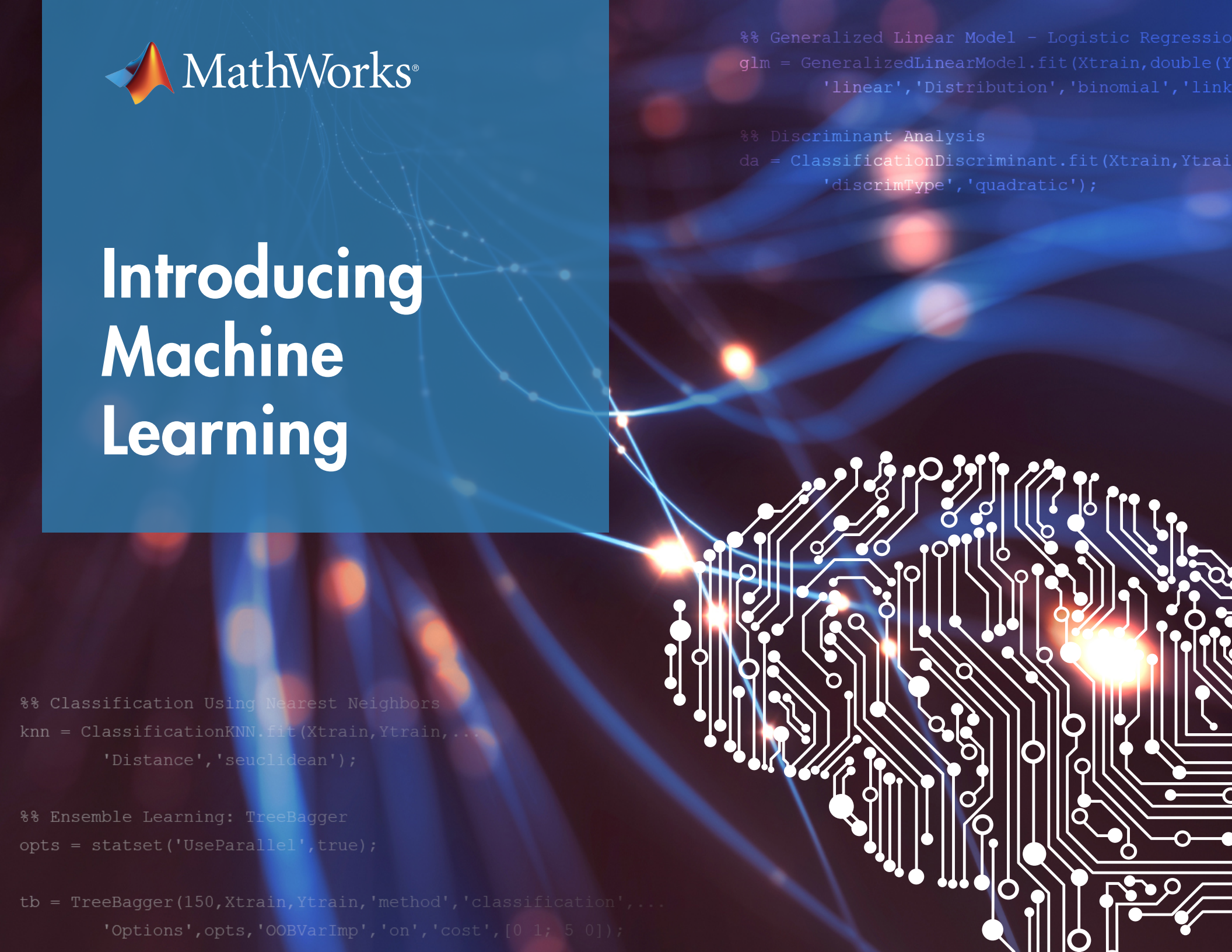
MathWorks®

Introducing Machine Learning



What is Machine Learning?

Machine learning teaches computers to do what comes naturally to humans and animals: learn from experience. Machine learning algorithms use computational methods to "learn" information directly from data without relying on a predetermined equation as a model. The algorithms adaptively improve their performance as the number of samples available for learning increases.

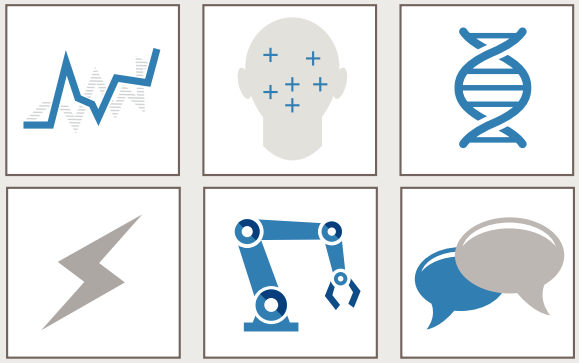
More Data, More Questions, Better Answers

Machine learning algorithms find natural patterns in data that generate insight and help you make better decisions and predictions. They are used every day to make critical decisions in medical diagnosis, stock trading, energy load forecasting, and more. Media sites rely on machine learning to sift through millions of options to give you song or movie recommendations. Retailers use it to gain insight into their customers' purchasing behavior.

Real-World Applications

With the rise in big data, machine learning has become particularly important for solving problems in areas like these:

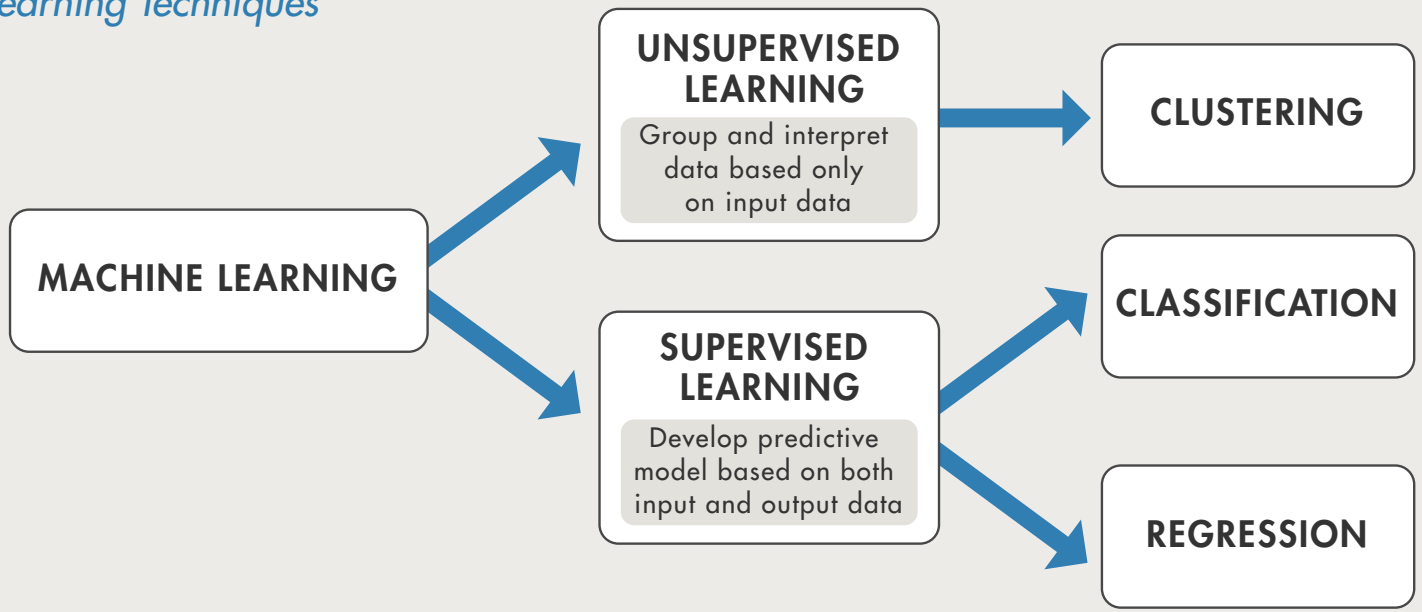
- Computational finance, for credit scoring and algorithmic trading
- Image processing and computer vision, for face recognition, motion detection, and object detection
- Computational biology, for tumor detection, drug discovery, and DNA sequencing
- Energy production, for price and load forecasting
- Automotive, aerospace, and manufacturing, for predictive maintenance
- Natural language processing



How Machine Learning Works

Machine learning uses two types of techniques: supervised learning, which trains a model on known input and output data so that it can predict future outputs, and unsupervised learning, which finds hidden patterns or intrinsic structures in input data.

Machine Learning Techniques



Supervised Learning

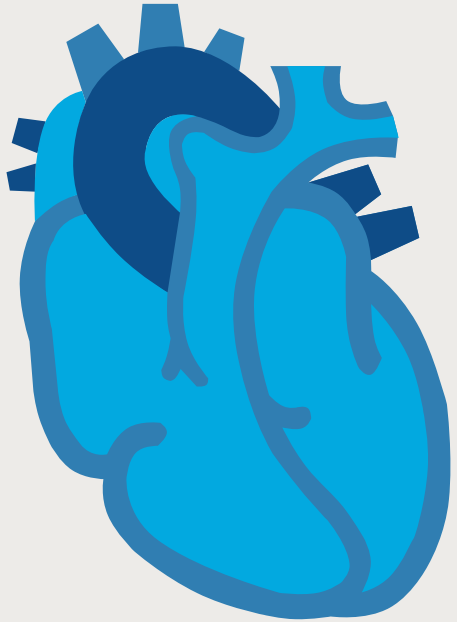
The aim of supervised machine learning is to build a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data.

Supervised learning uses classification and regression techniques to develop predictive models.

- **Classification techniques** predict discrete responses—for example, whether an email is genuine or spam, or whether a tumor is cancerous or benign. Classification models classify input data into categories. Typical applications include medical imaging, speech recognition, and credit scoring.
- **Regression techniques** predict continuous responses—for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading.

Using Supervised Learning to Predict Heart Attacks

Suppose clinicians want to predict whether someone will have a heart attack within a year. They have data on previous patients, including age, weight, height, and blood pressure. They know whether the previous patients had heart attacks within a year. So the problem is combining the existing data into a model that can predict whether a new person will have a heart attack within a year.

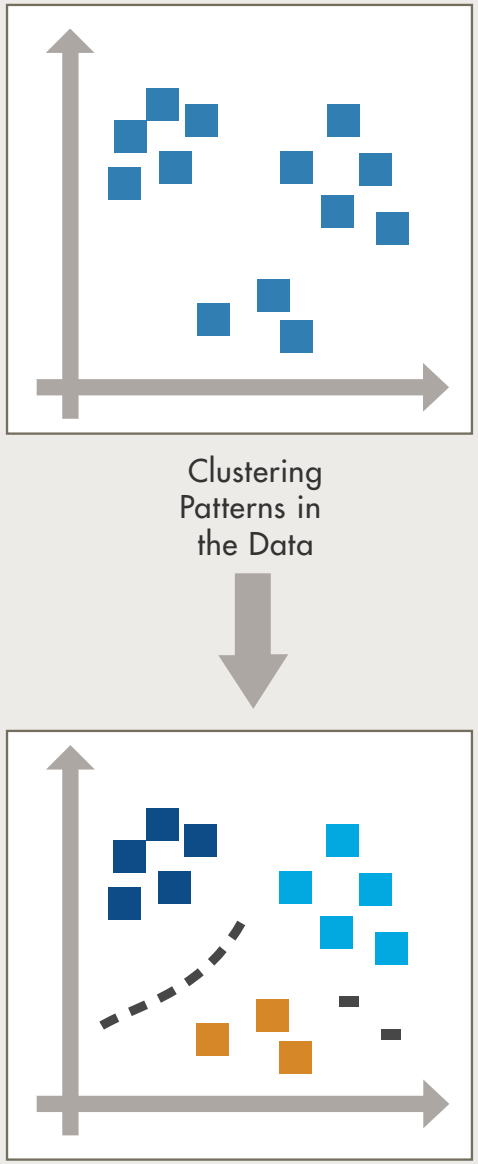


Unsupervised Learning

Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labeled responses.

Clustering is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data.

Applications for clustering include gene sequence analysis, market research, and object recognition.

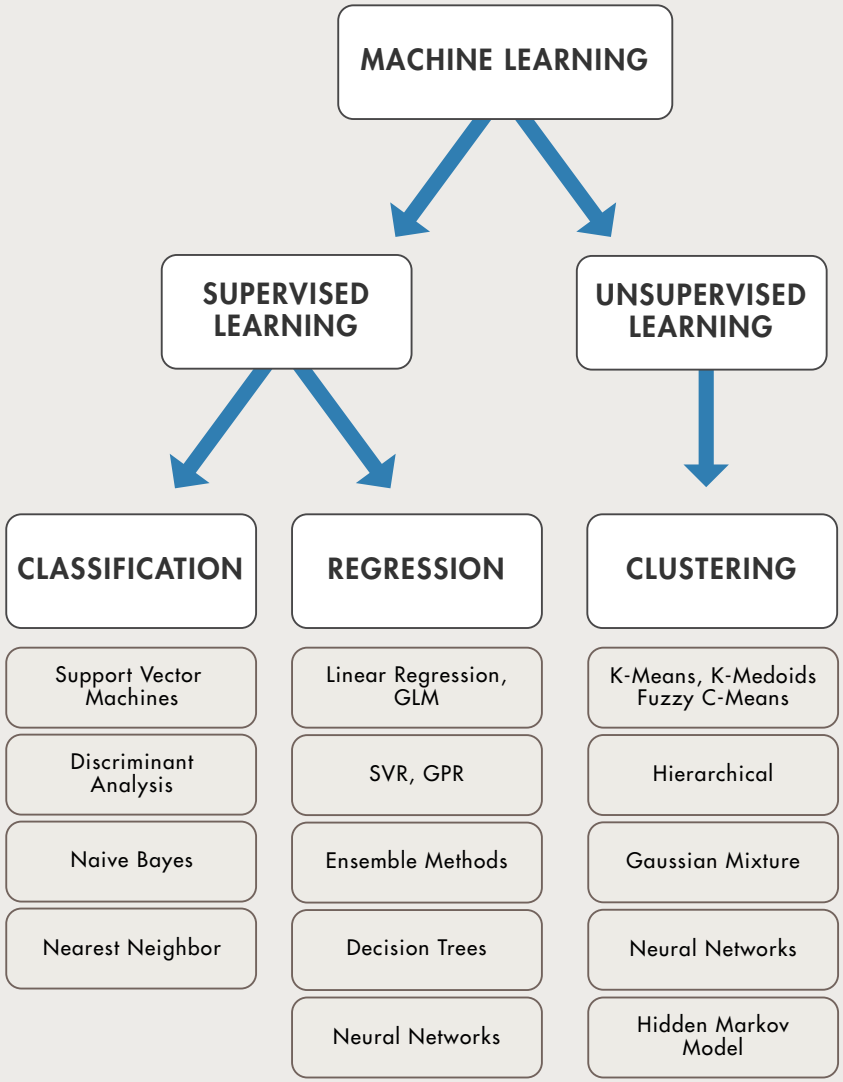


How Do You Decide Which Algorithm to Use?

Choosing the right algorithm can seem overwhelming—there are dozens of supervised and unsupervised machine learning algorithms, and each takes a different approach to learning.

There is no best method or one size fits all. Finding the right algorithm is partly just trial and error—even highly experienced data scientists can't tell whether an algorithm will work without trying it out. But algorithm selection also depends on the size and type of data you're working with, the insights you want to get from the data, and how those insights will be used.

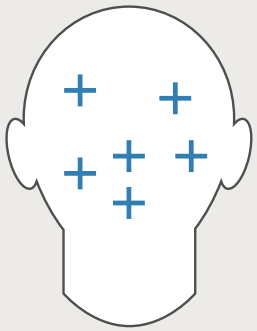
Selecting an Algorithm



When Should You Use Machine Learning?

Consider using machine learning when you have a complex task or problem involving a large amount of data and lots of variables, but no existing formula or equation. For example, machine learning is a good option if you need to handle situations like these:

Hand-written rules and equations are too complex—as in face recognition and speech recognition.



The rules of a task are constantly changing—as in fraud detection from transaction records.

The nature of the data keeps changing, and the program needs to adapt—as in automated trading, energy demand forecasting, and predicting shopping trends.

Real-World Examples

Creating Algorithms that Can Analyze Works of Art

Researchers at the Art and Artificial Intelligence Laboratory at Rutgers University wanted to see whether a computer algorithm could classify paintings by style, genre, and artist as easily as a human. They began by identifying visual features for classifying a painting's style. The algorithms they developed classified the styles of paintings in the database with 60% accuracy, outperforming typical non-expert humans.

The researchers hypothesized that visual features useful for style classification (a supervised learning problem) could also be used to determine artistic influences (an unsupervised problem).

They used classification algorithms trained on Google images to identify specific objects. They tested the algorithms on more than 1,700 paintings from 66 different artists working over a span of 550 years. The algorithm readily identified connected works, including the influence of Diego Velazquez's "Portrait of Pope Innocent X" on Francis Bacon's "Study After Velazquez's Portrait of Pope Innocent X."

Real-World Examples

Optimizing HVAC Energy Usage in Large Buildings

The heating, ventilation, and air-conditioning (HVAC) systems in office buildings, hospitals, and other large-scale commercial buildings are often inefficient because they do not take into account changing weather patterns, variable energy costs, or the building's thermal properties.

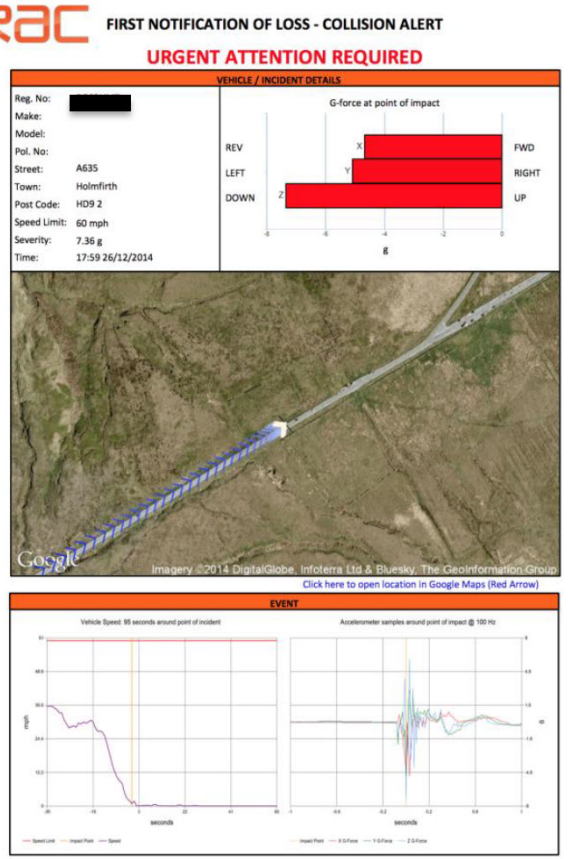
Building IQ's cloud-based software platform addresses this problem. The platform uses advanced algorithms and machine learning methods to continuously process gigabytes of information from power meters, thermometers, and HVAC pressure sensors, as well as weather and energy cost. In particular, machine learning is used to segment data and determine the relative contributions of gas, electric, steam, and solar power to heating and cooling processes. The building IQ platform reduces HVAC energy consumption in large-scale commercial buildings by 10% - 25% during normal operation.

Real-World Examples

Detecting Low-Speed Car Crashes

With more than 8 million members, the RAC is one of the UK's largest motoring organizations, providing roadside assistance, insurance, and other services to private and business motorists.

To enable rapid response to roadside incidents, reduce crashes, and mitigate insurance costs, the RAC developed an onboard crash sensing system that uses advanced machine learning algorithms to detect low-speed collisions and distinguish these events from more common driving events, such as driving over speed bumps or potholes. Independent tests showed the RAC system to be 92% accurate in detecting test crashes.



Learn More

Ready for a deeper dive? Explore these resources to learn more about machine learning methods, examples, and tools.

Watch

Machine Learning Made Easy 34:34

Signal Processing and Machine Learning Techniques for Sensor Data Analytics 42:45

Read

Machine Learning Blog Posts: Social Network Analysis, Text Mining, Bayesian Reasoning, and more

The Netflix Prize and Production Machine Learning Systems: An Insider Look

Machine Learning Challenges: Choosing the Best Model and Avoiding Overfitting

Explore

MATLAB Machine Learning Examples

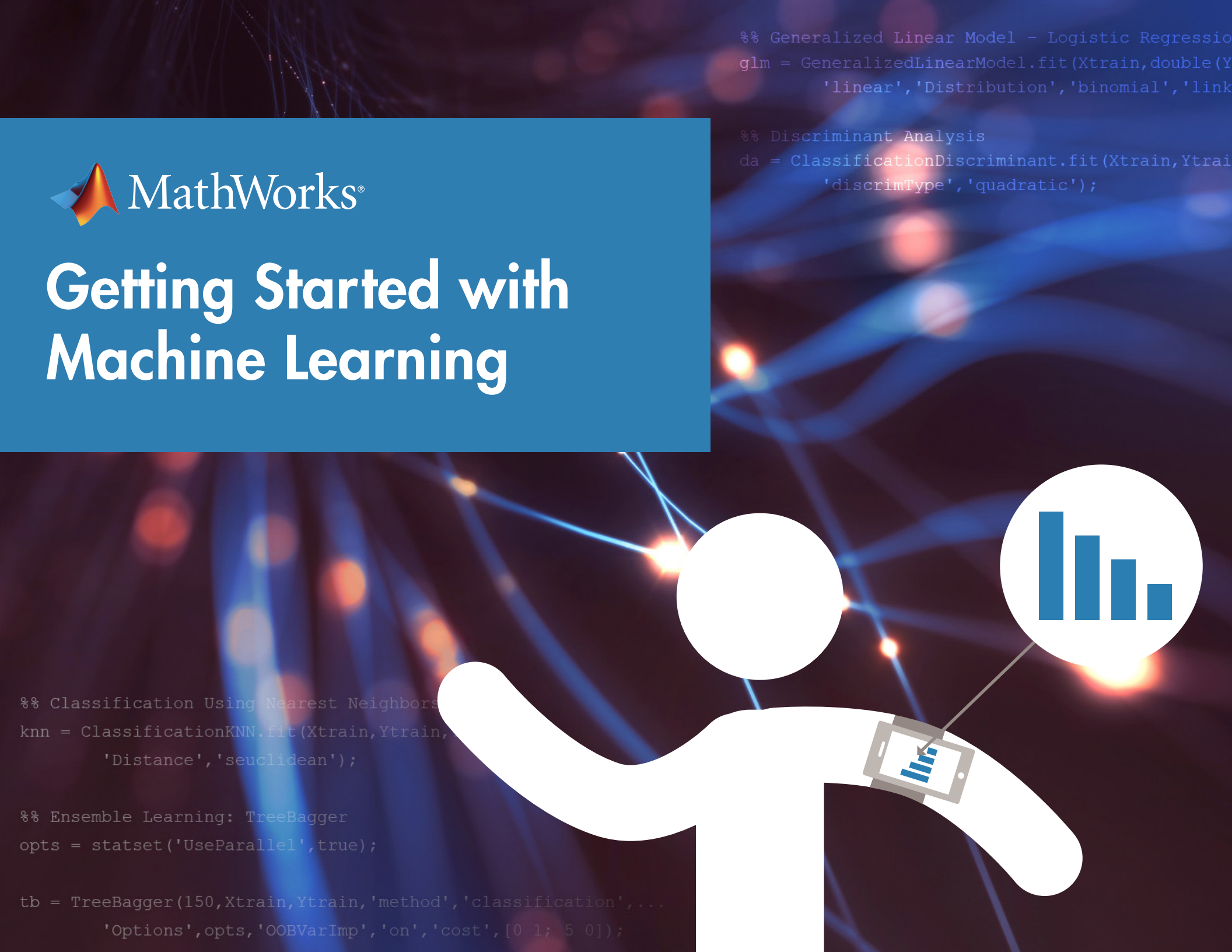
Machine Learning Solutions

Classify Data with the Classification Learner App

92991v00

MathWorks®

Getting Started with Machine Learning



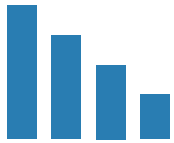
Rarely a Straight Line

With machine learning there's rarely a straight line from start to finish—you'll find yourself constantly iterating and trying different ideas and approaches. This chapter describes a systematic machine learning workflow, highlighting some key decision points along the way.

Machine Learning Challenges

Most machine learning challenges relate to handling your data and finding the right model.

Data comes in all shapes and sizes. Real-world datasets can be messy, incomplete, and in a variety of formats. You might just have simple numeric data. But sometimes you're combining several different data types, such as sensor signals, text, and streaming images from a camera.



Preprocessing your data might require specialized knowledge and tools. For example, to select features to train an object detection algorithm requires specialized knowledge of image processing. Different types of data require different approaches to preprocessing.

It takes time to find the best model to fit the data. Choosing the right model is a balancing act. Highly flexible models tend to overfit data by modeling minor variations that could be noise. On the other hand, simple models may assume too much. There are always tradeoffs between model speed, accuracy, and complexity.

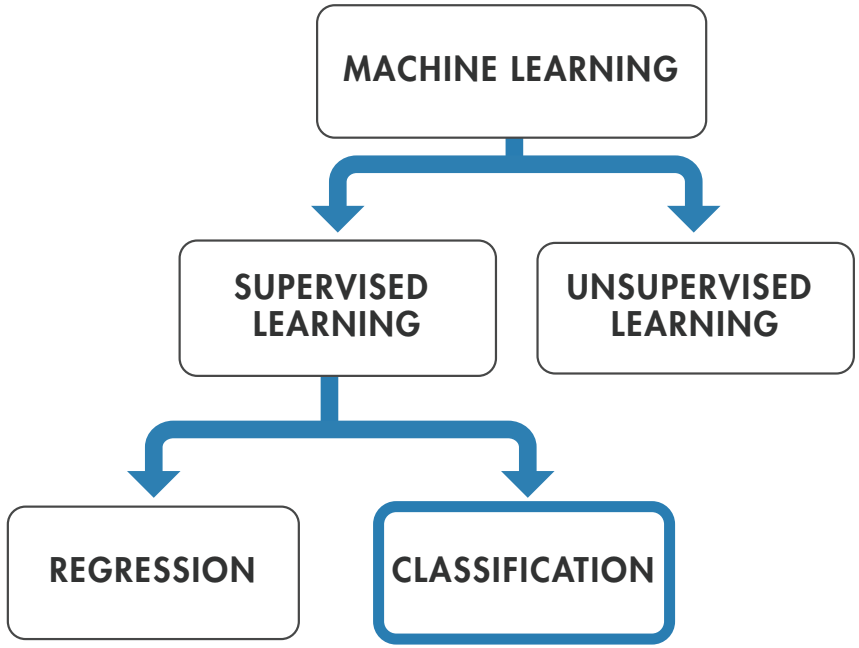
Sounds daunting? Don't be discouraged. Remember that trial and error is at the core of machine learning—if one approach or algorithm doesn't work, you simply try another. But a systematic workflow will help you get off to a smooth start.

Questions to Consider Before You Start

Every machine learning workflow begins with three questions:

- What kind of data are you working with?
- What insights do you want to get from it?
- How and where will those insights be applied?

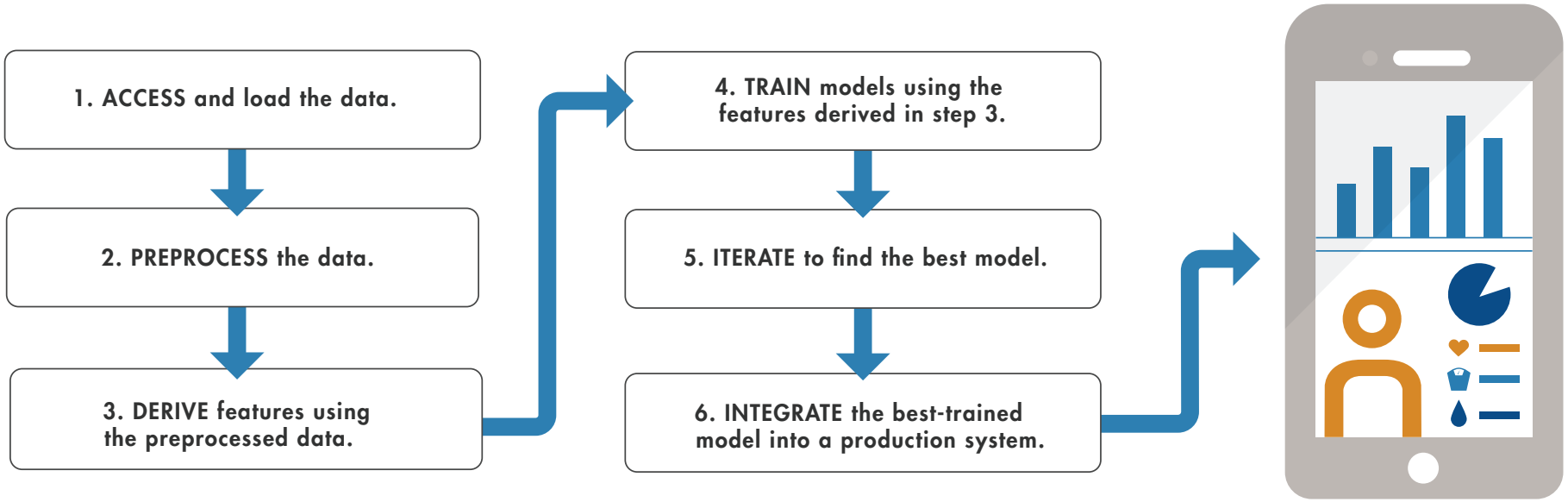
Your answers to these questions help you decide whether to use supervised or unsupervised learning.



Choose supervised learning if you need to train a model to make a prediction--for example, the future value of a continuous variable, such as temperature or a stock price, or a classification—for example, identify makes of cars from webcam video footage.

Choose unsupervised learning if you need to explore your data and want to train a model to find a good internal representation, such as splitting data up into clusters.

Workflow at a Glance



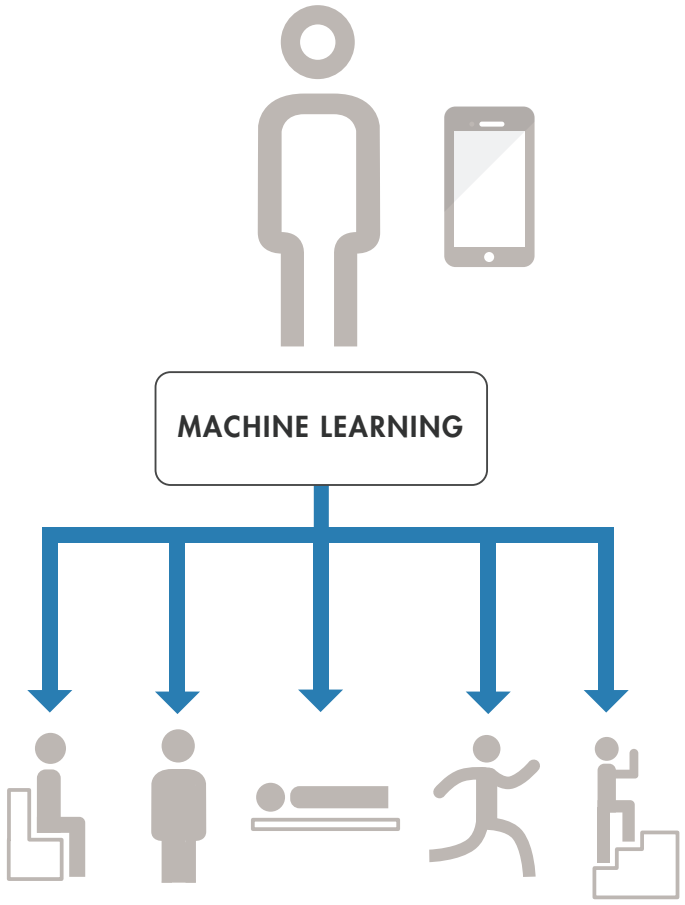
In the next sections we'll look at the steps in more detail, using a health monitoring app for illustration. The entire workflow will be completed in MATLAB®.

Training a Model to Classify Physical Activities

This example is based on a cell phone health-monitoring app. The input consists of three-axial sensor data from the phone's accelerometer and gyroscope. The responses, (or output), are the activities performed–walking, standing, running, climbing stairs, or lying down.

We want to use the input data to train a classification model to identify these activities. Since our goal is classification, we'll be applying supervised learning.

The trained model (or classifier) will be integrated into an app to help users track their activity levels throughout the day.



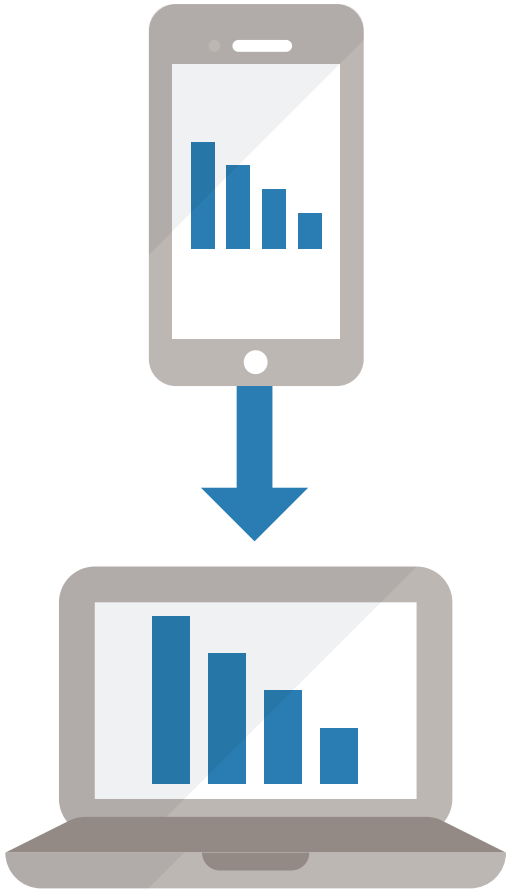
1 Step One: Load the Data

To load data from the accelerometer and gyroscope we do the following:

1. Sit down holding the phone, log data from the phone, and store it in a text file labeled "Sitting."
2. Stand up holding the phone, log data from the phone, and store it in a second text file labeled "Standing."
3. Repeat the steps until we have data for each activity we want to classify.

We store the labeled data sets in a text file. A flat file format such as text or CSV is easy to work with and makes it straightforward to import data.

Machine learning algorithms aren't smart enough to tell the difference between noise and valuable information. Before using the data for training, we need to make sure it's clean and complete.



2 Step Two: Preprocess the Data

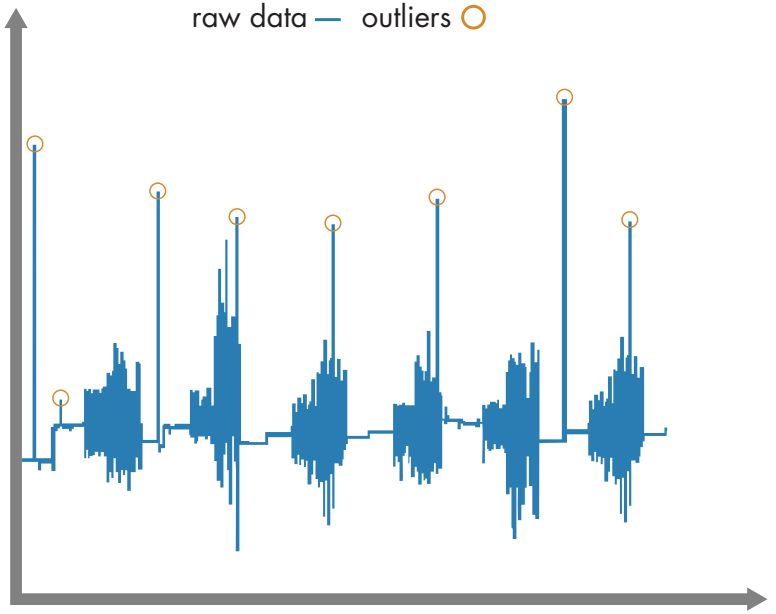
We import the data into MATLAB and plot each labeled set. To preprocess the data we do the following:

1. Look for outliers–data points that lie outside the rest of the data.

We must decide whether the outliers can be ignored or whether they indicate a phenomenon that the model should account for. In our example, they can safely be ignored (it turns out that we moved unintentionally while recording the data).

2. Check for missing values (perhaps we lost data because the connection dropped during recording).

We could simply ignore the missing values, but this will reduce the size of the data set. Alternatively, we could substitute approximations for the missing values by interpolating or using comparable data from another sample.



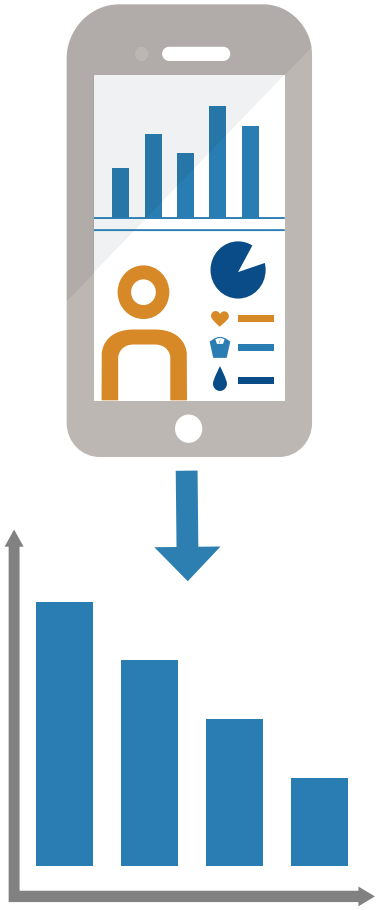
Outliers in the activity-tracking data.

In many applications, outliers provide crucial information. For example, in a credit card fraud detection app, they indicate purchases that fall outside a customer's usual buying patterns.

2 Step Two: Preprocess the Data *continued*

3. Remove gravitational effects from the accelerometer data so that our algorithm will focus on the movement of the subject, not the movement of the phone. A simple high-pass filter such as a biquad filter is commonly used for this.
4. Divide the data into two sets. We save part of the data for testing (the test set) and use the rest (the training set) to build models. This is referred to as holdout, and is a useful cross-validation technique.

By testing your model against data that wasn't used in the modeling process, you see how it will perform with unknown data.



3 Step Three: Derive Features

Deriving features (also known as feature engineering or feature extraction) is one of the most important parts of machine learning. It turns raw data into information that a machine learning algorithm can use.

For the activity tracker, we want to extract features that capture the frequency content of the accelerometer data. These features will help the algorithm distinguish between walking (low frequency) and running (high frequency). We create a new table that includes the selected features.

Use feature selection to:

- Improve the accuracy of a machine learning algorithm
- Boost model performance for high-dimensional data sets
- Improve model interpretability
- Prevent overfitting

3 Step Three: Derive Features *continued*

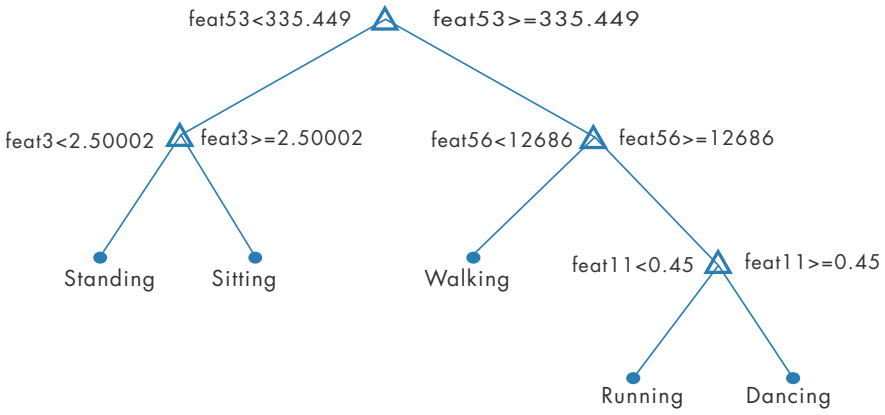
The number of features that you could derive is limited only by your imagination. However, there are a lot of techniques commonly used for different types of data.

Data Type	Feature Selection Task	Techniques
Sensor data	Extract signal properties from raw sensor data to create higher-level information	**Peak analysis** – perform an fft and identify dominant frequencies **Pulse and transition metrics** – derive signal characteristics such as rise time, fall time, and settling time **Spectral measurements** – plot signal power, bandwidth, mean frequency, and median frequency
Image and video data	Extract features such as edge locations, resolution, and color	**Bag of visual words** – create a histogram of local image features, such as edges, corners, and blobs **Histogram of oriented gradients (HOG)** – create a histogram of local gradient directions **Minimum eigenvalue algorithm** – detect corner locations in images **Edge detection** – identify points where the degree of brightness changes sharply
Transactional data	Calculate derived values that enhance the information in the data	**Timestamp decomposition** – break timestamps down into components such as day and month **Aggregate value calculation** – create higher-level features such as the total number of times a particular event occurred

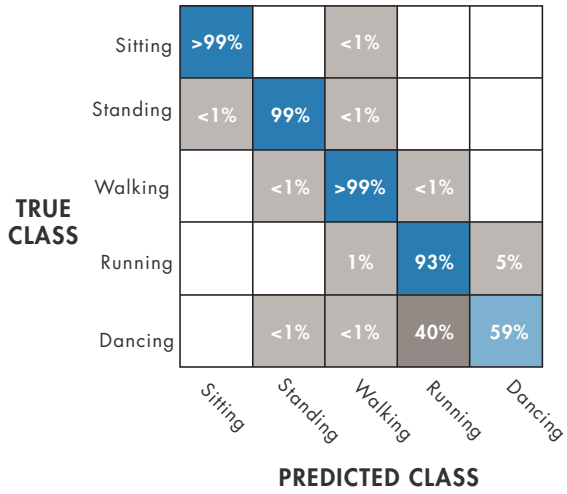
4 Step Four: Build and Train the Model

When building a model, it's a good idea to start with something simple; it will be faster to run and easier to interpret.

We start with a basic decision tree.



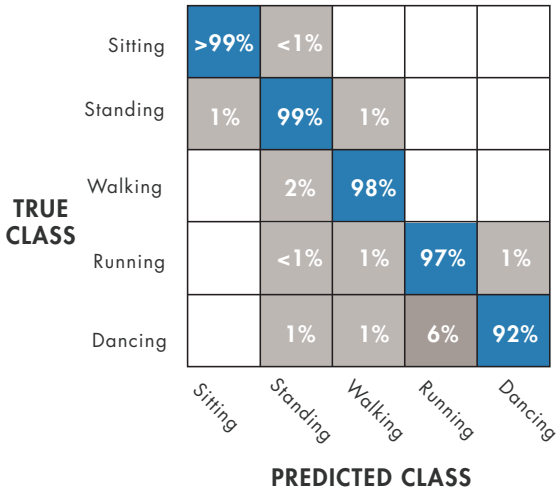
To see how well it performs, we plot the confusion matrix, a table that compares the classifications made by the model with the actual class labels that we created in step 1.



The confusion matrix shows that our model is having trouble distinguishing between dancing and running. Maybe a decision tree doesn't work for this type of data. We'll try a few different algorithms.

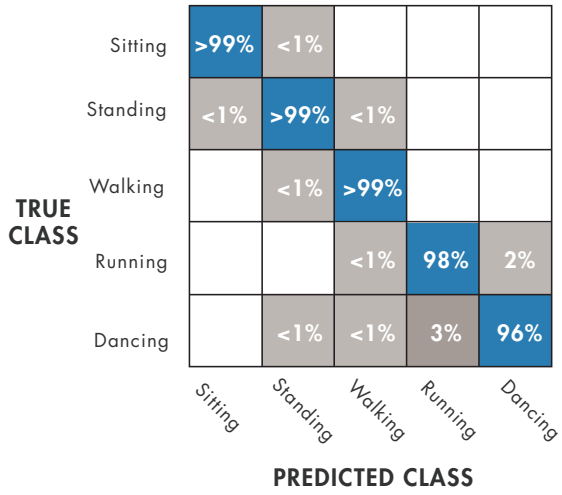
4 Step Four: Build and Train the Model *continued*

We start with a K-nearest neighbors (KNN), a simple algorithm that stores all the training data, compares new points to the training data, and returns the most frequent class of the "K" nearest points. That gives us 98% accuracy compared to 94.1% for the simple decision tree. The confusion matrix looks better, too:



However, KNNs take a considerable amount of memory to run, since they require all the training data to make a prediction.

We try a linear discriminant model, but that doesn't improve the results. Finally, we try a multiclass support vector machine (SVM). The SVM does very well—we now get 99% accuracy:



We achieved our goal by iterating on the model and trying different algorithms. If our classifier still couldn't reliably differentiate between dancing and running, we'd look into ways to improve the model.

5 Step Five: Improve the Model

Improving a model can take two different directions: make the model simpler or add complexity.

Simplify

First, we look for opportunities to reduce the number of features. Popular feature reduction techniques include:

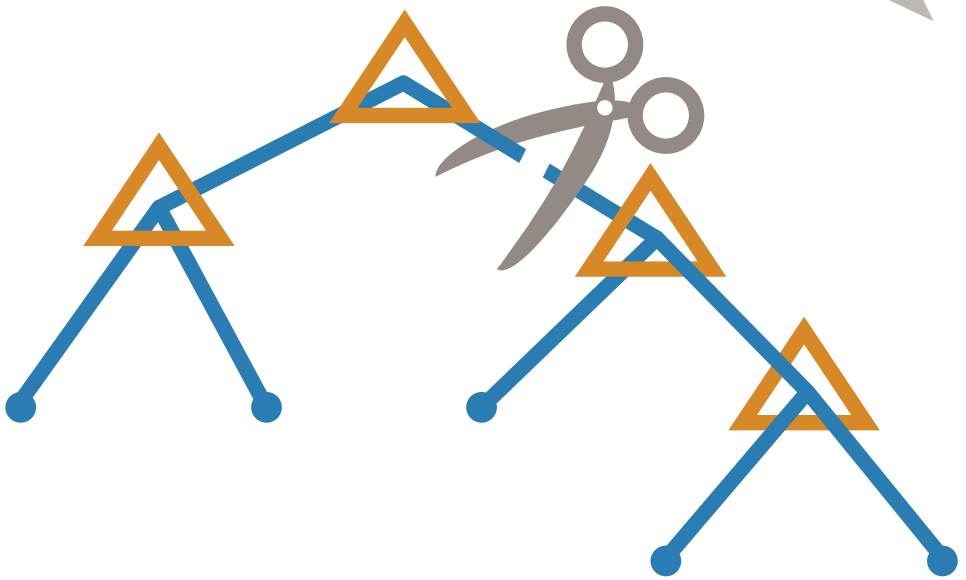
- Correlation matrix – shows the relationship between variables, so that variables (or features) that are not highly correlated can be removed.
- Principal component analysis (PCA) – eliminates redundancy by finding a combination of features that captures key distinctions between the original features and brings out strong patterns in the dataset.
- Sequential feature reduction – reduces features iteratively on the model until there is no improvement in performance.

Next, we look at ways to reduce the model itself. We can do this by:

- Pruning branches from a decision tree
- Removing learners from an ensemble

A good model includes only the features with the most predictive power. A simple model that generalizes well is better than a complex model that may not generalize or train well to new data.

In machine learning, as in many other computational processes, simplifying the model makes it easier to understand, more robust, and more computationally efficient.



5 Step Five: Improve the Model *continued*

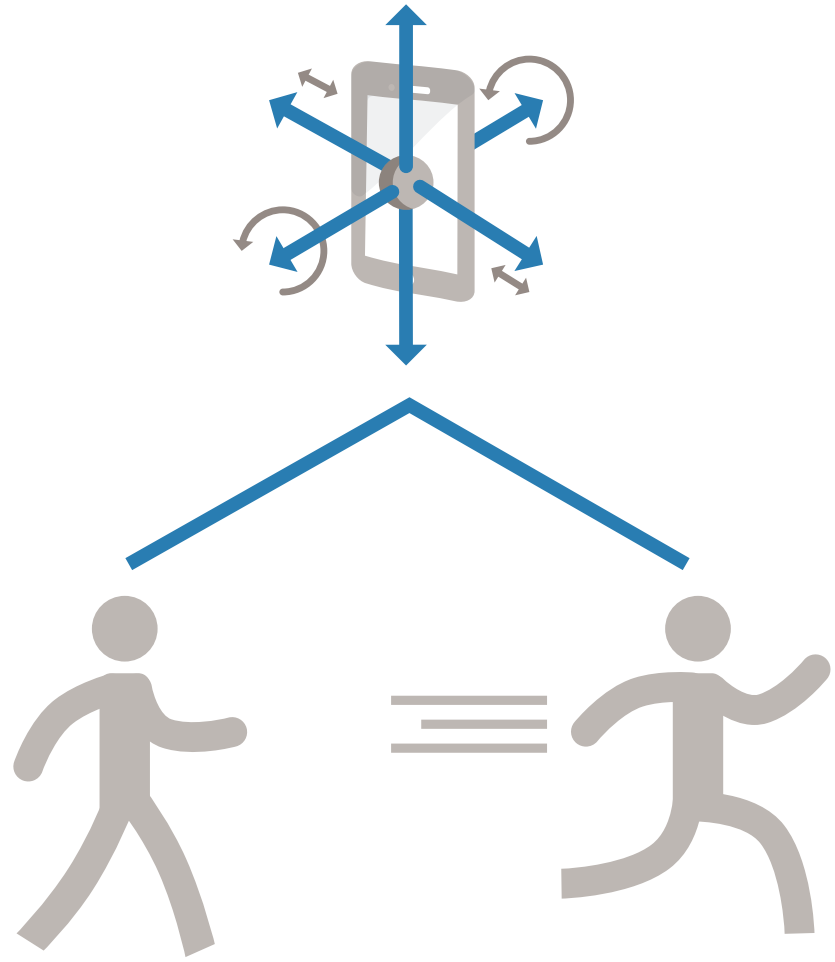
Add Complexity

If our model can't differentiate dancing from running because it is over-generalizing, then we need find ways to make it more fine-tuned. To do this we can either:

- Use model combination – merge multiple simpler models into a larger model that is better able to represent the trends in the data than any of the simpler models could on their own.
- Add more data sources – look at the gyroscope data as well as the acceleromter data. The gyroscope records the orientation of the cell phone during activity. This data might provide unique signatures for the different activities; for example, there might be a combination of acceleration and rotation that's unique to running.

Once we've adjusted the model, we validate its performance on the test data that we set aside during preprocessing.

If the model can reliably classify activities on the test data, we're ready to move it to the phone and start tracking.



Learn More

Ready for a deeper dive? Explore these resources to learn more about machine learning methods, examples, and tools.

Watch

Machine Learning Made Easy 34:34

Signal Processing and Machine Learning Techniques for Sensor Data Analytics 42:45

Read

Supervised Learning Workflow and Algorithms

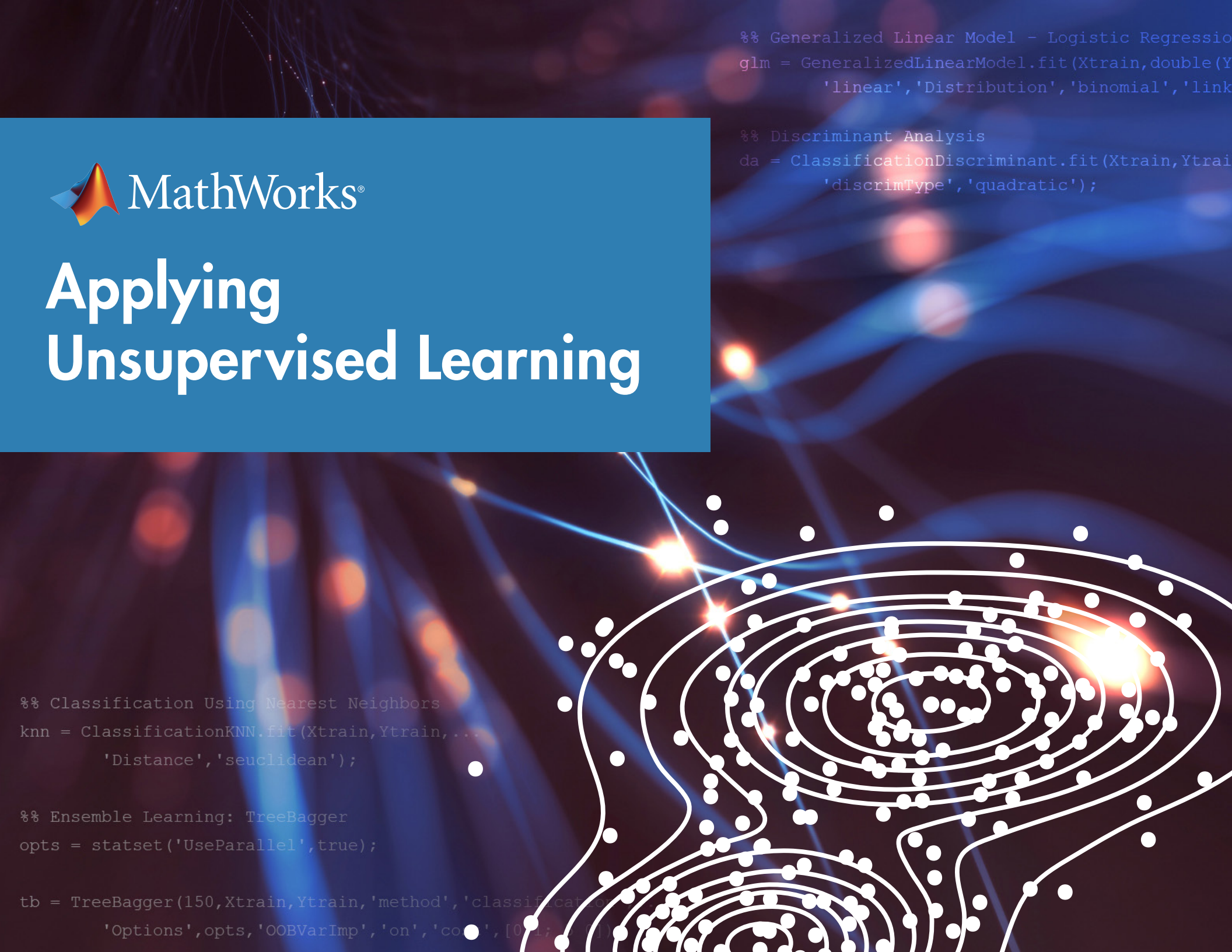
Data-Driven Insights with MATLAB Analytics: An Energy Load Forecasting Case Study

Explore

MATLAB Machine Learning Examples

Classify Data with the Classification Learner App

93014v00


MathWorks®
Applying
Unsupervised Learning
%% Generalized Linear Model - Logistic Regressio
glm = GeneralizedLinearModel.fit(Xtrain,double(Y
'linear','Distribution','binomial','link
%% Discriminant Analysis
da = ClassificationDiscriminant.fit(Xtrain,Ytrai
'discrimType','quadratic');
%% Classification Using Nearest Neighbors
knn = ClassificationKNN.fit(Xtrain,Ytrain,...
'Distance','seuclidean');
%% Ensemble Learning: TreeBagger
opts = statset('UseParallel',true);
tb = TreeBagger(150,Xtrain,Ytrain,'method','classificat
'Options',opts,'OOBVarImp','on','co

MathWorks

When to Consider Unsupervised Learning

Unsupervised learning is useful when you want to explore your data but don't yet have a specific goal or are not sure what information the data contains. It's also a good way to reduce the dimensions of your data.

Unsupervised Learning Techniques

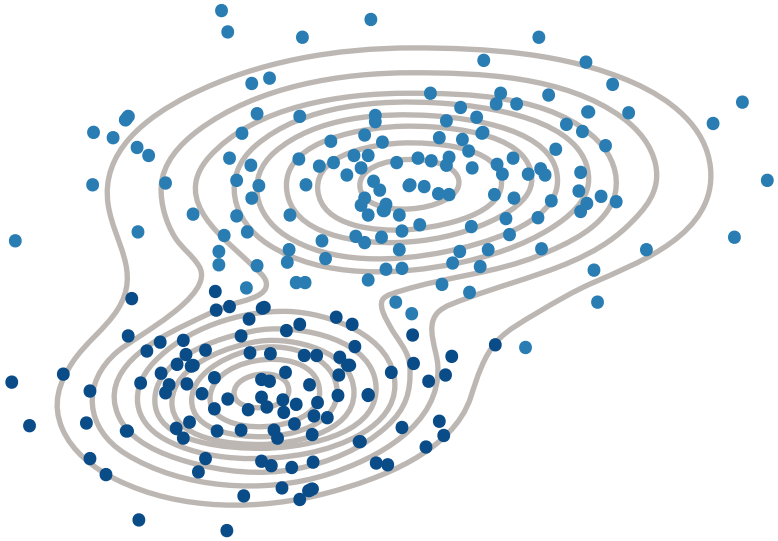
As we saw in section 1, most unsupervised learning techniques are a form of cluster analysis.

In cluster analysis, data is partitioned into groups based on some measure of similarity or shared characteristic. Clusters are formed so that objects in the same cluster are very similar and objects in different clusters are very distinct.

Clustering algorithms fall into two broad groups:

- Hard clustering, where each data point belongs to only one cluster
- Soft clustering, where each data point can belong to more than one cluster

You can use hard or soft clustering techniques if you already know the possible data groupings.



Gaussian mixture model used to separate data into two clusters.

If you don't yet know how the data might be grouped:

- Use self-organizing feature maps or hierarchical clustering to look for possible structures in the data.
- Use cluster evaluation to look for the "best" number of groups for a given clustering algorithm.

Common Hard Clustering Algorithms

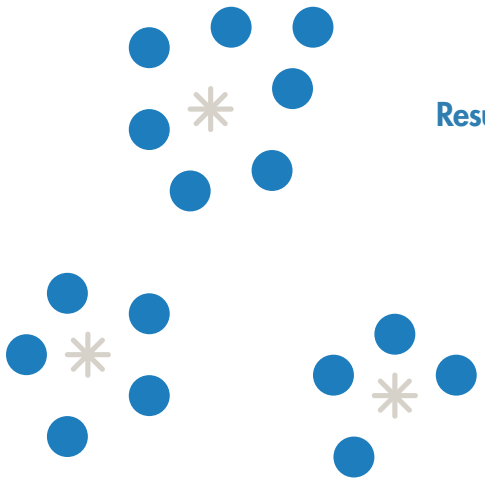
k-Means

How it Works

Partitions data into k number of mutually exclusive clusters. How well a point fits into a cluster is determined by the distance from that point to the cluster's center.

Best Used...

- When the number of clusters is known
- For fast clustering of large data sets



Result: Cluster centers

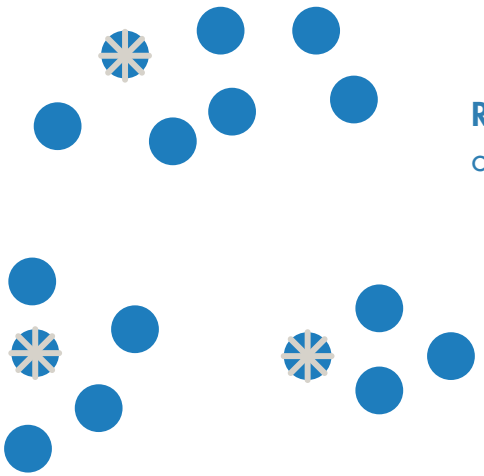
k-Medoids

How It Works

Similar to k-means, but with the requirement that the cluster centers coincide with points in the data.

Best Used...

- When the number of clusters is known
- For fast clustering of categorical data
- To scale to large data sets



Result: Cluster centers that coincide with data points

Common Hard Clustering Algorithms *continued*

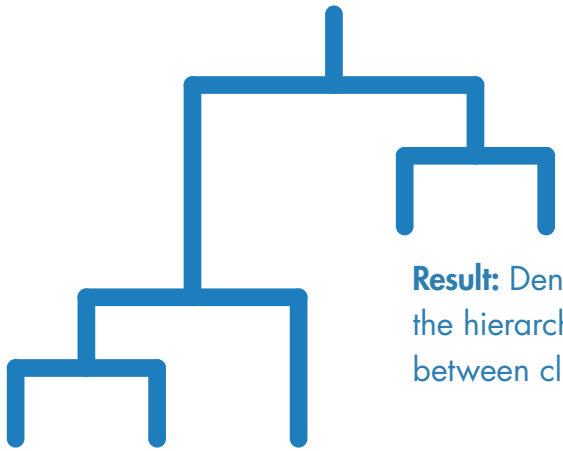
Hierarchical Clustering

How it Works

Produces nested sets of clusters by analyzing similarities between pairs of points and grouping objects into a binary, hierarchical tree.

Best Used...

- When you don't know in advance how many clusters are in your data
- You want visualization to guide your selection



Result: Dendrogram showing the hierarchical relationship between clusters

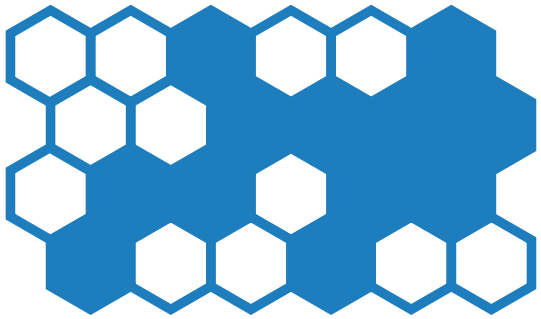
Self-Organizing Map

How It Works

Neural-network based clustering that transforms a dataset into a topology-preserving 2D map.

Best Used...

- To visualize high-dimensional data in 2D or 3D
- To deduce the dimensionality of data by preserving its topology (shape)



Result: Lower-dimensional (typically 2D) representation

Common Hard Clustering Algorithms *continued*

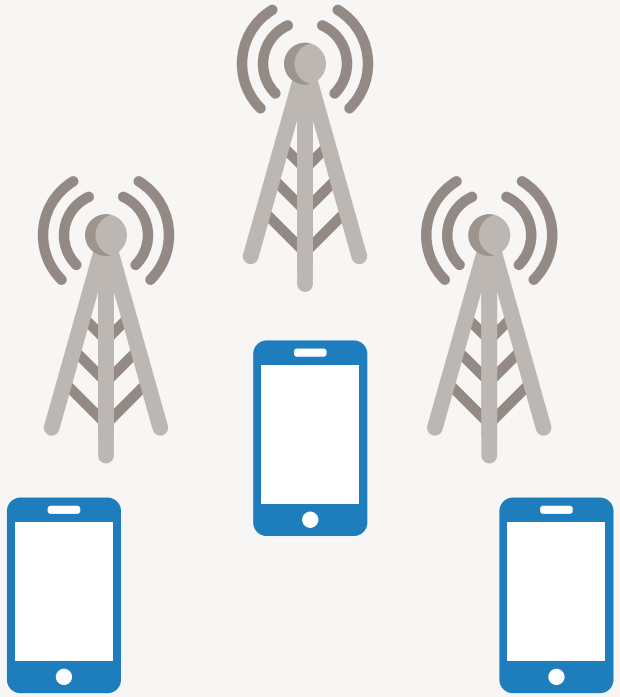
Example: Using k-Means Clustering to Site Cell Phone Towers

A cell phone company wants to know the number and placement of cell phone towers that will provide the most reliable service. For optimal signal reception, the towers must be located within clusters of people.

The workflow begins with an initial guess at the number of clusters that will be needed. To evaluate this guess, the engineers compare service with three towers and four towers to see how well they're able to cluster for each scenario (in other words, how well the towers provide service).

A phone can only talk to one tower at a time, so this is a hard clustering problem. The team uses k-means clustering because k-means treats each observation in the data as an object having a location in space. It finds a partition in which objects within each cluster are as close to each other as possible and as far from objects in other clusters as possible.

After running the algorithm, the team can accurately determine the results of partitioning the data into three and four clusters.



Common Soft Clustering Algorithms

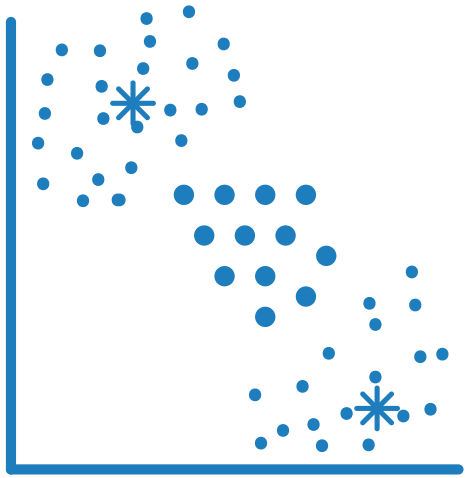
Fuzzy c-Means

How it Works

Partition-based clustering when data points may belong to more than one cluster.

Best Used...

- When the number of clusters is known
- For pattern recognition
- When clusters overlap



Result: Cluster centers (similar to k-means) but with fuzziness so that points may belong to more than one cluster

Gaussian Mixture Model

How It Works

Partition-based clustering where data points come from different multivariate normal distributions with certain probabilities.

Best Used...

- When a data point might belong to more than one cluster
- When clusters have different sizes and correlation structures within them

Result: A model of Gaussian distributions that give probabilities of a point being in a cluster

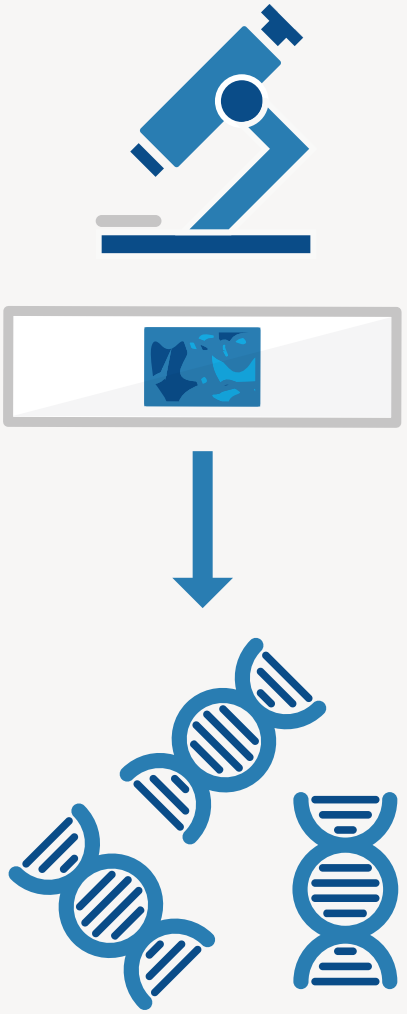
Common Soft Clustering Algorithms *continued*

Example: Using Fuzzy c-Means Clustering to Analyze Gene Expression Data

A team of biologists is analyzing gene expression data from microarrays to better understand the genes involved in normal and abnormal cell division. (A gene is said to be "expressed" if it is actively involved in a cellular function such as protein production.)

The microarray contains expression data from two tissue samples. The researchers want to compare the samples to determine whether certain patterns of gene expression are implicated in cancer proliferation.

After preprocessing the data to remove noise, they cluster the data. Because the same genes can be involved in several biological processes, no single gene is likely to belong to one cluster only. The researchers apply a fuzzy c-means algorithm to the data. They then visualize the clusters to identify groups of genes that behave in a similar way.



Improving Models with Dimensionality Reduction

Machine learning is an effective method for finding patterns in big datasets. But bigger data brings added complexity.

As datasets get bigger, you frequently need to reduce the number of features, or *dimensionality.*

Example: EEG Data Reduction

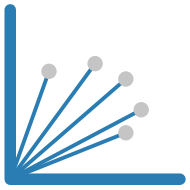
Suppose you have electroencephalogram (EEG) data that captures electrical activity of the brain, and you want to use this data to predict a future seizure. The data was captured using dozens of leads, each corresponding to a variable in your original dataset. Each of these variables contains noise. To make your prediction algorithm more robust, you use dimensionality reduction techniques to derive a smaller number of features. Because these features are calculated from multiple sensors, they will be less susceptible to noise in an individual sensor than would be the case if you used the raw data directly.

Common Dimensionality Reduction Techniques

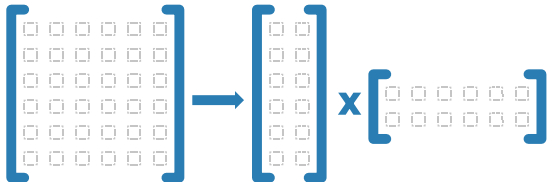
The three most commonly used dimensionality reduction techniques are:

Principal component analysis (PCA)—performs a linear transformation on the data so that most of the variance or information in your high-dimensional dataset is captured by the first few principal components. The first principal component will capture the most variance, followed by the second principal component, and so on.

Factor analysis—identifies underlying correlations between variables in your dataset to provide a representation in terms of a smaller number of unobserved latent, or common, factors.



Nonnegative matrix factorization—used when model terms must represent nonnegative quantities, such as physical quantities.



Using Principal Component Analysis

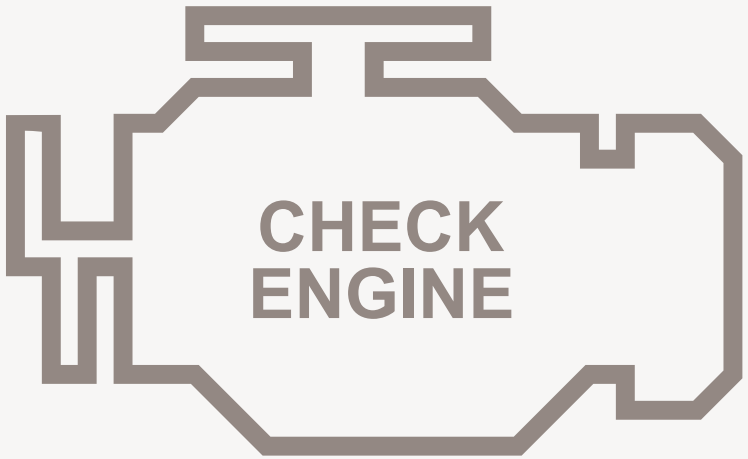
In datasets with many variables, groups of variables often move together. PCA takes advantage of this redundancy of information by generating new variables via linear combinations of the original variables so that a small number of new variables captures most of the information.

Each principal component is a linear combination of the original variables. Because all the principal components are orthogonal to each other, there is no redundant information.

Example: Engine Health Monitoring

You have a dataset that includes measurements for different sensors on an engine (temperatures, pressures, emissions, and so on). While much of the data comes from a healthy engine, the sensors have also captured data from the engine when it needs maintenance.

You cannot see any obvious abnormalities by looking at any individual sensor. However, by applying PCA, you can transform this data so that most variations in the sensor measurements are captured by a small number of principal components. It is easier to distinguish between a healthy and unhealthy engine by inspecting these principal components than by looking at the raw sensor data.



Using Factor Analysis

Your dataset might contain measured variables that overlap, meaning that they are dependent on one another. Factor analysis lets you fit a model to multivariate data to estimate this sort of interdependence.

In a factor analysis model, the measured variables depend on a smaller number of unobserved (latent) factors. Because each factor might affect several variables, it is known as a common factor. Each variable is assumed to be dependent on a linear combination of the common factors.

Example: Tracking Stock Price Variation

Over the course of 100 weeks, the percent change in stock prices has been recorded for ten companies. Of these ten, four are technology companies, three are financial, and a further three are retail. It seems reasonable to assume that the stock prices for companies in the same sector will vary together as economic conditions change. Factor analysis can provide quantitative evidence to support this premise.

Using Nonnegative Matrix Factorization

This dimension reduction technique is based on a low-rank approximation of the feature space. In addition to reducing the number of features, it guarantees that the features are nonnegative, producing models that respect features such as the nonnegativity of physical quantities.

Example: Text Mining

Suppose you want to explore variations in vocabulary and style among several web pages. You create a matrix where each row corresponds to an individual web page and each column corresponds to a word ("the","a","we", and so on). The data will be the number of times a particular word occurs on a particular page.

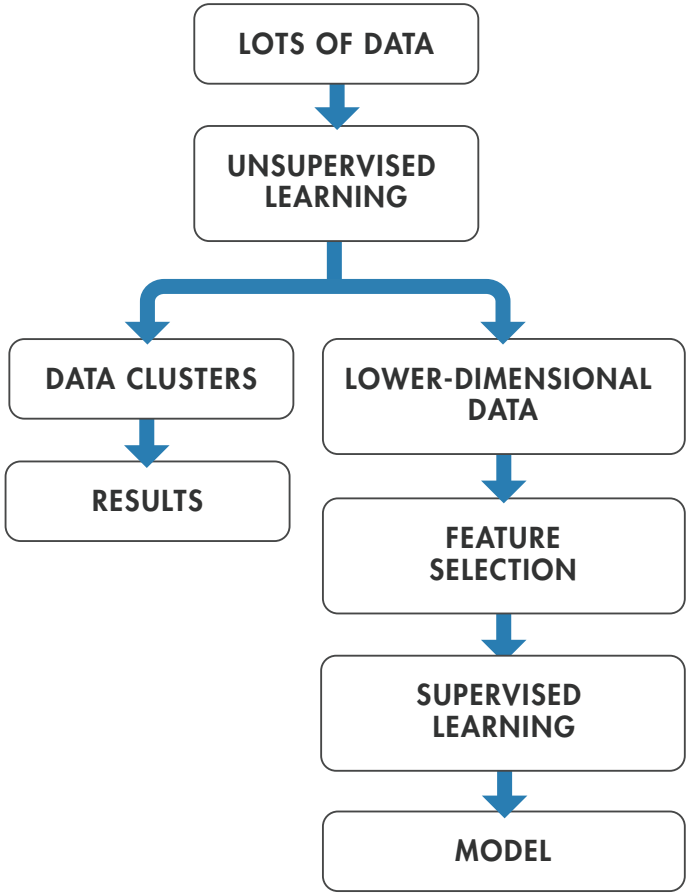
Since there more than a million words in the English language, you apply nonnegative matrix factorization to create an arbitrary number of features that represent higher-level concepts rather than individual words. These concepts make it easier to distinguish between, say, news, educational content, and online retail content.

Next Steps

In this section we took a closer look at hard and soft clustering algorithms for unsupervised learning, offered some tips on selecting the right algorithm for your data, and showed how reducing the number of features in your dataset improves model performance. As for your next steps:

- Unsupervised learning might be your end goal. For example, if you are doing market research and want to segment consumer groups to target based on web site behavior, a clustering algorithm will almost certainly give you the results you're looking for.
- On the other hand, you might want to use unsupervised learning as a preprocessing step for supervised learning. For example, apply clustering techniques to derive a smaller number of features, and then use those features as inputs for training a classifier.

In section 4 we'll explore supervised learning algorithms and techniques, and see how to improve models with feature selection, feature reduction, and parameter tuning.



Learn More

Ready for a deeper dive? Explore these unsupervised learning resources.

Clustering Algorithms and Techniques

k-Means

Use *K*-Means and Hierarchical Clustering to Find Natural Patterns in Data

Cluster Genes Using K-Means and Self-Organizing Maps

Color-Based Segmentation Using K-Means Clustering

Hierarchical Clustering

Connectivity-Based Clustering

Iris Clustering

Self-Organizing Maps

Cluster Data with a Self-Organizing Map

Fuzzy C-Means

Cluster Quasi-Random Data Using Fuzzy C-Means Clustering

Gaussian Mixture Models

Gaussian Process Regression Models

Cluster Data from Mixture of Gaussian Distributions

Cluster Gaussian Mixture Data Using Soft Clustering

Tune Gaussian Mixture Models

Image Processing Example: Detecting Cars with Gaussian Mixture Models

Dimensionality Reduction

Analyze Quality of Life in U.S. Cities Using PCA

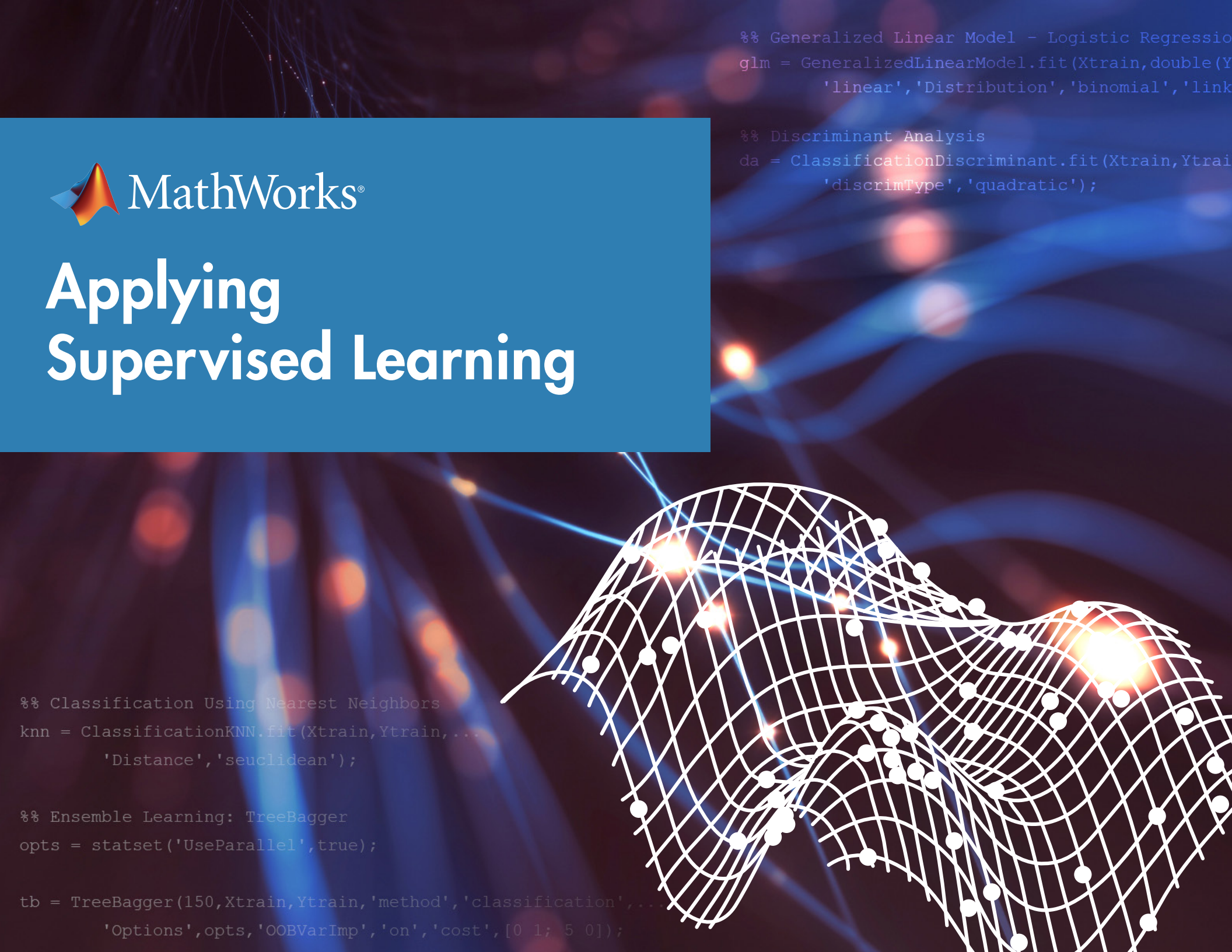
Analyze Stock Prices Using Factor Analysis

Nonnegative Factorization

Perform Nonnegative Matrix Factorization

Model Suburban Commuting Using Subtractive Clustering

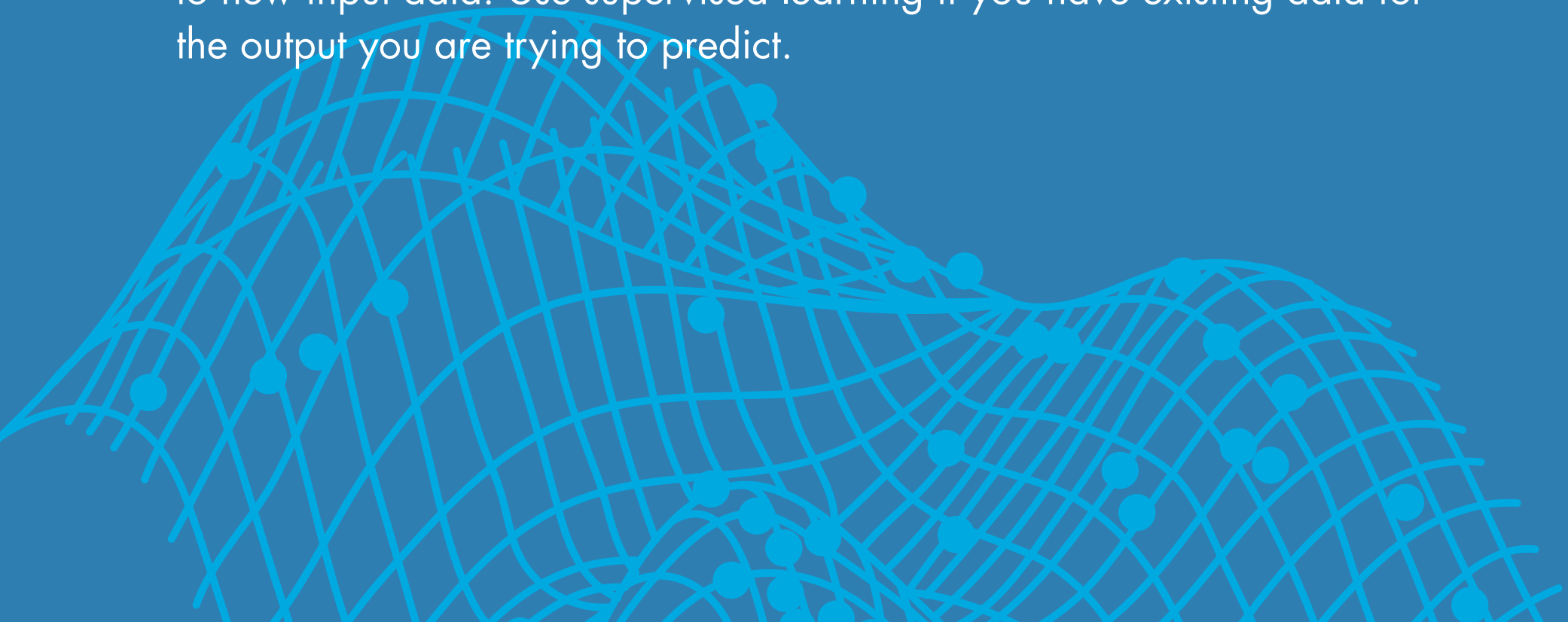
80823v00


MathWorks®
Applying
Supervised Learning
%% Generalized Linear Model - Logistic Regressio
glm = GeneralizedLinearModel.fit(Xtrain,double(Y
'linear','Distribution','binomial','link
%% Discriminant Analysis
da = ClassificationDiscriminant.fit(Xtrain,Ytrai
'discrimType','quadratic');
%% Classification Using Nearest Neighbors
knn = ClassificationKNN.fit(Xtrain,Ytrain,...
'Distance','seuclidean');
%% Ensemble Learning: TreeBagger
opts = statset('UseParallel',true);
tb = TreeBagger(150,Xtrain,Ytrain,'method','classification',...
'Options',opts,'OOBVarImp','on','cost',[0 1; 5 0]);

MathWorks®

When to Consider Supervised Learning

A supervised learning algorithm takes a known set of input data (the training set) and known responses to the data (output), and trains a model to generate reasonable predictions for the response to new input data. Use supervised learning if you have existing data for the output you are trying to predict.

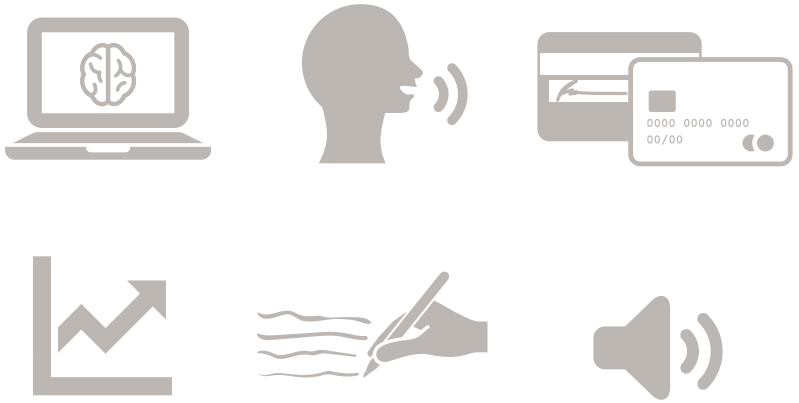


Supervised Learning Techniques

All supervised learning techniques are a form of classification or regression.

Classification techniques predict discrete responses—for example, whether an email is genuine or spam, or whether a tumor is small, medium, or large. Classification models are trained to classify data into categories. Applications include medical imaging, speech recognition, and credit scoring.

Regression techniques predict continuous responses—for example, changes in temperature or fluctuations in electricity demand. Applications include forecasting stock prices, handwriting recognition, and acoustic signal processing.



- Can your data be tagged or categorized? If your data can be separated into specific groups or classes, use classification algorithms.
- Working with a data range? If the nature of your response is a real number –such as temperature, or the time until failure for a piece of equipment--use regression techniques.

Selecting the Right Algorithm

As we saw in section 1, selecting a machine learning algorithm is a process of trial and error. It's also a trade-off between specific characteristics of the algorithms, such as:

- Speed of training
- Memory usage
- Predictive accuracy on new data
- Transparency or interpretability (how easily you can understand the reasons an algorithm makes its predictions)

Let's take a closer look at the most commonly used classification and regression algorithms.

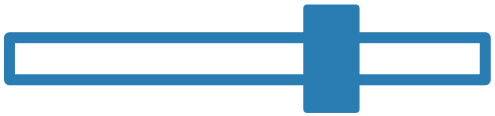
Using larger training datasets often yield models that generalize well for new data.

Speed of training

Memory usage

Predictive accuracy

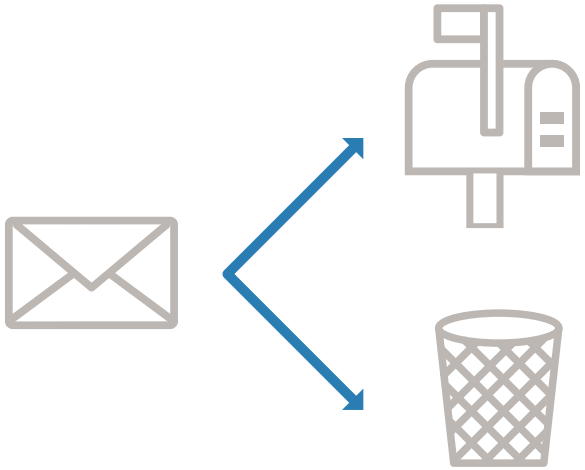
Interpretability



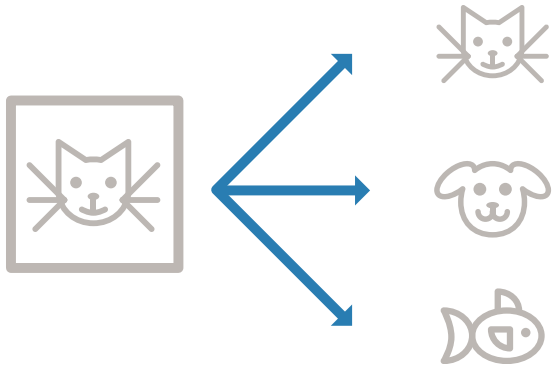
Binary vs. Multiclass Classification

When you are working on a classification problem, begin by determining whether the problem is binary or multiclass. In a binary classification problem, a single training or test item (instance) can only be divided into two classes—for example, if you want to determine whether an email is genuine or spam. In a multiclass classification problem, it can be divided into more than two—for example, if you want to train a model to classify an image as a dog, cat, or other animal.

Bear in mind that a multiclass classification problem is generally more challenging because it requires a more complex model.



Certain algorithms (for example, logistic regression) are designed specifically for binary classification problems. During training, these algorithms tend to be more efficient than multiclass algorithms.



Common Classification Algorithms

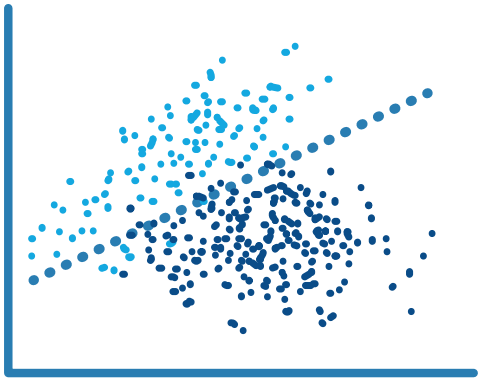
Logistic Regression

How it Works

Fits a model that can predict the probability of a binary response belonging to one class or the other. Because of its simplicity, logistic regression is commonly used as a starting point for binary classification problems.

Best Used...

- When data can be clearly separated by a single, linear boundary
- As a baseline for evaluating more complex classification methods



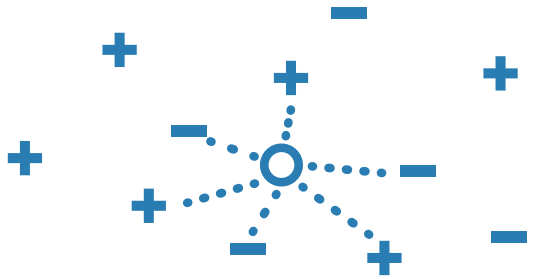
k Nearest Neighbor (kNN)

How it Works

kNN categorizes objects based on the classes of their nearest neighbors in the dataset. kNN predictions assume that objects near each other are similar. Distance metrics, such as Euclidean, city block, cosine, and Chebychev, are used to find the nearest neighbor.

Best Used...

- When you need a simple algorithm to establish benchmark learning rules
- When memory usage of the trained model is a lesser concern
- When prediction speed of the trained model is a lesser concern

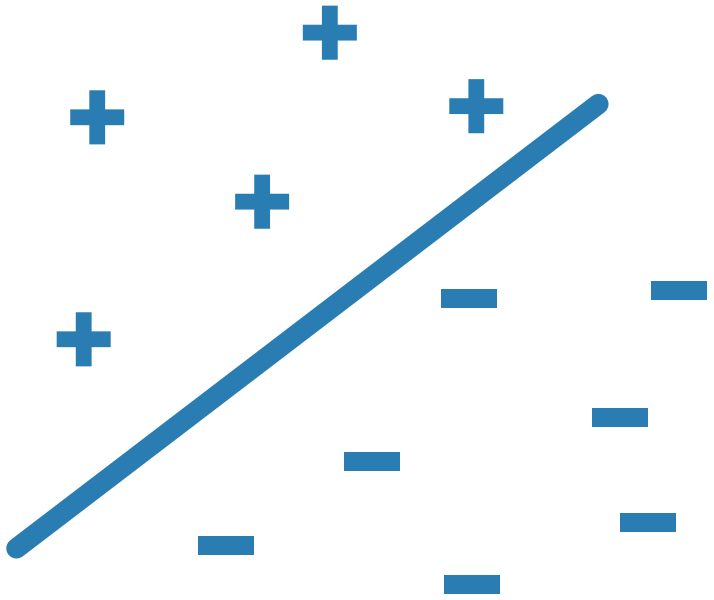


Common Classification Algorithms *continued*

Support Vector Machine (SVM)

How It Works

Classifies data by finding the linear decision boundary (hyperplane) that separates all data points of one class from those of the other class. The best hyperplane for an SVM is the one with the largest margin between the two classes, when the data is linearly separable. If the data is not linearly separable, a loss function is used to penalize points on the wrong side of the hyperplane. SVMs sometimes use a kernel transform to transform nonlinearly separable data into higher dimensions where a linear decision boundary can be found.



Best Used...

- For data that has exactly two classes (you can also use it for multiclass classification with a technique called error-correcting output codes)
- For high-dimensional, nonlinearly separable data
- When you need a classifier that's simple, easy to interpret, and accurate

Common Classification Algorithms *continued*

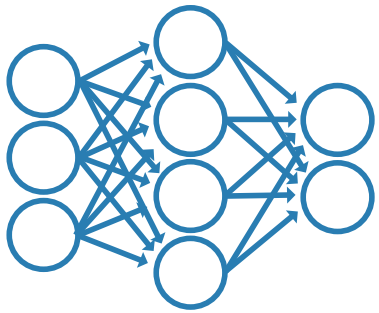
Neural Network

How it Works

Inspired by the human brain, a neural network consists of highly connected networks of neurons that relate the inputs to the desired outputs. The network is trained by iteratively modifying the strengths of the connections so that given inputs map to the correct response.

Best Used...

- For modeling highly nonlinear systems
- When data is available incrementally and you wish to constantly update the model
- When there could be unexpected changes in your input data
- When model interpretability is not a key concern



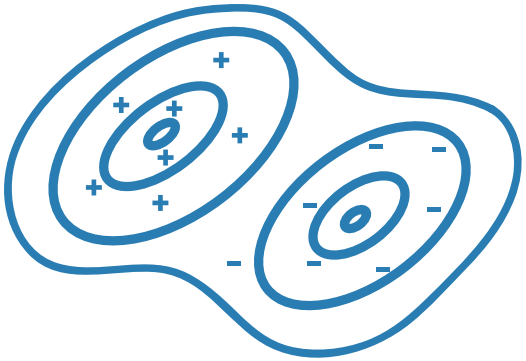
Naïve Bayes

How It Works

A naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. It classifies new data based on the highest probability of its belonging to a particular class.

Best Used...

- For a small dataset containing many parameters
- When you need a classifier that's easy to interpret
- When the model will encounter scenarios that weren't in the training data, as is the case with many financial and medical applications

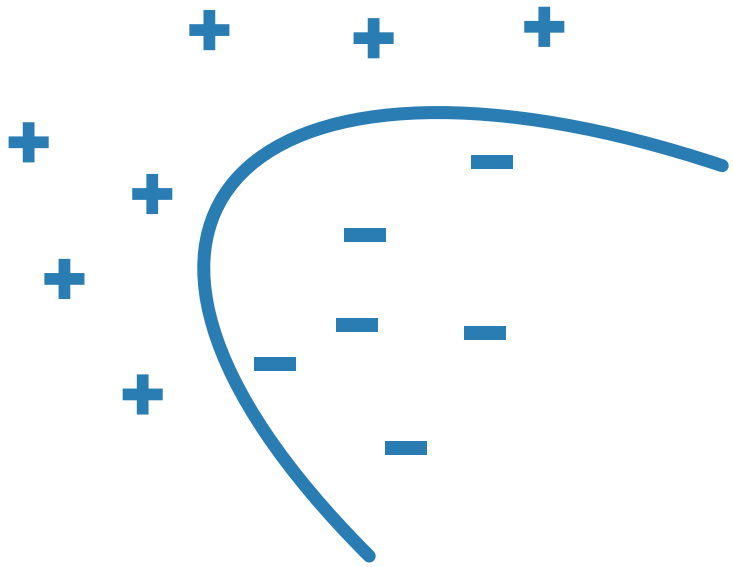


Common Classification Algorithms *continued*

Discriminant Analysis

How It Works

Discriminant analysis classifies data by finding linear combinations of features. Discriminant analysis assumes that different classes generate data based on Gaussian distributions. Training a discriminant analysis model involves finding the parameters for a Gaussian distribution for each class. The distribution parameters are used to calculate boundaries, which can be linear or quadratic functions. These boundaries are used to determine the class of new data.



Best Used...

- When you need a simple model that is easy to interpret
- When memory usage during training is a concern
- When you need a model that is fast to predict

Common Classification Algorithms *continued*

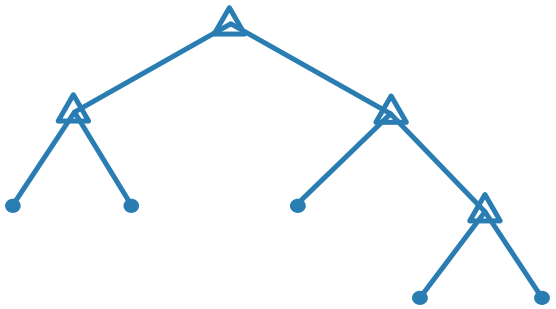
Decision Tree

How it Works

A decision tree lets you predict responses to data by following the decisions in the tree from the root (beginning) down to a leaf node. A tree consists of branching conditions where the value of a predictor is compared to a trained weight. The number of branches and the values of weights are determined in the training process. Additional modification, or pruning, may be used to simplify the model.

Best Used...

- When you need an algorithm that is easy to interpret and fast to fit
- To minimize memory usage
- When high predictive accuracy is not a requirement



Bagged and Boosted Decision Trees

How They Work

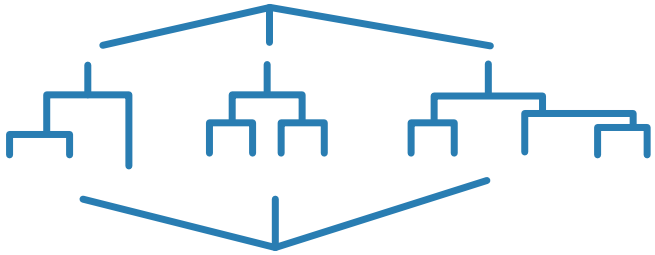
In these ensemble methods, several "weaker" decision trees are combined into a "stronger" ensemble.

A bagged decision tree consists of trees that are trained independently on data that is bootstrapped from the input data.

Boosting involves creating a strong learner by iteratively adding "weak" learners and adjusting the weight of each weak learner to focus on misclassified examples.

Best Used...

- When predictors are categorical (discrete) or behave nonlinearly
- When the time taken to train a model is less of a concern



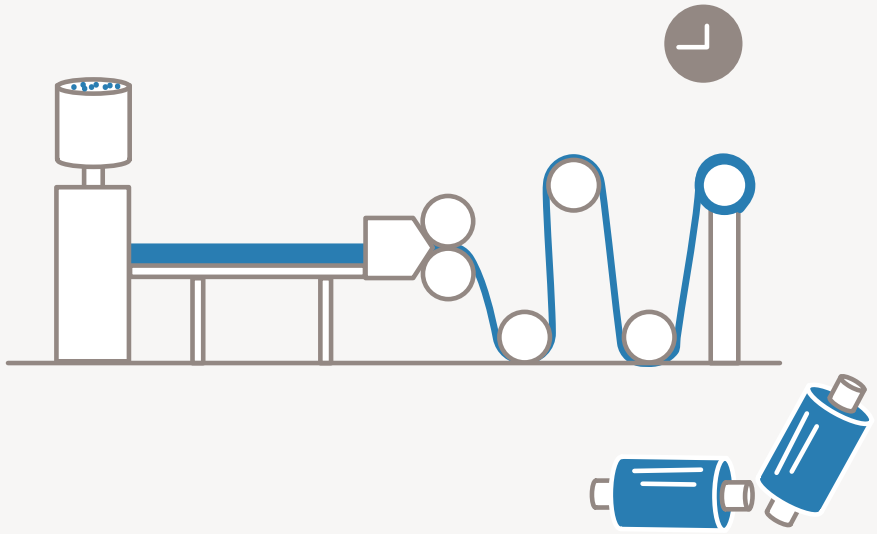
Common Classification Algorithms *continued*

Example: Predictive Maintenance for Manufacturing Equipment

A plastic production plant delivers about 18 million tons of plastic and thin film products annually. The plant's 900 workers operate 24 hours a day, 365 days a year.

To minimize machine failures and maximize plant efficiency, engineers develop a health monitoring and predictive maintenance application that uses advanced statistics and machine learning algorithms to identify potential issues with the machines so that operators can take corrective action and prevent serious problems from occurring.

After collecting, cleaning, and logging data from all the machines in the plant, the engineers evaluate several machine learning techniques, including neural networks, k-nearest neighbors, bagged decision trees, and support vector machines (SVMs). For each technique, they train a classification model using the logged machine data and then test the model's ability to predict machine problems. The tests show that an ensemble of bagged decision trees is the most accurate model for predicting the production quality.



Common Regression Algorithms

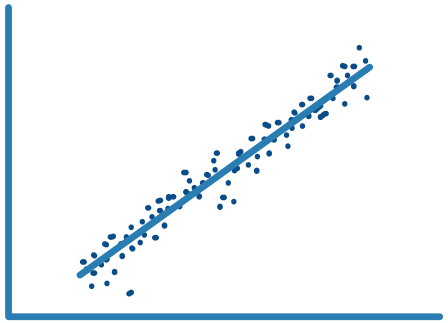
Linear Regression

How it Works

Linear regression is a statistical modeling technique used to describe a continuous response variable as a linear function of one or more predictor variables. Because linear regression models are simple to interpret and easy to train, they are often the first model to be fitted to a new dataset.

Best Used...

- When you need an algorithm that is easy to interpret and fast to fit
- As a baseline for evaluating other, more complex, regression models



Nonlinear Regression

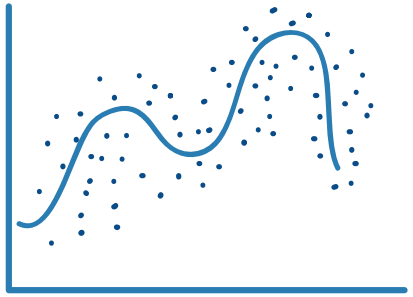
How It Works

Nonlinear regression is a statistical modeling technique that helps describe nonlinear relationships in experimental data. Nonlinear regression models are generally assumed to be parametric, where the model is described as a nonlinear equation.

"Nonlinear" refers to a fit function that is a nonlinear function of the parameters. For example, if the fitting parameters are b_0, b_1, and b_2: the equation $y = b_0+b_1x+b_2x^2$ is a linear function of the fitting parameters, whereas $y = (b_0x^{b1})/(x+b_2)$ is a nonlinear function of the fitting parameters.

Best Used...

- When data has strong nonlinear trends and cannot be easily transformed into a linear space
- For fitting custom models to data



Common Regression Algorithms *continued*

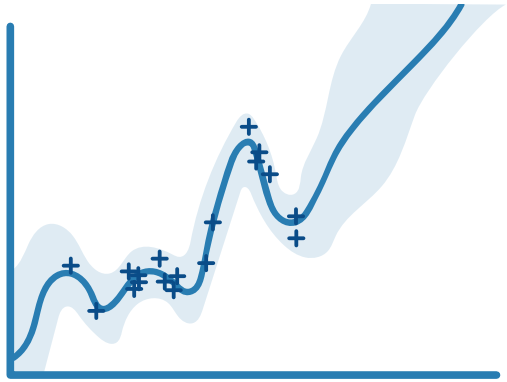
Gaussian Process Regression Model

How it Works

Gaussian process regression (GPR) models are nonparametric models that are used for predicting the value of a continuous response variable. They are widely used in the field of spatial analysis for interpolation in the presence of uncertainty. GPR is also referred to as Kriging.

Best Used...

- For interpolating spatial data, such as hydrogeological data for the distribution of ground water
- As a surrogate model to facilitate optimization of complex designs such as automotive engines



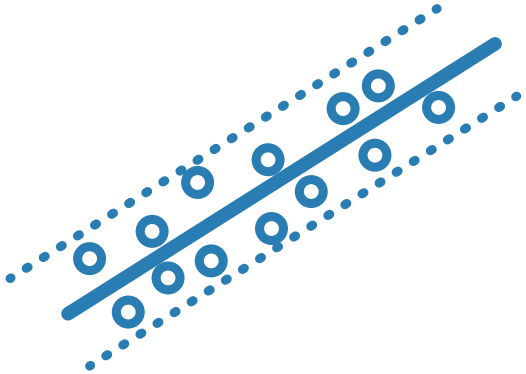
SVM Regression

How It Works

SVM regression algorithms work like SVM classification algorithms, but are modified to be able to predict a continuous response. Instead of finding a hyperplane that separates data, SVM regression algorithms find a model that deviates from the measured data by a value no greater than a small amount, with parameter values that are as small as possible (to minimize sensitivity to error).

Best Used...

- For high-dimensional data (where there will be a large number of predictor variables)



Common Regression Algorithms *continued*

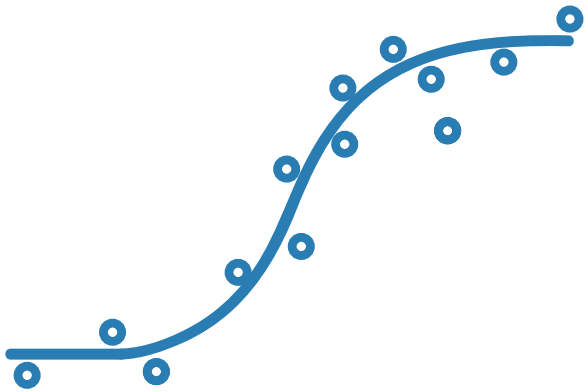
Generalized Linear Model

How it Works

A generalized linear model is a special case of nonlinear models that uses linear methods. It involves fitting a linear combination of the inputs to a nonlinear function (the link function) of the outputs.

Best Used...

- When the response variables have nonnormal distributions, such as a response variable that is always expected to be positive



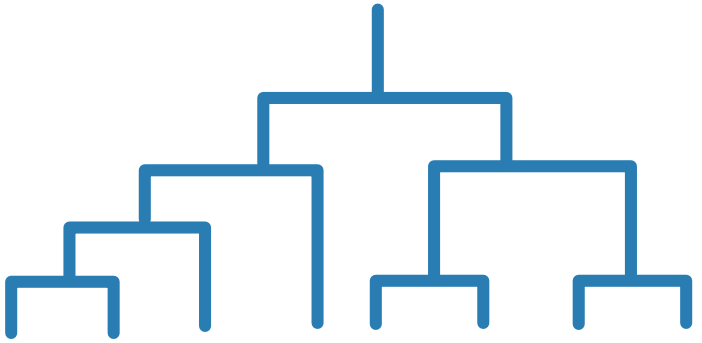
Regression Tree

How It Works

Decision trees for regression are similar to decision trees for classification, but they are modified to be able to predict continuous responses.

Best Used...

- When predictors are categorical (discrete) or behave nonlinearly

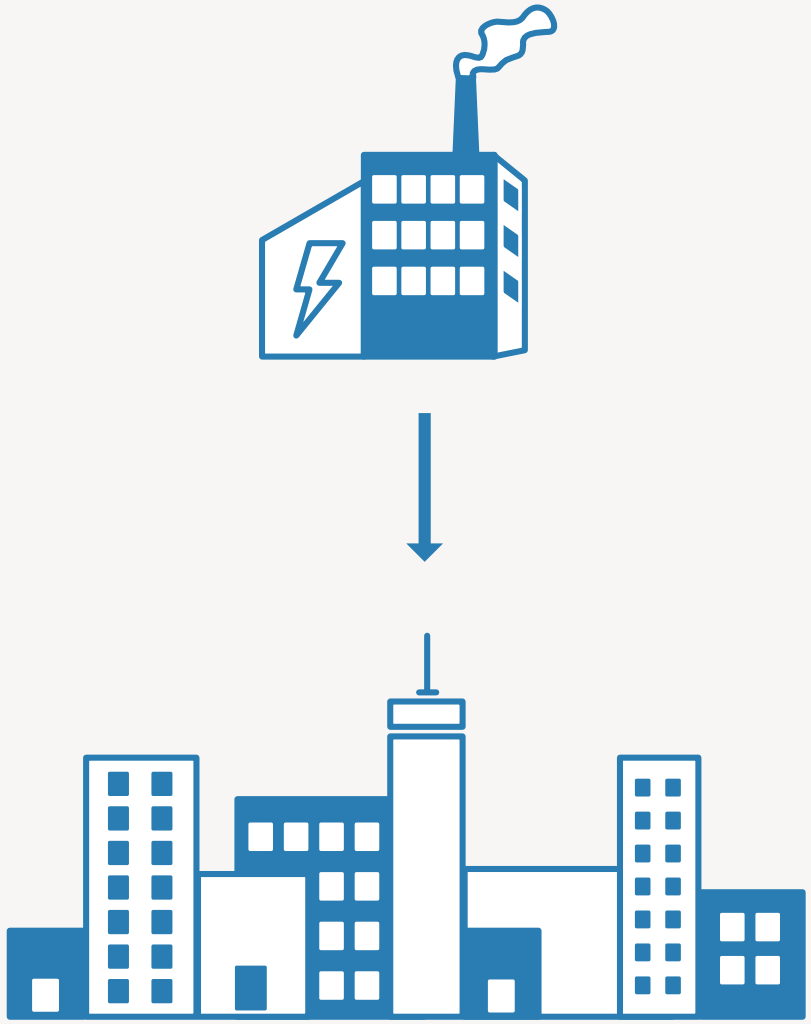


Common Regression Algorithms *continued*

Example: Forecasting Energy Load

Utility analysts at a large gas and electricity company developed models that predict energy demand for the following day. The models enable power grid operators to optimize resources and schedule power plant generation. Each model accesses a central database for historical power consumption and price data, weather forecasts, and parameters for each power plant, including maximum power out, efficiency, costs, and all the operation constraints that influence the plant dispatch.

Analysts looked for a model that provided a low mean absolute percent error (MAPE) to the testing data set. After trying several different types of regression models, it was determined that neural networks provided the lowest MAPE due to their ability to capture the nonlinear behavior of the system.



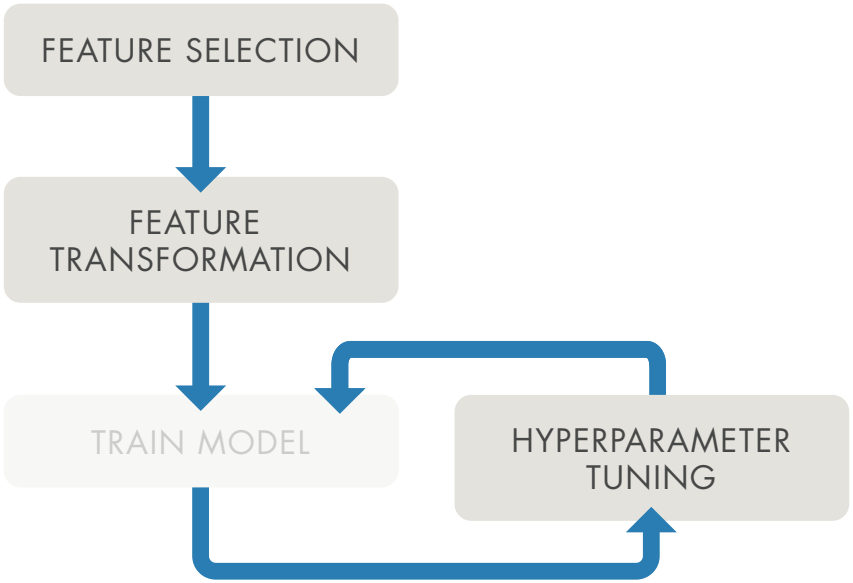
Improving Models

Improving a model means increasing its accuracy and predictive power and preventing overfitting (when the model cannot distinguish between data and noise). Model improvement involves feature engineering (feature selection and transformation) and hyperparameter tuning.

Feature selection: Identifying the most relevant features, or variables, that provide the best predictive power in modeling your data. This could mean adding variables to the model or removing variables that do not improve model performance.

Feature transformation: Turning existing features into new features using techniques such as principal component analysis, nonnegative matrix factorization, and factor analysis.

Hyperparameter tuning: The process of identifying the set of parameters that provides the best model. Hyperparameters control how a machine learning algorithm fits the model to the data.



Feature Selection

Feature selection is one of the most important tasks in machine learning. It's especially useful when you're dealing with high-dimensional data or when your dataset contains a large number of features and a limited number of observations. Reducing features also saves storage and computation time and makes your results easier to understand.

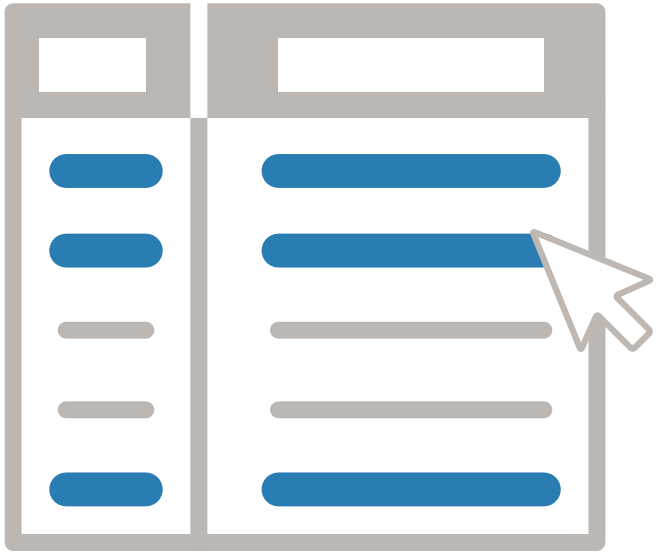
Common feature selection techniques include:

Stepwise regression: Sequentially adding or removing features until there is no improvement in prediction accuracy.

Sequential feature selection: Iteratively adding or removing predictor variables and evaluating the effect of each change on the performance of the model.

Regularization: Using shrinkage estimators to remove redundant features by reducing their weights (coefficients) to zero.

Neighborhood component analysis (NCA): Finding the weight each feature has in predicting the output, so that features with lower weights can be discarded.



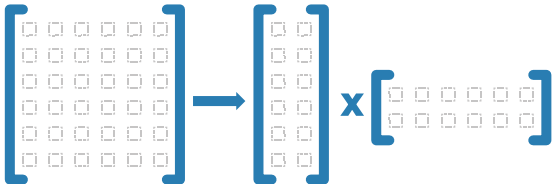
A model is only as good as the features you select to train it.

Feature Transformation

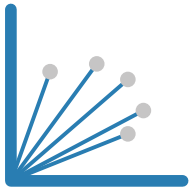
Feature transformation is a form of dimensionality reduction. As we saw in section 3, the three most commonly used dimensionality reduction techniques are:

Principal component analysis (PCA): Performs a linear transformation on the data so that most of the variance or information in your high-dimensional dataset is captured by the first few principal components. The first principal component will capture the most variance, followed by the second principal component, and so on.

Nonnegative matrix factorization: Used when model terms must represent nonnegative quantities, such as physical quantities.



Factor analysis: Identifies underlying correlations between variables in your dataset to provide a representation in terms of a smaller number of unobserved latent factors, or common factors.

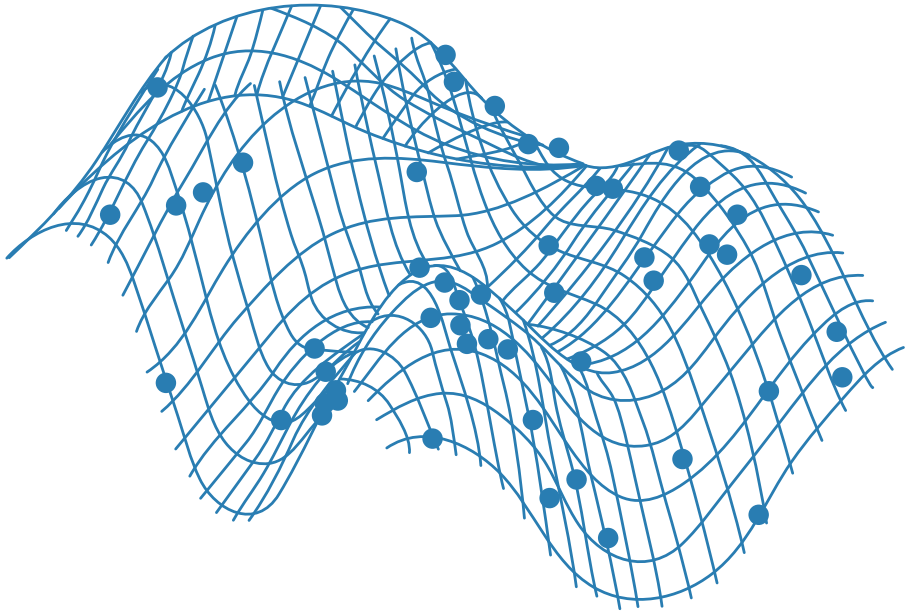


Hyperparameter Tuning

Like many machine learning tasks, parameter tuning is an iterative process. You begin by setting parameters based on a "best guess" of the outcome. Your goal is to find the "best possible" values—those that yield the best model. As you adjust parameters and model performance begins to improve, you see which parameter settings are effective and which still require tuning.

Three common parameter tuning methods are:

- Bayesian optimization
- Grid search
- Gradient-based optimization



A simple algorithm with well-tuned parameters often produces a better model than an inadequately tuned complex algorithm.

Learn More

Ready for a deeper dive? Explore these machine learning methods, examples, and tools.

Getting Started with Supervised Learning

Classification

Machine Learning with MATLAB: Getting Started with Classification

Introductory Classification Examples

Bayesian Brain Teaser

Explore Decision Trees Interactively

Support Vector Machines

K-Nearest Neighbor Classification

Train a Classification Ensemble

Predicting Tumor Class from Gene Expression Data Using Bagged Decision Trees

Regression

Linear Regression

What are Generalized Linear Models?

Regression Trees

Train a Regression Ensemble to Predict the Fuel Economy of a Car

Feature Selection

Selecting Features for Classifying High-Dimensional Data

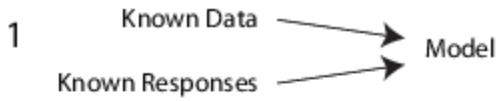
80827v00

Supervised Learning Workflow and Algorithms

What is Supervised Learning?

The aim of supervised, machine learning is to build a model that makes predictions based on evidence in the presence of uncertainty. As adaptive algorithms identify patterns in data, a computer "learns" from the observations. When exposed to more observations, the computer improves its predictive performance.

Specifically, a supervised learning algorithm takes a known set of input data and known responses to the data (output), and *trains* a model to generate reasonable predictions for the response to new data.



For example, suppose you want to predict whether someone will have a heart attack within a year. You have a set of data on previous patients, including age, weight, height, blood pressure, etc. You know whether the previous patients had heart attacks within a year of their measurements. So, the problem is combining all the existing data into a model that can predict whether a new person will have a heart attack within a year.

You can think of the entire set of input data as a heterogeneous matrix. Rows of the matrix are called *observations*, *examples*, or *instances*, and each contain a set of measurements for a subject (patients in the example). Columns of the matrix are called *predictors*, *attributes*, or *features*, and each are variables representing a measurement taken on every subject (age, weight, height, etc. in the example). You can think of the response data as a column vector where each row contains the output of the corresponding observation in the input data (whether the patient had a heart attack). To *fit* or *train* a supervised learning model, choose an appropriate algorithm, and then pass the input and response data to it.

Supervised learning splits into two broad categories: classification and regression.

- In *classification*, the goal is to assign a class (or *label*) from a finite set of classes to an observation. That is, responses are categorical variables. Applications include spam filters, advertisement recommendation systems, and image and speech recognition. Predicting whether a patient will have a heart attack within a year is a classification problem, and the possible classes are `true` and `false`. Classification algorithms usually apply to nominal response values. However, some algorithms can accommodate ordinal classes (see `fitcecoc`).

- In *regression*, the goal is to predict a continuous measurement for an observation. That is, the responses variables are real numbers. Applications include forecasting stock prices, energy consumption, or disease incidence.

  Statistics and Machine Learning Toolbox™ supervised learning functionalities comprise a stream-lined, object framework. You can efficiently train a variety of algorithms, combine models into an ensemble, assess model performances, cross-validate, and predict responses for new data.

  ### Steps in Supervised Learning

  While there are many Statistics and Machine Learning Toolbox™ algorithms for supervised learning, most use the same basic workflow for obtaining a predictor model. (Detailed instruction on the steps for ensemble learning is in Framework for Ensemble Learning.) The steps for supervised learning are:

- Prepare Data
- Choose an Algorithm
- Fit a Model
- Choose a Validation Method
- Examine Fit and Update Until Satisfied
- Use Fitted Model for Predictions

  #### Prepare Data

  All supervised learning methods start with an input data matrix, usually called `X` here. Each row of `X` represents one observation. Each column of `X` represents one variable, or predictor.

  Represent missing entries with `NaN` values in `X`. Statistics and Machine Learning Toolbox supervised learning algorithms can handle `NaN` values, either by ignoring them or by ignoring any row with a `NaN` value.

  You can use various data types for response data `Y`. Each element in `Y` represents the response to the corresponding row of `X`. Observations with missing `Y` data are ignored.

- For regression, `Y` must be a numeric vector with the same number of elements as the number of rows of `X`.
- For classification, `Y` can be any of these data types. This table also contains the method of including missing entries.

Data Type
Numeric vector
Categorical vector
Character array
Cell array of character vectors

Data Type
Logical vector

Choose an Algorithm

There are tradeoffs between several characteristics of algorithms, such as:

- Speed of training
- Memory usage
- Predictive accuracy on new data
- Transparency or interpretability, meaning how easily you can understand the reasons an algorithm makes its predictions

Details of the algorithms appear in Characteristics of Classification Algorithms. More detail about ensemble algorithms is in Choose an Applicable Ensemble Aggregation Method.

Fit a Model

The fitting function you use depends on the algorithm you choose.

Algorithm
Classification Trees
Regression Trees
Discriminant Analysis (classification)
k-Nearest Neighbors (classification)
Naive Bayes (classification)
Support Vector Machines (SVM) for classification
SVM for regression
Multiclass models for SVM or other classifiers
Classification Ensembles
Regression Ensembles
Classification or Regression Tree Ensembles (e.g., Random Forests [1]) in Parallel

For a comparison of these algorithms, see Characteristics of Classification Algorithms.

Choose a Validation Method

The three main methods to examine the accuracy of the resulting fitted model are:

- Examine the resubstitution error. For examples, see:
  - Classification Tree Resubstitution Error
  - Cross Validate a Regression Tree
  - Test Ensemble Quality

- Example: Resubstitution Error of a Discriminant Analysis Classifier
- Examine the cross-validation error. For examples, see:
- Cross Validate a Regression Tree
- Test Ensemble Quality
- Classification with Many Categorical Levels
- Cross Validating a Discriminant Analysis Classifier
- Examine the out-of-bag error for bagged decision trees. For examples, see:
- Test Ensemble Quality
- Regression of Insurance Risk Rating for Car Imports Using TreeBagger
- Classifying Radar Returns for Ionosphere Data Using TreeBagger

Examine Fit and Update Until Satisfied

After validating the model, you might want to change it for better accuracy, better speed, or to use less memory.

- Change fitting parameters to try to get a more accurate model. For examples, see:
- Tune RobustBoost
- Train Ensemble With Unequal Classification Costs
- Improving Discriminant Analysis Models
- Change fitting parameters to try to get a smaller model. This sometimes gives a model with more accuracy. For examples, see:
- Select Appropriate Tree Depth
- Prune a Classification Tree
- Surrogate Splits
- Regularize a Regression Ensemble
- Regression of Insurance Risk Rating for Car Imports Using TreeBagger
- Classifying Radar Returns for Ionosphere Data Using TreeBagger
- Try a different algorithm. For applicable choices, see:
- Characteristics of Classification Algorithms
- Choose an Applicable Ensemble Aggregation Method

When satisfied with a model of some types, you can trim it using the appropriate compact function (compact for classification trees, compact for regression trees, compact for discriminant analysis, compact for naive Bayes, compact for SVM, compact for ECOC models, compact for classification ensembles, and compact for regression ensembles). compact removes training data and other properties not required for prediction, e.g., pruning information for decision trees, from the model to reduce memory consumption. Because *k*NN classification models require all of the training data to predict labels, you cannot reduce the size of a ClassificationKNN model.

Use Fitted Model for Predictions

To predict classification or regression response for most fitted models, use the predict method:

```
Ypredicted = predict(obj,Xnew)
```

- obj is the fitted model or fitted compact model.
- Xnew is the new input data.
- Ypredicted is the predicted response, either classification or regression.

Characteristics of Classification Algorithms

This table shows typical characteristics of the various supervised learning algorithms. The characteristics in any particular case can vary from the listed ones. Use the table as a guide for your initial choice of algorithms. Decide on the tradeoff you want in speed, memory usage, flexibility, and interpretability.

Tip

Try a decision tree or discriminant first, because these classifiers are fast and easy to interpret. If the models are not accurate enough predicting the response, try other classifiers with higher flexibility.

To control flexibility, see the details for each classifier type. To avoid overfitting, look for a model of lower flexibility that provides sufficient accuracy.

Classifier	Multiclass Support	Categorical Predictor Support	Prediction Speed	Memory Usage	Interpretability
Decision Trees — fitctree	Yes	Yes	Fast	Small	Easy
Discriminant analysis — fitcdiscr	Yes	No	Fast	Small for linear, large for quadratic	Easy
SVM — fitcsvm	No. Combine multiple binary SVM classifiers using fitcecoc.	Yes	Medium for linear. Slow for others.	Medium for linear. All others: medium for multiclass, large for binary.	Easy for linear SVM. Hard for all other kernel types.
Naive Bayes — fitcnb	Yes	Yes	Medium for simple distributions. Slow for kernel distributions or high-dimensional data	Small for simple distributions. Medium for kernel distributions or high-dimensional data	Easy
Nearest neighbor — fitcknn	Yes	Yes	Slow for cubic. Medium for others.	Medium	Hard
Ensembles — fitensemble	Yes	Yes	Fast to medium depending on choice of algorithm	Low to high depending on choice of algorithm.	Hard

The results in this table are based on an analysis of many data sets. The data sets in the study have up to 7000 observations, 80 predictors, and 50 classes. This list defines the terms in the table.

Speed:

- Fast — 0.01 second
- Medium — 1 second
- Slow — 100 seconds

Memory

- Small — 1MB
- Medium — 4MB
- Large — 100MB

Note

The table provides a general guide. Your results depend on your data and the speed of your machine.

Categorical Predictor Support

This table describes the data-type support of predictors for each classifier.

Classifier	All predictors numeric	All predictors categorical
Decision Trees	Yes	Yes
Discriminant Analysis	Yes	No
SVM	Yes	Yes
Naive Bayes	Yes	Yes
Nearest Neighbor	Euclidean distance only	Hamming distance only
Ensembles	Yes	Yes, except subspace ensembles of discriminant anal

References

[1] Breiman, L. *Random Forests.* Machine Learning 45, 2001, pp. 5–32.