Boston Housing

Model Validation, Comparison and Selection in JMP® Pro

From *Building Better Models With JMP Pro*, Chapter 8, SAS Press (2015). Grayson, Gardner and Stephens.

Boston Housing

Model Validation, Comparison and Selection in JMP® Pro

Key ideas: Model validation, stepwise regression, regression trees, neural networks, validation statistics and model comparison.

Background

The objective of this study is to develop a model to predict the median value of homes in the Boston area. The data were originally collected and assembled in the mid-1970s (Harrison and Rubinfield, 1978), so this example is a bit dated. However, it is typical of a socioeconomic data set that is used to inform economic or public policy decisions, and the data set is well-known throughout the data mining community.

The Task

Our goal is to use the available data build a model that makes accurate predictions about home values in the Boston area. To ensure that the model predicts well for data not used to build the model, we use model validation. We will build different models (e.g., multiple regression, regression tree and neural network) in JMP Pro, compare the performance of these models, and select the best-performing model.

The Data   Boston Housing BBM.jmp

Each row in the data table is taken from a census tract (or town) in the Boston area. The response of interest is the continuous variable **mvalue**, which is the median home value (in $1,000) for towns in the Boston area in the 1970s.

Potential predictor variables in the data set are:

crim	Per capita crime rate for the census tract
zn	Proportion of a town's residential area zoned for lots larger than 25,000 square feet
indus	Proportion of non-retail business acres per town
chas	This is a 0/1 indicator variable. If the town bounds on the Charles River, the value is 0; otherwise the value is 1
Nox	Average annual nitrogen oxide concentrations in parts per hundred million
Rooms	Average number of rooms in owner-occupied units
Age	Proportion of owner units built prior to 1940
Distance	Weighted distances to five employment centers in the Boston region
Radial	Index of accessibility to radial highways
Tax	Full value property tax rate (per $10,000)
Pt	Pupil-to-teacher ratio by town school district
Lstat	Proportion of population that is "lower status," that is, proportion of adults without some high school education or that are classified as laborers

Analysis

The focus of this case study is model validation, comparison and selection. As such we omit perhaps the most important and time-consuming steps in the analytics process, data exploration and preparation[1]. A variety of graphical and statistical tools are used to identify potential gaps, explore potential data quality issues, and develop an in-depth understanding of the data and available variables.

We recommend the following approach to examining and understanding your data prior to modeling:

- Explore data one variable at a time, using **Cols > Cols Viewer**, **Analyze > Distribution**, and **Graph > Graph Builder**.
- Explore data two variables at a time, using **Graph Builder**, **Analyze > Tabulate** and **Fit Y by X**.
- Explore data three or more variables at a time using the same tools, plus **Rows > Data Filter**, the **Local Data Filter** (from the toolbar), **Graph > Scatterplot Matrix, Cols > Column Switcher**, and other graphical tools.

This initial data exploration provides insights into the available data, along with the steps needed to prepare the data for modeling. For example, you may need to address missing values, transform or derive new variables, or collect new data.

Note that we will also omit many of the background details for the different modeling approaches used. For additional information on multiple regression, regression trees and neural networks, refer to *Building Better Models With JMP Pro* or **JMP Help** (under the **Help** menu).

What Is Model Validation?

When we build a predictive model, there is a risk that we will model noise in the particular data set, rather than modeling the true relationship between predictors and the response. The resulting model may be overly complicated (*overfit*), or may not perform well when applied to new data. Overfitting can occur when a model is so complex that it attributes random noise in the response data to factors being used to make predictions. Validation helps us guard against this.

Here's the basic idea. When we use model validation, we withhold a subset of our data from the modeling process. The portion of data used to build the model is often referred to as the *training set*, and the data withheld is often referred to as the *validation set* (or the *holdout set*). We build the model using training data, and then apply the model to validation data to determine how well it performs. A third portion of the data, a *test set*, is also recommended. This subset is withheld from the modeling process, and is used to evaluate the final model.

There are different methods for validation available in the JMP modeling platforms: *Random holdback*, *k-fold cross-validation*, *excluded rows*, and using a *validation column* (the latter method is only available in JMP Pro). In this case study, we focus on using a validation column. With this approach, we divide the data set into a *training*, *validation* and (optionally) a *test* subset. In JMP Pro, this is done by creating an indicator column in the data table to specify the validation role for each row in the data table.

---

[1] For additional information on data exploration and preparing data for modeling, see *Building Better Models, Chapter 3 – Working with Data*.

A validation column, **Validation1**, has been added to the Boston Housing data table (see the last column in the data table shown in Exhibit 1). The first and fourth rows in the data table will be assigned to the validation set; the second, third and fifth rows will be assigned to the training set; the sixth row will be assigned to the test set, and so on.

Exhibit 1 A partial view of **t**he data table, with a validation column (Validation1)

	crim	zn	indus	chas	nox	rooms	age	distance	radial	tax	pt	lstat	mvalue	Validation1
1	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	4.98	24	Validation
2	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	9.14	21.6	Training
3	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	34.7	Training
4	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	2.94	33.4	Validation
5	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	5.33	36.2	Training
6	0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	5.21	28.7	Test
7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	12.43	22.9	Training
8	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	19.15	27.1	Training
9	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	29.93	16.5	Validation
10	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	17.1	18.9	Validation
11	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	20.45	15	Training
12	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	13.27	18.9	Training
13	0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	15.71	21.7	Validation
14	0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	8.26	20.4	Training
15	0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	10.26	18.2	Training

Rows in the training set will be used to build the model, rows in the validation set will be used to determine how complex the model needs to be, and rows in the test set will be used to assess how well the model will perform when applied to new data.

Note that the specific way in which the validation set is used may differ between modeling platforms[2]. For example:

- In forward stepwise regression, validation is used to decide when to stop adding additional terms into the model,
- In decision trees, validation is used to decide when to stop splitting the data, and
- In neural networks, validation is used to determine the best value for the penalty parameter used to prevent overfitting of the neural network.

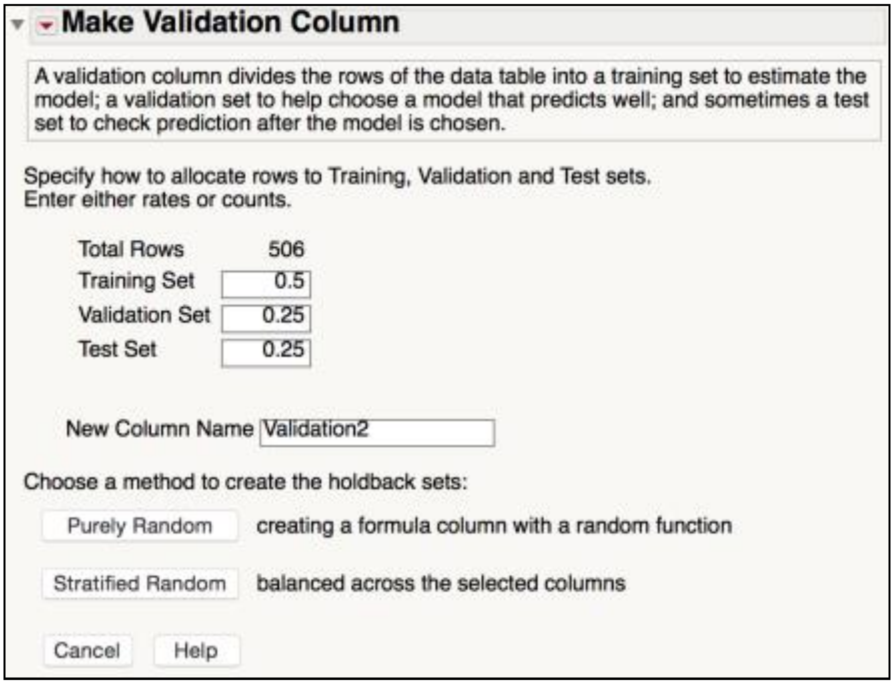
Creating Training, Validation and Test Subsets

One way to create a validation column in JMP Pro is to use the **Make Validation Column** utility (under **Cols > Model Utilities**). In the **Make Validation Column** dialog in Exhibit 2, we request a 50 percent training, 25 percent validation, and 25 percent test split. In this example, we use the **Purely Random** method to create the holdback sets. The name of the validation column that we create is **Validation2**.

Note: The **Stratified Random** option will perform a random selection with balance across variables that are currently selected in the data table. For instance, if we want each validation subset to contain the same relative proportions of **chas** = ***0*** and **chas** = ***1***, we could use **chas** as the stratification variable (that is, we would select the variable **chas** prior to launching the **Make Validation Column** utility).

[2] For more details on how validation is applied in the specific JMP Pro modeling platforms, see *Building Better Models, Chapter 8 – Using Cross Validation*, or refer to *Fitting Linear Models* or *Specialized Models* under **Help > Books**.

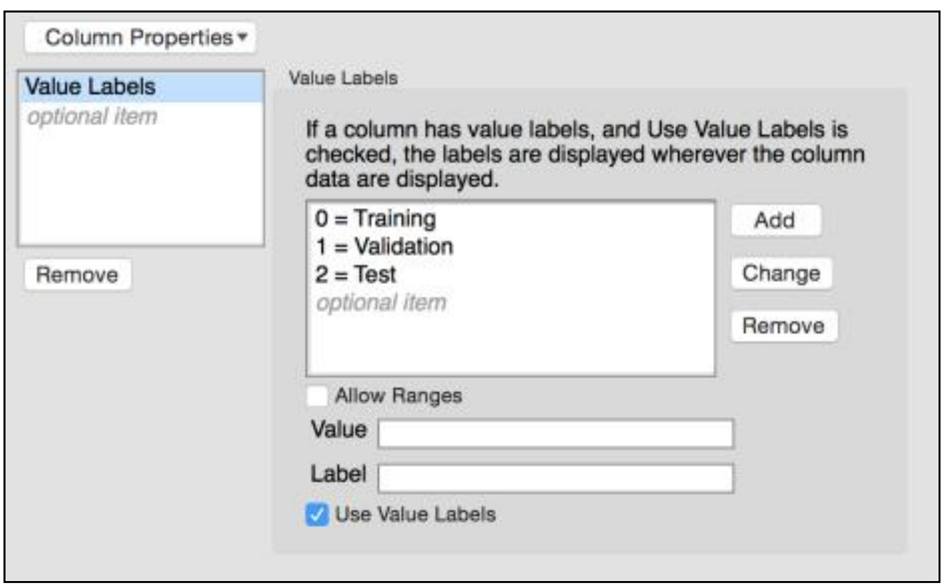
Exhibit 2 Using the Make Validation Column Utility



A new column, **Validation2**, is added to the data table (this column was previously created; we will use this column throughout this case study).

Note that the actual values stored are the numbers **0**, **1** and **2**, but the utility creates a **Value Labels** column property to display descriptive labels **Training**, **Validation** and **Test** in place of numeric values in the data table and output reports (to view the column properties, right click on the column name in the data table and select **Column Info**).

Exhibit 3 The Value Labels column property for the validation column



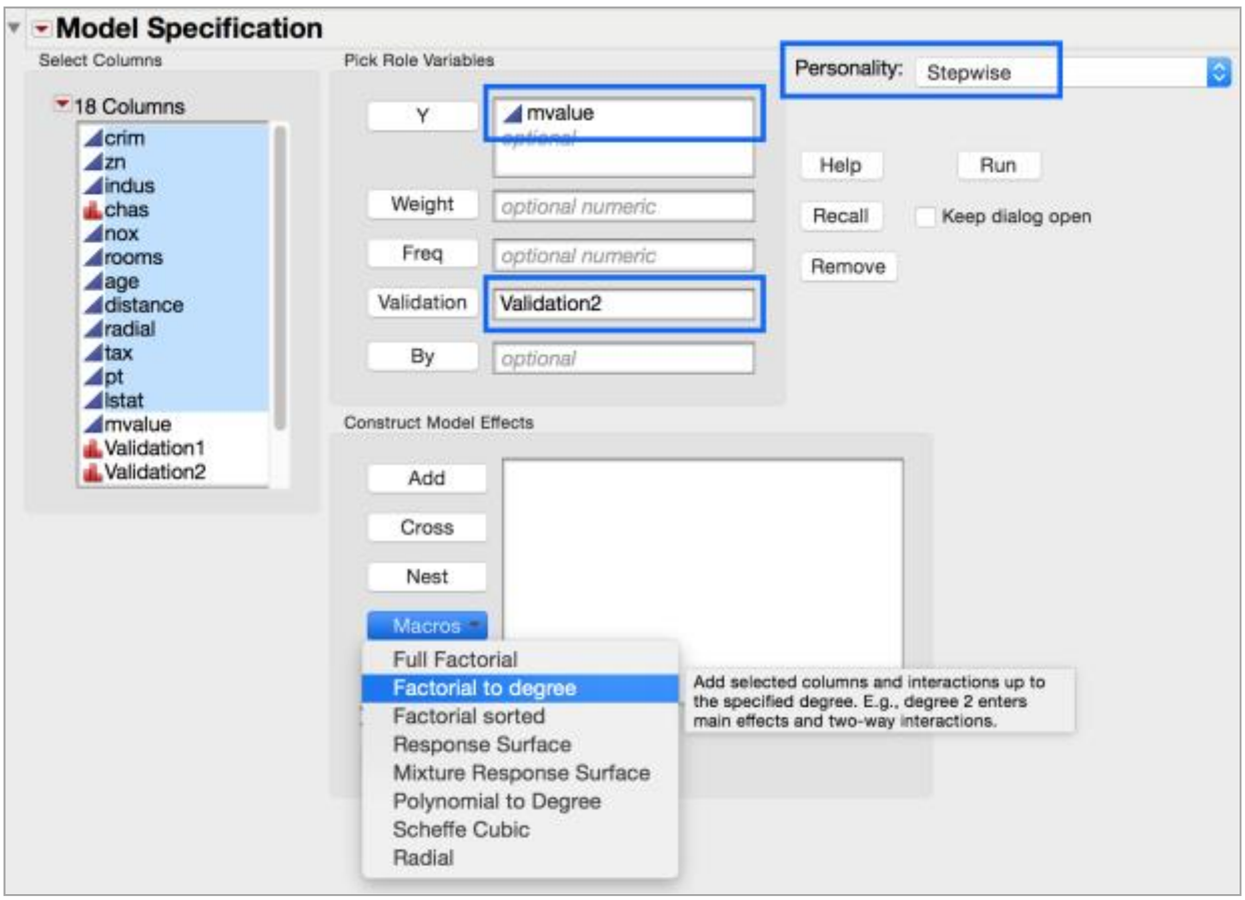
Using Cross-Validation to Build a Linear Regression Model

We now use the validation column (**Validation2**) in developing a model to predict the median home value for Boston Housing data. Since we have several potential predictors, the first modeling method that we use is stepwise regression[3].

We specify the model as shown in Exhibit 4, using the **Fit Model** dialog (from the **Analyze** menu). We enter **mvalue** as the Y variable. In this case, we specify a more complicated model than we will likely need. We enter all predictor variables as main (linear) effects, and then enter all of their two-factor interactions. To do this, we select predictors from the **Select Columns** list and **Factorial to Degree** from the **Macros** menu. The default "degree" that the model specification macro uses is "2". This adds the 12 factors (or main effects) plus 66 two-way (second-order) interaction terms to the list of model effects.

Finally, we specify the **Validation2** column for the **Validation** role, and select **Stepwise** from the list of modeling "Personalities."

Exhibit 4 Specifying a Stepwise Model with Interactions and Validation



[3] For more information on stepwise regression, see *Building Better Models With JMP Pro, Chapter 4 – Multiple Linear Regression*.

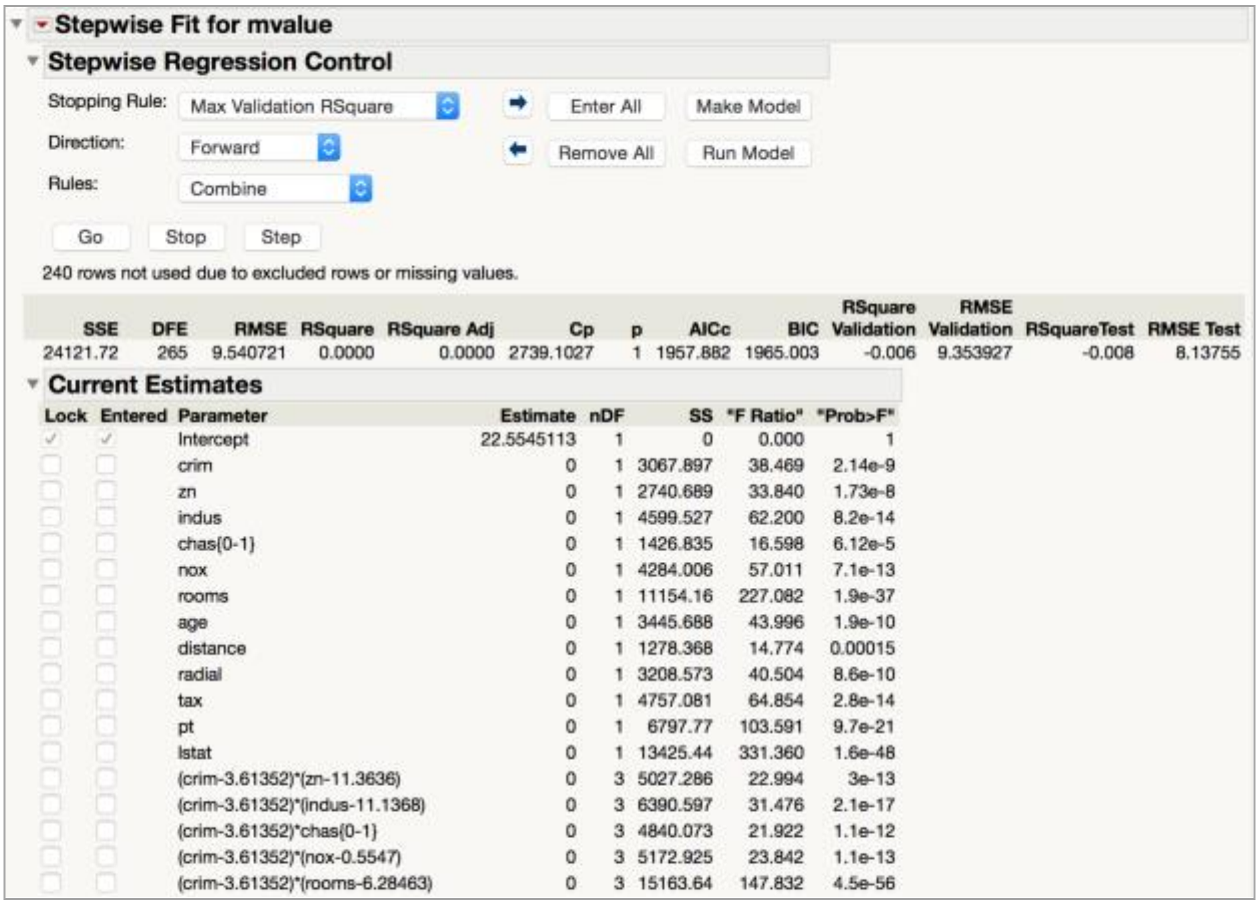
Choosing the Regression Model Terms With Stepwise Regression

We click **Run**, which opens the stepwise regression dialog (Exhibit 5).

Since we are using validation, the stopping rule that determines which model is selected is **Max Validation RSquare**. When we click **Go**, the forward stepwise regression modeling process begins. Terms are added to the model one at a time. Each term that is added to the model term (from the remaining terms) is the one that has the lowest *p*-value.

This process continues until there is no improvement in the **RSquare Validation** value. JMP will then continue to add terms (up to ten) in search of a higher value. The final model is the one resulting in the highest RSquare Validation

Exhibit 5 Boston Housing Stepwise Initial Dialog



SSE	DFE	RMSE	RSquare	RSquare Adj	Cp	p	AICc	BIC	RSquare Validation	RMSE Validation	RSquareTest	RMSE Test
24121.72	265	9.540721	0.0000	0.0000	2739.1027	1	1957.882	1965.003	-0.006	9.353927	-0.008	8.13755

Current Estimates

Lock	Entered	Parameter	Estimate	nDF	SS	"F Ratio"	"Prob>F"
✓	✓	Intercept	22.5545113	1	0	0.000	1
		crim	0	1	3067.897	38.469	2.14e-9
		zn	0	1	2740.689	33.840	1.73e-8
		indus	0	1	4599.527	62.200	8.2e-14
		chas{0-1}	0	1	1426.835	16.598	6.12e-5
		nox	0	1	4284.006	57.011	7.1e-13
		rooms	0	1	11154.16	227.082	1.9e-37
		age	0	1	3445.688	43.996	1.9e-10
		distance	0	1	1278.368	14.774	0.00015
		radial	0	1	3208.573	40.504	8.6e-10
		tax	0	1	4757.081	64.854	2.8e-14
		pt	0	1	6797.77	103.591	9.7e-21
		lstat	0	1	13425.44	331.360	1.6e-48
		(crim-3.61352)*(zn-11.3636)	0	3	5027.286	22.994	3e-13
		(crim-3.61352)*(indus-11.1368)	0	3	6390.597	31.476	2.1e-17
		(crim-3.61352)*chas{0-1}	0	3	4840.073	21.922	1.1e-12
		(crim-3.61352)*(nox-0.5547)	0	3	5172.925	23.842	1.1e-13
		(crim-3.61352)*(rooms-6.28463)	0	3	15163.64	147.832	4.5e-56

Exhibit 6 shows the history of terms added to the model. For each step we see a number of statistics, including the **RSquare Validation**.

In Step 18, JMP reports that the best RSquare Validation, across all steps, is 0.8002. This corresponds to the **RSquare Validation** at Step 7. So, the model selected is the model produced at Step 7.

Exhibit 6 Boston Housing Stepwise Regression Step History

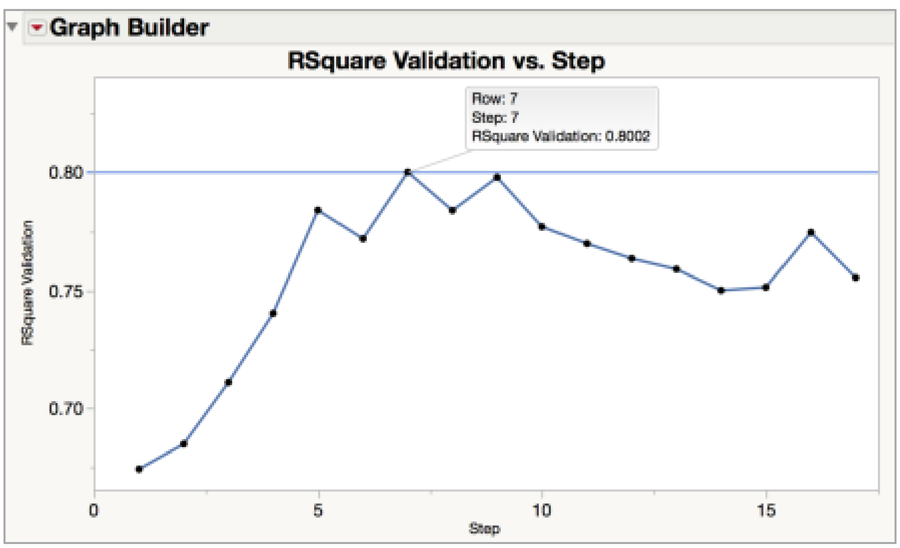
Step History

Step	Parameter	Action	"Sig Prob"	Seq SS	RSquare	Cp	p	AICc	BIC	RSquare Validation
1	(rooms-6.28463)*(lstat-12.6531)	Entered	0.0000	18257.84	0.7569	472.04	4	1587.86	1605.55	0.6742
2	(rooms-6.28463)*(pt-18.4555)	Entered	0.0000	861.4629	0.7926	368.79	6	1549.8	1574.45	0.6848
3	(distance-3.79504)*(lstat-12.6531)	Entered	0.0000	602.6045	0.8176	297.77	8	1519.93	1551.47	0.7109
4	(nox-0.5547)*(rooms-6.28463)	Entered	0.0000	341.0061	0.8317	259.31	10	1502.8	1541.18	0.7403
5	(crim-3.61352)*chas{0-1}	Entered	0.0000	521.2242	0.8533	200.42	13	1472.88	1521.37	0.7840
6	chas{0-1}*(tax-408.237)	Entered	0.0002	227.8395	0.8628	176.06	15	1459.68	1514.83	0.7718
7	(radial-9.54941)*(lstat-12.6531)	Entered	0.0002	220.0105	0.8719	152.66	17	1445.97	1507.7	0.8002
8	(rooms-6.28463)*(radial-9.54941)	Entered	0.0000	203.4196	0.8803	129.34	18	1430.17	1495.17	0.7839
9	(age-68.5749)*(distance-3.79504)	Entered	0.0003	184.9772	0.8880	110.31	20	1417.25	1488.72	0.7979
10	(crim-3.61352)*(pt-18.4555)	Entered	0.0029	96.17724	0.8920	100.34	21	1409.99	1484.66	0.7769
11	(crim-3.61352)*(lstat-12.6531)	Entered	0.0013	108.0885	0.8965	88.879	22	1401.11	1478.97	0.7698
12	(age-68.5749)*(radial-9.54941)	Entered	0.0002	138.8827	0.9022	73.589	23	1388.31	1469.33	0.7635
13	(nox-0.5547)*(age-68.5749)	Entered	0.0002	129.0342	0.9076	59.524	24	1375.78	1459.95	0.7590
14	chas{0-1}*(rooms-6.28463)	Entered	0.0017	89.20242	0.9113	50.419	25	1367.37	1454.67	0.7500
15	(crim-3.61352)*(nox-0.5547)	Entered	0.0117	56.08521	0.9136	45.436	26	1362.78	1453.19	0.7513
16	(crim-3.61352)*(rooms-6.28463)	Entered	0.0038	72.10261	0.9166	38.46	27	1355.92	1449.4	0.7746
17	(rooms-6.28463)*(age-68.5749)	Entered	0.0163	48.25812	0.9186	34.452	28	1351.98	1448.53	0.7554
18	Best	Specific	.	.	0.8719	152.66	17	1445.97	1507.7	0.8002

To get a better feel for how stepwise builds the model, we save the step history to a data table and use the **Graph Builder** to create a graph of **step** versus **RSquare Validation** (see Exhibit 7).

The model that was found after seven steps has a **Validation RSquare** of 0.8002. After 10 additional forward steps, **the Validation RSquare** does not get any larger. As such, the model that was found at Step 7 is chosen as the best model.

Exhibit 7 Graph of Stepwise Step History



(To save a table of JMP output to a data table, right-click on the table and select **Save to Data Table**. To create the graph, drag **Step** to the **X** zone, drag **RSquare Validation** to the **Y** zone, and then click on the line icon above the graph.)

To create the chosen model, we select **Make Model** in the Stepwise Fit window. This opens the **Fit Model** dialog with the chosen model specified. We then click **Run** to fit the model.

The fitted model results for our example are shown in Exhibit 8. As expected, the **RSquare** for the validation set is 0.8002 (see the bottom panel in Exhibit 8).

Exhibit 8 Fitted Model from Stepwise Regression

Response mvalue

Effect Summary

Actual by Predicted Plot

Summary of Fit

RSquare	0.871911
RSquare Adj	0.86368
Root Mean Square Error	3.522577
Mean of Response	22.55451
Observations (or Sum Wgts)	266

Analysis of Variance

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	44.621362	5.756732	7.75	<.0001*
crim	3.570038	0.716813	4.98	<.0001*
chas[0]	-7.558841	1.263498	-5.98	<.0001*
nox	-13.42288	4.244065	-3.16	0.0018*
rooms	3.2955199	0.450062	7.32	<.0001*
distance	-0.868399	0.193128	-4.50	<.0001*
radial	0.2779489	0.071045	3.91	0.0001*
tax	-0.049922	0.009974	-5.01	<.0001*
pt	-0.690317	0.132594	-5.21	<.0001*
lstat	-0.637494	0.058856	-10.83	<.0001*
(crim-3.61352)*chas[0]	-3.69331	0.715882	-5.16	<.0001*
chas[0]*(tax-408.237)	0.0390813	0.009534	4.10	<.0001*
(nox-0.5547)*(rooms-6.28463)	-11.5764	4.234603	-2.73	0.0067*
(rooms-6.28463)*(pt-18.4555)	-0.881481	0.161285	-5.47	<.0001*
(rooms-6.28463)*(lstat-12.6531)	-0.33509	0.049402	-6.78	<.0001*
(distance-3.79504)*(lstat-12.6531)	0.0405233	0.023826	1.70	0.0902
(radial-9.54941)*(lstat-12.6531)	-0.014773	0.004937	-2.99	0.0030*

Effect Tests

Crossvalidation

Source	RSquare	RASE	Freq
Training Set	0.8719	3.4082	266
Validation Set	0.8002	4.1688	127
Test Set	0.7195	4.2928	113

Saving the Prediction Formula

We save the model results (use the red triangle and then select **Save Columns > Prediction Formula**). This creates a new column—**Pred Formula mvalue**. This saved column contains the formula that calculates the predicted value for each row in the data table, and we can use this formula to predict new values. As we will see, we can also use a saved prediction formula to compare the performance of this model to others that we have created.

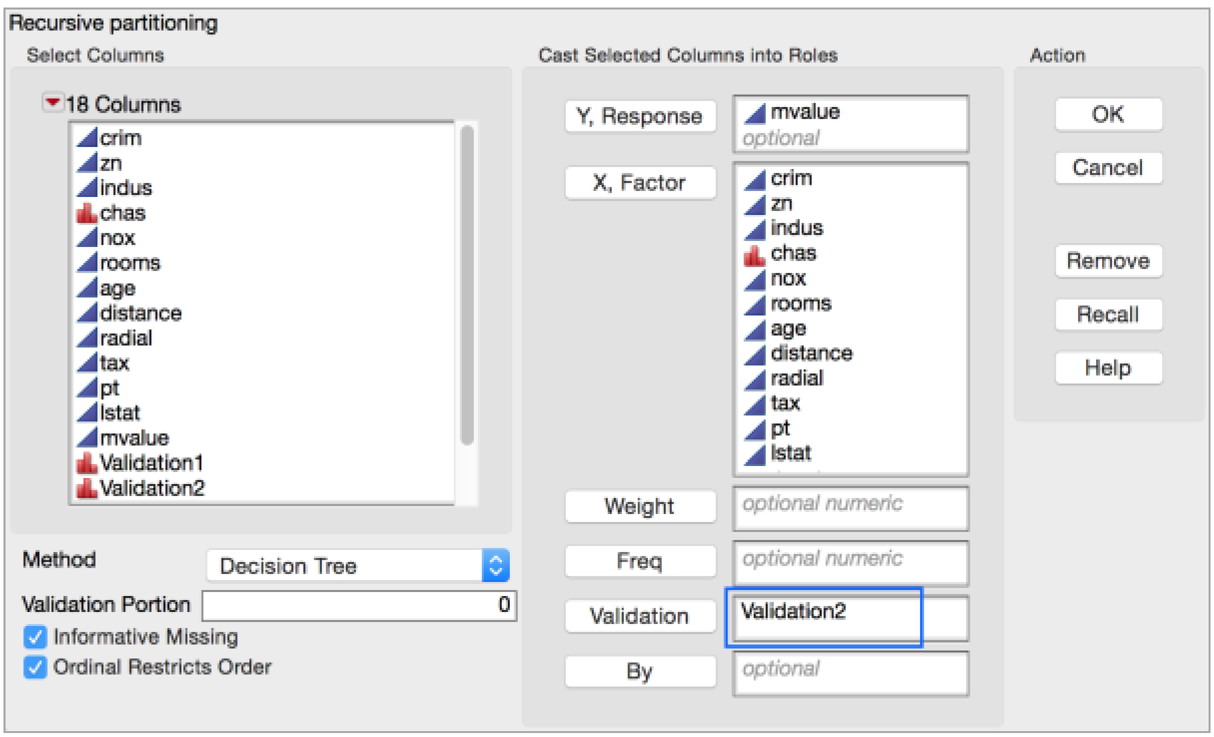
Using Validation to Build a Decision Tree Model

Even though the process described above helped us to choose the “best” linear regression model, the final chosen model is just one of many options available for building prediction models. Decision trees can sometimes have better predictive properties than regression models, so we fit a simple decision tree model to predict **mvalue**. We use validation to guide the decision tree-building process.

In this situation, splits in the decision tree are determined by the LogWorth statistic[4]. Here, we again use **Validation2** to specify the training, validation and test subsets. After each split of the decision tree, the validation **RSquare** is calculated. When the validation **RSquare** stops improving, the decision tree will continue to split in order to determine whether any improvement can be attained. It automatically stops splitting if there is no improvement after 10 additional splits.

Exhibit 9 shows the **Partition** platform launch dialog. We specify **mvalue** as the response and all of the potential predictors as factors. **Validation2** is chosen as the validation role column. (**Partition** is an option under **Analyze > Modeling**). We use the default settings and click **OK** to launch the analysis.

Exhibit 9 Boston Housing Partition Dialog with Validation2

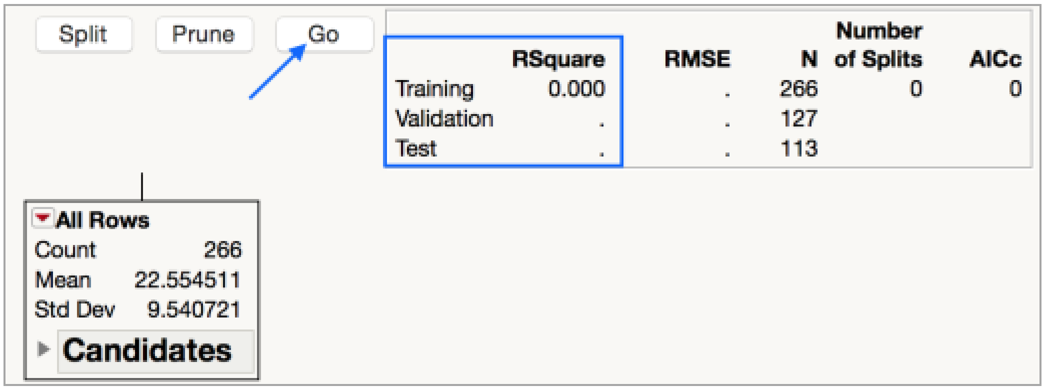


The initial model fit report is shown in Exhibit 10. Since we are using model validation, JMP reports **Training**, **Validation** and **Test RSquare**, along with other statistics.

[4] See *Building Better Models With JMP Pro, Chapter 6 - Decision Trees*, or the **JMP Help** for details.

We click **Go** to begin the automatic decision tree-building process.

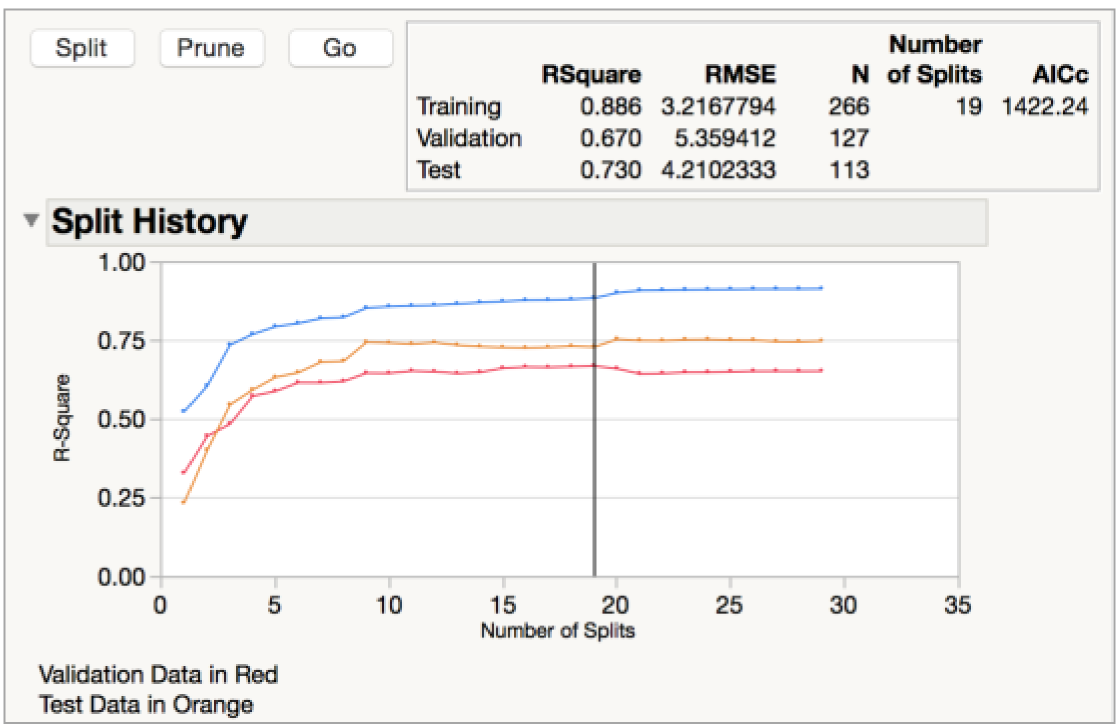
Exhibit 10 Boston Housing Initial Partition Output With Cross-Validation



In Exhibit 11, we see statistics for the final model and the **Split History** graph. The bottom line in the **Split History** graph corresponds to the validation **RSquare**, the middle is the test **RSquare**, and the top line is the training **RSquare**. The largest tree that was built had 29 splits. However, JMP stops adding branches to the tree after 10 splits if there is no further improvement in validation **RSquare**.

The chosen decision tree model—or the tree with the best validation **RSquare** (0.670)—has only 19 splits. Recall that validation **RSquare** for the stepwise model was 0.8002. Based on this statistic alone, the stepwise model outperforms the regression tree.

Exhibit 11 Fitted Decision Tree Model With Cross-Validation



We use the chosen decision tree model and save the prediction formula to the data table (use the top red triangle, **Save Columns > Save Prediction Formula**). This creates a new column, **mvalue Predictor**.

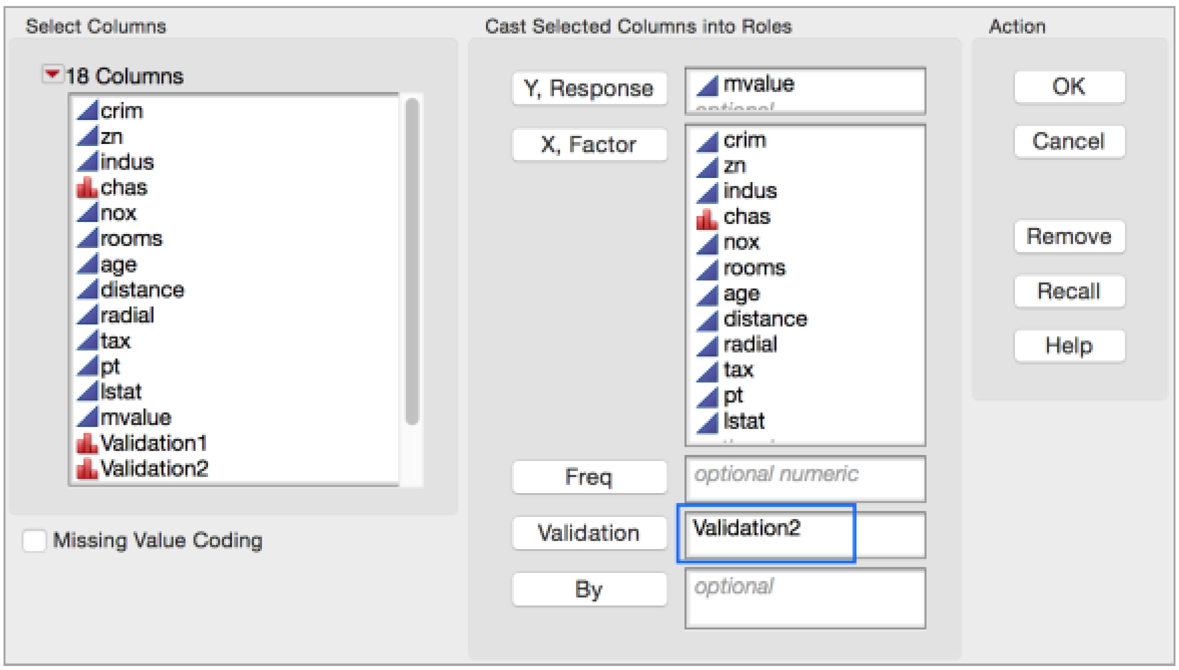
Fitting a Neural Network Model Using Validation

A third model is developed using the **Neural** platform (**Neural** is an option under **Analyze > Modeling**).

Neural networks are highly flexible, and can be used to model either categorical or continuous responses. Due to their flexibility, neural network models can often outperform other modeling methods; however, they are also prone to overfitting. As such, some form of validation is required when building neural network models in JMP[5].

We fit a neural network, again using the **Validation2** as the validation column (Exhibit 12).

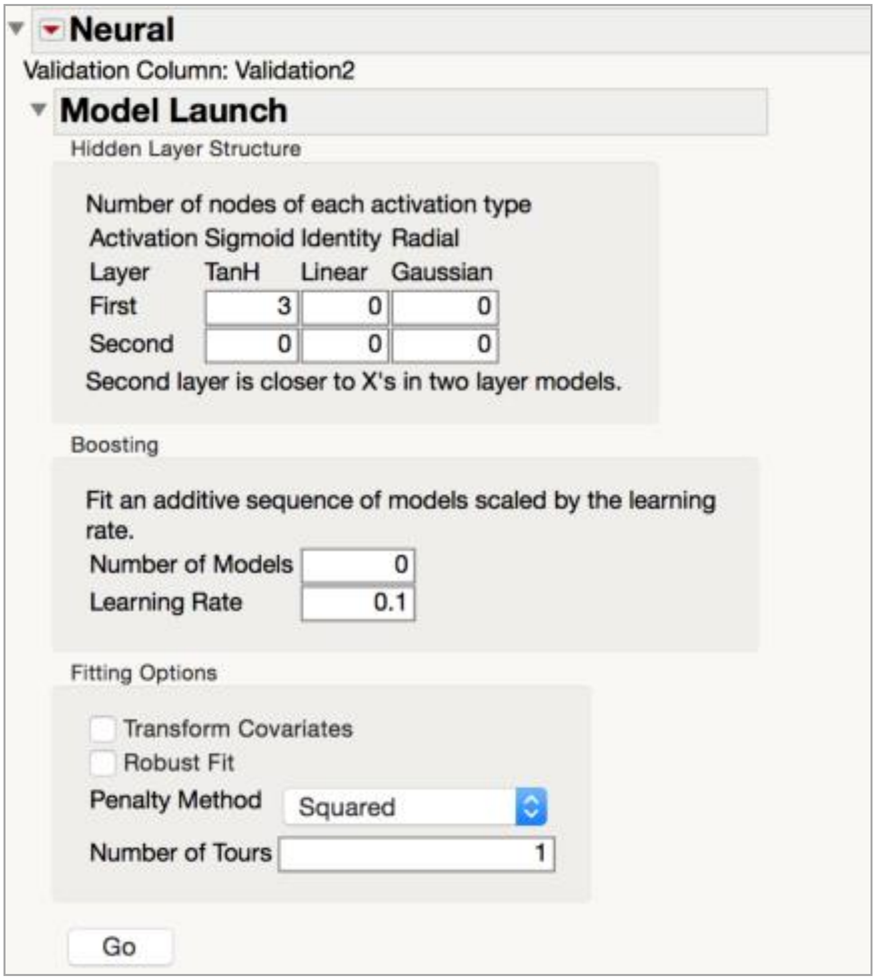
Exhibit 12 Boston Housing Neural Model Launch Dialog With Cross-Validation



The default model has one hidden layer and three "sigmoid" TanH nodes (see Exhibit 13). We use the default settings, and click **Go** to fit the neural model.

[5] For more information on fitting neural networks, see *Building Better Models With JMP Pro*, *Chapter 7 – Neural Networks*, or refer to the **JMP Help**.

Exhibit 13 Boston Housing Neural Model Launch



The results for the fitted model are shown in Exhibit 14.[6] The **RSquare** for the validation set is 0.8151; the highest validation **RSquare** of the three models we've created.

To view the model, we select **Diagram** from the red triangle for the model. The diagram shows the structure of a neural network model. Our model has an input layer with input variables (predictors), a hidden layer with three nodes, and an output layer. In each node in the hidden layer, a linear combination of input variables is transformed using the *TanH* function (which is similar to the logistic function). The output layer, which is used to predict the response (**mvalue**), is a linear combination of the outputs from the hidden layer.

---

[6] To obtain the same result for the neural model in this example, set the random seed to 1000 prior to launching the neural platform using the Random Seed Reset add-in, which can be installed from /community.jmp.com/docs/DOC-6601. Note that Windows and Mac may produce different results with the same random seed.

Exhibit 14 Boston Housing Neural Fitted Model with Cross-Validation

Neural

Validation Column: Validation2

Model Launch

Model NTanH(3)

Training — **mvalue**

Measures	Value
RSquare	0.8951951
RMSE	3.0828625
Mean Abs Dev	2.1596555
-LogLikelihood	676.91602
SSE	2528.075
Sum Freq	266

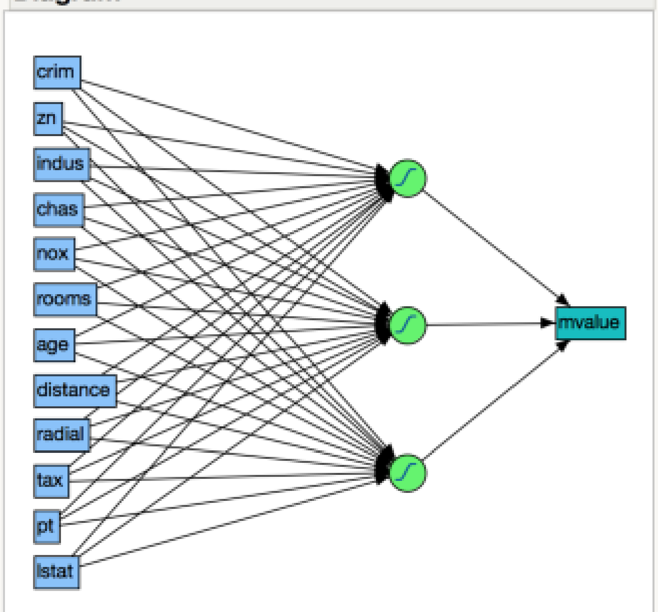
Validation — **mvalue**

Measures	Value
RSquare	0.8151387
RMSE	4.009683
Mean Abs Dev	2.7985532
-LogLikelihood	356.57164
SSE	2041.8498
Sum Freq	127

Test — **mvalue**

Measures	Value
RSquare	0.8294569
RMSE	3.3474768
Mean Abs Dev	2.6001761
-LogLikelihood	296.86743
SSE	1266.2329
Sum Freq	113

Diagram



As with the two previous models (i.e., stepwise regression and regression tree), we save the prediction formula to the data table (from the fitted model's red triangle, choose **Save Columns > Save Profile Formulas**). The new column is named **Predicted mvalue.**

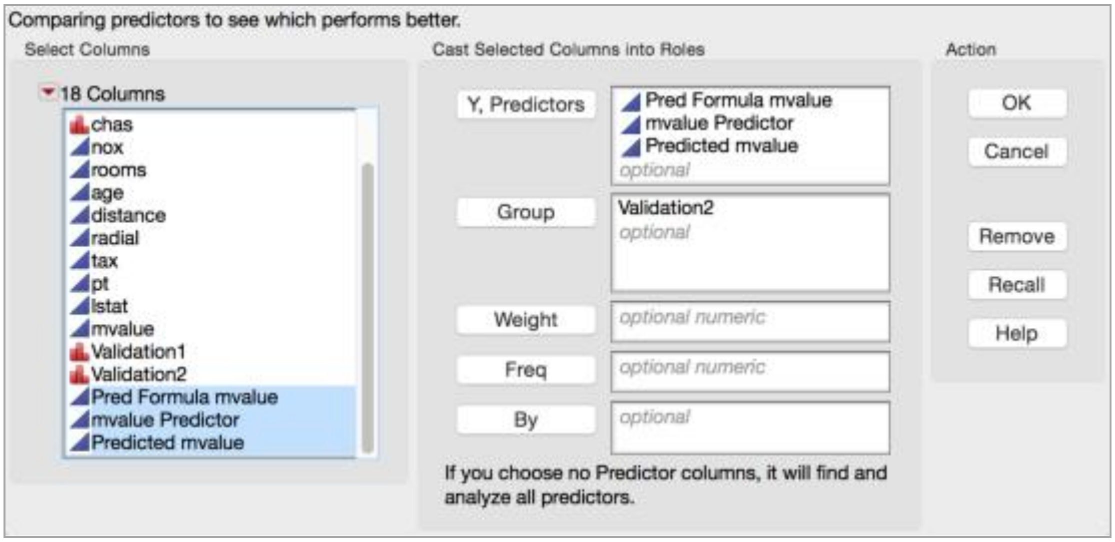
(Note: To view the hidden layer and output layer formulas as described above, select **Save Formulas** instead of **Save Profiler Formulas**. Formulas for the hidden layers are saved as **H1_1**, **H1_2**, and **H1_3**. The formula for the output layer, which is a linear combination of the hidden layers, is saved as **Predicted mvalue**. Right-click on the column names in the data table and select **Formula** to view the formulas.)

Model Comparison

We have fit three different models using the same validation column, and have saved prediction formulas for the three models to the data table. In JMP Pro, we use the **Model Comparison** platform to compare the performance of these models (select **Analyze > Modeling > Model Comparison**).

We specify the columns with prediction formulas for each model as **Y, Predictors** and select **Validation2** as the **Group** variable. Note that an alternative is to leave the **Y, Predictors** field blank and use **Validation2** as the **By** variable. With this setup, JMP will find columns with saved prediction formulas for you.

Exhibit 15 Model Comparison Dialog



In the resulting Model Comparison report (Exhibit 16), we use statistics from the test set for model comparison. Recall that the test set was not used in building our models. Since this set was completely withheld from all modeling efforts, it provides an "honest" assessment of model performance.

In addition to **RSquare**, the following statistics are provided:

- RASE (Root Average Squared Error) – this is similar to RMSE (Root Mean Square Error), but is not based on the degrees of freedom for the model. This makes it better than RMSE when comparing different types of models. For this measure, lower is better.
- AAE (Average Absolute Error) – a measure of the average prediction error. Lower is better.

The neural network model has the highest **RSquare** value (0.8295). It also has the lowest RASE. However, the AAE is slightly higher than the AAE for the Least Squares model.

Exhibit 16 Model Comparison Report

Model Comparison

Predictors

Measures of Fit for mvalue

Validation2	Predictor	Creator	.2.4.6.8	RSquare	RASE	AAE	Freq
Training	Pred Formula mvalue	Fit Least Squares		0.8719	3.4082	2.4207	266
Training	mvalue Predictor	Partition		0.8859	3.2168	2.1457	266
Training	Predicted mvalue	Neural		0.8952	3.0829	2.1597	266
Validation	Pred Formula mvalue	Fit Least Squares		0.8002	4.1688	2.8230	127
Validation	mvalue Predictor	Partition		0.6697	5.3594	3.3003	127
Validation	Predicted mvalue	Neural		0.8151	4.0097	2.7986	127
Test	Pred Formula mvalue	Fit Least Squares		0.7195	4.2928	2.5259	113
Test	mvalue Predictor	Partition		0.7302	4.2102	3.2313	113
Test	Predicted mvalue	Neural		0.8295	3.3475	2.6002	113

If we have to choose only one of these models from the perspective of performance and prediction error, we would select the neural network model.

Additional options for comparing models, such as the **Profiler**, are available from the top red triangle in the model comparison window.

Summary

Statistical Insights

The use of validation (or cross-validation) is central to our ability to develop predictive models that neither overfit nor underfit our data. As such, it is one of the most important concepts in predictive modeling. In many situations, we use two holdout subsets: training and validation. If we also withhold a test subset, then we have the ability to assess model performance on an unbiased sample that has not been used in the model building or model selection processes. This allows for good model evaluation, and makes it easy to compare and choose among competing models.

Implications

With the exception of the stepwise regression model—which included two-way interactions—we created relatively simple models using the default settings. To create a regression tree, we used the decision tree option, however the other tree methods (i.e., bootstrap forest and boosted tree) may lead to better performing models. We also created a simple neural network model using the default single hidden layer. Adding additional nodes with different activation functions, using two layers, and using options such as boosting may lead to models with improved predictive performance (at the cost of complexity).

In this case study we built predictive models for a continuous response. For classification models—which have categorical responses—the approach to model validation, comparison and selection is analogous. However, other statistics, such as the misclassification rate, are reported instead of RASE and AAE. Other options, such as the **ROC Curve**, **Lift Curve**, and **Confusion Matrix** are available from the top red triangle in the modeling platforms and in the **Model Comparison** platform.

JMP Features and Hints

We used the **Make Validation Column** utility to create a validation column, building three predictive models: a stepwise regression model using **Fit Model**, a regression tree (using **Partition**), and a neural network (using **Neural**). We saved prediction formulas for these models to the data table, and then used the **Model Comparison** platform to compare the performance of the three models on the test set.

Exercises

Exercise 1: Open the example data set Boston Housing BBM.jmp.

a. Use the **Make Validation Column** utility to create random training, validation and test subsets. Use 50 percent of the data for training, 25 percent for validation and 25 percent for the test.

b. Repeat Part A, but first select the column **chas** in the data table, and then create a stratified random sample.

c. When creating a validation column, it is important that the distributions of the variables across training, validation and test sets are similar.

    i. Explain why this is important.

    ii. For the validation column produced in part b, use built-in tools such as **Distribution**, **Tabulate**, and **Graph Builder** to explore the makeup of training, validation and test sets with respect to the response variable and predictor variables.

    Note: *If doing this as part of a class exercise, to produce the same validation set as others, set the random seed to 1,000 before creating the validation column using the add-in described in Footnote 6.*

Exercise 2: Use Boston Housing BBM.jmp for this exercise.

a. Create final prediction models with the validation column that you created in Exercise 1b, using the following modeling methods:

    1. Forward stepwise regression
    2. Decision tree
    3. Neural network

b. Compare the final models using the **Model Comparison** platform.

    1. Using the default statistics provided, is there one model that stands out as being better?
    2. What criteria did you use to determine the best-performing model?
    3. Describe the criteria that you used in Part 2.
    4. Explore the options available under the red triangle. Explain how the Profiler and first two plots can be useful in comparing and selecting the best performing model.
    5. Explain the value of using a test set. Why is it important to use a holdout sample for model evaluation and selection?

Exercise 3: Use the Credit Card Marketing BBM.jmp data for this exercise. This data set is described in the classification tree case study, and in Example 1 in Chapter 6 of *Building Better Models with JMP Pro*. The response is **Offer Accepted**.

a. Create a distribution of **Offer Accepted**. The target category is **Yes**. What percent of offers were accepted?

b. Use the **Make Validation Column** modeling utility to create training, validation and test sets, stratified on **Offer Accepted**. Again, if doing this as part of a class exercise, set the random seed to 1,000 prior to creating the validation column.

c. Explore this validation column as you did in Exercise 1c. What percent of each set is **Offer Accepted** = **Yes** and **Offer Accepted** = **No**? Are there any issues with using this validation column?

d. Create a decision tree and neural model to predict **Offer Accepted** using this validation column. Then compare these models using the **Model Comparison** platform.

    Which is your best-performing model? What is the misclassification rate?

SAS Institute Inc. World Headquarters **+1 919 677 8000**