Fundamentals of Predictive Analytics with JMP®

Third Edition

Ron Klimberg

The correct bibliographic citation for this manual is as follows: Klimberg, Ron. 2023. *Fundamentals of Predictive Analytics with JMP®, Third Edition*. Cary, NC: SAS Institute Inc.

Fundamentals of Predictive Analytics with JMP®, Third Edition

ISBN 978-1-68580-003-1 (Hardcover)
ISBN 978-1-68580-027-7 (Paperback)
ISBN 978-1-68580-000-0 (Web PDF)
ISBN 978-1-68580-001-7 (EPUB)
ISBN 978-1-68580-002-4 (Kindle)

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

April 2023

Contents

About This Book xiii

About The Author xvii

Acknowledgments xix

Dedication xxi

Chapter 1: Introduction 1
- Historical Perspective 1
- Two Questions Organizations Need to Ask 1
  - Return on Investment 2
  - Cultural Change 2
- Business Intelligence and Business Analytics 3
- Introductory Statistics Courses 4
  - The Problem of Dirty Data 6
  - Added Complexities in Multivariate Analysis 6
- Practical Statistical Study 6
  - Obtaining and Cleaning the Data 7
  - Understanding the Statistical Study as a Story 8
  - The Plan-Perform-Analyze-Reflect Cycle 8
  - Using Powerful Software 9
- Framework and Chapter Sequence 10

Chapter 2: Statistics Review 13
- Introduction 13
- Fundamental Concepts 1 and 2 13
  - FC1: Always Take a Random and Representative Sample 14
  - FC2: Remember That Statistics Is Not an Exact Science 15
- Fundamental Concept 3: Understand a *Z*-Score 16
- Fundamental Concept 4 17
  - FC4: Understand the Central Limit Theorem 17
  - Learn from an Example 18
- Fundamental Concept 5 22
  - Understand One-Sample Hypothesis Testing 22
  - Consider *p*-Values 23
- Fundamental Concept 6 23
  - Understand That Few Approaches and Techniques Are Correct—Many Are Wrong 24

Ways JMP Can Access Data in Excel 24
Three Possible Outcomes When You Choose a Technique 33
Exercises 34

Chapter 3: Dirty Data 35
Introduction 35
Data Set 36
Error Detection 37
Outlier Detection 39
Approach 1 41
Approach 2 43
Missing Values 45
Statistical Assumptions of Patterns of Missing 46
Conventional Correction Methods 48
The JMP Approach 51
Example Using JMP 52
General First Steps on Receipt of a Data Set 58
Exercises 58

Chapter 4: Data Discovery with Multivariate Data 61
Introduction 61
Use Tables to Explore Multivariate Data 63
PivotTables 63
Tabulate in JMP 65
Use Graphs to Explore Multivariate Data 66
Graph Builder 66
Scatterplot 71
Explore a Larger Data Set 73
Trellis Chart 73
Bubble Plot 76
Explore a Real-World Data Set 79
Use Correlation Matrix and Scatterplot Matrix to Examine Relationships of Continuous Variables 79
Use Graph Builder to Examine Results of Analyses 79
Generate a Trellis Chart and Examine Results 80
Use Dynamic Linking to Explore Comparisons in a Small Data Subset 85
Return to Graph Builder to Sort and Visualize a Larger Data Set 85
Exercises 87

Chapter 5: Regression and ANOVA 89
Introduction 89
Regression 89
Perform a Simple Regression and Examine Results 90
Understand and Perform Multiple Regression 92
Understand and Perform Regression with Categorical Data 104

Analysis of Variance 109
Perform a One-Way ANOVA 111
Evaluate the Model 111
Perform a Two-Way ANOVA 122
Exercises 130

Chapter 6: Logistic Regression 133
Introduction 133
Dependence Technique 133
The Linear Probability Model 134
The Logistic Function 135
A Straightforward Example Using JMP 137
Create a Dummy Variable 137
Use a Contingency Table to Determine the Odds Ratio 137
Calculate the Odds Ratio 140
Examine the Parameter Estimates 142
Compute Probabilities for Each Observation 147
Check the Model's Assumptions 148
A Realistic Logistic Regression Statistical Study 150
Understand the Model-Building Approach 151
Run Bivariate Analyses 154
Run the Initial Regression and Examine the Results 155
Convert a Continuous Variable to Discrete Variables 156
Producing Interaction Variables 158
Validate and Confusion Matrix 160
Exercises 161

Chapter 7: Principal Components Analysis 163
Introduction 163
Basic Steps in JMP 164
Produce the Correlations and Scatterplot Matrix 164
Create the Principal Components 164
Run a Regression of *y* on Prin1 and Excluding Prin2 167
Understand Eigenvalue Analysis 168
Conduct the Eigenvalue Analysis and the Bartlett Test 168
Verify Lack of Correlation 169
Dimension Reduction 169
Produce the Correlations and Scatterplot Matrix 169
Conduct the Principal Component Analysis 170
Determine the Number of Principal Components to Select 170
Compare Methods for Determining the Number of Components 172
Discovery of Structure in the Data 173
A Straightforward Example 173
An Example with Less Well-Defined Data 175
Exercises 177

Chapter 8: Least Absolute Shrinkage and Selection Operator and Elastic Net 179
Introduction 179
The Importance of the Bias-Variance Tradeoff 180
Ridge Regression 181
Least Absolute Shrinkage and Selection Operator 184
Perform the Technique 185
Examine the Results 185
Elastic Net 187
Perform the Technique 187
Compare with LASSO 187
Exercises 189

Chapter 9: Cluster Analysis 191
Introduction 191
Example Applications 191
An Example from the Credit Card Industry 192
The Need to Understand Statistics and the Business Problem 192
Hierarchical Clustering 193
Understand the Dendrogram 193
Understand the Methods for Calculating Distance between Clusters 193
Perform Hierarchical Clustering with Complete Linkage 194
Examine the Results 195
Consider a Scree Plot to Discern the Best Number of Clusters 196
Apply the Principles to a Small but Rich Data Set 198
Consider Adding Clusters in a Regression Analysis 201
k-Means Clustering 202
Understand the Benefits and Drawbacks of the Method 202
Choose *k* and Determine the Clusters 203
Perform *k*-Means Clustering 206
Change the Number of Clusters 206
Create a Profile of the Clusters with Parallel Coordinate Plots (Optional) 208
Perform Iterative Clustering 211
Score New Observations 213
k-Means Clustering versus Hierarchical Clustering 213
Exercises 214

Chapter 10: Decision Trees 217
Introduction 217
Benefits and Drawbacks 217
Definitions and an Example 218
Theoretical Questions 219
Classification Trees 220
Begin Tree and Observe Results 220
Use JMP to Choose the Split That Maximizes the LogWorth Statistic 222

Split the Root Node According to Rank of Variables 222
Split Second Node According to the College Variable 224
Examine Results and Predict the Variable for a Third Split 227
Examine Results and Predict the Variable for a Fourth Split 227
Examine Results and Continue Splitting to Gain Actionable Insights 228
Prune to Simplify Overgrown Trees 229
Examine Receiver Operator Characteristic and Lift Curves 229
Regression Trees 231
Understand How Regression Trees Work 231
Restart a Regression Driven by Practical Questions 235
Use Column Contributions and Leaf Reports for Large Data Sets 236
Exercises 237

Chapter 11: *k*-Nearest Neighbors 241
Introduction 241
Example—Age and Income as Correlates of Purchase 241
The Way That JMP Resolves Ties 243
The Need to Standardize Units of Measurement 243
k-Nearest Neighbors Analysis 244
Perform the Analysis 244
Make Predictions for New Data 245
k-Nearest Neighbor for Multiclass Problems 247
Understand the Variables 247
Perform the Analysis and Examine Results 248
The *k*-Nearest Neighbor Regression Models 250
Perform a Linear Regression as a Basis for Comparison 250
Apply the *k*-Nearest Neighbors Technique 250
Compare the Two Methods 250
Make Predictions for New Data 254
Limitations and Drawbacks of the Technique 254
Exercises 255

Chapter 12: Neural Networks 257
Introduction 257
Drawbacks and Benefits 257
A Simplified Representation 258
A More Realistic Representation 260
Understand Validation Methods 262
Holdback Validation 262
k-fold Cross Validation 263
Understand the Hidden Layer Structure 264
A Few Guidelines for Determining Number of Nodes 264
Practical Strategies for Determining Number of Nodes 265
The Method of Boosting 265

Understand Options for Improving the Fit of a Model 266
Complete the Data Preparation 267
Use JMP on an Example Data Set 269
Perform a Linear Regression as a Baseline 269
Perform the Neural Network Ten Times to Assess Default Performance 271
Boost the Default Model 272
Compare Transformation of Variables and Methods of Validation 273
Change the Architecture 276
Predict a Binary Dependent Variable 277
Exercises 279

Chapter 13: Bootstrap Forests and Boosted Trees 281
Introduction 281
Bootstrap Forests 282
Understand Bagged Trees 282
Perform a Bootstrap Forest 283
Perform a Bootstrap Forest for Regression Trees 288
Boosted Trees 289
Understand Boosting 289
Perform Boosting 289
Perform a Boosted Tree for Regression Trees 292
Use Validation and Training Samples 293
Exercises 298

Chapter 14: Model Comparison 299
Introduction 299
Perform a Model Comparison with Continuous Dependent Variable 300
Understand Absolute Measures 300
Understand Relative Measures 300
Understand Correlation between Variable and Prediction 301
Explore the Uses of the Different Measures 301
Perform a Model Comparison with Binary Dependent Variable 304
Understand the Confusion Matrix and Its Limitations 304
Understand True Positive Rate and False Positive Rate 305
Interpret Receiving Operator Characteristic Curves 306
Compare Two Example Models Predicting Churn 309
Perform a Model Comparison Using the Lift Chart 311
Train, Validate, and Test 313
Perform Stepwise Regression 313
Examine the Results of Stepwise Regression 316
Compute the MSE, MAE, and Correlation 316
Examine the Results for MSE, MAE, and Correlation 316
Understand Overfitting from a Coin-Flip Example 317
Use the Model Comparison Platform 318

Exercises 330

Chapter 15: Text Mining 333
Introduction 333
Historical Perspective 333
Unstructured Data 334
Developing the Document Term Matrix 335
Understand the Tokenizing Stage 335
Understand the Phrasing Stage 343
Understand the Terming Stage 344
Observe the Order of Operations 346
Developing the Document Term Matrix with a Larger Data Set 346
Generate a Word Cloud and Examine the Text 347
Examine and Group Terms 349
Add Frequent Phrases to List of Terms 350
Parse the List of Terms 350
Using Multivariate Techniques 350
Perform Latent Semantic Analysis 352
Perform Topic Analysis 357
Perform Cluster Analysis 358
Using Predictive Techniques 363
Perform Primary Analysis 364
Perform Logistic Regressions 365
Exercises 368

Chapter 16: Market Basket Analysis 371
Introduction 371
Association Analyses 371
Examples 372
Understand Support, Confidence, and Lift 372
Association Rules 373
Support 373
Confidence 373
Lift 374
Use JMP to Calculate Confidence and Lift 375
Use the A Priori Algorithm for More Complex Data Sets 375
Form Rules and Calculate Confidence and Lift 376
Analyze a Real Data Set 376
Perform Association Analysis with Default Settings 376
Reduce the Number of Rules and Sort Them 377
Examine Results 377

Target Results to Take Business Actions 378
Exercises 379

Chapter 17: Time Series Forecasting 381
Introduction 381
Discovery 382
Time Series Plot 383
Trend Analysis 385
Testing for Significant Linear Trend Component 386
Seasonal Component 388
Testing for Significant Seasonal Component 389
Cyclical Component 391
Autocorrelation 392
Lagging and Differencing 397
Lagging 397
Differencing 398
Decomposition 398
Stationarity 400
Randomness 401
Simple Moving Average and Simple Exponential Smoothing Models 409
Simple Moving Average 410
Simple Exponential Smoothing 414
Forecast Performance Measures 418
Autoregressive and Moving Average Models 421
ARIMA Models 423
ARIMA Modeling with Log Variable 425
ARIMA Modeling with Seasonality 426
Advanced Exponential Smoothing Models 430
State Space Smoothing Models 434
Holdback 437
Time Series Cross-Validation 438
Time Series Forecast 440
Exercises 445

Chapter 18: Statistical Storytelling 447
The Path from Multivariate Data to the Modeling Process 447
Early Applications of Data Mining 447
Numerous JMP Customer Stories of Modern Applications 448
Definitions of Data Mining 448
Data Mining 449
Predictive Analytics 449

A Framework for Predictive Analytics Techniques 450
The Goal, Tasks, and Phases of Predictive Analytics 451
The Difference between Statistics and Data Mining 453
SEMMA 454

References 457

Index 461

About This Book

What Does This Book Cover?

This book focuses on the business statistics intelligence component of business analytics. It covers processes to perform a statistical study that might include data mining or predictive analytics techniques. Some real-world business examples of using these techniques are as follows:

- target marketing
- customer relation management
- market basket analysis
- cross-selling
- forecasting
- market segmentation
- customer retention
- improved underwriting
- quality control
- competitive analysis
- fraud detection and management
- churn analysis

Specific applications can be found at https://www.jmp.com/en_my/customer-stories/customer-listing/featured.html. The bottom line, as reported by the KDNuggets poll (2008), is this: The median return on investment for data mining projects is in the 125–150% range. (See http://www.kdnuggets.com/polls/2008/roi-data-mining.htm.)

This book is *not* an introductory statistics book, although it does introduce basic data analysis, data visualization, and analysis of multivariate data. For the most part, your introductory statistics course has not completely prepared you to move on to real-world statistical analysis. The primary objective of this book is, therefore, to provide a bridge from your introductory statistics course to practical statistical analysis. This book is also not a highly technical book that dives deeply into the theory or algorithms, but it will provide insight into the “black box” of the methods covered. Analytics techniques covered by this book include the following:

- regression
- ANOVA
- logistic regression
- principal component analysis

- LASSO and Elastic Net
- cluster analysis
- decision trees
- *k*-nearest neighbors
- neural networks
- bootstrap forests and boosted trees
- text mining
- time series forecasting
- association rules

Is This Book for You?

This book is designed for the student who wants to prepare for his or her professional career and who recognizes the need to understand both the concepts and the mechanics of predominant analytic modeling tools for solving real-world business problems. This book is designed also for the practitioner who wants to obtain a hands-on understanding of business analytics to make better decisions from data and models, and to apply these concepts and tools to business analytics projects.

This book is for you if you want to explore the use of analytics for making better business decisions and have been either intimidated by books that focus on the technical details, or discouraged by books that focus on the high-level importance of using data without including the how-to of the methods and analysis.

Although not required, your completion of a basic course in statistics will prove helpful. Experience with the book's software, JMP Pro 17, is not required.

What's New in This Edition?

This third edition includes one new chapter on time series forecasting. All the old chapters from the second edition are updated to JMP 17. In addition, about 60% more end-of-chapter exercises are provided.

What Should You Know about the Examples?

This book includes tutorials for you to follow to gain hands-on experience with JMP.

Software Used to Develop the Book's Content

JMP Pro 17 is the software used throughout this book.

Example Code and Data

You can access the example code and data for this book by linking to its author page at http://support.sas.com/klimberg. Some resources, such as instructor resources and add-ins used in the book, can be found on the JMP User Community file exchange at https://community.jmp.com.

Where Are the Exercise Solutions?

We strongly believe that for you to obtain maximum benefit from this book you need to complete the examples in each chapter. At the end of each chapter are suggested exercises so that you can practice what has been discussed in the chapter. Professors and instructors can obtain the exercise solutions by requesting them through the author's SAS Press webpage at http://support.sas.com/klimberg.

We Want to Hear from You

SAS Press books are written *by* SAS Users *for* SAS Users. We welcome your participation in their development and your feedback on SAS Press books that you are using. Please visit sas.com/books.

About The Author

Ron Klimberg, PhD, is a professor at the Haub School of Business at Saint Joseph's University in Philadelphia, PA. Before joining the faculty in 1997, he was a professor at Boston University, an operations research analyst at the U.S. Food and Drug Administration, and an independent consultant. His current primary interests include multiple criteria decision making, data envelopment analysis, data visualization, data mining, and modeling in general. Klimberg was the 2007 recipient of the Tengelmann Award for excellence in scholarship, teaching, and research. He received his PhD from Johns Hopkins University and his MS from George Washington University.

Learn more about the author by visiting his author page, where you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more: http://support.sas.com/klimberg.

Chapter 13: Bootstrap Forests and Boosted Trees

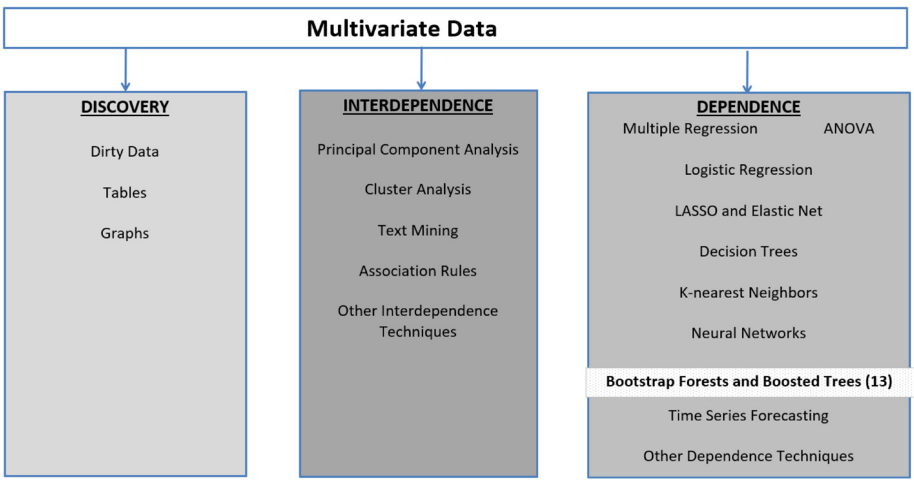
Introduction

Decision trees, discussed in Chapter 10, are easy to comprehend, easy to explain, can handle qualitative variables without the need for dummy variables, and (as long as the tree isn't too large) are easily interpreted. Despite all these advantages, trees suffer from one grievous problem: they are unstable.

In this context, unstable means that a small change in the input can cause a large change in the output. For example, if one variable is changed even a little, and if the variable is important, then it can cause a split high up in the tree to change and, in so doing, cause changes all the way down the tree. Trees can be very sensitive not just to changes in variables, but also to the inclusion or exclusion of variables.

Fortunately, there is a remedy for this unfortunate state of affairs. As shown in Figure 13.1, this chapter discusses two techniques, bootstrap forests and boosted trees, which overcome this instability and many times result in better models.

Figure 13.1: A Framework for Multivariate Analysis



Bootstrap Forests

The first step in constructing a remedy involves a statistical method known as “the bootstrap.” The idea behind the bootstrap is to take a single sample and turn it into several “bootstrap samples,” each of which has the same number of observations as the original sample. In particular, a bootstrap sample is produced by random sampling with replacement from the original sample. These several bootstrap samples are then used to build trees. The results for each observation for each tree are averaged to obtain a prediction or classification for each observation. This averaging process implies that the result will not be unstable. Thus, the bootstrap remedies the great deficiency of trees.

This chapter does not dwell on the intricacies of the bootstrap method. (If interested, see “The Bootstrap,” an article written by Shalizi (2010) in *American Scientist*. Suffice it to say that bootstrap methods are very powerful and, in general, do no worse than traditional methods that analyze only the original sample, and very often (as in the present case) can do much better.

It seems obvious now that you should take your original sample, turn it into several bootstrap samples, and construct a tree for each bootstrap sample. You could then combine the results of these several trees. In the case of classification, you could grow each tree so that it classified each observation—knowing that each tree would not classify each observation the same way.

Bootstrap forests, also called *random forests* in the literature, are a very powerful method, probably the most powerful method, presented in this book. On any particular problem, some other method might perform better. In general, however, bootstrap forests will perform better than other methods. Beware, though, of this great power. On some data sets, bootstrap forests can fit the data perfectly or almost perfectly. However, such a model will not predict perfectly or almost perfectly on new data. This is the phenomenon of “overfitting” the data, which is discussed in detail in Chapter 14. For now, the important point is that there is no reason to try to fit the data as well as possible. Just try to fit it well enough. You might use other algorithms as benchmarks, and then see whether bootstrap forests can do better.

Understand Bagged Trees

Suppose you grew 101 bootstrap trees. Then you would have 101 classifications (“votes”) for the first observation. If 63 of the votes were “yes” and 44 were “no,” then you would classify the first observation as a “yes.” Similarly, you could obtain classifications for all the other observations. This method is called “bagged trees,” where “bag” is shorthand for “bootstrap aggregation”—bootstrap the many trees and then aggregate the individual answers from all the trees. A similar approach can obtain predictions for each observation in the case of regression trees. This method uses the same data to build the tree and to compute the classification error.

An alternative method of obtaining predictions from bootstrapped trees is the use of "in-bag" and "out-of-bag" observations. Some observations, say two-thirds, are used to build the tree (these are the "in-bag" observations) and then the remaining one-third out-of-bag observations are dropped down the tree to see how they are classified. The predictions are compared to the truth for the out-of-bag observations, and the error rate is calculated on the out-of-bag observations. The reasons for using out-of-bag observations will be discussed more fully in Chapter 14. Suffice it to say that using the same observations to build the tree and then also to compute the error rate results in an overly optimistic error rate that can be misleading.

There is a problem with bagged trees, and it is that they are all quite similar, so their structures are highly correlated. We could get better answers if the trees were not so correlated, if each of the trees was more of an independent solution to the classification problem at hand. The way to achieve this was discovered by Breiman (2001). Breiman's insight was to not use all the independent variables for making each split. Instead, for each split, a subset of the independent variables is used.

To see the advantage of this insight, consider a node that needs to be split. Suppose variable *X*1 would split this node into two child nodes. Each of the two child nodes contains about the same number of observations, and each of the observations is only moderately homogeneous. Perhaps variable *X*2 would split this into two child nodes. One of these child nodes is small but relatively pure; the other child node is much larger and moderately homogenous. If *X*1 and *X*2 have to compete against each other in this spot, and if *X*1 wins, then you would never uncover the small, homogeneous node. On the other hand, if *X*1 is excluded and *X*2 is included so that *X*2 does not have to compete against *X*1, then the small, homogeneous pocket will be uncovered. A large number of trees is created in this manner, producing a forest of bootstrap trees. Then, after each tree has classified all the observations, voting is conducted to obtain a classification for each observation. A similar approach is used for regression trees.

Perform a Bootstrap Forest

To demonstrate bootstrap forests, use the Titanic data set, TitanicPassengers.jmp, the variables of which are described below in Table 13.1. It has 1,309 observations.

You want to predict who will survive:

1. Open the TitanicPassengers.jmp data set.
2. In the course of due diligence, you will engage in exploratory data analysis before beginning any modeling. This exploratory data analysis will reveal that **Body** correlates perfectly with not surviving (**Survived**), as selecting **Analyze** ▶ **Tabulate** (or **Fit Y by X)**, for these two variables will show. Also, **Lifeboat** correlates very highly with surviving (**Survived**), because very few of the people who got into a lifeboat failed to survive. So, use only the variables marked with an asterisk in Table 13.1.
3. Select **Analyze** ▶ **Predictive Modeling** ▶ **Partition**.

Table 13.1: Variables in the TitanicPassengers.jmp Data Set

Variable	Description
Passenger Class *	1 = first, 2 = second, 3 = third
Survived *	No, Yes
Name	Passenger name
Sex *	Male, female
Age *	Age in years
Siblings and Spouses *	Number of Siblings and Spouses aboard
Parents and Children *	Number of Parents and Children aboard
Ticket #	Ticket number
Fare *	Fare in British pounds
Cabin	Cabin number (known only for a few passengers)
Port *	Q = Queenstown, C = Cherbourg, S = Southampton
Lifeboat	16 lifeboats 1–16 and four inflatables A–D
Body	Body identification number for deceased
Home/Destination	Home or destination of traveler

4. Select **Survived** as **Y, response**. The other variables with asterisks in Table 13.1 are **X, Factor**.
5. For **Method**, choose **Bootstrap Forest**. **Validation Portion** is zero by default. **Validation** will be discussed in Chapter 14. For now, leave this at zero.
6. Click **OK**.

or

3. Select **Analyze ▶ Predictive Modeling ▶ Bootstrap Forest.**
4. Select **Survived** as **Y, response**. The other variables with asterisks in Table 13.1 are **X, Factor**.
5. **Validation Portion** is zero by default. Leave this at zero.
6. Click **OK**.

Understand the Options in the Dialog Box

Some of the options presented in the Bootstrap Forest dialog box, shown in Figure 13.2, are as follows:

- **Number of trees in the forest** is self-explanatory. There is no theoretical guidance on what this number should be. But empirical evidence suggests that there is no benefit to having a very large forest. 100 is the default. Try also 300 and 500. Setting the number of trees to be in the thousands probably will not be helpful.
- **Number of terms sampled per split** is the number of variables to use at each split. The default value is 6. If the original number of predictors is p, use $\sqrt{p}$ rounded down for

classification, and for regression use $p/3$ rounded down (Hastie et al. 2009, p. 592). These are only rough recommendations. After trying $\sqrt{p}$, try $2\sqrt{p}$ and $\sqrt{p}/2$, as well as other values, if necessary.

- **Bootstrap sample rate** is the proportion of the data set to resample with replacement. Just leave this at the default 1 so that the bootstrap samples have the same number of observations as the original data set.

    - **Minimum Splits Per Tree** and **Maximum Splits Per Tree** are self-explanatory.
    - **Minimum Size Split** is the minimum number of observations in a node that is a candidate for splitting. For classification problems, the minimum node size should be one. For regression problems, the minimum node size should be five as recommended by Hastie et al. (2009, page 592).
    - Do not check the box **Multiple Fits over number of terms**. The associated **Max Number of Terms** is only used when the box is checked. The interested reader is referred to the user guide for additional details.

For now, change the **Number of Terms Sampled per Split** to 1 and just click **OK**.

The output of the Bootstrap Forest should look like Figure 13.3.

Figure 13.2: The Bootstrap Forest Dialog Box

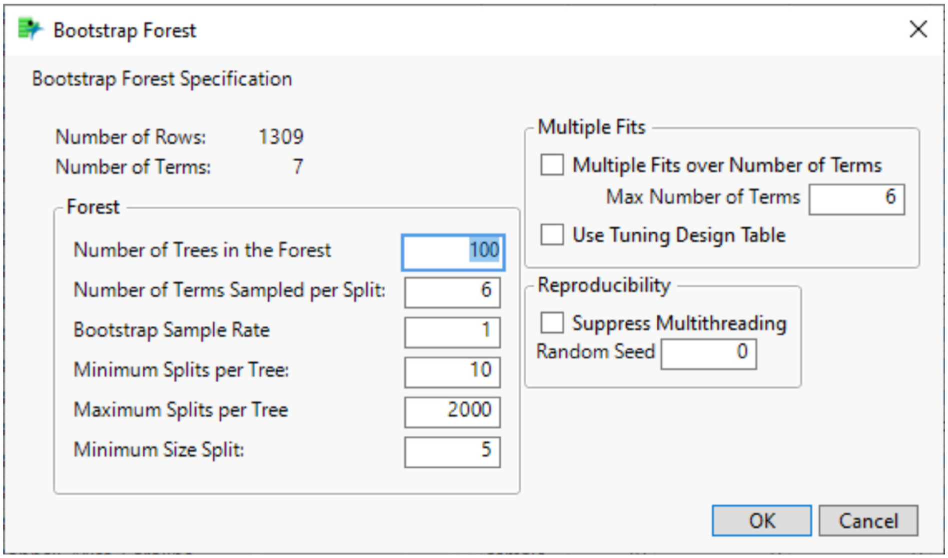
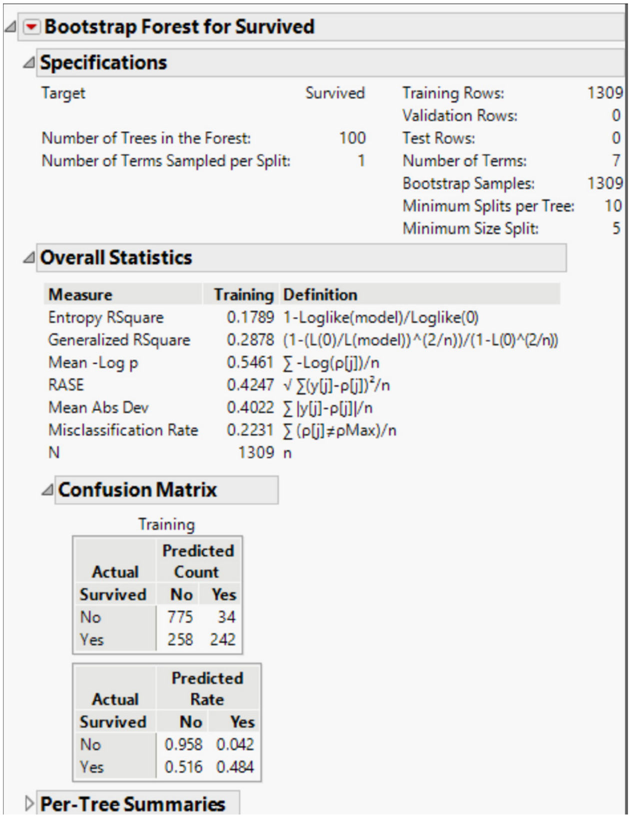


Figure 13.3: Bootstrap Forest Output for the TitanicPassengers.jmp Data Set



Select Options and Relaunch

Your results will be slightly different because this algorithm uses a random number generator to select the bootstrap samples. The sample size is 1,309. The lower left value in the first column of the Confusion Matrix, 258, and the top right value in the right-most column, 34, are the classification errors (discussed further in Chapter 14). Added together, 258 + 34 = 292, they compose the numerator of the reported misclassification rate in Figure 13.3: 292/1309 = 22.31%. Now complete the following steps:

1. Click the **Bootstrap Forest for Survived** red triangle and select **Redo ▶ Relaunch Analysis**.
2. The Partition dialog box appears. Click **OK**.
3. Now you are back in the Bootstrap Forest dialog box, as in Figure 13.2. Click **OK**. This time, double **the Number of Terms Sampled Per Split** to **2**.
4. Click **OK**.

The Bootstrap Forest output should look similar to Figure 13.4.

Figure 13.4: Bootstrap Forest Output with the Number of Terms Sampled per Split to 2

Bootstrap Forest for Survived

Specifications

Target	Survived	Training Rows:	1309
		Validation Rows:	0
Number of Trees in the Forest:	100	Test Rows:	0
Number of Terms Sampled per Split:	2	Number of Terms:	7
		Bootstrap Samples:	1309
		Minimum Splits per Tree:	10
		Minimum Size Split:	5

Overall Statistics

Measure	Training	Definition
Entropy RSquare	0.3645	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.5223	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.4226	∑ -Log(ρ[j])/n
RASE	0.3606	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.3030	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.1597	∑ (ρ[j]≠ρMax)/n
N	1309	n

Confusion Matrix

Training

Actual Survived	Predicted Count No	Predicted Count Yes
No	761	48
Yes	161	339

Actual Survived	Predicted Rate No	Predicted Rate Yes
No	0.941	0.059
Yes	0.322	0.678

Per-Tree Summaries

Examine the Improved Results

Notice the dramatic improvement. The error rate is now 15.97%. You could run the model again, this time increasing the **Number of Terms Sampled Per Split** to **3** and increasing the **Number of Trees** to **500**. These changes will again produce another dramatic improvement. Notice also that, although there are many missing values in the data set, Bootstrap Forest uses the full 1309 observations. Many other algorithms (for example, logistic regression) have to drop observations that have missing values.

An additional advantage of random forests is that, just like basic decision trees in Chapter 10 produced column contributions to show the important variables, random forests produce a similar ranking of variables. To get this list, click the **Bootstrap Forest for Survived** and select **Column Contributions**. This ranking can be especially useful in providing guidance for variable selection when later building logistic regressions or neural network models.

Perform a Bootstrap Forest for Regression Trees

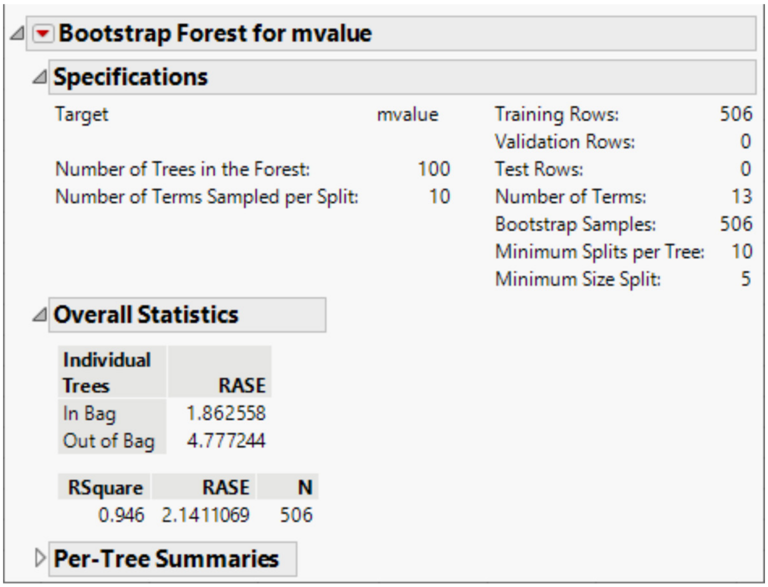
Now briefly consider random forests for regression trees. Use the data set MassHousing.jmp in which the target variable is median value:

1. Select **Analyze** ▶ **Predictive Modeling** ▶ **Partition**.
2. Select **mvalue** for **Y, Response** and all the other variables as **X, Factor**.
3. For method, select **Bootstrap Forest**.
4. Click **OK**.
5. In the Bootstrap Forest dialog box, leave everything at default and click **OK**.

The Bootstrap Forest output should look similar to Figure 13.5.

Under **Overall Statistics**, see the In-Bag and Out-of-Bag RMSE. Notice that the Out-of-Bag RMSE is much larger than the In-Bag RMSE. This is to be expected because the algorithm is fitting on the In-Bag data. It then applies the estimated model to data that were not used to fit the model to obtain the Out-of-Bag RMSE. You will learn much more about this topic in Chapter 14. What's important for your purposes is that you obtained RSquare = 0.946 and RMSE = 1.863 for the full data set (remember that your results will be different because of the random number generator). These values compare quite favorably with the results from a linear regression: RSquare = 0.7406 and RMSE = 4.745. You can see that bootstrap forest regression can offer a substantial improvement over traditional linear regression. Additionally, bootstrap forest regression addresses nonlinearity better than ordinary least squares.

Figure 13.5: Bootstrap Forest Output for the MassHousing.jmp Data Set



Bootstrap Forest for mvalue

Specifications

Target	mvalue	Training Rows:	506
		Validation Rows:	0
Number of Trees in the Forest:	100	Test Rows:	0
Number of Terms Sampled per Split:	10	Number of Terms:	13
		Bootstrap Samples:	506
		Minimum Splits per Tree:	10
		Minimum Size Split:	5

Overall Statistics

Individual Trees	RASE
In Bag	1.862558
Out of Bag	4.777244

RSquare	RASE	N
0.946	2.1411069	506

Per-Tree Summaries

Boosted Trees

Boosting is a general approach to combining a sequence of models in which each successive model changes slightly in response to the errors from the preceding model.

Understand Boosting

Boosting starts with estimating a model and obtaining residuals. The observations with the biggest residuals (where the model did the worst job) are given additional weight, and then the model is re-estimated on this transformed data set. In the case of classification, the misclassified observations are given more weight. After several models have been constructed, the estimates from these models are averaged to produce a prediction or classification for each observation. As was the case with bootstrap forests, this averaging implies that the predictions or classifications from the boosted tree model will not be unstable. When boosting, there is often no need to build elaborate models; simple models often suffice. In the case of trees, there is no need to grow the tree completely out; a tree with just a few splits often will do the trick. Indeed, simply fitting “stumps” (trees with only a single split and two leaves) at each stage often produces good results.

A boosted tree builds a large tree by fitting a sequence of smaller trees. At each stage, a smaller tree is grown on the scaled residuals from the prior stage, and the magnitude of the scaling is governed by a tuning parameter called the learning rate. The essence of boosting is that, on the current tree, it gives more weight to the observations that were misclassified on the prior tree.

Perform Boosting

Use Boosted Trees on the data set TitanicPassengers.jmp:

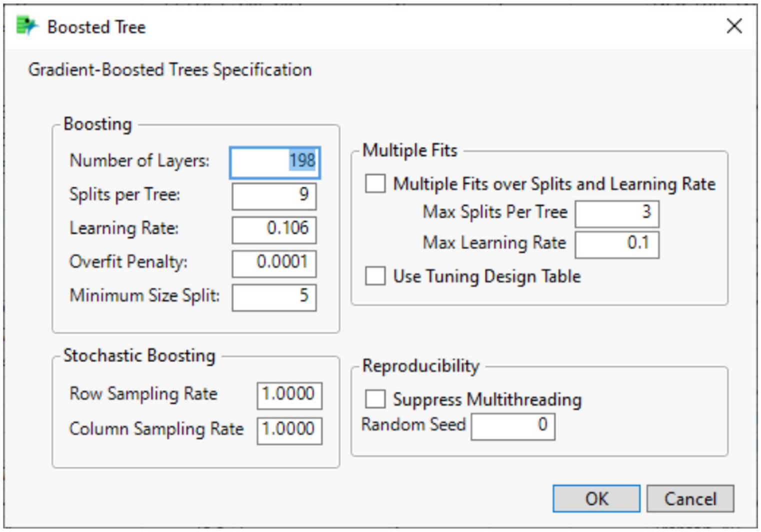
1. Select **Analyze** ▶ **Predictive Modeling** ▶ **Partition**.
2. For **Method**, select **Boosted Tree**.
3. Use the same variables as you did with Bootstrap Forests. Select **Survived** as **Y, response**. The other variables with asterisks in Table 13.1 are **X, Factor.**
4. Click **OK**.

or

1. Select **Analyze** ▶ **Predictive Modeling** ▶ **Boosted Tree.**
2. Select **Survived** as **Y, response**. The other variables with asterisks in Table 13.1 are **X, Factor**.
3. Click **OK**.

The Boosted Tree dialog box will appear, as shown in Figure 13.6.

Figure 13.6: The Boosted Tree Dialog Box



Understand the Options in the Dialog Box

The options are as follows:

- **Number of Layers** is the number of stages in the final tree. It is the number of trees to grow.
- **Splits Per Tree** is the number of splits for each stage (tree). If the number of splits is one, then "stumps" are being used.
- **Learning Rate** is a number between zero and one. A number close to one means faster learning, but at the risk of overfitting. Set this number close to one when the Number of Layers (trees) is small.
- **Overfit Penalty** helps protect against fitting probabilities equal to zero. It applies only to categorical targets.
- **Minimum Split Size** is the smallest number of observations to be in a node before it can be split.
- **Multiple Fits over splits and learning rate** will have JMP build a separate boosted tree for all combinations of splits and learning rate that the user chooses. Leave this box unchecked.

Select Options and Relaunch

For now, leave everything at default and click **OK**. The Bootstrap Tree output is shown in Figure 13.7. It shows a misclassification rate of 11.5%.

Figure 13.7: Boosted Tree Output for the TitanicPassengers.jmp Data Set

Boosted Tree for Survived

Specifications

Target	Survived	Number of training rows:	1309
Number of Layers:	198	Number of validation rows:	0
Splits per Tree:	9		
Learning Rate:	0.106		
Overfit Penalty:	0.0001		

Overall Statistics

Measure	Training	Definition
Entropy RSquare	0.5664	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.7195	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.2884	∑ -Log(ρ[j])/n
RASE	0.2970	$\sqrt{\sum(y[j]-\rho[j])^2/n}$
Mean Abs Dev	0.1994	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.1146	∑ (ρ[j]≠ρMax)/n
N	1309	n

Confusion Matrix

Training

Actual Survived	Predicted Count No	Predicted Count Yes
No	775	34
Yes	116	384

Actual Survived	Predicted Rate No	Predicted Rate Yes
No	0.958	0.042
Yes	0.232	0.768

Using the guidance given about the options, set the **Learning rate** high, to 0.9.

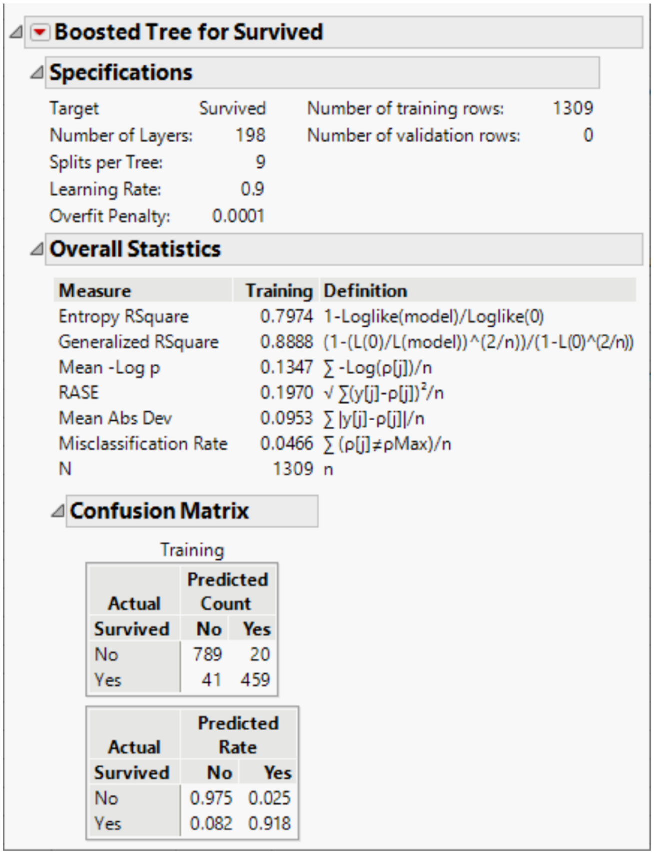
1. Click the **Boosted Tree for Survived** red triangle and select **Redo**. Choose **Relaunch Analysis**. The Partition dialog box appears. Click **OK**.
2. The Boosted Tree dialog box appears. Change the **Learning rate** to **0.9**.
3. Click **OK**.

Examine the Improved Results

The Bootstrap Tree output will look like Figure 13.8, which has an error rate of 4.7%.

This is a substantial improvement over the default model and better than the Bootstrap Forest models. You could run the model again and this time change the **Number of Layers** to 250. Because this is bigger than the default, you could have chosen 200 or 400. Change the **Learning Rate** to 0.4. Because this is somewhere between 0.9 and 0.1, you could have chosen 0.3 or 0.6. Change the number of **Splits Per Tree** to **5** (again, there is nothing magic about this number).

Figure 13.8: Boosted Tree Output with a Learning Rate of 0.9



Boosted Tree for Survived

Specifications

Target	Survived	Number of training rows:	1309
Number of Layers:	198	Number of validation rows:	0
Splits per Tree:	9		
Learning Rate:	0.9		
Overfit Penalty:	0.0001		

Overall Statistics

Measure	Training	Definition
Entropy RSquare	0.7974	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.8888	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.1347	∑ -Log(ρ[j])/n
RASE	0.1970	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.0953	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.0466	∑ (ρ[j]≠ρMax)/n
N	1309	n

Confusion Matrix

Training

Actual Survived	Predicted Count No	Predicted Count Yes
No	789	20
Yes	41	459

Actual Survived	Predicted Rate No	Predicted Rate Yes
No	0.975	0.025
Yes	0.082	0.918

Boosted Trees is a very powerful method that also works for regression trees as you will see immediately below.

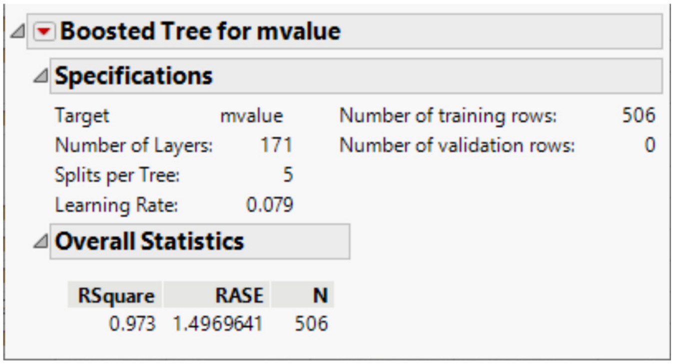
Perform a Boosted Tree for Regression Trees

Again, use the data set MassHousing.jmp.

1. Select **Analyze** ▶ **Predictive Modeling** ▶ **Partition**.
2. For **Method**, select **Boosted Tree**.
3. Select **mvalue** for the dependent variable, and all the other variables for independent variables.
4. Click **OK**.
5. Leave everything at default and click **OK**.

You should get the Boosted Tree output shown in Figure 13.9.

Figure 13.9: Boosted Tree Output for the MassHousing.jmp Data Set


Boosted Tree for mvalue

Specifications

Target	mvalue	Number of training rows:	506
Number of Layers:	171	Number of validation rows:	0
Splits per Tree:	5		
Learning Rate:	0.079		

Overall Statistics

RSquare	RASE	N
0.973	1.4969641	506

Boosting is better than the Bootstrap Forest in Figure 13.5 (look at RSquare and RMSE), to say nothing of the linear regression.

Next, relaunch the analysis and change the **Learning rate** to 0.9. This is a substantial improvement with a perfect fit with an RSquare of 1.0. This is not really surprising, because both Bootstrap Forests and Boosted Trees are so powerful and flexible that they often can fit a data set perfectly.

Use Validation and Training Samples

When using such powerful methods, you should not succumb to the temptation to the make the RSquared as high as possible because such models rarely predict well on new data. To gain some insight into this problem, you will consider one more example in this chapter in which you will use a manually selected holdout sample.

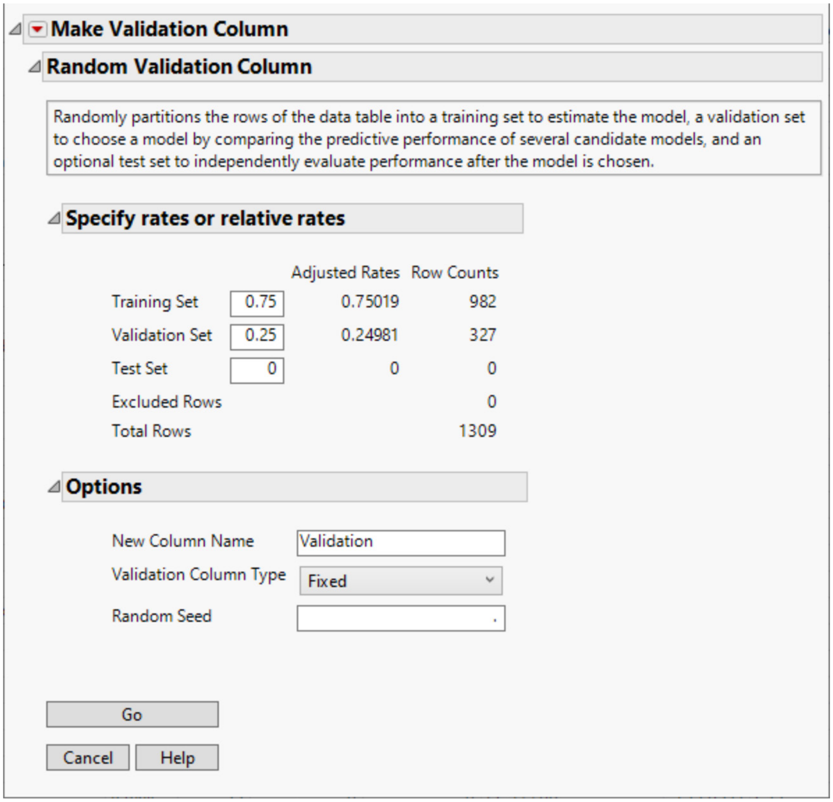
You will divide the data into two samples, a “training” sample that consists of, for example, 75% of the data, and a “validation” sample that consists of the remaining 25%. You will then rerun your three boosted tree models on the TitanicPassengers.jmp data set on the training sample. JMP will automatically use the estimated models to make predictions on the validation sample.

Create a Dummy Variable

To effect this division into training and validation samples, you will need a dummy variable that randomly splits the data into a 75% / 25% split:

1. Open TitanicPassengers.jmp.
2. Select **Analyze** ▶ **Predictive Modeling** ▶ **Make Validation Column**. Click **OK**.
3. The **Make Validation Column** report dialog box will appear, as shown in Figure 13.10.
4. Click **Go**.

Figure 13.10: The Make Validation Column Report Dialog Box



You will see that a new column called **Validation** has been added to the data table. You specified that the training set is to be 0.75 of the total rows, but this is really just a suggestion. The validation set will contain about 0.25 of the total rows.

Perform a Boosting at Default Settings

Run a Boosted Tree at default as before:

1. Select **Analyze** ▶ **Predictive Modeling** ▶ **Partition**.
2. As you did before, select **Survived** as **Y, response**. The other variables with asterisks in Table 13.1 are **X, Factor**.
3. Select the **Validation** column and then click **Validation**.
4. For Method, select **Boosted Tree**.
5. Click **OK.**
6. Click **OK** again for the options window. You are initially estimating this model with the defaults.

Examine Results and Relaunch

The results are presented in Figure 13.11. There are 982 observations in the training sample and 327 in the validation sample. Because 0.75 is just a suggestion and because the random number generator is used, your results will not agree exactly with the output in Figure 13.11. The error rate in the training sample is 15.1%, and the error rate in the validation sample is 21.4.

This strongly suggests that the model estimated on the training data predicts at least as well, if not better, on brand new data. The important point is that the model does not overfit the data (which can be detected when the performance on the training data is significantly better than the performance on new data). Now relaunch:

Figure 13.11: Boosted Tree Results for the TitanicPassengers.jmp Data Set with a Training and Validation Set

Boosted Tree for Survived

Specifications

Target	Survived	Number of training rows:	982
Validation Column:	Validation	Number of validation rows:	327
Number of Layers:	40		
Splits per Tree:	9		
Learning Rate:	0.106		
Overfit Penalty:	0.0001		

Overall Statistics

Measure	Training	Validation	Definition
Entropy RSquare	0.4764	0.2812	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.6377	0.4252	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.3474	0.4812	∑ -Log(ρ[j])/n
RASE	0.3294	0.3919	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.2365	0.2836	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.1507	0.2141	∑ (ρ[j]≠ρMax)/n
N	982	327	n

Confusion Matrix

Training

Actual Survived	Predicted Count No	Predicted Count Yes
No	572	38
Yes	110	262

Validation

Actual Survived	Predicted Count No	Predicted Count Yes
No	184	15
Yes	55	73

Training

Actual Survived	Predicted Rate No	Predicted Rate Yes
No	0.938	0.062
Yes	0.296	0.704

Validation

Actual Survived	Predicted Rate No	Predicted Rate Yes
No	0.925	0.075
Yes	0.430	0.570

Cumulative Validation

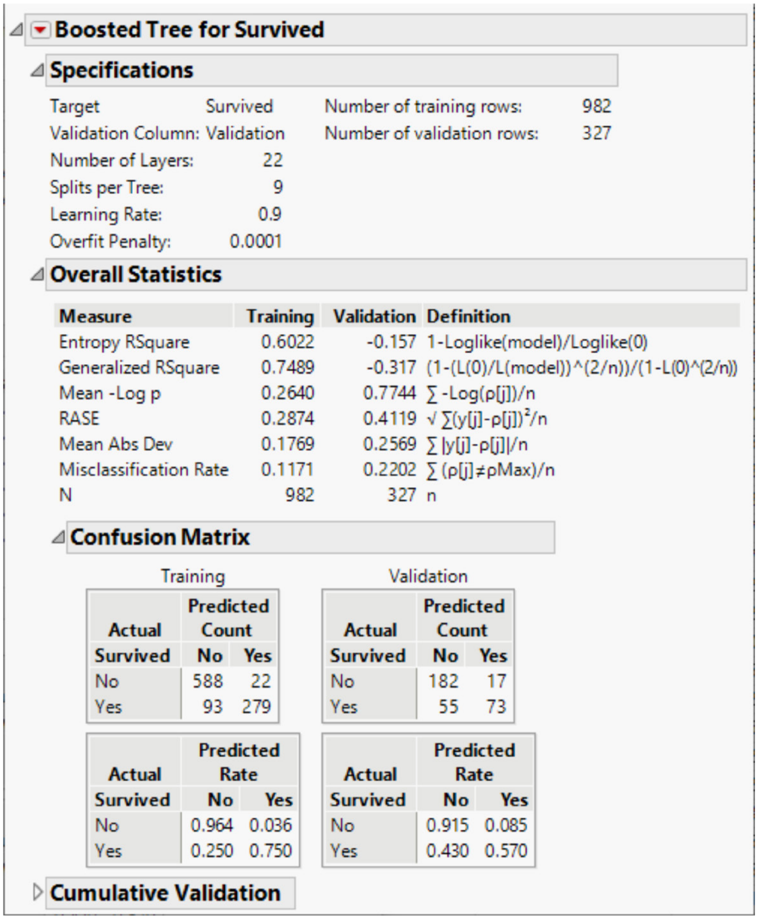
1. Click the **Boosted Tree for Survived** red triangle and select **Redo** and **Relaunch Analysis**.
2. Click **OK** to get the Boosted Trees dialog box for the options and change the learning rate to **0.9**.
3. Click **OK**.

Compare Results to Choose the Least Misleading Model

You should get results similar to Figure 13.12, where the training error rate is 11.7% and the validation error rate is 22.0%.

Now you see that the model does a better job of “predicting” on the sample data than on brand new data. This makes you think that perhaps you should prefer the default model because it does not mislead you into thinking you have more accuracy than you really do.

Figure 13.12: Boosted Tree Results with Learning Rate of 0.9



Boosted Tree for Survived

Specifications

Target	Survived	Number of training rows:	982
Validation Column:	Validation	Number of validation rows:	327
Number of Layers:	22		
Splits per Tree:	9		
Learning Rate:	0.9		
Overfit Penalty:	0.0001		

Overall Statistics

Measure	Training	Validation	Definition
Entropy RSquare	0.6022	-0.157	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.7489	-0.317	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.2640	0.7744	∑ -Log(ρ[j])/n
RASE	0.2874	0.4119	$\sqrt{\sum(y[j]-\rho[j])^2/n}$
Mean Abs Dev	0.1769	0.2569	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.1171	0.2202	∑ (ρ[j]≠ρMax)/n
N	982	327	n

Confusion Matrix

Training

Actual Survived	Predicted Count No	Predicted Count Yes
No	588	22
Yes	93	279

Validation

Actual Survived	Predicted Count No	Predicted Count Yes
No	182	17
Yes	55	73

Training

Actual Survived	Predicted Rate No	Predicted Rate Yes
No	0.964	0.036
Yes	0.250	0.750

Validation

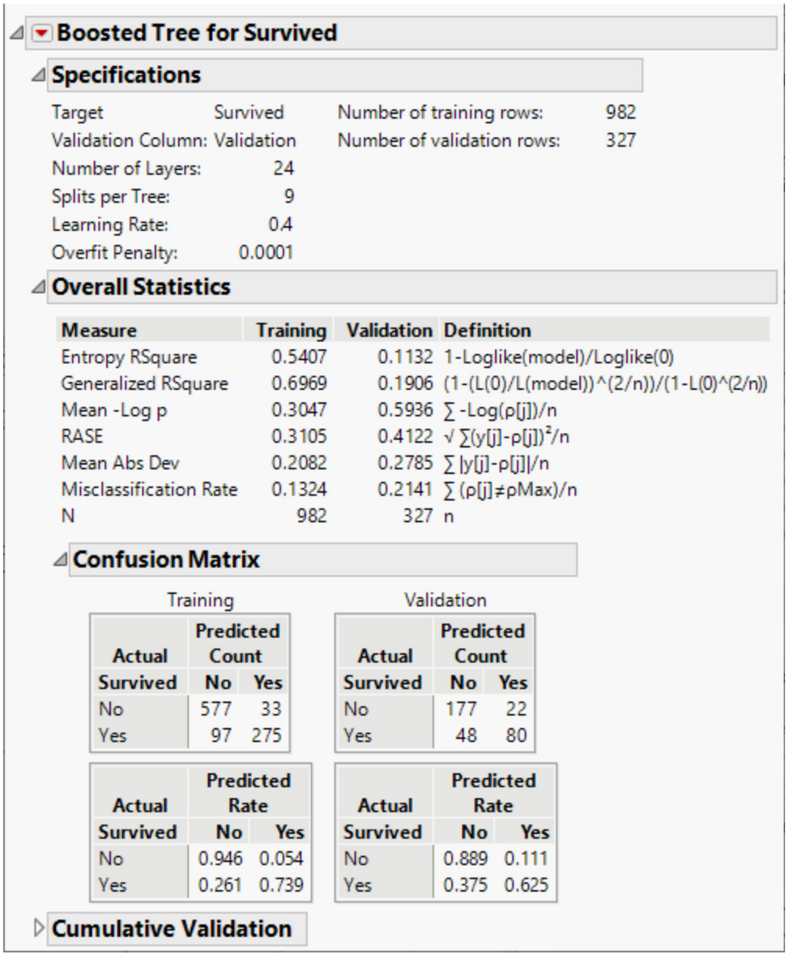
Actual Survived	Predicted Rate No	Predicted Rate Yes
No	0.915	0.085
Yes	0.430	0.570

Cumulative Validation

See if this pattern persists for the third model. Observe that the Number of Layers has decreased to 22, even though you specified it to be 198. This adjustment is automatically performed by JMP. As you did before, change the Learning Rate to 0.4. You should get similar results as shown in Figure 13.13, where the training error rate is 13.2% and the validation error rate is 21.4%. JMP again has changed the Number of Layers from the default 198 to 24.

It seems that no matter how you tweak the model to achieve better "in-sample" performance (that is, performance on the training sample), you always get about a 20% error rate on the brand-new data. So, which of the three models should you choose? The one that misleads you the least? The default model because its training sample performance is close to its validation sample performance? This idea of using "in-sample" and "out-of-sample" predictions to select the best model will be fully explored in the next chapter.

Figure 13.13: Boosted Tree Results with Learning Rate of 0.4



Boosted Tree for Survived

Specifications

Target	Survived	Number of training rows:	982
Validation Column:	Validation	Number of validation rows:	327
Number of Layers:	24		
Splits per Tree:	9		
Learning Rate:	0.4		
Overfit Penalty:	0.0001		

Overall Statistics

Measure	Training	Validation	Definition
Entropy RSquare	0.5407	0.1132	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.6969	0.1906	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.3047	0.5936	∑ -Log(ρ[j])/n
RASE	0.3105	0.4122	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.2082	0.2785	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.1324	0.2141	∑ (ρ[j]≠ρMax)/n
N	982	327	n

Confusion Matrix

Training

Actual Survived	Predicted Count No	Predicted Count Yes
No	577	33
Yes	97	275

Validation

Actual Survived	Predicted Count No	Predicted Count Yes
No	177	22
Yes	48	80

Training

Actual Survived	Predicted Rate No	Predicted Rate Yes
No	0.946	0.054
Yes	0.261	0.739

Validation

Actual Survived	Predicted Rate No	Predicted Rate Yes
No	0.889	0.111
Yes	0.375	0.625

Cumulative Validation

Predictions are created in the following way for both bootstrap forests and boosted trees. Suppose 38 trees are grown. The data for the new case is dropped down each tree (just as predictions were made for a single Decision Tree), and each tree makes a prediction. Then a "vote" is taken of all the trees, with a majority determining the winner. If, of the 38 trees, 20 predict "No" and the remaining 18 predict "Yes," then that observation is predicted to not survive.

Exercises

1. Without using a Validation column, run a logistic regression on the Titanic data and compare to the results in this chapter.

2. Can you improve on the results in Figure 13.3?

3. How high can you get the RSquare in the MassHousing example?

4. Without using a validation column, apply logistic regression, bootstrap forests, and boosted tree to the Churn data set.

5. Use a validation sample on boosted regression trees with MassHousing. How high can you get the RSquared on the validation sample? Compare this to your answer for Question 3.

6. Use a validation sample, and apply logistic regression, bootstrap forests, and boosted trees to the Churn data set. Compare this answer to your answer for Question 4.