HARVARD | BUSINESS | SCHOOL

Machine Learning Complementarities with Regression Analysis: Selection, Evaluation, and Interpretation of Machine Learning Methods for Exploratory Data Analysis and Pattern Detection

Prithwiraj (Raj) Choudhury
Ryan Allen
Michael G. Endres

Working Paper 19-032

Machine Learning Complementarities with Regression Analysis: Selection, Evaluation, and Interpretation of Machine Learning Methods for Exploratory Data Analysis and Pattern Detection

Prithwiraj (Raj) Choudhury
Harvard Business School

Ryan Allen
Harvard Business School

Michael G. Endres
Harvard University

Working Paper 19-032

Copyright © 2018, 2019 by Prithwiraj (Raj) Choudhury, Ryan Allen, and Michael G. Endres

Working papers are in draft form. This working paper is distributed for purposes of comment and discussion only. It may not be reproduced without permission of the copyright holder. Copies of working papers are available from the author.

Machine Learning Complementarities with Regression Analysis: Selection, Evaluation, and Interpretation of Machine Learning Methods for Exploratory Data Analysis and Pattern Detection

Prithwiraj (Raj) Choudhury,[1] Ryan Allen,[1] and Michael G. Endres

This draft May 27, 2019

Machine learning (ML) methods could have a larger footprint in strategy and management research if used to conduct exploratory data analysis and to identify robust patterns in the data that are relatively difficult to identify using traditional methods. Compared to traditional causal inference methods, ML algorithms make far fewer *a priori* assumptions about the functional form of the underlying model that best represents the data, thus researchers could use ML methods to explore novel and robust patterns in data. Once identified, these patterns can be tested using traditional econometric methods. Using a motivating example, we demonstrate how to implement three specific ML algorithms (decision trees, random forests, and neural networks) to reveal patterns in data and offer guidance on *selecting* ML algorithms. We also demonstrate how to use visual tools to *evaluate* the resulting model performance and *interpret* predictions of the model to illuminate patterns in the data. We also provide sample code to employ ML algorithms used in the paper.

Keywords: machine learning, supervised machine learning, exploratory data analysis, pattern detection, decision trees, random forests, neural networks, ROC curve, confusion matrix, partial dependence plots, heat maps

[1] Harvard Business School. Corresponding author: Raj Choudhury (email: pchoudhury@hbs.edu). The authors thank Kathy Eisenhardt, Dan Levinthal, Rory McDonald, Joe Mahoney, Evan Starr, Ron Tidhar, and participants at the 2018 Academy of Management and the 2018 Strategy Science conference held at the Wharton School for helpful comments on a prior draft. All errors remain ours.

1. INTRODUCTION

Machine learning (ML) methods represent an exciting but underutilized toolkit for strategy and management researchers. With notable exceptions (Kaplan & Vakili, 2015; Choudhury, Wang, Carlson, & Khanna, 2019; Gross, 2018; Menon, Choi, & Tabakovic, 2018; Furman & Teodoridis, 2018), the application of ML methods in our field remains low. Among other reasons, there has arguably been a mismatch in the stated objectives of ML methods and the goals of empirical researchers in strategy and management. In contrast to the traditional econometric approach in our fields, for which the goal is causal inference and to produce consistent *parameter estimates* $\hat{\beta}$ that describe the relationship between y and x, ML produces *predictions* of $\hat{y}$ given x (Mullainathan & Spiess, 2017). However, the key objective of strategy and management research is *not* to predict any particular $\hat{y}$, but to employ data and econometric methods to discover, document, and explain relevant causal conjectures (Van Maanen, Sorensen and Mitchell, 2007).

This paper attempts to broaden the footprint of ML methods in strategy and management research by demonstrating an alternative purpose of using ML methods: *as an exploratory tool to identify robust nonlinear or interactive patterns in quantitative data*. By revealing these patterns, ML can help researchers formulate more nuanced hypotheses grounded in data, which can be tested using traditional econometric tools. In other words, we argue that ML methods can serve as a complement to traditional econometric methods. Our exposition of *how* ML methods can help illuminate patterns in the data are also a complement to Puranam, Shrestha, He, & von Krogh (2018), who argue that such methods can aid inductive theorizing by revealing replicable associative patterns in data.

We argue that ML methods excel at exploratory data analysis and pattern detection because, unlike traditional methods, they identify complex patterns in x that relate to y using structure that was not specified *a priori*. In other words, rather than deductively testing a model specified *ex ante* by the researcher (as is often the case with traditional regression analysis focused on inference), ML

algorithms train a model inductively from the data. Research in the physical sciences has demonstrated that ML can help scientists uncover physical laws by inducing relationships in data (Ruff et al., 2017; Rudy, Alla, Brunton, & Kutz, 2018; Hirsh, Brunton, & Kutz, 2018). Like other forms of exploratory data analysis (see Cattaneo, Crump, Farrell, & Feng, 2019), we argue and demonstrate that ML can complement traditional econometric tools, such as linear regression analysis, because of its ability to uncover relationships that might otherwise go unnoticed by the researcher. These exploratory analyses can also be used as inputs to theorize plausible explanations for the patterns and to formulate testable hypotheses. Following the grounded theory tradition (Glaser & Strauss, 1967), ML may be viewed as a set of systematic tools that can help researchers develop more nuanced models that better fit the reality represented in quantitative data than does a prespecified model.[2]

As an illustrative example, we use several supervised ML algorithms to conduct an exploratory analysis of employee turnover at a large technology company (hereafter, TECHCO). First, we use three ML algorithms (a decision tree, a random forest, and a neural network) to train models using our data. Compared to the global linear fit of a traditional logistic regression model (one of the workhorse traditional models used to study employee turnover), the ML algorithms rely on fewer functional form assumptions; this configuration allows them to capture heterogeneous effects for different subgroups at different points in time. The results from using the ML methods reveal an interesting pattern: a small group of employees with low training performance were dramatically more likely to leave the firm during their first six months on the job.

[2] This process is known as *abduction*. In the words of Mantree and Ketokivi (2013) "we predict, confirm, and disconfirm through deduction, generalize through induction, and theorize through abduction" (Mantree and Ketokivi, 2013, p.72).

Next, we estimate a logistic regression model[3] to test how observable characteristics influence employee turnover. When we conduct this analysis *without* incorporating our awareness of patterns identified by ML methods, the results indicate that higher training performance is correlated with a lower relative hazard of turnover. We then incorporate insights drawn from the pattern revealed by the ML methods and use logistic regression again to approximate local linear effects for employees with high and low training performance, both before and after their first six months at the company. The updated results reveal that after the first six months of employment, training performance is significantly *positively* correlated with a hazard of turnover for employees with high training performance scores (> 4.0). Only the small subset of employees who scored poorly during training was significantly more likely to leave, and only during the first six months. The effect for this small subset of employees was large enough to drive a negative global effect at odds with the true positive effect for the large majority of employees. A well-trained econometrician might discover these or similar patterns in the data without ML methods, but it would be difficult and time consuming to do so in a systematic way, especially with a larger number of covariates and a large set of possible interactions or nonlinear polynomial terms.

In addition to advocating for ML as an exploratory tool and demonstrating with a concrete example, this paper provides a practical how-to guide for strategy and management researchers who want to use and present ML results. We provide useful tools for ML algorithm *selection*, as well as *evaluation* and *interpretation* of the resulting models. Selection among various ML algorithms involves considering trade-offs among performance, computational speed, interpretability, and ease of use. To evaluate performance of various models, we include two plots: (1) a plot of training and validation loss (error); and (2) a receiver operating characteristic (ROC) plot, a graphical comparison

[3] Here we are referring to a logistic regression model without regularization. The logistic regression can be considered a machine learning algorithm that can train a model with flexible functional form if a regularization term is included in its loss function. But in our paper, it differs from the other ML models because we do not regularize the model.

of the true positive rate to the false positive rate. For interpretation of the models, we create partial dependence plots and heat maps to visualize how changes in each covariate (and interactions between covariates) are related to changes in predicted outcomes. These plots can help researchers visualize the structure of the relationship between x and y; these insights can, in turn, generate specific hypotheses that can be locally tested using traditional econometric techniques.

Insights from ML can be powerful, but such exploratory analysis should be applied carefully. In the discussion, we warn of the limitations of ML, which is designed to solve the problem of prediction of $\hat{y}$, not inference of $\hat{\beta}$. Even when ML models yield regression coefficients, ML algorithms do not necessarily produce consistent $\hat{\beta}$ point estimates, and standard errors can be nonexistent, unreliable, or difficult to calculate. Therefore, as a general rule, ML should not be used on its own for statistical inference. We also summarize limitations related to algorithmic bias and suggest best practices for researchers to mitigate biased interpretation and ensure patterns identified in data are robust.

In summary, we do not propose ML as a replacement for traditional econometric techniques. Instead, ML can complement traditional methods by helping researchers identify patterns in data, which can later be tested rigorously using traditional econometric techniques like OLS or logistic regression. The remainder of the paper proceeds as follows: Section 2 presents a stylized motivating example for how ML methods can complement traditional econometric regression analysis. Section 3 presents the steps of implementing ML. Section 4 describes three specific ML algorithms. Sections 5, 6, and 7 give guidance for selection, evaluation, and interpretation of ML algorithms in research. The online appendix provides code and simulated data to help readers reproduce our results and apply ML tools to their own research.

2. MOTIVATING EXAMPLE: USING ML TO IDENTIFY PATTERNS IN THE DATA

Throughout the paper, we use an internal dataset from a large technology firm, TECHCO. The dataset covers 1,688 entry-level employees deployed to any of TECHCO's nine production centers. Covariates include an employee's performance scores in an intensive three-month onboarding training course (*Training Performance*), university verbal and math test scores (*Verbal Score* and *Logical Score*), date of arrival at the company (*Month Arrived*), and demographic information. The data also include the assigned production center's age (*Production Center Age*), its distance from the employee's hometown (*Distance*), and the similarity of its prevailing language to that of the employee's hometown (*Language Similarity*). Table 1 provides basic summary statistics. Our panel data consists of one observation for each month that an individual is employed at the company and a dependent variable, *Quit*, indicating whether the employee quit during that time period (1 if departed, 0 if stayed). Our goal is to estimate the relative hazard of turnover for a given employee at a given time.

Insert Table 1 about here

2.1 Stylized motivating example

Consider a naïve logistic regression model that includes each covariate of interest and a dummy variable for each time interval (months 1 through 40). It is possible to use a linear model[4] like the logistic regression to test nonlinear or interactive relationships by adding transformed covariates to the model. For example, we may add a squared covariate to test a U-shaped relationship or multiply two covariates to test an interaction effect. But, which terms should be included? Researchers informed by theory may model such complexities based on known relationships. However, it is

[4] The term *linear model* is used to mean that the model is a function of $\boldsymbol{\beta} \cdot \boldsymbol{x}$.

often unknown *a priori* how best to model each covariate, and the number of combinatorial possibilities increases rapidly as more covariates are considered.

Machine learning presents a solution to the problem of knowing how to best model the data. Later in the paper, we explain how to implement ML in research and demonstrate with three different ML algorithms. At the risk of oversimplification, we give a preview of insights from the decision tree (arguably the simplest ML algorithm we use) applied to the TECHCO data as a stylized motivating example. For now it suffices to say that the decision tree algorithm repeatedly splits the TECHCO observations into distinct subsets, and each subset is assigned a probability of turnover. The algorithm splits the data into subsets in a way that maximizes the model's predictive power (i.e., minimizes the "loss" or "error" of the model). The resulting model resembles a tree with nodes that split into branches, with the observations in all the terminal nodes ("leaves") summing to the entire sample.

Insert Figure 1 about here

Figure 1 is a visual representation of the decision tree model applied to the TECHCO turnover data. One of the desirable attributes of decision tree algorithms is the ease of visualization of the resulting model. For example, the top node (the "root node") of the tree in Figure 1 is labeled *Training Performance* <= 3.995. Thus, the single split that maximized predictive performance was to separate the data into the 36,331 observations whose training performance score exceeded 3.995 from the 647 observations whose scores were below 3.995.[5] Following the left-hand branch of the tree (labeled "True"), we see that within the 647 observations with low training performance, the split that maximizes predictive performance splits observations with *Time* <= 6.5 (months) from

[5] Though our dataset is based on 1,688 employees tracked over 40 months, the number of observations is not a full panel of 1,688*40 = 67,520. This is because the data stops tracking employees who leave, so those who leave before month 40 have fewer than 40 observations. Furthermore, we drop some observations with missing values.

those > 6.5 (months) and so on. This decision tree reveals an interesting pattern: the tree splits along only two dimensions, *Training Performance* and *Time*. This pattern is indicative that these dimensions may be characterized by important nonlinearities and interactions. Furthermore, the second terminal node on the bottom row (with *Training Performance* <= 3.995*, Time* <= 6.5*,* and *Time* > 4.5) has a much higher proportion of turnover cases than any other terminal node (46/100 are turnover events).

This relatively simple ML visualization is *not* necessarily the optimal ML method for detecting robust associations in the data on its own—the decision tree was just one tool we used in our analysis. That said, for the sake of a stylized motivating example, we can use insights from the decision tree to modify the naïve logistic regression model (later in the paper, we generalize and provide guidance on ML algorithm selection and evaluation). We make two simple modifications: (1) we estimate the logistic model separately on two subsets of the data—observations during the first six months on the job (*First 6 months*) and after six months (*After 6 months*); and (2) we add a dummy variable, *Low Training Performance* (1 for employees with *Training Performance* below 4; and 0 otherwise), and its interaction term with *Training Performance*. By making these modifications, we now allow the linear model to estimate the local effect of *Training Performance* on turnover for two different time periods, and we compare effects for employees who had high and low training scores as identified by the decision tree algorithm. The results from these local estimations reveal very different results than the original logistic regression. Results from the naïve regression are presented in column 1 of Table 2 and indicate a negative and statistically significant relation between training performance and the dependent variable, *Quit*. Columns 2 and 3 reveal that the original global estimate of the effect of *Training Performance* on turnover (-0.887) was driven entirely by the dramatically higher hazard of turnover for those with scores below 4 during the first six months. The local linear estimates reveal an effect with the *opposite sign* for the large majority of the individuals represented in the data (those

with *Training Performance* above 4 after the first six months, as reported in column 3 of Table 2). This finding demonstrates the potential cost of naively estimating global linear relationships in the data. ML helped us identify the most important heterogeneous subsets of the data for separate hypothesis testing. We acknowledge again that a well-trained econometrician might discover these or similar patterns in the data without ML methods, but it would be difficult and time consuming to do so in a systematic way, especially with a larger number of covariates and a large set of possible interactions or nonlinear polynomial terms.

Insert Table 2 about here

While helpful, the decision tree algorithm may not always be the best performing or most reliable algorithm, as it can be highly sensitive to small modifications (e.g., adding a covariate or changing the data sample dramatically changes the tree). For this reason, we recommend using multiple ML algorithms to identify robust patterns in the data. The remainder of the paper presents a general step-by-step framework for using ML, and uses the TECHCO data to demonstrate how to *select* between ML algorithms, as well as how to *evaluate* and *interpret* the resulting models.

3. IMPLEMENTING SUPERVISED MACHINE LEARNING ALGORITHMS: STEP-BY-STEP

Supervised machine learning is focused on the problem of prediction: it produces predictions of $\hat{y}$ from x given a set of "ground truth" labels for the dependent variable, y.[6] For example, for the task of visually identifying a chair, an ML algorithm does not need to be explicitly programmed to deductively test whether the object has four legs, a surface, and possibly back support—thus making it a chair. Such rule-based systems could fail to recognize the difference between a table and a stool.

[6] In contrast, unsupervised ML involves algorithms that are used on data that do not have known dependent variable (e.g., clustering data into like groups based on covariate values). Throughout this paper, we use the term ML to refer to supervised ML.

The supervised ML algorithm can simply be fed thousands of images marked "chair" or "not chair," and it induces tacit rules for what characteristics define a chair (Autor 2015). How is machine learning different from traditional econometric estimation? Consider the logistic regression,[7] which has a prespecified functional form: a linear coefficient for each covariate. By contrast, the functional forms of ML are not fully specified by the researcher; the form emerges from the data. The goal of using ML algorithms should be to find the model that best fits a sample dataset (also known as the training dataset), without *overfitting*, so that the model is able to make accurate out-of-sample predictions. The tension between fitting the in-sample data perfectly and generalizing to out-of-sample data is known to scholars as the bias-variance trade-off (see Figure 2). As a model is overfit (that is, relies too much on idiosyncrasies of in-sample data for prediction), its bias decreases but its variance increases, making it less generalizable; an underspecified model is biased when its simplicity does not describe the data very well.

Insert Figure 2 about here

ML methods include an arsenal of techniques such as cross-validation and regularization (discussed later in this section) that limit the model's power to describe in-sample data so that the model can perform well using out-of-sample data. This is helpful for exploratory data analysis because it helps us model the meaningful nonlinearities and interactions in $\boldsymbol{x}$ without overfitting the data. The objective of any ML algorithm is to minimize a loss function (i.e., objective function), which punishes a model for incorrectly predicting outcomes. There are two general classes of loss functions: those intended for "regression" of continuous dependent variables (examples of such loss functions include mean squared error, absolute error, quantile loss) and those aimed at "classification" for categorical dependent variables (examples of such loss functions include log-loss,

[7] We are referring to a logistic regression without regularization. With regularization, it is akin to ML.

hinge loss, 0-1 loss). An in-depth discussion of the choice of loss function is beyond the scope of this paper, but in practice the default loss functions provided by statistical packages are usually sufficient. Throughout this paper, we use a loss function foundational for many classification problems: the log-loss function. Appendix 1 provides greater detail for conceptually understanding the loss function.

We represent the predictions of each ML model—the "hypothesis"—with the term $h_\theta(\boldsymbol{x})$. The hypothesis function $h_\theta(x_i)$ outputs a specific predicted probability of a turnover event given the observation data x_i and parameter values θ (see Appendix 1 for detailed exposition of how the hypothesis fits into the loss function). When we use different models (e.g., the models built by the decision tree or neural network algorithm) in this paper, we are simply changing the functional form of $h_\theta(\boldsymbol{x})$ in the log-loss function. The loss (i.e., error) output by the loss function increases when these predictions are poor and decreases as the predictions improve (see example in Appendix 1). When an ML algorithm fits a model to the data using statistical software, an optimization algorithm iteratively tries different model parameter values θ (essentially the covariates of the model) until it finds the model that yields the minimum value of the loss function, subject to constraints placed on the model to limit overfitting. With the foundational understanding of the bias-variance trade-off and the loss function, we provide a general step-by-step framework designed to help researchers use ML to find a model that produces generalizable predictions.

Step 1: Preprocess the data

Preprocessing the data—including "feature engineering"[8] and handling missing data[9]—can be the most significant step for model predictive performance. ML practitioners often include many

[8] "Feature" is another word for covariates or functions of covariates. "Feature engineering" refers to selecting and creating features, modifying features (e.g., bucketing a continuous variable), or standardizing the values.
[9] Handling missing data can heavily influence model performance. Dropping missing observations can severely limit the number of observations and may create erroneous results if the excluded data are systematically related to the outcome

features (covariates or functions of covariates) and may select the most predictive features algorithmically.[10] However, it is important to exclude irrelevant covariates and anything else that predicts outcomes in-sample only due to a quirk in the sample data (e.g., a meaningless data label that correlates highly with the outcome). Researchers using ML for purposes of exploratory data analysis should selectively choose the most *theoretically relevant covariates*.[11] An essential piece of feature engineering is to standardize features/covariates used in models that calculate distance between points (e.g., neighbor methods like KNN or support vector machines, described later) or when used in algorithms that use a regularization term (e.g., neural networks). If not standardized, covariates with larger magnitudes will overwhelm features with smaller magnitudes as the algorithm assigns weights. Units are commonly converted to "z-scores" or "minmax" scores, which strip units so that all numerical magnitudes are comparable across covariates.

Step 2: Select an algorithm

The next step is to select the ML algorithm that will build the predictive model.[12] The model is simply a function that uses data as input and produces predictions $h_\theta(\boldsymbol{x})$. The algorithm attempts to find parameters of the model, θ, that minimize the loss function. In Section 4, we will introduce and implement three ML algorithms: decision tree, random forest, and neural network. Each algorithm

variable. As a solution, missing values can be imputed. Missing numerical values can simply be replaced with the covariate mean or median value, and missing categorical values can be replaced with the mode. Alternatively, missing values can be replaced with an estimated value—that is, run a regression model to learn what values predict the value for non-missing observations and fill in missing observations with the predicted values. If a covariate has many missing values and is not central to the prediction, it may be best to simply drop the covariate.

[10] Algorithmic feature selection is an important machine learning skill that is not covered in depth in this paper. Regularization, discussed in a later section, can be a form of feature selection.

[11] The risk of including too many variables is that when covariates are highly correlated, the ML algorithm may quasi-randomly select one covariate over another in different random samples of the data, yielding unreliable models and results that are difficult to interpret.

[12] What is the difference between the "algorithm" and the "model?" The "algorithm" is what is used to train the "model." The trained "model" is simply a function that takes data as input and outputs a predicted probability.

simply builds a different model to predict probabilities $h_\theta(\boldsymbol{x})$. In Section 5, we provide guidance on how to select among various algorithms.

Step 3: Choose regularization

Regularization is any constraint that restricts the descriptive capacity of the model—essentially smoothing the functional form—in order to prevent overfitting (recall Figure 2). This is done by tuning (i.e., adjusting the values of) algorithm-specific hyperparameters. A *hyperparameter* is any parameter of the algorithm that is set *before* estimating the model. Hyperparameters are not learned from the data; they are assigned to the model by the researcher. The ML algorithms we implement later in this paper all have specific hyperparameters, which can be "tuned" (i.e., adjusted) to avoid under- and overfitting. In decision trees, for example, we must set "stopping rules" to limit the growth of the tree. In theory, a decision tree could be allowed to split until each observation is represented by a leaf on the tree—but this model would be extremely overfit to the in-sample data. In each case, tuning hyperparameters is a delicate balancing act between bias (underfitting) and variance (overfitting). For some algorithms, an important hyperparameter is the choice of a "regularization term" (or "penalty term") to add to the loss function. Adding regularization terms to a loss function controls for overfitting by punishing the loss function for putting too much predictive weight on a covariate (Appendix 1describes how a regularization term in the loss function prevents overfitting; an example familiar to researchers is the LASSO regression).

Step 4: Partition the data for training, (cross-)validation, and testing

What are the optimal values at which to set the hyperparameters? Balancing bias and variance is achieved by partitioning the data so that the model is fitted using a "training sample" entirely distinct from the "validation sample" used to evaluate the model's predictive performance. This distinction allows researchers to tune the hyperparameters of the algorithm that is "learning" from training data

until its predictive performance *on the validation data* is optimized. A final subset of data, the holdout test set, is kept separate from both the training set and the validation set to give an even better estimate of predictive performance on data that were not used to train or validate the model (see step 7). Typically, the data are partitioned randomly into either three subsets (~60% training, ~20% validation, and ~20% holdout test) or into two subsets (~70% training-validation and ~30% holdout test) to be used for k-folds cross-validation. Throughout this paper we use k-folds cross-validation. This method of cross-validation is less sensitive to the idiosyncrasies of training and validation set selection and usually gives more reliable evaluations for smaller datasets, though it is more computationally intensive. In k-folds cross-validation, the training-validation data are split randomly into k equal-sized subsets of data. One by one, each of the k subsets is used as the validation data; the other k-1 subsets are used to train the model. The resulting k estimates of the loss / error from each trained model are averaged for the final estimate of the loss / error, thus making the evaluation of the model less subject to idiosyncrasies in any single split of the data. Throughout this paper, we use 10-fold cross-validation ($k = 10$), a common choice for k.

Step 5: Fit the model on the training set and evaluate predictive performance on the validation set

This is the core step, in which machine learning's actual "learning" takes place. In practice, this step is often the easiest part of the ML process. Thanks to statistical software like R and Python, fitting the model to the data often involves a single line of code (see the code in the Online Appendix). The model is "fitted" to the training data by an optimization algorithm, which chooses the parameter values $\boldsymbol{\theta}$ that produce predictions $\boldsymbol{h_\theta(x)}$ that minimize the loss function, subject to the hyperparameter constraints. An in-depth discussion of optimization algorithms is beyond the scope of this paper. Intuitively, an optimization algorithm will iteratively try different parameter values $\boldsymbol{\theta}$ for the model until the loss function is minimized. It is useful to be aware, however, that for complex models (e.g., neural networks), these algorithms may locate local rather than global

optima.[13] A signal of the presence of multiple local optima is that the model's fit results vary significantly with the choice of initial parameter values. Although there is no simple solution to this problem, it can sometimes be addressed with a better choice of initial parameter values or stronger regularization.

The fitted model is used to predict outcomes in the validation data, and the resulting predictions are evaluated against the true outcomes in the validation data using a performance metric. Throughout this paper, we use the log-loss score[14] (the total loss of the unregularized loss function) as the measure of model performance. To be clear, using 10-fold cross-validation, each model is "fitted" on training data, and predictive performance is evaluated on validation data 10 times. The final log-loss score is the average of the log-loss score from each of the 10 fitted models.

Step 6: Repeat steps 1–5, varying the algorithm, feature, hyperparameter, and regularization choices to maximize predictive performance

After fitting the model on the training data and evaluating its performance using validation data, the researcher ought to try different combinations of features and hyperparameters, with the goal of finding the specifications that yield the lowest validation loss. Beyond modifying algorithm hyperparameters, changing the algorithm itself (e.g., shifting from decision tree to neural network) can also improve predictive performance. It is difficult to know *a priori* which algorithm will yield the best estimates; often ML practitioners try as many algorithms as is feasible (starting with the simplest) and ultimately select the algorithms with the best performance. It is also possible to

[13] For example, one common algorithm is the gradient descent algorithm. Imagine that loss as a function of covariates x is represented in 3D space by a rugged multi-peaked landscape where peaks represent high loss and valleys represent low loss. The gradient descent algorithm finds the steepest route down from whatever hill it is initially positioned on, and it stops when it cannot descend any farther. Thus, in some cases, the initial values assigned to an algorithm can lead to substantially different predicted models. In the cases of linear and logistic regressions, the loss function is convex by virtue of the linear hypotheses; thus, this problem is not encountered. In general, however, the problem of multiple local minima can be quite challenging.

[14] The output of the loss function is a common metric. Other measures of predictive performance, such as F1 score, may be appropriate for different objectives.

combine models using ensemble methods—averaging the final estimates of several different models—to balance the strengths and weaknesses of each model.[15]

Thus far, for simplicity, we have asserted that the objective of the process is to minimize the validation loss (i.e., the output of the loss function on the validation set). More accurately, both the training loss and the validation loss should be minimized in such a way that the validation loss does not diverge from the training loss. Divergence of the training loss from the validation loss is a sign that the model overfits the in-sample data at the expense of performance on out-of-sample data. Figure 3 plots the training and validation loss of the random forest model predictions as a function of the "tree-depth" hyperparameter—a restriction on the maximum number of levels from any node to the root of the tree. We trained and evaluated the model (using cross validation) eight times, varying the "tree-depth" hyperparameter from values 1 through 8. Orange (upper) represents the validation loss, and blue (lower) represents the training loss of the model predictions across these eight hyperparameter values. Because the random forest algorithm has unbounded descriptive capacity in-sample, the training loss approaches 0 as regularization is eliminated. It appears that the best choice for tree depth would be around 3 or 4—the choice at which both training and validation losses are low, but validation loss has not diverged from training loss.

Insert Figure 3 about here

This process of minimizing validation loss (without diverging from the training loss) should be repeated for each important hyperparameter. Rather than doing so manually (i.e., by adjusting the value of one hyperparameter at a time), many statistical packages include support for "grid search" techniques that systematically fit and evaluate the model using every possible combination of a user-specified set of hyperparameter values. Despite these tools, it is difficult to try every possible

[15] Length limitations preclude detailed discussion of ensemble methods, but they merit mentioning. The random forest is an example of an ensemble method.

combination of algorithm, feature, and hyperparameter values. The actual process of tuning hyperparameters can be messy and recursive in nature—the code in the Online Appendix gives a more detailed guide for implementing this process in practice.

Step 7: Evaluate final predictive performance using the holdout test set

Once the algorithm has been optimally tuned, the final predictive performance is evaluated on the holdout test set. Because the holdout test set was not used in any fitting or evaluation of models in the previous steps, it represents the purest out-of-sample test available to evaluate model performance. This is a metric that can be reported to communicate concisely the final predictive performance of the model.

4. THREE EXAMPLE MACHINE LEARNING ALGORITHM IMPLEMENTATIONS

Following the ML framework, we implement three machine learning classification algorithms: (1) decision trees, (2) random forests, and (3) neural networks. We chose these particular algorithms for two reasons. First, they are widely used general purpose algorithms that pedagogically demonstrate a variety of ML algorithm attributes: the decision tree is easily interpretable, the random forest is a highly useful general-purpose algorithm that demonstrates ensemble techniques, and the neural network is the basis of many modern technological applications of ML. Second, these algorithms are relatively well suited to helping us estimate relative hazard for our survival data (i.e., predicted probabilities like $h_\theta = 0.03$) rather than drawing clear decision rule boundaries between classes (i.e., binary predictions such as $\hat{y} = 1$ or $\hat{y} = 0$).[16] For practical implementation, readers can follow

[16] These algorithms all optimize the log-loss function. The log-loss function is suitable to our data because we are most interested in comparing the probabilities of leaving rather than the predicted outcomes. Because the probability of any given employee quitting in a particular month is extremely low, it is very difficult to predict exactly when someone will leave. The purpose is not to draw clear boundaries between classes (i.e., binary predictions such as $\hat{y} = 1$ or $\hat{y} = 0$), but to learn relative hazard. Algorithms like the support vector machine, which uses the hinge loss function, would probably perform poorly because they are designed for hard decision rules. However, it is common to simply try multiple algorithms and see what works best.

along in the Online Appendix, which provides more detailed explanations, code, and data (simulated using the TECHCO data) to fit the models and replicate the results.

4.1 Decision trees (machine learning algorithm 1)

The algorithm repeatedly splits the multidimensional space $\boldsymbol{x}$ into two partitions, along one dimension at a time. The algorithm splits $\boldsymbol{x}$ so that each resulting partition is modeled as a constant value chosen to minimize total loss. Within each new partition, $\boldsymbol{x}$ is split once again, repeatedly splitting along one dimension at a time to minimize the loss within each sub-partition. Each split can be represented visually as a node with two branches, creating the overall impression of a tree with a root node representing the first split and the leaves as terminal nodes with a single constant value prediction for each partition of the data. To control for overfitting, the model is regularized by "stopping rule" hyperparameters that limit growth of the tree. Hyperparameters include limiting the depth of the tree (recall the tree depth hyperparameter for the random forest in Figure 3), requiring each leaf to contain a minimum number of observations, requiring a specified minimum decrease in the loss with each split, or requiring that splits occur only if the parent node contains a minimum number of observations.[17] Figure 1 (presented in Section 2 as the motivating example) displays our decision tree model.

4.2 Random forest (machine learning algorithm 2)

Although readily interpretable, decision trees are highly sensitive to values of the hyperparameters that determine how many splits the tree should make before overfitting the data (the stopping rule). Random forest algorithms, although not easily amenable to visual representation like decision trees, provide a solution to this problem. Random forests are an ensemble-based approach: many decision

[17] After tuning the model by trying many combinations of possible hyperparameter values, our best performing decision tree model allowed a maximum tree depth of 3 and allowed splits only if the decrease in loss was greater than 0.0002.

trees are generated, and the average[18] of their predictions is used as the random forest prediction. Each decision tree in the forest is generated by resampling the dataset (with replacement) at the unit of analysis and randomly selecting a subset of features (without replacement).[19] This process essentially bootstraps estimates from many decision tree models. In addition to decision tree hyperparameters, random forests can also be tuned by adjusting the size of the resampled dataset or the selection of the features used to construct the trees. Random forests usually outperform a single decision tree because the latter is less sensitive to idiosyncrasies of the training data and to the choice of features selected.[20]

4.3 Neural network (machine learning method 3)

Artificial neural network algorithms have attracted growing attention in recent years and have driven recent improvements in technologies such as image and voice recognition, translation, and autonomous vehicles. Figure 4 displays the general structure of an artificial neural network, which consists of a layer of input nodes (the covariates of the data), each connected to the nodes in one or more middle layers ("activation functions") which, in turn, are each connected to the output layer of nodes (predictions). Each connection between nodes represents a weight. The input data passes through the highly flexible functional form of a network of weights and activation functions that outputs a predicted value. With enough layers and nodes, a neural network could, in principle, approximate any hypothesis, regardless of its complexity. Neural networks can be highly complex and contain many hyperparameter choices. Though there is a rich literature on optimal hyperparameter tuning for neural networks, detailed discussion of it is beyond the scope of this

[18] Technically, a "consensus" is for random forest classification; "average" is for random forest regression.
[19] In our case, using panel data, the dataset is sampled at the employee level rather than the observation level.
[20] Although random forests often use decision trees built from randomly selected subsets of features at each split, we found that the best performance was achieved when all the features were used. After tuning the model by trying many combinations of possible hyperparameter values, our best performing random forest model used 100 decision trees, each with a maximum tree depth of 3, a maximum of 10 leaf nodes, and a minimum of 100 samples per leaf.

paper. Our purpose is simply to provide a basic foundation for understanding the method.[21] The Online Appendix demonstrates how we construct and tune a neural network model.

Insert Figure 4 about here

---

5. SELECTION OF MACHINE LEARNING ALGORITHMS

There is no secret recipe for selecting the best algorithm for the problem at hand, which can be more of an art than a science. In practice, even experienced data scientists do not know *ex ante* which algorithm to use, and it is commonplace to set up a "pipeline"[22] to estimate many models using various algorithms and choose the best performer. Nevertheless, we provide tentative guidance in this section. Table 3 lays out the strengths and typical uses of some of the most popular ML algorithms. Considerations for algorithm selection include:

Regression or classification? It is important to distinguish early on whether a prediction problem is a regression problem (a continuous real-number dependent variable, such as stock price) or a classification problem (a categorical dependent variable, such as filing for bankruptcy).[23] Some algorithms are suitable for both types of problems; they are simply used to minimize different loss functions. For example, a decision tree classifier minimizes the log-loss function, but a decision tree regression minimizes mean squared error.[24] However, using a decision tree classifier for a

---

[21] After hyperparameter tuning, our best model used a fully connected neural network with two hidden layers (with 32 nodes in the first layer and 12 nodes in the second). ReLUs (rectified linear units; f(x)=max(0,x)) were used as activation functions. The network is regularized using an L2 penalty term ($\lambda = 10^{-2}$) in the loss function. Because solving neural networks is a more complicated optimization problem than the other models, we also specified some optimization parameters. We used the Adam (Kingma & Ba, 2014) optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate = 0.001). As is typical of neural networks (and ML in general), we do not know why certain hyperparameter choices produced the best model; we simply tried different hyperparameters and regularization until we were able to find the model that performed best on the validation data.

[22] A pipeline is an automatic sequence of data processing and analysis steps that enables rapid implementation of any model. For example, implementing a pipeline could allow you to simultaneously clean the data, run cross-validation with many combinations of possible parameter choices, and fit the model.

[23] Classification problems may entail two categories (binomial classification) or more than two categories (multiclass classification).

[24] Mean squared error is $\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2$. It is technically possible to define custom loss functions, but that topic is beyond the scope of this paper. We merely want to clarify that many models can be used for both regression and

continuous dependent variable or a decision tree regression for a categorical dependent variable will not produce optimal results.

Insert Table 3 about here

(For classification problems) Linear separability. Some algorithms, such as support vector machines (SVM), are used to minimize a loss function (the hinge loss function[25]) that is designed to isolate points of one class from points of another class by the widest possible margin. This method is used to predict classification labels for data in which classes are linearly separable (that is, in which it is possible to draw a line that separates the classes). Strategy research that attempts to find the best line of separation between groups might benefit from such methods. In contrast, algorithms that use the log-loss function are relatively well suited to our data, in which turnover observations ($y = 1$) overlap significantly with non-turnover observations ($y = 0$).

(For classification problems) Labels or predicted probabilities? Although predicting a label $\hat{y}$ is a common task in practice (e.g., this loan will default), researchers using ML classification for exploratory data analysis may be more interested in the model's predicted probabilities $h_\theta(\boldsymbol{x})$ (e.g., this loan has a 0.26 probability of defaulting). Predicted probabilities are translated into a label using a decision threshold (e.g., loans with $h_\theta(\boldsymbol{x}) > 0.1$ will be labeled default). Algorithms that use the log-loss function are more appropriate for probabilistic interpretation than, for example, SVMs (which excel at hard categorization between distinct classes). In this paper's illustrative example, we stop one step before actually assigning a label using a decision threshold. It is nearly impossible to predict the exact time period when a specific person will leave TECHCO; instead, we compare the relative probabilities of turnover. (For more detail, see Section 6 on evaluating model performance.)

classification, depending on the loss function in question. In software implementations such as Python and R, implementing a model automatically minimizes a default loss function (often log-loss for classification and mean-squared error for regression).

[25] Hinge loss is $\sum_i \max(0, 1 - y_i \cdot f(x; \theta))$.

Consideration: functional form flexibility vs. size of data. As the number of observations grows, it becomes increasingly viable to use algorithms which are amenable to using highly flexible functional forms to achieve higher predictive performance without overfitting. For example, a neural network algorithm with a large number of layers and nodes can theoretically train a model that represents any nonlinear relationship, no matter its complexity. However, higher model complexity—characterized by higher numbers of layers and nodes—requires more data to prevent overfitting. A highly complex neural network trained on a few hundred observations would inevitably overfit the data and, thus, be relatively useless for both prediction and exploratory data analysis. By contrast, the functional form of a simpler decision tree is likely to give more reliable predictions for smaller data due to less overfitting.

Consideration: predictive performance/functional form flexibility vs. interpretability, computational speed, and ease of tuning. As a general rule of thumb, there is a trade-off between, on the one hand, functional form flexibility and predictive performance and, on the other hand, interpretability, computational speed, and how easy it is to tune the algorithm. For example, the logistic regression, which has linear coefficients, is highly interpretable. But, because it is not flexible in functional form, its predictive performance suffers in the case of nonlinear data. The decision tree model can be easily visualized, but does not perform as well as the more abstract random forest. For computational resources and speed, simple classifiers like Naïve Bayes require very little computation and, thus, excel on large datasets. Simplicity also enables the algorithm to perform well on small datasets due to not overfitting. But simplicity may sacrifice performance in some data—for example Naïve Bayes assumes features are independent, so it misses relevant information in the interactions between features (see Table 3 for more details on Naïve Bayes). By contrast, the neural network, which is computationally and data intensive, can represent highly interactive and nonlinear relationships. Finally, higher performing algorithms tend to be more difficult to tune; neural networks and gradient

boosting trees (see Table 3 for some details on gradient boosting trees) can be high performing but can require expert understanding of tuning the algorithm, to prevent overfitting.

6. EVALUATION OF MODEL PERFORMANCE: PLOTTING TRAINING AND VALIDATION LOSS, CONFUSION MATRIX, AND ROC CURVES

Evaluating the predictive performance of each model is essential not only for prediction problems, but also for evaluating how suitable each model is for exploratory data analysis and pattern recognition. *Before being used for exploratory data analysis*, models should be optimized to perform well. Once they are optimized using the process of regularization and cross-validation laid out in Section 3, the final predictive performance of the model can be reported by using simple metrics such as the loss of the model on the holdout test set or by using visualizations. Here we provide three visual tools that use different metrics for evaluating predictive performance. It is important to remember that performance is relative between models on a given dataset. For some datasets, it may be possible to perfectly predict outcomes, but for others, even the best model may appear to perform poorly (e.g., the extreme case of predicting totally random coin flips).

6.1 Plot: training and cross-validation loss

Figure 5 shows the predictive performance of each model relative to a logistic regression (with time fixed effects). The position of points on the plot indicates the training and cross-validation loss of each model; error bars are calculated from the variation among *k*-folds.[26] The figure helps us clearly visualize which models may be overfitted; anything above the dashed line has higher validation loss than training loss (relative to the logistic regression), a sign of overfitting to the training data.

Insert Figure 5 about here

---

[26] Because fluctuations in performance for each model are in part correlated with variation in samples, it is instructive to plot performance gains with respect to a baseline model—in this case, the logistic regression. By comparing *differences* in performance, the fluctuations associated with idiosyncrasies in the sample partially cancel, resulting in a stronger signal for performance gains with respect to the baseline.

The random forest and decision tree algorithms both offer small but statistically significant performance increases over the baseline logistic regression. The explanation for such small gains in performance is that meaningful interactions and nonlinearities among covariates are present only in a small subset of the data, and turnover events ($y = 1$) significantly overlap with non-turnover events ($y = 0$). In this case, these ML models are more useful for pattern detection than for predictive performance gains.

6.2 Confusion matrix

It should be noted that when ML predictions are used to make classification decision rules (e.g., assigning each observation a label $\hat{y} = 1$ or $\hat{y} = 0$ based on a decision threshold such as $h_\theta > 0.5$), it is important to use "accuracy" (correctly classified observations divided by total observations) as a prediction performance metric with caution. Especially when classifications in the outcome variable are imbalanced, accuracy gives a false impression of high predictive performance. For example, in a population for which only 1 percent of observations are blue and 99 percent are red, a model that always predicts red will be highly accurate at 99 percent. A better way to evaluate predictive performance is to use a confusion matrix—a two-by-two matrix displaying the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Performance metrics, such as the "F1 score," provide simple metrics that can balance precision (TP/(TP + FP)) and recall (TP/(TP + FN)). However, using a confusion matrix to evaluate the models in this paper would be a misapplication of the tool, which is meant to evaluate actual classifications at a specific decision threshold (not to compare probabilities).[27]

[27] In this paper, we are not interested in hard classification decision rules—that is, drawing hard boundaries between groups in order to assign labels (e.g., binary predictions such as $\hat{y} = 1$ or $\hat{y} = 0$). We are interested in the predicted probabilities h_θ that allow us to compare the relative hazards of turnover for different employees at different times. In our data, the predicted probability for a given choice of covariates has an intuitive definition: it is simply the density of turnover events within a given time interval. As the interval becomes smaller, the imbalance between *Quit* and *Not Quit* cases must necessarily increase and, therefore, the density of *Quit* events must decrease. This pattern is simply attributable to the data representation: the number of *Quit* cases is by definition fixed, but the number of *Not Quit* cases

6.3 Plot: ROC curve

Like the confusion matrix, the receiver operating characteristics (ROC) curve helps illustrate true and false positives and negatives rather than mere accuracy. However, the ROC curve uses predicted probabilities h_θ rather than classifications. Specifically, the plot illustrates the true positive rate (TP/(TP+FN)) and false positive rate (FP/(FP+TN)) as the decision threshold is varied from $h_\theta >= 1$ (no one is classified as positive) to $h_\theta >= 0$ (everyone is classified as positive).[28] Figure 6 presents the ROC curve of the random forest model as an example.

Insert Figure 6 about here

Intuitively, a model that randomly classified each point would produce points along the dotted 45° line (equally likely to classify someone as a true or false positive). The closer to the top left corner, the better the predictive performance. Any point below the line represents a classification that is worse than random. A perfect predictive model would have a point in the very top left corner, representing a model that gave no false negatives and no false positives. Although summarization sacrifices information about the trade-offs of true positive and false positives along the curve, the area under the curve (AUC) can be used to summarize the ROC curve. The AUC score is the probability that the model will assign a randomly chosen positive observation a higher h_θ than a randomly chosen negative one. In Figure 6, the random forest model has an AUC score of 0.73. For our data, this is probably close to the highest score possible with any model because there

grows arbitrarily as the observation time interval is shortened. Thus, a class imbalance is an artifact of the survival data representation. Although there are a variety of techniques to address class imbalances in ML, such as minority class resampling, their use in the context of survival analysis is not recommended. Resampling, for example, artificially inflates the representation of the minority class (in this case *Quit*) and, therefore, would lead to an inflated estimate for the predicted probability h_θ. Our data would yield a trivial confusion matrix. Given a decision threshold of 0.5 ($\hat{y} = 1\ if\ h_\theta > 0.5$), there are no predictions for $\hat{y} = 1$ because the likelihood of turnover in each arbitrarily chosen time interval is so low (and, in fact, can be made arbitrarily small by decreasing the time interval).

[28] Each point on the curve could be represented as its own confusion matrix.

is no hard boundary between *Quit* and *Not Quit* observations (i.e., the underlying hazard of turnover is less than 1 for all observations, rather than 1 in some regions and 0 in other regions).

7. INTERPRETATION AND VISUALIZATION OF PATTERNS IN DATA: PARTIAL DEPENDENCE PLOTS AND HEAT MAPS

While the decision tree model is highly visual in demonstrating nonlinearities in the data (recall Figure 1), not all ML-trained models offer the same ease of visualizing the model's structure. For example, a random forest model cannot be visualized as a tree because its estimates are taken from an ensemble of many trees. Although the model itself can be difficult to visualize, we can interpret models based on the how their predictions vary when we input ranges of covariate values into the model. This section discusses two tools for visualizing and interpreting any supervised ML model based on how its predictions vary with covariate values: partial dependence plots and heat maps.

7.1 Partial dependence plots

Partial dependence plots (PDPs) display how a predicted outcome changes in response to changes in a single covariate, while holding the values of the other covariates constant. For example, the PDPs in Figure 7 show how turnover hazard changes as a function of the covariates *Time*, *Training Performance*, and *Location Age*.[29]

Each plot was generated by considering 500 randomly sampled observations. For each sampled observation, h_θ (probability of employee turnover) was plotted as a function of the covariate of interest while holding all other covariates for that observation fixed. The predictions from each observation across the range of the covariate's values are represented by orange (solid) lines. The result is a distribution of 500 orange lines, one for each sampled observation. Each plot also shows the average h_θ of all the samples drawn (the dotted blue line).

[29] Although we created a PDP for each covariate in each model, we selectively display only these three covariates to demonstrate the differences between the logistic and random forest models.

Visualizing the effects reveals an interesting pattern: the partial dependence plots for the random forest model reveal nonlinear predictions. In this model, *Training Performance* was unrelated to turnover hazard, except for a sharp discontinuous jump in hazard for those with scores lower than 4.0. Figure 7 displays these results side by side with results for a logistic regression with time fixed effects (to be discussed in Section 8).

Insert Figure 7 about here

7.2 Heat maps

Another way to interpret models is to represent predicted probabilities as color (heat) on a two-dimensional space as a function of two covariates, while holding the other covariate values fixed. The disadvantage of this approach is the difficulty of visualizing the consistency of the predictions across observations with different covariate values (as was represented by the distribution of orange solid lines in the partial dependence plots). The advantage of heat maps relative to partial dependence plots is that they provide visually striking examples of the nonlinear *interactions* between covariates—in our case between *Training Performance* and *Time*. Using heat maps with a covariate on each axis, it is possible to consider not only the isolated nonlinear effects of single covariates, but also how the effect of that covariate depends on another.

Figure 8 displays the predicted probability of employee turnover h_θ for each model, as a function of *Training Performance* and *Time*.[30] It plots the prediction h_θ for each combination of points for 40 values of *Training Performance* and 40 values of *Time* (1600 total), holding other covariate values fixed at their average values. Several interesting patterns are revealed by using the heat maps. In the decision tree heat map, a conspicuous line separates *Training Performance* <= 3.995 from *Time* <=

[30] We also constructed heat maps for other combinations of covariates (not included) and found only noninteractive and linear effects.

6.5, exactly as in the depiction of a decision tree in Figure 1. The random forest model offers similar insights—that the hazard of turnover tends to increase over time, and that hazard is much higher for those with a training score below about 4, especially around 6 months. We see that the negative effect of training score on turnover estimated by the logistic model must have been driven by the narrow yellow strip of employees with training scores below 4 at about month 6. Compared to the decision tree and random forest predictions, the neural network predictions vary smoothly across the feature space and are not characterized as constants in each square region; however, because of the limited amount of data, this model may not be as reliable as it would have been if it were trained with more observations.

Insert Figure 8 about here

---

8. USING ECONOMETRIC METHODS TO ESTIMATE LOCAL LINEAR RELATIONSHIPS IDENTIFIED BY ML

Although the machine learning tools in this paper have already provided valuable insights about relationships in our data, they do not provide consistent coefficient estimates or reliable coefficient standard errors, both of which are necessary for statistically testing hypotheses in the data (Mullainathan & Spiess, 2017). To obtain consistent coefficient estimates and standard errors, we can use a traditional econometric model to estimate local linear relationships in heterogeneous subsets of the data identified by the ML algorithms. Estimating linear models around local regions of interest to interpret local relationships in complex global ML models is an increasingly popular method in the ML literature (see the LIME project of Ribeiro, Singh, & Guestrin, (2016)).

In the first section of this paper, we demonstrated how we modified a naïve logistic regression model to test heterogeneous effects identified by the decision tree (recall Table 2). This was a stylized oversimplification—in practice, the decision tree can be highly sensitive to slight

variations in hyperparameter tuning, feature engineering, or sample. Using multiple algorithms and visualization tools can add a measure of robustness. We employed two visualization tools—partial dependence plots and heat maps—to examine the predictions of our three ML models.

A striking insight from Figure 8 is the relatively poor model of the relationships among *Training Performance, Time,* and the hazard of turnover depicted in the logistic regression heat map. The other models all capture the dramatically higher turnover probabilities predicted for employees with low training scores during their first months on the job. The linear logistic regression model, however, can produce only a simple linear hypothesis: that those with lower *Training Performance* tend to have a higher hazard of turnover—with constants added for each time interval from the *Time* fixed effects. But that linear hypothesis does appear to represent reality: in the other models, the training scores did not appear to significantly affect hazards for any of the subjects but those with *Training Performance* <= 3.995 and *Time* = 6. This is the case because *Training Performance* was not modeled as a function of *Time* in the logistic regression linear model; it was assumed for all covariates that the effect β did not vary over time. We can change this assumption using modifications such as splitting the data and adding a dummy variable, as demonstrated in Table 2.

9. DISCUSSION AND CONCLUSION

This paper began with the goal of demonstrating how to use supervised ML methods to identify meaningful nonlinear and interactive patterns in data as a complement to traditional regression analysis. The advantage of ML over traditional regression-based inference is that rather than trying to force a series of researcher-specified models on the data, ML algorithms induce models from the data. ML algorithms are able to fit complex models to the data while balancing the bias-variance trade-off to generalize to out-of-sample data. We provide a step-by-step how-to-use-ML-methods framework, provide some guidance on choosing among ML methods, implement three common

ML algorithms (decision tree, random forest, and neural network), and present plots (using data on employee turnover) that readers can use to assess the resulting predictive models. We use insights from these plots to estimate local linear approximations of the models. Local linear estimation using traditional econometric methods yields interpretable and consistent coefficient estimates, as well as reliable standard errors for testing hypotheses.

When applied to our data, insights from the heat map representations of the ML models helped us estimate local linear effects for employees with different training performance scores in different time periods. This process revealed that employees with training scores under 3.995 were much more likely to leave, but only during the first six months on the job; thereafter, training scores had no effect on that group. For the vast majority of the dataset (employees with training scores above 3.995), higher scores actually had a significantly positive effect on turnover after the first six months. This positive effect was directly at odds with the negative global effect estimated by a naïve global linear fit of the data.

By providing these tools and examples, we hope to encourage the practice of using ML in empirical research and presenting ML models and visualizations in research papers. To this end, we provide conceptual guidance on how to implement ML and how to select the right algorithm for the job; we also provide several practical visual tools for purposes of model evaluation (loss plots, confusion matrix, and ROC curve) and interpretation (partial dependence plots and heat maps). By increasing ML interpretability, we encourage researchers to use ML not only for prediction, but also to develop better understanding of the complex phenomena they study.

However, because ML algorithms do not necessarily produce statistically consistent coefficient estimates or reliable standard errors, they still cannot be used for traditional deductive hypothesis testing (Mullainathan & Spiess, 2017). Thus, we present ML as a complement to, not a

substitute for, traditional econometric methods. This paper has demonstrated that we can approximate pieces of the complex global ML model by means of local linear estimation using traditional econometric techniques. ML allows researchers to observe complex patterns that may have gone unnoticed (see also Rihoux & Ragin, 2008; Cattaneo et al., 2019), but for now, traditional econometric estimation should be used for formal hypothesis testing. This situation may change; using ML for more complicated causal analysis is a cutting-edge methodological topic in statistics and economics (Athey & Imbens, 2015; Athey, 2017). It is also important to recognize that when using ML, causal claims can be made only if there is some source of exogenous variation—exactly the same limitation that characterizes traditional models. Given this, ML methods are *not* a substitute for methods aimed at causal inference. Researchers using these methods should be careful to avoid causal interpretations based on correlations.

ML methods have other important limitations. Importantly, data can be inherently biased, especially when the data is selected based on convenience (Cowgill & Tucker, in press) or suffer from input incompleteness (Choudhury, Starr, & Agarwal, 2018). Moreover, predictions can be highly sensitive to algorithm specifications. When two covariates are highly correlated, they contribute largely the same information to a model; thus, a slight change in sample or hyperparameter may mean that the algorithm chooses to use one covariate or the other in the model.[31] In our data, for example, if the employees' *Verbal Score* had been highly correlated with *Training Performance*, some models might have revealed significant nonlinearities in *Verbal Score* and none in *Training Performance*. Like traditional econometric models, researchers may settle on a poor model or even data mine and engage in feature engineering to find empirical relationships that

[31] Even simply retraining some models could produce a new result, since fits for some models are performed stochastically, using a random point in parameter space as the starting point.

confirm whatever they are looking for. To avoid these pitfalls, we suggest three best practices: (1) investigate the biases (e.g., selection effects) that are inherent in each dataset; (2) be sufficiently familiar with the research context to understand which among correlated covariates may be actually driving the predictions; and (3) compare results from using multiple ML algorithms and several models for each algorithm, and, if possible, use various measures to avoid misinterpreting a single idiosyncratic model fit.

As an area for future research, many applications of "unsupervised learning" techniques may also be extremely valuable for creating covariates useful for analysis (Kaplan & Vakili, 2015; Choudhury et al., 2019; Menon et al., 2018; Furman & Teodoridis, 2018) and/or for developing grounded hypotheses from quantitative data. This paper uses only supervised learning techniques—that is, techniques that try to correctly classify data with a labeled dependent variable. For example, clustering may help identify distinct concepts in text, and principal-component analysis could help reveal the most important factors driving a phenomenon.

In conclusion, we argue that ML-based methods will be an important empirical complement to traditional econometric techniques. ML follows the tradition of grounded theory by helping researchers identify patterns in the data and build models based on data rather than relying exclusively on existing literature or preconceived notions (Glaser & Strauss, 1967; Eisenhardt, 1989). In the age of big data, where the empirical researcher is faced with an ever-increasing trend in the number of observations and covariates, ML methods represent a new opportunity to develop more nuanced testable hypotheses from quantitative data by helping researchers find *robust* (across algorithms) nonlinear or interactive relationships in quantitative data. The visual tools related to evaluation and interpretation of ML models presented in this paper will also help researchers better understand their data and select ML models accordingly.

REFERENCES

Athey, S. (2017). Beyond prediction: Using big data for policy problems. *Science, 355*(6324), 483–485.

Athey, S., & Imbens, G. W. (2015). Machine learning methods for estimating heterogeneous causal effects. Working Paper No. 3350, Stanford Graduate School of Business, Stanford, CA. Retrieved from https://www.gsb.stanford.edu/faculty-research/working-papers/machine-learning-estimating-heretogeneous-casual-effects

Autor, D. H. (2015). Why are there still so many jobs? The history and future of workplace automation. *Journal of economic perspectives* 29.3: 3-30.

Cattaneo, M. D., Crump, R. K., Farrell, M. H., & Feng, Y. (2019). On Binscatter. Working Paper. Retrieved from http://arxiv.org/abs/1902.09608

Choudhury, P., Starr, E., & Agarwal, R. (2018). Machine learning and human capital: Experimental evidence on productivity complementarities. Working Paper, Harvard Business School, Boston, MA. Retrieved from https://www.hbs.edu/faculty/Publication%20Files/18-065_c065462c-0791-4356-8e09-46e1b251c1c8.pdf

Choudhury, P., Wang, D. Carlson, N. A., & Khanna, T. (2019). Machine Learning Approaches to Facial and Text Analysis: Discovering CEO Oral Communication Styles. Working Paper

Cowgill, B., & Tucker, C. E. (in press). Economics, fairness and algorithmic bias. *Journal of Economic Perspectives*.

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of Management Review, 14*(4), 532–550.

Furman, J. L., & Teodoridis, F. (2018). Automation, research technology, and researchers' trajectories: Evidence from computer science and electrical engineering. Working Paper. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3285286

Glaser, B. S., & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research*. New Brunswick, NJ: AldineTransaction.

Gross, D. (2018). Creativity under fire: The effects of competition on creative production (NBER Working Paper No. 25057). Retrieved from National Bureau of Economic Research website: https://www.nber.org/papers/w25057

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: Data mining, inference, and prediction*. New York: Springer.

Hirsh, S. M., Brunton, B. W., & Kutz, J. N. (2018). Data-driven spatiotemporal modal decomposition for time frequency analysis. Working Paper, University of Washington, Seattle, WA. Retrieved from https://arxiv.org/pdf/1806.08739.pdf

Kaplan, S., & Vakili, K. (2015). The double-edged sword of recombination in breakthrough innovation. *Strategic Management Journal, 36*(10), 1435–1457.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. Working Paper. Retrieved from https://arxiv.org/abs/1412.6980

Mantree, S., & Ketokivi, M. (2013). Reasoning in organization science. *Academy of Management Review*, *38*(1), 70–89.

Menon, A., Choi, J., & Tabakovic, H. (2018). What you say your strategy is and why it matters: Natural language processing of unstructured text. In G. Atinc (Eds.), *Academy of Management Proceedings*. Retrieved from https://journals.aom.org/doi/10.5465/AMBPP.2018.18319abstract

Microsoft. (2019). How to choose algorithms. Retrieved from https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-choice

Mullainathan, S., & Spiess, J. (2017). Machine learning: An applied econometric approach. *Journal of Economic Perspectives*, *31*(2), 87–106.

O'Neil, C. (2016). *Weapons of math destruction: How big data increases inequality and threatens democracy*. New York: Crown Publishers.

Puranam, P., Shrestha, Y. R., He, V. F., & von Krogh, G. (2018). Algorithmic induction through machine learning: Opportunities for management and organization research. (INSEAD Working Paper No. 2018/11/STR). Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3140617

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should I trust you? Explaining the predictions of any classifier. Working Paper. Retrieved from https://arxiv.org/abs/1602.04938

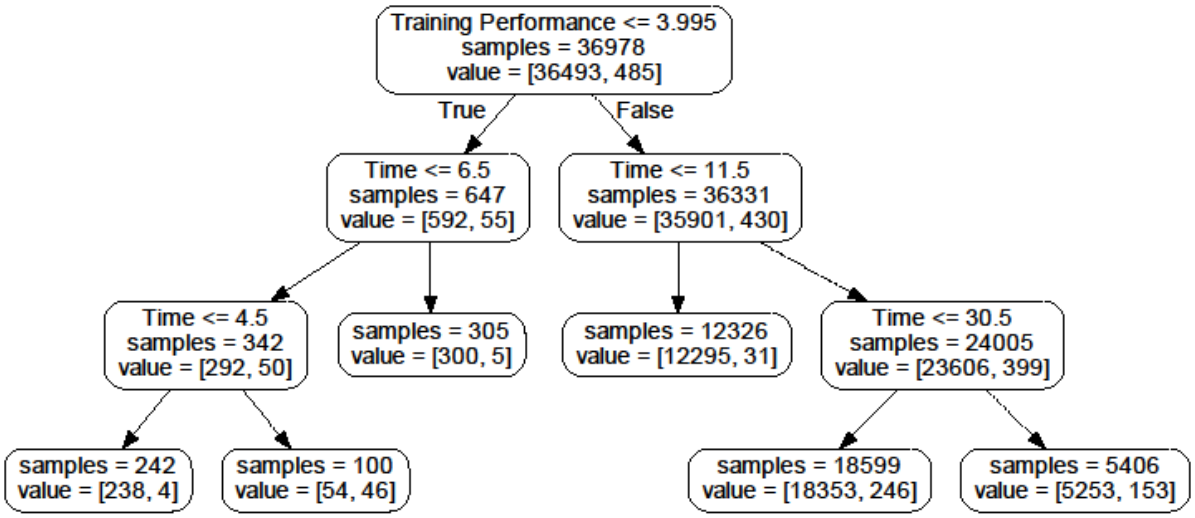
Rihoux, B., & Ragin, C. C. (Eds.). (2008).*Configurational comparative methods: Qualitative comparative analysis (QCA) and related techniques*. Thousand Oaks, CA: SAGE Publications.

Rudy, S., Alla, A., Brunton, S. L., & Kutz, J. N. (2018). Data-driven identification of parametric partial differential equations. Working Paper. Retrieved from https://arxiv.org/abs/1806.00732

Ruff, C. T., Lacoste, A., Nordio, F., Fanola, C. L., Silverman, M. G., Argentinis, E., Spangler, S., & Sabatine, S. (2017). Classification of cardiovascular proteins involved in coronary atherosclerosis and heart failure using Watson's cognitive computing technology. *Circulation, 136*(S1), A16678.

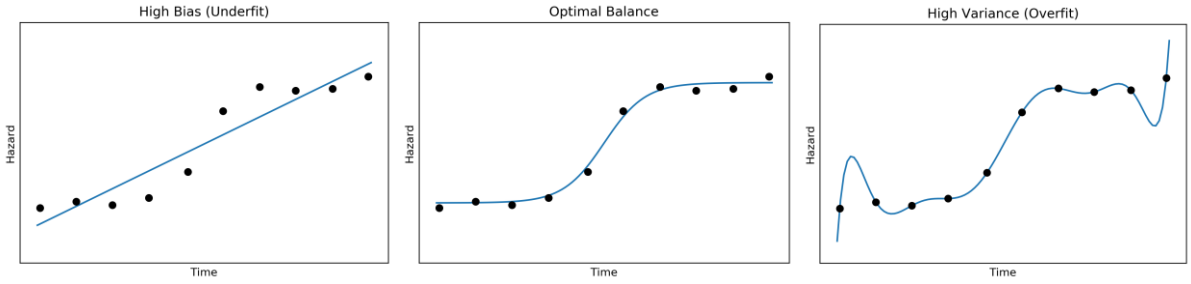
Scikit Learn. (2018). Choosing the right estimator. Retrieved from https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Van Maanen, J., Sørensen, J. B., & Mitchell, T. R. (2007). The interplay between theory and method. *Academy of Management Review*, *32*(4), 1145–1154.



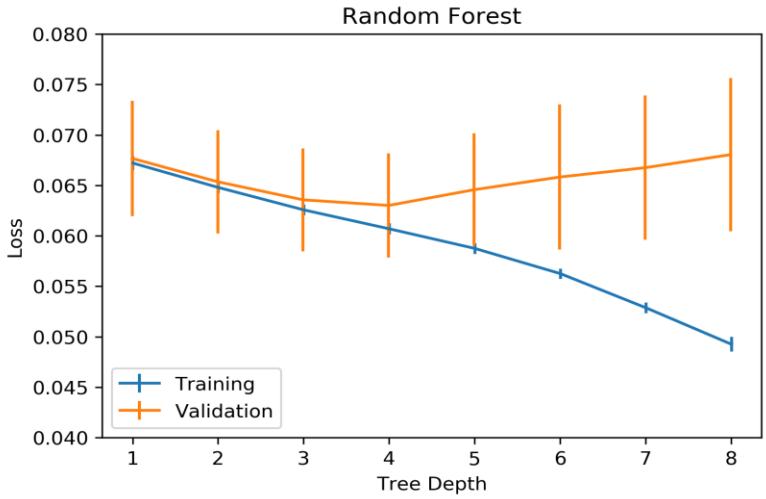
Notes: This image was created by graphviz in python (see Online Appendix). The model was trained by a decision tree algorithm to predict the dependent variable $Quit$ (1 if turnover occurs in a given time interval, 0 otherwise). The label *Samples* within each node denotes the number of observations within the subset of data represented by that node. For example, the top "root node" contains all 36,978 observations. Adding all the values of *Samples* in the terminal "leaf nodes" also sums to 36,978. The label *Value* within each node denotes the number of non-turnover events and the number of turnover events within that node. For example, the terminal node on the bottom left reads *value*=[238,4]. This indicates that of the 242 observations, 238 were *Quit* = 0 and only 4 were *Quit* = 1. The probability of turnover for these observations can be modeled by a constant value, 4 / 242 = 0.0165.

FIGURE 1. Decision tree model



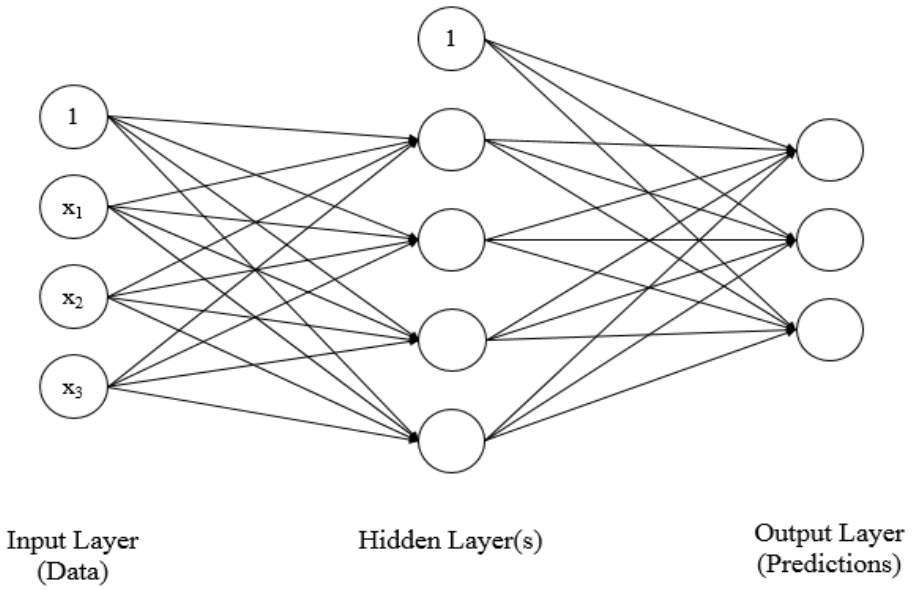
Notes: ML excels at balancing the bias variance trade-off. The left panel illustrates a high bias/low variance fit of the points, the right panel illustrates a high variance/low bias fit, and the middle represents a reasonable balance between bias and variance.

FIGURE 2. Illustrative depiction of bias-variance trade-off in one dimension



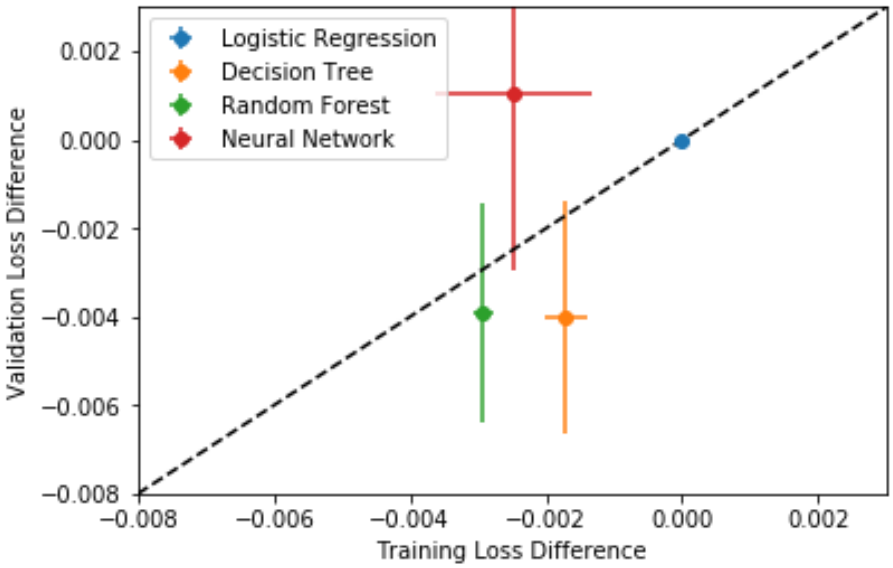
Notes: The blue (lower) and orange (upper) lines represent training loss and validation loss for each value (1–8) of the tree depth hyperparameter. Error bars represent cross-validation at 95 percent confidence intervals. The figure demonstrates the "elbow" where validation loss diverges from the training loss. Increasing the tree depth removes constraint from the model, allowing it to better describe the data and thereby decrease loss. However, removing too much constraint (i.e., increasing tree depth) can cause overfitting—where the model predicts well in the training data but not in the validation data. It appears that setting the tree depth hyperparameter to 3 or 4 would yield the best generalizable predictions. Note that for illustrative purposes, this random forest algorithm does not contain any regularization other than tree depth. The random forest algorithm used later in the paper differs slightly due to adjustment of other regularization hyperparameters.

FIGURE 3. Training and validation loss as a function of the random forest "tree depth" hyperparameter



Notes: The general structure of an artificial neural network consists of an input layer of neurons (the data), a middle layer (there can be multiple layers), and an output layer (the predicted probabilities); each node of each layer is connected by weights to each node of the next layer. The sum of the weighted data from the input layer is passed through to some function (an "activation function") in each node of the hidden layer. Then the output from each such node is summed and passed to an activation function in each node of the output layer. The output layer nodes are the model's predictions. The depicted network represents a multiclass classification problem;[32] for a single binary dependent variable like ours, the output layer consists of only one node. The weights of the nodes and layers are chosen to produce predictions in the output layer that minimize loss of the loss function—usually the log-loss function.[33] In contrast to the logistic regression, neural networks contain more layers and nodes. With enough layers and nodes, a neural network could, in principle, approximate any hypothesis, regardless of its complexity. The ever-present problem to avoid when implementing neural networks is overfitting. To avoid overfitting, hyperparameters such as the regularization term in the loss function, node connectivity (i.e., randomly dropping some nodes in cross-validation), number of layers, and number of nodes can all be adjusted.

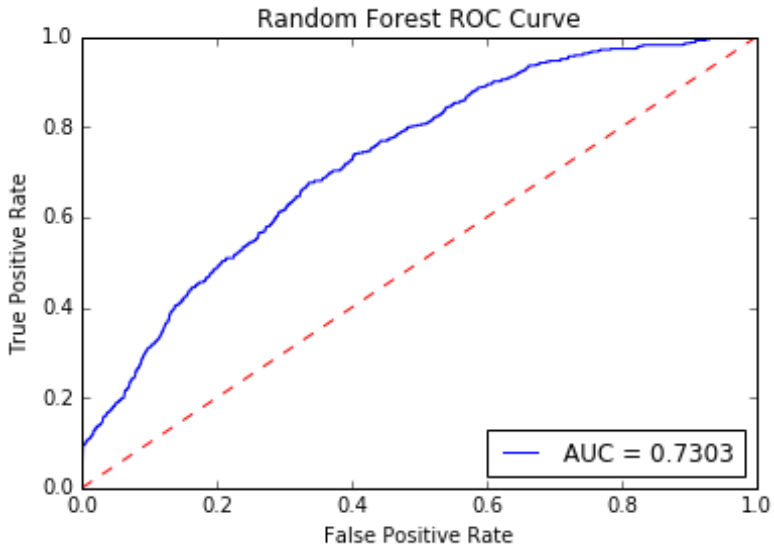
FIGURE 4. Neural network structure



Notes: This plot compares the models' predictive performance to a baseline model, the logistic regression**.** Each point represents the average *difference* in total training and validation loss compared to the logistic regression model, which is represented as the blue point (i.e., *loss from model – loss from logistic*). Taking the differences between models cancels out variation in the samples shared by all models, sharpening the contrast between models by removing noise. Points toward the lower-left corner are better predictions because they have lower loss than the logistic regression. Error bars represent variation in predictions yielded by the k-folds cross-validation. The dashed line represents a point at which the loss from the training set is equal to the loss on the validation set (relative to the baseline logistic regression). Anything above and to the left of the line would mean that validation loss is higher than training loss, indicating that the model is probably overfitted on the training data relative to the baseline logistic regression.

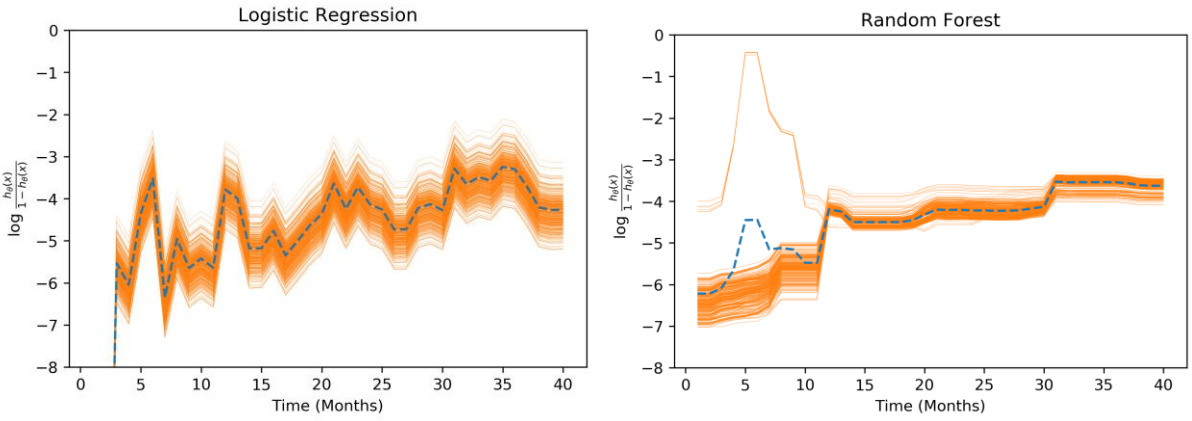
FIGURE 5. Plots of training and validation loss for each model relative to logistic regression

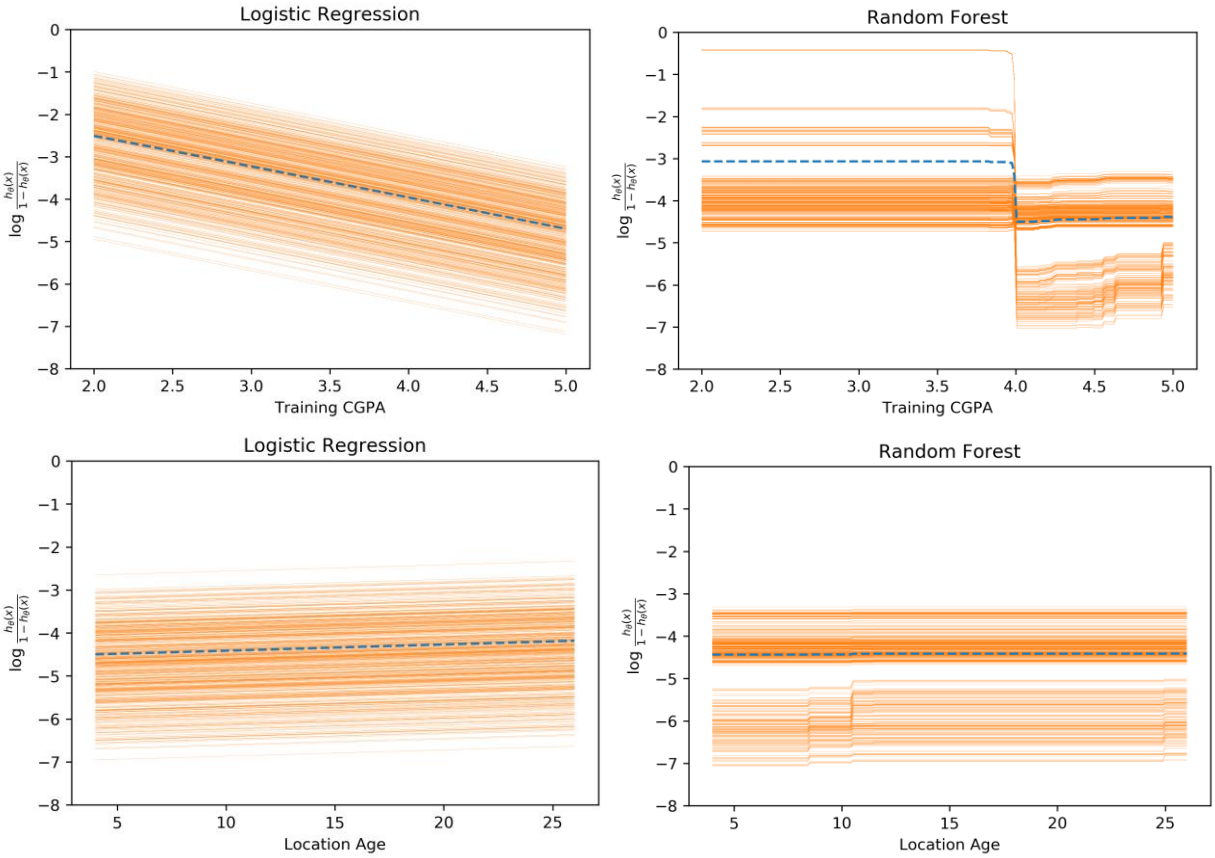
[32] For multiclass classification, use the cross-entropy loss function—the more general form of the log-loss function.
[33] The optimization process is more difficult than it is for a logistic regression and is solved using what is known as "the backpropagation algorithm"—essentially, an application of the "chain rule" for taking derivatives of nested functions.



Notes: The blue (solid) line plots the true positive rate (TP/(TP+FN)) vs. the false positive rate (FP/(FP+TN)) as the decision threshold is varied from $h_\theta \geq 1$ (no one is classified as positive) to $h_\theta \geq 0$ (everyone is classified as positive). The red dotted 45° line represents a model that randomly classifies each point (that is equally likely to classify someone as a true or false positive). The closer to the top-left corner, the better the predictive performance. Any point below the line represents classification that is worse than random. A perfect predictive model would have a point in the very top-left corner, representing a model that gave no false negatives and no false positives. The area under the curve (AUC score) of 0.73 means that there is a 0.73 probability that the random forest model will rank a randomly chosen positive observation higher than a randomly chosen negative observation. For our data, this is probably close to the highest possible score with any model because there is no hard boundary between *Quit* and *Not Quit* observations (i.e., the underlying hazard of turnover is less than 1 for all observations, rather than 1 in some regions and 0 in other regions).

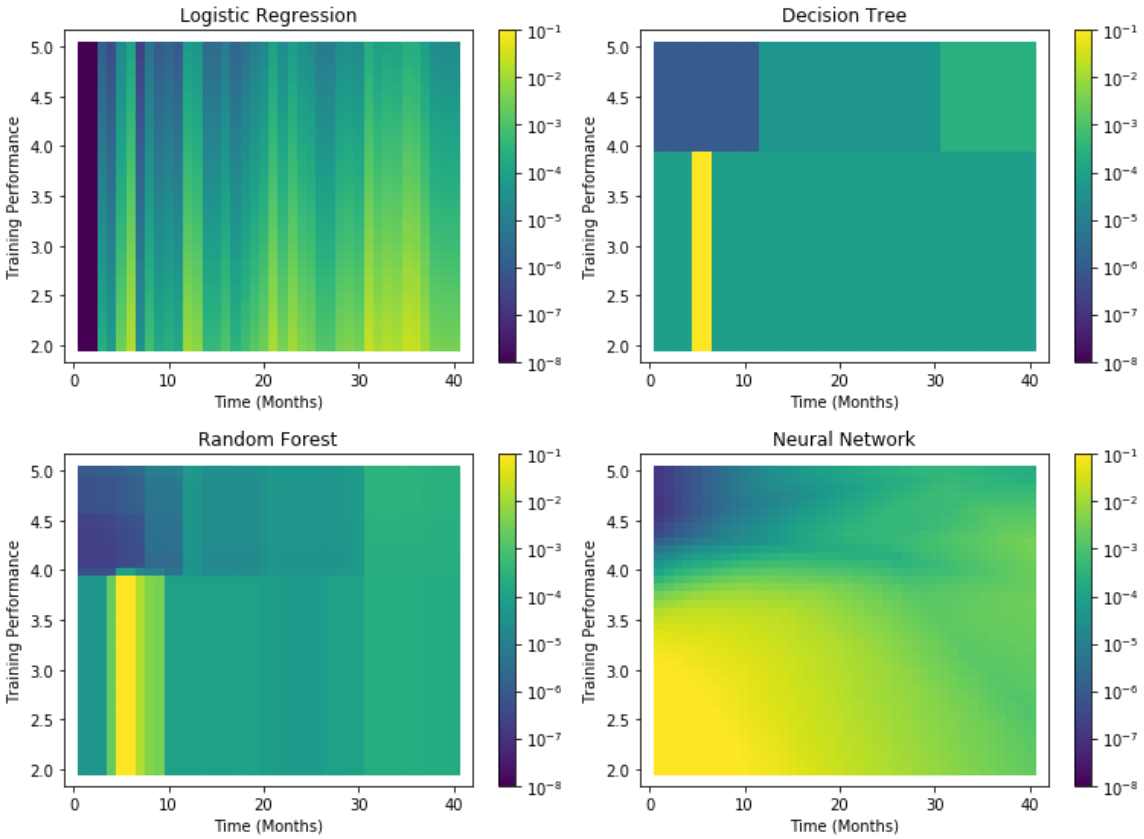
FIGURE 6. ROC curve





Notes: Logistic regression predictions appear in the left-hand plots; random forest predictions appear in the right-hand plots. Each vertical axis is on the same scale, with units as the log of the odds ratio of the predicted probability of an event. The plot for each variable was generated by randomly selecting 500 samples from the full dataset and, for each sample, predicting the outcome using 40 values of the variable across the entire variable range while holding all other variable values fixed. The predictions from each sample are represented by an orange (solid) line across the entire range. The result is a distribution of 500 orange lines in each plot, one for each sample. Each plot also shows the typical partial dependence of the hazard: the average over all the samples drawn (the dotted blue line). We produced such plots for each covariate; only three are included here due to space constraints. The other covariates were similar to the *Location Age* plot—a linear relationship with turnover hazard.

FIGURE 7. Partial dependence plots



Notes: These heat maps represent the probability of turnover predicted by each model along the dimensions *Training Performance* and *Time*. We used our models to predict the hazard of turnover for each point represented by each combination of 40 evenly spaced values of *Training Performance* and *Time* (1,600 points total). Other variable values are held constant at typical (average) values. All plots are on the same scale. Higher probabilities are represented in yellow; the lowest probabilities are represented in dark blue. We produced such plots for many combinations of covariates; only the two dimensions *Training Performance* and *Time* are included here due to space constraints. Other plots did not reveal meaningful interactions.

FIGURE 8. Heat map of predicted probability of turnover for *Training Performance* vs. *Time*

TABLE 1. Summary statistics

	N	Mean	SD	Min	Max
Has Quit	1,688	0.42	0.49	0.00	1.00
Time	1,688	30.70	10.10	1.00	40.00
Training Performance	1,688	4.49	0.43	2.00	5.00
Logical Score	1,627	4.72	3.67	-5.00	11.00
Verbal Score	1,627	4.27	3.98	-8.00	16.00
Average Literacy	1,580	76.78	8.50	45.70	95.80
Male	1,688	0.66	0.48	0.00	1.00
Production-Center Age	1,657	14.87	7.60	4.00	26.00
Distance	1,241	0.79	0.70	0.00	3.12
Language Similarity	1,240	60.04	35.13	1.25	100.00
Month Arrived	1,688	4.16	2.04	1.00	8.00

TABLE 2. Comparison of naïve logistic regression to local linear estimation informed by ML models

	Naïve Model	Local Linear Estimation of ML Insights	
		First 6 months	After 6 months
Dependent Variable: *Quit*	(1)	(2)	(3)
Training Performance	-0.887***	0.327	0.389*
	(0.182)	(1.174)	(0.181)
Low Training Performance		11.131*	1.850
		(5.552)	(4.144)
Low Training Performance x		-1.219	-0.291
Training Performance		(1.298)	(1.182)
Logical Score	-0.022	-0.037	-0.016
	(0.013)	(0.051)	(0.014)
Verbal Score	0.035**	-0.009	0.035**
	(0.012)	(0.043)	(0.013)
Average Literacy	0.015*	0.034	0.014*
	(0.006)	(0.022)	(0.007)
Male	0.038	0.260	-0.047
	(0.100)	(0.358)	(0.104)
Production-Center Age	0.014*	-0.005	0.017*
	(0.006)	(0.022)	(0.007)
Distance	0.115	0.250	0.091
	(0.100)	(0.470)	(0.106)
Language Similarity	0.004*	0.002	0.003
	(0.002)	(0.007)	(0.002)
Subset: First 6 Months	N/A	yes	No
Time Fixed Effects	Yes	yes	Yes
Month of Arrival Fixed Effects	Yes	yes	Yes
N	34,596	4,655	29,869

*p < 0.05, **p < 0.01, ***p < 0.001.

Notes: This table reports logistic regression results. The dependent variable is *Quit*, which takes a value of 1 if an employee quit in a given time interval and 0 otherwise. The first column *Naïve Model* simply includes a coefficient for each of our covariates. The second and third columns break up the sample and add a dummy variable and interaction effects to capture heterogeneous effects of different subpopulations. Column 2 reports results for observations during

the first six months on the job (*First 6 months*) and Column 3 reports results for a separate model run on observations after six months (*After 6 months*). These columns each include a dummy variable, *Low Training Performance* (1 for employees with *Training Performance* below 4 and 0 otherwise), and its interaction term with *Training Performance*. By making these modifications, we now allow the linear model to locally estimate the effect of *Training Performance* on turnover for two different time periods and compare effects for employees who had high and low training scores. The global effect in the first column is meaningfully different than the effects of the subpopulations in columns 2 and 3.

TABLE 3. Guidance for selecting supervised machine learning algorithms

Algorithm	(R)egression or (C)lassification?	Scale Features?	Performance	Interpretability	Speed	Easy to tune	Notes	Common Uses
*Decision Tree**	Both		◍	●	●	◍	Highly interpretable due to visualization of tree	Quick understanding of important features and partitions in data
*Random Forest**	Both		●	◍	◍	◍	Estimates trees in parallel; easy to tune and low memory footprint relative to gradient boosting trees	General purpose
*Neural Network**	Both	✓	●	○	○	○	Highly flexible functional form; difficult to tune and more reliable and useful with big data	Image recognition, language processing, forecasting, and more
K-nearest neighbors (KNN)	Both	✓	◍	◍	◍	◍	Lazy nonparametric estimation based entirely from values of *K* neighboring observations; high memory requirements; if used with panel data, time interval is a tunable hyperparameter	Useful when little is known about the distribution of data
Gradient Boosting Tree	Both		●	◍	○	○	Estimates trees sequentially; often outperforms random forest but harder to tune, slower, and more memory needed	General purpose high performance; especially good for unbalanced data
Support Vector Machine	Both	✓	◍	◍	●	●	Uses hinge loss function—good for drawing optimal boundaries between linearly separable classes; reliable with relatively few observations and many features	Image recognition (for example, character recognition) and text categorization
Naïve Bayes	C		◍	◍	●	●	Minimal structure; strongly assumes independence of features so cannot exploit interactions; scalable for large data and reliable with few observations	Multiclass classification; text classification, such as assigning emails to "spam" or "not spam"

* Demonstrated in this paper

○ Low ◍ Medium ● High

Notes: These comparisons are broad generalizations that may change frequently according to circumstance. The meaning of each column in the table is as follows: *(R)egression or (C)lassification*: the "R" indicates that the

algorithm is used for regression and "C" indicates the algorithm is used for classification. Many of these algorithms can be used for both. For example, when used for regression, the decision tree may be used to minimize the squared error (or "mean squared error") loss function; when used for classification, it may be used to solve the log-loss function. It is also possible to use other loss functions or even customize loss functions. *Scale Features*: the check indicates that the algorithm is sensitive to feature scaling—that is, features (covariates or functions of covariates) should be transformed to a standardized scale either by using "z-score" or "minmax." *Performance*: refers to the algorithm's capacity to achieve high predictive performance, usually by building a nuanced model with highly flexible functional form. *Interpretability*: loosely refers to how easy it is to conceptualize or interpret the algorithm's resulting predictive model. *Speed*: refers to computational speed of training a model (though, of course, these comparisons depend heavily on the data being used). *Easy to tune*: generally corresponds to fewer hyperparameter choices and models that do not require as much nuanced expertise to avoid overfitting. For similar resources, see Microsoft (2019) and Scikit Learn (2018).