs as well as possible. A flowchart provides a concise, easy-to-follow summary of key suggestions and heuristics, based on our accumulated experience and guided by SL optimality theory.

Key words: Super learner, stacking, ensemble machine learning, prediction, causal inference, statistical data analysis, model validation, risk assessment, disease epidemiology, health outcomes

Key Messages

- Select a performance metric that aligns with the intended use of the algorithm (e.g. risk prediction, binary outcome classification, causal inference) and accurately reflects how closely it approximates the target prediction function of interest by considering the recommendations provided in Step 1 of the flowchart and corresponding text.
- Calculate the effective sample size from the analytical dataset by following Step 2 of the flowchart, to use as a guide for defining the cross-validation scheme (Step 3 of the flowchart) and the specification of the collection of algorithms included in the super learner (SL) library (Step 4 of the flowchart).
- Construct SL libraries according to the following criteria: create a rich library of algorithms which is diverse in its learning strategies, able to adapt to a range of underlying functional forms for the target prediction function, computationally feasible and effective at handling high-dimensional data.
- Distinguish between an ensemble SL (eSL) that combines predictions from multiple individual algorithms, and a discrete SL (dSL) that selects a single, best, performer based on cross-validated performance.
- Justify the use of the dSL to select the best performing algorithm, including one or more eSLs alongside each individual algorithm in the original library.

Introduction

To answer scientific questions and test hypotheses, we curate and learn from data. This often entails using the data to learn a prediction function, a function that takes in covariate data and outputs a predicted value. Prediction is used in epidemiology in a variety of contexts: disease phenotyping, evaluating risk factors for certain health outcomes, mapping the spread of disease, and event forecasting. Prediction is also an integral component in the estimation of causal effects. Traditionally, prediction functions were learned by fitting prespecified parametric regression models to data; however, more flexible learning algorithms have been shown to produce more accurate results. For example, in-hospital mortality prediction scores based on machine learning have been shown to improve upon conventional, logistic regression-based scores.[1]

In practice it is difficult to choose a single algorithm (or 'learner'). There are many options, and no one is expert in all of them. Moreover, it is impossible to know in advance which learner is most suitable for the particular dataset and prediction task. The super learner (SL) solves the issue of algorithm selection by considering a large set of user-specified algorithms, from parametric regressions to non-parametric machine learning algorithms (e.g. neural nets, support vector machines, and decision and regression trees). It alleviates concerns over selecting the one 'right' algorithm while benefiting from considering a diverse set, including those recommended by collaborators, used in related research or specified by subject-matter experts. The SL is grounded in optimality theory that guarantees for large sample sizes that the SL will perform as well as possible, given the specified algorithms considered.[2] Numerous, diverse practical applications support SL's broad robustness for pure prediction tasks, and also for hypothesis testing and causal inference.[1,3–7] SL plays an important role in associational and causal effect estimation, including targeted maximum likelihood estimation (TMLE), where it is recommended for estimating outcome regressions, propensity scores, missingness mechanisms; and inverse probability weighting (IPW) and matching-based estimation.[5–7]

Existing SL software packages in the R programming language,[8] *SuperLearner*[9] and *sl3*,[10] make using SL straightforward. By providing a single interface with a rich and diverse variety of algorithms, analysts are not required to learn the ins and outs of different software packages. Nevertheless, they do need to make a few choices to define an SL that is well specified for their prediction task. This paper provides an overview of key components of an SL specification and a flowchart that guides analysts through the process (Figure 1). A glossary of terminology used in the text is also provided (Box 1).

Overview of the super learner

The idea to 'lump together (algorithms) into one grand melee' dates back to the early 1960s (George Barnard).[11] It was not until 1992 that a methodology for doing this, stacking, was proposed by David Wolpert.[12] Specifics of the implementation, referred to by Wolpert as an 'art' in 1992, became a science in 1996, when Leo Breiman demonstrated the utility of non-negative least squares (NNLS) regression for combining predictions from algorithms fit to the same dataset (meta-learning).[13] In 2007, previous theory provided by Mark van der Laan, Sandrine Dudoit and Aad van der Vaart[14–16] was extended, proving that in large samples stacking represented an optimal system for

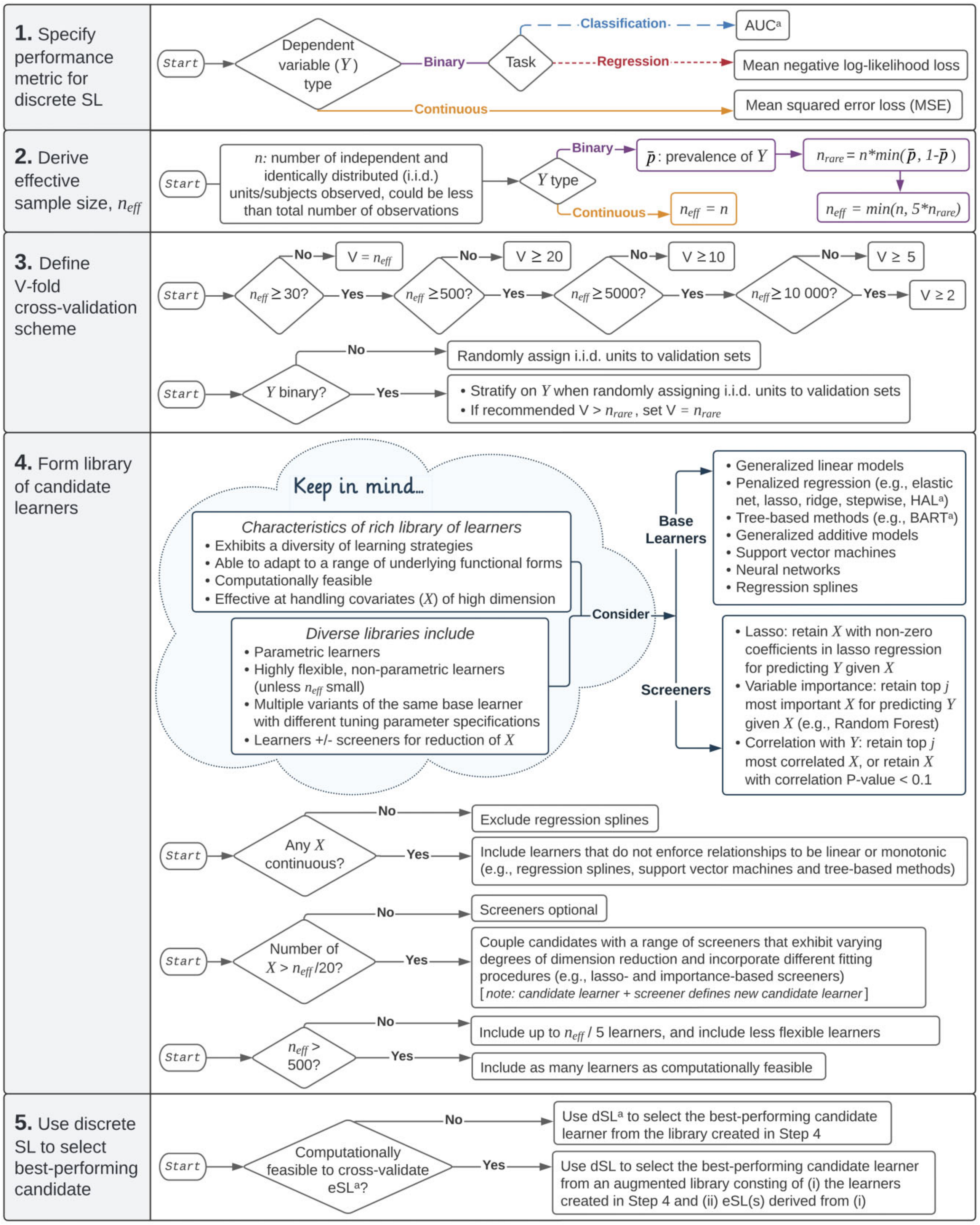
1. Specify performance metric for discrete SL
Start
Dependent variable (Y) type
Binary
Task
Classification
AUC[a]
Regression
Mean negative log-likelihood loss
Continuous
Mean squared error loss (MSE)
2. Derive effective sample size, n_{eff}
Start
n: number of independent and identically distributed (i.i.d.) units/subjects observed, could be less than total number of observations
Y type
Binary
$\bar{p}$: prevalence of Y
$n_{rare} = n*min(\bar{p}, 1-\bar{p})$
$n_{eff} = min(n, 5*n_{rare})$
Continuous
$n_{eff} = n$
3. Define V-fold cross-validation scheme
Start
$n_{eff} \geq 30$?
No
$V = n_{eff}$
Yes
$n_{eff} \geq 500$?
No
$V \geq 20$
Yes
$n_{eff} \geq 5000$?
No
$V \geq 10$
Yes
$n_{eff} \geq 10\ 000$?
No
$V \geq 5$
Yes
$V \geq 2$
Start
Y binary?
No
Randomly assign i.i.d. units to validation sets
Yes
• Stratify on Y when randomly assigning i.i.d. units to validation sets
• If recommended V > n_{rare}, set V = n_{rare}
4. Form library of candidate learners
Keep in mind...
Characteristics of rich library of learners
• Exhibits a diversity of learning strategies
• Able to adapt to a range of underlying functional forms
• Computationally feasible
• Effective at handling covariates (X) of high dimension
Diverse libraries include
• Parametric learners
• Highly flexible, non-parametric learners (unless n_{eff} small)
• Multiple variants of the same base learner with different tuning parameter specifications
• Learners +/- screeners for reduction of X
Consider
Base Learners
• Generalized linear models
• Penalized regression (e.g., elastic net, lasso, ridge, stepwise, HAL[a])
• Tree-based methods (e.g., BART[a])
• Generalized additive models
• Support vector machines
• Neural networks
• Regression splines
Screeners
• Lasso: retain X with non-zero coefficients in lasso regression for predicting Y given X
• Variable importance: retain top j most important X for predicting Y given X (e.g., Random Forest)
• Correlation with Y: retain top j most correlated X, or retain X with correlation P-value < 0.1
Start
Any X continuous?
No
Exclude regression splines
Yes
Include learners that do not enforce relationships to be linear or monotonic (e.g., regression splines, support vector machines and tree-based methods)
Start
Number of $X > n_{eff}/20$?
No
Screeners optional
Yes
Couple candidates with a range of screeners that exhibit varying degrees of dimension reduction and incorporate different fitting procedures (e.g., lasso- and importance-based screeners) [note: candidate learner + screener defines new candidate learner]
Start
$n_{eff} >$ 500?
No
Include up to n_{eff} / 5 learners, and include less flexible learners
Yes
Include as many learners as computationally feasible
5. Use discrete SL to select best-performing candidate
Start
Computationally feasible to cross-validate eSL[a]?
No
Use dSL[a] to select the best-performing candidate learner from the library created in Step 4
Yes
Use dSL to select the best-performing candidate learner from an augmented library consting of (i) the learners created in Step 4 and (ii) eSL(s) derived from (i)

Figure 1 Flowchart for specifying a super learner (SL). [a]AUC, area under the receiver-operating characteristic curve; HAL, highly adaptive lasso; BART, Bayesian additive regression trees; eSL, ensemble super learner; dSL, discrete super learner

Box 1 Glossary of terminology used in this paper

Term	Definition
Covariates, predictors (X)	The set of variables to use as input for predicting the outcome.
Outcome (Y)	The dependent variable to predict as a function of X.
Analytical dataset	The dataset containing observations on X and Y.
Prediction function	A function that takes in values of input variables and returns a predicted value. When Y is binary, the predicted value is a predicted probability.
Data-generating process (DGP), data-generating distribution	The unknown, underlying probability distribution describing the data at hand, characterizing the target population from which the data were drawn.
Target prediction function	The desired prediction function of interest for the target population, defined in terms of the DGP.
Statistical model	The set of probability distributions that could have given rise to the data at hand, embodying knowledge about the process that generated it and the study design.
Algorithm, learner, machine learning algorithm	A set of instructions that define a prediction function estimator, when fully specified. Estimating the prediction function (i.e. algorithm training/fitting) is an optimization problem; in learning the function of the input variables, the algorithm aims to optimize some performance metric/risk function (e.g. minimize the mean squared error).
Tuning parameter, hyperparameter	Algorithm parameters that control the learning process. Examples include the lasso penalty/shrinkage parameter and the number of trees in a random forest. They are typically chosen via some heuristic or resampling-based (e.g. cross-validation) predictive performance. A ‘fully specified’ algorithm is one with all tuning parameters specified.
Base algorithm, base learner	An algorithm that is not fully specified but defines a particular learning strategy. A base learner is used as a building block to define one or more fully specified learners. For example, lasso is not fully specified because it has the tuning parameter controlling the shrinkage of regression coefficients.
Library	The set of fully specified algorithms that will be considered by the super learner.
Candidate, candidate learner, candidate algorithm, candidate estimator	A fully specified algorithm included in the super learner library, optionally coupled with a screening algorithm.
Screener	A function that reduces the number of X, returning a subset of them, $\sim X \subseteq X$. Screeners can be based on relationships between X and Y that are learned from the analytical dataset, such as those listed in Step 4 of Figure 1. They can also be outcome blind, in the sense that the reduction of X is not based on Y, such as removing X that are extremely sparse. Screeners can be coupled with a candidate learner to define a new candidate learner that considers only $\tilde{X}$.
Meta-level covariates ($\hat{Y}$)	The set of variables to use as input by the meta-learner for predicting Y, which are transformations of X defined by trained candidates applied to X. Hence, the ‘meta’ nature of the meta-level covariates: they are covariates defined by a function of other covariates. Values of $\hat{Y}$ are predicted values returned by the trained candidates, given observed values of X (see Figure 2).
Meta-learner, meta-learning algorithm	A specified algorithm that is trained to consider $\hat{Y}$ as input variables. Hence, the ‘meta’ nature of the meta-learner: it learns from what is learned by the candidate learners (see Figure 2).
Meta-level dataset	The dataset for fitting the meta-learner, containing the cross-validated $\hat{Y}$ values and the corresponding validation set Y values (see Figure 2).
Super learner (SL), stacking	Just like any other algorithm, the SL is a prediction function estimator applied to input variables X. The SL’s estimated prediction function is characterized by the fitted candidates and the fitted meta-learner, defined as a function of the prediction functions estimated by the candidates (see Figure 2).
Discrete SL (dSL)	An SL that uses a winner-take-all meta-learner called the cross-validated selector. The dSL is identical to the candidate with the best cross-validated performance.
Ensemble SL (eSL)	An SL that uses any parametric or non-parametric algorithm as its meta-learner. Therefore, eSL predictions are defined as a combination of multiple candidates’ predictions. An example of eSL is provided in Figure 2. (Note that the dSL can be thought of as a highly constrained or superficial type of eSL, in which dSL predictions are a weighted combination of the candidates’ predictions, with weight of one given to predictions from the candidate with the best cross-validated predictive performance, and zero weight given to all other candidates’ predictions).

learning.[2] This theoretical grounding of stacking was when the algorithm took on an additional name, 'Super Learner'.

The SL learns a function of a specified set, or 'library', of algorithms (Figure 2). Each algorithm in the library is considered a 'candidate' estimator of the target prediction function (the desired prediction function for the target population, which is defined in terms of the unknown probability distribution that generated the data at hand). SLs can be grouped into two types, depending on their approach to meta-learning. An ensemble SL (eSL) can use any parametric or non-parametric algorithm to create a function that combines the predictions from each candidate. For example, NNLS produces a weighted sum of predictions from each candidate (Figure 2 Step 2). A discrete SL (dSL) uses a winner-takes-all approach to meta-learning; its predictions are identical to those from the best-performing candidate.

Cross-validation (CV) is essential to the SL in estimating how well a trained algorithm performs when making predictions on novel data drawn from the same distribution as the training data. V-fold CV, also known as K-fold CV, is a splitting of the dataset into V disjoint validation sets and corresponding training sets (the complement of each validation set, Figure 2 Step 1). For each fold, each candidate is trained on observations in the training set, and then predicted outcomes are obtained for observations in the validation set (cross-validated predictions, Figure 2 Step 1). A fold-specific evaluation of each trained candidate's predictive performance on the validation set is defined with respect to a risk function, or 'performance metric', such as the mean squared error (MSE). The CV performance of a trained candidate algorithm is its CV risk averaged over all validation sets.

Detailed descriptions outlining the SL procedure are widely available in the literature.[17,18] This paper focuses on practical advice on how to tailor the SL to robustify performance, based on characteristics of the data and the substantive goal. Key components are the library, V-fold CV scheme and the dSL's performance metric.

Guidelines for specifying a super learner

In this section, we provide step-by-step guidelines for tailoring the SL specification to perform as well as possible for the prediction task at hand. Our recommendations depend on the information available in the data and the prediction problem. The heuristics discussed here are accompanied by a flowchart (Figure 1). They are based on our accumulated experience and guided by the optimality theory of the SL. Introductions to the standard SL software (*SuperLearner* and *sl3* R packages), and examples, are freely available online.[19,20]

Preliminaries: Analytical dataset pre-processing

The analytical dataset consists of observations on an outcome (Y) and covariates (X). SL requires that this dataset does not contain any missing values. When cleaning the data, the analyst should consider omitting observations where the value of Y is an extreme outlier, to avoid unduly influencing the estimation of the prediction function. In high-dimensional settings, predictive performance is sometimes improved by reducing the number of covariates. In the pre-processing stage, only outcome-blind dimension reduction strategies may be considered. Examples include removing covariates that subject matter experts deem irrelevant; removing extremely sparse covariates; and, among a set of highly correlated covariates, removing one from a pair or creating a single summary measure from many. Covariate reduction strategies based on the outcome-covariate relationships in the data, such as using lasso,[21] should be incorporated within SL's CV procedure (see 'Screener couplings' in Flowchart Step 4 subsection).

Flowchart Step 1: Specify the performance metric for the discrete super learner

A performance metric, such as MSE, area under the receiver operating characteristic curve (AUC), or the mean negative log-likelihood loss, quantifies the success of an estimated prediction function (i.e. a trained algorithm). The chosen metric should: (i) align with the goals for the predictions in the real world; and (ii) be optimized (minimized or maximized) by the target prediction function. The latter guarantees that the trained algorithm's evaluation corresponds to its success in approximating the target. Rose's work on mortality risk score prediction provides a straightforward example, in which the target prediction function is defined first (conditional mean outcome, given covariates, in the target population), and then a function that is optimized by it (MSE) is selected as the performance metric.[3]

The flowchart (Figure 1) lists suitable metrics for different prediction tasks that are available in standard SL software. SL software vignettes describe how to modify the default performance metric, MSE.[19,20] Advanced users can define a custom risk function, as in Zheng *et al.*[4] so long as it is defined by a loss function that is uniformly bounded (easily achieved in practice by truncating predicted values to the range of Y).[2]

Flowchart Step 2: Derive the effective sample size

The amount that can be learned from a dataset depends on the complexity of the prediction task and on the amount of information in the data. When the sample size is larger,

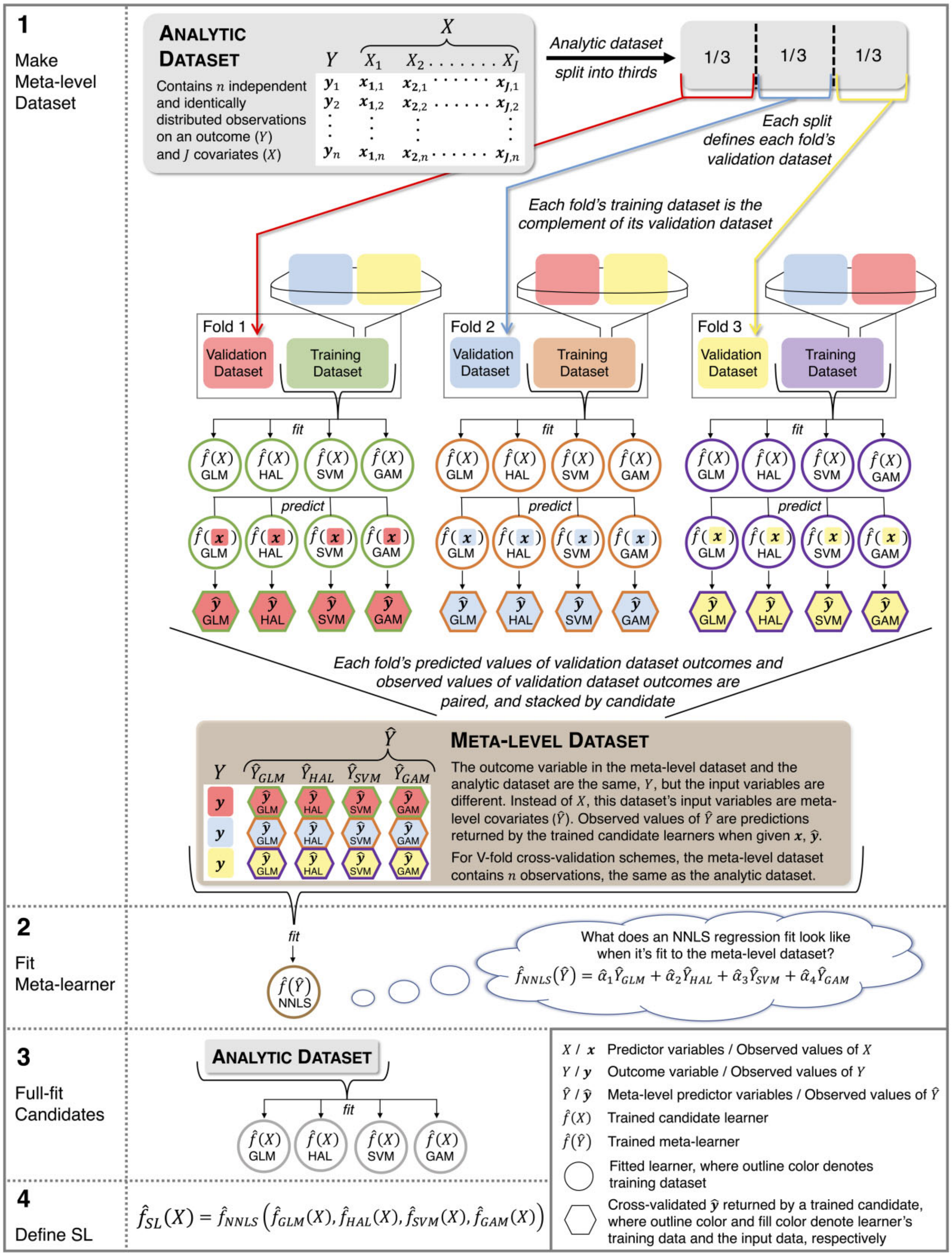
1
Make Meta-level Dataset
ANALYTIC DATASET
Contains n independent and identically distributed observations on an outcome (Y) and J covariates (X)
Y X_1 X_2 … X_J
y_1 $x_{1,1}$ $x_{2,1}$ … $x_{J,1}$
y_2 $x_{1,2}$ $x_{2,2}$ … $x_{J,2}$
y_n $x_{1,n}$ $x_{2,n}$ … $x_{J,n}$
Analytic dataset split into thirds
1/3 1/3 1/3
Each split defines each fold's validation dataset
Each fold's training dataset is the complement of its validation dataset
Fold 1 Fold 2 Fold 3
Validation Dataset
Training Dataset
fit
predict
$\hat{f}(X)$ GLM HAL SVM GAM
$\hat{y}$
Each fold's predicted values of validation dataset outcomes and observed values of validation dataset outcomes are paired, and stacked by candidate
META-LEVEL DATASET
Y $\hat{Y}_{GLM}$ $\hat{Y}_{HAL}$ $\hat{Y}_{SVM}$ $\hat{Y}_{GAM}$
The outcome variable in the meta-level dataset and the analytic dataset are the same, Y, but the input variables are different. Instead of X, this dataset's input variables are meta-level covariates ($\hat{Y}$). Observed values of $\hat{Y}$ are predictions returned by the trained candidate learners when given x, $\hat{y}$.
For V-fold cross-validation schemes, the meta-level dataset contains n observations, the same as the analytic dataset.
2
Fit Meta-learner
fit
$\hat{f}(\hat{Y})$ NNLS
What does an NNLS regression fit look like when it's fit to the meta-level dataset?
$\hat{f}_{NNLS}(\hat{Y}) = \hat{\alpha}_1\hat{Y}_{GLM} + \hat{\alpha}_2\hat{Y}_{HAL} + \hat{\alpha}_3\hat{Y}_{SVM} + \hat{\alpha}_4\hat{Y}_{GAM}$
3
Full-fit Candidates
ANALYTIC DATASET
fit
$\hat{f}(X)$ GLM HAL SVM GAM
4
Define SL
$\hat{f}_{SL}(X) = \hat{f}_{NNLS}\left(\hat{f}_{GLM}(X), \hat{f}_{HAL}(X), \hat{f}_{SVM}(X), \hat{f}_{GAM}(X)\right)$
X / x Predictor variables / Observed values of X
Y / y Outcome variable / Observed values of Y
$\hat{Y}$ / $\hat{y}$ Meta-level predictor variables / Observed values of $\hat{Y}$
$\hat{f}(X)$ Trained candidate learner
$\hat{f}(\hat{Y})$ Trained meta-learner
Fitted learner, where outline color denotes training dataset
Cross-validated $\hat{y}$ returned by a trained candidate, where outline color and fill color denote learner's training data and the input data, respectively

Figure 2 Illustration of the super learner (SL) with the following specification: V-fold cross-validation with three folds; library consisting of generalized linear model (GLM), highly adaptive lasso (HAL), support vector machine (SVM) and generalized additive model (GAM) candidates; and non-negative least squares (NNLS) regression meta-learner.

algorithms can typically learn more than when it is smaller. However, there can be a sparsity of information in the data, even at large sample sizes. For instance, if Y is a rare binary event, then the information content is limited by the size of the minority class. We introduce the effective sample size (n_{eff}) as a useful proxy for the information in the data; n_{eff} plays a role in heuristics for tailoring the SL's specifications to robustify performance.

If Y is continuous, we define $n_{eff} = n$, the number of independent and identically distributed (i.i.d.) observations in the dataset. If observations in the data are clustered, such as repeated measures on the same individual, then n is the number of independent clusters. If Y is binary, then the amount of information in the data is limited by n and also by the number of events or non-events, whichever is the minority (n_{rare}). In this case, we take the prevalence into account by defining $n_{eff} = min(n,\ 5 * n_{rare})$. The factor of five was chosen based on case-control study design literature, where investigations of statistical power show a sharp increase until a ratio of about 1:5 and then slow increases thereafter.[22] In the following, we discuss simple rules of thumb for how n_{eff} can affect the number V-fold CV folds and the choice of learners included in the library.

Flowchart Step 3: Define the V-fold cross-validation scheme

The CV predictive performance (CV risk) is an estimate of an algorithm's true out-of-sample predictive performance if it were trained on the entire analytical dataset (true risk). V-fold CV schemes assign every observation to a single validation set and to V-1 training sets, where V is the number of folds. The sample size in each validation set and training set is approximately n/V and $n - (n/\mathrm{V})$, respectively. Figure 2 depicts a V-fold CV scheme with three folds.

Selecting the number of folds

The number of folds, V, affects the bias and variance of the estimated CV risk.[23] This estimate more closely approximates the true risk when the distribution of the data in each training set is a faithful representation of the distribution in the full analytical dataset. Ensuring that the joint distribution of covariates in each training set reflects that in the full analytical dataset is accomplished by setting V large. For example, when V is 20, each training set contains 95% of the available observations. V typically ranges between 2 and 20, with larger values recommended for smaller n_{eff}.

As V increases from 2 to n, the sample size of each training set increases, so the algorithm's fits to each training set approach the full analytical dataset fit. This decreases bias in the CV risk estimate. Also, as V increases from 2 to n, the increase in overlap in the training sets increases the correlations among the algorithm's fits to the training data. This tends to increase the variance of the CV risk estimate, which is a mean of these correlated random variables. Theoretical optimality of the SL, which is applicable for large n, requires that V increases at a slower rate than n. Therefore, for large n, this rules out leave-one-out CV (LOOCV) and supports setting V large and less than n.[14–16]

We recommend choosing larger V in the SL while keeping computational feasibility in mind, as increasing V increases computation time. When n_{eff} is large, V can be reduced to improve computational feasibility, with little to no increased bias in the CV risk estimate. All of this considered, Step 3 of the flowchart recommends LOOCV for very small n_{eff}, and allows smaller values for V as n_{eff} grows.

Stratified cross-validation

When Y is binary, the prevalence of Y in each training set should match the overall prevalence. This can be achieved with stratified V-fold CV, an option in existing SL software for randomizing the assignment of observations to validation sets within strata of Y (see examples 4–6 in the main Supplement, available as Supplementary data at *IJE* online).

Cross-validation with clustered data

When the data are not i.i.d., clustered observations must be assigned as a group to the same validation and training sets. This ensures the validation data are completely independent of the training data, and the loss function is evaluated at the cluster level. V-fold CV with clustered data is specified with existing SL software by supplying a cluster identifier to the 'id' argument.

Flowchart Step 4: Form library of candidate learners

An ideal, rich library is diverse in its learning strategies, able to adapt to a range of underlying functional forms for the target prediction function, computationally feasible, and effective at handling high-dimensional data. Diverse libraries include parametric learners, non-parametric learners (when n_{eff} is not small), multiple variants of the same learner with different tuning parameter specifications, and learners with a range of screeners for dimension reduction when the data are high dimensional.

Base learners

Base learners are algorithms that are not fully specified but define a particular learning strategy, such as lasso[21] or random forest.[24] Many base learners are listed at the top right-hand corner of Step 4 in the flowchart (Figure 1). A base learner is a building block for constructing one or more fully specified learners having values provided for all tuning parameters.

Different types of base learners will fit the prediction function in different ways. For example, unlike a main-terms linear regression, a tree-based algorithm (e.g. random forest[24]) inherently models interactions and is unaffected by monotone data transformations. Since the underlying functional form of the target prediction function is unknown, it is a good idea to consider a variety of base learners and multiple variations of the same base learner with different tuning specifications; the latter can be done manually based on advanced knowledge, or automatically via the *caret* learners in standard SL software.[20,25] Reviews of different base learning strategies are widely available online and in the literature.[26,27] In the main Supplement (available as Supplementary data at *IJE* online), we provide seven examples of constructing SL libraries, all of which are accompanied by R code and discussion of each library's strengths and weaknesses.

There is no harm in including learners that perform poorly in the library, as they will be given a weight of zero by the dSL's meta-learner, and an eSL might use it in combination with predictions from other learners to produce a superior prediction function. This highlights the importance of specifying a rich and diverse library of learners. Also, since the SL theory holds for polynomial growth of the library with n, it is generally impossible to include too many learners in the library.[2] Downsizing the library might be warranted, however, to improve computational feasibility or when n_{eff} is small.

Screener couplings

Dimension reduction, or covariate screening, is essential when the dimensionality of the data is very large, and it can be practically useful in any SL or machine learning application. Screening of X that considers associations with Y must be incorporated within an algorithm's CV scheme to avoid biasing the CV estimate of predictive performance.[28] Any candidate learner can be paired with a screening function (screener) to establish a new candidate in the library. For example, consider pairing a lasso screener with a generalized linear model (GLM) candidate (e.g. logistic regression). In each CV fold, the lasso regression would be fit to the training data. Only the covariates with non-zero coefficients would be passed to the GLM, which would run a regression of Y on the reduced set of covariates in the fold's training data. Covariates retained in each CV fold may vary. In example 7 of the main Supplement (available as Supplementary data at *IJE* online), several learners were coupled with different types of screeners.

Smaller effective sample sizes

When n_{eff} is small, algorithms that have more structure and smooth over areas of little support in the data will likely perform better than more data-adaptive learners. Examples include parametric learners, like lasso, and lower-depth tree-based learners. Highly flexible and nonparametric learners, like neural nets with all but the simplest architecture, require more data to fit their parameters well; they are generally less appropriate for smaller n_{eff} due to the limited information in the data.

Risk functions

Algorithms that internally optimize risk functions that are not optimized by the target prediction function or are not well aligned with the prediction task may still be included in the library. For example, consider a binary classification prediction problem, where the dSL uses the AUC performance metric. The dSL's winner-take-all meta-learner will select the candidate with the highest CV AUC. The dSL's library of learners can of course contain algorithms whose objective is to optimize the AUC, but it can also contain algorithms that optimize other risk functions, such as the negative log-likelihood or MSE. In some binary classification problems, negative log-likelihood-minimizing algorithms can achieve higher AUC than AUC-maximizing algorithms.[29] This further motivates diversifying the library.

Respecting the statistical model

The statistical model is a set of probability distributions that could have given rise to the observations in the data.[7,14] It embodies knowledge regarding the data-generating process (DGP, the unknown, underlying probability distribution describing the data at hand) and is expressed by the analyst through their choice of algorithm(s) to fit to the data. If the DGP does not reside in the statistical model space, then the statistical model is misspecified, and this can lead to mild or severe bias in estimates and misleading results. Conveying knowledge about the DGP through the library can mitigate statistical model misspecification with the SL and avoid wasteful searching in irrelevant function spaces. For example, if it is known that there are interactions among covariates, then the analyst can include learners in the library that pick up on that explicitly (e.g. by including in the library a parametric regression learner with interactions specified in a formula) or implicitly (e.g. by including in the library tree-based algorithms that learn interactions empirically). To accommodate a range of possible underlying functional forms for the target prediction function, the library should be as rich and diverse as possible, with respect to n_{eff} and computational feasibility.

Step 5: Use the discrete super learner to select the candidate with the best cross-validated predictive performance

When the eSL is used, we recommend it be evaluated as another candidate in a dSL, as this allows for the eSL's V-fold CV risk to be compared with that of each individual learner considered in its library. If the eSL performs better than any other candidate, the dSL will end up selecting the eSL. Also under this approach, multiple eSLs that use more flexible meta-learner methods (e.g. non-parametric machine learning algorithms like highly adaptive lasso[30]) can be evaluated simultaneously.

Note that if the trained algorithm is going to be put into production to generate predictions on novel data and several learners offer near-optimal CV performance, additional considerations can contribute to deciding which algorithm to select. These include the level of interpretability, the number of covariates needed to evaluate the prediction, and the reliability, cost and ease with which the predictions can be accessed or generated.

Conclusion

In this work, we walked through each of the choices the analyst needs to make to define a SL that is specified according to the prediction task at hand and information in the data. We provided recommendations to assist the analyst in making these decisions, including the choice of learners included in the library, V-fold CV scheme, and the SL's meta-learner and performance metric. The flowchart provides a concise reference for guiding the analyst in how to effectively use SL in practice. For a variety of practical examples considered in the main Supplement (available as Supplementary data at *IJE* online), we construct SLs using the flowchart, discuss the quality of the SL specification and the results, and provide reproducible R code.

Ethics approval

Not applicable.

Data availability

A main file containing seven supplementary examples for constructing super learner libraries, all of which are accompanied by discussion of each library's strengths and weaknesses, and the datasets and reproducible R code for the Supplementary examples, are available as Supplementary data at *IJE* online.

Supplementary data

Supplementary data are available at *IJE* online.

Author contributions

R.V.P. drafted the article, figures and tables. As the principal investigator for the larger FDA-funded contract that supported this project, S.G. served as a driving force for this work. S.G. provided guidance on wording, formatting and organization throughout execution of this manuscript. M.J. v/d L. provided guidance on the super learning theory and its practical implications. Over dozens of meetings, R.V.P., M.J. v/d L., H.L. and S.G.'s accumulated experience was woven together by R.V.P. and S.G. to develop the main flowchart (Figure 1) and accompanying text on practical considerations for specifying a super learner. H.L. and S.G. helped minimize technical jargon and target the manuscript towards *IJE* readership. There were many iterations of this work, and M.J. v/d L., H.L. and S.G. critically revised each of them.

Funding

This work was supported by the United States Health and Human Services (HHS) Food and Drug Administration (FDA) (FDA Contract 75F40119C10155). The content is the view of the authors, and does not necessarily represent the official views of, nor an endorsement by, FDA/HHS, or the United States Government.

Conflict of interest

None declared.

References

1. Pirracchio R, Petersen ML, Carone M, Rigon MR, Chevret S, van der Laan MJ. Mortality prediction in intensive care units with the Super ICU Learner Algorithm (SICULA): a population-based study. *Lancet Respir Med* 2015;**3**:42–52.
2. van der Laan MJ, Polley EC, Hubbard AE. Super learner. *Stat Appl Genet Mol Biol* 2007;**6**:25.
3. Rose S. Mortality risk score prediction in an elderly population using machine learning. *Am J Epidemiol* 2013;**177**:443–52.
4. Zheng W, Balzer L, van der Laan M, Petersen M; SEARCH Collaboration. Constrained binary classification using ensemble learning: an application to cost-efficient targeted PrEP strategies. *Stat Med* 2018;**37**:261–79.
5. Pirracchio R, Petersen ML, van der Laan M. Improving propensity score estimators' robustness to model misspecification using super learner. *Am J Epidemiol* 2015;**181**:108–19.
6. Pirracchio R, Carone M. The balance super learner: a robust adaptation of the super learner to improve estimation of the average treatment effect in the treated based on propensity score matching. *Stat Methods Med Res* 2018;**27**:2504–18.
7. van der Laan MJ, Rose S. *Targeted Learning: Causal Inference for Observational and Experimental Data*. New York, NY: Springer, 2011.
8. R Core Team. *R: A Language and Environment for Statistical Computing*. 2022. https://R-project.org/ (9 November 2022, date last accessed).
9. Polley E, LeDell E, van der Laan M. *SuperLearner: Super Learner Prediction*. 2021. https://CRAN.R-project.org/package=SuperLearner (9 November 2022, date last accessed).

10. Coyle J, Hejazi N, Malenica I, Phillips RV, Sofrygin O. *sl3: Pipelines for Machine Learning and Super Learning*. 2022. https://github.com/tlverse/sl3 (9 November 2022, date last accessed).
11. Efron B, Morris C. Combining possibly related estimation problems. *J R Stat Soc Series B Stat Methodol* 1973;**35**:379–402.
12. Wolpert DH. Stacked generalization. *Neural Networks* 1992;**5**: 241–59.
13. Breiman L. Stacked regressions. *Mach Learn* 1996;**24**:49–64.
14. van der Laan MJ, Dudoit S. Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples. In: *U.C. Berkeley Division of Biostatistics Working Paper Series*. Working Paper 130. https://www.bepress.com/ucbbiostat/paper130 (9 November 2022, date last accessed).
15. Dudoit S, van der Laan MJ. Asymptotics of cross-validated risk estimation in estimator selection and performance assessment. *Stat Methodol* 2005;**2**:131–54.
16. van der Vaart AW, Dudoit S, van der Laan MJ. Oracle inequalities for multi-fold cross validation. *Stat Decis* 2006;**24**:351–71.
17. Naimi AI, Balzer LB. Stacked generalization: an introduction to super learning. *Eur J Epidemiol* 2018;**33**:459–64.
18. Polley EC, van der Laan MJ. Super learner in prediction. In: *U.C. Berkeley Division of Biostatistics Working Paper Series*. Working Paper 266. https://www.bepress.com/ucbbiostat/paper266 (9 November 2022, date last accessed).
19. Kennedy C. *Guide to SuperLearner*. 2017. https://cran.r-project.org/web/packages/SuperLearner/vignettes/Guide-to-SuperLearner.html (9 November 2022, date last accessed).
20. Phillips RV. *Chapter 6 Super Learning. Targeted Learning in R*. https://tlverse.org/tlverse-handbook/sl3.html (9 November 2022, date last accessed).
21. Tibshirani R. Regression shrinkage and selection via the lasso. *J R Stat Soc Series B Stat Methodol* 1996;**58**:267–88.
22. Woodward M. *Epidemiology: Study Design and Data Analysis*. Boca Raton, FL: CRC Press, 2013.
23. Arlot S, Celisse A. A survey of cross-validation procedures for model selection. *Stat Surv* 2010;**4**:40–79.
24. Breiman L. Random forests. *Mach Learn* 2001;**45**:5–32.
25. Kuhn M. Building predictive models in R using the caret package. *J Stat Softw* 2008;**28**:1–26.
26. Brownlee J. *A Tour of Machine Learning Algorithms. Machine Learning Mastery*. 2019. https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/ (9 November 2022, date last accessed).
27. Singh A, Thakur N, Sharma A. A review of supervised machine learning algorithms. In: *International Conference on Computing for Sustainable Global Development (INDIACom)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, 2016, pp. 1310–15.
28. Varma S, Simon R. Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics* 2006; **7**:1–8.
29. LeDell E, van der Laan MJ, Petersen M. AUC-maximizing ensembles through metalearning. *Int J Biostat* 2016;**12**:203–18.
30. Benkeser D, van der Laan M. The highly adaptive lasso estimator. In: *Proceedings of the International Conference on Data Science and Advanced Analytics. 17–19 October* 2016, Montreal, Canada. IEEE 2016, pp. 689–96.