**A Quick Introduction to Supervised vs. Unsupervised Learning**

The field of machine learning contains a massive set of algorithms that can be used for understanding data. These algorithms can be classified into one of two categories:

**1. Supervised Learning Algorithms:** Involves building a model to estimate or predict an output based on one or more inputs.

**2. Unsupervised Learning Algorithms:** Involves finding structure and relationships from inputs. There is no “supervising” output.

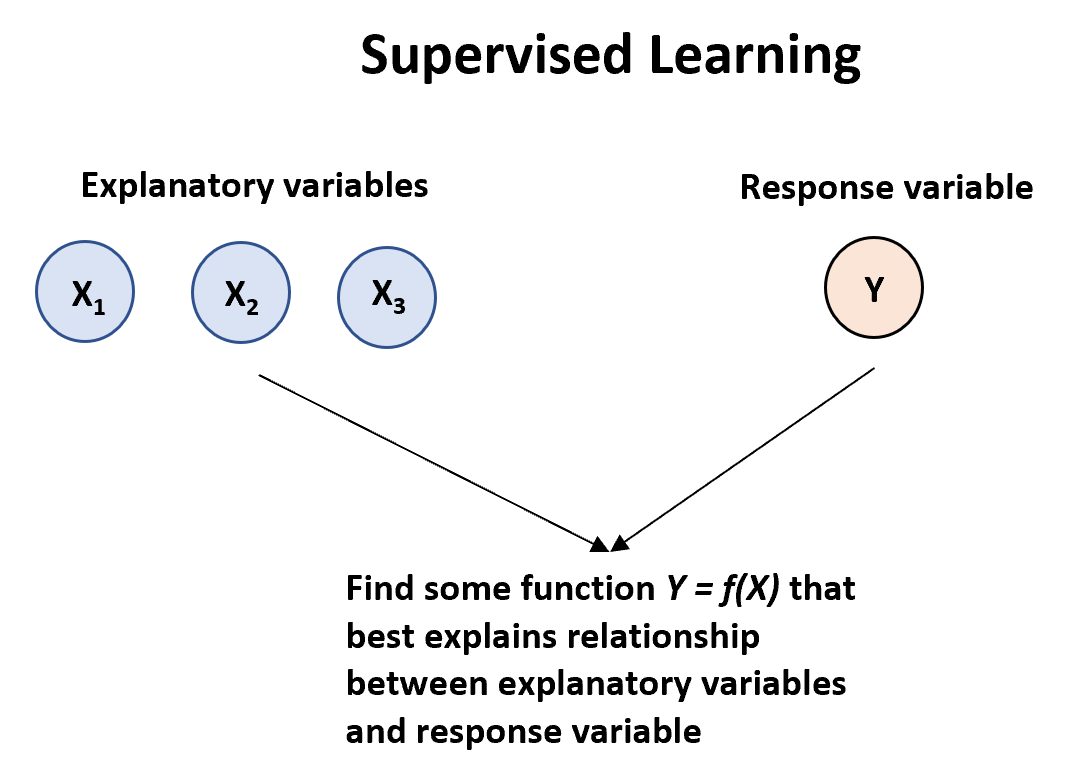
This tutorial explains the difference between these two types of algorithms along with several examples of each.

**Supervised Learning Algorithms**

A **supervised learning algorithm** can be used when we have one or more explanatory variables (X1, X2, X3, …, Xp) and a [response variable](https://www.statology.org/explanatory-response-variables/) (Y) and we would like to find some function that describes the relationship between the explanatory variables and the response variable:

**Y = *f*(X) + ε**

where *f* represents systematic information that X provides about Y and where ε is a random error term independent of X with a mean of zero.



There are two main types of supervised learning algorithms:

**1. Regression:** The output variable is continuous (e.g. weight, height, time, etc.)

* To understand the association between an employee’s age and education, as well as the calendar year, on his wage.

The Wage data involves predicting a continuous or quantitative output value.

* Suppose we have a dataset that contains three variables for 100 different houses: square footage, number of bathrooms, and selling price.

We could fit a regression model that uses square footage and number of bathrooms as explanatory variables and selling price as the response variable.

We could then use this model to predict the selling price of a house, based on its square footage and number of bathrooms. This is an example of a regression model because the response variable (selling price) is continuous.

**2. Classification:** The output variable is categorical (e.g. male or female, pass or fail, benign or malignant, etc.) non-numerical

* Predicting whether a given day’s stock market performance will fall into the Up bucket or the Down bucket.
* Suppose we have a dataset that contains three variables for 100 different college basketball players: average points per game, division level, and whether or not they got drafted into the NBA.

We could fit a classification model that uses average points per game and division level as explanatory variables and “drafted” as the response variable.

We could then use this model to predict whether or not a given player will get drafted into the NBA based on their average points per game and division level.

This is an example of a classification model because the response variable (“drafted”) is categorical. That is, it can only take on values in two different categories: “Drafted” or “Not drafted.”

The inputs go by different names, such as *predictors*, *independent variables*, *features*, or sometimes just *variables*.

The output variable in previous example, sales is variable often called the *response* or *dependent variable.*

There are two main reasons that we use supervised learning algorithms:

1. **Prediction:** We often use a set of explanatory variables to predict the value of some response variable

* *square footage* and *number of bedrooms* to predict *house price*

For instance, consider a company that is interested in conducting a direct-marketing campaign.

* The goal is to identify individuals who will respond positively to a mailing, based on observations of demographic variables measured on each individual.

In this case, the demographic variables serve as predictors, and response to the marketing campaign (either positive or negative) serves as the outcome.

The company is not interested in obtaining a deep understanding of the relationships between each individual predictor and the response; instead, the company simply wants an accurate model to predict the response using the predictors. This is an example of modeling for prediction.

1. **Inference:** We may be interested in understanding the way that a response variable is affected as the value of the explanatory variables change.

* how much does home price increase, on average, when the number of bedrooms increases by one

We instead want to understand the relationship between X and Y , or more specifically, to understand how Y changes as a function of X1, . . .,Xp.

* F(X) cannot be treated as a black box, because we need to know its exact form.
  + Which predictors are associated with the response?
  + What is the relationship between the response and each predictor?
  + Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?
* Consider the Advertising data, One may be interested in answering questions such as:

– Which media contribute to sales?

– Which media generate the biggest boost in sales? or

– How much increase in sales is associated with a given increase in TV advertising?

* The brand of a product that a customer might purchase based on variables such as price, store location, discount levels, competition price, and so forth.

– In this situation one is interested in how each of the individual variables affects the probability of purchase.

– For instance, what effect will changing the price of a product have on sales?

Finally, some modeling could be conducted both for prediction and inference. For example, in a real estate setting, one may seek to relate values of homes to inputs such as crime rate, zoning, distance from a river, air quality, schools, income level of community, size of houses, and so forth.

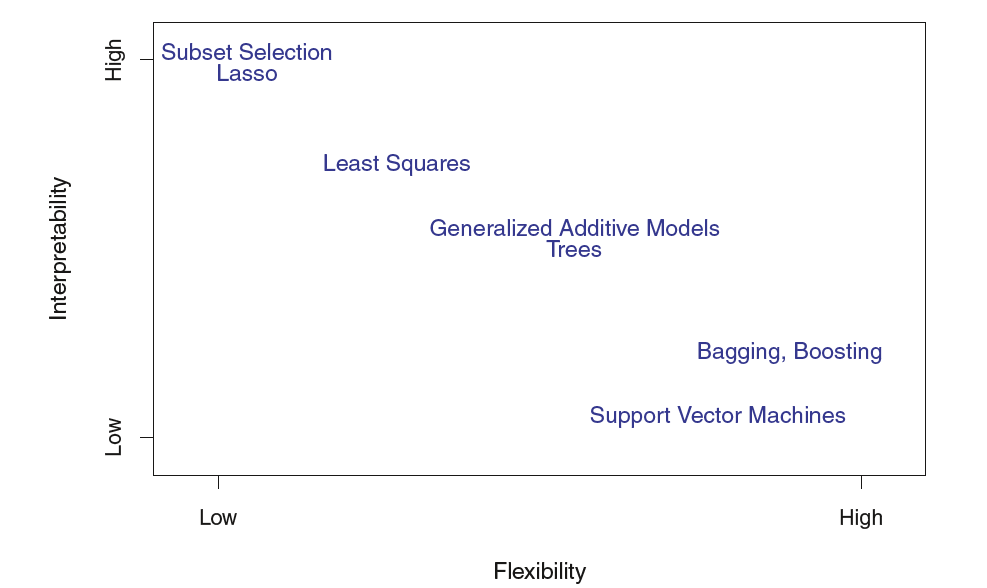
* In this case one might be interested in how the individual input variables affect the prices—that is, how much extra will a house be worth if it has a view of the river? This is an inference problem.
* Alternatively, one may simply be interested in predicting the value of a home given its characteristics: is this house under- or over-valued? This is a prediction problem.

Depending on whether our goal is inference or prediction (or a mix of both), we may use different methods for estimating the function *f*. For example, linear models offer easier interpretation but non-linear models that are difficult to interpret may offer more accurate prediction.

Here is a list of the most commonly used supervised learning algorithms:

* Linear regression
* Logistic regression
* Linear discriminant analysis
* Quadratic discriminant analysis
* Decision trees
* Naive bayes
* Support vector machines
* Neural networks

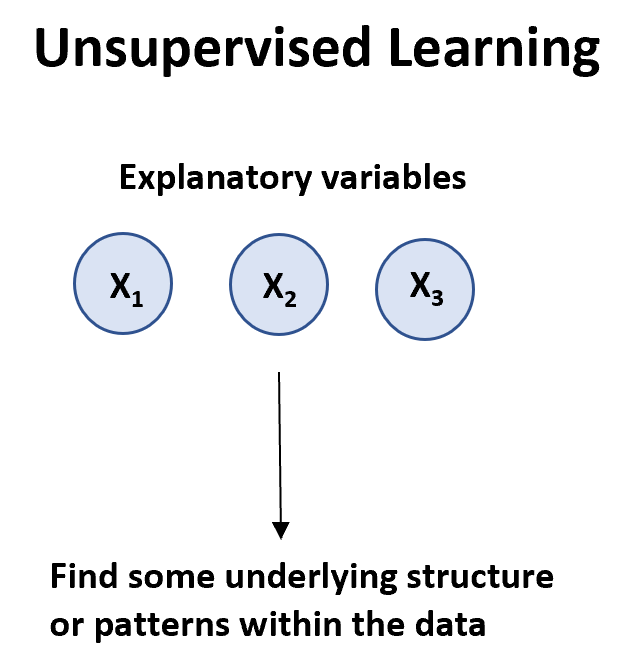
The Trade-Off Between Prediction Accuracy and Model Interpretability



* A representation of the tradeoff between flexibility and interpretability, using different statistical learning methods.
* In general, as the flexibility of a method increases, its interpretability decreases.
* when inference is the goal, there are clear advantages to using simple and relatively inflexible statistical learning methods.
* When interested in prediction, and the interpretability of the predictive model is simply not of interest.

**Unsupervised Learning Algorithms**

An **unsupervised learning algorithm** can be used when we have a list of variables (X1, X2, X3, …, Xp) and we would simply like to find underlying structure or patterns within the data.



There are two main types of unsupervised learning algorithms:

**1. Clustering:** Using these types of algorithms, we attempt to find “clusters” of [observations](https://www.statology.org/observation-in-statistics/) in a dataset that are similar to each other. This is often used in retail when a company would like to identify clusters of customers who have similar shopping habits so that they can create specific marketing strategies that target certain clusters of customers.

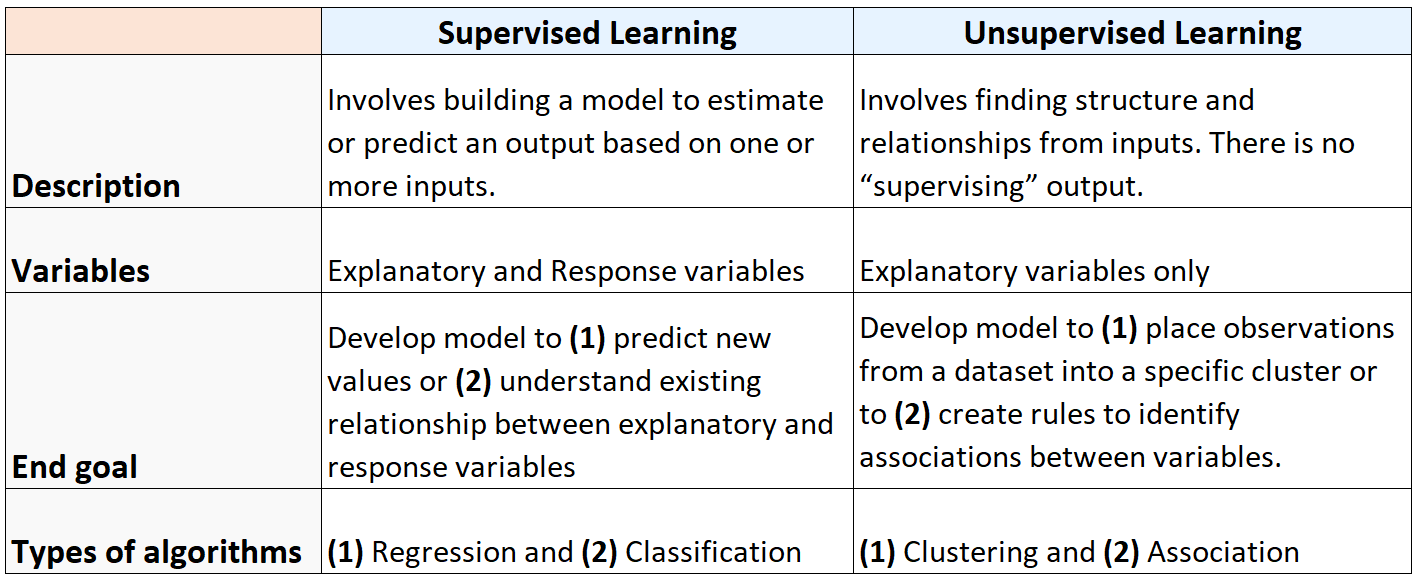
**2. Association:** Using these types of algorithms, we attempt to find “rules” that can be used to draw associations. For example, retailers may develop an association algorithm that says “if a customer buys product X they are highly likely to also buy product Y.”

Here is a list of the most commonly used unsupervised learning algorithms:

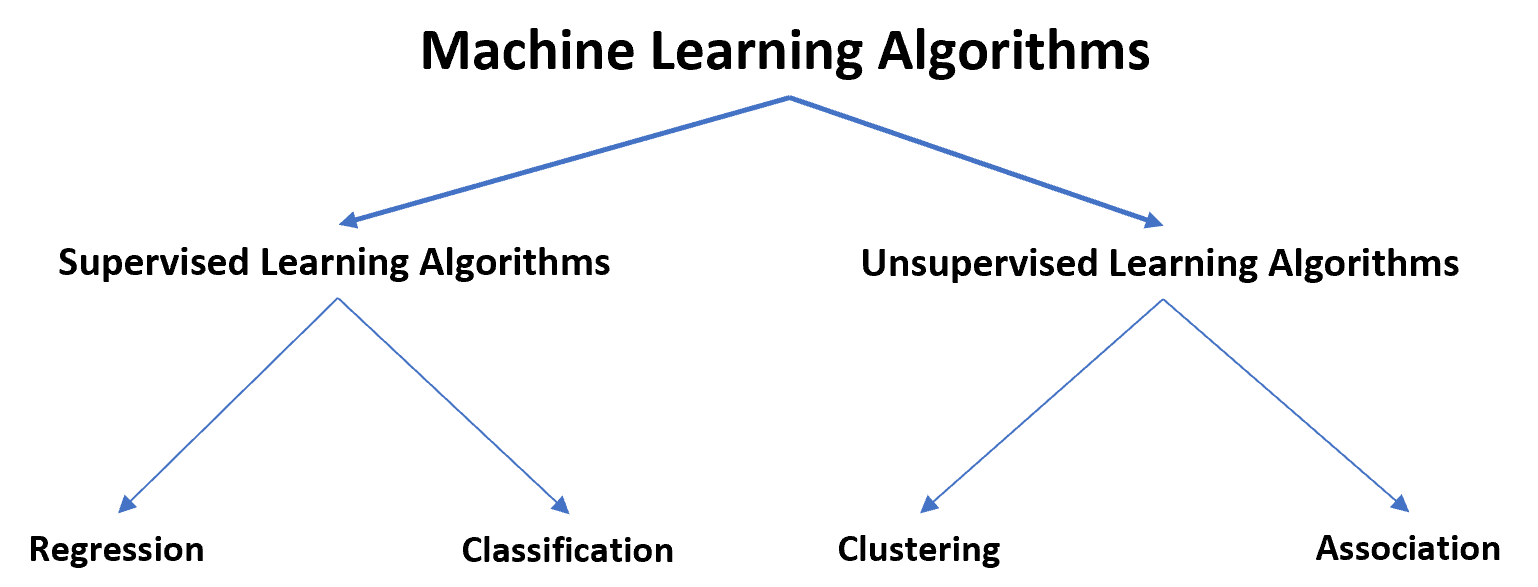
* Principal component analysis
* K-means clustering
* K-medoids clustering
* Hierarchical clustering
* Apriori algorithm

**Summary: Supervised vs. Unsupervised Learning**

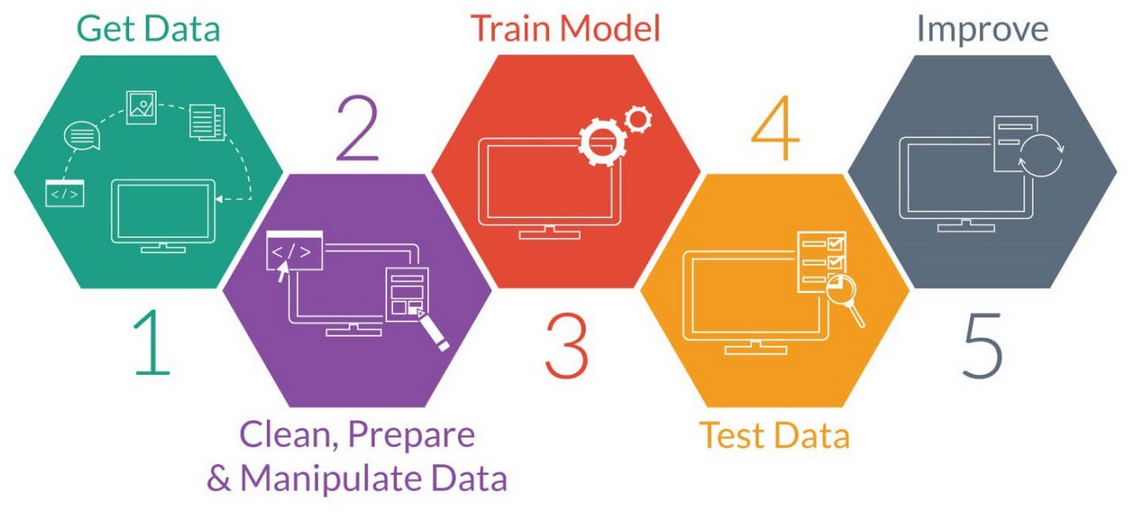
The following table summarizes the differences between supervised and unsupervised learning algorithms:



And the following diagram summarizes the types of machine learning algorithms:



**Problem Solving in Analytics**

****

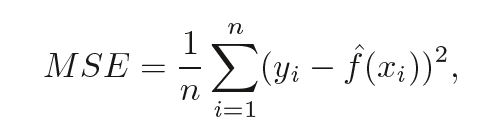
**Assessing Model Accuracy**

* *There is no free lunch in statistics:* no one method dominates all others over all possible data sets.
* Selecting the best approach can be one of the most challenging parts of performing statistical learning in practice.

**Measuring the Quality of Fit**

In order to evaluate the performance of a statistical learning method on a given data set, we need some way to measure how well its predictions actually match the observed data.

In the regression setting, the most commonly-used measure is the mean squared error (MSE)



For good model, it should be as small as possible. The closer the model predictions are to the observations, the smaller the MSE will be.

The most common way to measure the accuracy of a classification model is by simply calculating the percentage of correct classifications the model makes:

Accuracy = correction classifications / total attempted classifications \* 100%

For example, if a model correctly identifies whether or not a player will get drafted into the NBA 88 times out of 100 possible times then the accuracy of the model is:

Accuracy = (88/100) \* 100% = 88%

The higher the accuracy, the better a classification model is able to predict outcomes.

**We should always care about performance on test data. Why we should check on test data?**

* Suppose that we are interested test data in developing an algorithm to predict a stock’s price based on previous stock returns.
  + We can train the method using stock returns from the past 6 months.
  + But we don’t really care how well our method predicts last week’s stock price.
  + We instead care about how well it will predict tomorrow’s price or next month’s price.
* Suppose that we have clinical measurements (e.g. weight, blood pressure, height, age, family history of disease) for a number of patients, as well as information about whether each patient has diabetes.
  + We can use these patients to train a statistical learning method to predict risk of diabetes based on clinical measurements.
  + In practice, we want this method to accurately predict diabetes risk for *future patients* based on their clinical measurements. We are not very interested in train data, since we already know which of those patients have diabetes.

We only care about test MSE – the MSE when our model is applied to unseen data. This is because we only care about how the model will perform on unseen data, not existing data.

It turns out that the test MSE can always be decomposed into two parts:

1. The variance: Refers to the amount by which our function f would change if we estimated it using a different training set.
2. The bias: Refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.

Written in mathematical terms:

Test MSE = Var(f̂(x0)) + [Bias(f̂(x0))]2 + Var(ε)

Test MSE = Variance + Bias2 + Irreducible error

The third term, the irreducible error, is the error that cannot be reduced by any model simply because there always exists some noise in the relationship between the set of explanatory variables and the response variable.

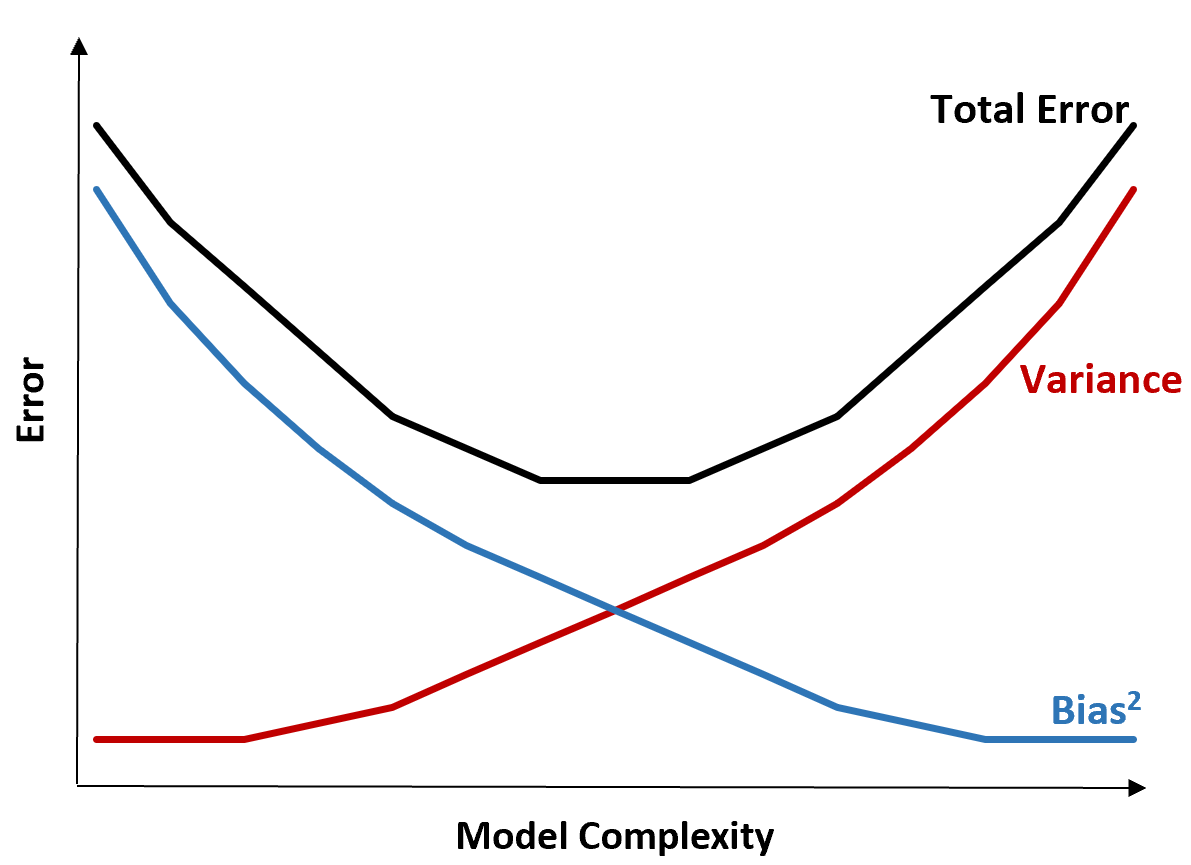
Models that have high bias tend to have low variance. For example, linear regression models tend to have high bias (assumes a simple linear relationship between explanatory variables and response variable) and low variance (model estimates won’t change much from one sample to the next).

However, models that have low bias tend to have high variance. For example, complex non-linear models tend to have low bias (does not assume a certain relationship between explanatory variables and response variable) with high variance (model estimates can change a lot from one training sample to the next).

**The Bias-Variance Tradeoff**

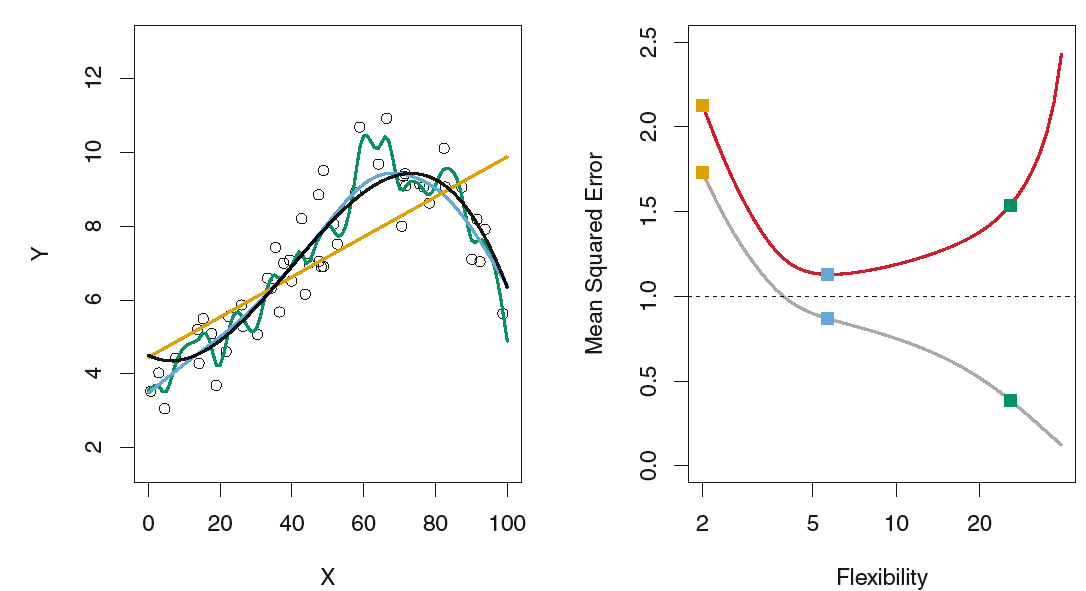
The bias-variance tradeoff refers to the tradeoff that takes place when we choose to lower bias which typically increases variance, or lower variance which typically increases bias.

The following chart offers a way to visualize this tradeoff:



The total error decreases as the complexity of a model increases but only up to a certain point. Past a certain point, variance begins to increase and total error also begins to increase.

In practice, we only care about minimizing the total error of a model, not necessarily minimizing the variance or bias. It turns out that the way to minimize the total error is to strike the right balance between variance and bias.



* *Three estimates of Y= f (X)are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).*
* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line).*
* *Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

In other words, we want a model that is complex enough to capture the true relationship between the explanatory variables and the response variable, but not overly complex such that it finds patterns that don’t really exist.

When a model is too complex, it overfits the data. This happens because it works too hard to find patterns in the training data that are just caused by random chance. This type of model is likely to perform poorly on unseen data.

But when a model is too simple, it underfits the data. This happens because it assumes the true relationship between the explanatory variables and the response variable is simpler than it actually is.

The way to pick optimal models in machine learning is to strike the balance between bias and variance such that we can minimize the test error of the model on future unseen data.

In practice, the most common way to minimize test MSE is to use cross-validation.

**An Easy Guide to K-Fold Cross-Validation**

To evaluate the performance of some model on a dataset, we need to measure how well the predictions made by the model match the observed data.

The most common way to measure this is by using the mean squared error (MSE), which is calculated as:

MSE = (1/n)\*Σ(yi – f(xi))2

where:

n: Total number of observations

yi: The response value of the ith observation

f(xi): The predicted response value of the ith observation

The closer the model predictions are to the observations, the smaller the MSE will be.

In practice, we use the following process to calculate the MSE of a given model:

1. Split a dataset into a training set and a testing set.

2. Build the model using only data from the training set.

3. Use the model to make predictions on the testing set and measure the test MSE.

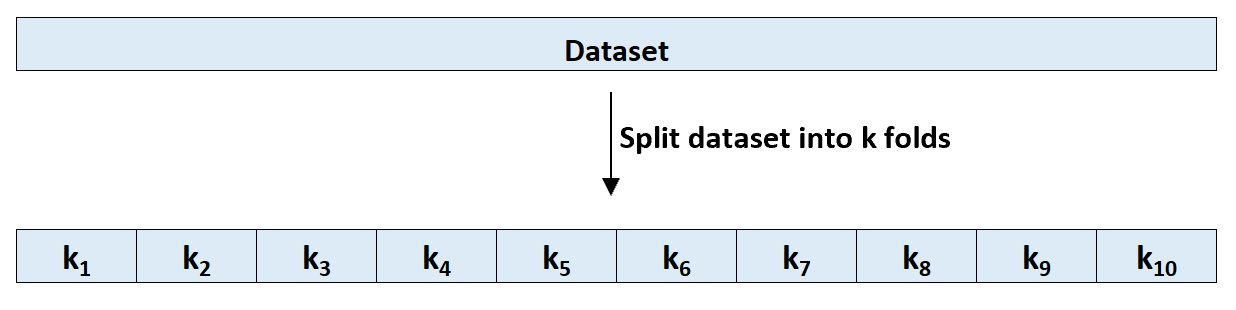
The test MSE gives us an idea of how well a model will perform on data it hasn’t previously seen. However, the drawback of using only one testing set is that the test MSE can vary greatly depending on which observations were used in the training and testing sets.

One way to avoid this problem is to fit a model several times using a different training and testing set each time, then calculating the test MSE to be the average of all of the test MSE’s.

This general method is known as cross-validation and a specific form of it is known as k-fold cross-validation.

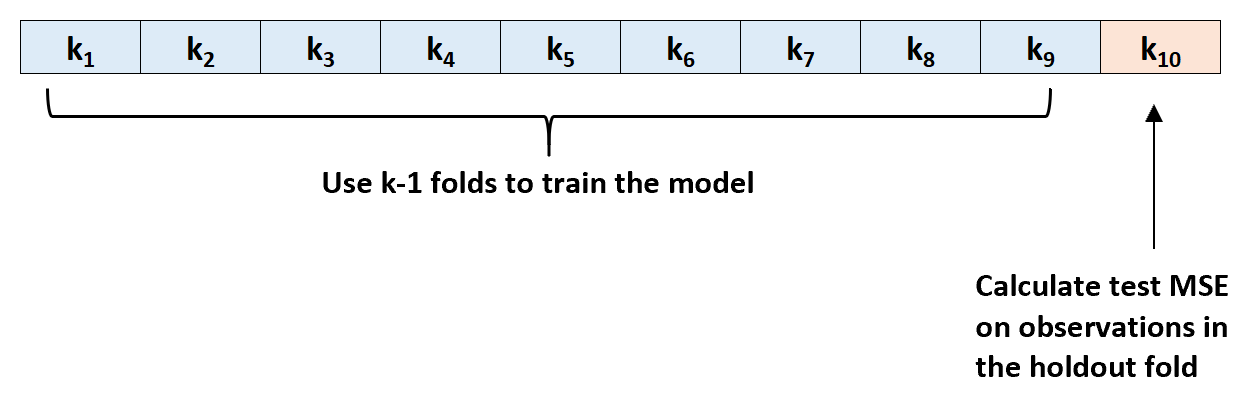
K-fold cross-validation uses the following approach to evaluate a model:

Step 1: Randomly divide a dataset into k groups, or “folds”, of roughly equal size.

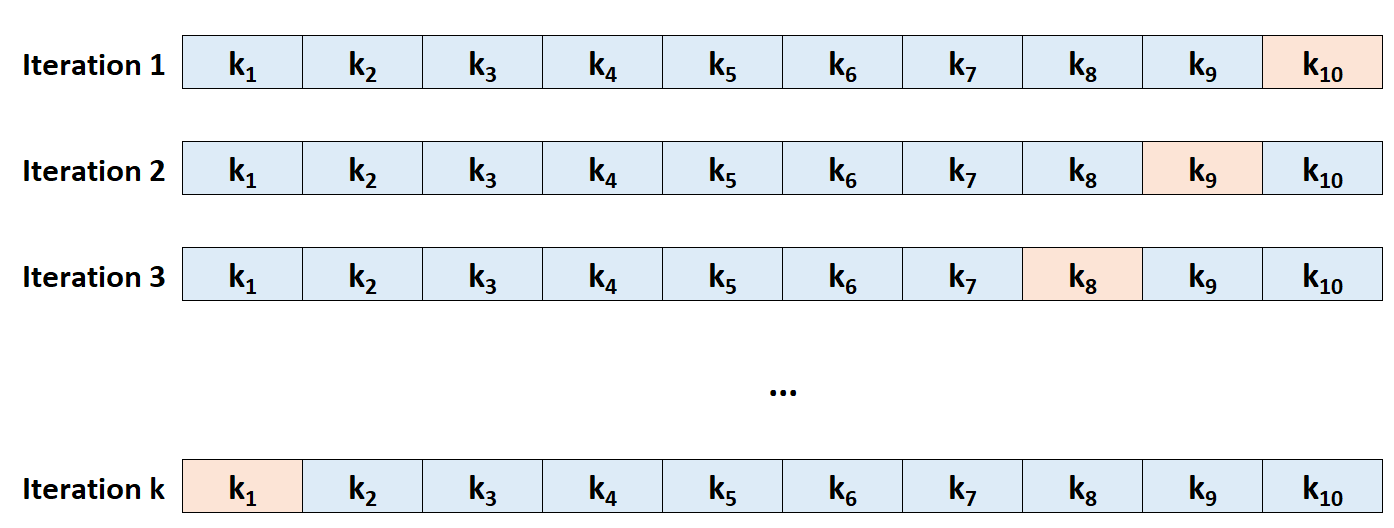


Splitting a dataset into k folds

Step 2: Choose one of the folds to be the holdout set. Fit the model on the remaining k-1 folds. Calculate the test MSE on the observations in the fold that was held out.



Step 3: Repeat this process k times, using a different set each time as the holdout set.



Step 4: Calculate the overall test MSE to be the average of the k test MSE’s.

Test MSE = (1/k)\*ΣMSEi

where:

k: Number of folds

MSEi: Test MSE on the ith iteration

In practice, we typically choose to use between 5 and 10 folds. As noted in An Introduction to Statistical Learning, this number of folds has been shown to offer an optimal balance between bias and variance and thus provide reliable estimates of test MSE:

**Advantages of K-Fold Cross-Validation**

When we split a dataset into just one training set and one testing set, the test MSE calculated on the observations in the testing set can vary greatly depending on which observations were used in the training and testing sets.

By using k-fold cross-validation, we’re able to use calculate the test MSE using several different variations of training and testing sets. This makes it much more likely for us to obtain an unbiased estimate of the test MSE.