Ch.57 Machine Learning in Political Science: Supervised Learning Models

in The SAGE Handbook of Research Methods in Political Science and International Relations

Santiago Olivella
Kelsey Shoub

May 24, 2019
Word Count: 5983

1 Introduction

The set of theoretical and computational approaches used under the rubric of "machine learning" (ML) is so diverse that it is easy to think of the field as a kind of catch-all, interdisciplinary exercise at the intersection between statistics, computer science and other affiliated disciplines. Such a perspective, however, would obscure the fact that these seemingly disparate approaches share a common goal: to improve a computer's performance on a given (typically predictive) task by identifying empirical relationships, patterns and trends in data that rely on minimal distributional and functional form assumptions on the part of analysts (Hastie, Tibshirani and Friedman, 2009; Jordan and Mitchell, 2015). This goal, which is what *learning* is typically taken to mean in this context, happens to be shared by a number of disciplines — including, of course, Political Science and International Relations.

Indeed, our discipline's approach to the field of ML has resulted not only in careful applications of tools devised in other domains, but also in the development of tools designed to address problems of specific disciplinary interest — such as the fast estimation of ideal points that easily scales to millions of subjects (Imai, Lo and Olmsted, 2016), the structural analysis of text-as-data with large corpora (Roberts et al., 2014), and the discovery of subgroup causal effects in contexts with relatively few unknown confounders in high-dimensional datasets (Ratkovic and Tingley, 2017).

The goal of this chapter is twofold: to present some of the most commonly used tools in the field of predictive ML, and to illustrate how these tools have been adopted (and often adapted) to answer questions of interest in Political Science. We focus on *supervised models*, where other contributions to the Handbook (e.g. Chapter 58 on Natural Language Processing and AI) offer insights into other approaches within ML and computational social sciences. Supervised models are those where both an outcome of interest *and* relevant predictors are available for a set observations and can be used to learn the patterns of their association, so as to better predict instances in which the outcome is *not* observed.

We begin the chapter with a simple taxonomy of ML tools using the supervised/unsupervised distinction, and offer a brief overview of some of the most important concepts in the field. We then discuss the main ideas behind three major supervised learning approaches, namely tree-based methods, and kernel-based approaches and support vector machines. Although we provide enough mathematical detail to understand the formal underpinnings of each approach, we emphasize the intuitions behind these models (rather than their theoretical foundations) to make the materials accessible to a wide audience. Finally, we conclude the chapter with some thoughts on the promise and perils of ML modeling, and offer some suggestions for additional readings.

2 Types of Machine Learning Models and Relevant Concepts

As is the case in many other social sciences, quantitative data analysis in our disciplines has typically relied on some variant of the linear model. The modal empirical exercise defines a specific functional relationship between the conditional expectation of an outcome of interest and a limited set of predictors suggested by theory, tradition, or a combination of both. Although useful in its parsimony and intelligibility, this common approach has a number of limitations.

For instance, classical regression models require positing a particular specification for the conditional expectation function at the onset of analysis. Even if a few different specifications are considered, choosing among them after estimation results have been observed can at best offer proof that there is *one* way in which the world is consistent with proposed hypotheses — not a very strong test of a hypothesis. In the absence of real competition from the myriad alternative plausible models, the typical approach offers few meaningful chances for the researcher to be proved wrong (Ho et al., 2007). Similarly, classical regression models are unable to handle situations in which the number of included predictors is far larger than the number of available observations, as is common in text-as-data applications discussed elsewhere in this volume. By increasing modeling flexibility while actively avoiding tracking sample idiosyncrasies too closely, ML models offer viable steps toward resolving these and many other issues raised by classical approaches.

Despite being unified by the same goals and general flexibility, there is a daunting variety of modeling approaches that fall under the purview of ML. Different taxonomies of models exist based on a variety of criteria. For instance, we can distinguish between **generative** models (which define a probability model for the joint distribution of outcomes and predictors) and **discriminative** models (which either define a probability model for the conditional distribution of outcomes given predictors, or otherwise propose no distributional assumptions whatsoever) (Bishop, 2016; Ng and Jordan, 2002). Examples of the former include the *latent Dirichlet allocation* and the *hidden Markov model*, while examples of the latter include models like *support vector machines* and *regression trees* (as well as less exotic models, like the logistic regression).

Perhaps the most commonly used dimension along which ML models tend to be classified, however, pertains the relevance of a target outcome in the learning task. If the goal is to learn structure in a set of inputs (or **features**) without appealing to a target outcome then we are said to face a **unsupervised learning** (or descriptive) problem (Murphy and Bach, 2012). Examples of such tasks include problems such as the organization of unstructured text into topics, and the estimation of latent traits (such as ideology) based on observed behaviors (such as roll-call votes or decisions to follow actors embedded in a social network).

On the other hand, when both features *and* outcomes are observed for some set of observations — typically called the **training set** — we are said to have a **supervised learning** problem. Examples of such tasks include classical problems such as email spam detection, hand-written digit recognition and object-detection in images. They also include the problems of learning a response surface defined over an input space, or (equivalently) of understanding how an outcome is related to inputs in potentially complex ways (Hastie, Tibshirani and Friedman, 2009). Although many interesting ML developments occur in the realm of unsupervised learning (as evidenced, for instance, by the discussions in Chapters 30, 31, 32, 50 and 58 of this Handbook), we focus on supervised learning techniques in this chapter, as they encompass the kinds of predictive tasks most commonly associated with ML.

Within supervised learning models, we can further distinguish between models based on the measurement type of the target outcome. If the outcome is continuous, learning tasks are referred to as **regression** problems. In turn, nominal outcomes (regardless of whether they have two or more categories) give rise to **classification** problems (Bishop, 2016; Murphy and Bach, 2012).[1] Regardless of whether they are regression or classification problems, however, supervised learning models share the same goal: learn the potentially complicated relationships that relate (combinations of) features $\mathbf{x}$ to the outcome of interest y *in general*, using information available in the set of observations for which the pair $(\mathbf{x}, y)$ is fully observed.

The *in general* qualification is an important one, however, as it is typically easy to learn even complicated relationships **in-sample** — that is, relationships that are conditional on the training set. The goal, however, is to learn relationships for which expected **generalization error** (i.e. the error that can be expected to ensue when learned relationships are evaluated *out-of-sample*, on a random **test set** of observations not involved in the learning process) is low (Hastie, Tibshirani and Friedman, 2009). In fact, while it is always possible to arbitrarily reduce training error (i.e. error as computed using the training sample) by making models arbitrarily complex, such flexibility typically results in high expected generalization error, as models start to **overfit** their training data (i.e. they start to pick up on idiosyncratic relationships that conditional on the set of observations used to train the models).[2]

Accordingly, and since the ultimate goal of supervised learning is to find *generalizable* patterns of association, models are typically subject to some from of **regularization** — typically in the form of a constraint that pushes the model toward parsimony — and are selected based on their ability to generate good out-of-samples predictions. Clearly, it is impossible to evaluate a model's performance on the universe of unsampled test instances, so an approximate measure of performance must be devised. Although several approaches are viable, none is more commonly used than **cross-validation** (CV) — the exercise of further splitting the training data into a training set and a **validation set** (used to evaluate predictive accuracy, but omitted from the learning phase). To

[1] Unsupervised models can be similarly classified based on the nature of the learning output: tasks that result in continuous outputs are typically called *dimensionality reduction* problems (and typically involve some variant of the factor-analytic model), whereas tasks that result in discrete outputs are typically called *clustering* problems.

[2] This is also an example of the so-called **bias-variance trade-off**, which suggests that there is an optimal level of bias that can be achieved such that only a small price is paid in terms of variance of the predictor. Generalization error, which can be thought of as a function of both bias *and* variance (in addition to fundamental uncertainty) of the estimator implied by the model, is therefore at a minimum when this balance is achieved (Hastie, Tibshirani and Friedman, 2009).

further minimize issues related to *bad draws*, multiple such splits are typically conducted.

The most popular approach to cross-validation, ***k*-fold** CV, partitions the training data into k subsets, estimates the model leaving each subset aside, evaluates predictive accuracy on the held-out set of observations, and approximates the generalization error of the specific model by taking the average of these held-out errors (Hastie, Tibshirani and Friedman, 2009). When models require the definition of so-called **tuning (hyper-)parameters** (i.e. ancillary parameters that govern the model's behavior, such as the number of topics in a topic model), multiple values are evaluated using this iterative estimation process, allowing researchers to base their value definitions on a data-driven procedure.

Overall, the typical workflow in ML involves, first (and crucially) representing raw data using quantitative features (e.g. transforming unstructured texts into a document feature matrix), choosing an appropriate model for the learning task at hand, tuning the model's hyper-parameters using some form of cross-validation, and evaluating the model's out-of-sample predictive accuracy (i.e. its expected generalization error) using an entirely separate test set.

To give a flavor of what some of these decisions look like, we now turn to a discussion of two of the most commonly used sets of models in supervised learning within Political Science: tree-based approaches and support vector machines.

3 Examples of supervised learning models

3.1 Tree-based approaches: CART, Random Forests and Tree Boosting

For a single-tree model, the goal is to partition the space of predictor features (i.e. the set of all unique combinations of predictor values) into contiguous regions within which prediction is easier, thus improving overall predictive accuracy by sorting observations into their respective bins. Intuitively, this goal is best achieved if the regions are defined by their degree of homogeneity with respect to the outcome of interest, so that region-specific predicted values are as close to the target as possible.

To clarify these abstract ideas, consider the task of predicting turnout based on age and education levels. The left panel of Figure 1 shows our feature space (defined by unique values of the two predictor features) as well as a hypothetical target turnout distribution, with darker shades indi-

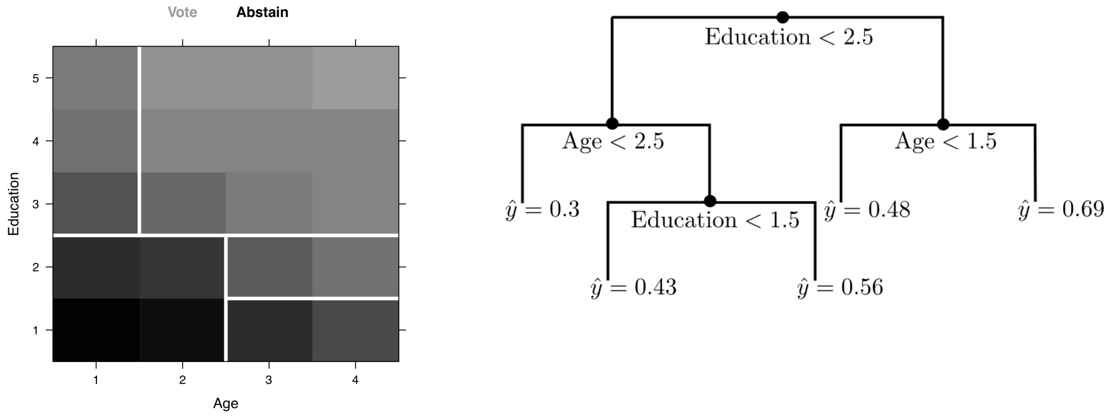

Figure 1: **Left Panel**: Feature space comprised of ages and education levels, partitioned into regions that are relatively homogeneous with respect to probability of turnout (indicated with lighter shades of gray for every combination of feature values). **Right Panel**: Binary-tree representation of feature-space partition depicted on the left panel. Values at terminal nodes indicate predicted turnout probability in each corresponding region.

cating higher turnout rates. In turn, the center panel shows a stratification of all observations into regions for which a simple, constant prediction model (e.g. using the average turnout rate for all observations within the region) offers a good approximation of the underlying turnout distribution. When we restrict regions to form non-overlapping "boxes" that are aligned with the coordinate axes of the feature space (i.e. such that their edges are parallel to said axes), the partition can be represented by a recursive tree — hence giving these models their name. The right panel of Figure 1 shows the recursive tree that corresponds to the partition depicted in the central panel, and is the result of fitting a *classification and regression tree* (or CART; see Breiman et al., 1984) model to the hypothetical data.

More formally, we can define the predictive task as one in which we need to learn parameters Θ in a function $f(\mathbf{x}_i|\Theta) = T(\mathbf{x}_i|\Theta) = \sum_{b=1}^{B} \hat{y}_b \mathbf{1}(x_i \in R_b)$, where R_b denotes the b_{th} region (out of B total regions) in the partition of the feature space, $\hat{y}_b$ is the constant prediction for all observations in that region (typically the average or the modal category for observations in the region), and $\mathbf{1}(\cdot)$ is the indicator function. The Θ parameters control both the variables for creating each split in the recursive tree, as well as the value at which each split occurs.

Finding the best Θ for any given loss function $L(y_i, f(\mathbf{x}_i))$ is prohibitively costly, as it requires solving an extremely hard discrete optimization problem. Instead, a simple greedy heuristic for finding a local optimum consists of sequentially picking a predictor-value pair p, v among the set of all predictors P that minimizes prediction loss among the induced regions. This approach, known as *recursive binary splitting*, forms the tree through a sequence of locally optimal recursive binary splits that starts with the entire feature space at its root[3] and ends when a given stopping criterion is reached (for instance, fewer than a pre-specified number of observations would fall into a new region).

Although the final model has many advantages (e.g. it is easy to interpret and visualize, it can capture complex interactions between predictors when grown to be deep enough, it performs a kind of automatic feature selection by choosing locally predictive variables, and it can be easily adjusted to accommodate values that are missing[4] or that are measured in different scales), it is also affected by serious limitations. First, growing a single, deep tree using binary recursive splitting can result in a grossly overfit model. In turn, and as an example of the common bias-variance trade-off, this high level of in-sample predictive accuracy usually comes at the expense of high estimator variance, as single trees grown recursively can often times yield wildly different predictions as a result of small changes in the training set. In addition, the constraint of creating regions that are axis-parallel makes it very hard for a single-tree models to accommodate additive relationships, there are no natural measures of prediction uncertainty, and variable choice in recursive binary splitting is biased toward choosing variables with many potential splitting points.

While there are strategies designed to ameliorate some of these issues,[5] the most common approach consists of building **ensembles** of trees. Formally, a tree-ensemble with M trees — grown in a slightly different way to induce variety and learn relationships using a "wisdom-of-

[3]non-binary splits can always be represented as a sequence of binary splits, and as a result most tree-based algorithms rely on the latter.

[4]Typically, missing predictors in the test-set are handled using *surrogate splits* — splits based on non-missing variables that result in similar reductions in loss as the original splitting feature.

[5]Cost-complexity pruning, for instance, was originally proposed by Breiman et al. (1984) to reduce the likelihood of overfitting, and consists of going over the internal nodes of a deeply grown tree and sequentially collapsing those that reduce loss the least. It has been shown that this is equivalent to finding a sub-tree that minimizes a penalized loss term, where the penalty term λB is linear on the number of terminal nodes B, and the choice of the strength of penalization λ is typically chosen via cross-validation. Yet another alternative is given by *conditional inference trees*, proposed by Hothorn, Hornik and Zeileis (2006), which choose splitting variables and values using a non-parametric significance test of association between predictors and the outcome of interest, thus reducing feature selection bias and focusing on statistically predictive variables.

crowds" approach — can defined by

$$f(\mathbf{x}_i|\mathbf{\Theta}) = \frac{1}{M}\sum_{m}^{M} T(\mathbf{x}_i|\Theta_m)$$

with the goal of learning optimal parameters

$$\tilde{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\operatorname{argmin}} \sum_i L\big(y_i, f(\mathbf{x}_i|\mathbf{\Theta})\big) \tag{1}$$

with strategies used to induce differences across trees resulting in a variety flavors of tree-ensemble methods. Although there is a cost to be paid in terms of interpretability — for example, making a prediction is no longer as easy as "dropping" an observation down the binary tree, and following it until we hit a terminal region with a corresponding predicted value — the gains in predictive accuracy and reduction of generalization error offer substantial advantages. We now present two of the most commonly used ensemble models (viz. random forests and gradient boosting machines).

3.1.1 Bagging and Random Forests

The first approach to building an ensemble of trees consists bootstrapping samples from the training set, fitting a single tree to each resampled set, and then aggregating predictions made by each such tree by taking their average, for instance. Adequately named *bagging* (a portmanteau for *b*ootstrap *agg*regat*ing*), this approach can help accommodate non-interactive relationships (by building trees with branches that are functions of a single predictor, for instance) as well as non-linear ones. However, the approach remains unsuccessful in terms of reducing estimator variance (and thus improving overall generalization error) unless an additional step is taken to de-correlate the trees that form the ensemble.

A simple way to reduce variance by way of reducing correlation among trees is to restrict the choice of each splitting variable to a random subset of predictors $p \subset P$, so that each bagged tree provides a truly different "perspective" on the prediction problem. In fact, it can be shown that correlation $\rho(\mathbf{x})$ declines as the relative size of p (i.e. the number of predictors used at any given splitting point) decreases Hastie, Tibshirani and Friedman (2009). This model, known as a *random forest* (or RFs; Breiman, 2001), is one of the most popular machine learning approaches used in

Political Science, with simple to use implementations in open-source software (e.g. `randomForest` in **R**) that require cross-validation of only a handful of parameters (including the ensemble size M and the size of the random subset of predictors at each splitting point $|p|$). And because trees can be grown in parallel, the approach has the potential to scale relatively well in the presence of large ensembles and datasets.

In addition to improving predictive accuracy by reducing variance and maintaining each tree's low bias properties, as well as increasing the set of functional associations the model can represent and performing the same kind of on-the-fly feature selection performed by single-tree approaches, RFs can easily provide estimates of generalization error by generating *out-of-bag* (i.e. out-of-sample) predictions for observations *not* in the bootstrap sample for each tree. Furthermore, measures of uncertainty can be readily produced using bias-corrected versions of the infinitesimal jackknife (Efron and Hastie, 2016; Wager, Hastie and Efron, 2014) and no additional computational costs.

3.1.2 Gradient Tree-Boosting

An alternative approach to building ensembles of trees is known as *boosting* (Schapire, 1990; Freund and Schapire, 1997). Boosting is an example of *forward stagewise additive modeling* — a procedure that tackles a complex optimization problem involving a sum of basis functions (such as 1) *sequentially*, in M steps. At each step, the parameters involved in the basis functions (in this case, trees) fit up until that point are left unchanged, thus reducing the complexity of the optimization problem by solving for the optimal parameters of a single basis function at each stage. As such, program in Equation 1 becomes

$$\tilde{\Theta}_m = \underset{\Theta_m}{\operatorname{argmin}} \sum_i L\big(y_i, f_{m-1}(\mathbf{x}_i) + \nu T_m(\mathbf{x}_i|\Theta_m)\big),$$

where $0 < \nu \leq 1$ is a step size that controls the learning rate, $f_{m-1}(\mathbf{x}_i)$ is the prediction of the tree ensemble fit in the first $m-1$ stages. What distinguishes boosting from other stagewise approaches is its focus on repeatedly modified versions of the training set, effectively transformed to gradually shift focus, at each stage, to observations that have been poorly fit by the ensemble up until that point.

A general formulation of this approach is given by *gradient boosting machines* (GBMs), which fits a tree $T_m(\mathbf{x}_i|\Theta_m)$ to the negative gradient of the loss function with respect to $f(\mathbf{x}_i)$ evaluated at $f_{m-1}(\mathbf{x}_i)$ at each stage of the sequential ensemble construction. In doing so, the model progressively concentrates on some measure of difference between predictions made up to the $m-1$ previous iterations and observed values, which is precisely the information contained in the negative gradient. For example, with (half) squared error loss, the i_{th} component of the negative gradient is given by

$$\frac{\partial}{\partial f(\mathbf{x}_i)}\frac{1}{2}[y_i - f(\mathbf{x}_i)]^2\Big|_{f_{m-1}(\mathbf{x}_i)} = y_i - f_{m-1}(\mathbf{x}_i)$$

which is the model's residual up to the $m-1$ step. Other differentiable loss functions make it possible to use gradient tree-boosting for a variety of problems classification and regression problems. The AdaBoost.M1 algorithm (Freund and Schapire, 1997), for instance, is a commonly used implementation of gradient boosting for binary classification using an exponential loss function and predictions in the set $\{-1, 1\}$.

Although not parallelizable like RFs (as they must be learned sequentially), GBMs are typically very fast to estimate. Tuning them requires defining values for the number of trees in the ensemble M, for the step size ν and for the tree depth of each ensemble member. In practice, all trees are usually grown on training sets to be "weak" learners (i.e. shallow trees, with depth defined primarily by theoretical order of anticipated interactions among predictors), ν is set to some small number (e.g. $\nu = 0.001$), and M is chosen by K-fold cross-validation from a fine grid (e.g. $M \in \{2, 3, \ldots, 5000\}$). Overall, gradient tree-boosted models have been found to have excellent predictive performance, prompting some to call them "the best off-the-shelf classifiers in the world" at one point in time (Breiman et al., 1998; Hastie, Tibshirani and Friedman, 2009). Good implementations in **R** include the `gbm` package (which can be cross-validated using `caret`) and `h2o.gbm` in the `h2o` package. Performance can usually be enhanced even further by incorporating some of the ideas behind RFs, such as using data subsampling at each stage of the ensemble creation, resulting in a variant of GBM called *stochastic gradient boosting* (Friedman, 2002), also implemented in `gbm` in **R**.

3.1.3 An application to small-group preference estimation

In addition to their use in forecasting tasks (e.g. Muchlinski et al., 2016; Kaufman, Kraft and Sen, 2018), tree-based models have commonly been used in the social and political sciences to study interactive effects and other types of conditional associations. In the study of causal relationships, for instance, tree-ensembles have been used to identify heterogeneous treatment effects (e.g. Imai and Strauss, 2011; Green and Kern, 2012; Wager and Athey, 2017). Their ability to identify complex functional forms without the need for researcher-defined specifications makes tree-based models ideal for another task: estimating preferences among small target populations using post-stratification of estimates obtained from non-representative samples.

In a survey of potential applications of tree-based models within Political Science, Montgomery and Olivella (2018) are able to reproduce and efficiently scale-up the exercise conducted by Ghitza and Gelman (2013). In their study, Ghitza and Gelman aim to estimate vote intentions of small groups of voters during the 2008 presidential election, as well their likelihood of turning out to vote. The groups, defined by intersections of geographic and socio-demographic characteristics (e.g. High-school educated Latino women between 18 and 25 who live in North Carolina), were assumed to have preferences that depended on the *combination* of these characteristics, thus requiring models that allowed for "deep interactions" on their right-hand sides.

The two-step approach they propose — which involves a predictive stage and a post-stratification stage, known as multi-level regression and post-stratification, or MRP — uses a random-intercepts model to model preferences as a function of these interactions, and post-stratifies predictions based on this model using highly granular frequency counts (e.g. census tables at the block level) that are then aggregated to whatever level is desired. Although interested in combinations of a large number of socio-demographic characteristics, Ghitza and Gelman's exercise is restricted by the computational limits imposed by the estimation of a large number of random effects involved in fully interactive specification of the first stage model.

To address this, Montgomery and Olivella replace the first stage model with a tree-based ensemble (viz. Bayesian Additive Regression Trees, or BART), which effectively only incorporates interactive effects when the training data support them, thus reducing the computational cost of the estimation without sacrificing potentially relevant model complexity. The results of both ex-

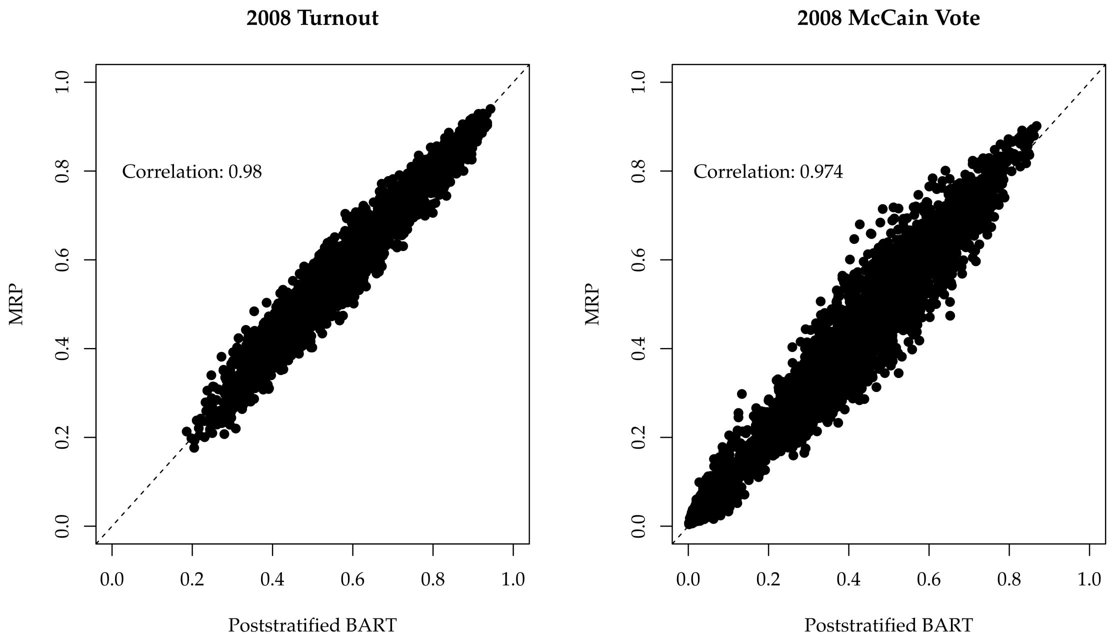

Figure 2: **Left Panel**: Post-stratified predictions of turnout during 2008 presidential election in the US, at low levels of demographic aggregation. Predictions produced by a multilevel model along the y-axis, and predictions along the x-axis produced by a Bayesian additive regression tree model. **Right Panel**: Post-stratified predictions of vote intention for McCain during 2008 presidential election in the US, at low levels of demographic aggregation. Predictions produced by a multilevel model along the y-axis, and predictions along the x-axis produced by a Bayesian additive regression tree model.

ercises are virtually identical when estimated using the same set of four predictors (viz. state, ethnicity, income, and age), as evidenced by Figure 2 (which presents the set of post-stratified MRP estimates vs. the post-stratified tree-based estimates). Contrary to the multilevel model, however, the tree-based approach can easily incorporate the full set of available predictors (a set of 8 discrete variables with a total set of 163,200 fully interacted categories) with very little additional computational overhead.

3.2 Kernel methods: Support Vector Machines

Yet another approach to the supervised learning problem is offered by support vector machines (SVMs), which have been most successfully used to solve classification problems — particularly when the number of predictive features is much larger than the number of observations n, or $p \gg n$. Examples of classes could be party affiliation, vote choice, sentiment, or whether or not a respondent received the experimental treatment. Like many tree-based models, they are "one of the 'must-have' tools in any machine learning toolbox"(Efron and Hastie, 2016, p. 387).

Intuitively, an SVM finds a hyperplane that maximizes the distance between it and the nearest instances of each class (the **support vectors**), thus producing the "cleanest" possible sorting of observations. This distance to the nearest instances, called the **margin**, generates a kind of buffer between types of observations; the optimization objective behind SVMs is to maximize the width of this buffer. To make these ideas more concrete, take an example where a researcher wants to predict whether instances are members of class A or class B. To do so, the researcher has two variables: $X1$ and $X2$. When fitting an SVM to predict which class each observation belongs to, the SVM finds the maximum-margin plane that separates these two classes while maximizing the distance between the plane and the nearest class instances.

Graphically, this is depicted on the left panel of Figure 3. In the figure, instances belonging to class A are shown as light-gray circles, and instances belonging to class B are shown as dark-gray triangles. The maximum-margin hyperplane is the solid line separating the two groups, and the margin (which touches the support vectors) is depicted using dashed lines equidistant from the hyperplane. As can be seen, all of the circles lie below the hyperplane (a line, in this case), while all of the triangles lie above the hyperplane: classes are perfectly linearly separable. Fitting an SVM (or, in this simple case, a *maximum margin classifier*) amounts to finding this separating

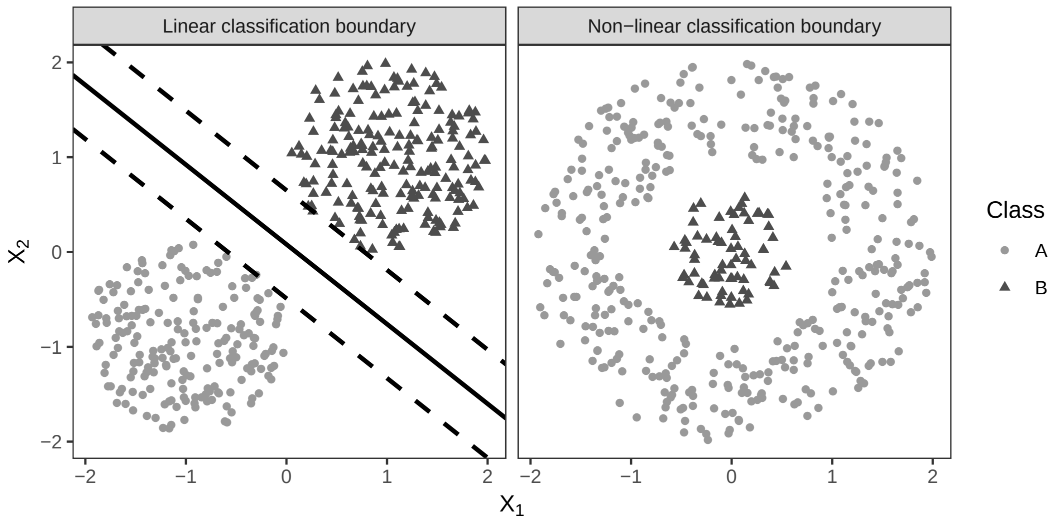

Figure 3: **Left Panel**: Classifying by a Maximum-Margin Hyperplane: a linearly-separable problem, with the corresponding separating line (solid line) and classification margins (dashed lines). **Right Panel**: Non-linearly separable classification problem of Class A (outer light gray triangles) and Class B (inner dark gray circles) instances.

line. Note that an important implication of this goal is that only those instances near the class boundary play a big role its definition, while those that remain far away from the boundary have little effect on its location and direction.

More formally, assume that a given data set consists of N instances represented by a set of features. In Figure 3, there are 40 instances, and the features used are the variables X1 and X2, where $x_{1i} \, \epsilon \, \mathbb{R}$ and $x_{2i} \, \epsilon \, \mathbb{R}$. Additionally, the target classes can be either -1, or $y_i \, \epsilon \, \{-1, 1\}$. A hyperplane is defined by:

$$x: \; f(x) \; = \; x^T\beta \; + \; \beta_0 \; = \; 0 \tag{2}$$

where β is a unit-length vector. Such a plane is separating if all instances of a class lie above it, and all instances of the other class lie below it (or, equivalently, if $y_i f(x) > 0$ for all training instances. Thus, a classification rule based on the separating plane would be $G(x) \; = \; \text{sign}[f(x)]$. Finally, note that (given $||\beta|| = 1$) the distance between any observation i and the separating plane is given by $y_i(x_i^T\beta + \beta_0)$. Thus, and for a margin M and set of support vectors $\mathcal{S}$, finding the maximum margin classifier amounts to finding

$$\underset{\beta,\beta_0}{\text{argmax}}\, M \text{ subject to } y_i(x_i^T\beta + \beta_0) \geq M \quad \forall i$$

To do so, the relevant Lagrangian dual objective function is maximized, and the solution for β is found to be:

$$\hat{\beta} \;=\; \sum_{i=1}^{N} \alpha_i y_i x_i \;=\; \sum_{i \in \mathcal{S}} \alpha_i y_i x_i, \tag{3}$$

where α_i are the Lagrange multipliers, and the second summation highlights the fact that only those observations in the support set affect the separating hyperplane.

3.2.1 The soft-margin classifier

So far, we have assumed that a hyperplane can perfectly separate instances across classes. When this is not the case, we must relax the constraint imposed on the distances between points and the hyperplane, and allow for a certain amount of *slack*. This slack will allow for instances to be within the margin, or even to cross the (quasi-)separating hyperplane.

Thus, although the objective of the optimization stays the same (i.e. maximize the margin), the **soft-margin** constraint is given by $y_i(x_i^T \beta + \beta_0) \geq M(1 - \epsilon_i)$, where the ϵ_i are the slack terms, themselves constrained so that that $\sum_i \epsilon_i \leq C$. The dual objective is now maximized subject to $0 \leq \alpha_i \leq C$ and $\Sigma_{i=1}^{N} \alpha_i y_i = 0$. As a result, C becomes a tuning parameter.

Specifically, the margin around the identified hyperplane(s) is larger for smaller values of C (e.g. $C = 0.01$) and is smaller for larger values of C (ex. $C = 1000$). Larger values of C thus result in greater focus of attention on the points located very close to the decision boundary, while smaller values involve data points farther away. It is these points (which effectively have Lagrange multipliers greater than zero) that now become the support vectors.[6]

Typically, the value for C is defined through cross-validation; when identifying *linear* boundaries, however, the results tend not to be too sensitive to the specification of C (Hastie, Tibshirani and Friedman, 2009). The choice matters considerably more when linear boundaries no longer generate optimal classifiers, as smaller values of C can result in severe over-fitting. The generalization of these ideas to non-linear decision boundaries is what is typically called a support vector *machine*.

[6]Incidentally, this is yet another manifestation of the bias-variance tradeoff: larger C values result in a low-bias/high-variance estimator, whereas smaller C values (and consequently a higher number of support vectors) results in low-variance/high bias estimators.

3.2.2 Support Vector Machines and Kernels

It is often the case that a non-linear classification boundary is needed in order to correctly classify instances. For instance, consider the right panel of Figure 3. Although the two classes are easily recognized as occupying different regions of feature space, no hyperplane across it would result in a good separation. The optimal decision boundary, which in this case corresponds to a circle, is not linear.

In the classical regression context, dealing with non-linear associations involved incorporating non-linear functions of predictors (e.g. higher order polynomials, log-transformations, etc.) into the specification of the conditional expectation of the outcome. SVMs adopt a similar strategy: instead of operating on the space defined by the original set of predictors (where no linear boundary can correctly separate classes of the target outcome), they operate on a transformed space of higher dimensions *in which linear separability becomes possible.*

Consider, once again, the classification problem illustrated on the right panel of Figure 3 (now reproduced as a tilted projection at the bottom of Figure 4). Suppose we add a third feature equal to the negative sum of squares of the original predictors. This results in the 3D scatterplot depicted on Figure 4, slightly tilted to improve visibility. In this new, three-dimensional feature space, observations are now arrayed on a conical surface, with instances of class B rising to its apex. It is now easy to define a plane, depicted in gray in Figure 4, that cuts the top of this cone and separates instances of the two classes. The projection of this separating plane back onto the original two-dimensional space generates the circular decision boundary we needed. Once again, the SVM's goal is to learn this separating plane.

What is remarkable about actual implementations of SVMs is that there is no need to explicitly define what these additional dimensions are, provided they can be expressed as functions of the pair-wise dot products $\langle x_i, x_j \rangle$ of the original features. Specifically, SVMs rely on **kernels** $K(x_i, x_j)$ — bivariate, symmetric, and positive-definite functions that define measures of proximity or similarity between observations, and which operate on the space of original features. In practice different kernels support different kinds of implicit added features. This approach, called the **kernel trick**, avoids explicit re-mapping onto higher-dimensional spaces. Like boosting, kernelization can be applied to many different types of learners (including GLMs; see, for instance Hainmueller and

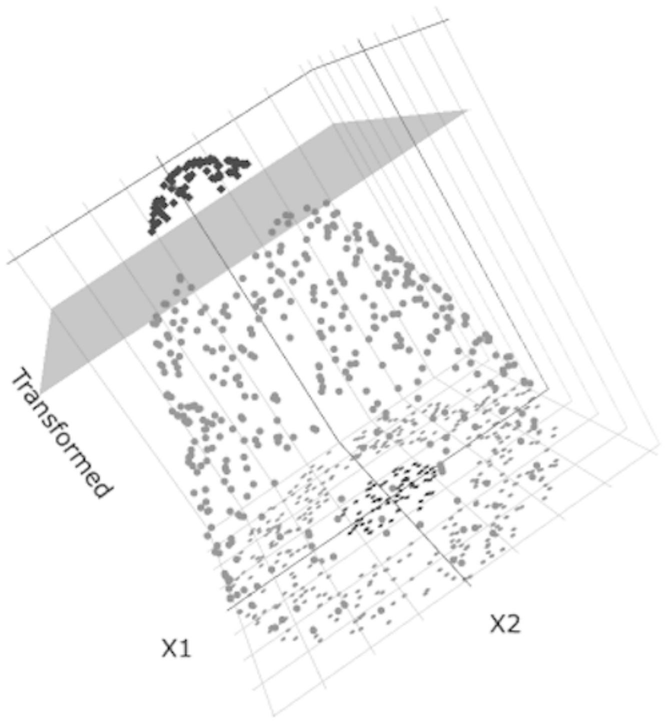

Figure 4: **Illustration of kernel trick**: Tilted projection of instances depicted in right panel of Figure 3 onto space with an additional dimension $z = -(X_1^2 + X_2^2)$ (original two-dimensional representation can be seen a the bottom). Observations of Class B (in dark gray) now cluster at the top of a conical surface. SVM learns the gray plane that cuts across this conical surface, resulting in perfect separation of Class A instances (light gray, below plane) and Class B instances (dark gray, above plane).

Table 1: Popular SVM Kernels

Kernel	Form
Linear	$K(x, x') = \langle x, x' \rangle$
dth Degree Polynomial	$K(x, x') = (1 + \langle x, x' \rangle)^d$
Radial Basis Function	$K(x, x') = \exp(-\|\|x - x'\|\|^2/c)$
Sigmoid (Neural Network)	$K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$

Hazlett, 2014).

In the SVM literature, popular kernels include the polynomial kernel, the radial basis function (RBF) kernel, and the sigmoid kernel. Table 1 provides a summary of the kernels. All of these (and many others) are implemented in the `svm` function included in **R** package `e1071`. The use of different kernels may result in different predictions and different weights being placed on the features. Additionally, values taken by different (hyper)parameters (such d in the Polynomial kernel, c in the RBF, or κ_1 and κ_2 in the Sigmoid kernel) must be defined by the researcher. Once again, in practice, cross-validation is a good strategy for choosing these values.

One interesting aspect of both the RBF kernel and the sigmoid kernel is that each is essentially a type of neural network. In fact, earlier editions of Hastie, Tibshirani and Friedman (2009) refer to the sigmoid kernel as the *neural network kernel.* The first is a type of neural network actually called an RBF network. The second is a standard neural network fit with a multilayer perceptron and one hidden layer.

Incorporating the kernel-transformed features into the SVM optimization problem is not difficult. In general, Equation 2 can be reformulated using kernels. Specifically, the function used to define the hyperplane can be rewritten as:

$$f(x) \;=\; \sum_{i \in \mathcal{S}} \alpha_i y_i K(x, x_i) \;+\; \beta_0. \tag{4}$$

with everything else (including soft constraints and prediction rules) staying as before.

Although we have discussed SVMs in the context of binary classification, there are extension to multinomial classification problems and even regression-like tasks, where they are typically called *support vector regressions* (Cortes and Vapnik, 1995; Hastie, Tibshirani and Friedman, 2009; Witten et al., 2016; Gründler and Krieger, 2015).

It is also worth noting that while SVM's are typically very fast and accurate (and scale particularly well in terms of the dimensionality of the original feature space), they have been shown to be very closely related to regularized logistic regression models (the so called *Loss + Penalty* representation of SVMs; Hastie, Tibshirani and Friedman, 2009), and they tend to provide similar results — particularly when classes are well separated.

3.2.3 An Application in the Study of Criticism in Congress

One common application of support vector machines is to learn characteristics of interest, effectively using the SVM as a measurement model. Examples of this are scattered throughout computer science and are becoming more common within political science. A good example is the recent study by the Pew Research Center's Data Labs team on partisan conflict and Congressional outreach 2017.

In light of heightened political hostility within and outside the walls of Congress, they questioned who more actively criticizes the other party, who "goes negative," and how the public responds to both of these strategies. To answer these questions, they looked at 108,235 Facebook posts and 94,521 press releases made by members of Congress between January 1, 2015 and April 30, 2016 (during the 114th Congress). Within the posts and releases, they needed to identify three characteristics of the text: whether the member of Congress in that document criticized someone or some group, whether they expressed disagreement, and whether they discussed bipartisanship.

To code every post by hand for each characteristics would have been prohibitively time intensive. As a alternative approach, the authors resorted to a **semi-supervised** approach. Specifically, they trained SVMs to identify the concepts of interest on a small hand coded subset of the data, and then used the trained model to classify the un-coded documents. First, they randomly sampled a subset of both the press releases and Facebook posts to code by hand.[7] The team at Pew then used Mechanical Turk to garner human-coded labels. Using these codes, they then re-trained an SVM for each concept of interest. To ensure that the final SVM used to classify the remaining data performed as well as possible, they compared penalty levels[8] and different kernels[9] with 5-fold cross validation to identify which combination resulted in the best performing algorithm out-of-sample.

[7]Because a sample of documents from every member of Congress would be needed and the use of each type of speech occurs in a small proportion of the text (approximately 10% or less of the time for each), random sampling would likely miss much of the picture. Instead, they drew a weighted random sample of documents.

[8]They tested γ at 1, 10, 100, 1000, and 10,000.

[9]They tested the linear and RBF kernels.

Finally, using the best performing model for each characteristic, they then applied labels to the remaining documents.[10]

The authors find that the party leadership and ideological extremists on both sides of the aisle are more likely to be critical of the other party and vocally disagree. Conversely, moderates in competitive districts are more likely to talk about bipartisanship. Additionally, they find that on Facebook the public tends to like, share, and comment on critical posts more than other posts.

4 Concluding Remarks: Promise and Perils of Machine Learning

Machine learning approaches and methods provide a wide variety of new and exciting tools to political scientists that are especially useful in the face of high-dimensional data sets. As political scientists seek answers to increasingly complex questions and using increasingly complex data (e.g. text or images), such techniques and methods will be increasingly relevant. However, such promise does not come without challenges and pitfalls. Political scientists face three general challenges when seeking to adopt and use machine learning techniques and other methods developed by computer scientists.

The first is derived from the fact that, for a long time, those developing machine learning approaches sought only to *predict* outcomes rather than *understand* the phenomena of interest. This is problematic for at least two reasons. First, many ML algorithms, methods, and models rely on black box techniques. As a result, interpreting the learned relationship between inputs and outcomes becomes much more difficult than with classic techniques used within political science. For example, while SVMs are fantastic tools for accurate classification, they offer no easy way of determining which features are most predictive. This implies that their use for theory evaluation is extremely limited. And while *partial dependence plots* (i.e. plots of marginal predicted outcomes as a function of features of interest; Friedman, 2001) can provide a sense of how the target changes as a function of given predictors, they remain underutilized and misunderstood.

Second, and partly as a result of the complex problems they tackle, ML approaches tend not to be robust to technical decisions made by researchers. While ML models take away researcher degrees of freedom and can help prevent common issues related to p-hacking or specification-related

[10]For a longer description of the process, see the methods note in the original report.

forking-paths (Gelman and Loken, 2013), these models come with their own set of potentially consequential decisions: How is data pre-processed to extract relevant features; How are initial values of parameters chosen; What set of criteria are used to evaluate goodness-of-fit? While cross-validation techniques can help justify and evaluate some of these choices, the added computational costs of these safe-guards can make this process prohibitively time-intensive, which results in many choosing to use defaults as a path of least resistance. Indeed, recent work has shown that these kinds of technical decisions can have important substantive consequences — consequences that may be hard for the discipline to identify (Denny and Spirling, 2018; Alvarez, 2016).

A byproduct of this second major challenge is a false sense of complacency induced by seemingly technical choices. It is naïve to think that data can "speak for itself," in a way that is untarnished by human biases. All data has a history — a distinctly *human* one at that. Contemporary sources of the massive online data-sets typically studied using the tools of ML, for instance, may reflect the commercial intents of those designing social platforms, rather than the actual preferences and typical behaviors of their users (Ruths and Pfeffer, 2014). Similarly, implicit and explicit human biases are embedded in data that exist already, and predictive models designed to learn patterns and trends will reproduce (and magnify) biases that went into the generation of data to begin with (Dressel and Farid, 2018).

Finally, it is important to remember that no amount of modeling sophistication or data size can, in its own right, correct the issues that plague observational studies. Selection on unobservables, interference, treatment heterogeneity — all of these issues will continue to pose threats to valid causal inferences. Unless description is the only goal of the learning exercise, researchers will need to justify the kinds of inferential jumps needed to move from prediction to explanation (Grimmer, 2015). Of course, this challenge is also, perhaps, the discipline's greatest opportunity to continue offering meaningful contributions to the field of ML: given our history and scholarly interests, we are uniquely positioned to generate analytic strategies at the intersection of sophisticated computation and careful attention to issues of causal identification with observational data.

References

Alvarez, R Michael. 2016. *Computational social science.* Cambridge University Press.

Bishop, C.M. 2016. *Pattern Recognition and Machine Learning*. Information Science and Statistics Springer New York.
URL: *https://books.google.com/books?id=kOXDtAEACAAJ*

Breiman, Leo. 2001. "Random forests." *Machine learning* 45(1):5–32.

Breiman, Leo, Jerome H Friedman, Richard A Olshen and Charles J Stone. 1984. "Classification and regression trees." *Wadsworth International Group* .

Breiman, Leo et al. 1998. "Arcing classifier (with discussion and a rejoinder by the author)." *The annals of statistics* 26(3):801–849.

Cortes, Corinna and Vladimir Vapnik. 1995. "Support-vector networks." *Machine learning* 20(3):273–297.

Denny, Matthew J and Arthur Spirling. 2018. "Text preprocessing for unsupervised learning: why it matters, when it misleads, and what to do about it." *Political Analysis* 26(2):168–189.

Dressel, Julia and Hany Farid. 2018. "The accuracy, fairness, and limits of predicting recidivism." *Science advances* 4(1):eaao5580.

Efron, Bradley and Trevor Hastie. 2016. *Computer age statistical inference*. Vol. 5 Cambridge University Press.

Freund, Yoav and Robert E Schapire. 1997. "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of computer and system sciences* 55(1):119–139.

Friedman, Jerome H. 2001. "Greedy function approximation: a gradient boosting machine." *Annals of statistics* pp. 1189–1232.

Friedman, Jerome H. 2002. "Stochastic gradient boosting." *Computational Statistics & Data Analysis* 38(4):367–378.

Gelman, Andrew and Eric Loken. 2013. "The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time." *Department of Statistics, Columbia University* .

Ghitza, Yair and Andrew Gelman. 2013. "Deep interactions with MRP: Election turnout and voting patterns among small electoral subgroups." *American Journal of Political Science* 57(3):762–776.

Green, Donald P and Holger L Kern. 2012. "Modeling heterogeneous treatment effects in survey experiments with Bayesian additive regression trees." *Public opinion quarterly* 76(3):491–511.

Grimmer, Justin. 2015. "We are all social scientists now: how big data, machine learning, and causal inference work together." *PS: Political Science & Politics* 48(1):80–83.

Gründler, Klaus and Tommy Krieger. 2015. Using support vector machines for measuring democracy. Technical report Discussion Paper Series, Chair of Economic Order and Social Policy, Universität Würzburg.

Hainmueller, Jens and Chad Hazlett. 2014. "Kernel regularized least squares: Reducing misspecification bias with a flexible and interpretable machine learning approach." *Political Analysis* 22(2):143–168.

Hastie, Trevor, Robert Tibshirani and Jerome Friedman. 2009. *The elements of statistical learning.* Vol. 1 Springer series in statistics New York.

Ho, Daniel E, Kosuke Imai, Gary King and Elizabeth A Stuart. 2007. "Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference." *Political analysis* 15(3):199–236.

Hothorn, Torsten, Kurt Hornik and Achim Zeileis. 2006. "Unbiased recursive partitioning: A conditional inference framework." *Journal of Computational and Graphical statistics* 15(3):651–674.

Imai, Kosuke and Aaron Strauss. 2011. "Estimation of heterogeneous treatment effects from randomized experiments, with application to the optimal planning of the get-out-the-vote campaign." *Political Analysis* 19(1):1–19.

Imai, Kosuke, James Lo and Jonathan Olmsted. 2016. "Fast estimation of ideal points with massive data." *American Political Science Review* 110(4):631–656.

Jordan, Michael I and Tom M Mitchell. 2015. "Machine learning: Trends, perspectives, and prospects." *Science* 349(6245):255–260.

Kaufman, Aaron, Peter Kraft and Maya Sen. 2018. "Improving Supreme Court Forecasting Using Boosted Decision Trees." *URL: j. mp/sctfore* .

Messing, Solomon, Patrick VanKessel, Adam Hughes, Rachel Blum and Nick Judd. 2017. "Partisan Conflict and Congressional Outreach." *Pew Research Center Report* .

Montgomery, Jacob M and Santiago Olivella. 2018. "Tree-Based Models for Political Science Data." *American Journal of Political Science* 62(3):729–744.

Muchlinski, David, David Siroky, Jingrui He and Matthew Kocher. 2016. "Comparing random forest with logistic regression for predicting class-imbalanced civil war onset data." *Political Analysis* 24(1):87–103.

Murphy, K.P. and F. Bach. 2012. *Machine Learning: A Probabilistic Perspective.* Adaptive Computation and Machine Learning series MIT Press.
URL: *https://books.google.com/books?id=RC43AgAAQBAJ*

Ng, Andrew Y and Michael I Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*. pp. 841–848.

Ratkovic, Marc and Dustin Tingley. 2017. "Sparse estimation and uncertainty with application to subgroup analysis." *Political Analysis* 25(1):1–40.

Roberts, Margaret E, Brandon M Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson and David G Rand. 2014. "Structural topic models for open-ended survey responses." *American Journal of Political Science* 58(4):1064–1082.

Ruths, Derek and Jürgen Pfeffer. 2014. "Social media for large studies of behavior." *Science* 346(6213):1063–1064.

Schapire, Robert E. 1990. "The strength of weak learnability." *Machine learning* 5(2):197–227.

Wager, Stefan and Susan Athey. 2017. "Estimation and inference of heterogeneous treatment effects using random forests." *Journal of the American Statistical Association* (just-accepted).

Wager, Stefan, Trevor Hastie and Bradley Efron. 2014. "Confidence intervals for random forests: The jackknife and the infinitesimal jackknife." *The Journal of Machine Learning Research* 15(1):1625–1651.

Witten, Ian H, Eibe Frank, Mark A Hall and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.