Chapter 15

Model Quality and the Bias-Variance Tradeoff

We have now seen several different algorithms (and meta-algorithms) for supervised learning. We've examined linear models for classification and regression (Chapters 8 and 9), KNN (Chapters 2 and 3), decision trees (Chapters 7), and ensemble methods – usually based on decision trees – including random forests (Chapter 13) and boosting (Chapter 14). In this short chapter, we'll take a step back and talk about some key themes that are relevant to all of these methods.

15.1 Measuring Error: Loss and Objective Functions

Most of us think of error as a fairly intuitive concept, and so far in our discussions of model building and model quality, we've avoided any formal definitions. However, as we begin to extend the concepts we've learned from classification and regression to more challenging problem classes (e.g., survival analysis), a more formal conception of what error means and how to minimize it algorithmically becomes increasingly crucial. Here are some examples:

- In classification, error typically means either the total number of mis-

classified points,

$$\text{error} = \sum_{i=1}^{n} \mathcal{I}\left(y^{(i)} \neq \hat{y}^{(i)}\right),$$

or a weighted average of the number of misclassified points per class. Another way of quantifying classification error is **AUC**, the area under the receiver operating characteristic curve, which is the probability that a randomly chosen positive example (where $y = 1$) will receive a higher score from the model (higher $\hat{y}$) than a randomly chosen negative example (where $y = 0$).

- In regression, error typically means the **mean-squared error (MSE)**,

$$\text{error} = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2,$$

although there are also other ways of quantifying error. For example, you could use the **mean absolute error**

$$\text{error} = \frac{1}{n} \sum_{i=1}^{n} |y^{(i)} - \hat{y}^{(i)}|.$$

Neither is "wrong" or "right", but they have different properties. For example, the MSE gives a higher weight to large errors (outliers). It's also differentiable, which is why it's used so much more often in, e.g., neural networks than the mean absolute error.

- In survival analysis, error is typically quantified using something called **Harrell's concordance index**, which takes into account censored observations[1].

Question 15.1

Harrell's concordance (C) index is probably unfamiliar to you. It is calculated like this:

[1]For a detailed explanation and some experimental results, see Kattan MW, Hess KR and Beck JR (1998). "Experiments to determine whether recursive partitioning (CART) or an artificial neural network overcomes theoretical limitations of Cox proportional hazards regression." *Computers and Biomedical Research*, 31(5), 363-373.

1. Create a list of all possible pairs of patients. There will be $n(n-1)/2$ pairs.
2. Eliminate all pairs for which the patient with the shorter follow-up time does not experience the event of interest (i.e., is censored). The remaining patient pairs are considered "usable" since the patient with the shorter time-to-event is identifiable.
3. Count the number of usable patient pairs for which the patient with the shorter follow-up time had the higher predicted hazard for the event. That is, you want the number of pairs for which the model's predictions about relative times to event are consistent with the observed data.
4. The *C* statistic is the number of consistent pairs divided by the number of usable pairs. The error is defined as $1 - C$.

Here are four patients and the predicted mean time to event from two different survival models (low value means lower time to event). All times are in years. Calculate the *C* statistic for both models.

Patient	Follow-up Time	Observed?	Model 1 Score	Model 2 Score
1	8.3	1	4.6	5.2
2	6.5	0	2.3	7.1
3	2.7	1	0.6	6.7
4	7.4	1	4.7	6.6

First Patient	Second Patient	Usable	Model 1 Consistent	Model 2 Consistent
1	2			
1	3			
1	4			
2	3			
2	4			
3	4			

You should find that the *C* statistic for Model 1 is 0.75, while the *C* statistic for Model 2 is only 0.25. Model 1 is clearly the better model.

The definition of error used to optimize a particular model is called a **loss function**. Loss functions are a general concept from optimization and

decision theory; they represent the cost associated with a decision. If we think of a supervised learning model as an engine for making decisions, the loss is how "bad", on average, those decisions will be. Loss functions are, in turn, part of a broader class of functions called **objective functions**. Most learning algorithms work by either minimizing some measure of "badness" (a loss function) or maximizing some measure of "goodness" (negative of the loss, alternatively called a **reward function**).

Question 15.2

Imagine what would happen if, in Question 15.1, we simply set a follow-up time of 7 years and treated the problem as a classification problem, calculating the AUC for the two models based on that time horizon and ignoring censoring.

Positive Patient	Negative Patient	Model 1 Ranks Pos Higher?	Model 2 Ranks Pos Higher?
3	1	1	0
3	2	1	1
3	4	1	1

We would calculate an AUC of 1.0 for Model 1 and 0.67 for Model 2. Do you think this is a good approach to quantifying error for this problem? Why or why not?

The choice of how to define error, and which loss function to use for quantifying that error, ultimately rests with the model builder. The process of model training is about adjusting the available parameters of the model to minimize the loss.

15.2 Goodness of Fit vs. Generalizability

Once we have an appropriate objective function, we can set about building a model to optimize it. This requires us to think about what constitutes a "good model". It's a bit more complicated than simply minimizing the loss on our training data, because ideally we want a model that is both accurate

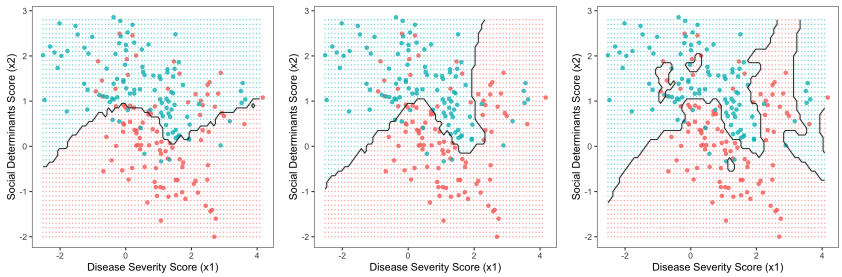
and **parsimonious**, meaning that it is as simple as possible without sacrificing performance. Another way of thinking about this is that we create models to tell stories about the data we see. If they are good stories, they will do two things:

1. Explain the structure of the data that are used to train them (high **goodness of fit**)
2. Make accurate predictions on new data (good **generalizability**)

For a supervised learning model, we quantify (1) using the **training error** (error on the training set) and (2) using the **test error** (error on an independent test set).

Question 15.3

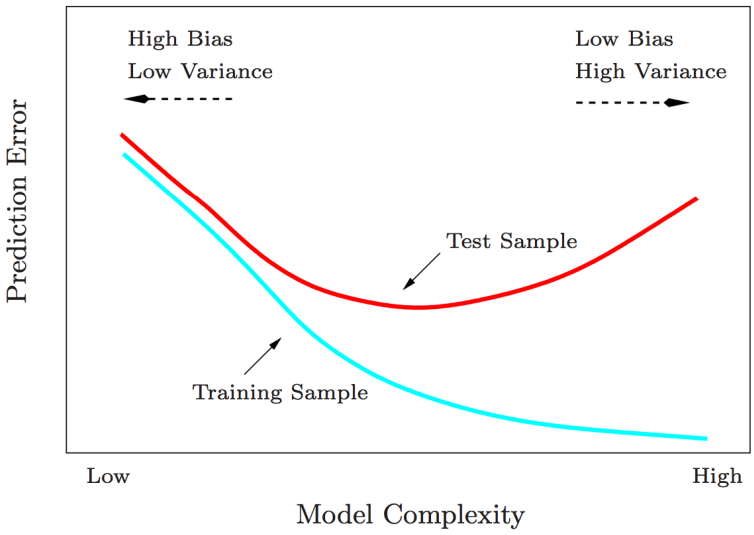
Here we see three decision boundaries for KNN with different values of K (the number of neighbors considered in making a prediction). The data are for the two-class classification problem first discussed in Chapter 2. From left to right, $K = 50, 15$, and 3. What are the tradeoffs in moving from left to right in terms of (a) training error/goodness of fit and (b) test error/generalizability?



In supervised learning, **model complexity** (loosely defined as the effective number of parameters the model must fit) is, therefore, a very important consideration. A model that is not complex enough will fail to capture all of the structure in the training data, and both its training and test error will suffer as a result. We call this situation **underfitting**. Conversely, a model that is too complex may fit the training data very well – maybe perfectly – but will fail to generalize well to new data. We call this situation **overfitting**.

15.3 Bias vs. Variance

It turns out that there is a general principle governing model complexity in supervised learning called the **bias-variance tradeoff**. It is perhaps best illustrated by this figure, which comes from the excellent (and free) book *Elements of Statistical Learning*, by Tibshirani, Hastie, and Friedman (Figure 2.11):



The figure shows us that test error, our measurement of generalization error, comes from two different sources:

$$\text{test error} = \text{bias} + \text{variance}.$$

The term **bias** refers to error that results from underfitting, while **variance** results from overfitting.

Error due to bias can often be reduced by introducing more/different features or by using a different learning algorithm. Error due to variance can often be reduced by increasing the size or diversity of the training data. That's why it's important to know which situation you're in; gathering more training data when your model is too simple is unlikely to help you, and deploying an extremely fancy (and complicated) deep learning model when

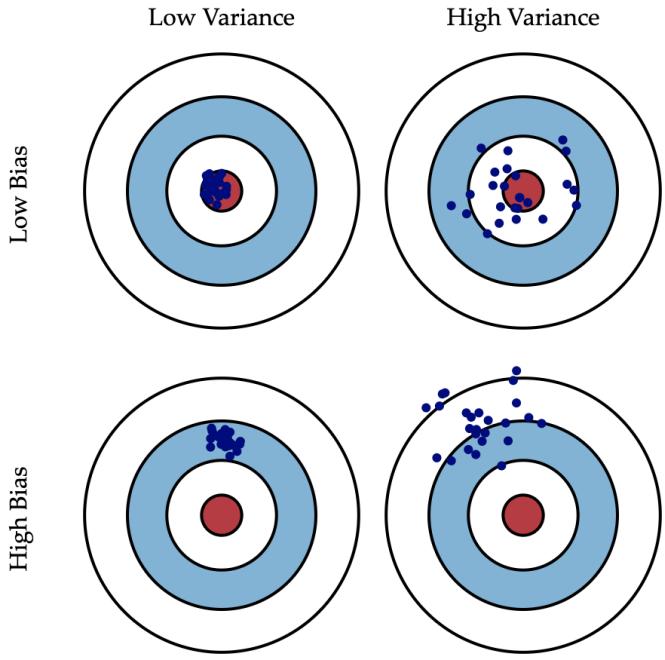
logistic regression has already overfit your training data is also unlikely to help.

Another consideration is that even the "sweet spot" (minimum test error) on the bias-variance tradeoff curve can still be high. This occurs when the data you have available simply aren't sufficient to answer the question – maybe you have the wrong features or there is no relationship between the features and the outcome. Or maybe your features aren't measured correctly. In my experience, these types of issues are not considered often enough in the clinical domain.

Question 15.4

A recent review article had the provocative title "A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models." (Christodoulou E, Ma J, Collins GS, Steyerberg EW, Verbakel JY, Van Calster B. *Journal of Clinical Epidemiology*, 2019, 110:12-22). Assuming this finding is true, what is it telling us about the nature of the error in most clinical risk prediction models? What does it suggest about what we should be doing to improve the performance of these models?

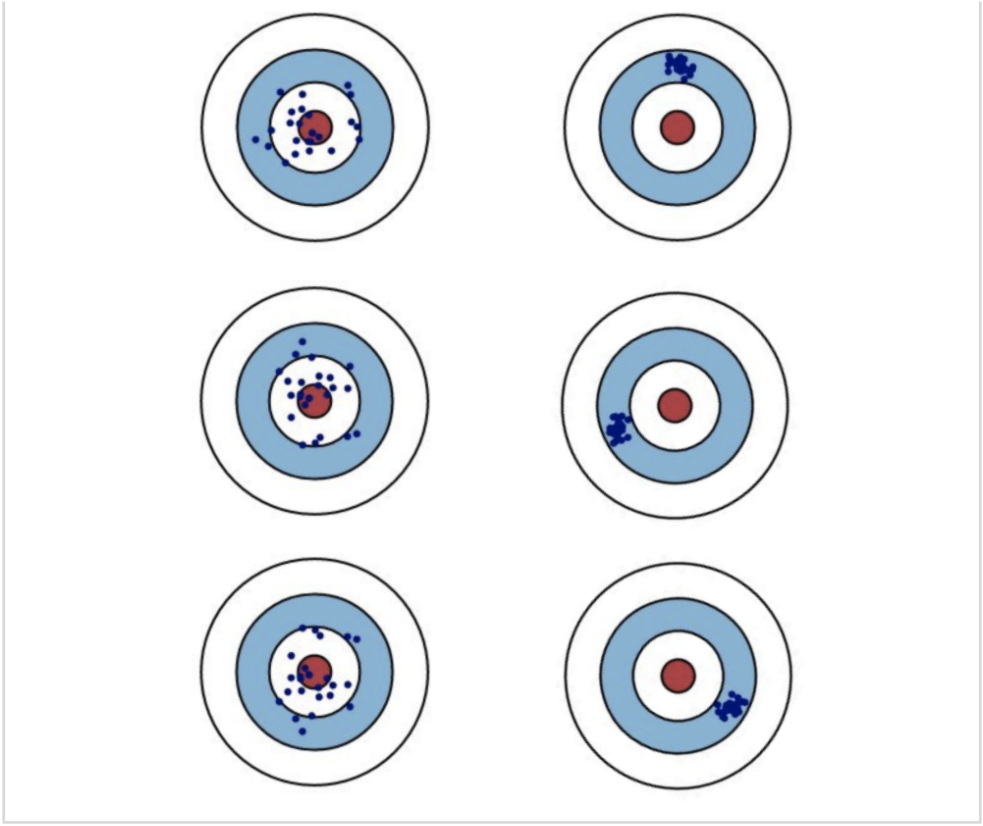
Another way of thinking about bias and variance is in terms of what happens when you make slight changes to the training data (for example, by collecting new data from the same patient population, or drawing repeated bootstrap samples from the same training set). The figure below is from the article *Understanding the Bias-Variance Tradeoff*, by Scott Fortmann-Roe:



Think of each dot as representing a single test example evaluated under the same model trained on slightly different datasets. The center of the target is the prediction the model should make for that test example. In the case of high bias and low variance, all of the models are off, but they are "wrong in the same way". Even if you average their predictions, the answer is still way off the mark. In the case of high variance, the models all make incorrect predictions, but their predictions are off in different directions. As a result, if you average their outputs, you'll get closer to the right answer.

Question 15.5

We saw random forests in Chapter 13 and were introduced to boosting in Chapter 14. It is interesting to compare the two methods because they reduce test error in different ways: one primarily tackles variance, while the other primarily tackles bias. Which one is which, and how does each algorithm succeed in reducing its respective form of error? Hint: The image below may help. One column represents three trees from a random forest; the other represents three trees from a boosting model.



Chapter 16

Feature Selection

Modern clinical datasets tend to suffer from an overabundance of features. In fact, there are often more available features than there are training examples. Not all of these features will contribute equal information about the outcome. Including dozens or hundreds of predictors in a supervised learning model does not guarantee higher accuracy; in fact, it is more likely to lead to models that are unnecessarily complex and overfit (see Chapter 15). Even when features are related to the outcome, they may contribute information that is redundant with other features in the study.

In cases like these, the model designer will either need to choose a subset of features manually or incorporate some form of **feature selection**: a process that automatically or semi-automatically decides which features are most relevant to the model and discards the others. The goal of feature selection is to remove useless and redundant features in a principled way.

16.1 Example: The Pima Indians Dataset

The so-called "Pima Indians diabetes dataset" was collected in the 1980s. It includes information on 768 women from the Pima people, who live near Phoenix, Arizona. The Pima were, as of the late 1980s, under continuous study by the National Institute of Diabetes and Digestive and Kidney Diseases

because of their high incidence of diabetes[1]. There are eight predictors in the dataset and one outcome. The predictors are:

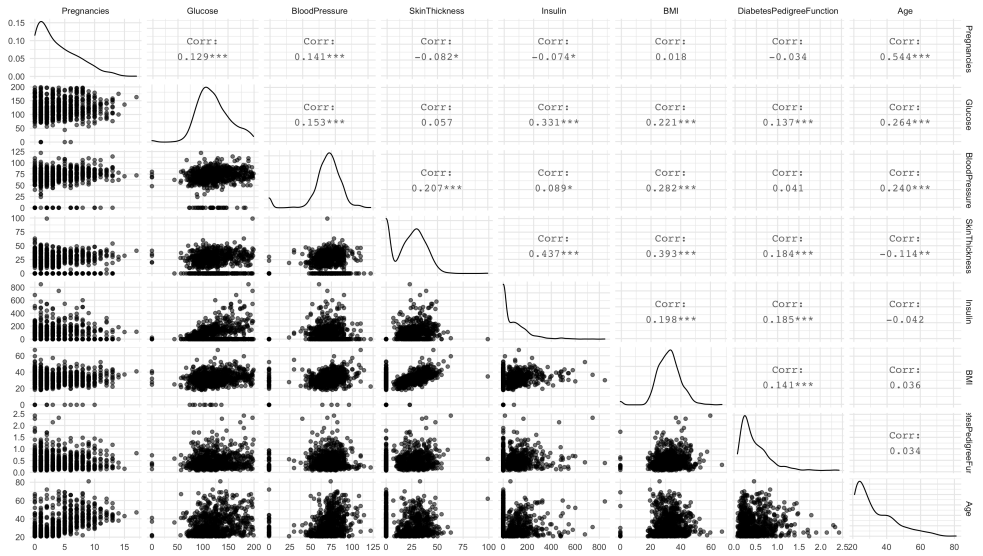
Predictor	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration in a two-hour oral glucose tolerance test
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skin fold thickness (mm)
Insulin	Two-hour serum insulin (μU/mL)
BMI	Body mass index (weight in kg/(height in m)^2)
DiabetesPedigreeFunction	Diabetes pedigree function (developed by research team; described in paper)
Age	Age in years

The outcome is whether or not the woman went on to develop type II diabetes within 5 years from the time of the survey.

16.2 Correlograms

For datasets with a manageable number of features, A good way to alert oneself to the presence of highly correlated predictors is to create a **correlogram**, or scatterplot matrix, which looks at associations between all pairs of variables. A correlogram for the Pima dataset is below.

[1]The causative factors behind this high diabetes rate are not clear. Some scholars believe that it was driven by a sudden shift in diet during the last century from traditional agricultural crops to processed foods, together with a decline in physical activity L. O. Schulz, P. H. Bennett, E. Ravussin, J. R. Kidd, K. K. Kidd, J. Esparza, and M. E. Valencia. "Effects of traditional and western environments on prevalence of type 2 diabetes in Pima Indians in Mexico and the US". in: *Diabetes Care* 29.8 (2006), pp. 1866–1871.



Question 16.1

This correlogram quantifies correlation using a metric called the **Pearson correlation coefficient**. Which pairs of predictors are the most tightly correlated? Are they positively or negatively correlated? How might you modify your dataset to eliminate redundancies in the information contributed by the different predictors?

16.3 Univariate vs. Multivariate Models

The presence of correlations will affect different types of models in different ways, and some suffer more than others. In the clinical research literature, the standard approach to assessing and accounting for correlations is to start with **univariate** models, in which each predictor's association with the outcome is studied on its own. Those predictors that display some association with the outcome are then incorporated into larger **multivariate** models.

Here are the results of eight univariate logistic regression models that capture the effect of each predictor in the Pima dataset on the outcome of diabetes vs. no diabetes. The coefficients on each predictor are called the **unadjusted** coefficients, and the p-values on the predictor-specific hypothesis

tests are called unadjusted p-values. The exponentiated coefficients are called unadjusted odds ratios[2].

Predictor	Unadjusted Coefficient	Unadjusted Odds Ratio	Unadjusted P-value
Pregnancies	0.137	1.147	<0.001
Glucose	0.038	1.039	<0.001
BloodPressure	0.007	1.007	0.073
SkinThickness	0.010	1.010	0.039
Insulin	0.002	1.002	<0.001
BMI	0.094	1.100	<0.001
DiabetesPedigreeFunction	1.083	2.953	<0.001
Age	0.042	1.043	<0.001

Now, here is a multivariate logistic regression model that includes all eight predictors. The coefficients, exponentiated coefficients, and p-values are often called **adjusted**.

Predictor	Adjusted Coefficient	Adjusted Odds Ratio	Adjusted P-value
Pregnancies	0.123	1.131	<0.001
Glucose	0.035	1.036	<0.001
BloodPressure	-0.013	0.987	0.011
SkinThickness	0.001	1.001	0.929
Insulin	-0.001	0.999	0.186
BMI	0.090	1.094	<0.001
DiabetesPedigreeFunction	0.945	2.573	0.002
Age	0.015	1.015	0.111

Alternatively, one might say that the odds ratios here measure the effect of each predictor, **controlling for** the effects of the other predictors.

[2]See Chapter 9 if you don't understand why you're exponentiating or where the term "odds ratio" comes from. The odds ratio compares the odds of having a positive outcome among two groups separated by a one unit difference of the predictor in question, all else being the same.

Question 16.2

How can the odds ratio for Insulin be so close to 1.0 yet its p-value so low? (Hint: See Section 12.5.)

Question 16.3

Why might the coefficient and p-value for SkinThickness change so much in the shift from unadjusted to adjusted?

16.4 Filter Methods

The approach of creating univariate models and then incorporating the best-performing predictors into multivariate models is part of a broader class of feature selection methods called **filter methods**. Filter methods select subsets of variables as a preprocessing step, independently of the supervised learning model that will eventually be implemented. These methods use **proxy measures** to rank variables; the proxy measure is often chosen to be computationally fast so that large numbers of features can be sifted through quickly.

A predetermined threshold of the proxy measure is usually used to determine which features pass to the multivariate modeling stage. Alternatively, the modeler may decide on a fixed number of features to include. Some examples of filter methods include:

- Any kind of univariate model (e.g. univariate logistic or linear regression)
- Any kind of hypothesis test (e.g. t-test, chi-squared test; see Chapter 6)
- Any kind of correlation coefficient (e.g. Pearson, Spearman)

- Mutual information[3]

$$MI(X_i, Y) = \sum_x \sum_y P(X_i = x, Y = y) \log \frac{P(X_i = x, Y = y)}{P(X_i = x)P(Y = y)}$$

- Variance thresholding (simply remove features with low variance)

Question 16.4

If you wanted to use the univariate logistic regression models above in Section 16.3 as a filter for a downstream model (potentially not even multivariate logistic regression - it could be a decision tree, etc.), how would you rank them and how would you decide on an appropriate cutoff?

Question 16.5

How would you apply a filter-based selection method in a case where you had dozens of different predictors of different types (e.g. some categorical, some binary, some numeric)?

Question 16.6

How might you choose the appropriate threshold for a filter-based method in a data-driven way?

Question 16.7

What is problematic about testing each potential feature, one at a time?

16.5 Wrapper Methods

Filter methods are just one possible approach to feature selection. An alternative to filter methods are **wrapper methods**. These methods use a search algorithm to traverse the space of possible features, evaluating each subset

[3]The mutual information, in another format, is the most common splitting criterion used for decision trees; see Chapter 7. In the case of continuous variables, the sums are replaced by integrals.

by running the chosen model using that subset. They are generally computationally intensive (e.g., imagine trying to find the optimal subset of 10,000 features, or even 50) so **heuristics** generally have to be used to pare down the search space. Some examples of wrapper methods include:

- **Exhaustive search.** Try all possible subsets of features. If there are m features, this means trying 2^m possible subsets.
- **Forward selection.** Start with a baseline (e.g., intercept only) model. Add in each of m possible predictors individually and take the best one based on some performance criterion. Repeat, adding one predictor at each step, until the performance criterion stops getting better or you run out of predictors.
- **Backward elimination.** Start with a complete model (all predictors included). Try removing each predictor and take the one whose removal causes the performance criterion to increase the most. Repeat, removing one predictor at each step, until the performance criterion stops getting better or you are left with no predictors (null model).
- **Forward-backward selection.** A combination of forward selection and backward elimination.
- **Simulated annealing.** Add or remove predictors with some probability depending on how well the model is doing. At each stage, if the new model is better, accept it; it becomes the new baseline. If the new model is worse, accept it with some probability, p, that decreases over time according to a "cooling schedule". This helps prevent the variable selection process from getting stuck in local optima.

Question 16.8

Why is exhaustive search problematic for almost any reasonably sized m?

Question 16.9

Here is the output of forward selection for the Pima example, using R's *MASS* package and the **Akaike Information Criterion (AIC)** as the model performance metric.

```
Start:  AIC=995.48
Outcome ~ 1
                           Df Deviance    AIC
+ Glucose                   1   808.72 812.72
+ BMI                       1   920.71 924.71
+ Age                       1   950.72 954.72
+ Pregnancies               1   956.21 960.21
+ DiabetesPedigreeFunction  1   970.86 974.86
+ Insulin                   1   980.81 984.81
+ SkinThickness             1   989.19 993.19
+ BloodPressure             1   990.13 994.13
<none>                          993.48 995.48

Step:  AIC=812.72
Outcome ~ Glucose

                           Df Deviance    AIC
+ BMI                       1   771.40 777.40
+ Pregnancies               1   784.95 790.95
+ DiabetesPedigreeFunction  1   796.99 802.99
+ Age                       1   797.36 803.36
<none>                          808.72 812.72
+ SkinThickness             1   807.07 813.07
+ Insulin                   1   807.77 813.77
+ BloodPressure             1   808.59 814.59

Step:  AIC=777.4
Outcome ~ Glucose + BMI

                           Df Deviance    AIC
+ Pregnancies               1   744.12 752.12
+ Age                       1   755.68 763.68
+ DiabetesPedigreeFunction  1   762.87 770.87
+ Insulin                   1   767.79 775.79
+ BloodPressure             1   769.07 777.07
<none>                          771.40 777.40
+ SkinThickness             1   770.20 778.20

Step:  AIC=752.12
Outcome ~ Glucose + BMI + Pregnancies

                           Df Deviance    AIC
+ DiabetesPedigreeFunction  1   734.31 744.31
+ BloodPressure             1   738.43 748.43
```

```
+ Age                      1   742.10 752.10
<none>                         744.12 752.12
+ Insulin                  1   742.43 752.43
+ SkinThickness            1   743.60 753.60

Step:  AIC=744.31
Outcome ~ Glucose + BMI + Pregnancies +
          DiabetesPedigreeFunction

               Df Deviance    AIC
+ BloodPressure  1   728.56 740.56
+ Insulin        1   731.51 743.51
<none>               734.31 744.31
+ Age            1   732.51 744.51
+ SkinThickness  1   733.06 745.06

Step:  AIC=740.56
Outcome ~ Glucose + BMI + Pregnancies +
          DiabetesPedigreeFunction +
          BloodPressure

               Df Deviance    AIC
+ Age            1   725.46 739.46
+ Insulin        1   725.97 739.97
<none>               728.56 740.56
+ SkinThickness  1   728.00 742.00

Step:  AIC=739.46
Outcome ~ Glucose + BMI + Pregnancies +
          DiabetesPedigreeFunction +
          BloodPressure + Age

               Df Deviance    AIC
+ Insulin        1   723.45 739.45
<none>               725.46 739.46
+ SkinThickness  1   725.19 741.19

Step:  AIC=739.45
Outcome ~ Glucose + BMI + Pregnancies +
          DiabetesPedigreeFunction +
          BloodPressure + Age + Insulin

               Df Deviance    AIC
<none>               723.45 739.45
```

```
+ SkinThickness  1   723.45 741.45
```

What does the final model look like? Which predictor is missing from the final model? Note: AIC is an estimate of out-of-sample prediction error and depends on the likelihood; thus it does not work for models that do not calculate some form of likelihood.

Question 16.10

Here is the output of backward selection for the Pima example, again using R's *MASS* package and AIC as the model performance metric.

```
Start:  AIC=741.45
Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
    Insulin + BMI + DiabetesPedigreeFunction + Age

                          Df Deviance    AIC
- SkinThickness            1   723.45 739.45
- Insulin                  1   725.19 741.19
<none>                         723.45 741.45
- Age                      1   725.97 741.97
- BloodPressure            1   729.99 745.99
- DiabetesPedigreeFunction 1   733.78 749.78
- Pregnancies              1   738.68 754.68
- BMI                      1   764.22 780.22
- Glucose                  1   838.37 854.37

Step:  AIC=739.45
Outcome ~ Pregnancies + Glucose + BloodPressure + Insulin + BMI +
    DiabetesPedigreeFunction + Age

                          Df Deviance    AIC
<none>                         723.45 739.45
- Insulin                  1   725.46 739.46
- Age                      1   725.97 739.97
- BloodPressure            1   730.13 744.13
- DiabetesPedigreeFunction 1   733.92 747.92
- Pregnancies              1   738.69 752.69
- BMI                      1   768.77 782.77
- Glucose                  1   840.87 854.87
```

What does the final model look like? How does it compare to the model obtained through forward selection?

16.6 Embedded Methods

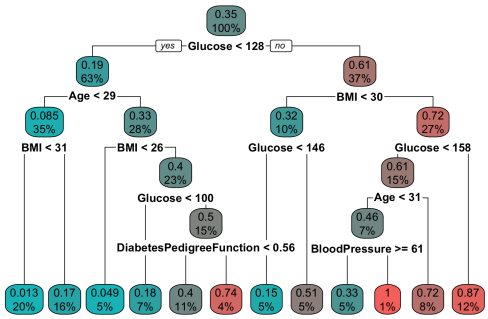
The third and final class of feature selection methods are called **embedded methods**. Embedded methods perform feature selection during the process of model training. They are usually specific to a particular type of model.

16.6.1 Decision Trees

One example of an embedded method is a decision tree (see Chapters 7, 13, and 14), which implicitly performs feature selection by placing the most informative predictors at the top of the tree and ignoring those that are unassociated with the outcome.

Question 16.11

Here is the decision tree produced by CART, using information gain/mutual information as the splitting criterion as usual:



Which features were selected for this tree and which were ignored? How were the features transformed from their original forms in the dataset?

16.6.2 Regularized Models

Another example of an embedded method is **regularization**. The easiest way to understand regularization is through our discussion of maximum likelihood estimation for GLMs in Chapter 12. The goal of maximum likelihood estimation is to find the set of model coefficients, βs, that maximize the joint probability (likelihood) of our observed data given the model. The trouble with this is that more complex models, with more parameters, will generally fit the data better: i.e. produce a higher likelihood.

Regularization addresses this by introducing a penalty term on the likelihood that is proportional to the size of the parameters. In L_1 regularization, a.k.a. **Lasso**, the penalty term is proportional to the absolute values of the coefficients. It looks like this:

$$\lambda \sum_{j=1}^{p} |\beta_j|$$

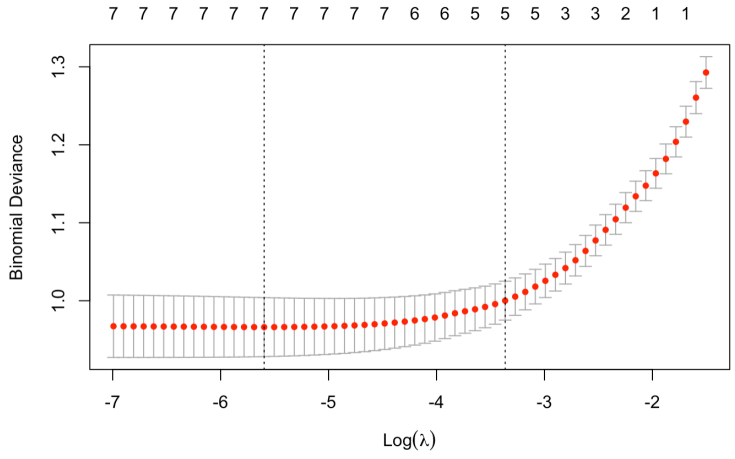
where p is the number of predictors. This creates a tradeoff in the model between the likelihood and the number of parameters. During optimization, the model will set the coefficients on predictors to zero if including those predictors does not sufficiently improve the likelihood. The relative importance of the penalty term and likelihood is adjusted using the parameter λ. We will see regularized regression methods in much greater detail in Chapter 17.

Question 16.12

Here is the raw model output from the multivariate logistic regression model that includes all eight predictors:

```
Call:
glm(formula = Outcome ~ . - LogDiabetesPedigreeFunction, family = "binomial",
    data = d)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5566  -0.7274  -0.4159   0.7267   2.9297

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.4046964  0.7166359 -11.728  < 2e-16 ***
Pregnancies               0.1231823  0.0320776   3.840 0.000123 ***
Glucose                   0.0351637  0.0037087   9.481  < 2e-16 ***
BloodPressure            -0.0132955  0.0052336  -2.540 0.011072 *
SkinThickness             0.0006190  0.0068994   0.090 0.928515
Insulin                  -0.0011917  0.0009012  -1.322 0.186065
BMI                       0.0897010  0.0150876   5.945 2.76e-09 ***
DiabetesPedigreeFunction  0.9451797  0.2991475   3.160 0.001580 **
Age                       0.0148690  0.0093348   1.593 0.111192
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 993.48  on 767  degrees of freedom
Residual deviance: 723.45  on 759  degrees of freedom
AIC: 741.45

Number of Fisher Scoring iterations: 5
```

Now let's consider what happens when we use a L_1 regularized logistic regression model, produced using the R package *glmnet*. Here is what happens to the model's error (assessed using 10-fold cross validation; measured using a metric called **binomial deviance**) when we vary λ:




```
Measure: Binomial Deviance

      Lambda Measure      SE Nonzero
```

```
min 0.004468  0.9686 0.02647       7
1se 0.028723  0.9922 0.02118       5
```

We choose λ to be equal to the value that produces the minimum deviance. Here are the coefficients of the final model:

```
9 x 1 sparse Matrix of class "dgCMatrix"
                                      1
(Intercept)              -8.048785391
Pregnancies               0.115632123
Glucose                   0.033559189
BloodPressure            -0.010901115
SkinThickness             .
Insulin                  -0.000837989
BMI                       0.083305233
DiabetesPedigreeFunction  0.847558021
Age                       0.013503422
```

Compare this output to the results of models obtained through forward and backward selection methods, as well as to the full (unregularized) logistic regression model. What are the advantages and disadvantages of the regularization approach vs. wrappers and filters?