# Overview of Machine Learning Part 1
## Fundamentals and Classic Approaches

Farhad Maleki, PhD[a], Katie Ovens, PhD[b], Keyhan Najafian, MSc[a], Behzad Forghani, MEng[c,d], Caroline Reinhold, MD, MSc[a,c], Reza Forghani, MD, PhD[a,c,d,e,f,g,*]

**KEYWORDS**

• Machine learning • Supervised learning • Unsupervised learning • Classification • Regression • Dimensionality reduction • Visualization • Clustering

**KEY POINTS**

- Small sample sizes and missing values are two common characteristics of medical datasets that make developing predictive models challenging.
- Classic machine learning methods can be used to build predictive models using small datasets.
- Developing a machine learning application requires following a workflow including data acquisition, exploratory data analysis, data cleaning, dimensionality reduction, model building and validation, and model evaluation.
- Understanding model complexity, variance, and bias is required for the successful application of machine learning.
- Following best practices for model evaluation is necessary for developing generalizable models.

## INTRODUCTION

As health data and computer power become increasingly available, the main challenge is to gain actionable insight from these data. Machine learning (ML) methods have proved to be a viable approach for mining valuable insight from data across various fields.[1–4] This article provides an overview of the classic supervised and unsupervised ML methods as well as fundamental concepts required for understanding how to develop generalizable and high-performance ML applications. It also describes the important steps for developing a ML model and how decisions made in these steps affect model performance and ability to generalize. The content of this article complements other accompanying articles in this issue, especially the ones focused on deep neural networks and ML model evaluation.

Funding: R. Forghani is a clinical research scholar (chercheur-boursier clinicien) supported by the Fonds de recherche en santé du Québec (FRQS) and has an operating grant jointly funded by the FRQS and the Fondation de l'Association des radiologistes du Québec (FARQ).

[a] Augmented Intelligence & Precision Health Laboratory (AIPHL), Department of Radiology and Research Institute of the McGill University Health Centre, 5252 Boulevard de Maisonneuve Ouest, Montreal, Quebec H4A 3S5, Canada; [b] Department of Computer Science, University of Saskatchewan, 176 Thorvaldson Bldg, 110 Science Place, Saskatoon S7N 5C9, Canada; [c] Department of Radiology, McGill University, 1650 Cedar Avenue, Montreal, Quebec H3G1A4, Canada; [d] Gerald Bronfman Department of Oncology, McGill University, Suite 720, 5100, Maisonneuve Boulevard West, Montreal, Quebec H4A3T2, Canada; [e] Segal Cancer Centre, Lady Davis Institute for Medical Research, Jewish General Hospital, 3755 Cote Ste-Catherine Road, Montreal, Quebec H3T 1E2, Canada; [f] Department of Otolaryngology - Head and Neck Surgery, Royal Victoria Hospital, McGill University Health Centre, 1001 boul. Decarie Boulevard, Montreal, Quebec H3A 3J1, Canada; [g] 4intelligent Inc., Cote St-Luc, Quebec H3X 4A6, Canada
* Corresponding author. Room C02.5821, 1001 Decarie Blvd, Montreal, Quebec H4A 3J1, Canada.
*E-mail address:* reza.forghani@mcgill.ca

Neuroimag Clin N Am 30 (2020) e17–e32
https://doi.org/10.1016/j.nic.2020.08.007

### *Overview of Different Model Learning Approaches*

From 1 perspective, ML approaches can be categorized as supervised learning, unsupervised learning, and reinforcement learning (RL). In supervised learning, models are trained using data where a label is provided for each observation in the dataset. This label could be a categorical variable such as benign versus malignant tumor status, or a continuous variable such as gene expression fold change. The phrase supervised learning comes from the fact that these methods need supervision in the form of labeled examples. In supervised learning, the goal is to use several input features to predict the value of 1 or several outcomes.[5] Although supervised learning can be credited for most successful uses of ML, it needs large volumes of labeled data,[6] which has been one of the major challenges in developing such models in contexts where building large-scale labeled datasets is impractical. Besides its tedious and labor-intensive nature, manual labeling requires expertise that is rare and expensive in domains such as health care. Therefore, building large-scale datasets may be impractical, impeding the use of the full potential of supervised learning. Unsupervised methods can be used when these challenges arise.

The goal of unsupervised learning is to find associations and patterns among input data points.[5] Unsupervised learning is also used to find a dense representation for high-dimensional data.[6] Unlike supervised learning methods, these approaches do not require preparing labeled data as input, which saves on time and labor. Clustering,[7] association mining,[8] and dimensionality reduction[9] are among the most common uses of unsupervised learning. Using these approaches, previously unknown patterns can be found among data points. Although domain knowledge can easily be embedded for supervised learning approaches (as the labeled data), incorporating such knowledge cannot easily be embedded into the unsupervised learning methods. These models also cannot be directly used for predictive modeling because there is no outcome variable assigned to a single input data point.

Semisupervised learning is a middle ground between supervised learning and unsupervised learning in which a small portion of data points used for developing an ML model are labeled, and the rest are unlabeled.[10,11] The main objective of semisupervised learning is to improve the model performance in scenarios where the number of labeled samples is small but there are a large number of unlabeled samples available. These unlabeled samples can be used for improving the model performance.

RL is another category of ML. RL focuses on developing software agents capable of increasing their cumulative reward resulting from taking actions in an environment.[12,13] **Fig. 1** provides a schematic view of an RL system. In RL systems, a software agent operating in an environment gets feedback resulting from its previous actions. Often, the provided feedback is for a sequence of actions leading to a final outcome, such as winning in a game of chess or Go. Examples of the environments that an agent operates in could be the game board in the game of chess or Go or a simulation environment for training self-driving cars.[14] RL methods have outperformed human champions in playing games such as chess and Go. For these applications, providing a large number of trial-and-error experiments is straightforward and computationally efficient, resulting in the superior performance of RL methods.[12,13] Unlike supervised, unsupervised, or semisupervised learning systems, RL systems are not trained on sample datasets. Instead, they learn through trial and error. Although this eliminates the need for data labeling, it often limits the utility of RL methods to applications in which trial-and-error experiments can be efficiently simulated.

## TERMINOLOGY

It is important to be familiar with the terminology used in ML studies (**Table 1**).

## PREDICTIVE MODELING WORKFLOW IN MACHINE LEARNING

The main steps for developing supervised ML models are described here. However, many of

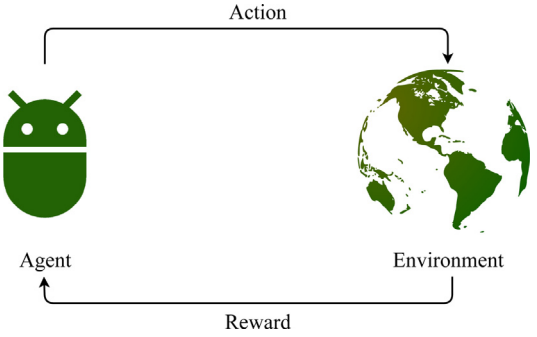



**Fig. 1.** In RL, an agent conducts a sequence of actions in an environment. The agent then receives a reward for its actions resulting in a desired outcome. Also, the agent preserves a state variable representing the current situation of the agent in the environment.

**Table 1**
**Summary of terminology used**

| | |
|---|---|
| Data point | An observation with 1 or more categorical or numerical attributes. The phrases sample, observation, and data point are used in this article interchangeably |
| Feature | A feature refers to an attribute of a data point and can represent a categorical or numerical value. Often the phrases data attribute, input variable, and feature are used interchangeably |
| Majority class | The category with the largest number of data points |
| Minority class | The category with the smallest number of data points |
| Class imbalance | A scenario in which there is a large difference between the number of samples in the majority class and the number of samples in the minority class |
| Estimator | A computational model used for estimating a variable |
| Predictive modeling | The process of developing an estimator |
| Classification model | An estimator for predicting the category of a given data point among a predetermined set of a priori known values |
| Regression task | A predictive model where a continuous value is assigned to each input data point |
| Hyperparameter | Model settings that are not learned during the model training process but may have a substantial effect on model performance |
| Generalization error | The error made by a model when applied to data that were not used for model development |

the concepts discussed apply to unsupervised ML models as well. Although the common perception for developing a ML application focuses on model training and evaluation, there are other steps essential to the successful development of generalizable and high-performing ML models. Fig. 2 shows a workflow showing the essential steps in developing such models.

### *Data Acquisition*

Data acquisition is a fundamental step in developing any ML workflow. Failing to include key attributes or confounding factors might lead to unintended consequences for the downstream analysis in model development and cripple the whole study. Data entry error is another common source of error in the health care domain. Most often, simple spreadsheets are used as the means

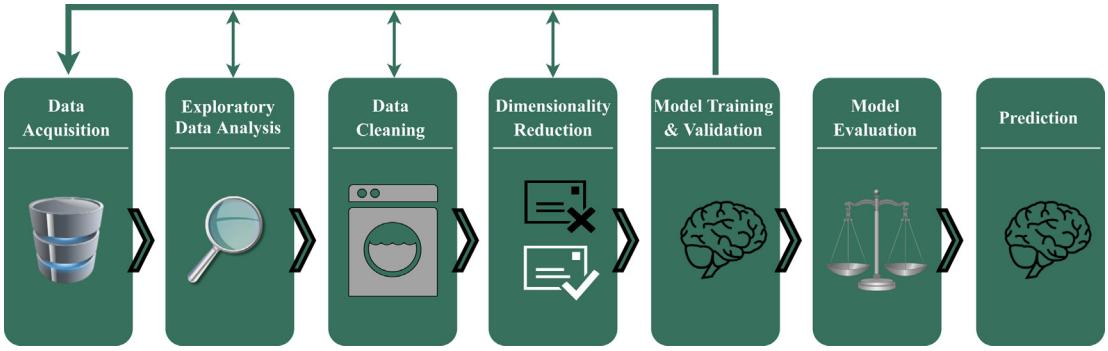



Fig. 2. An ML model development workflow. ML model development is an iterative process. A dataset is split into training, validation, and test sets. The training and validation sets are used for model training and validation, and the test set is used for model evaluation. To provide an unbiased estimate of the generalization error, a test set is locked away and is not used during model development. This test set should be used only once in the model evaluation step. Alteration of the model after observing the results on the test set makes the model error on the test set a biased and overoptimistic estimate of the model generalization error.

for data collection. Therefore, errors occur when failing to enter a record, resulting in missing values; entering incorrect values for a data record; and inconsistent notation for a specific record across the dataset. These kinds of errors could be easily avoided by developing data entry tools that enforce validity rules for data records. For example, a validity rule for the age attribute could be that age must not be negative or larger than 150. For categorical features, validity rules that enforce the selection of a value from a predetermined set of values could eliminate the inconsistency in notation across a dataset.

In retrospective studies, data have already been collected; therefore, there is no room to develop data entry pipelines, but awareness of these common issues could benefit other steps in a ML workflow, such as exploratory data analysis (EDA) and data cleaning, which are explained next.

### Exploratory Data Analysis

EDA is an important step in a ML application and refers to operations such as calculating summary statistics and visualization of various features and their interactions. EDA could reveal characteristics of the data and set the directions for downstream analysis such as data cleaning, feature selection, hypothesis generation, and the choice of ML models.

Although data visualization of high-dimensionality data points is challenging, it still is an insightful EDA approach. For visualizing high-dimensional data points, it is necessary to map them to a two-dimensional or three-dimensional space before visualization. This process should be accomplished in a way that the transformed data points represent the true characteristics of the original data points. Principal components analysis (PCA) is one approach that is commonly used in the process of visualizing high-dimensional data. The first 2 or 3 principal components of PCA are used to map the original data points to a two-dimensional or three-dimensional space, respectively. **Fig. 3**A shows the application of PCA for visualizing the test set of the Modified National Institute of Standards and Technology (MNIST) handwritten digit dataset.[15]

Another approach that is widely used for visualizing high-dimensional data is t-distributed stochastic neighbor embedding (t-SNE).[16] The main advantage of t-SNE is its capability to preserve the local structure among data points after transforming them into a two-dimensional or three-dimensional space.[16] This process could be observed from **Fig. 3**B, because data points corresponding with different handwriting styles of each digit cluster together when visualized by t-SNE.

Multidimensional scaling (MDS) is another common visualization technique that, given the pairwise distance between $n$ data points, creates a low-dimensional representation that preserves the relative distance between data points.[17]

### Data Cleaning

The presence of erroneous, corrupted, irrelevant, or missing values in a dataset used for developing a ML application could lead to models that perform poorly when applied to external data. Data cleaning refers to the processes of eliminating or amending the erroneous, corrupted, irrelevant, or missing values in a dataset. It is an essential step in building a ML application and often comprises a substantial portion of time and effort allotted to developing an ML application. Skipping this step often leads to learning models that are negatively affected by erroneous or irrelevant attributes of data points, leading to low ability to generalize.

Missing values, which refer to the absence of observations for an attribute of a data point, is a common phenomenon in health care data. Missing values may result from a measurement not being made for a patient or because of data entry error, achieving conclusive results from one test that eliminates the need for the other tests, losing access to patients before taking all measures, and so on.

Most ML techniques cannot intrinsically handle missing values. Consequently, such data points might be removed from the study or a prediction might be made as a substitute for the missing values. The former is only applicable where the number of samples with missing values is negligible compared with the number of samples in the dataset. The latter, which is also referred to as data imputation, is often the only choice when working with datasets with a large number of missing values or when working with small datasets. The most straightforward approach is to use statistical measures such as mean or median for continuous variables and mode for categorical variables. Developing ML models to predict missing values from the other features has also been used.[18]

### Dimensionality Reduction

Building ML models using high-dimensional data is challenging; such models tend to overfit in the absence of large-scale datasets (**Fig. 4**). Often there is a large number of features available, of

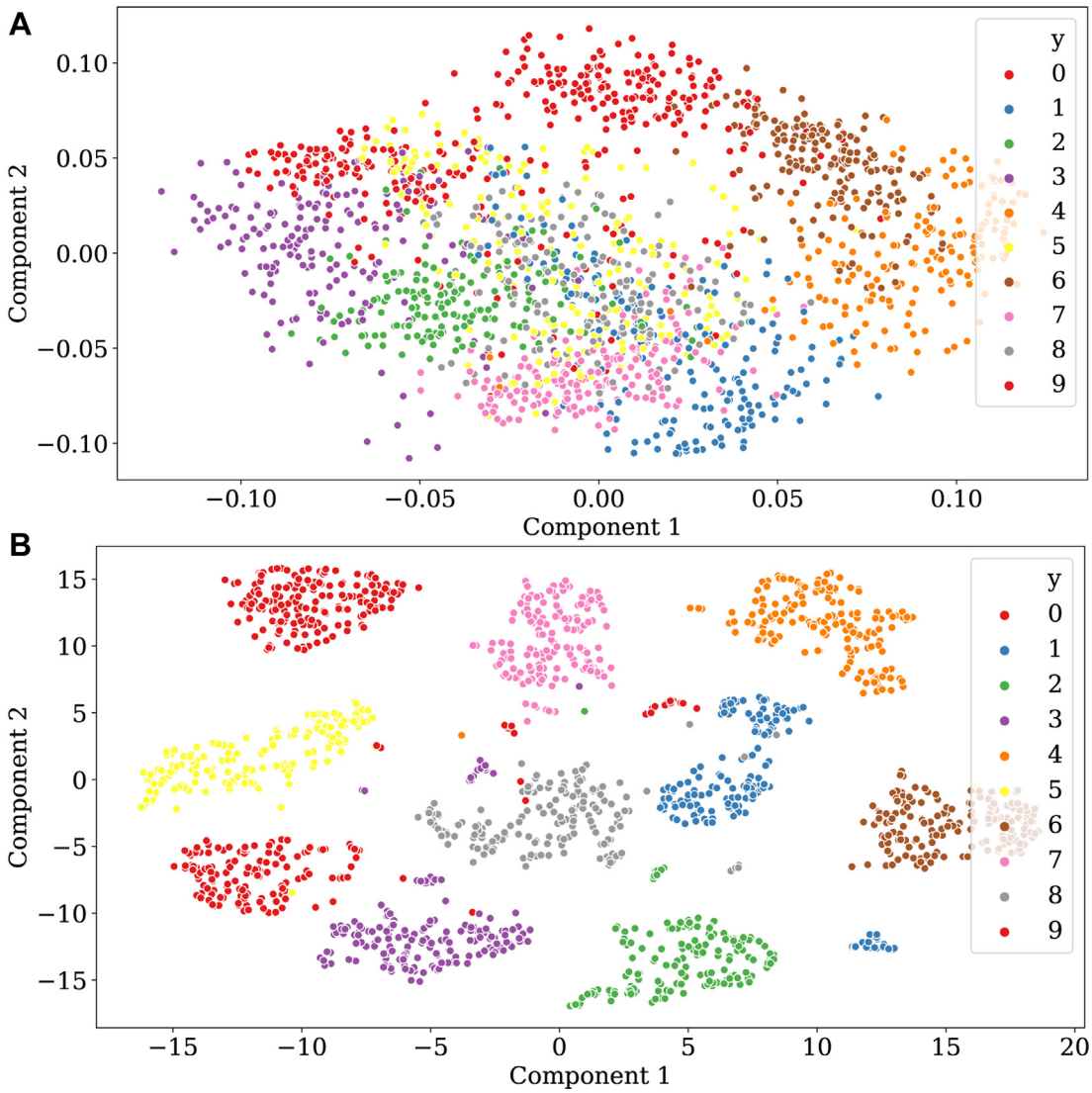



**Fig. 3.** Visualizing the test set of the Modified National Institute of Standards and Technology (MNIST) handwritten digit dataset[15] using PCA (*A*) and t-distributed stochastic neighbor embedding (t-SNE) (*B*).

which some might be irrelevant or redundant. Including all of these features does not improve the performance of ML models. Instead, considering them as the model inputs often leads to performance degradation and overfitting. In addition, including these features makes the model more complex and more difficult to interpret.

Dimensionality reduction refers to the methodologies used to transform the original high-dimensional data points to data points in a lower dimension while preserving the key properties of the original data points relevant to a task at hand. Dimensionality reduction methods are categorized as feature extraction and feature selection techniques.

Feature extraction tries to combine the current features and provide a low-dimensional representation of the original data point. Methods based on PCA, and linear discriminant analysis are among the most widely used approaches for feature extraction.[5]

In contrast, feature selection refers to the methodology used for selecting a subset of the original features while preserving the key properties of the original data points. The common families of feature selection methods are filter methods, wrapper methods, and embedded methods.[19,20]

In filter methods, a function representing the strength of association between a given feature and the outcome of interest is used to select

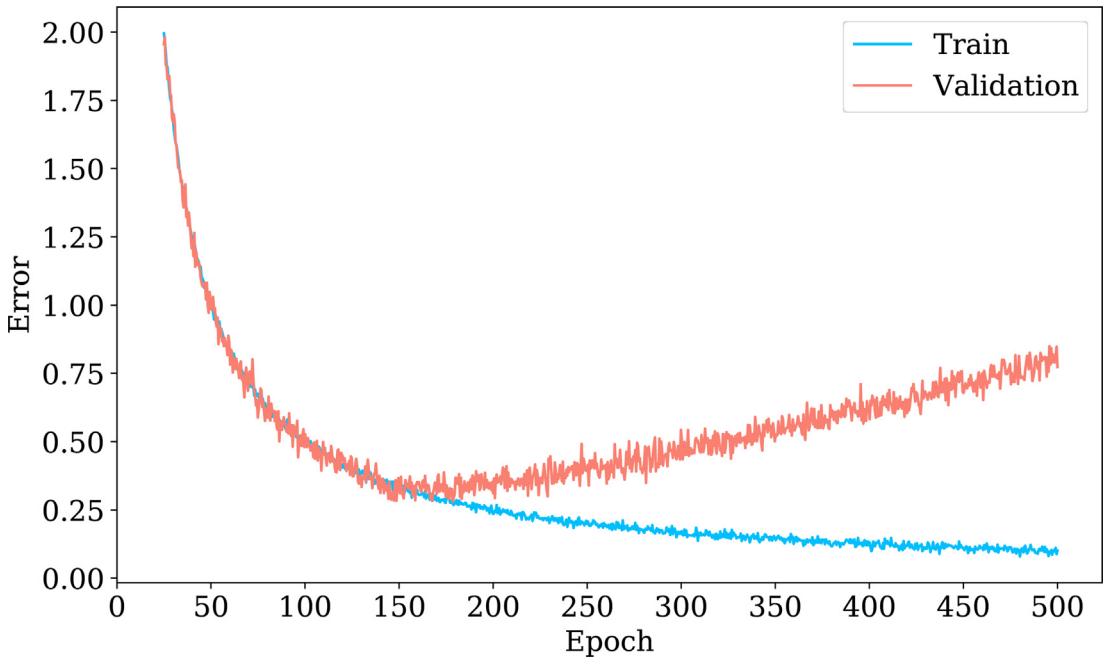
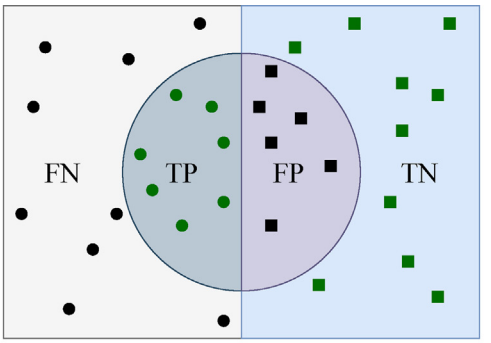



Fig. 4. Training and validation error of a multilayer perceptron (MLP). The final model at iteration 500 is considered an overfitted model, as depicted by the large difference between training and validation error. A remedy for such a scenario is to use an early stopping criterion to stop model training when training error and validation error start to diverge.



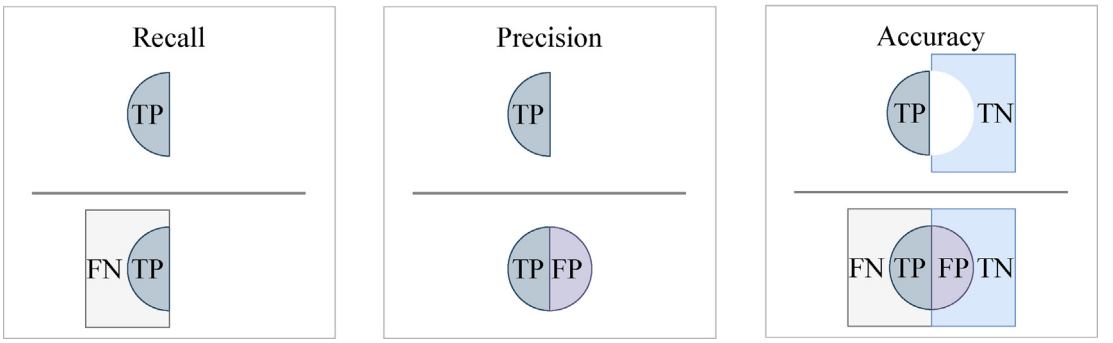




Fig. 5. Visualizing accuracy, recall (aka sensitivity), and precision, which are the common performance measures for classification tasks. Given samples from two categories, we can refer to samples as positive or negative (two classes). The left rectangle represents positive samples, and the right rectangle represents negative samples. The circle contains all samples predicted as positive. Given the model predictions, each sample can be considered as TP (true positive), TN (true negative), FP (false positive), or FN (false negative).

features that have a strong relationship with the outcome of interest. For example, in classification tasks, the $\chi^2$test can be used to decide whether a categorical variable should be selected. Similarly, analysis of variance (ANOVA) can be used for deciding whether a continuous feature should be selected. Although they are computationally informative, these methods might lead to the selection of redundant features, because they independently compare each feature with the outcome variable.

Wrapper methods use an estimator (a ML model) for selecting a set of features. These methods are categorized into 3 main groups: forward selection, backward elimination, and recursive feature elimination methods. Starting from an empty set of features, forward selection methods assess the goodness of each new feature by its contribution to improving the model performance. The feature with the highest contribution is added to the set of selected features. Backward elimination follows a similar approach. Starting from the set of all features, a backward elimination method estimates the performance degradation when dropping each feature. The feature that leads to the least performance degradation will be eliminated. This process is continued in an iterative manner until a predetermined number of features are left. In recursive feature elimination, an estimator is used to assign feature importance weights to features. Given a dataset, this estimator is trained using all features. Then the features with low feature importance will be eliminated. The estimator is trained again using the remaining features, and some other features with low importance values will be eliminated. This process is continued recursively until a desired number of features remain.

Embedded methods are another group of feature selection methods that incorporate feature selection as part of the learning process. Least absolute shrinkage and selection operator (LASSO) and ridge regression are two of the most well-known embedded methods for feature selection.[5]

### *Model Training and Validation*

After conducting dimensionality reduction, the reduced or transformed set of features is used to train ML models. The available data are split to train, validation, and test sets. The train set is used to train models. These models could be all from the same family of models differentiated by model hyperparameters (eg, decision trees of different heights) or they could be from different families of ML models (or a mixture of both). From these sets of models, often the best performing one is selected based on performance measures (**Fig. 5**) on validation data.

The successful development of ML models requires the availability of datasets that are representative of the task at hand. One of the challenges in building ML models is data imbalance. Data imbalance, also referred to as class imbalance, is an integral characteristic of many datasets in the health care domain and is considered an impediment to developing generalizable models in the health care domain, where class imbalance happens because of the rarity of a disease or phenotype. This imbalance results in a dataset with a small number of samples from the rare classes. Class imbalance can be measured as the difference between the size of the majority class and the size of the minority class.

Models constructed using imbalanced data may have a very large bias. Therefore, class imbalance should be evaluated and dealt with when building ML models.[21,22] Oversampling and undersampling are two common strategies for dealing with class imbalance. In oversampling, observations from the minority classes are selected using sampling with replacement to increase the number of samples from the minority classes. In contrast, undersampling uses a bootstrapping approach where a smaller number of observations from the majority class are selected to decrease the class imbalance. The main shortcoming of the undersampling approach is that it discards a large number of samples from the majority class. Therefore, it may lead to information loss. Oversampling has also been criticized for changing the underlying distribution of the minority class, because a few observations in the minority class are used repeatedly to represent the data distribution for the minority class.

Understanding the concepts of bias and variance and their relationship with model complexity is essential for developing generalizable and high-performing ML models. Given a set of labeled data points, ML models are trained to provide a close estimate of the unknown mapping between inputs and outputs. Often models are based on simplifying assumptions such that the estimates made by the model may systematically differ from the actual mapping. For example, a linear regression model assumes that there is a linear relationship between the inputs and output. When used for predicting a nonlinear relationship, this leads to a systematic error between the predicted and expected output. Model variance refers to the deviation in model predictions caused by the use of different training datasets; it is a measure of how much a learned model changes when using different training sets, all sampled from the same underlying distribution.

The bias and variance are directly related to the model complexity. The complexity of a model is associated with the number of parameters it uses and how these parameters interact to make the model predictions. When comparing a linear model and a multilayer perceptron (MLP), both developed for the same task, the linear model is considered less complex. Both linear and MLP models are explained in relation to supervised learning methods. In general, a high-complexity model tends to have a high variance, and a low-complexity model tends to have a high bias. High-complexity models tend to overfit (ie, achieve a low error on training data and high error on test data). In contrast, models with low complexity tend to underfit. Such models achieve a high error on the training set compared with the irreducible error, which is the minimum achievable error for the task at hand given a dataset.

Controlling model complexity is a strategy for developing generalizable and high-performing ML models. For example, the performance of an underfitted MLP can often be improved by increasing its complexity through introducing more layers or increasing the number of neurons in each layer. For dealing with an overfitted MLP model, the model complexity can be decreased by decreasing the number of layers or the number of neurons per layer. Another approach used for controlling model complexity is the use of regularization.[6] For a more detailed discussion of algorithm training and validation, see the accompanying article in this issue concerning ML algorithm validation.

### *Model Evaluation*

Proper evaluation is essential for developing generalizable models. A comprehensive guideline for conducting sound and rigorous model evaluation is outside the scope of this article and is provided in a different article in this issue. Model evaluation is briefly discussed here.

As a common practice when evaluating a ML model, available data are partitioned into training, validation, and test sets. For small datasets, it is a common practice that 70% of the data points are used for training, 15% for validation, and 15% for testing. The training set is used for model training, the validation set is used for hyperparameter tuning, and the test set is used for model evaluation. The test should not be exposed during model training and hyperparameter tuning so that the error on the test set provides an unbiased estimate of the generalization error for the model. This estimate can be achieved through a nested cross-validation approach.[23]

During model training and hyperparameter tuning, performance measures are used to guide the training process and select the hyperparameters. The learning algorithm tries to update the model parameters in a way that the performance measure is improved. Also, during hyperparameter tuning, a performance measure is used to select hyperparameter values. Such performance measures should reflect the key performance indicators for the application. Choosing inappropriate performance measures may lead to unwanted consequences. For example, when selecting the hyperparameter C for a support vector machine (SVM) classifier developed in the presence of class imbalance, using accuracy for selecting hyperparameters may result in selecting values that cause the minority class to be ignored.

## SUPERVISED LEARNING

In supervised learning, a dataset of labeled data points is used for building predictive models. These models can be either for predicting categorical values (ie, classification tasks), or for predicting continuous values (ie, regression tasks). Several widely used supervised learning methods used for predictive modeling are described here.

### *Linear and Logistic Regression*

A linear regression model is built on the assumption that the response variable (the model output, also referred to as dependent variable) is a linear function of explanatory variables (input variables, also referred to as independent variables). **Fig. 6** shows a simple regression model for 1 explanatory variable and 1 response variable. In general, a linear regression model for *n* explanatory variables

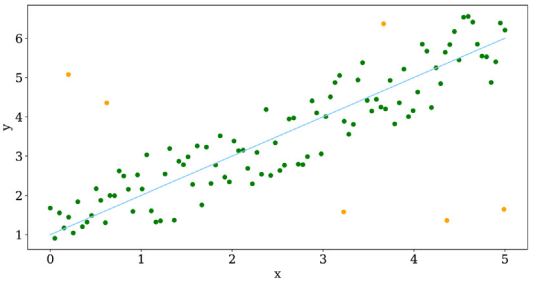



**Fig. 6.** A simple linear regression model describing the relationship between an explanatory (input) variable on the x axis and a response (output) variable on the y axis. Each circle represents a data point. The linear regression model (*blue line*) explains the relationship between the explanatory and response variable for most of the data points, except for the outliers (*orange*).

$x_1, \cdots, x_n$ and a response variable $y$ can be represented as a linear function $y = a_1x_1 + \cdots + a_nx_n + b$, where $a_i$ $(1 \leq i \leq n)$ and $b$ are referred to as regression coefficients and intercept, respectively.

The linear regression can be used for predicting continuous variables. In a binary classification problem, where the goal is to predict a binary output variable, logistic regression can be used to assign a class to each input data point. In logistic regression, a sigmoid function is used to adapt the linear regression to a binary classification as follows:

$$y = \frac{1}{1+e^{a_1x_1+\cdots+a_nx_n+b}}$$

where $e$ is the natural number. For a given data point, this equation generates a value between 0 and 1, representing its class-membership probability. This value is compared against a threshold value (eg, 0.5) to predict the class of the data point. **Fig. 7** shows a logistic regression model for a single explanatory variable.

## Support Vector Machines

Assume that there is a dataset of samples, each belonging to 1 of 2 known classes. Also, assume that the data are linearly separable; that is, there exists a hyperplane that correctly separates samples from the 2 classes. Examples of hyperplanes are a straight line (in two-dimensional space) or a plane (in three-dimensional space). As shown in **Fig. 8**A, there is often more than 1 such hyperplane. Although these hyperplanes might all perform equally well on training data, their performance on test data might substantially vary. Therefore, the natural question is: which one leads to better performance and more generalizable results when applied to previously unseen data?

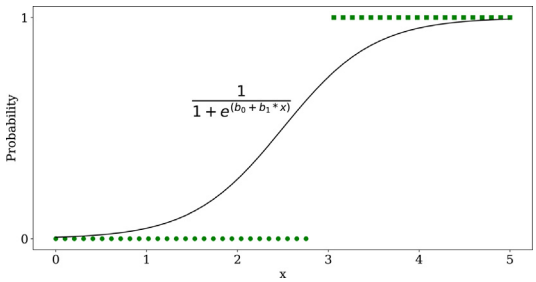



**Fig. 7.** A logistic regression model for a binary classification task (circles vs squares). Each data point is represented by a circle or a square, depending on its class membership. The x axis shows the explanatory (input) variable, and the y axis shows the probability of a data point being a member of the class square.

SVMs[24] answer this question by selecting 2 parallel hyperplanes that separate the 2 classes, such that the margin, which is the distance between these 2 hyperplanes, is maximal.[25] These 2 hyperplanes are called margin maximizing hyperplanes. SVM uses the support vectors, which are the data points on the margin maximizing hyperplanes, to define a decision surface between these margin maximization hyperplanes. The decision surface can then be used for classification (**Fig. 8**B).

Because of the existence of noise and outliers, there might be overlap between data points from 2 classes (**Fig. 8**C). The soft-margin SVM has been proposed to address this challenge. Soft-margin SVM, by relaxing the condition on the margin maximizing hyperplanes, allows some data points to violate these separating hyperplanes (see **Fig. 8**C). Soft-margin SVM uses a nonnegative hyperparameter C to control the amount of relaxation (ie, violation of the margin maximizing hyperplanes). A value of C equal to 0 leads to a hard margin SVM (ie, no violation of the margin maximizing hyperplanes is allowed). The higher the value of C, the more relaxed SVM will be regarding violations of margin maximizing hyperplanes. Larger values of C make SVM less sensitive to the noise in the training data, resulting in increasing bias and decreasing variance of the model. For data that are not linearly separable (**Fig. 8**D), SVM uses a so-called kernel trick.[5] The main idea behind kernel trick is that samples from 2 classes that are not linearly separable in the space of the original data may be linearly separable in higher dimensions. When using the kernel trick, this is efficiently achieved without directly transforming the data to higher dimensions. Radial basis function kernels and polynomial kernels are commonly used when building SVM models.[5] SVM can also be adopted and used for regression tasks.[5]

Binary classification models such as SVM and logistic regression can also be used for multiclass classification tasks through a 1-versus-1 or 1-versus-rest approach. Suppose there are $n$ different classes. In the 1-versus-1 approach, a binary classifier is developed for each pair of classes. This binary classifier results in $n \times (n+1)/2$ different binary classifiers. For a given input, a majority vote among these binary classifiers will be used to determine the predicted class.

## K-nearest Neighbors

K-nearest neighbors (K-NN) is the most intuitive ML technique commonly used for classification tasks, although it can be used for regression tasks as well. A K-NN classifier uses a set of

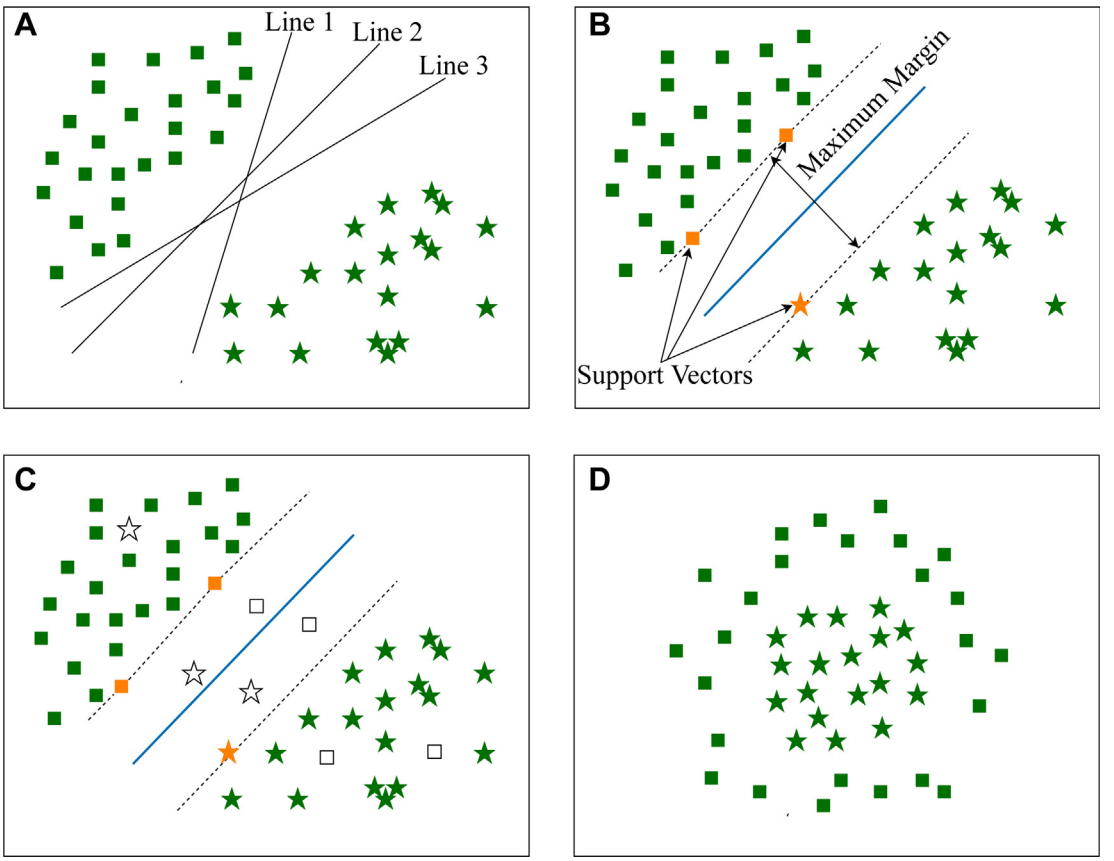



Fig. 8. Classification scenarios using SVM. Data points are represented as a star or a square, with the shape representing the class of the data point. (*A*) There might be more than 1 hyperplane that classifies data equally well. (*B*) SVM. Support vectors are the data points (*orange*) on the margin maximizing hyperplanes (*dashed lines*). The blue line represents the SVM decision surface, and any data point above the decision surface is classified as one class (class square here), and any data point under the decision surface is considered as the other class (class star here). (*C*) The soft-margin SVM, where the condition on margin maximizing hyperplanes is relaxed, and some data points (*white*) can violate these margins. (*D*) A dataset that is not linearly separable; that is, there is no hyperplane capable of separating samples from these 2 classes in two-dimensional space. SVM uses the kernel trick to deal with datasets that are not linearly separable.

labeled data points. For a new unlabeled data point, the K-NN classifier predicts the class label as the vote of majority among the *k* labeled data points that are the most similar to this observation. The value of *k* is considered a hyperparameter for this method. The degree of similarity is measured using a distance function. Commonly used distance functions for continuous variables are euclidean distance (also known as L2 norm), Manhattan distance (also known as L1 norm), Pearson correlation, and Spearman correlation. Hamming distance, which is the number of mismatches between attributes (ie, features) of 2 data points, is also commonly used for categorical variables. Fig. 9 shows a K-NN binary classification model. The value of *k* is selected empirically by observing the model performance across different values for hyperparameter *k*. Small values for *k* leads to a high model variance and low bias and large values of *k* lead to low variance and high bias.

K-NN is a nonparametric approach, intuitive to understand, and easy to implement. It does not have an explicit training and can be easily adapted to changes simply by updating its set of labeled observations. On the negative side, K-NN has a long inference time because each new observation must be compared against the set of labeled observations. It also leads to poor performance when applied to imbalanced datasets. The performance of K-NN is also sensitive to the choice of its hyperparameter *k*. When using K-NN, it is crucial to use homogeneous features because the distance between data points with heterogeneous

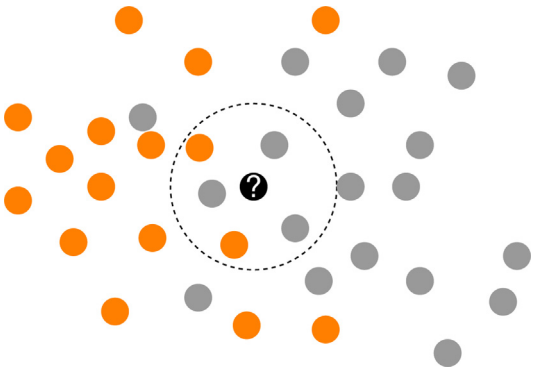


Fig. 9. K-NN classifier for k = 5. To classify a new data point, depicted with a question mark, K-NN calculates the distance of the new data point from each data point in the training set. In this example, the euclidean distance is used as the distance metric. The data points associated with the k = 5 smallest distance values (data points inside the *dashed circle*) are used to predict a class label for the new data point. The most frequent class between these 5 data points (*gray*) is the predicted class for the new observation.

features might be affected by scale and variability of a few features leading to a loss of information from the other features. For example, assume 3 features $x_1$,$x_2$, and $x_3$, where $x_1$ and $x_2$ are normalized variables with values between 0 and 1 and $x_3$ is a variable with values between 1 and 100. When calculating distance between data points with such features, the contribution of $x_1$ and $x_2$ to a distance function such as euclidean distance will be negligible and $x_3$ will overwhelm the calculated distance values. In this scenario, scaling $x_3$ to values between 0 and 1 will make the calculated distances more meaningful. Alternatively, a customized distance function can be used that is not sensitive to heterogeneous features.

### Naive Bayes

Naive Bayes methods are probabilistic classifiers designed based on the Bayes theorem. In order to predict the class of unknown observations, these approaches rely on the naive assumptions that each pair of features are conditionally independent given their corresponding class label, which means that, for a given class, each feature is statistically independent of the other features; that is, the value of a feature $x_i$ is not related to the value of a feature $x_j$. Naive Bayes methods are computationally efficient and, despite their naive assumptions, have been successfully used for applications such as text classification.[26] For a given observation x = $(x_1, x_2, \ldots, x_n)$, the probability of being a member of class $y$ can be calculated as follows:

$$P(y\,|x_1,\,\cdots,x_n) = \frac{P(y) \times P(x_1|y) \times \cdots \times P(x_n|y)}{P(x_1,\,\cdots,x_n)}$$

**Equation 1**

For an observation $x$, the class $y$ with the highest $P(y|x_1, x_2, \ldots, x_n)$ is predicted as the model output. The value of $P(x_1, x_2, \ldots, x_n)$ does not have any effect on selecting the class with the highest $P(y|x_1, x_2, \ldots, x_n)$. Therefore, Equation 1 and Equation 2 lead to predicting the same class label,

$$P(y\,|x_1,\,\cdots,x_n) \approx\; P(y) \times P(x_1|y) \times \cdots \times P(x_n|y)$$

**Equation 2**

where P(y) can be estimated as the relative frequency of class y using the data points in the training datasets. Naive Bayes methods differ regarding their approach for calculating the likelihood of the feature $x_i$ given class y; that is, $P(x_i|y)$ $(1 \leq i \leq n)$. For example, gaussian naive Bayes method uses a gaussian distribution and multinomial naive Bayes uses a multinomial distribution for calculating $P(x_i|y)$.[5]

### Decision Trees

A decision tree resembles a tree structure with internal and leaf nodes (Fig. 10). Each internal node in the tree represents a test on 1 particular attribute of the data, and each branch of the tree represents a test outcome. Leaf nodes are not associated with a test; they represent all samples that meet the test outcomes of the predecessors of the leaf node; that is, the nodes in the path connecting the root (the first test) to the leaf node.

A decision tree is built in an iterative manner. Starting from the whole set of training data points, an attribute and a test on that attribute leading to the best class separation among the data points is selected. According to the test outcome, 2 or more nodes that are connected to the current node are created, and each sample is assigned to 1 of these nodes. If the selected attribute is a categorical variable with $n$ possible values, often $n$ new nodes are generated, and samples are partitioned and assigned to these nodes. For attributes that are continuous variables, often 2 new nodes are generated, and all samples with their attribute values less than or equal to a threshold are assigned to 1 node, and the rest of the samples are assigned to the other node. The value of this threshold can be selected among the attribute values for samples in the training dataset. If samples in a new node meet a stopping criterion, that node is considered as a leaf node and an

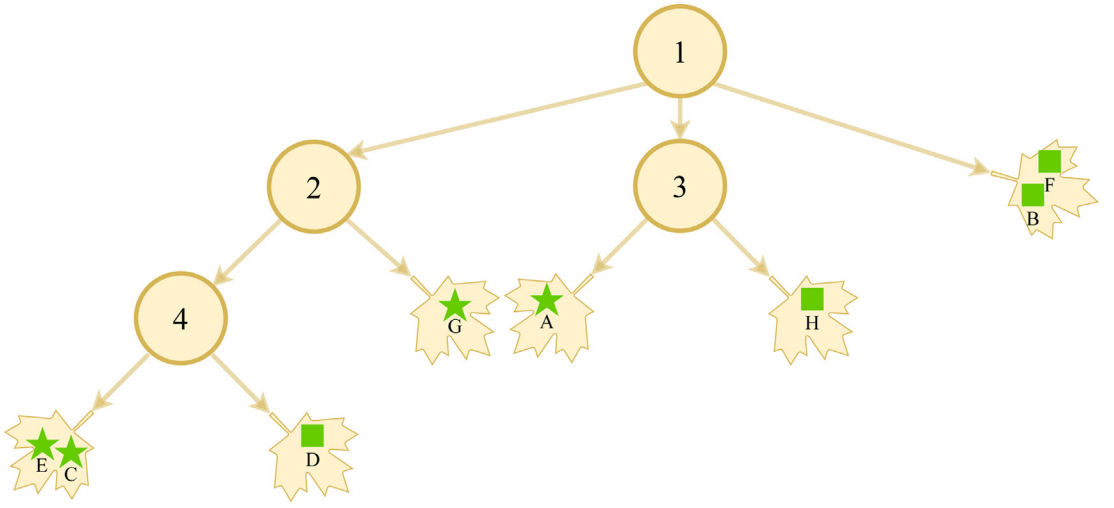



**Fig. 10.** A decision tree. Node 1 is considered the root of the tree. Nodes 2, 3, and 4 are internal nodes, and the rest of the nodes are considered the leaf nodes for the tree. An internal node corresponds to a test of an attribute of the data points. Each outgoing edge from an internal node corresponds to 1 possible result of a test. A new unlabeled data point starts from the root; based on the result of the test corresponding with the root, it goes to the next node. If the next node is an internal node, the same procedure is applied. In addition, this new data point is assigned to a leaf node. The class label associated with that leaf node is used as the prediction for the new unlabeled data point.

internal node if otherwise. For samples assigned to a leaf node, no further partitioning is required. In contrast, for the samples assigned to an internal node, the same partitioning process is applied. This process continues recursively until there is no unprocessed internal node.

When building a decision tree, the goal is to identify attributes that, when applying a test on them, separates the samples optimally. The generated decision tree can then be used to predict the target class or value for a new data point. Gini index and entropy are commonly used to estimate the goodness of an attribute and its resulting partitions.[5]

Decision trees are intuitive and easy to interpret. They can handle categorical and continuous attributes and require little to no data preprocessing. However, decision trees are known to achieve poor performance for imbalanced datasets. They are also sensitive to noise in the training data and can easily overfit. Among strategies used for alleviating this issue are limiting the depth (height) of the tree or setting a minimum number of samples required for further partitioning. Decision trees with high depth have high variance and low bias and shallow decision trees lead to low variance and high bias for the model. Bagging and boosting[5] are strategies commonly used to deal with high model variance or high bias in decision trees, respectively. These approaches can also be applied to other models.

### *Random Forest*

A random forest[27] is an instance of a family of ensemble learning methods[28,29] that are known as bootstrap aggregation or bagging.[5] The main goal of bagging is to address the high variance of models, which leads to overfitting and lack of ability to generalize. Bagging is a general concept, not limited to decision trees, and can be applied to deal with high variance.

A random forest model is made up of a collection of decision trees. The results of these decision trees for a given data point are then combined, using a vote of majority for classification and averaging for regression tasks, to make a prediction to achieve a performance better than the performance of each single decision tree alone. These trees are independently trained on random subsamples of the training set.

For each decision tree in a random forest, a training set is generated by random sampling with replacement from the original training set. Also, a random subset of features is selected. The decision tree then is trained using the generated training set and the selected features. This method results in lower correlations across the decision trees, which translates to improving the overall performance of the resulting random forest model.[30]

### Neural Networks

Most of the recent success of ML is caused by the availability of a large volume of data and computer power as well as the advances and innovation in deep learning. Deep learning models consist of a large number of computational units arranged across many layers to make learning complex tasks feasible. Deep MLP,[6] recurrent neural networks,[31] and convolutional neural networks[32] are among the most widely used deep learning approaches. These approaches are outside of the scope of this article and are discussed in another accompanying article in this issue. MLP is briefly discussed here. MLP provided a systematic methodology for training neural network models using the backpropagation algorithm and can be considered as the cornerstone for the success of deep learning models.

A MLP is built on the concept of the perceptron that was proposed as an artificial model for the neurons in the brain and their connectivity.[6] Each neuron is a computational unit that first calculates a linear function of its inputs and then applies a nonlinear function, referred to as activation function, to the resulting value. The linear function is characterized as a set of parameters, referred to as weight and bias values. Note that activation functions are essential to introduce nonlinearity; otherwise, even a complex network of neurons cannot learn nonlinear functions.

A MLP consists of several consecutive layers of stacked neurons. **Fig. 11** shows an example of a MLP, composed of 4 layers. Each layer is made by stacking several neurons. The leftmost layer and the rightmost layer are referred to as the input layer and the output layer, respectively. The 2 layers in the middle are called hidden layers. In a MLP, the output of the neurons from each layer is fed to all neurons in the subsequent layer.

These network parameters (ie, the parameters for all neurons in the network) are learned through the backpropagation algorithm in an iterative process. This process starts by randomly initializing the parameters. In each iteration, these parameters are updated with the goal of assigning weights that lead to minimum prediction error; that is, the minimum discrepancy between the network output and the ground truth (labels). A loss function, reflecting the difference between the network predictions and the ground truth, is used to calculate the loss values in each iteration. The calculated loss values are then used to update the network parameters using the backpropagation algorithm. The training process is continued until a stopping criterion is satisfied. The network with the trained parameters can then be used to make predictions for unlabeled data.

### Unsupervised Learning

As mentioned before, unsupervised learning methods do not rely on labeled data. These methods are often used to discover patterns among data points or to provide a low-dimensional representation of data points.

### Principal Component Analysis

PCA is the most widely used dimensionality reduction technique.[33] Given a set of $n$-dimensional data points, PCA generates a new set of variables referred to as principal components that are linear combinations of the original features. The first principal component is selected in a way that, if data points are projected onto it, most of the variation of the original data can be explained in that dimension. The consecutive principal components are designed to explain the maximum amount of the unexplained variation in the data; this procedure continues until all the variance is explained using all principal components. The first few principal components can often explain most of the variation in the dataset; therefore, they can be used to project the original data points to these principal components, resulting in reducing the data dimensionality while preserving most of the variation in the original data points.

### Clustering

Clustering is an unsupervised means for finding interesting patterns in the data, such as finding groups of samples that share common characteristics. K-means clustering and hierarchical clustering are among the most widely used clustering methods.

#### K-means clustering

K-means clustering aims to partition the data points of a dataset into $k$ clusters. This partitioning is accomplished by first initializing the center of each cluster based on an initialization strategy. One strategy is to randomly choose $k$ observations from the dataset and select these data points as the initial centroids of the $k$ clusters. In an iterative manner, each time all data points in the dataset are assigned to one of the $k$ clusters based on their proximity to the cluster centroids. Next, each cluster centroid is updated to be the average of all data points in its corresponding cluster. This process continues until there is no change in the cluster centroids or a stopping criterion, such as the maximum number of iterations, is satisfied.

K-means is a scalable approach with guaranteed convergence. However, the converged solution might be nonoptimal. K-means can easily

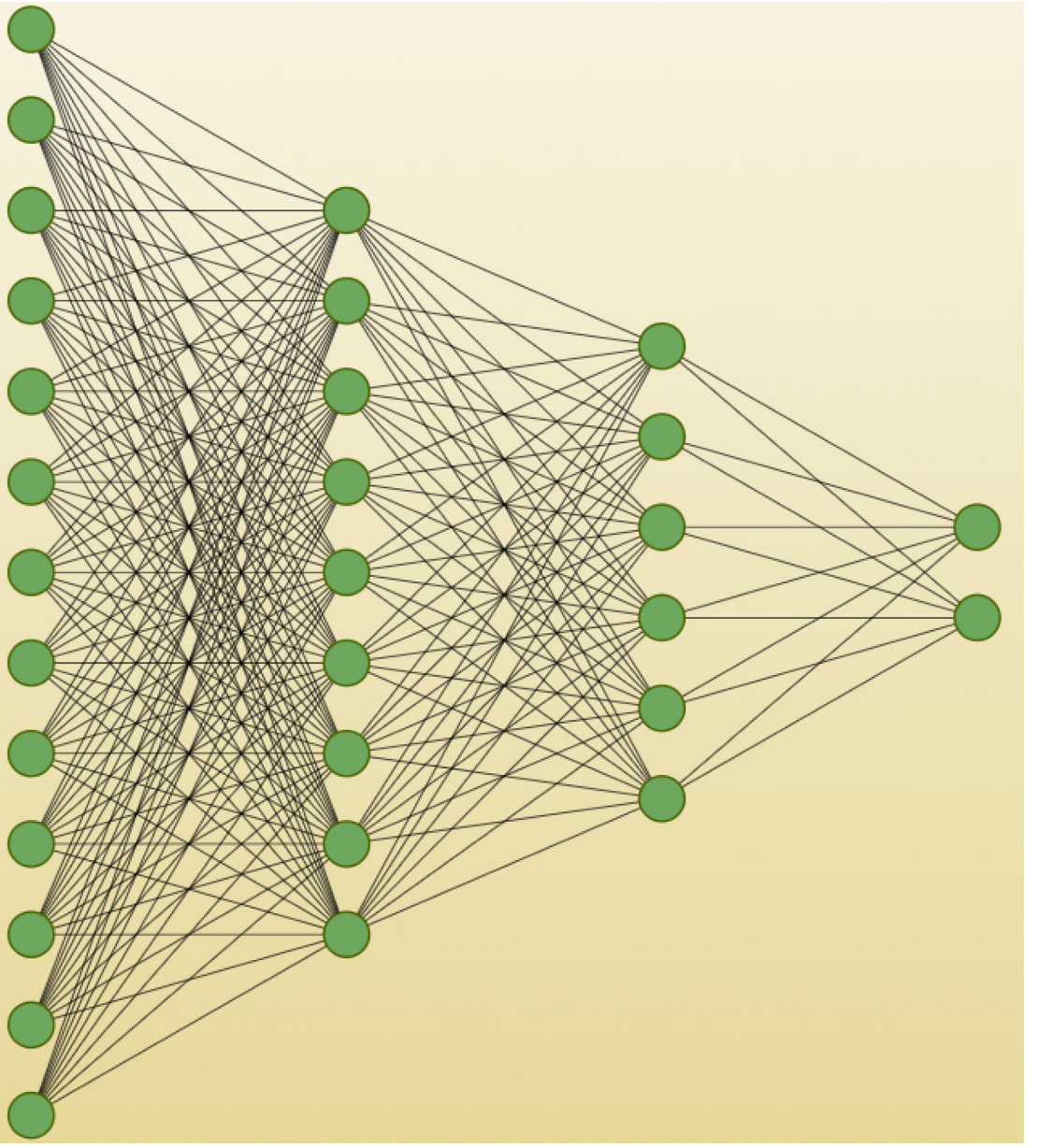


**Fig. 11.** A multilayer perceptron composed of 4 layers. The leftmost and rightmost layers are referred to as input and output layers, respectively. The 2 layers in the middle are called hidden layers. Each layer consists of several artificial neurons, depicted as green circles. Neurons in each layer are connected to all of the neurons in the subsequent layer.

adapt to the changes in data distribution by introducing the new data points and updating its centroids. On the negative side, K-means is sensitive to the choice of $k$ (ie, the number of clusters $k$). Therefore, exhaustive search and heuristic approaches[34] are used for selecting the value of $k$. K-means is also sensitive to the choice of initial centroids; therefore, in practice, it is run in parallel several times, and the result of the best performing run is selected based on a performance evaluation measure.[34] Note that K-means requires the features to be homogeneous; otherwise, the distance between data points might be driven by the features with large absolute values, as discussed for K-NN.

### *Hierarchical clustering*

Hierarchical clustering is a clustering approach that aims at creating a hierarchy of clusters (**Fig. 12**). Hierarchical clustering methods can

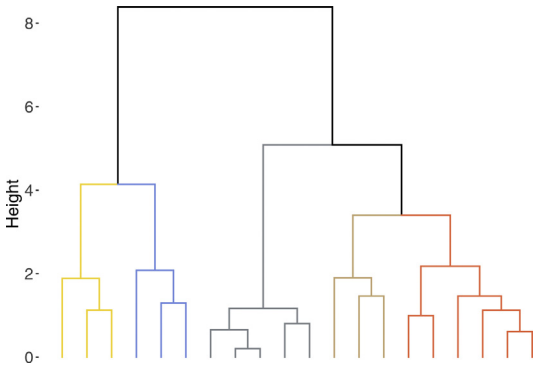



**Fig. 12.** A hierarchical clustering developed using an agglomerative approach with complete linkage and euclidian distance as its metric. Initially, each data point is present in its own cluster. These clusters are then joined in an iterative manner until there is 1 cluster containing all samples. A particular height in the dendrogram can be selected to establish a clustering for the data. Here, choosing a value of 4 for height leads to 5 clusters identified by different colors.

be divided into agglomerative and divisive methods. For a dataset of *n* data points, agglomerative methods start with *n* clusters, 1 for each data point. Then, in an iterative process, they try to merge these clusters until there is 1 cluster containing all *n* data points. In each iteration, the 2 most similar clusters are selected and merged. The similarity between clusters is quantified by a linkage criterion and a metric, such as euclidean or Manhattan distance. The metric is used for quantifying the distance between data points, and the linkage method is used to quantify the similarity between clusters. Complete linkage, single linkage, and average linkage are the most common linkage methods. Given 2 clusters A and B, complete linkage defines the distance between A and B as the distance between the 2 data points (1 from A and 1 from B) with the maximum distance. Single-linkage considers the distance between A and B to be the distance between the 2 data points (1 from A and 1 from B) with the smallest distance. Average-linkage methods calculate the distance between clusters A and B as the average of the distance between data points in A from the data points in B.

Divisive hierarchical clustering methods are similar to the agglomerative ones. The only difference is that instead of starting from *n* distinct clusters and trying to join them, they start with 1 cluster containing the *n* data points. Through an iterative process, they then split these clusters until reaching *n* clusters, 1 for each data point.

## CONCLUDING REMARKS

Because of the abundance of data and computer power, ML has the potential to contribute to solving complex problems in academia and industry. The health domain is a data-rich domain with data from different modalities. However, most often ML applications in this domain have been developed using a small number of samples because of factors such as rarity of disease or phenotype under study, limited financial and expert resources, and difficulty of developing multi-institutional large-scale datasets because of patient privacy and/or legal concerns.

When dealing with small datasets, a deep understanding of the necessary steps and considerations for developing ML models is crucial for making high-performing and generalizable applications. Following best practices for data acquisition, exploratory analysis, dimensionality reduction, model training, hyperparameter tuning, and model evaluation become a necessity in such scenarios.

In this article, common supervised and unsupervised classic ML models are reviewed. It discusses various steps in a ML workflow and how decisions made in these steps affect the performance and ability to generalize of ML models. It further discusses the model complexity and how it affects the model bias and variance for different models. Understanding the essential concepts and methods explained in this article provides the foundation for practitioners to develop high-performing and generalizable ML models.

## REFERENCES

1. Lee I, Shin YJ. Machine learning for enterprises: applications, algorithm selection, and challenges. Bus Horiz 2020;63(2):157–70.
2. Shatte AB, Hutchinson DM, Teague SJ. Machine learning in mental health: a scoping review of methods and applications. Psychol Med 2019; 49(9):1426–48.
3. Rajkomar A, Dean J, Kohane I. Machine learning in medicine. N Engl J Med 2019;380(14):1347–58.
4. Leo M, Sharma S, Maddulety K. Machine learning in banking risk management: A literature review. Risks 2019;7(1):29.
5. Friedman J, Hastie T, Tibshirani R. The elements of statistical learning, vol. 1. New York: Springer Series in Statistics; 2001.

6. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge (MA): MIT press; 2016.
7. Jain AK. Data clustering: 50 years beyond K-means. Pattern Recogn Lett 2010;31(8):651–66.
8. Ceglar A, Roddick JF. Association mining. ACM Comput Surv 2006;38(2). 5–es.
9. Lee JA, Verleysen M. Nonlinear dimensionality reduction. New York: Springer Science & Business Media; 2007.
10. Cheplygina V, de Bruijne M, Pluim JP. Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis. Med Image Anal 2019;54:280–96.
11. Van Engelen JE, Hoos HH. A survey on semi-supervised learning. Machine Learn 2020;109(2): 373–440.
12. Silver D, Hubert T, Schrittwieser J, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science 2018;362(6419):1140–+.
13. Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. Nature 2016;529(7587):484–9.
14. Amini A, Gilitschenski I, Phillips J, et al. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. IEEE Robot Autom Lett 2020;5(2):1143–50.
15. Alpaydin E, Kaynak C. Cascading classifiers. Kybernetika 1998;34(4):369–74.
16. Lvd Maaten, Hinton G. Visualizing data using t-SNE. J Mach Learn Res 2008;9(Nov):2579–605.
17. Borg I, Groenen PJ. Modern multidimensional scaling: theory and applications. New York: Springer Science & Business Media; 2005.
18. Donders ART, van der Heijden GJMG, Stijnen T, et al. Review: A gentle introduction to imputation of missing values. J Clin Epidemiol 2006;59(10): 1087–91.
19. Li J, Cheng K, Wang S, et al. Feature selection: A data perspective. ACM Comput Surv (Csur) 2017; 50(6):1–45.
20. Dash M, Liu H. Feature selection for classification. Intell Data Anal 1997;1(3):131–56.
21. Haixiang G, Yijing L, Shang J, et al. Learning from class-imbalanced data: Review of methods and applications. Expert Syst Appl 2017;73:220–39.
22. Kaur H, Pannu HS, Malhi AK. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. ACM Comput Surv (Csur) 2019;52(4):1–36.
23. Cawley GC, Talbot NL. On over-fitting in model selection and subsequent selection bias in performance evaluation. J Mach Learn Res 2010;11: 2079–107.
24. Cortes C, Vapnik V. Support-vector networks. Machine Learn 1995;20(3):273–97.
25. Noble WS. What is a support vector machine? Nat Biotechnol 2006;24(12):1565–7.
26. Berry MW, Kogan J. Text mining: applications and theory. West Sussex (UK): John Wiley & Sons; 2010.
27. Breiman L. Random forests. Mach Learn 2001;45(1): 5–32.
28. Polikar R. Ensemble based systems in decision making. IEEE Circuits and Systems Magazine 2006;6(3):21–45.
29. Herrera F, Charte F, Rivera AJ, et al. Ensemble-based classifiers. In: Herrera F, Charte F, Rivera AJ, et al, editors. Multilabel classification : problem analysis, metrics and techniques. Cham (Switzerland): Springer International Publishing; 2016. p. 101–13.
30. Biau G, Scornet E. A random forest guided tour. TEST 2016;25(2):197–227.
31. Yu Y, Si X, Hu C, et al. A review of recurrent neural networks: LSTM cells and network architectures. Neural Comput 2019;31(7):1235–70.
32. Yoo H-J. Deep convolution neural networks in computer vision: a review. IEIE Transactions on Smart Processing & Computing 2015;4(1):35–43.
33. Murphy KP. Machine learning: a probabilistic perspective. Cambridge (MA): MIT Press; 2012.
34. Arbelaitz O, Gurrutxaga I, Muguerza J, et al. An extensive comparative study of cluster validity indices. Pattern Recogn 2013;46(1):243–56.