Credit Card Marketing
Classification Trees

From *Building Better Models With JMP® Pro*, Chapter 6, SAS Press (2015). Grayson, Gardner and Stephens.

Credit Card Marketing

Classification Trees

Key ideas: Classification trees, validation, confusion matrix, misclassification, leaf report, ROC curves, lift curves.

Background

A bank would like to understand the demographics and other characteristics associated with whether a customer accepts a credit card offer. Observational data is somewhat limited for this kind of problem, in that often the company sees only those who respond to an offer. To get around this, the bank designs a focused marketing study, with 18,000 current bank customers. This focused approach allows the bank to know who does and does not respond to the offer, and to use existing demographic data that is already available on each customer.

The designed approach also allows the bank to control for other potentially important factors so that the offer combination isn't confused or confounded with the demographic factors. Because of the size of the data and the possibility that there are complex relationships between the response and the studied factors, a decision tree is used to find out if there is a smaller subset of factors that may be more important and that warrant further analysis and study.

The Task

We want to build a model that will provide insight into why some bank customers accept credit card offers. Because the response is categorical (either Yes or No) and we have a large number of potential predictor variables, we use the **Partition** platform to build a classification tree for **Offer Accepted**. We are primarily interested in understanding characteristics of customers who have accepted an offer, so the resulting model will be exploratory in nature.[1]

The Data   Credit Card Marketing BBM.jmp

The data set consists of information on the 18,000 current bank customers in the study.

Customer Number: A sequential number assigned to the customers (this column is hidden and excluded – this unique identifier will not be used directly).

Offer Accepted: Did the customer accept (Yes) or reject (No) the offer.

Reward: The type of reward program offered for the card.

Mailer Type: Letter or postcard.

Income Level: Low, Medium or High.

# Bank Accounts Open: How many non-credit-card accounts are held by the customer.

[1] In *exploratory modeling*, the goal is to understand the variables or characteristics that drive behaviors or particular outcomes. In *predictive modeling,* the goal is to accurately predict new observations and future behaviors, given the current information and situation.

Overdraft Protection: Does the customer have overdraft protection on their checking account(s) (Yes or No).

Credit Rating: Low, Medium or High.

# Credit Cards Held: The number of credit cards held at the bank.

# Homes Owned: The number of homes owned by the customer.

Household Size: Number of individuals in the family.

Own Your Home: Does the customer own their home? (Yes or No).

Average Balance: Average account balance (across all accounts over time).

Q1, **Q2**, **Q3** and **Q4 Balance**: Average balance for each quarter in the last year.

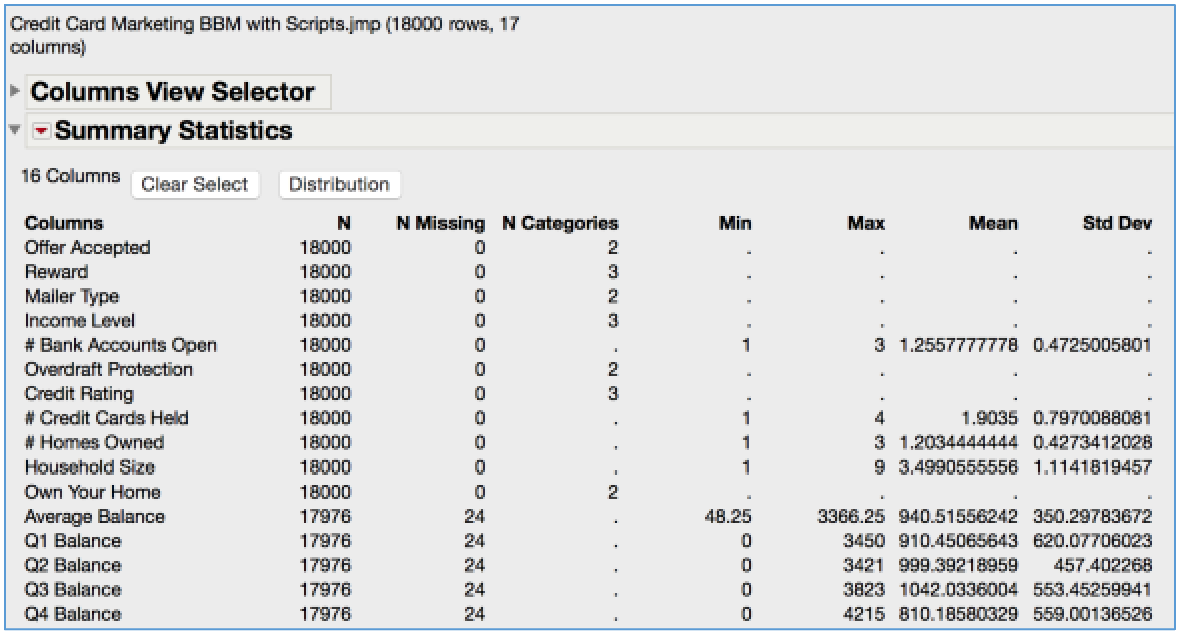
Prepare for Modeling

We start by getting to know our data. We explore the data one variable at a time, two at a time, and many variables at a time to gain an understanding of data quality and of potential relationships. Since the focus of this case study is classification trees, only some of this work is shown here. We encourage you to thoroughly understand your data and take the necessary steps to prepare your data for modeling before building exploratory or predictive models.

Exploring Data One Variable at a Time

Since we have a relatively large data set with many potential predictors, we start by creating numerical summaries of each of our variables using the Columns Viewer (see Exhibit 1). (Under the **Cols** menu, select all variables and click **Show Summary**. To deselect the variables, click **Clear Select**).

Exhibit 1 Credit, Summary Statistics for All Variables With Columns Viewer



Credit Card Marketing BBM with Scripts.jmp (18000 rows, 17 columns)

Columns View Selector

Summary Statistics

16 Columns

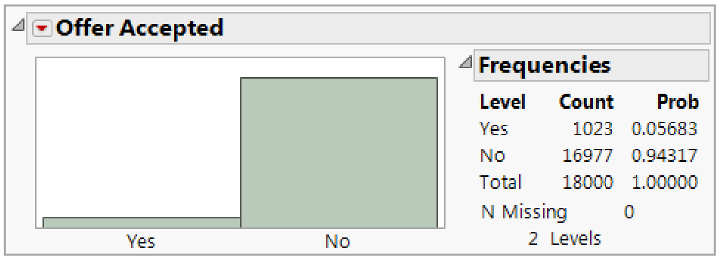
Columns	N	N Missing	N Categories	Min	Max	Mean	Std Dev
Offer Accepted	18000	0	2	.	.	.	.
Reward	18000	0	3	.	.	.	.
Mailer Type	18000	0	2	.	.	.	.
Income Level	18000	0	3	.	.	.	.
# Bank Accounts Open	18000	0	.	1	3	1.2557777778	0.4725005801
Overdraft Protection	18000	0	2	.	.	.	.
Credit Rating	18000	0	3	.	.	.	.
# Credit Cards Held	18000	0	.	1	4	1.9035	0.7970088081
# Homes Owned	18000	0	.	1	3	1.2034444444	0.4273412028
Household Size	18000	0	.	1	9	3.4990555556	1.1141819457
Own Your Home	18000	0	2	.	.	.	.
Average Balance	17976	24	.	48.25	3366.25	940.51556242	350.29783672
Q1 Balance	17976	24	.	0	3450	910.45065643	620.07706023
Q2 Balance	17976	24	.	0	3421	999.39218959	457.402268
Q3 Balance	17976	24	.	0	3823	1042.0336004	553.45259941
Q4 Balance	17976	24	.	0	4215	810.18580329	559.00136526

Under **N Categories**, we see that each of our categorical variables has either two or three levels. **N Missing** indicates that we are missing 24 observations for each of the balance columns. (Further investigation indicates that these values are missing from the same 24 customers.) The other statistics provide an idea of the centering, spread and shapes of the continuous distributions.

Next, we graph our variables one at a time. (Use **Analyze > Distribution**, select all of the variables as **Y, Columns**, and click **OK.** Click **Stack** from the top red triangle for a horizontal layout).

In Exhibit 2, we see that only around 5.68 percent of the 18,000 offers were accepted.

Exhibit 2 Credit, Distribution of Offer Accepted



Frequencies

Level	Count	Prob
Yes	1023	0.05683
No	16977	0.94317
Total	18000	1.00000

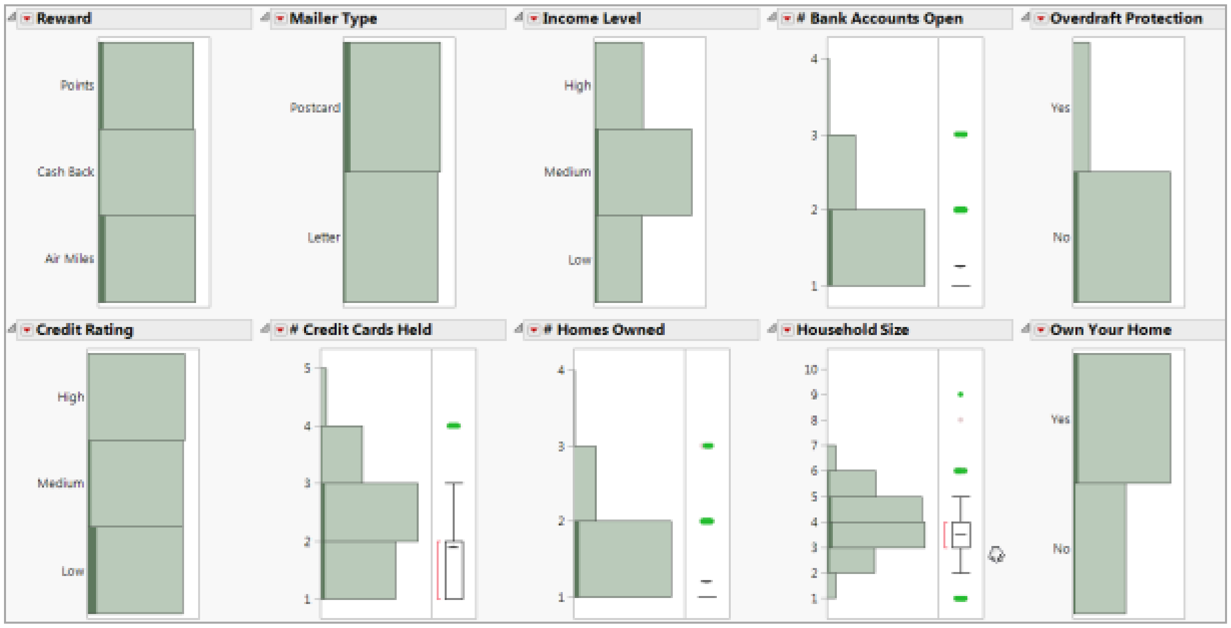
N Missing 0
2 Levels

We select the **Yes** level in **Offer Accepted** and then examine the distribution of accepted offers (the shaded area) across the other variables in our data set (the first 10 variables are shown in Exhibit 3).

Our two experimental variables are **Reward** and **Mailer Type**. Offers promoting **Points** and **Air Miles** are more frequently accepted than those promoting **Cash Back**, while **Postcards** are accepted more often than **Letters**. Offers also appear to be accepted at a higher rate by customers with low to medium income, no overdraft protection and low credit ratings.

Note that both **Credit Rating** and **Income Level** are coded as **Low**, **Medium** and **High**. Peeking at the data table, we see that the modeling types for these variables are both *nominal*, but they should be coded as *ordinal* variables. To change modeling type, right-click on the modeling type icon in the data table or in any dialog window, and then select the correct modeling type.

Exhibit 3 Credit, Distribution of First 10 Variables



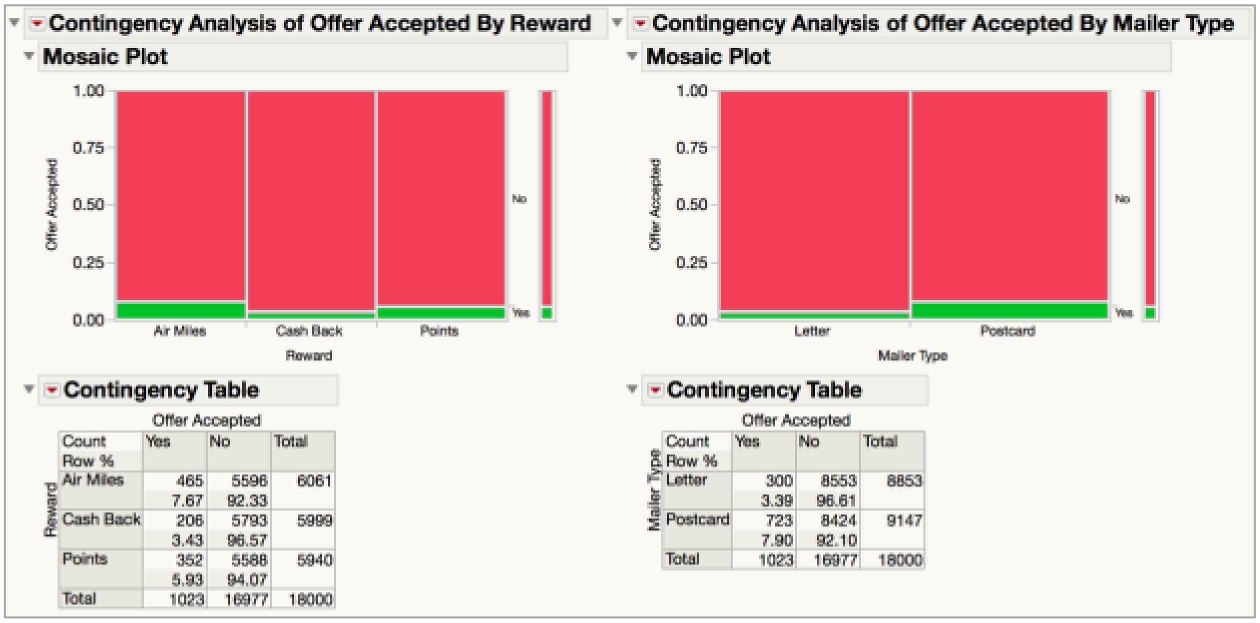
Exploring Data Two Variables at a Time

We explore relationships between our response and potential predictor variables using **Analyze > Fit Y by X** (select **Offer Accepted** as **Y, Response** and the predictors as **X, Factor**, and click **OK**.)

This two-way analysis shows potential relationships between **Offer Accepted** and several variables, including **Reward**, **Mailer Type**, **Income Level**, and **Credit Rating.**

The first two analyses are shown in Exhibit 4. Note that tests for association between the predictors and **Offer Accepted** are also provided by default (these are not shown in Exhibit 4), and additional statistical options and tests are provided under the top red triangles.

Exhibit 4 Credit Fit Y by X, Offer Accepted versus Reward and Mailer Type



Count / Row %	Yes	No	Total
Air Miles	465	5596	6061
	7.67	92.33	
Cash Back	206	5793	5999
	3.43	96.57	
Points	352	5588	5940
	5.93	94.07	
Total	1023	16977	18000

Count / Row %	Yes	No	Total
Letter	300	8553	8853
	3.39	96.61	
Postcard	723	8424	9147
	7.90	92.10	
Total	1023	16977	18000

Although we haven't thoroughly explored this data, thus far we've learned that:

- Only a small percentage – roughly 5.68 percent – of offers are accepted.
- We are missing some data for the **Balance** columns, but are not missing values for any other variable.
- Both of our experimental variables (**Reward** and **Mailer Type**) appear to be related to whether or not an offer is accepted.
- Two variables, **Income Level** and **Credit Rating**, should be coded as **Ordinal** instead of **Nominal**.

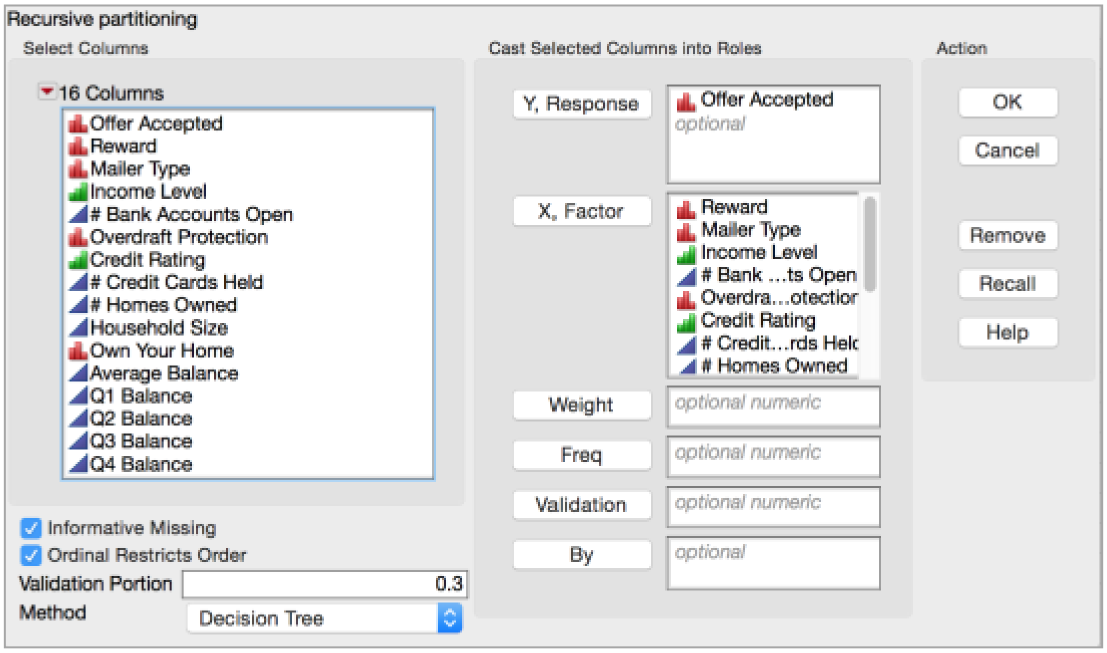
Again, we encourage you to thoroughly explore your data and to investigate and resolve potential data quality issues before building a model. Other tools, such as scatterplots and the **Graph Builder**, should also be used.

While we have only superficially explored the data in this example (as we will also do in future examples), the primary purpose of this exploration is to get to know the variables used in this case study. As such, it is intentionally brief.

The Partition Model Dialog

Having a good sense of data quality and potential relationships, we now fit a partition model to the data using **Analyze > Modeling > Partition**, with **Offer Accepted** as **Y, Response** and all of the other variables as **X, Factor** (see Exhibit 5).

Exhibit 5 Credit, Partition Dialog Window



Recursive partitioning

Select Columns

16 Columns
- Offer Accepted
- Reward
- Mailer Type
- Income Level
- # Bank Accounts Open
- Overdraft Protection
- Credit Rating
- # Credit Cards Held
- # Homes Owned
- Household Size
- Own Your Home
- Average Balance
- Q1 Balance
- Q2 Balance
- Q3 Balance
- Q4 Balance

☑ Informative Missing
☑ Ordinal Restricts Order
Validation Portion 0.3
Method Decision Tree

Cast Selected Columns into Roles

Y, Response	Offer Accepted; optional
X, Factor	Reward; Mailer Type; Income Level; # Bank ...ts Open; Overdra...otectior; Credit Rating; # Credit...rds Helc; # Homes Owned
Weight	optional numeric
Freq	optional numeric
Validation	optional numeric
By	optional

A Bit About Model Validation

When we build a predictive model, there is a risk that the model is overly complicated (*overfit*), or that the model will not perform well when applied to new data. *Model validation* (or *cross-validation*) is often used to protect against over-fitting.[2] There are two methods for model validation available from the **Partition** dialog window in JMP Pro: Specify a **Validation Portion** or select a **Validation** column (note that other methods are available from within the platform).

In this example, we'll use a random hold out portion (30 percent) to protect against overfitting (to do so, enter 0.30 in the **Validation Portion** field). This will assign 70 percent of the records to the *training set*, which is used to build the model. The remaining 30 percent will be assigned to the hold out *validation set*, which will be used to see how well the model performs on data not used to build the model.

Other Partition Model Dialog Options

Two additional options in the dialog window are **Informative Missing** and **Ordinal Restricts Order**. These are selected by default. In this example, we have two ordinal predictors, **Credit Rating** and

[2] For background information on model validation and protecting against overfitting, see en.wikipedia.org/wiki/Overfitting. For more information on validation in JMP and JMP Pro, see *Building Better Models with JMP Pro*, Chapter 6 and Chapter 8, or search for "validation" in the JMP Help.

Income Level. We also have missing values for the five balance columns. The **Informative Missing** option tells JMP to include rows with missing values in the model, and the **Ordinal Restricts Order** option tells JMP to respect the ordered categories for ordinal variables. For more information on these options, see **JMP Help**.

The completed **Partition** dialog window is displayed in Exhibit 5.

Building the Classification Tree

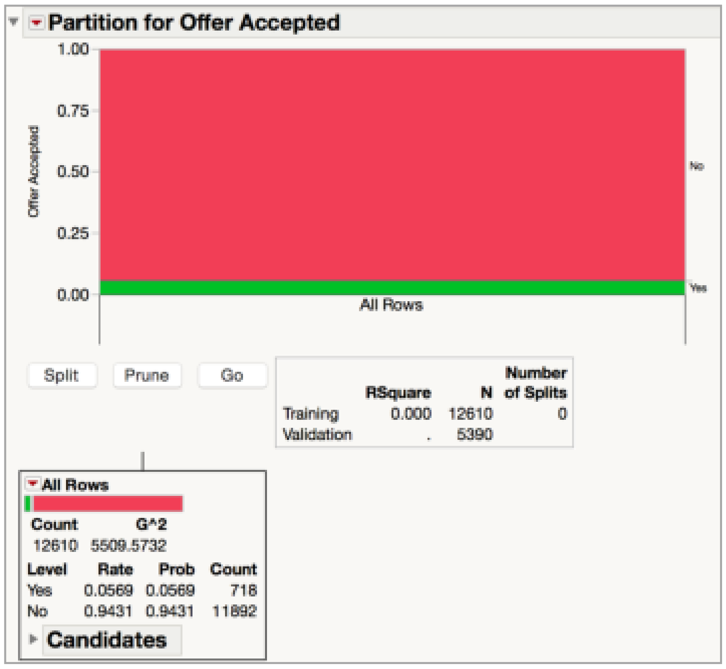
Initial results show the overall breakdown of **Offer Accepted** (Exhibit 6).[3] Recall that roughly 5.7 percent of offers were accepted.

Below the graph, we see that 12,610 observations are assigned to the training set. These observations will be used to build the model. The remaining 5,390 observations in the validation set will be used to check model performance and to stop tree growth.

Note that we have changed some of the default settings:

- Since we have a relatively large data set, points were removed from the graph (click on the **top red triangle** and select **Display Options > Show Points**).
- The response rates and counts are displayed in the tree nodes (select **Display Options > Show Split Count** from the top red triangle).

Exhibit 6 Credit, Partition Initial Window



	RSquare	N	Number of Splits
Training	0.000	12610	0
Validation	.	5390	

All Rows

Count	G^2
12610	5509.5732

Level	Rate	Prob	Count
Yes	0.0569	0.0569	718
No	0.9431	0.9431	11892

[3] Note: Since a random holdout is used, your results may be different. To obtain the same results as shown here, use the **Random Seed Reset** add-in to set the random seed to 123 before launching the **Partition** platform. The add-in can be downloaded and installed from the JMP User Community: community.jmp.com/docs/DOC-6601.

How Classification Trees Are Formed

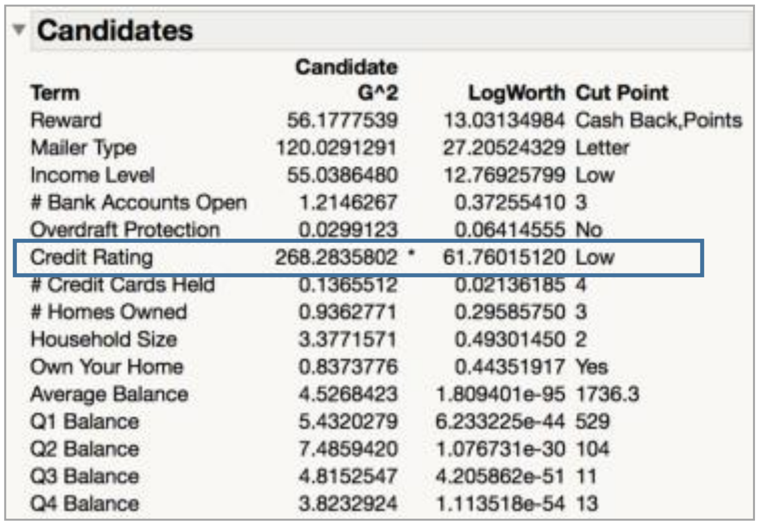
When building a classification tree, JMP iteratively splits the data based on the values of predictors to form subsets. These subsets form the “branches” of the tree. Splits are made at predictor values that cause the greatest difference in proportions (for the outcome variable) in the resulting subsets.

A measure of the dissimilarity in proportions between the two subsets is the *likelihood ratio chi-square statistic* and its associated *p*-value. The lower the *p*-value, the greater the difference between the groups. When JMP calculates this chi-square statistic in the **Partition** platform, it is labeled *G^2,* and the *p*-value that is calculated is adjusted to account for the number of splits that are being considered. The adjusted *p*-value is transformed to a log scale using the formula **-log10(adjusted p-value)**. This value is called the *LogWorth*. The bigger the LogWorth value, the better the split (Sall, 2002).

To find the split with the largest difference between subgroups (and the corresponding largest value of LogWorth), we need to consider all possible splits. For each variable, the best split location, or *cut point*, is determined, and the split with the highest LogWorth is chosen as the optimal split location.

JMP reports the G^2 and LogWorth values, along with the best cut points for each variable, under **Candidates** (use the gray disclosure icon next to **Candidates** to display). A peek at the candidates in our example indicates that the first split will be on **Credit Rating**, with **Low** in one branch and **High** and **Medium** in the other (Exhibit 7).

Exhibit 7 Credit, Partition Initial Candidate Splits



Candidates

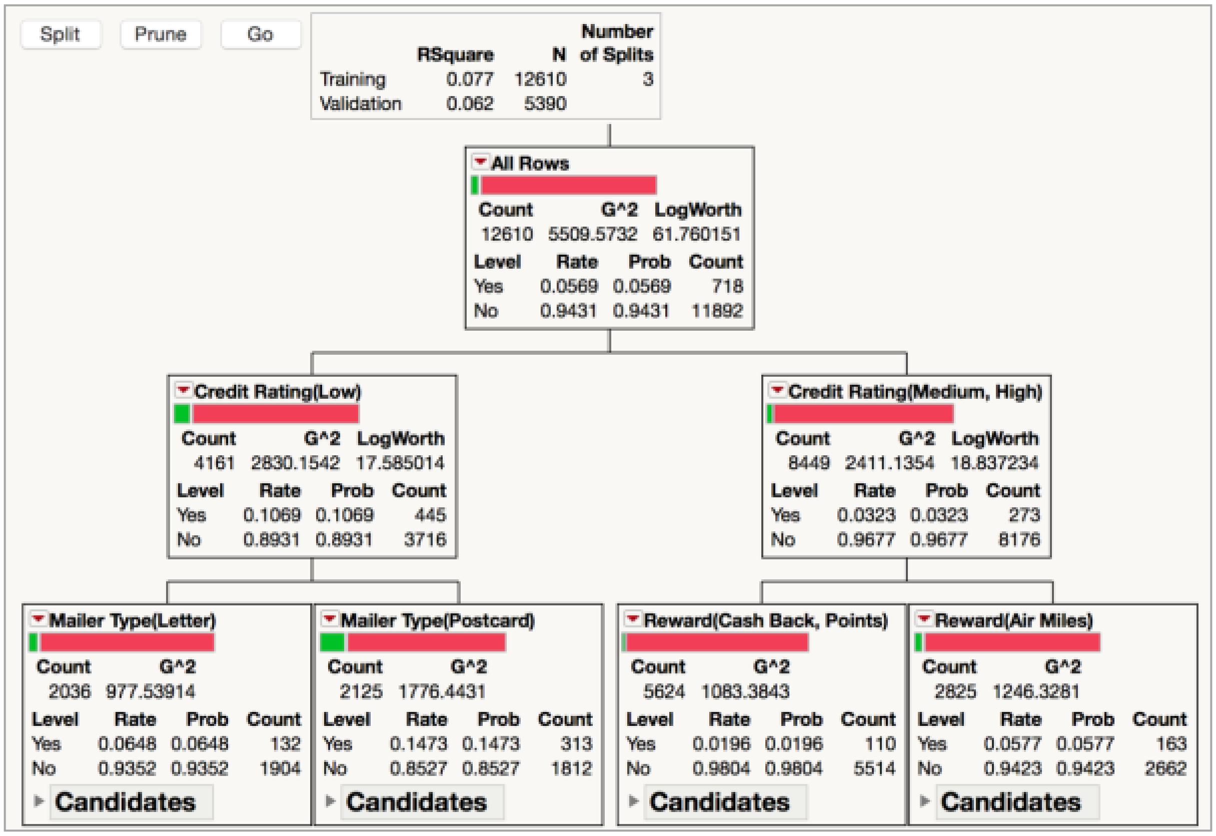
Term	Candidate G^2	LogWorth	Cut Point
Reward	56.1777539	13.03134984	Cash Back,Points
Mailer Type	120.0291291	27.20524329	Letter
Income Level	55.0386480	12.76925799	Low
# Bank Accounts Open	1.2146267	0.37255410	3
Overdraft Protection	0.0299123	0.06414555	No
Credit Rating	268.2835802 *	61.76015120	Low
# Credit Cards Held	0.1365512	0.02136185	4
# Homes Owned	0.9362771	0.29585750	3
Household Size	3.3771571	0.49301450	2
Own Your Home	0.8373776	0.44351917	Yes
Average Balance	4.5268423	1.809401e-95	1736.3
Q1 Balance	5.4320279	6.233225e-44	529
Q2 Balance	7.4859420	1.076731e-30	104
Q3 Balance	4.8152547	4.205862e-51	11
Q4 Balance	3.8232924	1.113518e-54	13

The tree after three splits (click **Split** three times) is shown in Exhibit 8.

Not surprisingly, the model is split on **Credit Rating**, **Reward** and **Mailer Type**. The lowest probability of accepting the offer (0.0196) is **Credit Rating(Medium, High**) and **Reward(Cash Back, Points)**. The highest probability (0.1473) is **Credit Rating(Low)** and **Mailer Type(Postcard)**.

After each split, the model *RSquare* (or, *Entropy RSquare*) updates (this is shown at the top of Exhibit 8). *RSquare* is a measure of how much variability in the response is being explained by the model. Without a validation set, we can continue to split until the minimum split size is achieved in each branch. (The minimum split size is an option under the top red triangle, which is set to 5 by default.) However, additional splits are not necessarily beneficial and lead to more complex and potentially overfit models.

Exhibit 8 Credit, Partition after Three Splits



	RSquare	N	Number of Splits
Training	0.077	12610	3
Validation	0.062	5390	

All Rows

Count	G^2	LogWorth
12610	5509.5732	61.760151

Level	Rate	Prob	Count
Yes	0.0569	0.0569	718
No	0.9431	0.9431	11892

Credit Rating(Low)

Count	G^2	LogWorth
4161	2830.1542	17.585014

Level	Rate	Prob	Count
Yes	0.1069	0.1069	445
No	0.8931	0.8931	3716

Credit Rating(Medium, High)

Count	G^2	LogWorth
8449	2411.1354	18.837234

Level	Rate	Prob	Count
Yes	0.0323	0.0323	273
No	0.9677	0.9677	8176

Mailer Type(Letter)

Count	G^2
2036	977.53914

Level	Rate	Prob	Count
Yes	0.0648	0.0648	132
No	0.9352	0.9352	1904

Mailer Type(Postcard)

Count	G^2
2125	1776.4431

Level	Rate	Prob	Count
Yes	0.1473	0.1473	313
No	0.8527	0.8527	1812

Reward(Cash Back, Points)

Count	G^2
5624	1083.3843

Level	Rate	Prob	Count
Yes	0.0196	0.0196	110
No	0.9804	0.9804	5514

Reward(Air Miles)

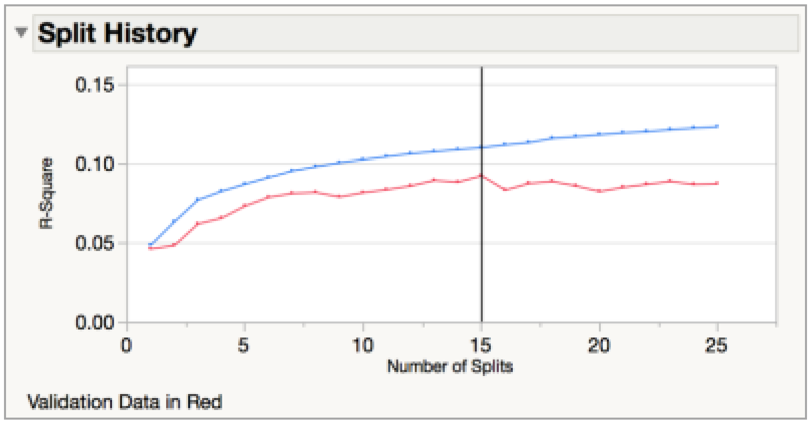
Count	G^2
2825	1246.3281

Level	Rate	Prob	Count
Yes	0.0577	0.0577	163
No	0.9423	0.9423	2662

Since we have a validation set, we click **Go** to automate the tree-building process. When this option is used, the final model will be based on the model with the *maximum value of the Validation RSquare* statistic.

The **Split History Report** (Exhibit 9) shows how the *RSquare* value changes for training and validation data after each split (note that the *y*-axis has been rescaled for illustration). The vertical line is drawn at 15, the number of splits used in the final model.

Exhibit 9 Credit, Split History After Fifteen Splits



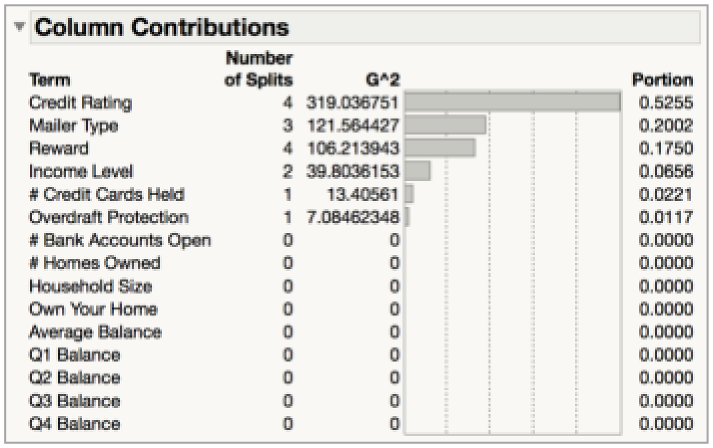
This illustrates both the concept of overfitting and the importance of using validation. With each split, the *RSquare* for the training data continues to increase. However, after 15 splits, the validation *RSquare* (the lower line in Exhibit 9) starts to decrease. For the validation set, which was not used to build the model, additional splits are not improving our ability to predict the response.

Understanding the Model

To summarize which variables are involved in these 15 splits, we turn on **Column Contributions** (from the top red triangle). This table indicates which variables are most important in terms of the overall contribution to the model (see Exhibit 10).

Credit Rating, **Mailer Type**, **Reward** and **Income Level** contribute most to the model. Several variables, including the five balance variables, are not involved in any of the splits.

Exhibit 10 Credit, Split History after Fifteen Splits



Column Contributions

Term	Number of Splits	G^2	Portion
Credit Rating	4	319.036751	0.5255
Mailer Type	3	121.564427	0.2002
Reward	4	106.213943	0.1750
Income Level	2	39.8036153	0.0656
# Credit Cards Held	1	13.40561	0.0221
Overdraft Protection	1	7.08462348	0.0117
# Bank Accounts Open	0	0	0.0000
# Homes Owned	0	0	0.0000
Household Size	0	0	0.0000
Own Your Home	0	0	0.0000
Average Balance	0	0	0.0000
Q1 Balance	0	0	0.0000
Q2 Balance	0	0	0.0000
Q3 Balance	0	0	0.0000
Q4 Balance	0	0	0.0000

Model Classifications and the Confusion Matrix

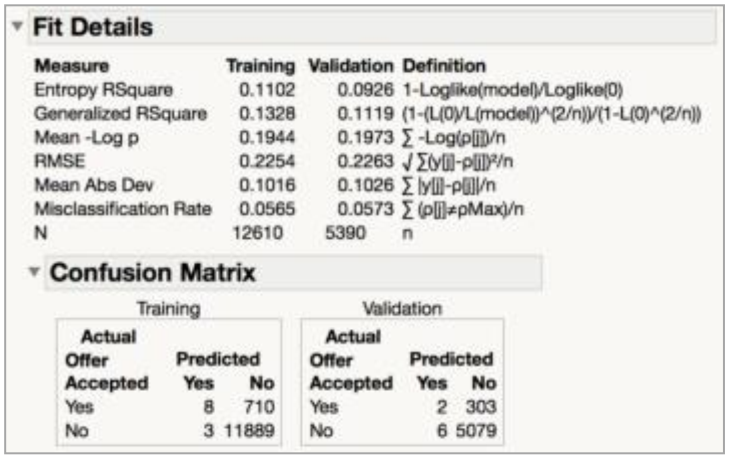
One overall measure of model accuracy is the *Misclassification Rate* (select **Fit Details** from the top red triangle). The misclassification rate for our validation data is 0.0573, or 5.73 percent. The numbers behind the misclassification rate can be seen in the *confusion matrix* (bottom, in Exhibit 11). Here, we focus on the misclassification rate and confusion matrix for the validation data. Since these data were not used in building the model, this approach provides a better indication of how well the model classifies our response, **Offer Accepted**.

There are four possible outcomes in our classification:

- An accepted offer is correctly classified as an accepted offer.
- An accepted offer is misclassified as not accepted.
- An offer that was not accepted is correctly classified as not accepted.
- An offer that was not accepted is misclassified as accepted.

One observation is that there were few cases wherein the model predicted that the offer would be accepted (see value “2” in the **Yes** column of the validation confusion matrix in Exhibit 11.) When the target variable is unbalanced (i.e., there are far more observations in one level than in the other), the model that is fit will usually result in probabilities that are small for the underrepresented category.

Exhibit 11 Credit, Fit Details With Confusion Matrix



Fit Details

Measure	Training	Validation	Definition
Entropy RSquare	0.1102	0.0926	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.1328	0.1119	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.1944	0.1973	∑ -Log(p[j])/n
RMSE	0.2254	0.2263	√ ∑(y[j]-p[j])²/n
Mean Abs Dev	0.1016	0.1026	∑ |y[j]-p[j]|/n
Misclassification Rate	0.0565	0.0573	∑ (p[j]≠pMax)/n
N	12610	5390	n

Confusion Matrix

Training

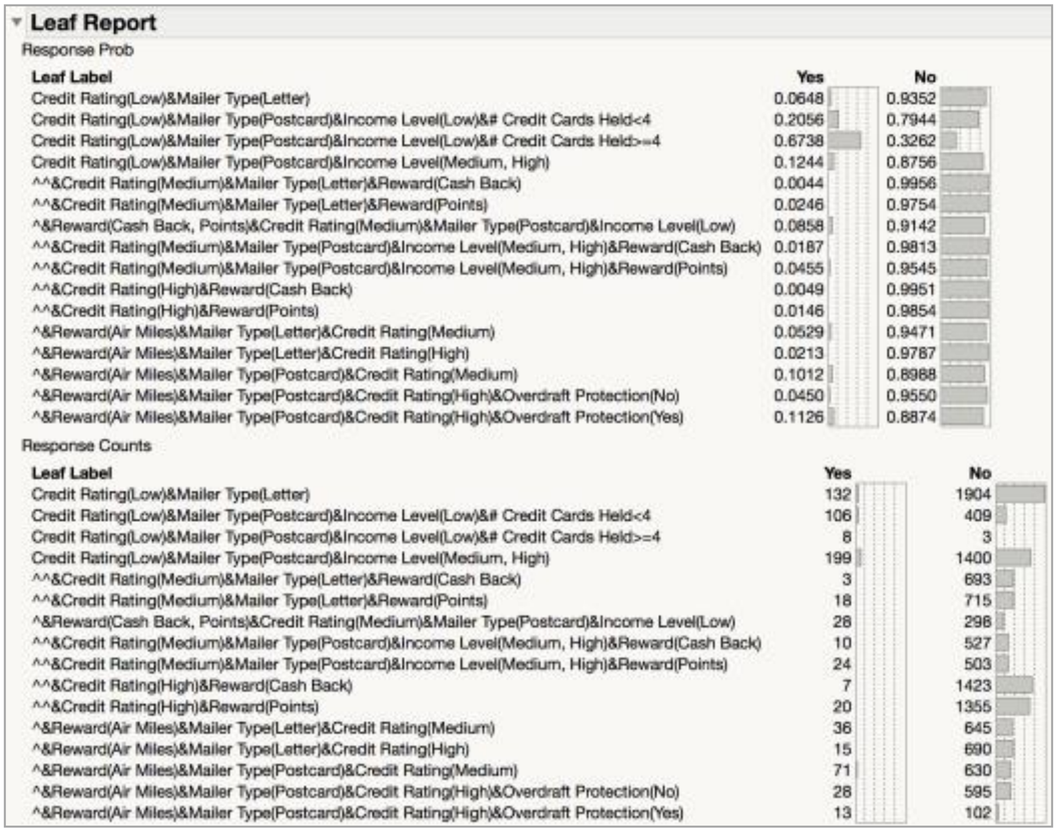
Actual Offer Accepted	Predicted Yes	Predicted No
Yes	8	710
No	3	11889

Validation

Actual Offer Accepted	Predicted Yes	Predicted No
Yes	2	303
No	6	5079

In this case, the overall rate of **Yes** (i.e., offer accepted) is 5.68 percent, which is close to the misclassification rate for this model. However, when we examine the **Leaf Report** for the fitted model (Exhibit 12), we see that there are branches in the tree that have much richer concentrations of **Offer Accepted = Yes** than the overall average rate. (Note that results in the **Leaf Report** are for the training data.)

Exhibit 12 Credit, Leaf Report for Fitted Model



Leaf Report

Response Prob

Leaf Label	Yes	No
Credit Rating(Low)&Mailer Type(Letter)	0.0648	0.9352
Credit Rating(Low)&Mailer Type(Postcard)&Income Level(Low)&# Credit Cards Held<4	0.2056	0.7944
Credit Rating(Low)&Mailer Type(Postcard)&Income Level(Low)&# Credit Cards Held>=4	0.6738	0.3262
Credit Rating(Low)&Mailer Type(Postcard)&Income Level(Medium, High)	0.1244	0.8756
^^&Credit Rating(Medium)&Mailer Type(Letter)&Reward(Cash Back)	0.0044	0.9956
^^&Credit Rating(Medium)&Mailer Type(Letter)&Reward(Points)	0.0246	0.9754
^&Reward(Cash Back, Points)&Credit Rating(Medium)&Mailer Type(Postcard)&Income Level(Low)	0.0858	0.9142
^^&Credit Rating(Medium)&Mailer Type(Postcard)&Income Level(Medium, High)&Reward(Cash Back)	0.0187	0.9813
^^&Credit Rating(Medium)&Mailer Type(Postcard)&Income Level(Medium, High)&Reward(Points)	0.0455	0.9545
^^&Credit Rating(High)&Reward(Cash Back)	0.0049	0.9951
^^&Credit Rating(High)&Reward(Points)	0.0146	0.9854
^&Reward(Air Miles)&Mailer Type(Letter)&Credit Rating(Medium)	0.0529	0.9471
^&Reward(Air Miles)&Mailer Type(Letter)&Credit Rating(High)	0.0213	0.9787
^&Reward(Air Miles)&Mailer Type(Postcard)&Credit Rating(Medium)	0.1012	0.8988
^&Reward(Air Miles)&Mailer Type(Postcard)&Credit Rating(High)&Overdraft Protection(No)	0.0450	0.9550
^&Reward(Air Miles)&Mailer Type(Postcard)&Credit Rating(High)&Overdraft Protection(Yes)	0.1126	0.8874

Response Counts

Leaf Label	Yes	No
Credit Rating(Low)&Mailer Type(Letter)	132	1904
Credit Rating(Low)&Mailer Type(Postcard)&Income Level(Low)&# Credit Cards Held<4	106	409
Credit Rating(Low)&Mailer Type(Postcard)&Income Level(Low)&# Credit Cards Held>=4	8	3
Credit Rating(Low)&Mailer Type(Postcard)&Income Level(Medium, High)	199	1400
^^&Credit Rating(Medium)&Mailer Type(Letter)&Reward(Cash Back)	3	693
^^&Credit Rating(Medium)&Mailer Type(Letter)&Reward(Points)	18	715
^&Reward(Cash Back, Points)&Credit Rating(Medium)&Mailer Type(Postcard)&Income Level(Low)	28	298
^^&Credit Rating(Medium)&Mailer Type(Postcard)&Income Level(Medium, High)&Reward(Cash Back)	10	527
^^&Credit Rating(Medium)&Mailer Type(Postcard)&Income Level(Medium, High)&Reward(Points)	24	503
^^&Credit Rating(High)&Reward(Cash Back)	7	1423
^^&Credit Rating(High)&Reward(Points)	20	1355
^&Reward(Air Miles)&Mailer Type(Letter)&Credit Rating(Medium)	36	645
^&Reward(Air Miles)&Mailer Type(Letter)&Credit Rating(High)	15	690
^&Reward(Air Miles)&Mailer Type(Postcard)&Credit Rating(Medium)	71	630
^&Reward(Air Miles)&Mailer Type(Postcard)&Credit Rating(High)&Overdraft Protection(No)	28	595
^&Reward(Air Miles)&Mailer Type(Postcard)&Credit Rating(High)&Overdraft Protection(Yes)	13	102

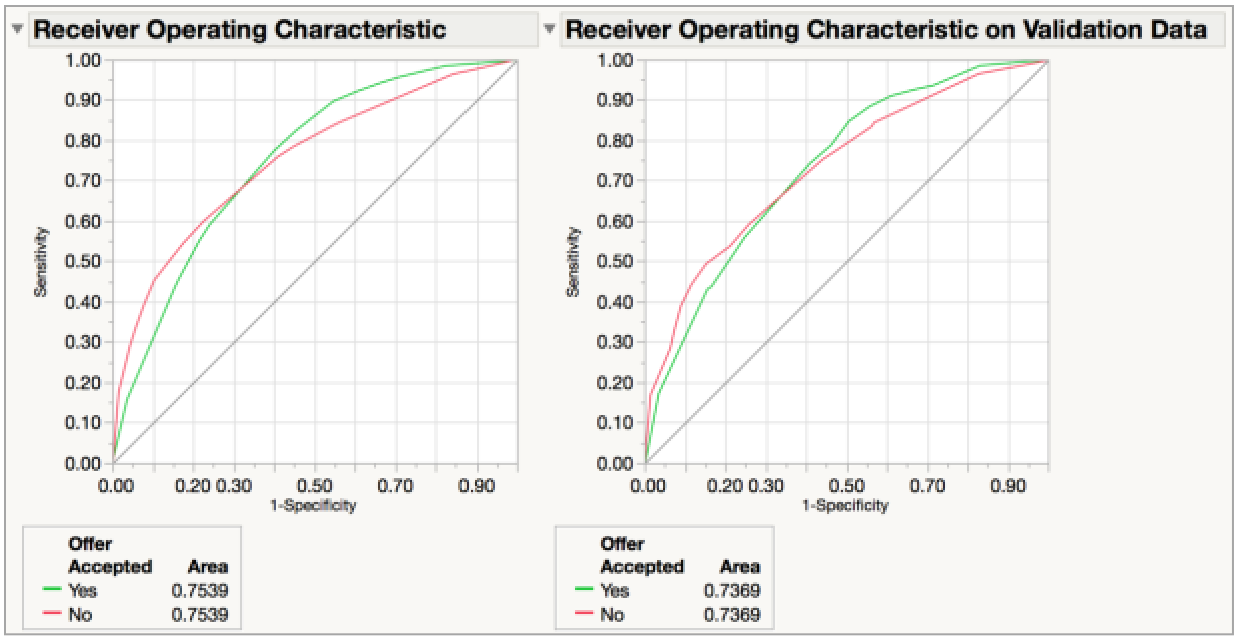
The model has probabilities of **Offer Accepted** = **Yes** in the range [0.0044, 0.6738]. When JMP classifies rows with the model, it uses a default of Prob > 0.5 to make the decision. In this case, only one of the predicted probabilities of **Yes** is > 0.5, and this one branch (or node) has only 11 observations: 8 yes and 3 no under **Response Counts** in the bottom table in Exhibit 12. The next highest predicted probability of **Offer Accepted = Yes** is 0.2056. As a result, all other rows are classified as **Offer Accepted = No**.

The ROC Curve

Two additional measures of accuracy used when building classification models are *Sensitivity* and *1-Specificity*. Sensitivity is the true positive rate. In our example, this is the ability of our model to correctly classify **Offer Accepted** as **Yes**. The second measure, 1-Specificity, is the false positive rate. In this case, a false positive occurs when an offer was not accepted, but was classified as **Yes** (accepted).

Instead of using the default decision rule of *Prob > 0.5*, we examine the decision rule *Prob* > *T*, where we let the decision threshold T range from 0 to 1. We plot Sensitivity (on the *y*-axis) versus 1-Specificity (on the *x*-axis) for each possible threshold value. This creates a *Receiver Operating Characteristic* (*ROC*) curve. The ROC curve for our model is displayed in Exhibit 13 (this is a top red triangle option in the **Partition** report).

Exhibit 13 Credit, ROC Curve for Offer Accepted



Offer Accepted	Area
🟩 Yes	0.7539
🟥 No	0.7539

Offer Accepted	Area
🟩 Yes	0.7369
🟥 No	0.7369

Conceptually, what the ROC curve measures is the ability of the predicted probability formulas to rank an observation. Here, we simply focus on the **Yes** outcome for the **Offer Accepted** response variable. We save the probability formula to the data table, and then sort the table from highest to lowest probability. If this probability model can correctly classify the outcomes for **Offer Accepted**, we would expect to see more **Yes** response values at the top (where the probability for **Yes** is highest) than **No** responses. Similarly, at the bottom of the sorted table, we would expect to see more **No** than **Yes** response values.

Constructing an ROC Curve

What follows is a practical algorithm to quickly draw an ROC curve after the table has been sorted by the predicted probability. Here, we walk through the algorithm for **Offer Accepted = Yes**, but this is done automatically in JMP for each response category.

For each observation in the sorted table, starting at the observation with the highest probability, **Offer Accepted = Yes**:

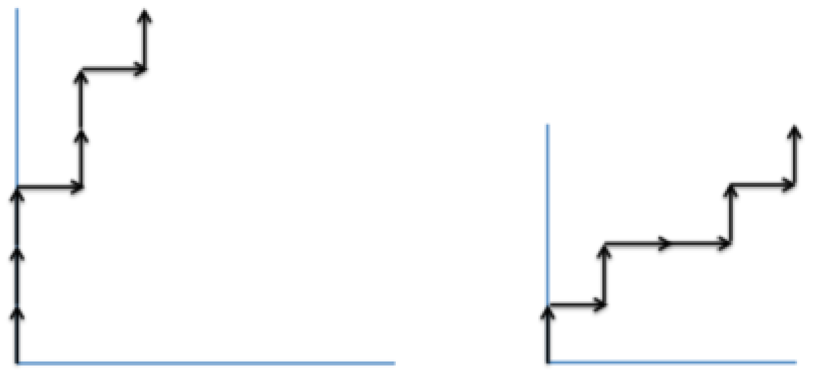
- If the observed response value is **Yes**, then a vertical line segment (increasing along the Sensitivity axis) is drawn. The length of the line segment is 1/(total number of **Yes** responses in the table).
- If the observed response value is **No**, then a horizontal line segment (increasing along the 1-Specificity axis) is drawn. The length of the line segment is 1/(total number of “No” responses in the data table).

Simple ROC Curve Examples

We use a simple example to illustrate. Suppose we have a data table with only 8 observations. We sort these observations from high to low based on the probability that the **Outcome** = **Yes**. The sorted actual response values are **Yes**, **Yes**, **Yes**, **No**, **Yes**, **Yes**, **No** and **Yes**. This results in the ROC curve on the left of Exhibit 14. Arrows have been added to show the steps in the ROC curve construction. The first 3 line segments are drawn up because the first three sorted values have **Outcome** = **Yes.**

Now, suppose that we have a different probability model that we use to rank the observations, resulting in the sorted outcomes **Yes**, **No**, **Yes**, **No**, **No**, **Yes**, **No** and **Yes**. The ROC curve for this situation is shown on the right of Exhibit 14. The first ROC curve moves “up” faster than the second curve. This is an indication that the first model is doing a better job of separating the **Yes** responses from the **No** responses based on the predicted probability.

Exhibit 14 ROC Curve Examples



Referring back to the sample ROC curve in Exhibit 13, we see that JMP has also displayed a diagonal reference line on the chart, which represents the *Sensitivity = 1-Specificity* line. If a probability model cannot sort the data into the correct response category, then it may be no better than simply sorting at random. In this case, the ROC curve for a “random ranking” model would be similar to this diagonal line. A model that sorts the data perfectly, with all the **Yes** responses at the top of the sorted table, would have an ROC Curve that goes from the origin of the graph straight up to sensitivity = 1, then straight over to 1-specificity = 1. A model that sorts perfectly can be made into a classifier rule that classifies perfectly; that is, a classifier rule that has a sensitivity of 1.0 and 1-specificity of 0.0.

The *area under the curve*, or *AUC* (labeled **Area** in Exhibit 13) is a measure of how well our model sorts the data. The diagonal line, which would represent a random sorting model, has an AUC of 0.5. A perfect sorting model has an AUC of 1.0. The area under the curve for **Offer Accepted** = **Yes** is 0.7369 (see Exhibit 13), indicating that the model predicts better than the random sorting model.

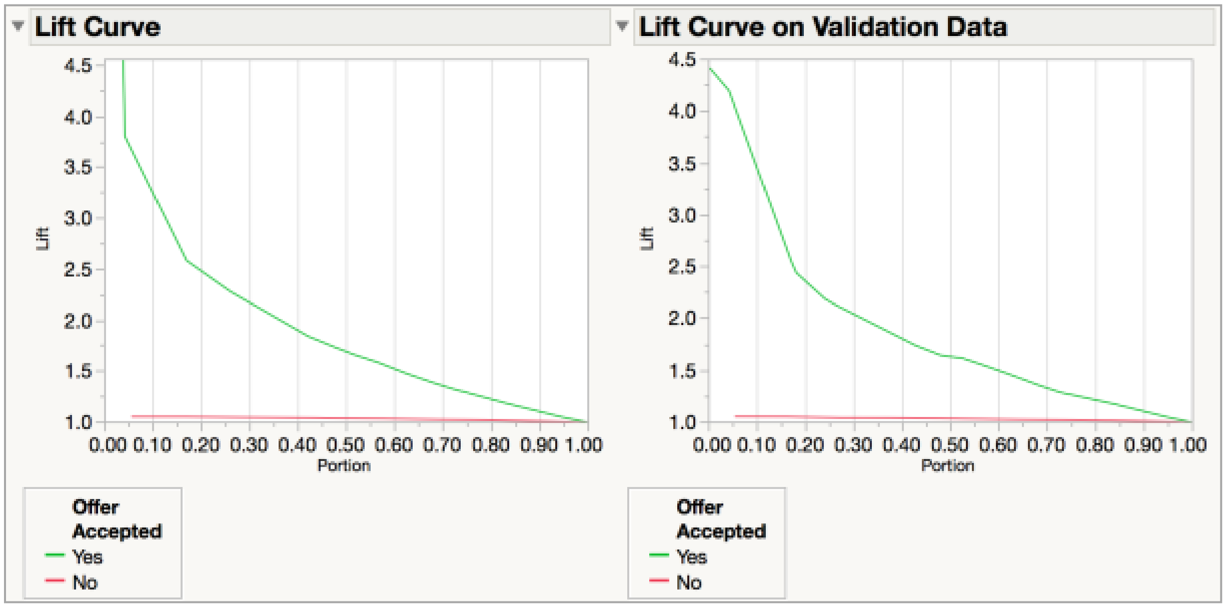
The Lift Curve

Another measure of how well a model can sort outcomes is the model ***lift***. As with the ROC curve, we examine the table that is sorted in descending order of predicted probability. For each sorted row, we calculate the sensitivity and divide that by the proportion of values in the table whereby **Offer Accepted** = **Yes**. This value is the model lift.

Lift is a measure of how much "richness" in the response we achieve by applying a classification rule to the data. A **Lift Curve** plots the **Lift** (on the *y*-axis) against the **Portion** (on the *x*-axis). Again, consider the data table that has been sorted by the predicted probability of a given outcome. As we go down the table from the top to the bottom, portion is the relative position of the row that we are considering. The top 10 percent of rows in the sorted table corresponds to a portion of 0.1, the top 20 percent of rows corresponds to a portion of 0.2, and so on. The lift for **Offer Accepted** = **Yes** for a given portion is simply the proportion of **Yes** responses in this portion, divided by overall proportion of **Yes** responses in the entire data table.

The higher the lift at a given portion, the better our model is at correctly classifying the outcome within this portion. For **Offer Accepted = Yes**, the lift at Portion = 0.15 is roughly 2.5 (see Exhibit 15). This means that in rows in the data table corresponding to the top 15% of the model's predicted probabilities, the number of actual **Yes** outcomes is 2.5 times higher than we would expect if we had chosen 15 percent of rows from the data set at random. If the model does not sort the data well, then the lift will hover at around 1.0 across all of portion values.

Exhibit 15 Lift Curve for Offer Accepted



Lift provides another measure of how good our model is at classifying outcomes; it is particularly useful when the overall predicted probabilities are lower than 0.5 for the outcome that we wish to predict.

Though in this example the majority of the predicted probabilities of **Offer Accepted = Yes** were less than 0.2, the lift curve indicates that there are threshold values that we could use with the predicted probability model to create a classifier rule that will be better than guessing at random. This rule can be used to identify portions of our data that contain a much richer number of customers who are likely to accept an offer.

For categorical response models, the misclassification rate, confusion matrix, ROC curve and lift curve all provide measures of model accuracy; each of these should be used to assess the quality of the prediction model.

Summary

Statistical Insights

In this case study, a classification tree was used to predict the probability of an outcome based on a set of predictor variables using the **Partition** platform. If the response variable is continuous rather than categorical, then a regression tree can be used to predict the mean of the response. Construction of regression trees is analogous to construction of classification trees, however splits are based on the mean response value rather than the probability of outcome categories.

Implications

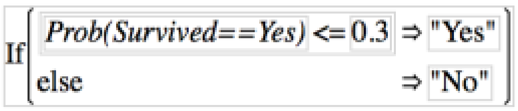
This model was created for explanatory rather than predictive purposes. Our goal was to understand the characteristics of customers most likely to accept a credit card offer. In a predictive model, we are more interested in creating a model that accurately predicts the response (i.e., predicts future customer behavior) than we are in identifying important variables or characteristics.

JMP Features and Hints

In this case study we used the **Columns Viewer** and **Distribution Platforms** to explore variables one at a time, utilizing **Fit Y by X** to explore the relationship between our response (or target variable) and predictor variables. This exploratory work was only partially completed herein.

We weighed the possibility of using the misclassification rate and confusion matrix as overall measures of model accuracy, and introduced the ROC and lift curves as additional measures of accuracy. As discussed, ROC and lift curves are particularly useful in cases where the probability of the target response category is low.

Note: The cutoff for classification used throughout JMP is 0.50. In some modeling situations it may be desirable to change the cutoff of classification (say, when the probability of response is extremely low). This effect can be achieved manually by saving the prediction formula to the data table, and then creating a new formula column that classifies the outcome based on a specified cutoff. In the sample formula below, JMP will classify an outcome as “Yes” if the predicted probability of survival is <= 0.30. The **Tabulate** platform (under **Analyze**) can then be used to manually create a confusion matrix.



$$\text{If}\begin{cases} \textit{Prob(Survived==Yes)} <= 0.3 & \Rightarrow \text{"Yes"} \\ \text{else} & \Rightarrow \text{"No"} \end{cases}$$

An add-in from the JMP User Community can also be used to change the cut-off for classification (community.jmp.com/docs/DOC-6901). This add-in allows the user to enter a range of values for the cutoff, and produces confusion matrices for each cutoff value. The goal is to find a cutoff that minimizes the misclassification rate on the validation set.

Exercises

Exercise 1: Use the **Credit Card Marketing BBM.jmp** data set to answer the following questions:

a. The Column Contributions output after 15 splits is shown in Exhibit 10. Interpret this output. How can this information be used by the company? What is the potential value of identifying these characteristics?
b. Recreate the output shown in Exhibits 9-11, but instead use the split button to manually split. Create a classification tree with 25 splits.
c. How did the Column Contributions report change?
d. How did the Misclassification Rate and Confusion Matrix change? How did the ROC or Lift Curves change? Did these additional splits provide any additional (useful) information?
e. Why is this an exploratory model rather than a predictive model? Describe the difference between exploratory and predictive models.

Exercise 2: Use the **Titanic Passengers.jmp** data set in the JMP Sample Data Library (under the Help menu) for this exercise.

This data table describes the survival status of 1,309 of the 1,324 individual passengers on the Titanic. Information on the 899 crew members is not included. Some of the variables are described below:

Name: Passenger Name

Survived: Yes or No

Passenger Class: 1, 2, or 3 corresponding to 1st, 2nd or 3rd class

Sex: Passenger sex

Age: Passenger age

Siblings and Spouses: The number of siblings and spouses aboard

Parents and Children: The number of parents and children aboard

Fare: The passenger fare

Port: Port of embarkment (C = Cherbourg; Q = Queenstown; S = Southampton)

Home/Destination: The home or final intended destination of the passenger

Build a classification tree for **Survived** by determining which variables to include as predictors. Do not use model validation for this exercise. Use **Column Contributions** and **Split History** to determine the optimal number of splits.

a. Which variables, if any, did you choose not to include in the model? Why?
b. How many splits are in your final tree?
c. Which variables are the largest contributors?
d. What is your final model? Save the prediction formula for this model to the data table (we will refer to it in the next exercise).

e. What is the misclassification rate for this model? Is the model better at predicting survival or non-survival? Explain.
f. What is the area under the ROC curve for **Survived**? Interpret this value. Does the model do a better job of classifying survival than a random model?
g. What is the lift for the model at portion = 0.1 and at portion = 0.25? Interpret these values.

Exercise 3: Use the **Titanic Passengers.jmp** data set for this exercise. Use the Fit Model platform to create a logistic regression model for **Survived?** using the other variables as predictors. Include interaction terms you think might be meaningful or significant in predicting the probability of survival.

For information on fitting logistic regression models, see the guide and video at community.jmp.com/docs/DOC-6794.

a. Which variables are significant in predicting the probability of survival? Are any of the interaction terms significant?
b. What is the misclassification rate for your final the logistic model?
c. Compare the misclassification rates for the logistic model and the partition model created in Exercise 2. Which model is better? Why?
d. Compare this model to the model produced using a classification tree. Which model would be easier to explain to a non-technical person? Why?