Understand the problem By Understanding the Data Predictive Model Building: Balancing Performance, Complexity, and the Big Data

- This chapter discusses the factors affecting the performance of machine learning models. The chapter provides technical definitions of *performance* for different types of machine learning problems.
- In an e-commerce application, for example, good performance might mean returning correct search results or presenting ads that site visitors frequently click.
- The goal of selecting and fitting a predictive algorithm is to achieve the best possible performance.
- Achieving performance goals involves three factors:
    - Complexity of the problem
    - Complexity of the algorithmic model employed
    - The amount and richness of the data available
- <u>The Basic Problem: Understanding Function Approximation:</u>
    - The problem statement for a specific class of predictive problem has two types of variables:
        - The variable that you are attempting to predict (E.g.: whether a visitor to a website will click an ad)
        - Other variables (E.g.: the visitor's demographics or past behavior on the site) that you can use to make the prediction.
    - Problems of this type are referred to as *function approximation problems*.
    - In a function approximation problem, the designer starts with a collection of historical examples for which the correct answer is known. For example, historical web log files will indicate whether a visitor clicked an ad when shown the ad.
    - The data scientist next has to find other data that can be used to build a predictive model. For example, to predict whether a site visitor will click an ad, the data scientist might try using other pages that the visitor viewed before seeing the ad. If the user is registered with the site, data on past purchases or pages viewed might be available for making a prediction.
    - The variable being predicted is referred to by a number of different names, such as *target, label* and *outcome.*

- The variables being used to make the predictions are variously called *predictors, regressors, features* and *attributes*.
- Determining what attributes to use for making predictions is called *feature engineering*. Data cleaning and feature engineering take 80% to 90% of a data scientist's time.
- <u>Working with Training Data:</u>
  - The data scientist starts algorithm development with a training set.
  - The process of building a predictive model is called training.
  - The training set consists of outcome examples and the assemblage (collection) of features chosen by the data scientist. The training set comprises two types of data:
    - The outcomes you want to predict
    - The features available for making the prediction
  - Following table provides an example of a training set. The leftmost column contains outcomes (whether a site visitor clicked a link) and features to be used to make predictions about whether visitors will click the link in the future:

OUTCOMES: CLICKED ON LINK	**FEATURE 1: GENDER**	**FEATURE2: MONEY SPENT ON SITE**	**FEATURE3: AGE**
YES	M	0	25
NO	F	250	32
YES	M	12	17

*** Data for Machine Learning Problem ***

  - The predictor values can be arranged in the form of matrix. The table of predictors will be called *X*, and it has the following form:

$$X = \begin{matrix} x11 & x12 \ldots\ldots. & x1n \\ x21 & x22 \ldots\ldots. & x2n \\ \ldots & \ldots & \ldots \\ xm1 & xm2 \ldots\ldots. & xmn \end{matrix}$$

<u>Notation for set of predictors</u>

- Referring to the above data set in table, x_{11} would be M (Gender), x_{12} would be 0.00 (money spent on site), x_{21} would be F (Gender) and so on.
- Sometimes it will be convenient to refer to all the attribute values for a particular example. For that purpose, x_i will refer to the i^{th} row of X. For the data set in above Table, X_2 would be a row vector containing the values F, 250, 32.
- Using the example of predicting ad clicks, the predictors might include demographic data about the site visitor like marital status, yearly income and etc.
- Attributes such as marital status, gender or the state of residence go by several different designations. They may be called *factor* or *categorical.*
- Attributes like age or income that are represented by numbers are called *numeric* or *real-valued.*
- The distinction between these two types of attributes is important because some algorithms may not handle one type or the other. For example, linear methods require numeric attributes.
- The targets corresponding to each row in X are arranged in a column vector Y as follows:

$$Y = \begin{matrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{matrix}$$

- The target y_i corresponds to x_i – the predictors in the *i*th row of X.
- Targets may be of several different forms. For example, they may be real numbers, like if the objective were to predict how much a customer will spend.
- When the targets are real numbers, the problem is called a *regression problem.*

        - If the targets are two-valued, the problem is called a *binary classification problem. (Predicting whether a customer will click an ad)*
        - If the targets contain several discrete values, the problem is a *multiclass classification problem. (Predicting which of several ads a customer will click.)*
        - The function pred() uses the attribute x_i to predict y_i.
    - Assessing Performance of Predictive Models:
        - Good performance means using the attributes X_i to generate a prediction that is close to y_i, but close has different meanings for different problems.
        - For a regression problem where y_i is a real number, performance is measured in terms like the MSE (Mean Squared Error) or the MAE (Mean Absolute Error).

$$Mean\,squared\,error = \left(\frac{1}{m}\right)\sum_{i=1}^{m}\left(y_i - pred\left(x_i\right)\right)^2$$

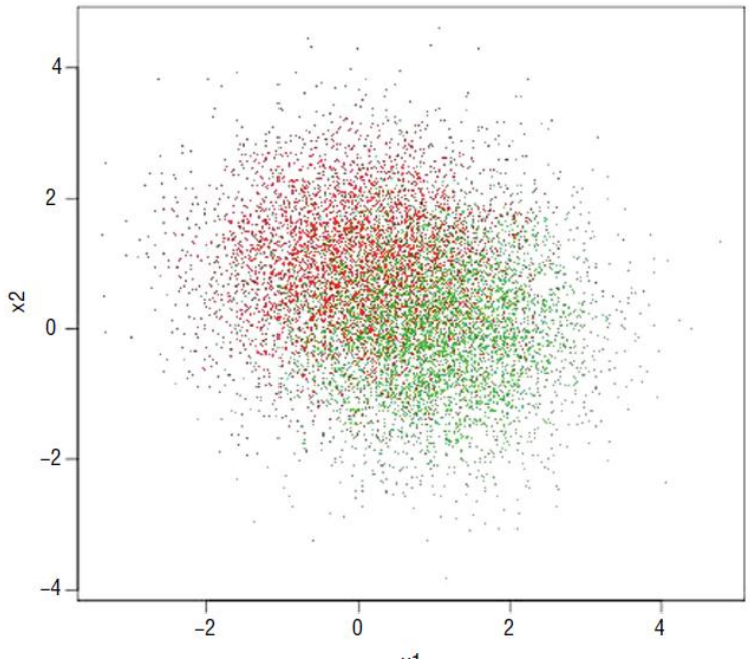
$$Mean\,absolute\,error = \left(\frac{1}{m}\right)\sum_{i=1}^{m}\left|y_i - pred\left(x_i\right)\right|$$

        - In a regression problem, the target (y_i) and the prediction, pred(x_i), are both real numbers, so it makes sense to describe the error as being the numeric difference between them.
        - MSE squares the errors and averages over the data set to produce a measure of the overall level of errors.
        - MAE averages the absolute values of the errors instead of averaging the squares of the errors.
- Factors Driving Algorithm Choices and Performance – Complexity and Data:
    - Several factors affect the overall performance of a predictive algorithm.
    - Among these factors are the complexity of the problem, the complexity of the model used and the amount of training data available.

- The following sections describe how these factors interrelate to determine performance.
- Contrast Between a Simple Problem and a Complex Problem:
    - The above section described several ways to quantify performance and highlighted the importance of performance on new data.
    - The goal of designing a predictive model is to make accurate predictions on new examples (such as new visitors to your site).
    - As a practicing data scientist, you will want an estimate of an algorithm's performance so that you can set expectations with your customer and compare algorithms with one another.
    - Best practice in predictive modeling requires that you hold out some data from the training set.
    - These held-out examples have labels associated with them and can be compared to predictions produced by models training on the remaining data.
    - Statisticians refer to this technique as out-of-sample error because it is an error on data not used in training.
    - The important thing is that the only performance that counts is the performance of the model when it is run against new examples.
    - One of the factors affecting performance is the complexity of the problem being solved. Following figure shows a relatively simple classification problem in two dimensions. There are two groups of points: dark and light points. The dark points are randomly drawn from a 2D Gaussian distribution centered at (1,0) with unit variance in both dimensions. The light points are also drawn from a Gaussian distribution having the same variance but centered at (0,1).
    - The attributes for the problem are the two axes in the plot: x1 and x2. The classification task is to draw some boundaries in the x1 , x2 plane to separate the light points from the dark points.
    - About the best that can be done in this circumstance is to draw a 45-degree line in the plot—that is the line where x1 equals x2 .

- In a precise probabilistic sense, that is the best possible classifier for this problem.

***Figure - 1 A Simple Classification Problem ***



- Following figure depicts a more complicated problem.
- The points shown in following figure are generated by drawing points at random.
- The main difference from the random draw that generated above figure is that the points in below figure are drawn from several distributions for the light points and several different ones for dark.
- This is called a mixture model. The general goal is basically the same: draw boundaries in the x1 , x2 plane to separate the light points from the dark points. In below figure, however, it is clear that a linear boundary will not separate the points as well as a curve.

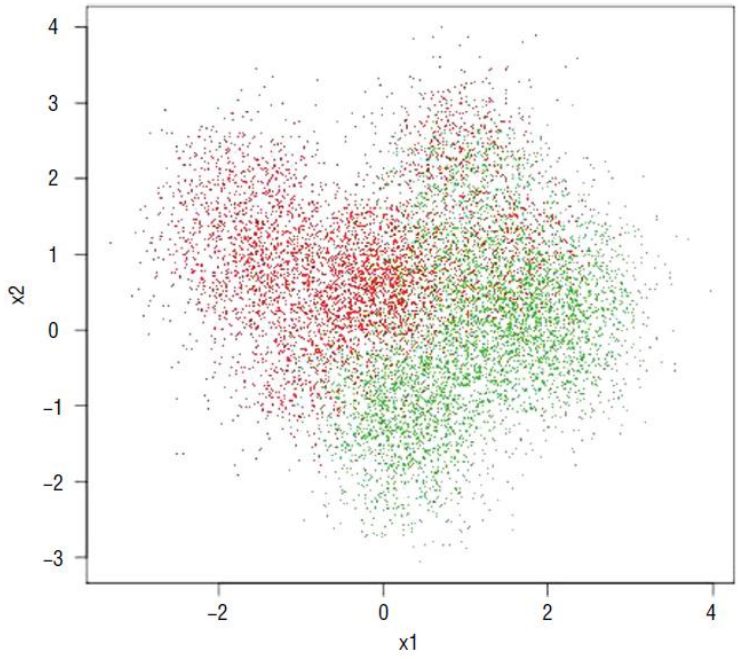
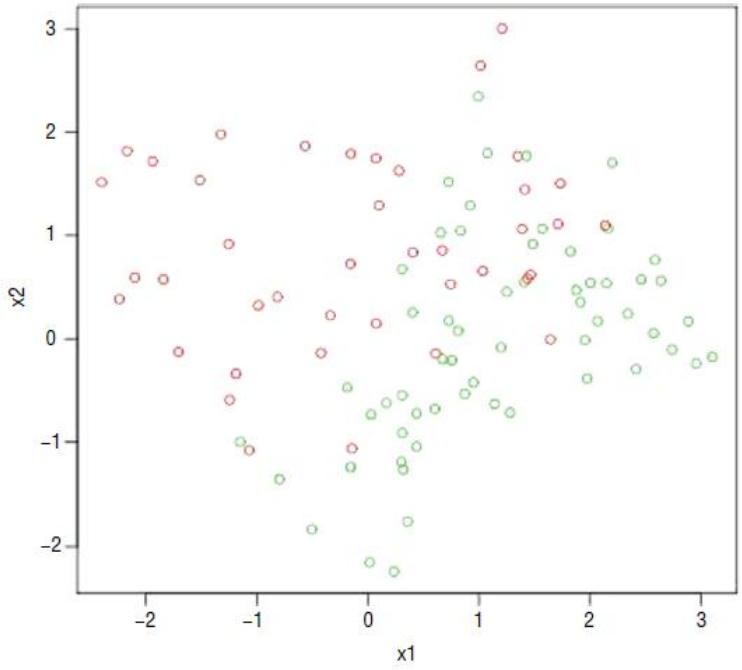


***Figure – 2 A Complicated Classification Problem ***

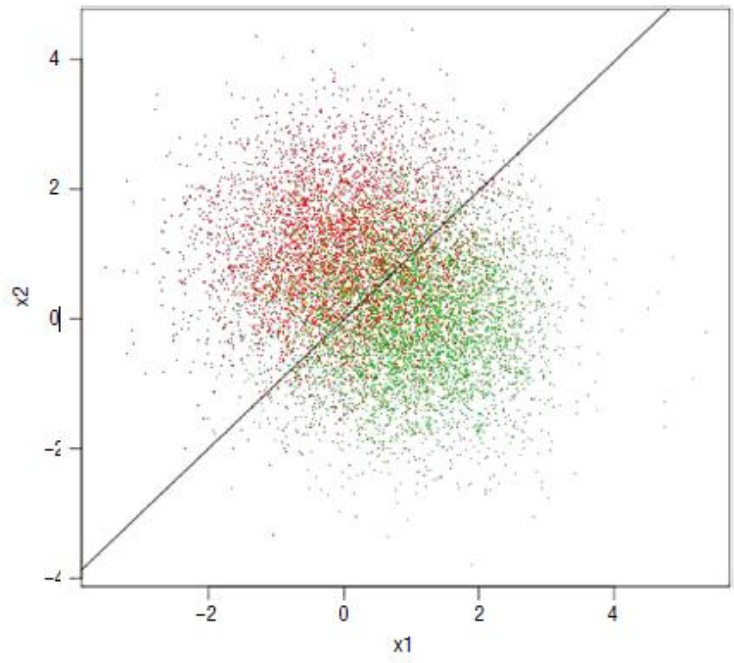
- However, complexity of the decision boundaries is not the only factor influencing whether linear or nonlinear methods will deliver better performance. Another important factor is the size of the data set.
- Following figure illustrates this element of performance. The points plotted in the figure are a 1 percent subsample of data plotted in above figure.

***Figure – 3 A Complicated Classification Problem W/o much data ***



- In Figure - 2, there was enough data to visualize the curved boundaries delineating (explaining) the sets of light and dark points.
- Without as much data, the sets are not so easily discerned (differentiated) visually, and in this circumstance, a linear model may give equal or better performance than a nonlinear model.
- However, if the model is not complicated, as in Figure - 1, or there is not sufficient data, as in Figure - 3, a linear model may produce the best answer.

- Contrast Between a Simple Model and a Complex Model:
  - The previous section showed visual comparison between simple and complex problems. This section describes how the various models available to solve these problems differ from one another.
  - Intuitively (Naturally), it seems that a complex model should be fit to a complex problem, but the visual example from the above section demonstrates that data set size may dictate that a simple model fits a complex problem better than a complex model.



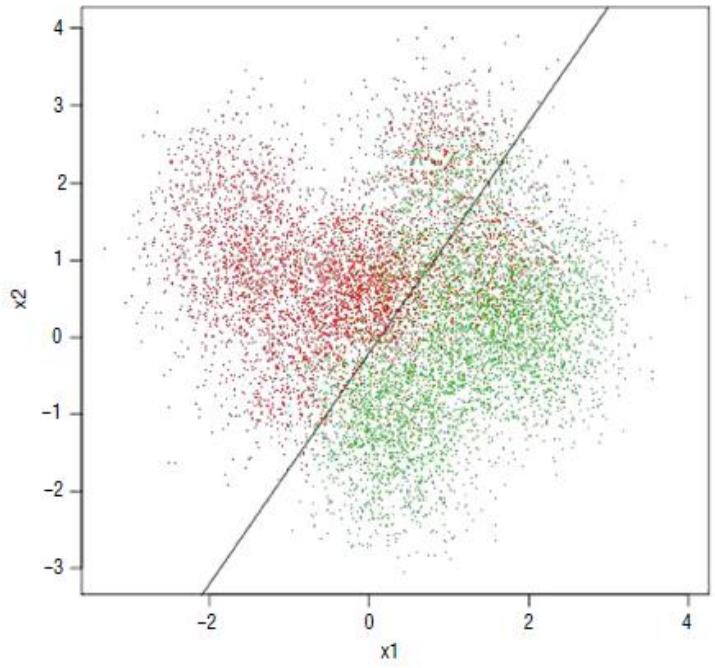
*** A Linear Model fit to simple data ***

- Above figure shows a linear model fit to the simple problem. The model was generated using the *glmnet* algorithm. The linear model fit

to these data divides the data roughly in half. The line in the above figure is given by:

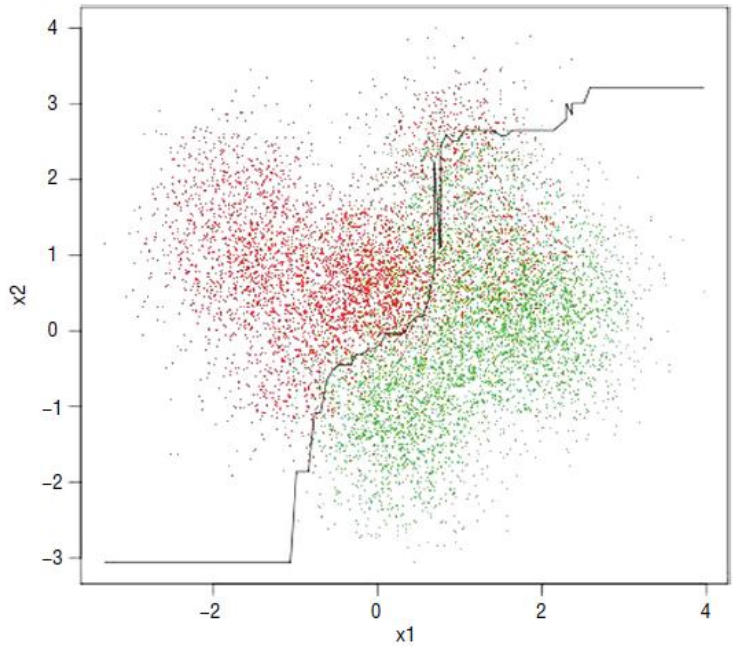
$$x_2 = -0.01 + 0.99x_1$$

- A more complicated problem with more complicated decision boundaries gives a complicated model an opportunity to outperform a simple linear model.



*** A Linear Model fit to complex data ***

- Above figure shows a linear model fit to data indicating a nonlinear decision boundary. In these circumstances, the linear model misclassifies regions as dark when they should be light and vice versa.
- Following figure- A shows how much better a complicated model can do with complicated data. The model used to generate this decision boundary is an ensemble (collection) of 1,000 binary decision tree constructed using the gradient boosting algorithm. The nonlinear decision boundary curves are used to better delineate regions where the dark points are denser and regions where the light points are denser.



*** Figure – A: Ensemble Model fit to complex data ***

- Factors Driving Predictive Algorithm Performance:
  - These results explain the excitement over large volumes of data.
  - Accurate predictions for complicated problems require large volumes of data.
  - But the size isn't quite a precise enough measure.
  - The shape of the data also matters.
- Choosing an Algorithm: Linear or Nonlinear?
  - The visual examples you have just seen give some idea of the performance tradeoffs (swaps) between linear and nonlinear predictive models.
  - Linear models are preferable when the data set has more columns than rows or when the underlying problem is simple.
  - Nonlinear models are preferable for complex problems with many more rows than columns of data.
  - An additional factor is training time. Fast linear techniques train much faster than nonlinear techniques.
  - Choosing a nonlinear model (say an ensemble method) entails training a number of different models of differing complexity.

- For example, the ensemble model that generated the decision boundary in Figure - A was one of roughly a thousand different models generated during the training process. These models had a variety of different complexities. Some of them would have given a much cruder approximation to the boundaries that are visually apparent in Figure - A.
- This section has used data sets and classifier solutions that can be visualized in order to give you an intuitive grasp of the factors affecting the performance of the predictive models you build. Generally, you'll use numeric measures of performance instead of relying on pictures.
- The next section describes the methods and considerations for producing numeric performance measures for predictive models and how to use these to estimate the performance your models will achieve when deployed.

- Measuring the Performance of Predictive Models:
  - This section covers two broad areas relating to performance measures for predictive models.
  - The first one is the different metrics that you can use for different types of problems (for example, using MSE for a regression problem and misclassification error for a classification problem). In the literature (and in machine learning competitions), you will also see measures like receiver operating curves (ROC curves) and area under the curve (AUC). Besides that, these ideas are useful for optimizing performance.
  - The second broad area consists of techniques for gathering out-of-sample error estimates. Recall that out-of-sample errors are meant to simulate errors on new data. It's an important part of design practice to use these techniques to compare different algorithms and to select the best model complexity for a given problem complexity and data set size.
  - Performance Measures for Different Types of Problems:

- Performance measures for regression problems are relatively straightforward.
- In a regression problem, both the target and the prediction are real numbers.
- Error is naturally defined as the difference between the target and the prediction.
- It is useful to generate statistical summaries of the errors for comparisons and for diagnostics.
- The most frequently used summaries are the Mean Squared Error (MSE) and the Mean Absolute Error (MAE).

- Simulating Performance of Deployed Models:
  - The examples from the preceding section demonstrated the need for testing performance on data not included in the training set to get a useful estimate of expected performance once a predictive model is deployed.
  - The example broke the available labeled data into two subsets. One subset, called the training set, contained approximately two-thirds of the available data and was used to fitting an ordinary least squares model.
  - The second subset, which contained the remaining third of the available data, was called the test set and was used only for determining performance (not used during training of the model).
  - This is a standard procedure in machine learning.
  - Test set sizes range from 25 percent to 35 percent of the data, although there aren't any hard-and-fast rules about the sizes.
  - One thing to keep in mind is that the performance of the trained model deteriorates (fails) as the size of the training data set shrinks.
  - Taking out too much data from the training set can prove detrimental (harmful) to end performance.

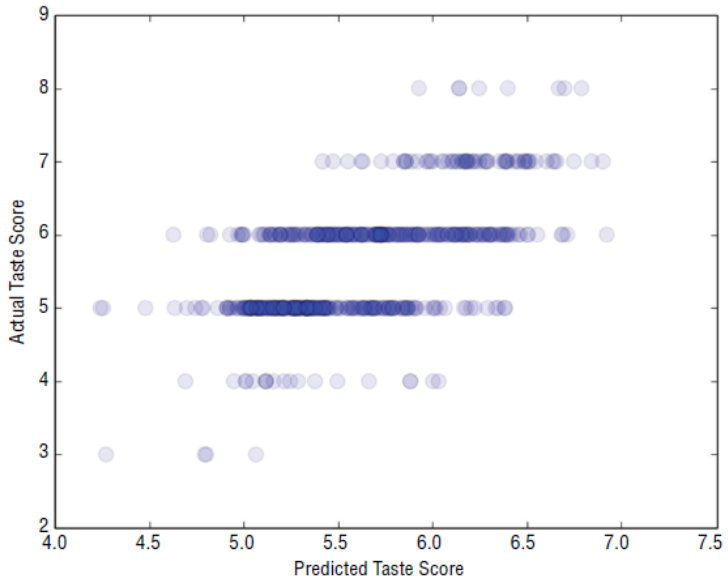
Block 1	Block 2	Block 3	Block 4	Block 5
Block 1	Block 2	Block 3	Block 4	Block 5
N – fold cross - validation				

- Another approach to holding out data is called n-fold cross-validation. Above Figure shows schematically how a data set is divided up for training and testing with n-fold cross-validation.
- The set is divided into n disjointed sets of roughly equal sizes. In the figure, n is 5. Several training and testing passes are made through the data.
- In the first pass, the first block of data is held out for testing, and the remaining n-1 are used for training.
- In the second pass, the second block is held out for testing, and the other n-1 are used for training. This process is continued until all the data have been held out (five times for the five-fold example depicted in Figure).
- The n-fold cross-validation process yields an estimate of the prediction error and has several samples of the error so that it can estimate error bounds on the error. It can keep more of the data in the training set, which generally gives lower generalization errors and better final performance.
- For example, if the 10-fold cross-validation is chosen, then only 10 percent of the data is held out for each training pass. These features of n-fold cross-validation come at the expense of taking more training time.
- The approach of taking a fixed holdout set has the advantage of faster training, because it employs only one pass through the training data.
- After a model has been trained and tested, it is good practice to recombine the training and test data into a single set and retrain the model on the larger data set.
- This section supplied you with tools to quantify the performance of your predictive model.

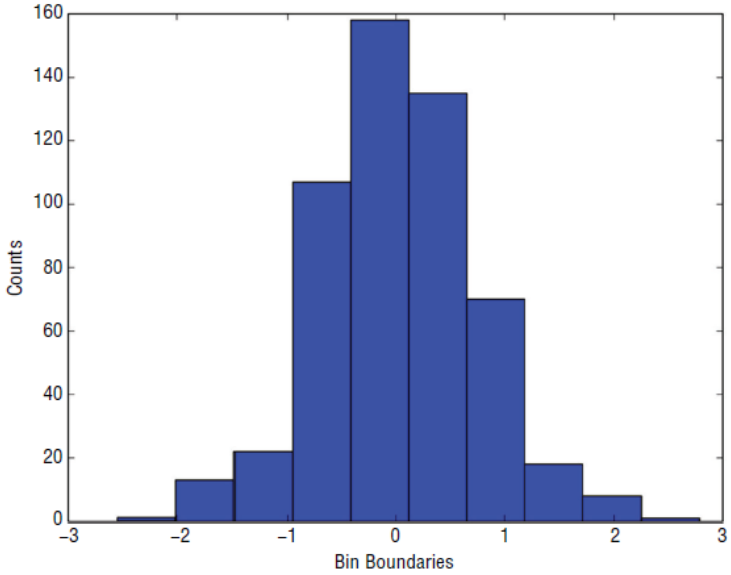
- Achieving Harmony Between Model and Data:
    - This section uses ordinary least squares (OLS) regression to illustrate several things.
    - First, it illustrates how OLS can sometimes overfit a problem. Overfitting means that there's a significant discrepancy between errors on the training data and errors on the test data.
    - Second, it introduces two methods for overcoming the overfit problem with OLS.
    - These methods will cultivate (help) your intuition (perception).
    - In addition, the methods for overcoming overfitting have a property that is common to most modern machine learning algorithms.
    - Modern algorithms generate a number of models of varying complexity and then use out-of-sample performance to balance model complexity, problem complexity, and data set richness and thus determine which model to deploy.
    - Fortunately, there's been a lot of work on ordinary least squares regression since its invention more than 200 years ago by Gauss and Legendre.
    - This section introduces two of the methods for adjusting the throttle on ordinary least squares regression. One is called *forward stepwise regression*; the other is called *ridge regression*.
    - Evaluating and Understanding Predictive Model:
        - Several other plots are helpful in understanding the performance of a trained algorithm and can point the way to making improvements in its performance.
        - Figure - A shows a scatter plot of the true labels plotted versus the predicted labels for points in the test set. The scatter plot shows horizontal rows of points.
        - When the true values take on a small number of values, it is useful to make the data points partially transparent so that the darkness can indicate the accumulation of many points in one area of the graph.
        - Actual taste scores of 5 and 6 are reproduced fairly well. The more extreme values are not as well predicted by the system.

- Figure - B shows a histogram of the prediction error for forward stepwise prediction predicting wine taste scores. Sometimes the error histogram will have two or more discrete peaks (heights).

*** Figure – A: Actual Taste Scores v/s. predictions generated with forward stepwise regression ***



*** Figure – B: Histogram of wine taste prediction error with forward stepwise regression ***



- Perhaps it will have a small peak on the far right or far left of the graph.

- In that case, it may be possible to find an explanation for the different peaks in the error and to reduce the prediction error by adding a new attribute that explains the membership in one or the other of the groups of points.
- You want to note several things about this output. First, let's reiterate the process. The process is to train a family of models (in this case, ordinary linear regression trained on column-wise subsets of X).
- The series of models is parameterized (in this case, by the number of attributes that are used in the linear model).
- The model to deploy is chosen to minimize the out-of-sample error.
- The number of attributes to be incorporated in the solution can be called a complexity parameter. Models with larger complexity parameters have more free parameters and are more likely to overfit the data than less-complex models.
- Also note that the attributes have become ordered by their importance in predicting quality.
- In the list of column numbers and the associated list of attribute names, the first in the list is the first attribute chosen, the second was next, and so on. The attributes used come out in a nice ordered list.
- This is an important and desirable feature of a machine learning technique.
- Early stages of a machine learning task mostly involve hunting for (or constructing) the best set of attributes for making predictions. Having techniques to rank attributes in order of importance helps in that process. The other algorithms developed in this book will also have this property.
- The last observation regards picking a model from the family that machine learning techniques generate.
- The more complicated the model, the less well it will generalize. It is better to err on the side of a less-complicated model.
- The earlier example indicates that there's very little degradation in performance between the 9th (best) model and the 10th model (a

change in the 4th significant digit). Best practice would be to remove those attributes even if they were better in the 4th significant digit in order to be conservative.

Summary

- This chapter provided visual demonstrations of problem complexity and model complexity and discussed how those factors and data set sizes combine to determine classifier performance on a given problem.
- The discussion then turned to a number of different metrics for prediction performance associated with the different problem types (regression, classification, and multiclass classification) that arise as part of the function approximation problem.
- The chapter described n-fold cross-validation for estimating performance on new data.
- The chapter introduced the conceptual framework that a machine learning technique produces a parameterized family of models and that one of these is selected for deployment on the basis of out-of-sample performance.
- Several examples based on modifications of ordinary least squares regression (forward stepwise regression and ridge regression) then instantiated that conceptual framework.