REFRESHER READING

2021 CFA PROGRAM • LEVEL II • READING 7

Quantitative Methods

Machine Learning

by Kathleen DeRose, CFA, Matthew Dixon, PhD, FRM, and Christophe Le Lannou

Kathleen DeRose, CFA, is at New York University, Stern School of Business (USA). Matthew Dixon, PhD, FRM, is at Illinois Institute of Technology, Stuart School of Business (USA). Christophe Le Lannou is at dataLearning (United Kingdom).

LEARNING OUTCOMES

Mastery	*The candidate should be able to:*
☐	**a.** distinguish between supervised machine learning, unsupervised machine learning, and deep learning;
☐	**b.** describe overfitting and identify methods of addressing it;
☐	**c.** describe supervised machine learning algorithms—including penalized regression, support vector machine, k-nearest neighbor, classification and regression tree, ensemble learning, and random forest—and determine the problems for which they are best suited;
☐	**d.** describe unsupervised machine learning algorithms—including principal components analysis, k-means clustering, and hierarchical clustering—and determine the problems for which they are best suited;
☐	**e.** describe neural networks, deep learning nets, and reinforcement learning.

INTRODUCTION 1

Investment firms are increasingly using technology at every step of the investment management value chain—from improving their understanding of clients to uncovering new sources of alpha and executing trades more efficiently. Machine learning techniques, a central part of that technology, are the subject of this reading. These techniques first appeared in finance in the 1990s and have since flourished with the explosion of data and cheap computing power.

This reading provides a high-level view of machine learning (ML). It covers a selection of key ML algorithms and their investment applications. Investment practitioners should be equipped with a basic understanding of the types of investment problems that machine learning can address, an idea of how the algorithms work, and the vocabulary to interact with machine learning and data science experts. While investment practitioners need not master the details and mathematics of machine learning, as domain experts in investments they can play an important role in the implementation of these techniques by being able to source appropriate model inputs, interpret model outputs, and translate outputs into appropriate investment actions.

Section 2 gives an overview of machine learning in investment management. Section 3 defines machine learning and the types of problems that can be addressed by supervised and unsupervised learning. Section 4 describes evaluating machine learning algorithm performance. Key supervised machine learning algorithms are covered in Section 5, and Section 6 describes key unsupervised machine learning algorithms. Neural networks, deep learning nets, and reinforcement learning are covered in Section 7. Section 8 provides a decision flowchart for selecting the appropriate ML algorithm. The reading concludes with a summary.

2 MACHINE LEARNING AND INVESTMENT MANAGEMENT

a Distinguish between supervised machine learning, unsupervised machine learning, and deep learning

The growing volume and exploding diversity of data, as well as the perceived increasing economic value of insights extracted from these data, have inspired rapid growth in data science. This newly emerging field combines mathematics, computer science, and business analytics. It also strikes out in a new direction that relies on learning—from basic learning functions that map relationships between variables to advanced neural networks that mimic physical processes that absorb, order, and adapt to information.

Machine learning has theoretical and practical implications for investment management. For example, machine learning could potentially reshape accepted wisdom about asset risk premiums and reconfigure investment management business processes. Large datasets and learning models are already affecting investment management practices—from client profiling to asset allocation, stock selection, portfolio construction and risk management, and trading.

Machine learning applications are at each step of the asset and wealth management value chain. Chatbots answer basic retirement savings questions, learning from their interactions with investors. Machine learning methods can be used to generate alpha signals used in security selection by creating a non-linear forecast for a single time series, by deriving a forecast from a suite of predefined factors, or even by choosing input signals from existing or newly found data. For example, researchers using textual analysis have found that year-over-year changes in annual (10-K) and quarterly (10-Q) filings, particularly negative changes in the management discussion and risk sections, can strongly predict equity returns.

Machine learning methods can help calculate target portfolio weights that incorporate client restrictions and then dynamically weight them to maximize a Sharpe ratio. Another use of machine learning methods is better estimation of the variance–covariance matrix via principal components analysis, which reduces the number of variables needed to explain the variation in the data. Research suggests that machine learning solutions outperform mean–variance optimization in portfolio construction. Machine learning techniques are already creating better order flow management tools with non-linear trading algorithms that reduce the costs of implementing portfolio decisions. These developments have caused an evolution in the automation of tools, processes, and businesses (such as robo-advising).

WHAT IS MACHINE LEARNING?

3

We now discuss some fundamental concepts of machine learning, including a definition and an overview of key types of machine learning, such as supervised and unsupervised ML.

3.1 Defining Machine Learning

Statistical approaches and machine learning techniques both analyze observations to reveal some underlying process; however, they diverge in their assumptions, terminology, and techniques. Statistical approaches rely on foundational assumptions and explicit models of structure, such as observed samples that are assumed to be drawn from a specified underlying probability distribution. These a priori restrictive assumptions can fail in reality.

In contrast, machine learning seeks to extract knowledge from large amounts of data with fewer such restrictions. The goal of machine learning algorithms is to automate decision-making processes by generalizing (i.e., "learning") from known examples to determine an underlying structure in the data. The emphasis is on the ability of the algorithm to generate structure or predictions from data without any human help. An elementary way to think of ML algorithms is to "find the pattern, apply the pattern."

Machine learning techniques are better able than statistical approaches (such as linear regression) to handle problems with many variables (high dimensionality) or with a high degree of non-linearity. ML algorithms are particularly good at detecting change, even in highly non-linear systems, because they can detect the preconditions of a model's break or anticipate the probability of a regime switch.

Machine learning is broadly divided into three distinct classes of techniques: supervised learning, unsupervised learning, and deep learning/reinforcement learning.

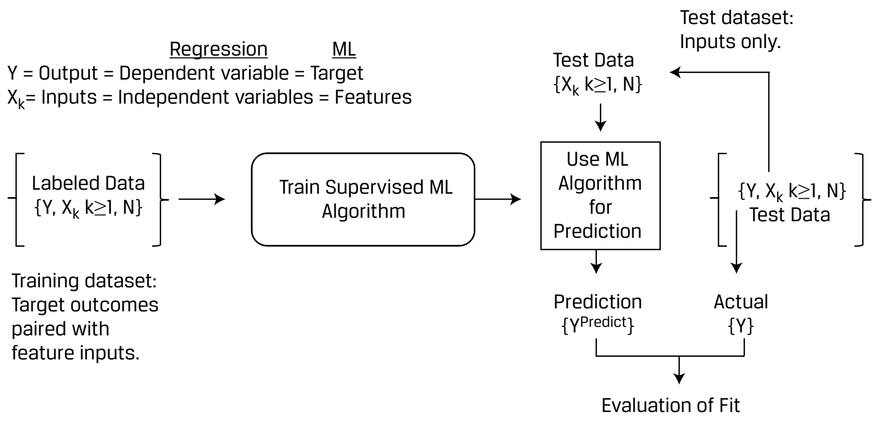
3.2 Supervised Learning

Supervised learning involves ML algorithms that infer patterns between a set of inputs (the *X*'s) and the desired output (*Y*). The inferred pattern is then used to map a given input set into a predicted output. Supervised learning requires a **labeled dataset**, one that contains matched sets of observed inputs and the associated output. Applying the ML algorithm to this dataset to infer the pattern between the inputs and output is called "training" the algorithm. Once the algorithm has been trained, the inferred pattern can be used to predict output values based on new inputs (i.e., ones not in the training dataset).

Multiple regression is an example of supervised learning. A regression model takes matched data (*X*'s, *Y*) and uses it to estimate parameters that characterize the relationship between *Y* and the *X*'s. The estimated parameters can then be used to predict *Y* on a new, different set of *X*'s. The difference between the predicted and actual *Y* is used to evaluate how well the regression model predicts out-of-sample (i.e., using new data).

The terminology used with ML algorithms differs from that used in regression. Exhibit 1 provides a visual of the supervised learning model training process and a translation between regression and ML terminologies.

Exhibit 1 Overview of Supervised Learning


Regression ML
Y = Output = Dependent variable = Target
X_k = Inputs = Independent variables = Features
Labeled Data {Y, X_k k≥1, N}
Training dataset: Target outcomes paired with feature inputs.
Train Supervised ML Algorithm
Test Data {X_k k≥1, N}
Test dataset: Inputs only.
Use ML Algorithm for Prediction
{Y, X_k k≥1, N} Test Data
Prediction {Y^Predict}
Actual {Y}
Evaluation of Fit

In supervised machine learning, the dependent variable (Y) is the **target** and the independent variables (X's) are known as **features**. The labeled data (training dataset) is used to train the supervised ML algorithm to infer a pattern-based prediction rule. The fit of the ML model is evaluated using labeled test data in which the predicted targets ($Y^{Predict}$) are compared to the actual targets (Y^{Actual}).

An example of supervised learning is the case in which ML algorithms are used to predict whether credit card transactions are fraudulent or legitimate. In the credit card example, the target is a binary variable with a value of 1 for "fraudulent" or 0 for "non-fraudulent." The features are the transaction characteristics. The chosen ML algorithm uses these data elements to train a model to predict the likelihood of fraud more accurately in new transactions. The ML program "learns from experience" if the percentage of correctly predicted credit card transactions increases as the amount of input from a growing credit card database increases. One possible ML algorithm to use would be to fit a logistic regression model to the data to provide an estimate of the probability a transaction is fraudulent.

Supervised learning can be divided into two categories of problems—regression and classification—with the distinction between them being determined by the nature of the target (Y) variable. If the target variable is continuous, then the task is one of regression (even if the ML technique used is not "regression"; note this nuance of ML terminology). If the target variable is categorical or ordinal (i.e., a ranked category), then it is a classification problem. Regression and classification use different ML techniques.

Regression focuses on making predictions of continuous target variables. Most readers are already familiar with multiple linear regression (e.g., ordinary least squares) models, but other supervised learning techniques exist, including non-linear models. These non-linear models are useful for problems involving large datasets with large numbers of features, many of which may be correlated. Some examples of problems belonging to the regression category are using historical stock market returns to forecast stock price performance or using historical corporate financial ratios to forecast the probability of bond default.

Classification focuses on sorting observations into distinct categories. In a regression problem, when the dependent variable (target) is categorical, the model relating the outcome to the independent variables (features) is called a "classifier." You should already be familiar with logistic regression as a type of classifier. Many classification models are binary classifiers, as in the case of fraud detection for credit card transactions. Multi-category classification is not uncommon, as in the case of classifying firms into multiple credit rating categories. In assigning ratings, the outcome variable is ordinal, meaning the categories have a distinct order or ranking (e.g., from low to high creditworthiness). Ordinal variables are intermediate between categorical variables and continuous variables on a scale of measurement.

3.3 Unsupervised Learning

Unsupervised learning is machine learning that does not make use of labeled data. More formally, in unsupervised learning, we have inputs (*X*'s) that are used for analysis without any target (*Y*) being supplied. In unsupervised learning, because the ML algorithm is not given labeled training data, the algorithm seeks to discover structure within the data themselves. As such, unsupervised learning is useful for exploring new datasets because it can provide human experts with insights into a dataset too big or too complex to visualize.

Two important types of problems that are well suited to unsupervised machine learning are reducing the dimension of data and sorting data into clusters, known as dimension reduction and clustering, respectively.

Dimension reduction focuses on reducing the number of features while retaining variation across observations to preserve the information contained in that variation. Dimension reduction may have several purposes. It may be applied to data with a large number of features to produce a lower dimensional representation (i.e., with fewer features) that can fit, for example, on a computer screen. Dimension reduction is also used in many quantitative investment and risk management applications where it is critical to identify the most predictive factors underlying asset price movements.

Clustering focuses on sorting observations into groups (clusters) such that observations in the same cluster are more similar to each other than they are to observations in other clusters. Groups are formed based on a set of criteria that may or may not be prespecified (such as the number of groups). Clustering has been used by asset managers to sort companies into groupings driven by data (e.g., based on their financial statement data or corporate characteristics) rather than conventional groupings (e.g., based on sectors or countries).

3.4 Deep Learning and Reinforcement Learning

More broadly in the field of artificial intelligence, additional categories of machine learning algorithms are distinguished. In **deep learning**, sophisticated algorithms address complex tasks, such as image classification, face recognition, speech recognition, and natural language processing. Deep learning is based on **neural networks** (NNs), also called artificial neural networks (ANNs)—highly flexible ML algorithms that have been successfully applied to a variety of supervised and unsupervised tasks characterized by large datasets, non-linearities, and interactions among features. In **reinforcement learning**, a computer learns from interacting with itself or data generated by the same algorithm. Deep learning and reinforcement learning principles have been combined to create efficient algorithms for solving a range of highly complex problems in robotics, health care, and finance.

3.5 Summary of ML Algorithms and How to Choose among Them

Exhibit 2 is a guide to the various machine learning algorithms organized by algorithm type (supervised or unsupervised) and by type of variables (continuous, categorical, or both). We will not cover linear or logistic regression since they are covered elsewhere in readings on quantitative methods. The extensions of linear regression, such as penalized regression and least absolute shrinkage and selection operator (LASSO), as well as the other ML algorithms shown in Exhibit 2, will be covered in this reading.

Exhibit 2 Guide to ML Algorithms

	ML Algorithm Type	
Variables	**Supervised (Target Variable)**	**Unsupervised (No Target Variable)**
Continuous	**Regression**	**Dimension Reduction**
	• Linear; Penalized Regression/LASSO	• Principal Components Analysis (PCA)
	• Logistic	**Clustering**
	• Classification and Regression Tree (CART)	• *K*-Means
	• Random Forest	• Hierarchical
Categorical	**Classification**	**Dimension Reduction**
	• Logistic	• Principal Components Analysis (PCA)
	• Support Vector Machine (SVM)	**Clustering**
	• *K*-Nearest Neighbor (KNN)	• *K*-Means
	• Classification and Regression Tree (CART)	• Hierarchical
Continuous or Categorical	Neural Networks	Neural Networks
	Deep Learning	Deep Learning
	Reinforcement Learning	Reinforcement Learning

EXAMPLE 1

Machine Learning Overview

1 Which of the following *best* describes machine learning? Machine learning:

A is a type of computer algorithm used just for linear regression.

B is a set of algorithmic approaches aimed at generating structure or predictions from data without human intervention by finding a pattern and then applying the pattern.

C is a set of computer-driven approaches adapted to extracting information from linear, labeled datasets.

2 Which of the following statements is *most* accurate? When attempting to discover groupings of data without any target (Y) variable:

A an unsupervised ML algorithm is used.

B an ML algorithm that is given labeled training data is used.

C a supervised ML algorithm is used.

3 Which of the following statements concerning supervised learning *best* distinguishes it from unsupervised learning? Supervised learning involves:

A training on labeled data to infer a pattern-based prediction rule.

B training on unlabeled data to infer a pattern-based prediction rule.

C learning from unlabeled data by discovering underlying structure in the data themselves.

4 Which of the following *best* describes dimension reduction? Dimension reduction:

- **A** focuses on classifying observations in a dataset into known groups using labeled training data.
- **B** focuses on clustering observations in a dataset into unknown groups using unlabeled data.
- **C** focuses on reducing the number of features in a dataset while retaining variation across observations to preserve the information in that variation.

Solution to 1:

B is correct. A is incorrect because machine learning algorithms are typically not used for linear regression. C is incorrect because machine learning is not limited to extracting information from linear, labeled datasets.

Solution to 2:

A is correct. B is incorrect because the term “labeled training data” means the target (Y) is provided. C is incorrect because a supervised ML algorithm is meant to predict a target (Y) variable.

Solution to 3:

A is correct. B is incorrect because supervised learning uses labeled training data. C is incorrect because it describes unsupervised learning.

Solution to 4:

C is correct. A is incorrect because it describes classification, not dimension reduction. B is incorrect because it describes clustering, not dimension reduction.

OVERVIEW OF EVALUATING ML ALGORITHM PERFORMANCE 4

b Describe overfitting and identify methods of addressing it

Machine learning algorithms promise several advantages relative to a structured statistical approach in exploring and analyzing the structure of very large datasets. ML algorithms have the ability to uncover complex interactions between feature variables and the target variable, and they can process massive amounts of data quickly. Moreover, many ML algorithms can easily capture non-linear relationships and may be able to recognize and predict structural changes between features and the target. These advantages mainly derive from the non-parametric and non-linear models that allow more flexibility when inferring relationships.

The flexibility of ML algorithms comes with a price, however. ML algorithms can produce overly complex models with results that are difficult to interpret, may be sensitive to noise or particulars of the data, and may fit the training data too well. An ML algorithm that fits the training data too well will typically not predict well using new data. This problem is known as **overfitting**, and it means that the fitted algorithm does not **generalize** well to new data. A model that generalizes well is a model that retains its explanatory power when predicting using out-of-sample (i.e., new) data. An overfit model has incorporated the noise or random fluctuations in the training data into its learned relationship. The problem is that these aspects often do not apply to new data the algorithm receives and so will negatively impact the model's ability to generalize, therefore reducing its overall predictive value. The evaluation of any ML algorithm thus focuses on its prediction error on new data rather than on its goodness of fit on the data with which the algorithm was fitted (i.e., trained).

Generalization is an objective in model building, so the problem of overfitting is a challenge to attaining that objective. These two concepts are the focus of the discussion below.

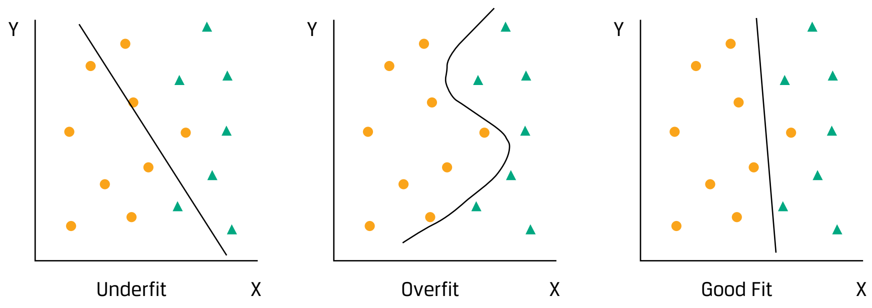
4.1 Generalization and Overfitting

To properly describe generalization and overfitting of an ML model, it is important to note the partitioning of the dataset to which the model will be applied. The dataset is typically divided into three non-overlapping samples: (1) **training sample** used to train the model, (2) **validation sample** for validating and tuning the model, and (3) **test sample** for testing the model's ability to predict well on new data. The training and validation samples are often referred to as being "in-sample," and the test sample is commonly referred to as being "out-of-sample." We will return shortly to the topic of partitioning the dataset.

To be valid and useful, any supervised machine learning model must generalize well beyond the training data. The model should retain its explanatory power when tested out-of-sample. As mentioned, one common reason for failure to generalize is overfitting. Think of overfitting as tailoring a custom suit that fits only one person. Continuing the analogy, underfitting is similar to making a baggy suit that fits no one, whereas robust fitting, the desired result, is similar to fashioning a universal suit that fits all people of similar dimensions.

The concepts of underfitting, overfitting, and good (or robust) fitting are illustrated in Exhibit 3. Underfitting means the model does not capture the relationships in the data. The left graph shows four errors in this underfit model (three misclassified circles and one misclassified triangle). Overfitting means training a model to such a degree of specificity to the training data that the model begins to incorporate noise coming from quirks or spurious correlations; it mistakes randomness for patterns and relationships. The algorithm may have memorized the data, rather than learned from it, so it has perfect hindsight but no foresight. The main contributors to overfitting are thus high noise levels in the data and too much complexity in the model. The middle graph shows no errors in this overfit model. **Complexity** refers to the number of features, terms, or branches in the model and to whether the model is linear or non-linear (non-linear is more complex). As models become more complex, overfitting risk increases. A good fit/robust model fits the training (in-sample) data well and generalizes well to out-of-sample data, both within acceptable degrees of error. The right graph shows that the good fitting model has only one error, the misclassified circle.

Exhibit 3 Underfitting, Overfitting, and Good Fitting



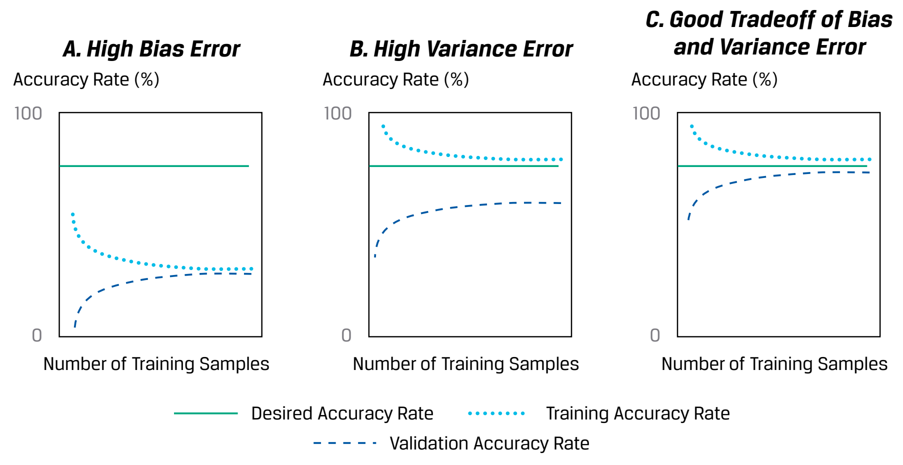
4.2 Errors and Overfitting

To capture these effects and calibrate degree of fit, data scientists compare error rates in- and out-of-sample as a function of both the data and the algorithm. Total in-sample errors (E_{in}) are generated by the predictions of the fitted relationship relative to actual target outcomes on the training sample. Total out-of-sample errors (E_{out}) are from either the validation or test samples. Low or no in-sample error but large out-of-sample error are indicative of poor generalization. Data scientists decompose the total out-of-sample error into three sources:

1. **Bias error**, or the degree to which a model fits the training data. Algorithms with erroneous assumptions produce high bias with poor approximation, causing underfitting and high in-sample error.
2. **Variance error**, or how much the model's results change in response to new data from validation and test samples. Unstable models pick up noise and produce high variance, causing overfitting and high out-of-sample error.
3. **Base error** due to randomness in the data.

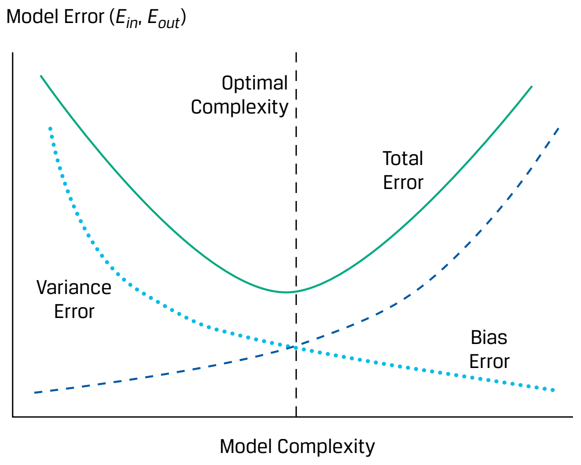
A **learning curve** plots the accuracy rate (= 1 – error rate) in the validation or test samples (i.e., out-of-sample) against the amount of data in the training sample, so it is useful for describing under- and overfitting as a function of bias and variance errors. If the model is robust, out-of-sample accuracy increases as the training sample size increases. This implies that error rates experienced in the validation or test samples (E_{out}) and in the training sample (E_{in}) converge toward each other and toward a desired error rate (or, alternatively, the base error). In an underfitted model with high bias error, shown in the left panel of Exhibit 4, high error rates cause convergence below the desired accuracy rate. Adding more training samples will not improve the model to the desired performance level. In an overfitted model with high variance error, shown in the middle panel of Exhibit 4, the validation sample and training sample error rates fail to converge. In building models, data scientists try to simultaneously minimize both bias and variance errors while selecting an algorithm with good predictive or classifying power, as seen in the right panel of Exhibit 4.

Exhibit 4 Learning Curves: Accuracy in Validation and Training Samples



Out-of-sample error rates are also a function of model complexity. As complexity increases in the training set, error rates (E_{in}) fall and bias error shrinks. As complexity increases in the test set, however, error rates (E_{out}) rise and variance error rises. Typically, linear functions are more susceptible to bias error and underfitting, while non-linear functions are more prone to variance error and overfitting. Therefore, an optimal point of model complexity exists where the bias and variance error curves intersect and in- and out-of-sample error rates are minimized. A **fitting curve**, which shows in- and out-of-sample error rates (E_{in} and E_{out}) on the y-axis plotted against model complexity on the x-axis, is presented in Exhibit 5 and illustrates this trade-off.

Exhibit 5 Fitting Curve Shows Trade-Off between Bias and Variance Errors and Model Complexity



Finding the optimal point (managing overfitting risk)—the point just before the total error rate starts to rise (due to increasing variance error)—is a core part of the machine learning process and the key to successful generalization. Data scientists express the trade-off between overfitting and generalization as a trade-off between *cost* (the difference between in- and out-of-sample error rates) and *complexity*. They use the trade-off between cost and complexity to calibrate and visualize under- and overfitting and to optimize their models.

4.3 Preventing Overfitting in Supervised Machine Learning

We have seen that overfitting impairs generalization, but overfitting potential is endemic to the supervised machine learning process due to the presence of noise. So, how do data scientists combat this risk? Two common methods are used to reduce overfitting: (1) preventing the algorithm from getting too complex during selection and training, which requires estimating an overfitting penalty, and (2) proper data sampling achieved by using **cross-validation**, a technique for estimating out-of-sample error directly by determining the error in validation samples.

The first strategy comes from Occam's razor, the problem-solving principle that the simplest solution tends to be the correct one. In supervised machine learning, it means limiting the number of features and penalizing algorithms that are too complex or too flexible by constraining them to include only parameters that reduce out-of-sample error.

The second strategy comes from the principle of avoiding sampling bias. But sampling bias can creep into machine learning in many ways. The challenge is having a large enough dataset to make both training and testing possible on representative samples. An unrepresentative sample or reducing the training sample size too much could obscure its true patterns, thereby increasing bias. In supervised machine learning, the technique for reducing sampling bias is through careful partitioning of the dataset into three groups: (1) training sample, the set of labeled training data where the target variable (Y) is known;(2) validation sample, the set of data used for making structural choices on the degree of model complexity, comparing various solutions, and tuning the selected model, thereby validating the model; and (3) test sample, the set of data held aside for testing to confirm the model's predictive or classifying power. The goal, of course, is to deploy the tested model on fresh data from the same domain.

To mitigate the problem of such **holdout samples** (i.e., data samples not used to train the model) reducing the training set size too much, modelers use special cross-validation techniques. One such technique is ***k*-fold cross-validation**, in which the data (excluding test sample and fresh data) are shuffled randomly and then are divided into k equal sub-samples, with $k - 1$ samples used as training samples and one sample, the kth, used as a validation sample. Note that k is typically set at 5 or 10. This process is then repeated k times, which helps minimize both bias and variance by insuring that each data point is used in the training set $k - 1$ times and in the validation set once. The average of the k validation errors (mean E_{val}) is then taken as a reasonable estimate of the model's out-of-sample error (E_{out}). A limitation of k-fold cross-validation is that it cannot be used with time-series data, where only the most recent data can reasonably be used for model validation.

In sum, mitigating overfitting risk by avoiding excessive out-of-sample error is critical to creating a supervised machine learning model that generalizes well to fresh datasets drawn from the same distribution. The main techniques used to mitigate overfitting risk in model construction are complexity reduction (or regularization) and cross-validation.

EXAMPLE 2

Evaluating ML Algorithm Performance

Shreya Anand is a portfolio manager based in the Mumbai headquarters office of an investment firm, where she runs a high-dividend-yield fund for wealthy clients. Anand has some knowledge of data science from her university studies. She is interested in classifying companies in the NIFTY 200 Index—an index of large- and mid-cap companies listed on the National Stock Exchange of India—into two categories: dividend increase and no dividend increase. She assembles data for training, validating, and testing an ML-based model that consists of 1,000 observations of NIFTY 200 companies, each consisting of 25 features (fundamental and technical) and the labeled target (dividend increase or no dividend increase).

After training her model, Anand discovers that while it is good at correctly classifying using the training sample, it does not perform well on new data. In consulting her colleagues about this issue, Anand hears conflicting explanations about what constitutes good generalization in an ML model:

Statement 1	The model retains its explanatory power when predicting using new data (i.e., out-of-sample).
Statement 2	The model shows low explanatory power after training using in-sample data (i.e., training data).
Statement 3	The model loses its explanatory power when predicting using new data (i.e., out-of-sample).

1 Which statement made to Anand is *most* accurate?

A Statement 1

B Statement 2

C Statement 3

2 Anand's model is *most likely* being impaired by which of the following?

A Underfitting and bias error

B Overfitting and variance error

C Overfitting and bias error

3 By implementing which one of the following actions can Anand address the problem?

A Estimate and incorporate into the model a penalty that decreases in size with the number of included features.

B Use the *k*-fold cross-validation technique to estimate the model's out-of-sample error, and then adjust the model accordingly.

C Use an unsupervised learning model.

Solution to 1:

A, Statement 1, is correct. B, Statement 2, is incorrect because it describes a poorly fitting model with high bias. C, Statement 3, is incorrect because it describes an overfitted model with poor generalization.

Solution to 2:

B is correct. Anand's model is good at correctly classifying using the training sample, but it does not perform well using new data. The model is overfitted, so it has high variance error.

Solution to 3:

B is correct. A is incorrect because the penalty should increase in size with the number of included features. C is incorrect because Anand is using labeled data for classification, and unsupervised learning models do not use labeled data.

5 SUPERVISED MACHINE LEARNING ALGORITHMS

c Describe supervised machine learning algorithms—including penalized regression, support vector machine, K-nearest neighbor, classification and regression tree, ensemble learning, and random forest—and determine the problems for which they are best suited

Supervised machine learning models are trained using labeled data, and depending on the nature of the target (Y) variable, they can be divided into two types: regression for a continuous target variable and classification for a categorical or ordinal target variable. As shown in Exhibit 2 under regression, we will now cover penalized regression and LASSO. Then, as shown under classification, we will introduce support vector machine (SVM), *k*-nearest neighbor (KNN), and classification and regression tree (CART) algorithms. Note that CART, as its name implies, can be used for both classification and regression problems.

In the following discussion, assume we have a number of observations of a target variable, Y, and n real valued features, $X_1, \ldots, X_n$, that we may use to establish a relationship (regression or classification) between $\mathbf{X}$ (a vector of the X_i) and Y for each observation in our dataset.

5.1 Penalized Regression

Penalized regression is a computationally efficient technique used in prediction problems. In practice, penalized regression has been useful for reducing a large number of features to a manageable set and for making good predictions in a variety of large datasets, especially where features are correlated (i.e., when classical linear regression breaks down).

In a large dataset context, we may have many features that potentially could be used to explain *Y*. When a model is fit to training data, the model may so closely reflect the characteristics of the specific training data that the model does not perform well on new data. Features may be included that reflect noise or randomness in the training dataset that will not be present in new or future data used for making predictions. That is the problem of overfitting, and penalized regression can be described as a technique to avoid overfitting. In prediction, out-of-sample performance is key, so relatively parsimonious models (that is, models in which each variable plays an essential role) tend to work well because they are less subject to overfitting.

Let us suppose that we standardize our data so the features have a mean of 0 and a variance of 1. Standardization of features will allow us to compare the magnitudes of regression coefficients for the feature variables. In ordinary linear regression (i.e., ordinary least squares, or OLS), the regression coefficients $\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_K$ are chosen to *minimize* the sum of the squared residuals (i.e., the sum of the squared difference between the actual values, Y_i, and the predicted values, $\hat{Y}_i$), or

$$\textstyle\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2.$$

Penalized regression includes a constraint such that the regression coefficients are chosen to minimize the sum of squared residuals *plus* a penalty term that increases in size with the number of included features. So, in a penalized regression, a feature must make a sufficient contribution to model fit to offset the penalty from including it. Therefore, only the more important features for explaining *Y* will remain in the penalized regression model.

In one popular type of penalized regression, **LASSO**, or least absolute shrinkage and selection operator, the penalty term has the following form, with $\lambda > 0$:

$$\text{Penalty term} = \lambda \sum_{k=1}^{K} \left|\hat{b}_k\right|.$$

In addition to minimizing the sum of the squared residuals, LASSO involves minimizing the sum of the absolute values of the regression coefficients (see the following expression). The greater the number of included features (i.e., variables with non-zero coefficients), the larger the penalty term. Therefore, penalized regression ensures that a feature is included only if the sum of squared residuals declines by more than the penalty term increases. All types of penalized regression involve a trade-off of this type. Also, since LASSO eliminates the least important features from the model, it automatically performs a type of feature selection.

$$\textstyle\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 + \lambda\sum_{k=1}^{K}\left|\hat{b}_K\right|.$$

Lambda (λ) is a **hyperparameter**—a parameter whose value must be set by the researcher before learning begins—of the regression model and will determine the balance between fitting the model versus keeping the model parsimonious. In practice, a hyperparameter is set by reviewing model performance repeatedly at different settings on the validation set, and hence the test set is also essential to avoid overfitting of hyperparameters to the validation data.

Note that in the case where λ = 0, the LASSO penalized regression is equivalent to an OLS regression. When using LASSO or other penalized regression techniques, the penalty term is added only during the model building process (i.e., when fitting the model to the training data). Once the model has been built, the penalty term is no longer needed, and the model is then evaluated by the sum of the squared residuals generated using the test dataset.

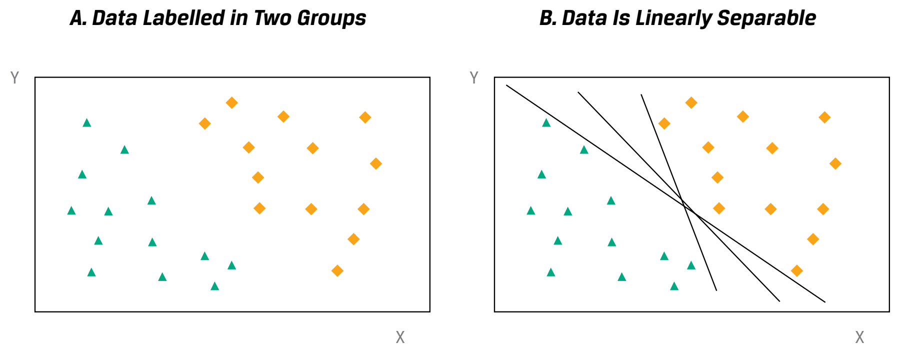
With today's availability of fast computation algorithms, investment analysts are increasingly using LASSO and other regularization techniques to remove less pertinent features and build parsimonious models. **Regularization** describes methods that reduce statistical variability in high-dimensional data estimation problems—in this case, reducing regression coefficient estimates toward zero and thereby avoiding complex models and the risk of overfitting. LASSO has been used, for example, for forecasting default probabilities in industrial sectors where scores of potential features, many collinear, have been reduced to fewer than 10 variables, which is important given the relatively small number (about 100) of observations of default.

Regularization methods can also be applied to non-linear models. A long-term challenge of the asset management industry in applying mean–variance optimization has been the estimation of stable covariance matrixes and asset weights for large portfolios. Asset returns typically exhibit strong multi-collinearity, making the estimation of the covariance matrix highly sensitive to noise and outliers, so the resulting optimized asset weights are highly unstable. Regularization methods have been used to address this problem. The relatively parsimonious models produced by applying penalized regression methods, such as LASSO, tend to work well because they are less subject to overfitting.

5.2 Support Vector Machine

Support vector machine (SVM) is one of the most popular algorithms in machine learning. It is a powerful supervised algorithm used for classification, regression, and outlier detection. Despite its complicated-sounding name, the notion is relatively straightforward and best explained with a few pictures. The left panel in Exhibit 6 presents a simple dataset with two features (x and y coordinates) labeled in two groups (triangles and crosses). These binary labeled data are noticeably separated into two distinct regions, which could represent stocks with positive and negative returns in a given year. These two regions can be easily separated by an infinite number of straight lines; three of them are shown in the right panel of Exhibit 6. The data are thus linearly separable, and any of the straight lines shown would be called a **linear classifier**—a binary classifier that makes its classification decision based on a linear combination of the features of each data point.

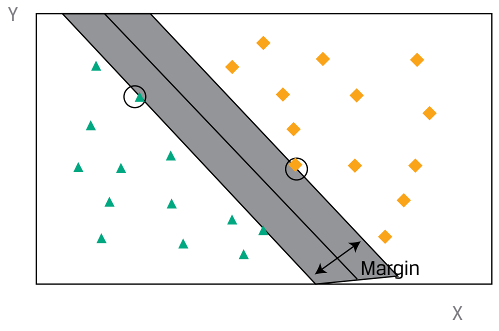
Exhibit 6 Scatterplots and Linear Separation of Labeled Data



With two dimensions or features (x and y), linear classifiers can be represented as straight lines. Observations with n features can be represented in an n-dimension space, and the dataset would be linearly separable if the observations can be separated into two distinct regions by a linear space boundary. The general term for such a space boundary is an n-dimensional hyperplane, which with n = 1 is called a line and with n = 2 is called a plane.

Support vector machine is a linear classifier that determines the hyperplane that optimally separates the observations into two sets of data points. The intuitive idea behind the SVM algorithm is maximizing the probability of making a correct prediction (here, that an observation is a triangle or a cross) by determining the boundary that is the furthest away from all the observations. In Exhibit 7, SVM separates the data by the maximum margin, where the margin is the shaded strip that divides the observations into two groups. The straight line in the middle of the shaded strip is the discriminant boundary, or boundary, for short. We can see that the SVM algorithm produces the widest shaded strip (i.e., the one with the maximum margin on either side of the boundary). The margin is determined by the observations closest to the boundary (the circled points) in each set, and these observations are called support vectors. Adding more training data away from the support vectors will not affect the boundary. In our training datasets, however, adding data points which are close to the hyperplane may move the margin by changing the set of support vectors.

Exhibit 7 Linear Support Vector Machine Classifier



In Exhibit 7, SVM is classifying all observations perfectly. Most real-world datasets, however, are not linearly separable. Some observations may fall on the wrong side of the boundary and be misclassified by the SVM algorithm. The SVM algorithm handles this problem by an adaptation called **soft margin classification**, which adds a penalty to the objective function for observations in the training set that are misclassified. In essence, the SVM algorithm will choose a discriminant boundary that optimizes the trade-off between a wider margin and a lower total error penalty.

As an alternative to soft margin classification, a non-linear SVM algorithm can be run by introducing more advanced, non-linear separation boundaries. These algorithms may reduce the number of misclassified instances in the training datasets but are more complex and, so, are prone to overfitting.

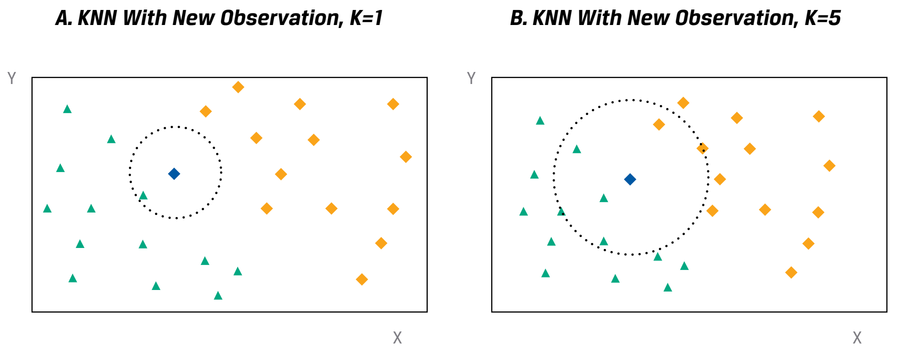
SVM has many applications in investment management. It is particularly suited for small to medium-size but complex high-dimensional datasets, such as corporate financial statements or bankruptcy databases. Investors seek to predict company failures for identifying stocks to avoid or to short sell, and SVM can generate a binary classification (e.g., bankruptcy likely versus bankruptcy unlikely) using many fundamental and technical feature variables. SVM can effectively capture the characteristics of such data with many features while being resilient to outliers and correlated features.

SVM can also be used to classify text from documents (e.g., news articles, company announcements, and company annual reports) into useful categories for investors (e.g., positive sentiment and negative sentiment).

5.3 *K*-Nearest Neighbor

***K*-nearest neighbor** (KNN) is a supervised learning technique used most often for classification and sometimes for regression. The idea is to classify a new observation by finding similarities ("nearness") between this new observation and the existing data. Going back to the scatterplot in Exhibit 6, let us assume we have a new observation: The diamond in Exhibit 8 needs to be classified as belonging to either the cross or the triangle category. If $k = 1$, the diamond will be classified into the same category as its nearest neighbor (i.e., the triangle in the left panel). The right panel in Exhibit 8 presents the case where $k = 5$, so the algorithm will look at the diamond's five nearest neighbors, which are three triangles and two crosses. The decision rule is to choose the classification with the largest number of nearest neighbors out of the five being considered. So, the diamond is again classified as belonging to the triangle category.

Exhibit 8 *K*-Nearest Neighbor Algorithm



Let us suppose we have a database of corporate bonds classified by credit rating that also contains detailed information on the characteristics of these bonds. Such features would include those of the issuing company (e.g., asset size, industry, leverage ratios, cash flow ratios) and of the bond issue itself (e.g., tenor, fixed/floating coupon, embedded options). Now, assume a new bond is about to be issued with no credit rating. By nature, corporate bonds with similar issuer and issue characteristics should be given a similar credit rating. So, by using KNN, we can predict the implied credit rating of the new bond based on the similarities of its characteristics to those of the bonds in our database.

KNN is a straightforward, intuitive model that is still very powerful because it is non-parametric; the model makes no assumptions about the distribution of the data. Moreover, it can be used directly for multi-class classification. A critical challenge of KNN, however, is defining what it means to be "similar" (or near). Besides the selection of features, an important decision relates to the distance metric used to model similarity because an inappropriate measure will generate poorly performing models. The choice of a correct distance measure may be even more subjective for ordinal or categorical data. For example, if an analyst is looking at the similarities in market performance of various equities, he or she may consider using the correlation between the stocks' historical returns as an appropriate measure of similarity.

Knowledge of the data and understanding of the business objectives of the analysis are critical aspects in the process of defining similarity. KNN results can be sensitive to inclusion of irrelevant or correlated features, so it may be necessary to select features manually. By doing so, the analyst removes less valuable information to keep the most relevant and pertinent information. If done correctly, this process should generate a more representative distance measure. KNN algorithms tend to work better with a small number of features.

Finally, the number *k*, the hyperparameter of the model, must be chosen with the understanding that different values of *k* can lead to different conclusions. For predicting the credit rating of an unrated bond, for example, should *k* be the 3, 15, or 50 bonds most similar to the unrated bond? If *k* is an even number, there may be ties and no clear classification. Choosing a value for *k* that is too small would result in a high error rate and sensitivity to local outliers, but choosing a value for *k* that is too large would dilute the concept of nearest neighbors by averaging too many outcomes. In practice, several different techniques can be used to determine an optimal value for *k*, taking into account the number of categories and their partitioning of the feature space.

The KNN algorithm has many applications in the investment industry, including bankruptcy prediction, stock price prediction, corporate bond credit rating assignment, and customized equity and bond index creation. For example, KNN is useful for determining bonds that are similar and those that are dissimilar, which is critical information for creating a custom, diversified bond index.

5.4 Classification and Regression Tree

Classification and regression tree (CART) is another common supervised machine learning technique that can be applied to predict either a categorical target variable, producing a classification tree, or a continuous target variable, producing a regression tree. CART is commonly applied to binary classification or regression.

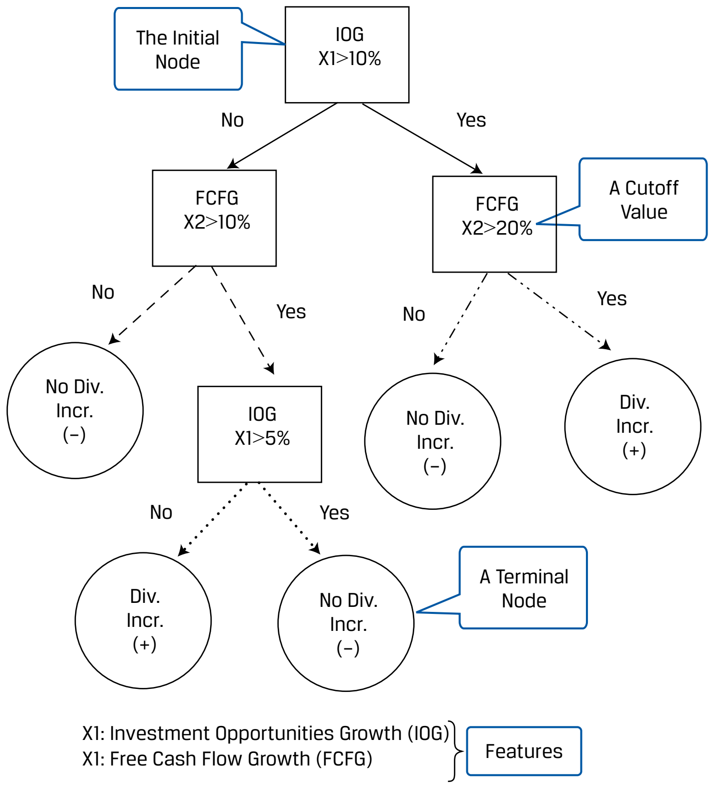
CART will be discussed in the context of a simplified model for classifying companies by whether they are likely to increase their dividends to shareholders. Such a classification requires a binary tree: a combination of an initial root node, decision nodes, and terminal nodes. The root node and each decision node represent a single feature (*f*) and a cutoff value (*c*) for that feature. As shown in Panel A of Exhibit 9, we start at the initial root node for a new data point. In this case, the initial root node represents the feature investment opportunities growth (IOG), designated as X1, with a cutoff value of 10%. From the initial root node, the data are partitioned at decision nodes into smaller and smaller subgroups until terminal nodes that contain the predicted labels are formed. In this case, the predicted labels are either dividend increase (the cross) or no dividend increase (the dash).

Also shown in Panel A of Exhibit 9, if the value of feature IOG (X1) is greater than 10% (Yes), then we proceed to the decision node for free cash flow growth (FCFG), designated as X2, which has a cutoff value of 20%. Now, if the value of FCFG is not greater than 20% (No), then CART will predict that that data point belongs to the no dividend increase (dash) category, which represents a terminal node. Conversely, if the value of X2 is greater than 20% (Yes), then CART will predict that that data point belongs to the dividend increase (cross) category, which represents another terminal node.

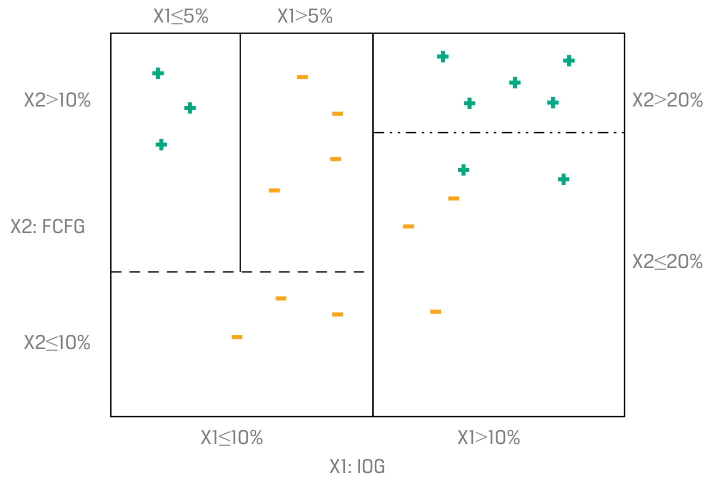
It is important to note that the same feature can appear several times in a tree in combination with other features. Moreover, some features may be relevant only if other conditions have been met. For example, going back to the initial root node, if IOG is not greater than 10% ($X1 \leq 10\%$) and FCFG is greater than 10%, then IOG appears again as another decision node, but this time it is lower down in the tree and has a cutoff value of 5%.

Exhibit 9 Classification and Regression Tree—Decision Tree and Partitioning of the Feature Space

A. Decision Tree



B. Partitioning of the Feature (X1, X2) Space



We now turn to how the CART algorithm selects features and cutoff values for them. Initially, the classification model is trained from the labeled data, which in this hypothetical case are 10 instances of companies having a dividend increase (the crosses) and 10 instances of companies with no dividend increase (the dashes). As shown in Panel B of Exhibit 9, at the initial root node and at each decision node, the feature space (i.e., the plane defined by X1 and X2) is split into two rectangles for values above and below the cutoff value for the particular feature represented at that node. This can be seen by noting the distinct patterns of the lines that emanate from the decision nodes in Panel A. These same distinct patterns are used for partitioning the feature space in Panel B.

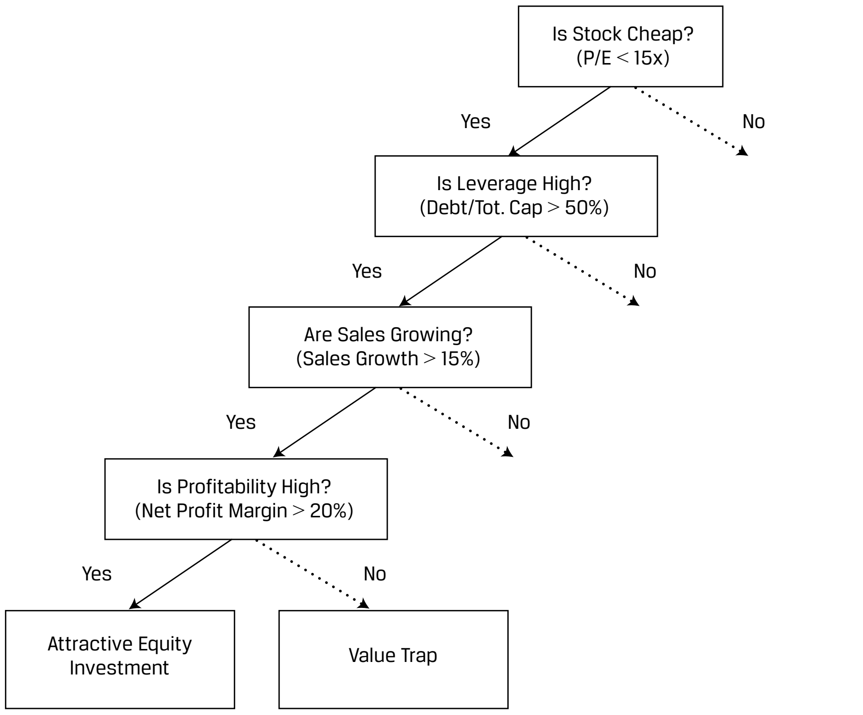
The CART algorithm chooses the feature and the cutoff value at each node that generates the widest separation of the labeled data to minimize classification error (e.g., by a criterion, such as mean-squared error). After each decision node, the partition of the feature space becomes smaller and smaller, so observations in each group have lower within-group error than before. At any level of the tree, when the classification error does not diminish much more from another split (bifurcation), the process stops, the node is a terminal node, and the category that is in the majority at that node is assigned to it. If the objective of the model is classification, then the prediction of the algorithm at each terminal node will be the category with the majority of data points. For example, in Panel B of Exhibit 9, the top right rectangle of the feature space, representing IOG (X1) > 10% and FCFG (X2)> 20%, contains five crosses, the most data points of any of the partitions. So, CART would predict that a new data point (i.e., a company) with such features belongs to the dividend increase (cross) category. However, if instead the new data point had IOG (X1) > 10% and FCFG (X2) ≤ 20%, then it would be predicted to belong to the no dividend increase (dash) category—represented by the lower right rectangle, with two crosses but with three dashes. Finally, if the goal is regression, then the prediction at each terminal node is the mean of the labeled values.

CART makes no assumptions about the characteristics of the training data, so if left unconstrained, it potentially can perfectly learn the training data. To avoid such overfitting, regularization parameters can be added, such as the maximum depth of the tree, the minimum population at a node, or the maximum number of decision nodes. The iterative process of building the tree is stopped once the regularization criterion has been reached. For example, in Panel B of Exhibit 9, the upper left rectangle of the feature space (determined by X1 ≤ 10%, X2 > 10%, and X1 ≤ 5% with three crosses) might represent a terminal node resulting from a regularization criterion with minimum population equal to 3. Alternatively, regularization can occur via a **pruning** technique that can be used afterward to reduce the size of the tree. Sections of the tree that provide little classifying power are pruned (i.e., cut back or removed).

By its iterative structure, CART can uncover complex dependencies between features that other models cannot reveal. As demonstrated in Exhibit 9, the same feature can appear several times in combination with other features and some features may be relevant only if other conditions have been met.

As shown in Exhibit 10, high profitability is a critical feature for predicting whether a stock is an attractive investment or a value trap (i.e., an investment that, although apparently priced cheaply, is likely to be unprofitable). This feature is relevant only if the stock is cheap: For example, in this hypothetical case, if P/E is less than 15, leverage is high (debt to total capital > 50%) and sales are expanding (sales growth > 15%). Said another way, high profitability is irrelevant in this context if the stock is not cheap *and* if leverage is not high *and* if sales are not expanding. Multiple linear regression typically fails in such situations where the relationship between the features and the outcome is non-linear.

Exhibit 10 Stylized Decision Tree—Attractive Investment or Value Trap?


Is Stock Cheap? (P/E < 15x)
Yes
No
Is Leverage High? (Debt/Tot. Cap > 50%)
Yes
No
Are Sales Growing? (Sales Growth > 15%)
Yes
No
Is Profitability High? (Net Profit Margin > 20%)
Yes
No
Attractive Equity Investment
Value Trap

CART models are popular supervised machine learning models because the tree provides a visual explanation for the prediction. This contrasts favorably with other algorithms that are often considered to be "black boxes" because it may be difficult to understand the reasoning behind their outcomes and thus to place trust in them. CART is a powerful tool to build expert systems for decision-making processes. It can induce robust rules despite noisy data and complex relationships between high numbers of features. Typical applications of CART in investment management include, among others, enhancing detection of fraud in financial statements, generating consistent decision processes in equity and fixed-income selection, and simplifying communication of investment strategies to clients.

5.5 Ensemble Learning and Random Forest

Instead of basing predictions on the results of a single model as in the previous discussion, why not use the predictions of a group—or an ensemble—of models? Each single model will have a certain error rate and will make noisy predictions. But by taking the average result of many predictions from many models, we can expect to achieve a reduction in noise as the average result converges toward a more accurate prediction. This technique of combining the predictions from a collection of models is called **ensemble learning**, and the combination of multiple learning algorithms is known as the **ensemble method**. Ensemble learning typically produces more accurate and more stable predictions than the best single model. In fact, in many prestigious machine learning competitions, an ensemble method is often the winning solution.

Ensemble learning can be divided into two main categories: (1) aggregation of heterogeneous learners (i.e., different types of algorithms combined with a voting classifier) or (2) aggregation of homogeneous learners (i.e., a combination of the same algorithm using different training data that are based, for example, on a bootstrap aggregating, or bagging, technique, as discussed later).

5.5.1 *Voting Classifiers*

Suppose you have been working on a machine learning project for some time and have trained and compared the results of several algorithms, such as SVM, KNN, and CART. A **majority-vote classifier** will assign to a new data point the predicted label with the most votes. For example, if the SVM and KNN models are both predicting the category "stock outperformance" and the CART model is predicting the category "stock underperformance," then the majority-vote classifier will choose stock outperformance." The more individual models you have trained, the higher the accuracy of the aggregated prediction up to a point. There is an optimal number of models beyond which performance would be expected to deteriorate from overfitting. The trick is to look for diversity in the choice of algorithms, modeling techniques, and hypotheses. The (extreme) assumption here is that if the predictions of the individual models are independent, then we can use the law of large numbers to achieve a more accurate prediction.

5.5.2 *Bootstrap Aggregating (Bagging)*

Alternatively, one can use the same machine learning algorithm but with different training data. **Bootstrap aggregating (or bagging)** is a technique whereby the original training dataset is used to generate n new training datasets or bags of data. Each new bag of data is generated by random sampling with replacement from the initial training set. The algorithm can now be trained on n independent datasets that will generate n new models. Then, for each new observation, we can aggregate the n predictions using a majority-vote classifier for a classification or an average for a regression. Bagging is a very useful technique because it helps to improve the stability of predictions and protects against overfitting the model.

5.5.3 *Random Forest*

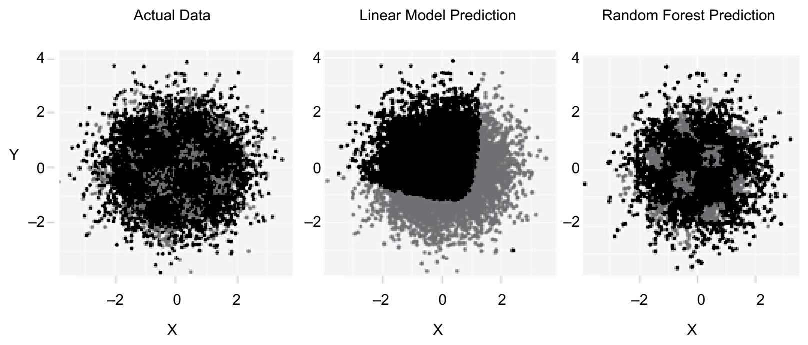
A **random forest classifier** is a collection of a large number of decision trees trained via a bagging method. For example, a CART algorithm would be trained using each of the n independent datasets (from the bagging process) to generate the multitude of different decision trees that make up the random forest classifier.

To derive even more individual predictions, added diversity can be generated in the trees by randomly reducing the number of features available during training. So, if each observation has n features, one can randomly select a subset of m features (where $m < n$) that will then be considered by the CART algorithm for splitting the dataset at each of the decision nodes. The number of subset features (m), the number of trees to use, the minimum size (population) of each node (or leaf), and the maximum depth of each tree are all hyperparameters that can be tuned to improve overall model prediction accuracy. For any new observation, we let all the classifier trees (the "random forest") undertake classification by majority vote—implementing a machine learning version of the "wisdom of crowds." The process involved in random forest construction tends to reduce variance and protect against overfitting on the training data. It also reduces the ratio of noise to signal because errors cancel out across the collection of slightly different classification trees. However, an important drawback of random forest is that it lacks the ease of interpretability of individual trees; as a result, it is considered a relatively black box type of algorithm.

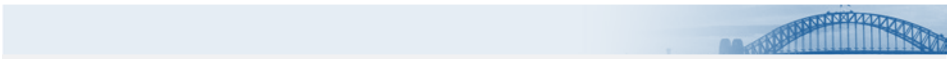
Exhibit 11 presents three scatterplots of actual and predicted defaults by small and medium-sized businesses with respect to two features, X and Y—for example, firm profitability and leverage, respectively. The left plot shows the actual cases of default in light shade and no default in dark shade, while the middle and right plots present the predicted defaults and no defaults (also in light and dark shades, respectively). It is clear from the middle plot, which is based on a traditional linear regression model, that the model fails to predict the complex non-linear relationship between the features.

Conversely, the right plot, which presents the prediction results of a random forest model, shows that this model performs very well in matching the actual distribution of the data.

Exhibit 11 Credit Defaults of Small- and Medium-Sized Borrowers



Source: Bacham and Zhao (2017).



Ensemble Learning with Random Forest

In making use of voting across classifier trees, random forest is an example of ensemble learning: Incorporating the output of a collection of models produces classifications that have better signal-to-noise ratios than the individual classifiers. A good example is a credit card fraud detection problem that comes from an open source dataset on Kaggle.[1] Here, the data contained several anonymized features that might be used to explain which transactions were fraudulent. The difficulty in the analysis arises from the fact that the rate of fraudulent transactions is very low; in a sample of 284,807 transactions, only 492 were fraudulent (0.17%). This is akin to finding a needle in a haystack. Applying a random forest classification algorithm with an oversampling technique—which involves increasing the proportional representation of fraudulent data in the training set—does extremely well. Despite the lopsided sample, it delivers **precision** (the ratio of correctly predicted fraudulent cases to all predicted fraudulent cases) of 89% and **recall** (the ratio of correctly predicted fraudulent cases to all actual fraudulent cases) of 82%.

Despite its relative simplicity, random forest is a powerful algorithm with many investment applications. These include, for example, use in factor-based investment strategies for asset allocation and investment selection or use in predicting whether an IPO will be successful (e.g., percent oversubscribed, first trading day close/IPO price) given the attributes of the IPO offering and the corporate issuer. Later, in a mini-case study, Deep Neural Network–Based Equity Factor Model, we present further details of how supervised machine learning is used for fundamental factor modeling.

1 See www.kaggle.com/mlg-ulb/creditcardfraud (accessed 1 October 2018).

EXAMPLE 3

Support Vector Machine and *K*-Nearest Neighbor

Rachel Lee is a fixed-income portfolio manager with Zeta Investment Management Company. Zeta manages an investment-grade bond portfolio for small, conservative institutions and a non-investment-grade (i.e., high-yield) bond portfolio for yield-seeking, high-net-worth individuals. Both portfolios can hold unrated bonds if the characteristics of the unrated bonds closely match those of the respective portfolio's average holding.

Lee is discussing an upcoming straight, 10-year fixed-coupon bond issue with senior credit analyst Marc Watson. Watson comments that although the bond's issuer, Biotron Corporation, has not had this issue rated, his analysis of the company's profitability, cash flow, leverage, and coverage ratios places the issue near the borderline between low investment-grade (Baa3/BBB–) and high non-investment-grade (Ba1/BB+) bonds.

Lee decides to use machine learning methods to confirm the implied credit rating of Biotron Corporation.

1 State the type of problem being addressed by Lee.

2 State two ML algorithms that Lee could use to explore the implied credit rating of Biotron Corporation, and then describe how each algorithm could be applied.

Lee decides to apply the two identified ML algorithms. Both algorithms clearly support a high non-investment-grade rating. Watson states that because both ML algorithms agree on the rating, he has confidence in relying on the rating.

3 State one argument in support of Watson's viewpoint.

Solution to 1:

Lee is addressing a supervised learning classification problem because she must determine whether Biotron's upcoming bond issue would be classified as investment grade or non-investment grade.

Solution to 2:

One suitable ML algorithm is SVM. The SVM algorithm is a linear classifier that aims to find the optimal hyperplane—the one that separates observations into two distinct sets by the maximum margin. So, SVM is well suited to binary classification problems, such as the one facing Lee (investment grade versus non-investment grade). In this case, Lee could train the SVM algorithm on data—characteristics (features) and rating (target)—of low investment-grade (Baa3/BBB–) and high non-investment-grade (Ba1/BB+) bonds. Lee would then note on which side of the margin the new data point (Biotron's new bonds) lies.

The KNN algorithm is also well suited for classification problems because it classifies a new observation by finding similarities (or nearness) between the new observation and the existing data. Training the algorithm with data as for SVM, the decision rule for classifying Biotron's new bonds is which classification is in the majority among its *k*-nearest neighbors. Note that *k* (a hyperparameter) must be pre-specified by Lee.

Solution to 3:

If the ML algorithms disagreed on the classification, the classification would be more likely to be sensitive to the algorithm's approach to classifying data. Because the classification of Biotron's new issue appears robust to the choice of ML algorithm (i.e., both algorithms agree on the rating), the resulting classification will more likely be correct.

EXAMPLE 4

CART and Ensemble Learning

Laurie Kim is a portfolio manager at Hilux LLC, a high-yield bond investment firm. The economy has been in recession for several months, and high-yield bond prices have declined precipitously as credit spreads have widened in response to the weak macroeconomic environment. Kim, however, believes this is a good time to buy because she expects to profit as credit spreads narrow and high-yield bond prices rise in anticipation of economic recovery.

Based on her analysis, Kim believes that corporate high-yield bonds in the credit quality range of B/B2 to CCC/Caa2 are the most attractive. However, she must carefully select which bonds to buy and which bonds to avoid because of the elevated default risk caused by the currently weak economy.

To help with her bond selection, Kim turns to Hilux's data analytics team. Kim has supplied them with historical data consisting of 19 fundamental and 5 technical factors for several thousand high-yield bond issuers and issues labeled to indicate default or no default. Kim requests that the team develop an ML-based model using all the factors provided that will make accurate classifications in two categories: default and no default. Exploratory data analysis suggests considerable non-linearities among the feature set.

1 State the type of problem being addressed by Kim.
2 Describe the dimensionality of the model that Kim requests her analytics team to develop.
3 Evaluate whether a CART model is appropriate for addressing her problem.
4 Describe how a CART model operates at each node of the tree.
5 Describe how the team might avoid overfitting and improve the predictive power of a CART model.
6 Describe how ensemble learning might be used by the team to develop even better predictions for Kim's selection of corporate high-yield bonds.

Solution to 1:

Kim is addressing a classification problem because she must determine whether bonds that she is considering purchasing in the credit quality range of B/B2 to CCC/Caa2 will default or not default.

Solution to 2:

With 19 fundamental and 5 technical factors (i.e., the features), the dimensionality of the model is 24.

Solution to 3:

The CART model is an algorithm for addressing classification problems. Its ability to handle complex, non-linear relationships makes it a good choice to address the modeling problem at hand. An important advantage of CART is that its results are relatively straightforward to visualize and interpret, which should help Kim explain her recommendations based on the model to Hilux's investment committee and the firm's clients.

Solution to 4:

At each node in the decision tree, the algorithm will choose the feature and the cutoff value for the selected feature that generates the widest separation of the labeled data to minimize classification error.

Solution to 5:

The team can avoid overfitting and improve the predictive power of the CART model by adding regularization parameters. For example, the team could specify the maximum depth of the tree, the minimum population at a node, or the maximum number of decision nodes. The iterative process of building nodes will be stopped once the regularization criterion has been reached. Alternatively, a pruning technique can be used afterward to remove parts of the CART model that provide little power to correctly classify instances into default or no default categories.

Solution to 6:

The analytics team might use ensemble learning to combine the predictions from a collection of models, where the average result of many predictions leads to a reduction in noise and thus more accurate predictions. Ensemble learning can be achieved by an aggregation of either heterogeneous learners—different types of algorithms combined with a voting classifier—or homogeneous learners—a combination of the same algorithm but using different training data based on the bootstrap aggregating (i.e., bagging) technique. The team may also consider developing a random forest classifier (i.e., a collection of many decision trees) trained via a bagging method.

CASE STUDY: CLASSIFICATION OF WINNING AND LOSING FUNDS

The following case study was developed and written by Matthew Dixon, PhD, FRM.

A research analyst for a fund of funds has been tasked with identifying a set of attractive exchange-traded funds (ETFs) and mutual funds (MFs) in which to invest. She decides to use machine learning to identify the best (i.e., winners) and worst (i.e., losers) performing funds and the features which are most important in such an identification. Her aim is to train a model to correctly classify the winners and losers and then to use it to predict future outperformers. She is unsure of which type of machine learning classification model (i.e., classifier) would work best, so she reports and cross-compares her findings using several different well-known machine learning algorithms.

The goal of this case is to demonstrate the application of machine learning classification to fund selection. Therefore, the analyst will use the following classifiers to identify the best and worst performing funds:

- classification and regression tree (CART),

- support vector machine (SVM),
- *k*-nearest neighbors (KNN), and
- random forests.

Data Description

In the following experiments, the performance of each fund is learned by the machine learning algorithms based on fund type and size, asset class composition, fundamentals (i.e., valuation multiples), and sector composition characteristics. To form a cross-sectional classifier, the sector composition and fund size reported on 15 February 2019 are assumed to be representative of the latest month over which the fund return is reported. Exhibit 12 presents a description of the dataset.

Exhibit 12 Dataset Description

Dataset: MF and ETF Data

There are two separate datasets, one for MFs and one for ETFs, consisting of fund type, size, asset class composition, fundamental financial ratios, sector weights, and monthly total return labeled to indicate the fund as being a winner, a loser, or neither. Number of observations: 6,085 MFs and 1,594 ETFs.

Features: Up to 21, as shown below:

General (six features):

1 cat_investment*: Fund type, either "blend," "growth," or "value"

2 net_assets: Total net assets in US dollars

3 cat_size: Investment category size, either "small," "medium," or "large" market capitalization stocks

4 portfolio_cash**: The ratio of cash to total assets in the fund

5 portfolio_stocks: The ratio of stocks to total assets in the fund

6 portfolio_bonds: The ratio of bonds to total assets in the fund

Fundamentals (four features):

1 price_earnings: The ratio of price per share to earnings per share

2 price_book: The ratio of price per share to book value per share

3 price_sales: The ratio of price per share to sales per share

4 price_cashflow: The ratio of price per share to cash flow per share

Sector weights (for 11 sectors) provided as percentages:

1 basic_materials

2 consumer_cyclical

3 financial_services

4 real_estate

5 consumer_defensive

6 healthcare

7 utilities

8 communication_services

9 energy

10 industrials

11 technology

Exhibit 12 (Continued)

Labels:

Winning and losing ETFs or MFs are determined based on whether their returns are one standard deviation or more above or below the distribution of one-month fund returns across all ETFs or across all MFs, respectively. More precisely, the labels are:

1, if fund_return_1 month ≥ mean(fund_return_1 month) + one std.dev(-fund_return_1 month), indicating a winning fund;

-1, if fund_return_1 month ≤ mean(fund_return_1 month) – one std.dev(-fund_return_1 month), indicating a losing fund; and

0, otherwise.

*Feature appears in the ETF dataset only.
**Feature appears in the MF dataset only.
Data sources: Kaggle, Yahoo Finance on 15 February 2019.

Methodology

The classification model is trained to determine whether a fund's performance is one standard deviation or more above the mean return (Label 1), within one standard deviation of the mean return (Label 0), or one standard deviation or more below the mean return (Label -1), where the mean return and standard deviation are either for all ETFs or all MFs, depending on the particular fund's type (ETF or MF). Performance is based on the one-month return of each fund as of 15 February 2019.

This procedure results in most of the funds being labeled as "0" (or average). After removing missing values in the dataset, there are 1,594 and 6,085 observations in the ETF and MF datasets, respectively. The data table is a 7,679 × 22 matrix, with 7,679 rows for each fund observation (1,594 for ETFs and 6,085 for MFs) and 22 columns for the 21 features plus the return label, and all data are recorded as of 15 February 2019.

The aim of the experiment is to identify not only winning and losing funds but also the features which are useful for distinguishing winners from losers. An important caveat, however, is that no claim is made that such features are causal.

A separate multi-classifier, with three classes, is run for each dataset. Four types of machine learning algorithms are used to build each classifier: (i) CART, (ii) SVM, (iii) KNN, and (iv) random forest. Random forest is an example of an ensemble method (based on bagging), whereas the other three algorithms do not use bagging.

A typical experimental design would involve using 70% of the data for training and holding 15% for tuning model hyperparameters and the remaining 15% of the data for testing. For simplicity, we shall not tune the hyperparameters but simply use the default settings without attempting to fine tune each one for best performance. So, in this case, we do not withhold 15% of the data for validation but instead train the classifier on a random split of 70% of the dataset, with the remaining 30% of the dataset used for testing. Crucially, for fairness of evaluation, each algorithm is trained and tested on identical data: The same 70% of observations are used for training each algorithm, and the same 30% are used for testing each one. The most important hyperparameters and settings for the algorithms are shown in Exhibit 13.

Exhibit 13 Parameter Settings for the Four Machine Learning Classifiers

1 CART: maximum tree depth: 5 levels

(continued)

Exhibit 13 (Continued)

2 SVM: cost parameter: 1.0

3 KNN: number of nearest neighbors: 4

4 Random forest: number of trees: 100; maximum tree depth: 20 levels

The choices of hyperparameter values for the four machine learning classifiers are supported by theory, academic research, practice, and experimentation to yield a satisfactory bias–variance trade-off. For SVM, the cost parameter is a penalty on the margin of the decision boundary. A large cost parameter forces the SVM to use a thin margin, whereas a smaller cost parameter widens the margin. For random forests, recall that this is an ensemble method which uses multiple decision trees to classify, typically by majority vote. Importantly, no claim is made that these choices of hyperparameters are universally optimal for any dataset.

Results

The results of each classifier are evaluated separately on the test portion of the ETF and MF datasets. The evaluation metrics used are based on Type I and Type II classification errors, where a Type I error is a false positive (FP) and a Type II error is a false negative (FN). Correct classifications are true positive (TP) and true negative (TN).

- The first evaluation metric is **accuracy**, the percentage of correctly predicted classes out of total predictions. So, high accuracy implies low Type I and Type II errors.
- **F1 score**, the second evaluation metric, is the weighted average of precision and recall. Precision is the ratio of correctly predicted positive classes to all predicted positive classes, and recall is the ratio of correctly predicted positive classes to all actual positive classes.

F1 score is a more appropriate evaluation metric to use than accuracy when there is unequal class distribution ("class imbalance") in the dataset, as is the case here. As mentioned, most of the funds in the ETF and MF datasets are designated as "0," indicating average performers.

Exhibit 14 shows the comparative performance results for each algorithm applied to the ETF dataset. These results show the random forest model is the most accurate (0.812), but once class imbalance is accounted for using F1 score (0.770), random forest is about as good as CART. Generally, ensemble methods, such as random forest, are expected to be at least as good as their single-model counterparts because ensemble forecasts generalize better out-of-sample. Importantly, while the relative accuracies and F1 scores across the different methods provide a basis for comparison, they do not speak to the absolute performance. In this regard, values approaching 1 suggest an excellent model, whereas values of approximately 1/3 would indicate the model is useless: 1/3 is premised on three (+1, 0, -1) equally distributed labels. However, because the distribution of classes is often not balanced, this ratio typically requires some adjustment.

Exhibit 14 Comparison of Accuracy and F1 Score for Each Classifier Applied to the ETF Dataset

	CART	SVM	KNN	Random Forest
Accuracy	0.770	0.774	0.724	0.812
F1 score	0.769	0.693	0.683	0.770

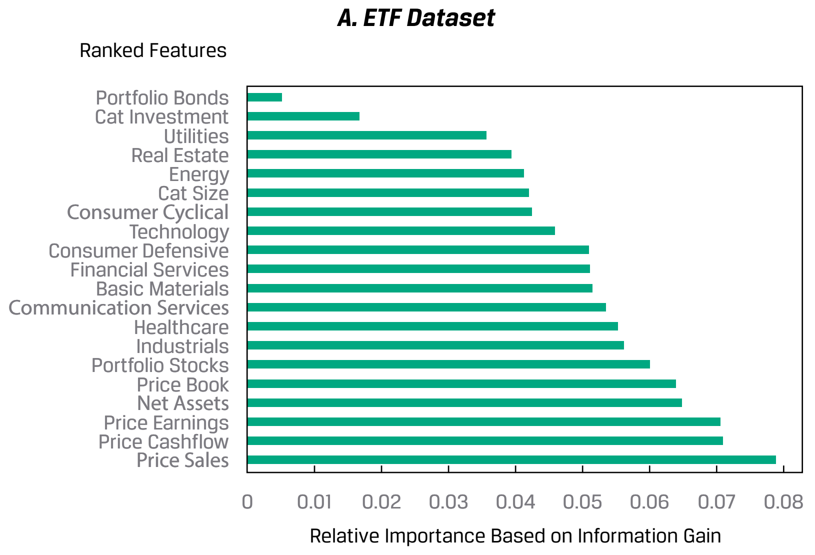
Exhibit 15 shows that the random forest model outperforms all the other classifiers under both metrics when applied to the MF dataset. Overall, the accuracy and F1 score for the SVM and KNN methods are similar for each dataset, and these algorithms are dominated by CART and random forest, especially in the larger MF dataset. The difference in performance between the two datasets for all the algorithms is to be expected, since the MF dataset is approximately four times larger than the ETF dataset and a larger sample set generally leads to better model performance. Moreover, the precise explanation of why random forest and CART outperform SVM and KNN is beyond the scope of this case. Suffice it to say that random forests are well known to be more robust to noise than most other classifiers.

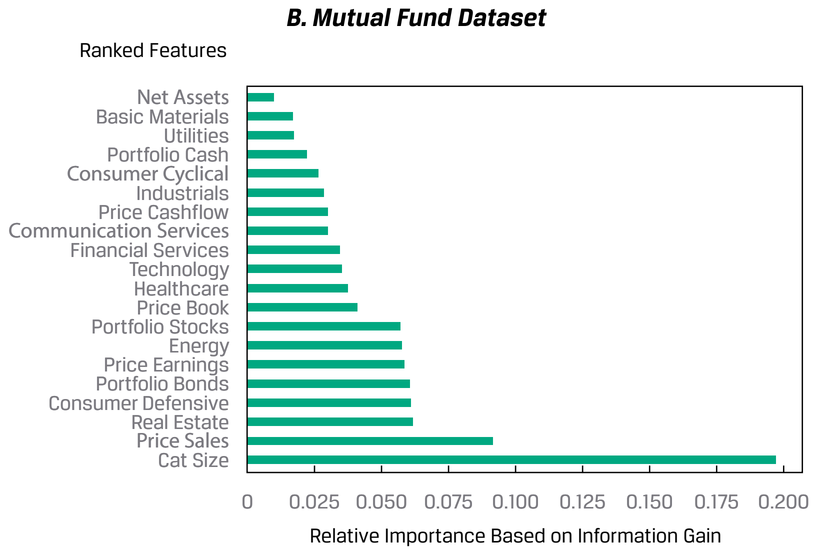
Exhibit 15 Comparison of Accuracy and F1 Score for Each Classifier Applied to the Mutual Fund Dataset

	CART	SVM	KNN	Random Forest
Accuracy	0.959	0.859	0.856	0.969
F1 score	0.959	0.847	0.855	0.969

Exhibit 16 presents results on the relative importance of the features in the random forest model for both the ETF (Panel A) and MF (Panel B) datasets. Relative importance is determined by **information gain**, which quantifies the amount of information that the feature holds about the response. Information gain can be regarded as a form of non-linear correlation between Y and X. Note the horizontal scale of Panel B (MF dataset) is more than twice as large as that of Panel A (ETF dataset), and the bar colors represent the feature rankings, not the features themselves.

Exhibit 16 Relative Importance of Features in the Random Forest Model





The prices-to-sales (price_sales) and prices-to-earnings (price_earnings) ratios are observed to be important indicators of performance, at about 0.08–0.09 and 0.06–0.07, respectively, in the random forest models for each dataset. The ratio of stocks to total assets (portfolio_stocks), at 0.06, is another key feature. Moreover, the industrials, health care, and communication services sector weightings are relatively important in the ETF dataset, while the real estate, consumer defensive, and energy sector weightings are key features in the MF dataset for differentiating between winning and losing funds.

Another important observation is that the category of the fund size (cat_size) is by far the most important feature in the model's performance for the MF dataset (≈ 0.20), whereas it is of much less importance for model performance using the ETF dataset (≈ 0.04). Conversely, net assets is a relatively important feature for model performance using the ETF dataset (0.065), while it is the least important feature when the random forest model is applied to the MF dataset (0.01).

Conclusion

The research analyst has trained and tested machine learning–based models that she can use to identify potential winning and losing ETFs and MFs. Her classification models use input features based on fund type and size, asset class composition, fundamentals, and sector composition characteristics. She is more confident in her assessment of MFs than of ETFs, owing to the substantially larger sample size of the former. She is also confident that any imbalance in class has not led to misinterpretation of her models' results, since she uses F1 score as her primary model evaluation metric. Moreover, she determines that the best performing model using both datasets is an ensemble-type random forest model. Finally, she concludes that while fundamental ratios, asset class ratios, and sector composition are important features for both models, net assets and category size also figure prominently in discriminating between winning and losing ETFs and MFs.

EXAMPLE 5

Classification of Funds

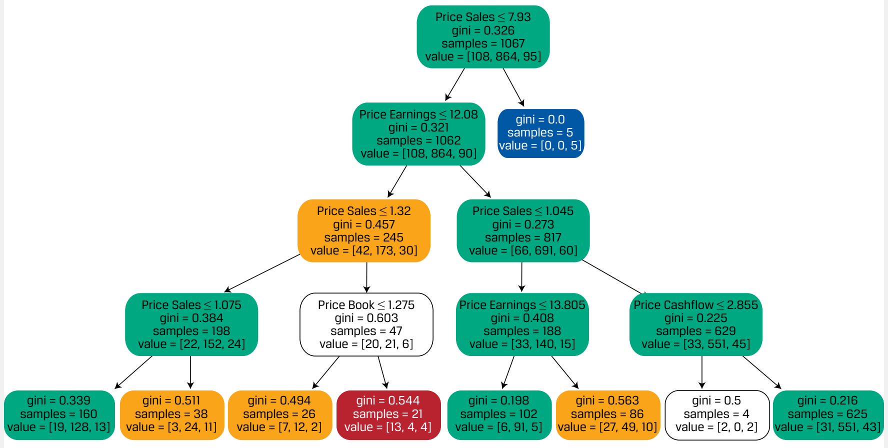
The research analyst from the previous case uses CART to generate the decision tree shown in Exhibit 17, which she will use to predict whether and explain why a new ETF is likely to be a winner (+1), an average performer (0), or a loser (-1). This ETF's fundamental valuation ratios are as follows: Price-to-sales = 2.29, price-to-earnings = 7.20, price-to-book = 1.41, and price-to-cash flow = 2.65. Note that the sample size is 1,067 ETFs and the CART model uses just valuation ratios, because these are deemed the most important features for ETF performance classification.

1. Explain the CART model's prediction for performance of the new ETF: winner, loser, or average performer.

2. Calculate the probability that the fund will be in the class predicted by the CART model.

3. Explain why the analyst should be cautious in basing the ETF's predicted performance solely on the CART-generated decision tree.

Exhibit 17 CART-Based Decision Tree for EFT Performance Classification



Price Sales ≤ 7.93 gini = 0.326 samples = 1067 value = [108, 864, 95]
Price Earnings ≤ 12.08 gini = 0.321 samples = 1062 value = [108, 864, 90]
gini = 0.0 samples = 5 value = [0, 0, 5]
Price Sales ≤ 1.32 gini = 0.457 samples = 245 value = [42, 173, 30]
Price Sales ≤ 1.045 gini = 0.273 samples = 817 value = [66, 691, 60]
Price Sales ≤ 1.075 gini = 0.384 samples = 198 value = [22, 152, 24]
Price Book ≤ 1.275 gini = 0.603 samples = 47 value = [20, 21, 6]
Price Earnings ≤ 13.805 gini = 0.408 samples = 188 value = [33, 140, 15]
Price Cashflow ≤ 2.855 gini = 0.225 samples = 629 value = [33, 551, 45]
gini = 0.339 samples = 160 value = [19, 128, 13]
gini = 0.511 samples = 38 value = [3, 24, 11]
gini = 0.494 samples = 26 value = [7, 12, 2]
gini = 0.544 samples = 21 value = [13, 4, 4]
gini = 0.198 samples = 102 value = [6, 91, 5]
gini = 0.563 samples = 86 value = [27, 49, 10]
gini = 0.5 samples = 4 value = [2, 0, 2]
gini = 0.216 samples = 625 value = [31, 551, 43]

Legend:
• Darkest shade, 5th (last) level: Winner (Class = +1)

(continued)

Exhibit 17 (Continued)

- Light to medium shade: Average Performer (Class = 0); note that the medium shade indicates more confidence in the classification.
- Darkest shade, 2nd level: Loser (Class = -1)
- White: Inconclusive, either because there is a tie with multiple categories or there are too few samples
- Value: The number of sample cases in each of the three classes: Winner, Average Performer, or Loser
- Path: Left path is True and right path is False.

Solution to 1:

Based on its valuation ratios (P/S = 2.29; P/E = 7.20; P/B = 1.41), the new ETF is predicted to be a winner because the decision path leads to the dark shaded, 5th level ("winner") terminal node. The split criteria and decisions are as follows:

Initial node: P/S ≤ 7.93 and EFT P/S = 2.29, so True.

2nd-level node: P/E ≤ 12.08 and EFT P/E = 7.20, so True.

3rd-level node: P/S ≤ 1.32 and EFT P/S = 2.29, so False.

4th-level node: P/B ≤ 1.275 and EFT P/B = 1.41, so False.

5th-level (terminal) node: darkest shaded terminal node indicates "winner."

Solution to 2:

The output from the CART model in the darkest shaded, 5th level (winner) terminal node is [13, 4, 4], which indicates it includes 13 funds of Class +1 (winners), 4 funds of Class 0 (average performers), and 4 funds of Class -1 (losers). Thus, the probability predicted by the CART model that this ETF will be in the "winner" class is 13/21, or 62%. There are also equal probabilities of it being an average performer (19%) or a loser (19%).

Solution to 3:

There are several reasons why the analyst should be cautious in basing the ETF's predicted performance solely on the CART-generated decision tree. First, this CART model had a maximum depth of just five levels. Truncating at five levels facilitates visualization, but a more realistic decision path is likely to be nuanced and so would require greater depth. Second, only some of the important variables (from Exhibit 16) were used in generating this tree, again for simplicity of visualization. A CART model using additional features, including fund asset class ratios, sector composition, and, especially, net assets would be expected to generate a more accurate (using F1 score) model. Finally, the number of funds reaching the darkest shaded, 5th level ("winner") terminal node (21) is small compared to the total sample size (1,067), so there may be too few clear winners (13) under this decision path from which to draw a statistically significant conclusion. Besides increasing the maximum tree depth and adding more features, another approach the analyst might take in this case for achieving a more accurate model is random forest; being an ensemble classifier, a random forest model would generalize out-of-sample better than any single CART model.

UNSUPERVISED MACHINE LEARNING ALGORITHMS

6

d Describe unsupervised machine learning algorithms—including principal components analysis, K-means clustering, and hierarchical clustering—and determine the problems for which they are best suited

Unsupervised learning is machine learning that does not use labeled data (i.e., no target variable); thus, the algorithms are tasked with finding patterns within the data themselves. The two main types of unsupervised ML algorithms shown in Exhibit 2 are dimension reduction, using principal components analysis, and clustering, which includes *k*-means and hierarchical clustering. These will now be described in turn.

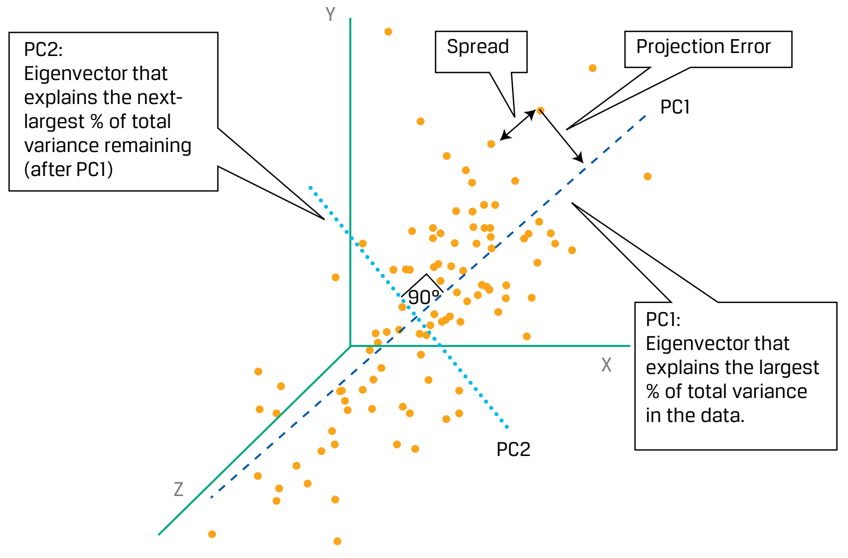
6.1 Principal Components Analysis

Dimension reduction is an important type of unsupervised learning that is used widely in practice. When many features are in a dataset, representing the data visually or fitting models to the data may become extremely complex and "noisy" in the sense of reflecting random influences specific to a dataset. In such cases, dimension reduction may be necessary. Dimension reduction aims to represent a dataset with many typically correlated features by a smaller set of features that still does well in describing the data.

A long-established statistical method for dimension reduction is **principal components analysis (PCA)**. PCA is used to summarize or transform highly correlated features of data into a few main, uncorrelated composite variables. A **composite variable** is a variable that combines two or more variables that are statistically strongly related to each other. Informally, PCA involves transforming the covariance matrix of the features and involves two key concepts: eigenvectors and eigenvalues. In the context of PCA, **eigenvectors** define new, mutually uncorrelated composite variables that are linear combinations of the original features. As a vector, an eigenvector also represents a direction. Associated with each eigenvector is an eigenvalue. An **eigenvalue** gives the proportion of total variance in the initial data that is explained by each eigenvector. The PCA algorithm orders the eigenvectors from highest to lowest according to their eigenvalues—that is, in terms of their usefulness in explaining the total variance in the initial data (this will be shown shortly using a scree plot). PCA selects as the first principal component the eigenvector that explains the largest proportion of variation in the dataset (the eigenvector with the largest eigenvalue). The second principal component explains the next-largest proportion of variation remaining after the first principal component; this process continues for the third, fourth, and subsequent principal components. Because the principal components are linear combinations of the initial feature set, only a few principal components are typically required to explain most of the total variance in the initial feature covariance matrix.

Exhibit 18 shows a hypothetical dataset with three features, so it is plotted in three dimensions along the *x*-, *y*-, and *z*-axes. Each data point has a measurement (x, y, z), and the data should be standardized so that the mean of each series (x's, y's, and z's) is 0 and the standard deviation is 1. Assume PCA has been applied, revealing the first two principal components, PC1 and PC2. With respect to PC1, a perpendicular line dropped from each data point to PC1 shows the vertical distance between the data point and PC1, representing **projection error**. Moreover, the distance between each data point in the direction that is parallel to PC1 represents the spread or variation of the data along PC1. The PCA algorithm operates in such a way that it finds PC1 by selecting the line for which the sum of the projection errors for all data points is minimized and for which the sum of the spread between all the data is maximized. As a consequence of these selection criteria, PC1 is the unique vector that accounts for the largest proportion of the variance in the initial data. The next-largest portion of the remaining variance is best explained by PC2, which is at right angles to PC1 and thus is uncorrelated with PC1. The data points can now be represented by the first two principal components. This example demonstrates the effectiveness of the PCA algorithm in summarizing the variability of the data and the resulting dimension reduction.

Exhibit 18 First and Second Principal Components of a Hypothetical Three-Dimensional Dataset


Y
PC2:
Eigenvector that explains the next-largest % of total variance remaining (after PC1)
Spread
Projection Error
PC1
90°
PC1:
Eigenvector that explains the largest % of total variance in the data.
X
PC2
Z

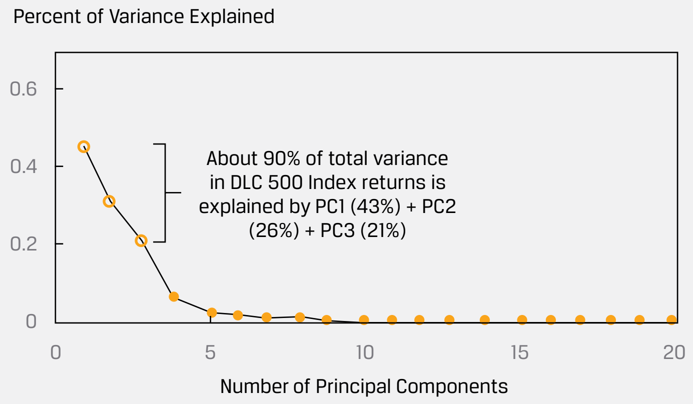
It is important to know how many principal components to retain because there is a trade-off between a lower-dimensional, more manageable view of a complex dataset when a few are selected and some loss of information. **Scree plots**, which show the proportion of total variance in the data explained by each principal component, can be helpful in this regard (see the accompanying sidebar). In practice, the smallest number of principal components that should be retained is that which the scree plot shows as explaining a desired proportion of total variance in the initial dataset (often 85% to 95%).

Scree Plots for the Principal Components of Returns to the Hypothetical DLC 500 and VLC 30 Equity Indexes

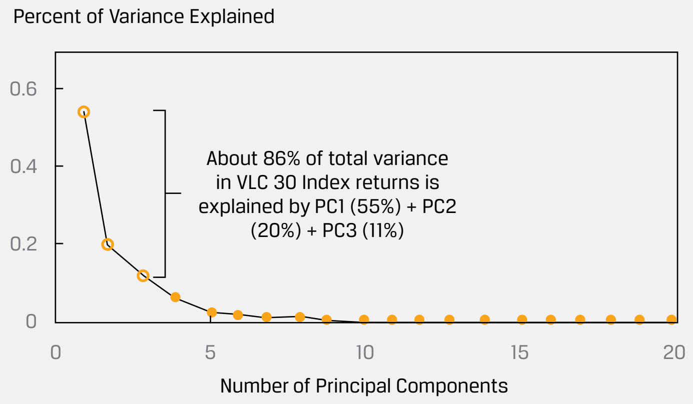
In this illustration, researchers use scree plots and decide that three principal components are sufficient for explaining the returns to the hypothetical Diversified Large Cap (DLC) 500 and Very Large Cap (VLC) 30 equity indexes over the last 10-year period. The DLC 500 can be thought of as a diversified index of large-cap companies covering all economic sectors, while the VLC 30 is a more concentrated index of the 30 largest publicly traded companies. The dataset consists of index prices and more than 2,000 fundamental and technical features. Multi-collinearity among the features is a typical problem because that many features or combinations of features tend to have overlaps. To mitigate the problem, PCA can be used to capture the information and variance in the data. The following scree plots show that of the 20 principal components generated, the first 3 together explain about 90% and 86% of the variance in the value of the DLC 500 and VLC 30 indexes, respectively. The scree plots indicate that for each of these indexes, the incremental contribution to explaining the variance structure of the data is quite small after about the fifth principal component. Therefore, these less useful principal components can be ignored without much loss of information.

Scree Plots of Percent of Total Variance Explained by Each Principal Component for Hypothetical DLC 500 and VLC 30 Equity Indexes

A. Diversified Large Cap 500 Index (DLC 500)



B. Very Large Cap 30 Index (VLC 30)



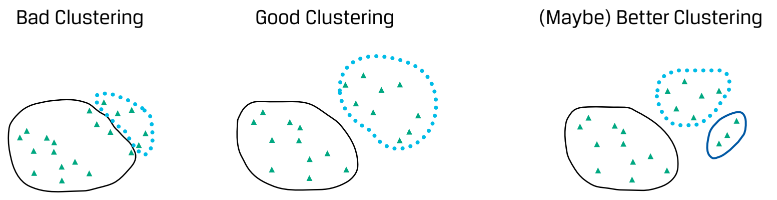
The main drawback of PCA is that since the principal components are combinations of the dataset's initial features, they typically cannot be easily labeled or directly interpreted by the analyst. Compared to modeling data with variables that represent well-defined concepts, the end user of PCA may perceive PCA as something of a "black box."

Reducing the number of features to the most relevant predictors is very useful, even when working with datasets having as few as 10 or so features. Notably, dimension reduction facilitates visually representing the data in two or three dimensions. It is typically performed as part of exploratory data analysis, before training another supervised or unsupervised learning model. Machine learning models are quicker to train, tend to reduce overfitting (by avoiding the curse of dimensionality), and are easier to interpret if provided with lower-dimensional datasets.

6.2 Clustering

Clustering is another type of unsupervised machine learning, which is used to organize data points into similar groups called clusters. A **cluster** contains a subset of observations from the dataset such that all the observations within the same cluster are deemed "similar." The aim is to find a good clustering of the data—meaning that the observations inside each cluster are similar or close to each other (a property known as cohesion) and the observations in two different clusters are as far away from one another or are as dissimilar as possible (a property known as separation). Exhibit 19 depicts this intra-cluster cohesion and inter-cluster separation.

Exhibit 19 Evaluating Clustering—Intra-Cluster Cohesion and Inter-Cluster Separation



Clustering algorithms are particularly useful in the many investment problems and applications in which the concept of similarity is important. Applied to grouping companies, for example, clustering may uncover important similarities and differences among companies that are not captured by standard classifications of companies by industry and sector. In portfolio management, clustering methods have been used for improving portfolio diversification.

In practice, expert human judgment has a role in using clustering algorithms. In the first place, one must establish what it means to be "similar." Each company can be considered an observation with multiple features, including such financial statement items as total revenue and profit to shareholders, a wide array of financial ratios, or any other potential model inputs. Based on these features, a measure of similarity or "distance" between two observations (i.e., companies) can be defined. The smaller the distance, the more similar the observations; the larger the distance, the more dissimilar the observations.

A commonly used definition of distance is the Euclidian distance, the straight-line distance between two points. A closely related distance useful in portfolio diversification is correlation, which is the average Euclidian distance between a set of standardized points. Roughly a dozen different distance measures are used regularly in ML. In practice, the choice of the distance measures depends on the nature of the data (numerical or not) and the business problem being investigated. Once the relevant distance measure is defined, similar observations can be grouped together. We now introduce two of the more popular clustering approaches: *k*-means and hierarchical clustering.

6.2.1 K-*Means Clustering*

***K*-means** is an algorithm that repeatedly partitions observations into a fixed number, *k*, of non-overlapping clusters. The number of clusters, *k*, is a model hyperparameter. Each cluster is characterized by its **centroid** (i.e., center), and each observation is assigned by the algorithm to the cluster with the centroid to which that observation is closest. Notably, once the clusters are formed, there is no defined relationship between them.

The *k*-means algorithm follows an iterative process. It is illustrated in Exhibit 20 for *k* = 3 and a set of observations on a variable that can be described by two features. In Exhibit 20, the horizontal and vertical axes represent, respectively, the first and second features. For example, an investment analyst may want to group a set of firms into three groups according to two numerical measures of management quality. The algorithm groups the observations in the following steps:

1 *K*-means starts by determining the position of the *k* (here, 3) initial random centroids.

2 The algorithm then analyzes the features for each observation. Based on the distance measure that is used, *k*-means assigns each observation to its closest centroid, which defines a cluster.

3 Using the observations within each cluster, *k*-means then calculates the new (*k*) centroids for each cluster, where the centroid is the average value of their assigned observations.

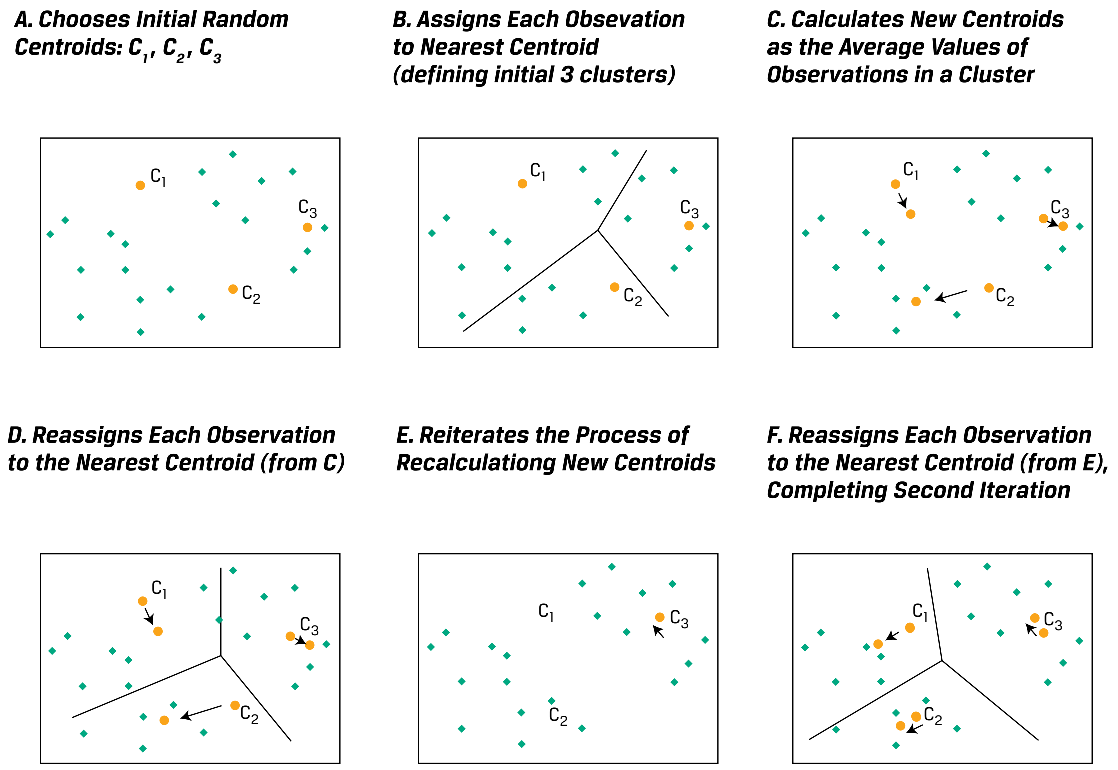
4 *K*-means then reassigns the observations to the new centroids, redefining the clusters in terms of included and excluded observations.

5 The process of recalculating the new (*k*) centroids for each cluster is reiterated.

6 *K*-means then reassigns the observations to the revised centroids, again redefining the clusters in terms of observations that are included and excluded.

The *k*-means algorithm will continue to iterate until no observation is reassigned to a new cluster (i.e., no need to recalculate new centroids). The algorithm has then converged and reveals the final *k* clusters with their member observations. The *k*-means algorithm has minimized intra-cluster distance (thereby maximizing cohesion) and has maximized inter-cluster distance (thereby maximizing separation) under the constraint that *k* = 3.

Exhibit 20 Example of 3-Means Algorithm


A. Chooses Initial Random Centroids: C_1, C_2, C_3
B. Assigns Each Obsevation to Nearest Centroid (defining initial 3 clusters)
C. Calculates New Centroids as the Average Values of Observations in a Cluster
D. Reassigns Each Observation to the Nearest Centroid (from C)
E. Reiterates the Process of Recalculationg New Centroids
F. Reassigns Each Observation to the Nearest Centroid (from E), Completing Second Iteration
C_1
C_2
C_3

The *k*-means algorithm is fast and works well on very large datasets, those with hundreds of millions of observations. However, the final assignment of observations to clusters can depend on the initial location of the centroids. To address this problem, the algorithm can be run several times using different sets of initial centroids, and then one can choose the clustering that is most useful given the business purpose.

One limitation of this technique is that the hyperparameter, *k*, the number of clusters in which to partition the data, must be decided before *k*-means can be run. So, one needs to have a sense of how many clusters are reasonable for the problem under investigation and the dataset being analyzed. Alternatively, one can run the algorithm using a range of values for *k* to find the optimal number of clusters—the *k* that minimizes intra-cluster distance and thereby maximizes intra-cluster similarity (i.e., cohesion) and that maximizes inter-cluster distance (i.e., separation). However, note that the final results can be subjective and dependent on the context of the problem and the particular training set. In practice, it is common to make the final choice of *k* based on face validity, such that the clusters feel sensible and are interpretable. This decision is greatly assisted by using summary information about the centroids and ranges of values and naming example items in each cluster.

For example, consider the Russell 3000 Index, which tracks the 3,000 highest market capitalization stocks in the United States. These 3,000 stocks can be grouped in 10, 50, or even more clusters based on their financial characteristics (e.g., total assets, total revenue, profitability, leverage) and operating characteristics (e.g., employee headcount, R&D intensity). Because companies in the same standard industry classification can have very different financial and operating characteristics, using *k*-means to derive different clusters can provide insights and understanding into the nature of "peer" groups. As mentioned, the exact choice of the *k*, the number of clusters, will depend on the level of precision or segmentation desired. In a similar vein, clustering can be used to classify collective investment vehicles or hedge funds as an alternative to standard classifications. Clustering analysis can also help visualize the data and facilitate detecting trends or outliers.

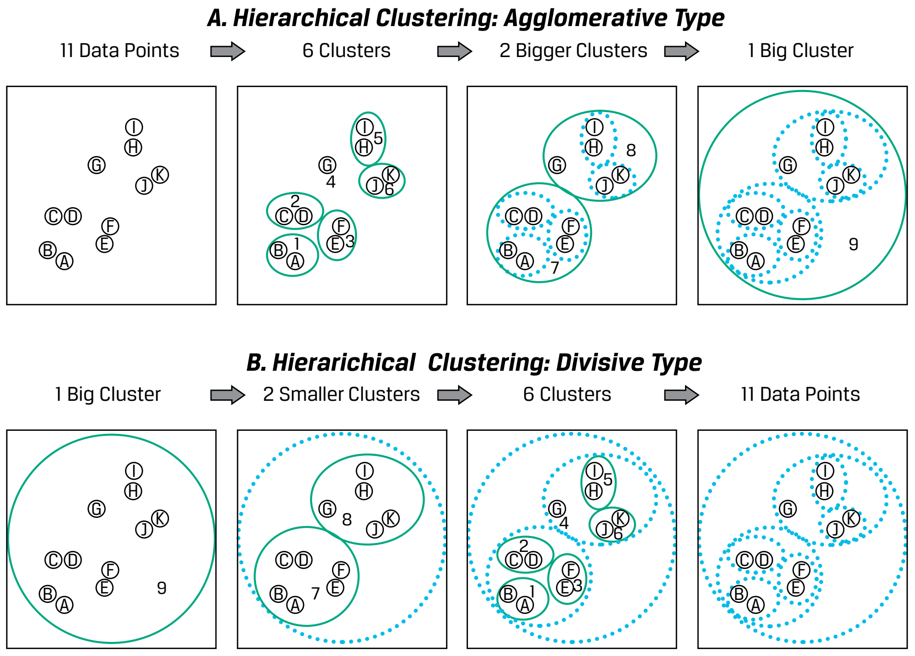
In sum, the *k*-means algorithm is among the most used algorithms in investment practice, particularly in data exploration for discovering patterns in high-dimensional data or as a method for deriving alternatives to existing static industry classifications.

6.2.2 *Hierarchical Clustering: Agglomerative and Divisive*

Hierarchical clustering is an iterative procedure used to build a hierarchy of clusters. In *k*-means clustering, the algorithm segments the data into a predetermined number of clusters; there is no defined relationship among the resulting clusters. In hierarchical clustering, however, the algorithms create intermediate rounds of clusters of increasing (in "agglomerative") or decreasing (in "divisive") size until a final clustering is reached. The process creates relationships among the rounds of clusters, as the word "hierarchical" suggests. Although more computationally intensive than *k*-means clustering, hierarchical clustering has the advantage of allowing the investment analyst to examine alternative segmentations of data of different granularity before deciding which one to use.

Agglomerative clustering (or bottom-up hierarchical clustering) begins with each observation being treated as its own cluster. Then, the algorithm finds the two closest clusters, defined by some measure of distance (similarity), and combines them into one new larger cluster. This process is repeated iteratively until all observations are clumped into a single cluster. A hypothetical example of how agglomerative clustering develops a hierarchical clustering scheme is depicted in the top part of Exhibit 21, where observations are lettered (A to K) and circles around observations denote clusters. The process begins with 11 individual clusters and then generates a sequence of groupings. The first sequence includes five clusters with two observations each and one cluster with a single observation, G, for a total of six clusters. It then generates two clusters—one cluster with six observations and the other with five observations. The final result is one large cluster containing all 11 observations. It is easily seen that this final large cluster includes the two main sub-clusters, with each containing three smaller sub-clusters.

Exhibit 21 Agglomerative and Divisive Hierarchical Clustering



By contrast, **divisive clustering** (or top-down hierarchical clustering) starts with all the observations belonging to a single cluster. The observations are then divided into two clusters based on some measure of distance (similarity). The algorithm then progressively partitions the intermediate clusters into smaller clusters until each cluster contains only one observation. Divisive clustering is depicted in the bottom part of Exhibit 21, which begins with all 11 observations in one large cluster. Next, the algorithm generates two smaller clusters, one with six observations and the other with five observations, and then six clusters, with two observations each except for observation G, which is its own cluster. Finally, 11 clusters are generated, with each cluster containing only one observation.

Although this is not a typical outcome (because the two methods generally use different algorithms), in this hypothetical illustration, the agglomerative and divisive clustering methods produced the same result: two main sub-clusters each having three smaller sub-clusters. The analyst could decide between using a six- or a two-cluster representation of the data. The agglomerative method is the approach typically used with large datasets because of the algorithm's fast computing speed. The agglomerative clustering algorithm makes clustering decisions based on local patterns without initially accounting for the global structure of the data. As such, the agglomerative method is well suited for identifying small clusters. However, because the divisive method starts with a holistic representation of the data, the divisive clustering algorithm is designed to account for the global structure of the data and thus is better suited for identifying large clusters.

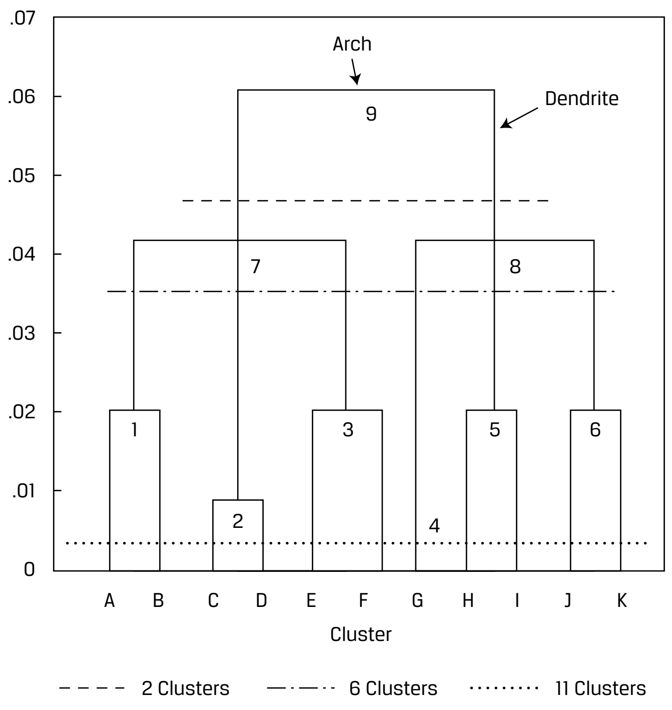
To decide on the closest clusters for combining in the agglomerative process or for dividing in the divisive process, an explicit definition for the distance between two clusters is required. Some commonly used definitions for the distance between two clusters involve finding the minimum, the maximum, or the average of the straight-line distances between all the pairs of observations in each cluster.

6.2.3 *Dendrograms*

A type of tree diagram for visualizing a hierarchical cluster analysis is known as a **dendrogram**, which highlights the hierarchical relationships among the clusters. Exhibit 22 shows a dendrogram representation for the clustering shown in Exhibit 21. First, a few technical points on dendrograms bear mentioning—although they may not all be apparent in Exhibit 22. The x-axis shows the clusters, and the y-axis indicates some distance measure. Clusters are represented by a horizontal line, the arch, which connects two vertical lines, called dendrites, where the height of each arch represents the distance between the two clusters being considered. Shorter dendrites represent a shorter distance (and greater similarity) between clusters. The horizontal dashed lines cutting across the dendrites show the number of clusters into which the data are split at each stage.

The agglomerative algorithm starts at the bottom of the dendrite, where each observation is its own cluster (A to K). Agglomerative clustering then generates the six larger clusters (1 to 6). For example, Clusters A and B combine to form Cluster 1, and Observation G remains its own cluster, now Cluster 4. Moving up the dendrogram, two larger clusters are formed, where, for example, Cluster 7 includes Clusters 1 to 3. Finally, at the top of the dendrogram is the single large cluster (9). The dendrogram readily shows how this largest cluster is composed of the two main sub-clusters (7 and 8), each having three smaller sub-clusters (1 to 3 and 4 to 6, respectively). The dendrogram also facilitates visualization of divisive clustering by starting at the top of the largest cluster and then working downward until the bottom is reached, where all 11 single-observation clusters are shown.

Exhibit 22 Dendrogram of Agglomerative Hierarchical Clustering



Clustering has many applications in investment management. For example, portfolio diversification can be approached as a clustering problem with the aim of optimally diversifying risks by investing in assets from multiple different clusters. Because the clusters have maximum inter-cluster separation, diversifying among them helps ensure that the portfolio reflects a wide diversity of characteristics with well-diversified risk. In contrast, information that investments are concentrated in a cluster indicates a high probability of concentrated risk. Finally, it is important to note that while the results of clustering algorithms are often difficult to evaluate (because the resulting clusters themselves are not explicitly defined), they are still very useful in practice for uncovering important underlying structure (namely, similarities among observations) in complex datasets.

EXAMPLE 6

Investment Uses of Clustering Algorithms

István Perényi is a portfolio manager of the Europe Diversified Equity Fund ("the Fund") within the Diversified Investment Management Company (DIMCO) fund family. The Fund is benchmarked to the STOXX Europe 600 Index, which spans 17 countries, 19 industry sectors, and three market capitalization groupings (large-, mid-, and small-cap).

Examining the Fund's most recent performance, Perényi is concerned that the Fund's holdings, although approximately aligned with the STOXX Europe 600 Index's country weights, may have unrecognized risk biases and concentrations. Perényi asks Elsa Lund, DIMCO's chief risk officer, to investigate the Fund's diversification. Lund asks her analysts for ideas on how Perényi's request can be addressed and receives three suggestions:

Suggestion 1	Estimate the country, industry, and market cap exposures of each Fund holding, aggregate them, and compare the aggregate exposures to the benchmark's exposures. Then, examine mismatches for evidence of unexpected biases or concentrations.
Suggestion 2	Identify inherent groupings among fund holdings based on a broad set of eight numerical (operating and financial) measures related to the holdings' characteristics. Then, examine the groupings for evidence of unexpected biases or concentrations.
Suggestion 3	Regress the return of the Fund on a set of country equity market indexes and sector indexes based on the Fund's benchmark. Then, examine the regression coefficients for evidence of unexpected biases or concentrations.

Lund has several questions for analyst Greg Kane about using one or more clustering machine learning algorithms in relation to addressing Perényi's request.

Lund asks whether any information needs to be specified for the ML clustering algorithms no matter which one is used. Kane replies that only the distance measure that the algorithm will use and the hyperparameter, k, for k-means clustering need to be specified.

Lund further asks whether there would be an advantage to using k-means clustering as opposed to hierarchical clustering. Kane replies that in his opinion, hierarchical clustering is the more appropriate algorithm.

1 Which analyst suggestion is *most likely* to be implemented using machine learning?

A Suggestion 1

B Suggestion 2

C Suggestion 3

2 Kane's reply to Lund's first question about specification of ML clustering models is:

- **A** correct.
- **B** not correct, because other hyperparameters must also be specified.
- **C** not correct, because the feature set for describing the measure used to group holdings must also be specified.

3 The best justification for Kane's preference for hierarchical clustering in his reply to Lund's second question is that Kane is *most likely* giving consideration to:

- **A** the speed of the algorithms.
- **B** the dimensionality of the dataset.
- **C** the need to specify the hyperparameter, *k*, in using a *k*-means algorithm.

Solution to 1:

B is correct. A machine learning clustering algorithm could be used to implement Suggestion 2. A and C are incorrect because Suggestions 1 and 3, respectively, can be addressed easily using traditional regression analysis.

Solution to 2:

C is correct. Beyond specifying a distance measure and the *k* for *k*-means, whichever clustering algorithm is selected, the feature set used to group holdings by similarities must also be specified. Operating and financial characteristics of the companies represented in the Fund's portfolio are examples of such features.

Solution to 3:

C is correct. The value of the hyperparameter, *k*, the number of distinct groups into which the STOXX Europe 600 Index can be segmented, is not known and needs to be specified in advance by the analyst. Using a hierarchical algorithm, the sorting of observations into clusters will occur without any prior input on the analyst's part.

CASE STUDY: CLUSTERING STOCKS BASED ON CO-MOVEMENT SIMILARITY

The following case study was developed and written by Matthew Dixon, PhD, FRM.

An endowment fund's Investment Committee is seeking three "buy" recommendations for the fund's large-cap equity portfolio. An analyst working for the Investment Committee is given a subset of eight stocks from the S&P 500 Index and asked to determine the co-movement similarity (i.e., correlation) of their returns. Specifically, for diversification purposes, the Investment Committee wants the correlation of returns between the recommended stocks to be low, so the analyst decides to use clustering to identify the most similar stocks and then choose one stock from each cluster. Although this case study focuses mainly on hierarchical agglomerative clustering, the analyst's results using other clustering algorithms (i.e., divisive clustering and *k*-means) are also briefly discussed. Exhibit 23 provides a description of the data used by the analyst.

Exhibit 23 Dataset of Eight Stocks from the S&P 500 Index

Description: Daily adjusted closing prices of eight S&P 500 member stocks

Trading Dates: 30 May 2017 to 24 May 2019

Number of Observations: 501

Stocks (Ticker Symbols): AAPL, F, FB, GM, GS, GOOG, JPM, and UBS

The following steps are taken by the analyst to perform the hierarchical agglomerative cluster analysis:

1 Collect panel data on adjusted closing prices for the stocks under investigation.

2 Calculate the daily log returns for each stock, where each time series of stock returns is an n-vector ($n = 500$).

3 Run the agglomerative hierarchical clustering algorithm.

  a The algorithm calculates the pairwise distance (i.e., Euclidean distance) between vectors of any two stocks' returns. Each pairwise distance is an element of a distance matrix (i.e., dissimilarity matrix) with zero diagonals.

  b The algorithm starts with each stock as its own cluster, finds the pair of clusters which are closest to each other, and then redefines them as a new cluster.

  c The algorithm finds the distances from this new cluster to the remaining return clusters. Using a process called average (centroid) linkage, it determines the distances from the center of the new cluster to the centers of the remaining clusters. Note that there are several other linkage methods, but whichever method is selected, the algorithm proceeds in the same fashion: It combines the pair of clusters which are closest, redefines them as a new cluster, and recalculates the distances to the remaining clusters.

4 Repeat Step 3c until the data are aggregated into a single large cluster.

5 Plot the resulting dendrogram to visualize the hierarchical clusters and draw the highest horizontal line intersecting three (i.e., the desired number of clusters, since the Investment Committee wants three "buy" recommendations) vertical lines (or dendrites) to determine the appropriate cluster configuration.

Exhibit 24 shows for illustrative purposes a subset of the panel data on daily returns, calculated from the adjusted closing prices of the eight stocks collected in Step 1. The clustering is performed on the daily returns.

Exhibit 24 Subset of Stock Returns, Calculated from Adjusted Closing Prices, for Clustering

Date	JPM	UBS	GS	FB	AAPL	GOOG	GM	F
2017-05-31	–0.021	–0.007	–0.033	–0.006	–0.006	–0.011	0.012	0.004
2017-06-01	0.011	0.013	0.018	0.000	0.003	0.002	0.015	0.026
2017-06-02	–0.005	–0.002	–0.008	0.014	0.015	0.009	0.001	–0.005
2017-06-05	0.002	–0.007	0.003	0.000	–0.010	0.008	0.000	–0.009
2017-06-06	0.002	0.002	0.003	–0.005	0.003	–0.007	–0.001	–0.012

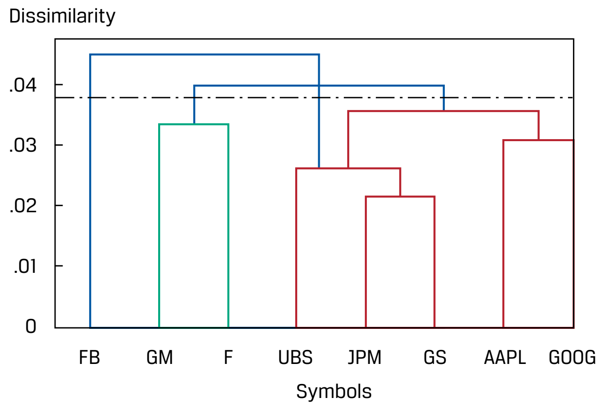
The results of the remaining steps are described using the distance matrix shown in Exhibit 25.

Exhibit 25 Distance Matrix for Hierarchical Agglomerative Clustering

	JPM	UBS	GS	FB	AAPL	GOOG	GM	F
JPM	0.000	0.243	0.215	0.456	0.364	0.332	0.358	0.348
UBS	0.243	0.000	0.281	0.460	0.380	0.338	0.384	0.385
GS	0.215	0.281	0.000	0.471	0.375	0.345	0.383	0.393
FB	0.456	0.460	0.471	0.000	0.437	0.357	0.491	0.480
AAPL	0.364	0.380	0.375	0.437	0.000	0.307	0.445	0.456
GOOG	0.332	0.338	0.345	0.357	0.307	0.000	0.405	0.422
GM	0.358	0.384	0.383	0.491	0.445	0.405	0.000	0.334
F	0.348	0.385	0.393	0.480	0.456	0.422	0.334	0.000

The distance matrix reveals the closest pair of stocks is JPM and GS, with a distance of 0.215. Therefore, this pair becomes the first combined cluster as shown in the dendrogram in Exhibit 26. Note that the vertical distance connecting the various clusters represents the Euclidean distance between clusters, so the arch between this pair has a height of 0.215. Now that JPM and GS are paired in a cluster (i.e., GS_JPM), we treat the mean of their two return vectors as a new point.

Exhibit 26 Dendrogram for Hierarchical Agglomerative Clustering



From the distance matrix, the average distance of UBS to the new cluster (i.e., GS_JPM) is the sum of the distance between UBS and JPM, 0.243, and the distance between UBS and GS, 0.281, divided by two, which is 0.262 (= (0.243 + 0.281)/2). Since this distance is smaller than the distance between any of the other unpaired stock clusters, UBS is merged with this cluster to create a new cluster (i.e., GS_JPM_UBS). The height of the arch in the dendrogram for this new cluster is 0.262, which is now observed to contain three banking sector stocks. Although not shown in the dendrogram, the cluster is identified by the return vector averaged over the three stocks.

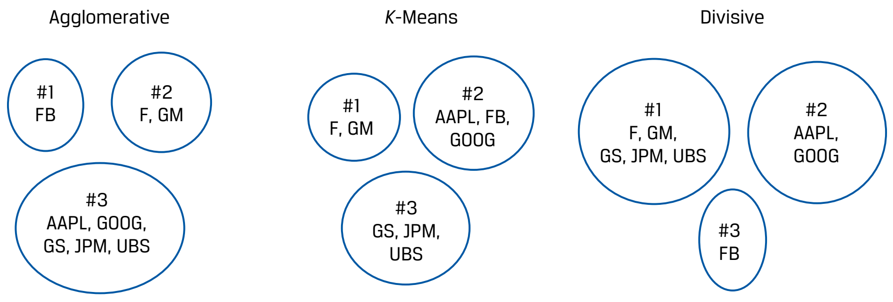
The next closest pair of points, whether stock to stock or stock to cluster, is AAPL and GOOG, with a distance of 0.307, so the algorithm merges these two points into a second cluster (i.e., AAPL_GOOG), with an arch height of 0.307. Next, GM and F are paired into a third cluster (i.e., F_GM), with an arch height of 0.334. Finally, the first two clusters are merged to form a five-stock cluster (i.e., GS_JPM_UBS_AAPL_GOOG), with an arch height of 0.356. Note that this value is determined by taking the average distance between the three banks and AAPL and GOOG: 0.356 = (0.364 + 0.380 + 0.375 + 0.332 + 0.338 + 0.345)/6. The result is three separate clusters: the five-stock cluster, F_GM, and FB by itself. Also, note the horizontal dashed line that cuts the dendrogram into three distinct clusters, with FB as its own cluster.

This agglomerative hierarchical clustering analysis reveals some interesting preliminary results—largely grouping the stocks by their sectors but also uncovering some anomalies. In particular, FB is found to behave quite differently, in terms of return co-movement similarity, from the other technology stocks (AAPL and GOOG). Also, AAPL and GOOG are found to behave more like the bank stocks and less like the auto stocks (F and GM), which appear in their own cluster.

In contrast to agglomerative clustering, the divisive clustering algorithm starts with all stocks assigned to one large cluster and then splits the cluster into sub-clusters recursively, until each stock occupies its own cluster. Determining how to split the first cluster requires searching over all combinations of possible splits, so it is too numerically intensive to cover the details here. However, results of the first two splits for divisive clustering, into three clusters, are shown in Exhibit 27. Results for *k*-means, with *k* = 3, and agglomerative clustering are also presented.

Exhibit 27 Comparison of Results of Different Clustering Algorithms

	Agglomerative	K-means	Divisive
AAPL	3	2	2
F	2	1	1
FB	1	2	3
GM	2	1	1
GOOG	3	2	2
GS	3	3	1
JPM	3	3	1
UBS	3	3	1



Whereas the assignment of the cluster number (1, 2, 3), shown in the upper panel, can be taken as arbitrary across each algorithm, the useful information is in the grouping of like stocks. As seen in the stylized clusters in the lower panel, all three clustering algorithms agree that bank stocks belong in the same cluster. Both hierarchical agglomerative and *k*-means algorithms also agree that auto stocks belong in their own separate cluster. *K*-means clusters the stocks precisely by industry sector, whereas hierarchical agglomerative and divisive clustering identify FB as an outlier and place it in its own cluster. In general, the most agreement is expected between the two hierarchical clustering algorithms, although their results are not guaranteed to match, even when using the same linkage process. *K*-means starts with three clusters ($k = 3$) and iteratively swaps points in and out of these clusters using a partitioning mechanism different from that of hierarchical clustering. Thus, *k*-means results are typically not expected to match those of hierarchical clustering.

In conclusion, based on the analyses of the co-movement similarity of returns among the eight stocks using the agglomerative clustering algorithm and the Investment Committee's requirement that the correlation of returns between the recommended stocks should be low, the analyst's recommendation should be as follows:

- buy FB,
- buy the most attractive of the two auto stocks (F or GM), and
- buy the most attractive of the three bank stocks (GS, JPM, or UBS).

EXAMPLE 7

Hierarchical Agglomerative Clustering

Assume the analyst is given the same set of stocks as previously excluding F and GM (i.e., no auto stocks)—so now, six stocks. Using the information from this mini-case study, answer the following questions:

1. Describe how the inputs to the hierarchical agglomerative clustering algorithm would differ from those in the mini-case study.
2. Describe the three clusters that would now result from running the hierarchical agglomerative clustering algorithm.
3. Explain why these results differ from the previous case, with eight stocks (including the two auto stocks).
4. Describe the analyst's new recommendation to the Investment Committee.

Solution to 1:

The panel data on closing prices and daily log returns would include the same stocks as before but without F and GM—so, AAPL, FB, GOOG, GS, JPM, and UBS. The distance matrix would also appear the same except without F, GM, or any of the pairwise distances between them and the remaining stocks.

Solution to 2:

The three clusters that would now result from running the agglomerative clustering algorithm are GS_JPM_UBS (i.e., one cluster of three bank stocks), AAPL_GOOG (i.e., one cluster of two technology stocks), and FB by itself.

Solution to 3:

The agglomerative clustering algorithm now combines GS and JPM and then UBS, as before, to form a bank cluster. Next, and as previously, the algorithm combines AAPL and GOOG into a cluster. However, without the auto stocks, there is no need to combine AAPL_GOOG with the bank cluster. There are now three distinct clusters, since (as before) the algorithm treats FB as its own cluster, given the high degree of return co-movement dissimilarity between FB and the other clusters (i.e., AAPL_GOOG, and GS_JPM_UBS).

Solution to 4:

The analyst's new recommendation to the Investment Committee would be to buy FB, buy the cheapest of AAPL or GOOG, and buy the most attractive of the three bank stocks (GS, JPM, or UBS).

7 NEURAL NETWORKS, DEEP LEARNING NETS, AND REINFORCEMENT LEARNING

e Describe neural networks, deep learning nets, and reinforcement learning

The artificial intelligence revolution has been driven in large part by advances in neural networks, deep learning algorithms, and reinforcement learning. These sophisticated algorithms can address highly complex machine learning tasks, such as image classification, face recognition, speech recognition, and natural language processing. These complicated tasks are characterized by non-linearities and interactions between large numbers of feature inputs. We now provide an overview of these algorithms and their investment applications.

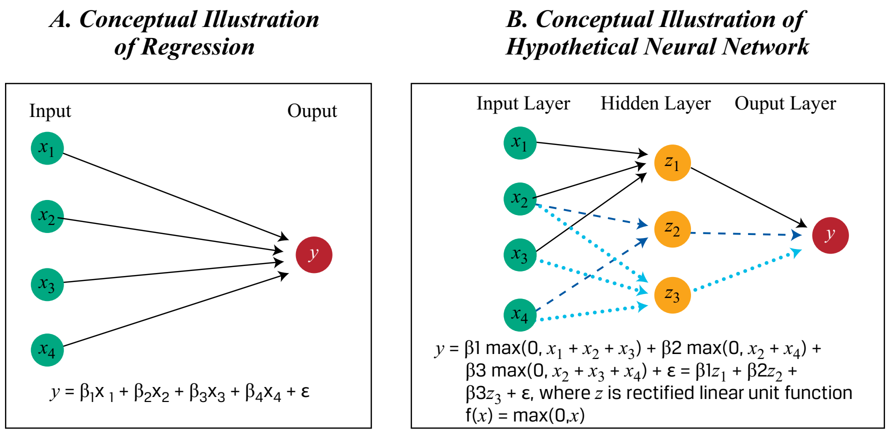
7.1 Neural Networks

Neural networks (also called artificial neural networks, or ANNs) are a highly flexible type of ML algorithm that have been successfully applied to a variety of tasks characterized by non-linearities and complex interactions among features. Neural networks are commonly used for classification and regression in supervised learning but are also important in reinforcement learning, which does not require human-labeled training data.

Exhibit 28 shows the connection between multiple regression and neural networks. Panel A represents a hypothetical regression for data using four inputs, the features x_1 to x_4, and one output—the predicted value of the target variable y. Panel B shows a schematic representation of a basic neural network, which consists of nodes (circles) connected by links (arrows connecting nodes). Neural networks have three types of layers: an input layer (here with a node for each of the four features); hidden layers, where learning occurs in training and inputs are processed on trained nets; and an output layer (here consisting of a single node for the target variable y), which passes information outside the network.

Besides the network structure, another important difference between multiple regression and neural networks is that the nodes in the neural network's hidden layer transform the inputs in a non-linear fashion into new values that are then combined into the target value. For example, consider the popular rectified linear unit (ReLU) function, $f(x) = \max(0, x)$, which takes on a value of zero if there is a negative input and takes on the value of the input if it is positive. In this case, y will be equal to β_1 times z_1, where z_1 is the maximum of $(x_1 + x_2 + x_3)$ or 0, plus β_2 times z_2, the maximum of $(x_2 + x_4)$ or 0, plus β_3 times z_3, the maximum of $(x_2 + x_3 + x_4)$ or 0, plus an error term.

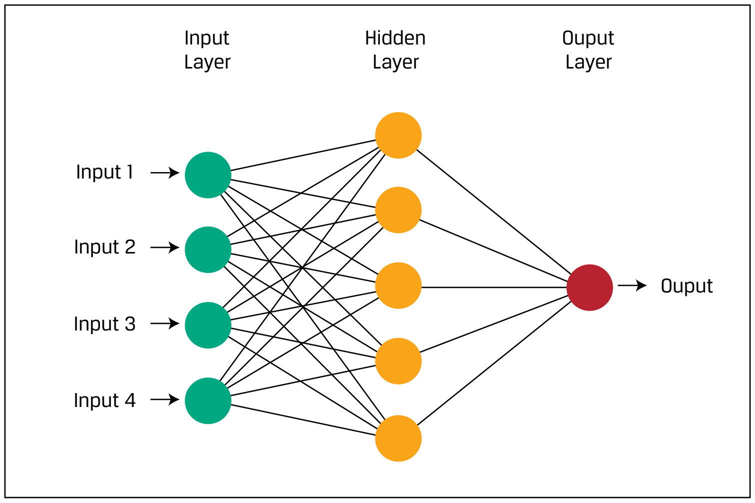
Exhibit 28 Regression and Neural Networks (Regression with Transformed Features)


A. Conceptual Illustration of Regression
Input
Ouput
x_1
x_2
x_3
x_4
y
$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon$
B. Conceptual Illustration of Hypothetical Neural Network
Input Layer
Hidden Layer
Ouput Layer
x_1
x_2
x_3
x_4
z_1
z_2
z_3
y
$y = \beta 1 \max(0, x_1 + x_2 + x_3) + \beta 2 \max(0, x_2 + x_4) + \beta 3 \max(0, x_2 + x_3 + x_4) + \varepsilon = \beta 1 z_1 + \beta 2 z_2 + \beta 3 z_3 + \varepsilon$, where z is rectified linear unit function $f(x) = \max(0,x)$

Note that for neural networks, the feature inputs would be scaled (i.e., standardized) to account for differences in the units of the data. For example, if the inputs were positive numbers, each could be scaled by its maximum value so that their values lie between 0 and 1.

Exhibit 29 shows a more complex neural network, with an input layer consisting of four nodes (i.e., four features), one hidden layer consisting of five hidden nodes, and an output node. These three numbers—4, 5, and 1—for the neural network are hyperparameters that determine the structure of the neural network.

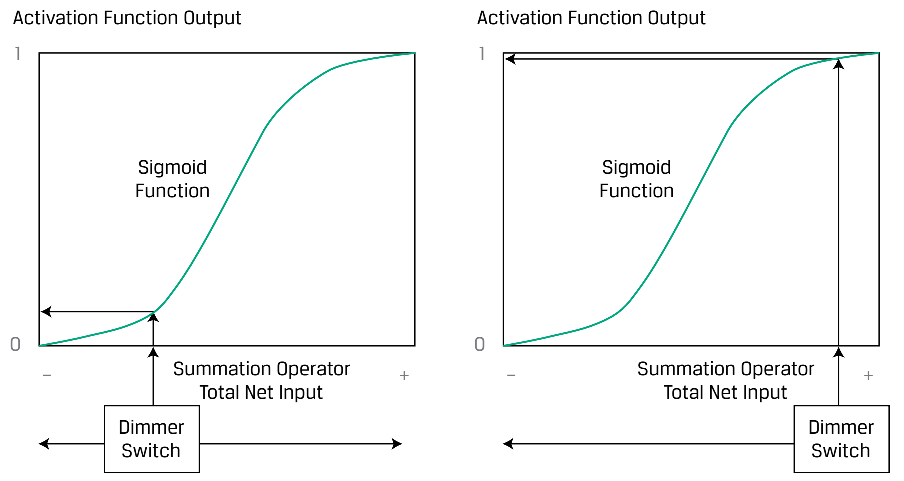
Exhibit 29 A More Complex Neural Network with One Hidden Layer



Now consider any of the nodes to the right of the input layer. These nodes are sometimes called "neurons" because they process information received. Take the topmost hidden node. Four links connect to that node from the inputs, so the node gets four values transmitted by the links. Each link has a weight meant to represent its importance (initially these weights may be assigned randomly). Each node has, conceptually, two functional parts: a summation operator and an activation function. Once the node receives the four input values, the **summation operator** multiplies each value by its respective weight and then sums the weighted values to form the total net input. The total net input is then passed to the **activation function**, which transforms this input into the final output of the node. Informally, the activation function operates like a light dimmer switch that decreases or increases the strength of the input. The activation function, which is chosen by the modeler (i.e., a hyperparameter), is characteristically non-linear, such as an S-shaped (sigmoidal) function (with output range of 0 to 1) or the rectified linear unit function shown in Panel B of Exhibit 28. Non-linearity implies that the rate of change of output differs at different levels of input.

This activation function is shown in Exhibit 30, where in the left graph a negative total net input is transformed via the S-shaped function into an output close to 0. This low output implies the node does not trigger, so there is nothing to pass to the next node. Conversely, in the right graph a positive total net input is transformed into an output close to 1, so the node does trigger. The output of the activation function is then transmitted to the next set of nodes if there is a second hidden layer or, as in this case, to the output layer node as the predicted value. The process of transmission just described (think of forward pointing arrows in Exhibit 29) is referred to as **forward propagation**.

Exhibit 30 Activation Function as "Light Dimmer Switch" at Each Node in a Neural Network


Activation Function Output
1
0
Sigmoid Function
Summation Operator Total Net Input
–
+
Dimmer Switch
Activation Function Output
1
0
Sigmoid Function
Summation Operator Total Net Input
–
+
Dimmer Switch

Starting with an initialized set of random network weights (i.e., the weights assigned to each of the links), training a neural network in a supervised learning context is an iterative process in which predictions are compared to actual values of labeled data and evaluated by a specified performance measure (e.g., mean squared error). Then, the network weights are adjusted to reduce total error of the network. (If the process of adjustment works backward through the layers of the network, this process is called **backward propagation**). Learning takes place through this process of adjustment to the network weights with the aim of reducing total error. Without proliferating notation relating to nodes, the gist of the updating can be expressed informally as

New weight = (Old weight) – (Learning rate) × (Partial derivative of the total error with respect to the old weight),

where partial derivative is a gradient or rate of change of the total error with respect to the change in the old weight and **learning rate** is a hyperparameter that affects the magnitude of adjustments. When learning is completed, all the network weights have assigned values; these are the parameters of the network.

The structure of a network in which all the features are interconnected with non-linear activation functions allows neural networks to uncover and approximate complex non-linear relationships among features. Broadly speaking, when more nodes and more hidden layers are specified, a neural network's ability to handle complexity tends to increase (but so does the risk of overfitting).

Asset pricing is a noisy, stochastic process with potentially unstable relationships that challenge modeling processes, so researchers are asking if machine learning can improve our understanding of how markets work. Research comparing statistical and machine learning methods' abilities to explain and predict equity prices so far indicates that simple neural networks produce models of equity returns at the individual stock and portfolio level that are superior to models built using traditional statistical methods due to their ability to capture dynamic and interacting variables. This suggests that ML-based models, such as neural networks, may simply be better able to cope with the non-linear relationships inherent in security prices. However, the trade-offs in using neural networks are their lack of interpretability (i.e., black box nature) and the large amounts of data and high computation intensity needed to train such models; thus, neural networks may not be a good choice in many investment applications.

7.2 Deep Learning Neural Networks

The previous discussion of neural networks was limited to types of neural networks referred to as "shallow neural networks"—exhibiting just one hidden layer. Neural networks with many hidden layers—at least 2 but potentially more than 20—are known as **deep neural networks** (DNNs). DNNs are the foundation of deep learning and have proven to be successful across a wide range of artificial intelligence applications. Advances in DNNs have driven developments in many complex activities, such as image, pattern, and speech recognition. To state the operation of DNNs succinctly, they take a set of inputs x from a feature set (the input layer), which are then passed to a layer of non-linear mathematical functions (neurons) with weights w_{ij} (for neuron i and input j), each of which usually produces a scaled number in the range (0, 1) or (−1, 1). These numbers are then passed to another layer of functions and into another and so on until the final layer produces a set of probabilities of the observation being in any of the target categories (each represented by a node in the output layer). The DNN assigns the category based on the category with the highest probability. The DNN is trained on large datasets; during training, the weights, w_i, are determined to minimize a specified loss function.

In practice, while the number of nodes in the input and the output layers are typically determined by the characteristics of the features and predicted output, many model hyperparameters still must be decided, particularly the number of hidden layers, the number of nodes per hidden layer, and their connectivity and activation architecture. The objective is to choose them to achieve the best out-of-sample performance, but it is still a challenge with no simple solution. As such, a good starting point is a "reasonable" guess for hyperparameters based on experience and literature. The researcher can then observe the result and adjust the hyperparameters incrementally until the model performance goal is reached. In practice, DNNs require substantial time to train, and systematically varying the hyperparameters may not be feasible. So, for many problems with relatively small datasets, one can start with just two or three hidden layers and a few hundred nodes before tuning the parameters until a model with acceptable predictive power is achieved.

DNNs have been shown to be useful in general for pattern recognition problems (e.g., character and image recognition), credit card fraud detection, vision and control problems in autonomous cars, natural language processing (such as machine translation), and other applications. DNNs have become hugely successful because of a confluence of three developments: (1) the availability of large quantities of machine-readable data to train models, (2) advances in analytical methods for fitting these models, and (3) fast computers, especially new chips in the graphics processing unit (GPU) class, tailored for the type of calculations done on DNNs.

Several financial firms are experimenting with DNNs for trading as well as automating their internal processes. Culkin and Das (2017) described how they trained DNNs to price options, mimicking the Black–Scholes–Merton model. Their research used the same six input parameters for the model as input layer features—spot price, strike, time to maturity, dividend yield, risk-free interest rate, and volatility—with four hidden layers of 100 neurons each and one output layer. The predicted option prices out-of-sample were very close to the actual option prices: A regression of predicted option prices on actual prices had an R^2 of 99.8%.

7.3 Reinforcement Learning

Reinforcement learning (RL) made headlines in 2017 when DeepMind's AlphaGo program beat the reigning world champion at the ancient game of Go. The RL framework involves an agent that is designed to perform actions that will maximize its rewards over time, taking into consideration the constraints of its environment. In the case of AlphaGo, a virtual gamer (the agent) uses his or her console commands (the actions) with the information on the screen (the environment) to maximize his or her score (the reward). Unlike supervised learning, reinforcement learning has neither direct labeled data for each observation nor instantaneous feedback. With RL, the algorithm needs to observe its environment, learn by testing new actions (some of which may not be immediately optimal), and reuse its previous experiences. The learning subsequently occurs through millions of trials and errors. Academics and practitioners are applying RL in a similar way in investment strategies where the agent could be a virtual trader who follows certain trading rules (the actions) in a specific market (the environment) to maximize its profits (its reward). The success of RL in dealing with the complexities of financial markets is still an open question.

EXAMPLE 8

Deep Neural Networks

Glen Mitsui is the chief investment officer for a large Australian state's Public Employees' Pension Fund (PEPF), which currently has assets under management (AUM) of A$20 billion. The fund manages one-quarter of its assets internally, with A$5 billion mostly in domestic government and corporate fixed-income instruments and domestic equities. The remaining three-quarters of AUM, or A$15 billion, is managed by nearly 100 mostly active external asset managers and is invested in a wide range of asset classes, including foreign fixed income and equities, domestic and foreign hedge funds, REITs, commodities, and derivatives.

PEPF has a small staff of four investment professionals tasked with selecting and monitoring these external managers to whom it pays more than A$400 million in fees annually. Performance (compared to appropriate benchmarks) of many of PEPF's external managers has been lagging over the past several years. After studying the situation, Mitsui concludes that style drift may be an important factor in explaining such underperformance, for which PEPF is not happy to pay. Mitsui believes that machine learning may help and consults with Frank Monroe, professor of data analysis at Epsilon University.

Monroe suggests using a deep neural network model that collects and analyzes the real-time trading data of PEPF's external managers and compares them to well-known investment styles (e.g., high dividend, minimum volatility, momentum, growth, value) to detect potential style drift. Mitsui arranges for

Monroe to meet with PEPF's investment committee (IC) to discuss the matter. As a junior data analyst working with Monroe, you must help him satisfy the following requests from the IC:

1 Define a deep neural network.

2 Evaluate Monroe's opinion on the applicability of deep neural networks to Mitsui's problem.

3 Describe the functions of the three groups of layers of a deep neural network.

Solution to 1:

A deep neural network is a neural network (NN) with many hidden layers (at least 2 but often more than 20). NNs and DNNs have been successfully applied to a wide variety of complex tasks characterized by non-linearities and interactions among features, particularly pattern recognition problems.

Solution to 2:

Mitsui wants to detect patterns of potential style drift in the daily trading of nearly 100 external asset managers in many markets. This task will involve the processing of huge amounts of complicated data. Monroe is correct that a DNN is well suited to PEPF's needs.

Solution to 3:

The input layer, the hidden layers, and the output layer constitute the three groups of layers of DNNs. The input layer receives the inputs (i.e., features) and has as many nodes as there are dimensions of the feature set. The hidden layers consist of nodes, each comprising a summation operator and an activation function that are connected by links. These hidden layers are, in effect, where the model is learned. The final layer, the output layer, produces a set of probabilities of an observation being in any of the target style categories (each represented by a node in the output layer). For example, if there are three target style categories, then three nodes in the output layer are activated to produce outputs that sum to one. So, output (Style Category I, 0.7; Style Category II, 0.2; Style Category III, 0.1) would indicate that the model assigns the greatest probability to an observation being in Style Category I and the least probability to Style Category III. The DNN assigns the observation to the style category with the highest probability.

CASE STUDY: DEEP NEURAL NETWORK–BASED EQUITY FACTOR MODEL

The following case study was developed and written by Matthew Dixon, PhD, FRM.

An investment manager wants to select stocks based on their predicted performance using a fundamental equity factor model. She seeks to capture superior performance from stocks with the largest excess return using a non-linear factor model and so chooses a deep neural network to predict the stock returns. The goal of this mini-case study is to demonstrate the application of deep neural networks to fundamental equity factor modeling. We shall focus on using feed-forward (i.e., forward propagation) network regression in place of ordinary least squares linear regression. Since neural networks are prone to over-fitting, we shall use LASSO penalization, the same penalty score–based approach used previously with regression, to mitigate this issue.

Introduction

Cross-sectional fundamental factor models are used extensively by investment managers to capture the effects of company-specific factors on individual securities. A fixed universe of N assets is first chosen, together with a set of K fundamental factors. Each asset's sensitivity (i.e., exposure or loading) to a fundamental factor is represented by beta, B, and the factors are represented by factor returns (f_t). There are two standard approaches to estimating a factor model: (i) adopt time-series regression (TSR) to recover loadings if factors are known or (ii) use cross-sectional regression (CSR) to recover factor returns from known loadings. We shall follow the CSR approach; the factor exposures are used to predict a stock's return (r_t) by estimating the factor returns using multivariate linear regression (where ε_t is the model error at time t):

$$r_t = Bf_t + \varepsilon_t.$$

However, this CSR model is too simplistic to capture non-linear relationships between stock returns and fundamental factors. So, instead we use a deep neural network to learn the non-linear relationships between the betas (B) and asset returns (r_t) at each time t. The goal of deep learning is to find the network weights which minimize the out-of-sample mean squared error (MSE) between the predicted stock returns, $\hat{r}$, and the observed stock returns, r. We shall see that simply increasing the number of neurons in the network will increase predictive performance using the in-sample data but to the detriment of out-of-sample performance; this phenomenon is the bias–variance trade-off. To mitigate this effect, we add a LASSO penalty term to the loss function to automatically shrink the number of non-zero weights in the network. In doing so, we shall see that this leads to better out-of-sample predictive performance.

Note that each weight corresponds to a link between a node in the previous and current layer. Reducing the number of weights generally means that the number of connections—not the number of nodes—is reduced. The exception is when all weights from the neurons in the previous layer are set to zero—in which case the number of nodes in the current layer would be reduced. In the special case when the previous layer is the input layer, the number of features is also reduced.

We shall illustrate the data preparation and the neural network fitting using six fundamental equity factors. This choice of number and type of fundamental factor is arbitrary, and an investment manager may use many more factors in her or his model, often representing industry sectors and sub-sectors using dummy variables.

Data Description

A description of the stock price and fundamental equity factor data used for training and evaluating the neural network is shown in Exhibit 31.

Exhibit 31 Dataset of S&P 500 Stocks and Fundamental Factors

Description:

A subset of S&P 500 Index stocks, historical monthly adjusted closing prices, and corresponding monthly fundamental factor loadings.

Time period: June 2010 to November 2018

Number of periods: 101

Number of stocks (N): 218 stocks

Number of features (K): 6

Features: Fundamental equity factors:

Exhibit 31 (Continued)

1 Current enterprise value (i.e., market values of equity + preferred stock + debt − cash − short-term investments)

2 Current enterprise value to trailing 12-month EBITDA

3 Price-to-sales ratio

4 Price-to-earnings ratio

5 Price-to-book ratio

6 Log of stock's market capitalization (i.e., share price × number of shares outstanding)

Output: Monthly return for each stock over the following month.

We define the universe as the top 250 stocks from the S&P 500, ranked by market capitalization as of June 2010. All stock prices and factor loadings are sourced from Bloomberg. An illustrative extract of the data is given in Exhibit 32. Note that after removing stocks with missing factor loadings, we are left with 218 stocks.

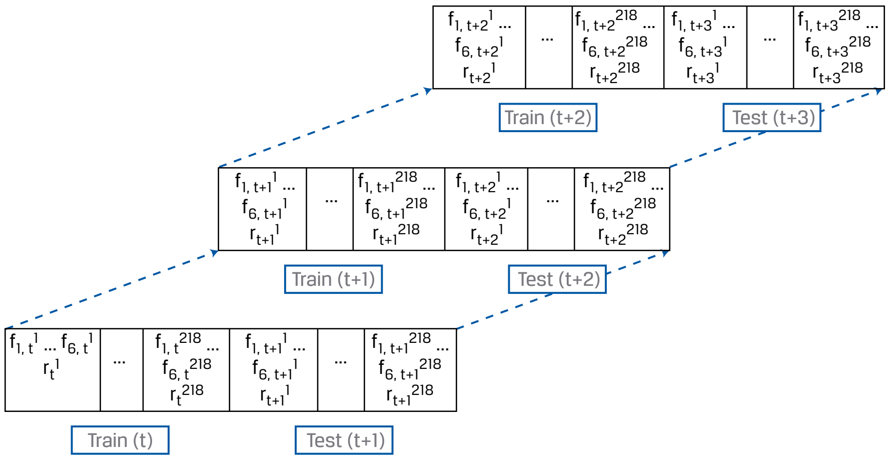
Exhibit 32 Extract of Six Factor Loadings and Return for Three Selected Stocks

TICKER	CURR_EV ($ Mil.)	CURR_EV_ TO_T12M_ EBITDA (X)	PX_ TO_ SALES (X)	PX_ TO_ EARN (X)	PX_ TO_ BOOK (X)	LOG_ CAP ($ Mil.)	RETURN (%)
SWK	10,775.676	30.328	1.138	16.985	1.346	9.082970	−0.132996
STZ	7,433.553	15.653	1.052	10.324	1.480	8.142253	−0.133333
SRE	19,587.124	10.497	1.286	10.597	1.223	9.314892	−0.109589

Experimental Design

The method used to train the deep neural network is time-series cross-validation (i.e., walk-forward optimization), as depicted in Exhibit 33. At each time period, the investment manager fits a new model; each factor (f_1 to f_6) is a feature in the network, and the loadings of the factors for each stock is a feature vector observation (i.e., the set of observations for each stock for each period), leading to $N = 218$ observations of pairs of feature vectors and output (monthly return, r_t) in the training set per period. The network is initially trained at period t, and then it is tested over the next period, $t + 1$, which also has $N = 218$ observations of pairs of feature vectors and output. In the next iteration, the $t + 1$ data become the new training set and the revised model is tested on the $t + 2$ data. The walk-forward optimization of the neural network continues until the last iteration: model training with $t + 99$ data (from Period 100) and testing with $t + 100$ data (from the last period, 101).

Exhibit 33 Time-Series Cross-Validation on Asset Returns (Walk-Forward Optimization)—The First Three Iterations



$f_{1,t}^{1}$ … $f_{6,t}^{1}$ r_{t}^{1}
…
$f_{1,t}^{218}$ … $f_{6,t}^{218}$ r_{t}^{218}
$f_{1,t+1}^{1}$ … $f_{6,t+1}^{1}$ r_{t+1}^{1}
$f_{1,t+1}^{218}$ … $f_{6,t+1}^{218}$ r_{t+1}^{218}
$f_{1,t+2}^{1}$ … $f_{6,t+2}^{1}$ r_{t+2}^{1}
$f_{1,t+2}^{218}$ … $f_{6,t+2}^{218}$ r_{t+2}^{218}
$f_{1,t+3}^{1}$ … $f_{6,t+3}^{1}$ r_{t+3}^{1}
$f_{1,t+3}^{218}$ … $f_{6,t+3}^{218}$ r_{t+3}^{218}
Train (t)
Test (t+1)
Train (t+1)
Test (t+2)
Train (t+2)
Test (t+3)

We use a feed-forward neural network with six input nodes (i.e., neurons), two hidden layers, and one output neuron. There are 50 neurons in each hidden layer to intentionally over-specify the number of parameters needed in the model, meaning bias (variance) is substantially lower (higher) than optimal. LASSO penalization is then used to automatically shrink the parameter set. Additionally, it is important for the number of nodes in each hidden layer not to exceed the number of observations in the training set (50 nodes per layer versus 218 observations). The model training in period *t* involves finding the optimal bias-versus-variance trade-off. Once fitted, we record the in-sample MSE and the out-of-sample MSE in addition to the optimal regularization parameter. This procedure is then repeated sequentially over the horizon of 100 remaining periods, tuning the hyperparameters at each stage using cross-validation. The end result of this procedure is a fitted model, trained monthly on the current cross-sectional data and for which hyperparameters have been tuned at each step.

Results

Exhibit 34 presents the results from model evaluation; it compares the in-sample and out-of-sample MSEs of the deep neural network over all 101 months. Note that the out-of-sample error (dotted line) is typically significantly larger than the in-sample error (solid line). However, as the time periods pass and the model is repeatedly trained and tested, the difference between the out-of-sample and in-sample MSEs narrows dramatically.

Exhibit 34 In-Sample and Out-of-Sample MSE for Each Training and Testing Period

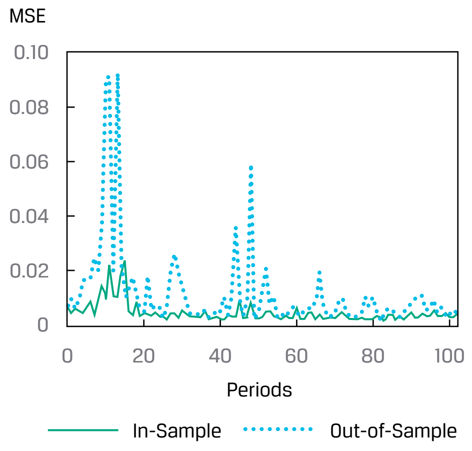
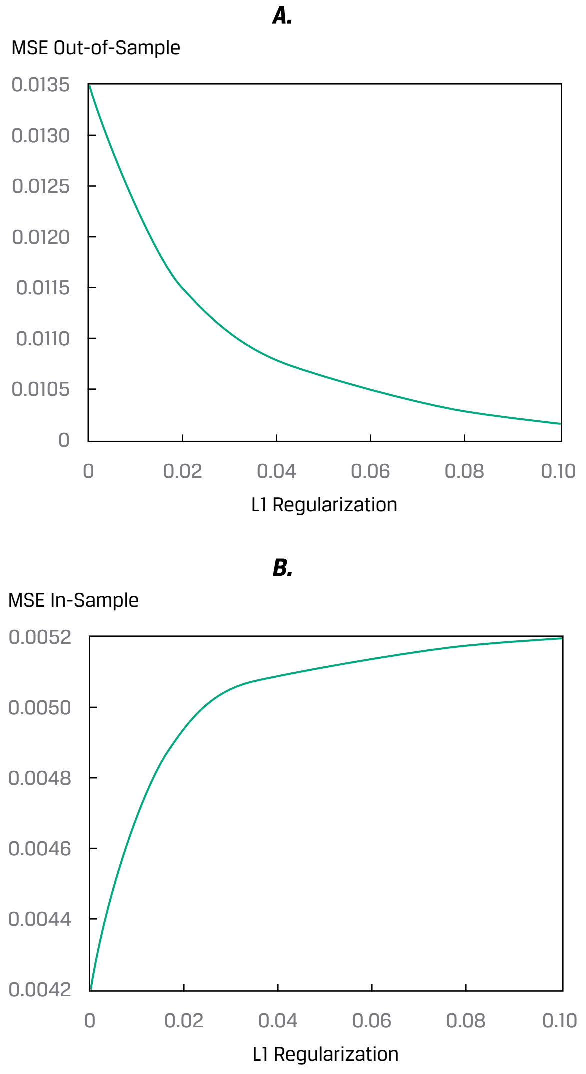


Exhibit 35 shows the effect of LASSO regularization on the in-sample MSE (lower panel, B) and the out-of-sample MSE (upper panel, A) for the first iteration of the time-series cross-validation (training with data from period *t* and testing with data from period *t* +1). The degree of LASSO regularization needed is found by cross-validation using 50 neurons in each hidden layer. Increasing the LASSO regularization, which reduces the number of non-zero weights in the model, introduces more bias and hence increases the in-sample error. Conversely, increasing the LASSO regularization reduces the model's variance and thereby reduces the out-of-sample error. Overall, the amount of LASSO regularization needed is significant, at 0.10; typically the regularization hyperparameter is between 0.001 and 1.0. Also, the out-of-sample and in-sample MSEs have not yet converged. There is still a substantial gap, of roughly 0.0051 (= 0.01025 – 0.0052), and the slope of the curves in each plot suggests the optimal value of the regularization hyperparameter is significantly more than 0.10. Note that the value of the regularization hyperparameter is not interpretable and does not correspond to the number of weights eliminated. Suffice it to say, the larger the value of the regularization hyperparameter, the more the loss is being penalized.

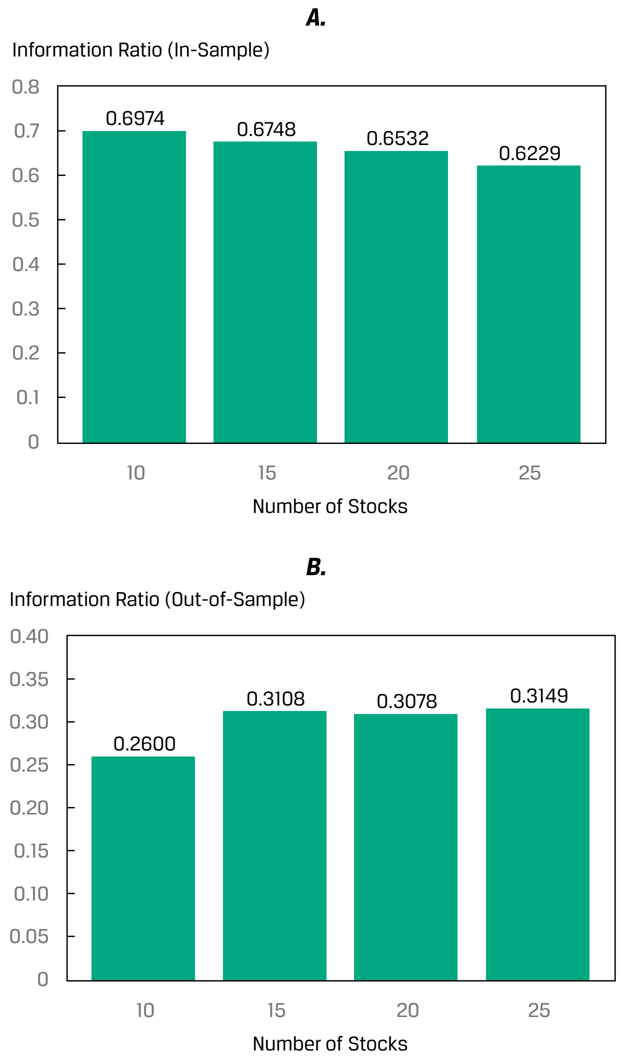
Exhibit 35 LASSO Regularization for Optimizing Bias–Variance Trade-Off (First Iteration)



It is important to recognize that although the out-of-sample MSE of this deep learning neural network is key to characterizing its predictive performance, it does not necessarily follow that a stock selection strategy based on the neural network will be successful. This is because the neural network predicts the next month's expected (i.e., mean) asset returns and not the full distribution of returns. Hence a simple stock selection strategy—measured by information ratios (recall the information ratio, or IR, is alpha divided by nonsystematic risk, so it measures the abnormal return per unit of risk for a well-diversified portfolio) of the portfolio returns—that selects stocks ranked by predicted returns will not necessarily lead to positive information ratios.

Exhibit 36 presents the information ratios found by back-testing a simple stock selection strategy that picks the top performing stocks determined by the neural network's forecasted returns realized in month *t* +1 using features observed in month *t*. Note these IRs do not account for transaction costs, interest rates, or any other fees. The upper panel (A) shows the best-case scenario; the neural network in-sample prediction is used to select the *n* (where *n* is 10, 15, 20, or 25) top performing stocks. The IRs are shown for each of the different-sized portfolios; they range from 0.697 to 0.623. Note that as a rule of thumb, IRs in the range of 0.40–0.60 are considered quite good. The lower panel (B) shows the IRs from back-test results for the same strategy applied to the out-of-sample data. The out-of-sample IRs range from 0.260 to 0.315 and so are substantially smaller than in-sample IRs.

Exhibit 36 Information Ratios from Back-Testing a Stock Selection Strategy Using Top Performers from the Neural Network



Importantly, the out-of-sample performance provides the most realistic assessment of the likely future investment performance from applying this deep learning neural network to stock selection. It is a baseline for further model refinements, including adding more fundamental and macroeconomic factors. With such refinements, it can be expected that the out-of-sample IRs should improve substantially.

EXAMPLE 9

Deep Learning–Based Fundamental Factor Model

A research analyst, Jane Hinton, has been tasked with further developing the deep learning–based fundamental factor model. She decides to refine the model by adding four more fundamental factors (such as debt leverage and R&D intensity) given by firm characteristics and by including dummy variables for 11 industrial sectors. Moreover, she additionally expands the universe of stocks to 420 from 218 by using a supplementary data source.

1 Describe how Jane would modify the inputs of the neural network architecture for this new dataset.
2 Describe the size of the new training and test datasets.
3 Describe any additional changes to the architecture and hyperparameters of the neural network that Jane would likely need to make to ensure good performance of the network.
4 Explain how Jane should evaluate whether the new model leads to improved portfolio performance.

Solution to 1:

Jane adds four more fundamental factors and 11 dummy variables, to represent each industrial sector, for a total of 21 (= 4 + 11 + 6) features. Therefore, the refined neural network will have 21 input neurons. The output layer will remain the same. Note that concerns of collinearity of the features through the dummy variables or high correlation, which are problematic for linear regression, are not an issue for a deep learning–based model.

Solution to 2:

There are now data on 420 stocks, for each of the 101 time periods, consisting of factor loadings for the 21 features and the monthly return for each stock. Per the time-series cross-validation method, the test dataset in the current iteration will become the training dataset in the next iteration.

Solution to 3:

Jane should find the new optimal LASSO regularization hyperparameter using time-series cross-validation. Alternatively, she may find the optimal bias–variance trade-off by first increasing the number of neurons in the hidden layers and then performing the cross-validation.

Solution to 4:

Once Jane has found the optimal LASSO hyperparameter and network architecture, she will use the model to forecast the out-of-sample monthly asset returns (i.e., the model forecasts from factor loadings which are not in the training set). She will then rank and select the top predicted performers and finally measure the realized monthly portfolio return. She will then repeat the experiment by moving forward one month in the dataset and repeating the out-of-sample forecast of the asset returns, until she has generated forecasts for all time periods. Finally, Jane will calculate the information ratios from the mean and standard deviation of the monthly portfolio excess returns.

EXAMPLE 10

Summing Up the Major Types of Machine Learning

1 As used in supervised machine learning, classification problems involve the following *except*:

- **A** binary target variables.
- **B** continuous target variables.
- **C** categorical target variables.

2 Which of the following *best* describes penalized regression? Penalized regression:

- **A** is unrelated to multiple linear regression.
- **B** involves a penalty term that is added to the predicted target variable.
- **C** is a category of general linear models used when the number of features and overfitting are concerns.

3 CART is *best* described as:

- **A** an unsupervised ML algorithm.
- **B** a clustering algorithm based on decision trees.
- **C** a supervised ML algorithm that accounts for non-linear relationships among the features.

4 A neural network is *best* described as a technique for machine learning that is:

- **A** exactly modeled on the human nervous system.
- **B** based on layers of nodes connected by links when the relationships among the features are usually non-linear.
- **C** based on a tree structure of nodes when the relationships among the features are linear.

5 Hierarchical clustering is *best* described as a technique in which:

- **A** the grouping of observations is unsupervised.
- **B** features are grouped into a pre-specified number, *k*, of clusters.
- **C** observations are classified according to predetermined labels.

6 Dimension reduction techniques are *best* described as a means to reduce a set of features to a manageable size:

- **A** without regard for the variation in the data.
- **B** while increasing the variation in the data.
- **C** while retaining as much of the variation in the data as possible.

Solution to 1:

B is correct. A and C are incorrect because when the target variable is binary or categorical (not continuous), the problem is a classification problem.

Solution to 2:

C is correct. A is incorrect because penalized regression is related to multiple linear regression. B is incorrect because penalized regression involves adding a penalty term to the sum of the squared regression residuals.

Solution to 3:

C is correct. A is incorrect because CART is a supervised ML algorithm. B is incorrect because CART is a classification and regression algorithm, not a clustering algorithm.

Solution to 4:

B is correct. A is incorrect because neural networks are not exactly modeled on the human nervous system. C is incorrect because neural networks are not based on a tree structure of nodes when the relationships among the features are linear.

Solution to 5:

A is correct. B is incorrect because it refers to *k*-means clustering. C is incorrect because it refers to classification, which involves supervised learning.

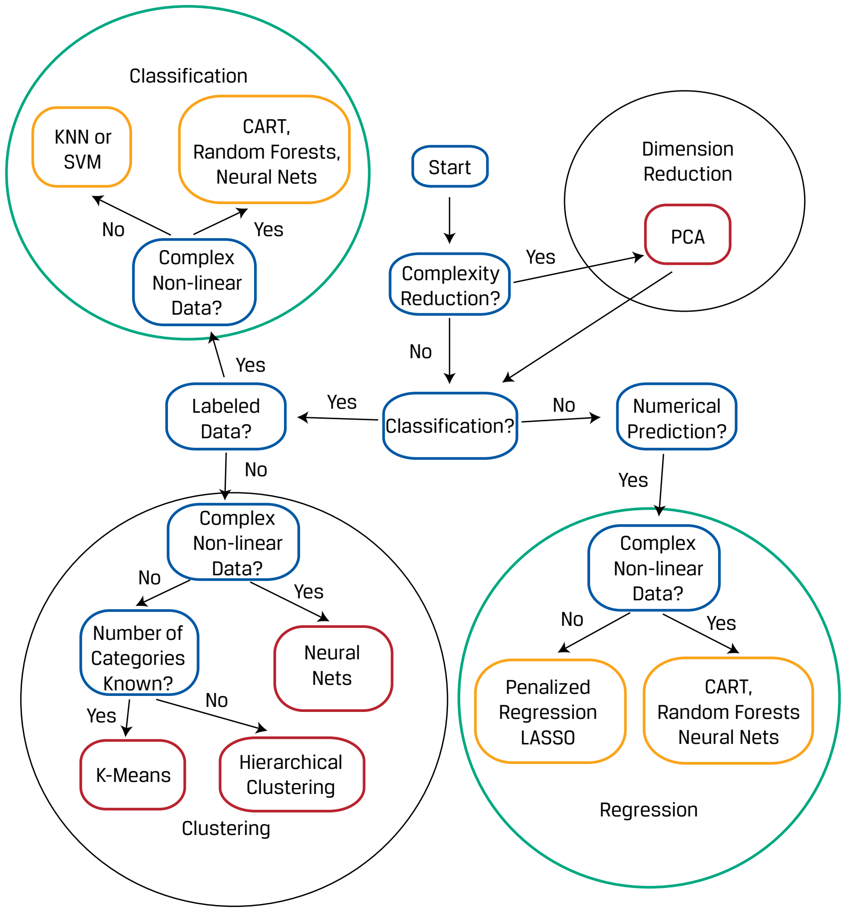
Solution to 6:

C is correct because dimension reduction techniques, such as PCA, are aimed at reducing the feature set to a manageable size while retaining as much of the variation in the data as possible.

8 CHOOSING AN APPROPRIATE ML ALGORITHM

Exhibit 37 presents a simplified decision flowchart for choosing among the machine learning algorithms which have been discussed. The dark-shaded ovals contain the supervised ML algorithms, the light-shaded ovals contain the unsupervised ML algorithms, and the key questions to consider are shown in the unshaded rounded rectangles.

Exhibit 37 Stylized Decision Flowchart for Choosing ML Algorithms



First, start by asking, Are the data complex, having many features that are highly correlated? If yes, then dimension reduction using principal components analysis is appropriate.

Next, is the problem one of classification or numerical prediction? If numerical prediction, then depending on whether the data have non-linear characteristics, the choice of ML algorithms is from a set of regression algorithms—either penalized regression/LASSO for linear data or CART, random forest, or neural networks for non-linear data.

If the problem is one of classification, then depending on whether the data are labeled, the choice is either from a set of classification algorithms using labeled data or from a set of clustering algorithms using unlabeled data.

If the data are labeled, then depending on whether the data have non-linear characteristics, the choice of classification algorithm would be *K*-nearest neighbor and support vector machine for linear data or CART, random forest, or neural networks (or deep neural networks) for non-linear data.

Finally, if the data are unlabeled, the choice of clustering algorithm depends on whether the data have non-linear characteristics. The choice of clustering algorithm would be neural networks (or deep neural networks) for non-linear data or for linear data, *K*-means with a known number of categories and hierarchical clustering with an unknown number of categories.

SUMMARY

Machine learning methods are gaining usage at many stages in the investment management value chain. Among the major points made are the following:

- Machine learning aims at extracting knowledge from large amounts of data by learning from known examples to determine an underlying structure in the data. The emphasis is on generating structure or predictions without human intervention. An elementary way to think of ML algorithms is to "find the pattern, apply the pattern."
- Supervised learning depends on having labeled training data as well as matched sets of observed inputs (*X*'s, or features) and the associated output (*Y*, or target). Supervised learning can be divided into two categories: regression and classification. If the target variable to be predicted is continuous, then the task is one of regression. If the target variable is categorical or ordinal (e.g., determining a firm's rating), then it is a classification problem.
- With unsupervised learning, algorithms are trained with no labeled data, so they must infer relations between features, summarize them, or present underlying structure in their distributions that has not been explicitly provided. Two important types of problems well suited to unsupervised ML are dimension reduction and clustering.
- In deep learning, sophisticated algorithms address complex tasks (e.g., image classification, natural language processing). Deep learning is based on neural networks, highly flexible ML algorithms for solving a variety of supervised and unsupervised tasks characterized by large datasets, non-linearities, and interactions among features. In reinforcement learning, a computer learns from interacting with itself or data generated by the same algorithm.

- Generalization describes the degree to which an ML model retains its explanatory power when predicting out-of-sample. Overfitting, a primary reason for lack of generalization, is the tendency of ML algorithms to tailor models to the training data at the expense of generalization to new data points.
- Bias error is the degree to which a model fits the training data. Variance error describes how much a model's results change in response to new data from validation and test samples. Base error is due to randomness in the data. Out-of-sample error equals bias error plus variance error plus base error.
- *K*-fold cross-validation is a technique for mitigating the holdout sample problem (excessive reduction of the training set size). The data (excluding test sample and fresh data) are shuffled randomly and then divided into *k* equal sub-samples, with $k - 1$ samples used as training samples and one sample, the *k*th, used as a validation sample.
- Regularization describes methods that reduce statistical variability in high-dimensional data estimation or prediction problems via reducing model complexity.
- LASSO (least absolute shrinkage and selection operator) is a popular type of penalized regression where the penalty term involves summing the absolute values of the regression coefficients. The greater the number of included features, the larger the penalty. So, a feature must make a sufficient contribution to model fit to offset the penalty from including it.
- Support vector machine (SVM) is a classifier that aims to seek the optimal hyperplane—the one that separates the two sets of data points by the maximum margin (and thus is typically used for classification).
- *K*-nearest neighbor (KNN) is a supervised learning technique most often used for classification. The idea is to classify a new observation by finding similarities ("nearness") between it and its *k*-nearest neighbors in the existing dataset.
- Classification and regression tree (CART) can be applied to predict either a categorical target variable, producing a classification tree, or a continuous target variable, producing a regression tree.
- A binary CART is a combination of an initial root node, decision nodes, and terminal nodes. The root node and each decision node represent a single feature (*f*) and a cutoff value (*c*) for that feature. The CART algorithm iteratively partitions the data into sub-groups until terminal nodes are formed that contain the predicted label.
- Ensemble learning is a technique of combining the predictions from a collection of models. It typically produces more accurate and more stable predictions than any single model.
- A random forest classifier is a collection of many different decision trees generated by a bagging method or by randomly reducing the number of features available during training.
- Principal components analysis (PCA) is an unsupervised ML algorithm that reduces highly correlated features into fewer uncorrelated composite variables by transforming the feature covariance matrix. PCA produces eigenvectors that define the principal components (i.e., the new uncorrelated composite variables) and eigenvalues, which give the proportion of total variance in the initial data that is explained by each eigenvector and its associated principal component.

- *K*-means is an unsupervised ML algorithm that partitions observations into a fixed number (k) of non-overlapping clusters. Each cluster is characterized by its centroid, and each observation belongs to the cluster with the centroid to which that observation is closest.
- Hierarchical clustering is an unsupervised iterative algorithm that is used to build a hierarchy of clusters. Two main strategies are used to define the intermediary clusters (i.e., those clusters between the initial dataset and the final set of clustered data).
- Agglomerative (bottom-up) hierarchical clustering begins with each observation being its own cluster. Then, the algorithm finds the two closest clusters, defined by some measure of distance, and combines them into a new, larger cluster. This process is repeated until all observations are clumped into a single cluster.
- Divisive (top-down) hierarchical clustering starts with all observations belonging to a single cluster. The observations are then divided into two clusters based on some measure of distance. The algorithm then progressively partitions the intermediate clusters into smaller clusters until each cluster contains only one observation.
- Neural networks consist of nodes connected by links. They have three types of layers: an input layer, hidden layers, and an output layer. Learning takes place in the hidden layer nodes, each of which consists of a summation operator and an activation function. Neural networks have been successfully applied to a variety of investment tasks characterized by non-linearities and complex interactions among variables.
- Neural networks with many hidden layers (at least 2 but often more than 20) are known as deep neural networks (DNNs) and are the backbone of the artificial intelligence revolution.
- Reinforcement learning (RL) involves an agent that should perform actions that will maximize its rewards over time, taking into consideration the constraints of its environment.

REFERENCES

Bacham, Dinesh, and Janet Zhao. 2017. “Machine Learning: Challenges, Lessons, and Opportunities in Credit Risk Modeling.” *Moody’s Analytics Risk Perspectives* 9:28–35.

Culkin, Robert, and Sanjiv R. Das. 2017. “Machine Learning in Finance: The Case of Deep Learning for Option Pricing.” *Journal of Investment Management* 15 (4): 92–100.

PRACTICE PROBLEMS

The following information relates to Questions 1–10

Alef Associates manages a long-only fund specializing in global smallcap equities. Since its founding a decade ago, Alef maintains a portfolio of 100 stocks (out of an eligible universe of about 10,000 stocks). Some of these holdings are the result of screening the universe for attractive stocks based on several ratios that use readily available market and accounting data; others are the result of investment ideas generated by Alef's professional staff of five securities analysts and two portfolio managers.

Although Alef's investment performance has been good, its Chief Investment Officer, Paul Moresanu, is contemplating a change in the investment process aimed at achieving even better returns. After attending multiple workshops and being approached by data vendors, Moresanu feels that data science should play a role in the way Alef selects its investments. He has also noticed that much of Alef's past outperformance is due to stocks that became takeover targets. After some research and reflection, Moresanu writes the following email to the Alef's CEO.

Subject: Investment Process Reorganization

I have been thinking about modernizing the way we select stock investments. Given that our past success has put Alef Associates in an excellent financial position, now seems to be a good time to invest in our future. What I propose is that we continue managing a portfolio of 100 global small-cap stocks but restructure our process to benefit from machine learning (ML). Importantly, the new process will still allow a role for human insight, for example, in providing domain knowledge. In addition, I think we should make a special effort to identify companies that are likely to be acquired. Specifically, I suggest following the four steps which would be repeated every quarter.

Step 1 We apply ML techniques to a model including fundamental and technical variables (features) to predict next quarter's return for each of the 100 stocks currently in our portfolio. Then, the 20 stocks with the lowest estimated return are identified for replacement.

Step 2 We utilize ML techniques to divide our investable universe of about 10,000 stocks into 20 different groups, based on a wide variety of the most relevant financial and non-financial characteristics. The idea is to prevent unintended portfolio concentration by selecting stocks from each of these distinct groups.

Step 3 For each of the 20 different groups, we use labeled data to train a model that will predict the five stocks (in any given group) that are most likely to become acquisition targets in the next one year.

(Continued)

Step 4 Our five experienced securities analysts are each assigned four of the groups, and then each analyst selects their one best stock pick from each of their assigned groups. These 20 "high-conviction" stocks will be added to our portfolio (in replacement of the 20 relatively underperforming stocks to be sold in Step 1).

A couple of additional comments related to the above:

Comment 1 The ML algorithms will require large amounts of data. We would first need to explore using free or inexpensive historical datasets and then evaluate their usefulness for the ML-based stock selection processes before deciding on using data that requires subscription.

Comment 2 As time passes, we expect to find additional ways to apply ML techniques to refine Alef's investment processes.

What do you think?
Paul Moresanu

---

1 The machine learning techniques appropriate for executing Step 1 are *most* likely to be based on:

A regression

B classification

C clustering

2 Assuming regularization is utilized in the machine learning technique used for executing Step 1, which of the following ML models would be *least* appropriate:

A Regression tree with pruning.

B LASSO with lambda (λ) equal to 0.

C LASSO with lambda (λ) between 0.5 and 1.

3 Which of the following machine learning techniques is *most* appropriate for executing Step 2:

A K-Means Clustering

B Principal Components Analysis (PCA)

C Classification and Regression Trees (CART)

4 The hyperparameter in the ML model to be used for accomplishing Step 2 is?

A 100, the number of small-cap stocks in Alef's portfolio.

B 10,000, the eligible universe of small-cap stocks in which Alef can potentially invest.

C 20, the number of different groups (i.e. clusters) into which the eligible universe of small-cap stocks will be divided.

5 The target variable for the labelled training data to be used in Step 3 is *most* likely which one of the following?

A A continuous target variable.

B A categorical target variable.

C An ordinal target variable.

6 Comparing two ML models that could be used to accomplish Step 3, which statement(s) *best* describe(s) the advantages of using Classification and Regression Trees (CART) instead of K-Nearest Neighbor (KNN)?

Statement I	For CART there is no requirement to specify an initial hyperparameter (like K).
Statement II	For CART there is no requirement to specify a similarity (or distance) measure.
Statement III	For CART the output provides a visual explanation for the prediction.

A Statement I only.

B Statement III only.

C Statements I, II and III.

7 Assuming a Classification and Regression Tree (CART) model is used to accomplish Step 3, which of the following is *most* likely to result in model overfitting?

A Using the k-fold cross validation method

B Including an overfitting penalty (i.e., regularization term).

C Using a fitting curve to select a model with low bias error and high variance error.

8 Assuming a Classification and Regression Tree (CART) model is initially used to accomplish Step 3, as a further step which of the following techniques is most likely to result in more accurate predictions?

A Discarding CART and using the predictions of a Support Vector Machine (SVM) model instead.

B Discarding CART and using the predictions of a K-Nearest Neighbor (KNN) model instead.

C Combining the predictions of the CART model with the predictions of other models – such as logistic regression, SVM, and KNN – via ensemble learning.

9 Regarding Comment #2, Moresanu has been thinking about the applications of neural networks (NNs) and deep learning (DL) to investment management. Which statement(s) *best* describe(s) the tasks for which NNs and DL are well-suited?

Statement I	NNs and DL are well-suited for image and speech recognition, and natural language processing.
Statement II	NNs and DL are well-suited for developing single variable ordinary least squares regression models.
Statement III	NNs and DL are well-suited for modelling non-linearities and complex interactions among many features.

A Statement II only.

B Statements I and III.

C Statements I, II and III.

10 Regarding neural networks (NNs) that Alef might potentially implement, which of the following statements is *least* accurate?

A NNs must have at least 10 hidden layers to be considered deep learning nets.

B The activation function in a node operates like a light dimmer switch since it decreases or increases the strength of the total net input.

C The summation operator receives input values, multiplies each by a weight, sums up the weighted values into the total net input, and passes it to the activation function.

SOLUTIONS

1 A is correct. The target variable (quarterly return) is continuous, hence this calls for a supervised machine learning based regression model.

B is incorrect, since classification uses categorical or ordinal target variables, while in Step 1 the target variable (quarterly return) is continuous.

C is incorrect, since clustering involves unsupervised machine learning so does not have a target variable.

2 B is correct. It is least appropriate because with LASSO, when $\lambda = 0$ the penalty (i.e., regularization) term reduces to zero, so there is no regularization and the regression is equivalent to an ordinary least squares (OLS) regression.

A is incorrect. With Classification and Regression Trees (CART), one way that regularization can be implemented is via pruning which will reduce the size of the regression tree—sections that provide little explanatory power are pruned (i.e., removed).

C is incorrect. With LASSO, when λ is between 0.5 and 1 the relatively large penalty (i.e., regularization) term requires that a feature makes a sufficient contribution to model fit to offset the penalty from including it in the model.

3 A is correct. K-Means clustering is an unsupervised machine learning algorithm which repeatedly partitions observations into a fixed number, *k*, of non-overlapping clusters (i.e., groups).

B is incorrect. Principal Components Analysis is a long-established statistical method for dimension reduction, not clustering. PCA aims to summarize or reduce highly correlated features of data into a few main, uncorrelated composite variables.

C is incorrect. CART is a supervised machine learning technique that is most commonly applied to binary classification or regression.

4 C is correct. Here, 20 is a hyperparameter (in the K-Means algorithm), which is a parameter whose value must be set by the researcher before learning begins.

A is incorrect, because it is not a hyperparameter. It is just the size (number of stocks) of Alef's portfolio.

B is incorrect, because it is not a hyperparameter. It is just the size (number of stocks) of Alef's eligible universe.

5 B is correct. To predict which stocks are likely to become acquisition targets, the ML model would need to be trained on categorical labelled data having the following two categories: "0" for "not acquisition target", and "1" for "acquisition target".

A is incorrect, because the target variable is categorical, not continuous.

C is incorrect, because the target variable is categorical, not ordinal (i.e., 1st, 2nd, 3rd, etc.).

6 C is correct. The advantages of using CART over KNN to classify companies into two categories ("not acquisition target" and "acquisition target"), include all of the following: For CART there are no requirements to specify an initial hyperparameter (like K) or a similarity (or distance) measure as with KNN, and CART provides a visual explanation for the prediction (i.e., the feature variables and their cut-off values at each node).

A is incorrect, because CART provides all of the advantages indicated in Statements I, II and III.

B is incorrect, because CART provides all of the advantages indicated in Statements I, II and III.

7 C is correct. A fitting curve shows the trade-off between bias error and variance error for various potential models. A model with low bias error and high variance error is, by definition, overfitted.

A is incorrect, because there are two common methods to reduce overfitting, one of which is proper data sampling and cross-validation. K-fold cross validation is such a method for estimating out-of-sample error directly by determining the error in validation samples.

B is incorrect, because there are two common methods to reduce overfitting, one of which is preventing the algorithm from getting too complex during selection and training, which requires estimating an overfitting penalty.

8 C is correct. Ensemble learning is the technique of combining the predictions from a collection of models, and it typically produces more accurate and more stable predictions than the best single model.

A is incorrect, because a single model will have a certain error rate and will make noisy predictions. By taking the average result of many predictions from many models (i.e., ensemble learning) one can expect to achieve a reduction in noise as the average result converges towards a more accurate prediction.

B is incorrect, because a single model will have a certain error rate and will make noisy predictions. By taking the average result of many predictions from many models (i.e., ensemble learning) one can expect to achieve a reduction in noise as the average result converges towards a more accurate prediction.

9 B is correct. NNs and DL are well-suited for addressing highly complex machine learning tasks, such as image classification, face recognition, speech recognition and natural language processing. These complicated tasks are characterized by non-linearities and complex interactions between large numbers of feature inputs.

A is incorrect, because NNs and DL are well-suited for addressing highly complex machine learning tasks, not simple single variable OLS regression models.

C is incorrect, because NNs and DL are well-suited for addressing highly complex machine learning tasks, not simple single variable OLS regression models.

10 A is correct. It is the least accurate answer because neural networks with many hidden layers—at least 3, but often more than 20 hidden layers—are known as deep learning nets.

B is incorrect, because the node's activation function operates like a light dimmer switch which decreases or increases the strength of the (total net) input.

C is incorrect, because the node's summation operator multiplies each (input) value by a weight and sums up the weighted values to form the total net input. The total net input is then passed to the activation function.