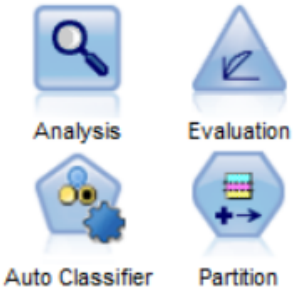
Section 9:
Assessing and Selecting Predictive Models

- Assessing Model Performance
- The Analysis Node
- Success Criteria Scenarios
- The Evaluation Node
- The Partition Node
- The Auto Classifier Node



As we have already seen, the CRISP-DM process explicitly contains a step for assessing the results of the modelling stage: Evaluation. It's important to understand that what might be regarded as an excellent model in one set of circumstances might also be regarded as completely inadequate in another. Consider the issue of false positives and false negatives. Models always generate some errors for any outcome that we try to predict or estimate. In fact, in most cases models that claim close to 100% accuracy are usually badly flawed. Nevertheless, the goal of the modelling process is of course to try to minimise this error. But even when trying to predict a two-category outcome, we can't always assume the level of accuracy in both outcomes is the same. In other words, just because the model is 85% accurate *overall,* doesn't mean that it is 85% accurate in predicting *both* outcomes. It may well predict every record to have the *same value* (e.g. no response) and if 85% of the data happens to have that outcome, then the overall accuracy is 85%. Moreover, with certain predicted outcomes, we may be more tolerant of error. If a model is trained to detect a fatal illness it is more likely that the analyst will aim for one that has as few *false negatives* as possible, because in such circumstances, a false negative would mean that the model *fails* to detect the disease when the patient *actually has it*. With such serious consequences we tend to err on the side of caution, so this situation could be regarded as more dangerous than predicting that the patient probably has the disease when in fact they don't (a false positive).

Unfortunately, it is not an uncommon situation for analysts to spend considerable time preparing and transforming data before finally building predictive models only to find that they don't know how useful or valuable the results are. For this reason, the Evaluation phase of CRISP-DM compels us to assess the resultant models *in terms of the success criteria defined during the Business Understanding stage.*

Assessing Model Performance

By following CRISP-DM, we should have documented the model success criteria early in the process. In this section, we can explore some example success criteria that might be applied to the sample datasets we have been working with thus far. For now though, let's consider

what additional pragmatic factors are associated choosing one predictive model over another.

Accuracy

It goes without saying that predictive models have to be sufficiently accurate at predicting an outcome to be regarded as useful. But what me mean by 'sufficiently accurate' depends on the context that the model is to be used in. If we wish to predict an outcome that only occurs 1% of the time, then technically speaking, any model with an accuracy of greater than 1% at predicting that particular outcome is an improvement. For this reason, it is essential that analysts establish a *baseline* against which to judge the model. As we have seen already, there are *costs* associated with *false positives* and *false negative*s and we must be aware of these when assessing the model accuracy. Moreover, within statistics and predictive analytics, there are several specific metrics that can be used to measure accuracy and 'model fit' such as overall accuracy, lift values, area under the curve, R-square etc. So care must be taken to choose a criterion that helps us select a model that is both accurate and useful.

Interpretability

There are many different techniques and algorithms that can be employed to generate a predictive model. Some approaches (especially machine learning algorithms) yield *'black box'* models. These are models that cannot be directly interpreted in the same way that certain statistical or rule-based models can. They are often comprised of large numbers of hidden rules, variable weights or data transformations that the analyst cannot inspect or make sense of. In some fields model accuracy is deemed to be more important than interpretability, so these algorithms are regularly employed. In other disciplines however (such as credit scoring, epidemiology or social research), being able to understand *how* the model generates predictions is of paramount importance. In an ideal world, most analysts would prefer to generate models that are highly accurate *and* easily interpretable.

Stability

Analytical models are based on samples of data collected under certain circumstances and within specific time-frames. We should not be surprised to find that a model fails to generate accurate results when applied to a range of values or circumstances that are very different from the ones it was developed under. For example, model accuracy may decay over time as changes in fashion, demographics, competitor behaviour or market offerings begin to look very different from the time period that the model was developed in. Also, if the model is based on an unrepresentative sample, we can find that it generates inaccurate predictions or even wild estimations when encountering an unfamiliar case. Even certain relatively novel combinations of demographic factors such as ethnicity, gender, age and region can mean that the model is unable to accurately predict the outcome of interest. Stable models however,

are able to generate reliable predictions and estimates with a wide range of data combinations over a useful period of time before they need to be updated or 'refreshed' with new data.

Coherence

As mentioned earlier, many analysts only work with models that can be directly *interpreted*. One of the reasons for doing this to make sure that the model *makes sense.* It is not unusual to discover that a model utilises counter-intuitive rules or non-sensical relationships to estimate a value. Examples include price rises increasing the likelihood to purchase, missing values for variables such as age generating higher estimates of revenue or low satisfaction scores reducing likelihood to churn. In many situations, there may be a sensible explanation for these contradictory relationships. However, more often than not, what is really driving the relationship is a *hidden* variable that explains what's going on. Perhaps price rises increase the likelihood to purchase because they are related to higher demand in the market. Therefore, demand is the driving factor and price is merely a function of it. Missing data for variables such as age may indicate that the person registered for a product or service through a different channel (e.g. *in person* as opposed to via the website) and that in reality the channel is the key predictor. Even lower satisfaction scores could simply indicate that a person *cares* more about a service or has previously complained and subsequently received a discount so lowering their likelihood to defect. Coherent models are valued not only because of the obvious insights they deliver, but because they provide reassurance that the model is not based on a combination of spurious relationships.

Simplicity

Most predictive models are *multivariate* in nature. They are developed from a combination of interrelationships between variables or *model terms.* Many of them can be likened to a house of cards where each layer is precariously added to previous tier. Complexity therefore not only leads to issues with interpretability but also stability. For these reasons alone, simplicity is a key criterion when selecting a predictive model. For example, an analyst might well reject a model with an overall accuracy of 85% but based on 18 variables in favour of one with an accuracy of 82% but only based on 8 variables. This is because including the terms from an additional 10 variables to gain a mere 3% of accuracy may be regarded as poor trade-off.

Performance

A final consideration is the computational performance of the model. Various modelling algorithms utilise resources very differently from one another. Some might not work well with categorical data and require data transformations to perform effectively. Some algorithms

might take a very long time to build a final model or require significant memory allocation and processing power. These requirements can mean that it takes much longer to refine and uncover a final satisfactory model. In a similar vein, when the model is deployed to generate predictions, it may require unacceptable resources in terms of computing power and time to 'score' new cases. In fact, there have been documented examples of predictive models winning predictive analytics competitions by achieving the best score based on a specific accuracy measure only to be deemed unusable in the real world due to their computational resource requirements.

The Analysis Node

The Analysis node is specifically designed to evaluate the performance of predictive models. To see how it works, from the Section 9 folder open the following stream.

Section 9 start.str

Figure 9.1 shows the stream.

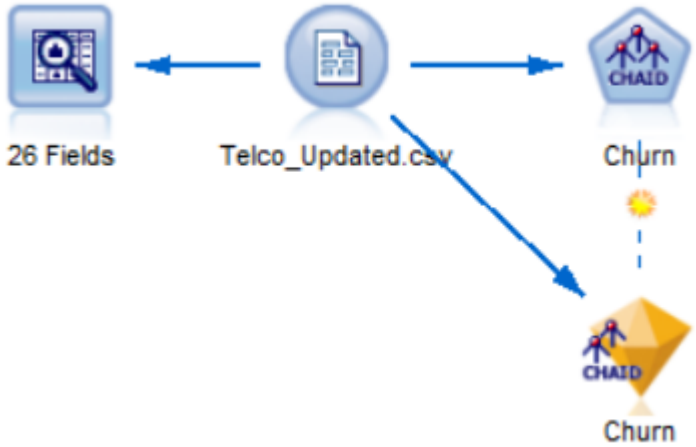


Figure 9.1 The SPSS Modeler stream 'Section 9 Start.str'

The stream contains a basic CHAID model generated in the previous section. To view the overall accuracy of the model, highlight the model nugget and from the Output tab:

Double click the Analysis node

Figure 9.2 shows the Analysis node attached to the CHAID nugget.

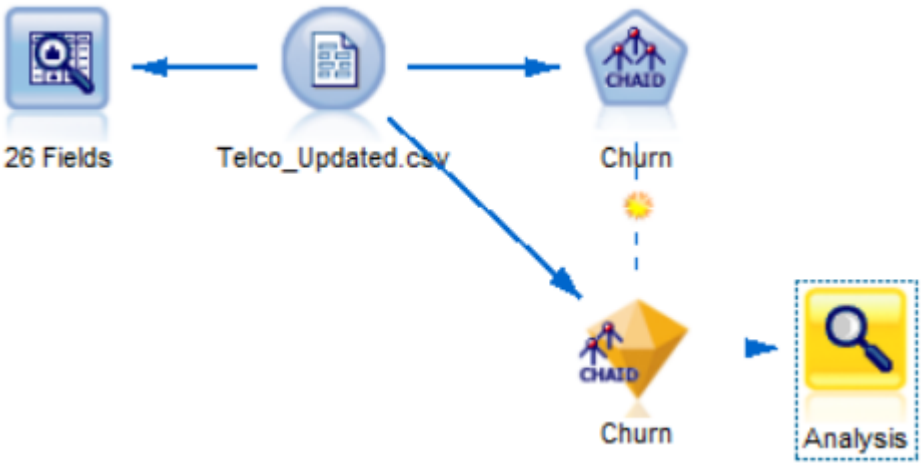


Figure 9.2 The Analysis node attached to the CHAID model nugget

Right-click the Analysis node and select Run

Figure 9.3 shows the Analysis node output.

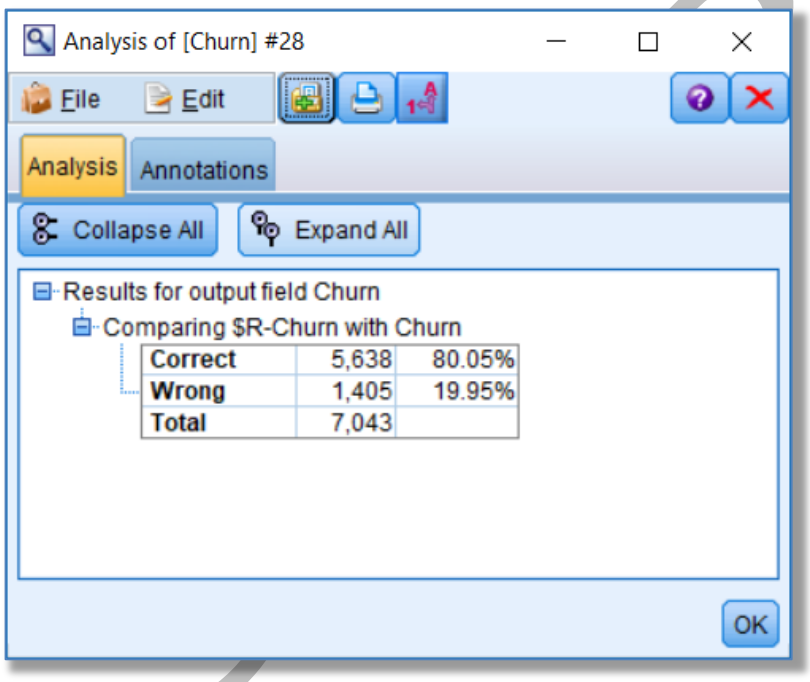


Figure 9.3 The default Analysis node output showing the CHAID model performance

As we can see from the default Analysis node output, the CHAID model has correctly predicted the outcome in 80% of the cases. The node itself simply calculates how often the two values in the target field (Churn) coincide with the predicted outcomes in the $R-Churn field that the model nugget generates. However, we must remember that 73% of the data is comprised of customers who have not churned. This represents our baseline against which to compare the model performance. In which case, if the model itself simply predicted every case to be a current customer, the analysis node would indicate that the model was accurate

73% of the time. To see how well the model does at predicting the individual outcomes ('Yes'/'No') for whether or not the customer has churned, we must re-run the Analysis node and request additional output. To do so:

Click 'OK' to close the existing Analysis node output

Right-click on the attached Analysis node and edit it

As figure 9.4 shows, there are multiple options available for evaluating the model performance. In this case however, we need only request that the node includes a simple crosstab to compare the classification accuracy of the predicted values against the actual outcomes. To do so, check the box marked:

Coincidence matrices (for symbolic targets)

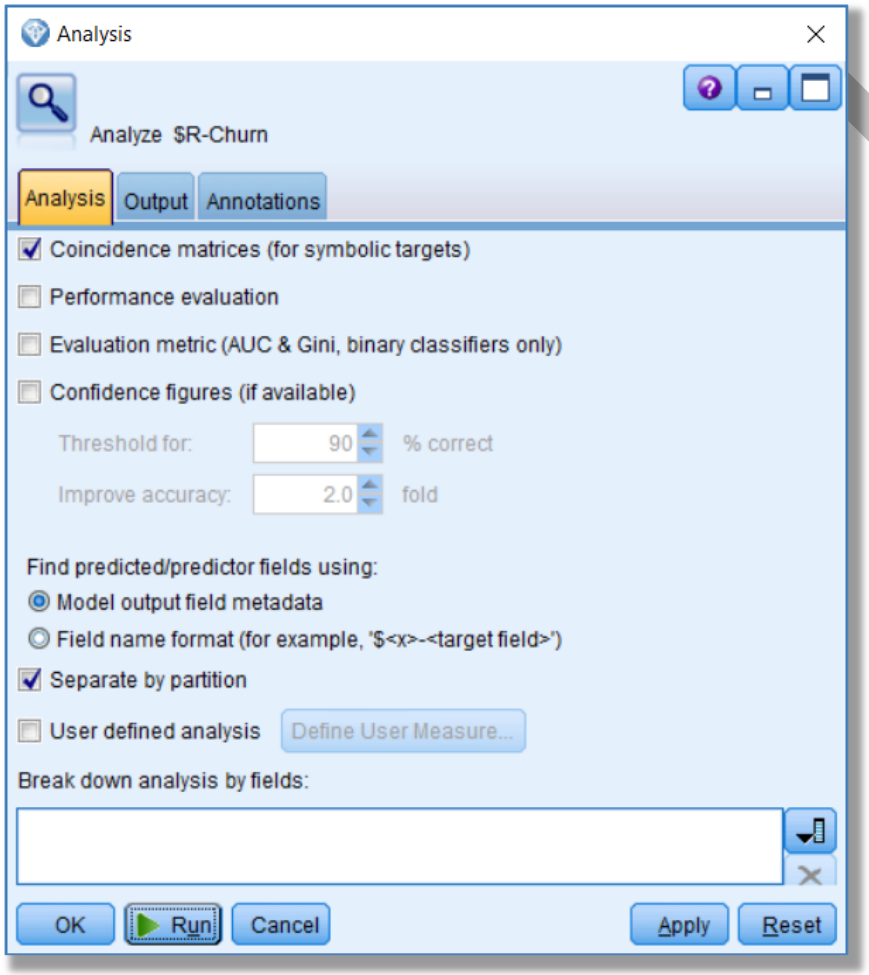


Figure 9.4 Requesting a coincidence matrix (crosstab) to see how well the model predicts the individual outcomes

Now click:

Run

Figure 9.5 shows the updated Analysis node output.

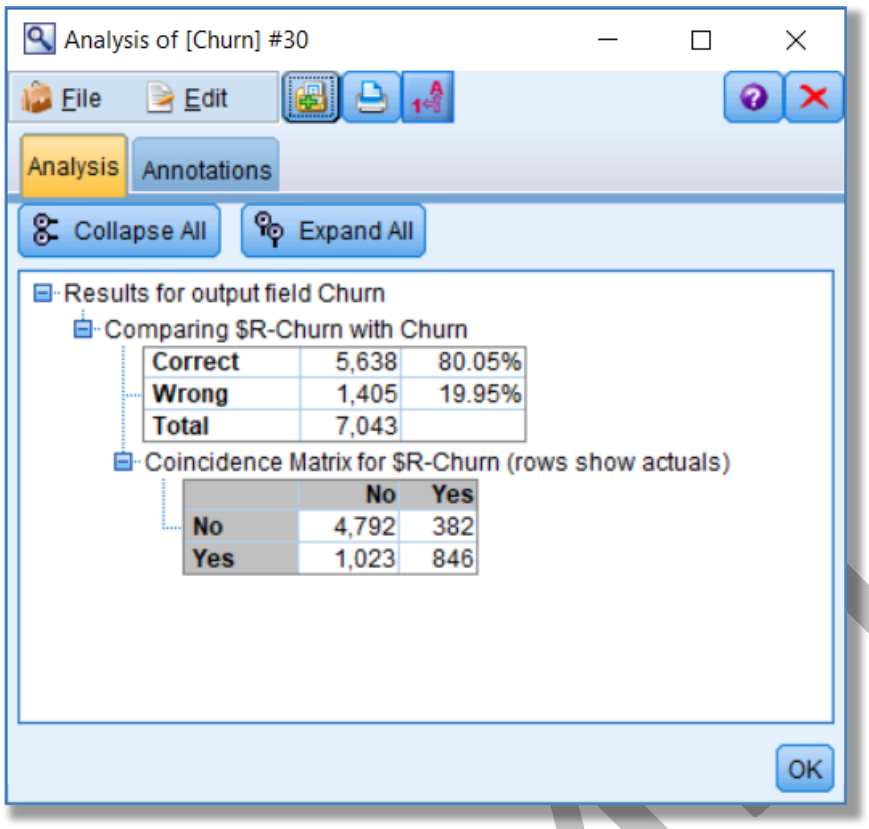


Figure 9.5 The Analysis node output showing a coincidence matrix (crosstab) of the predicted outcomes against the actuals

This time the output from the Analysis node tells us how many of the cases that the model expected to churn had in fact done so (as well as how many of those it predicted to be current customers remained with the telco). The output itself directs us to view the categories in the rows as the *actual* values with the *predicted* outcomes shown in the column dimension. Therefore, we can see that 4,792 people were correctly predicted to remain current customers. However, the model erroneously predicted 382 of the current customers to have churned when in fact they had not (the false positives). Worse still, it predicted 1,023 of the customers who had in fact churned to remain current customers (the false negatives). On the other hand, it correctly identified 846 of the customers who churned. How are the cases being assigned to these predicted groups? Remember that the model generates a probability (or confidence) value for each case. Whichever outcome group ('Yes' or 'No') has the highest probability value, is assigned as the predicted outcome. So even though the baseline probability for a customer churning is around 27%, even if the model finds that a person has a 49% chance of being a churner, they are *still* predicted to be a current customer because the probability that they aren't a churner is slightly higher (51%). At this stage, we need to decide what is more important, accurately predicting current customers at the expense of accurately predicting churners, accurately predicting churners at the expense of accurately predicting current customers or are both groups equally important?

To illustrate this further, from the Section 9 folder open the following stream.

Section 9 Choose best model.str

Figure 9.6 shows the stream.

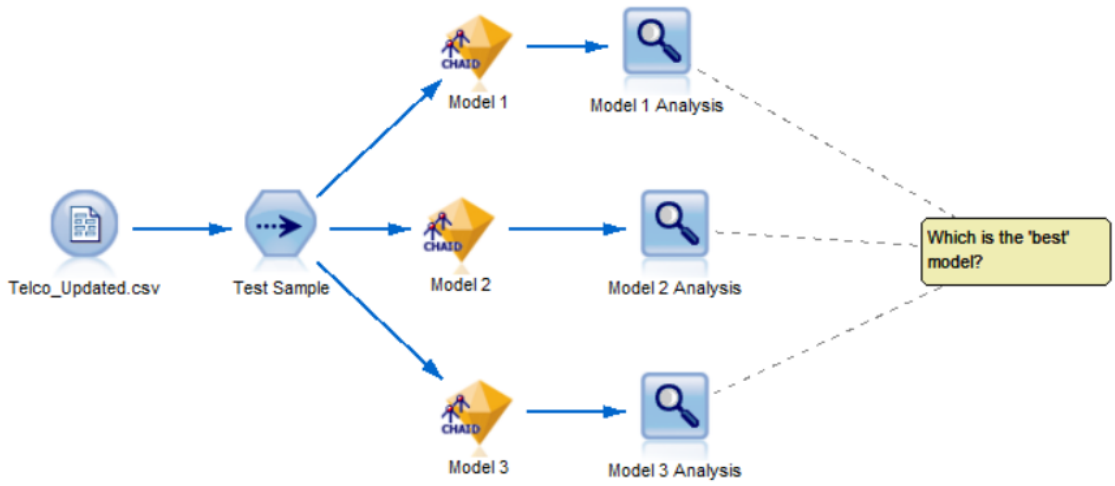


Figure 9.6 The Modeler stream 'Section 9 Choose best model.str'

The stream shows three alternative CHAID models built on the same random sub-sample of data but using different parameter settings and applied back to the same test sample. Figures 9.7a to 9.7c show the output from the three Analysis nodes attached to each respective model.

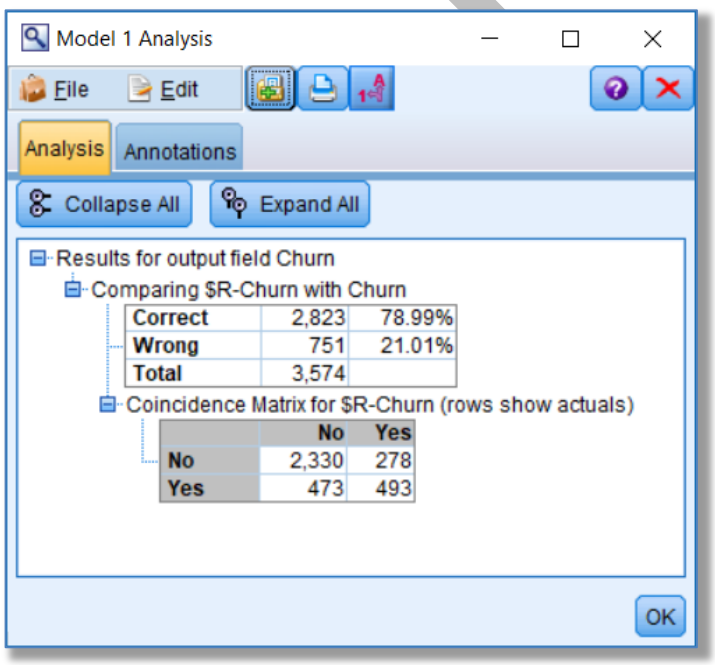


Figure 9.7a Analysis node output with coincidence matrix for Model 1

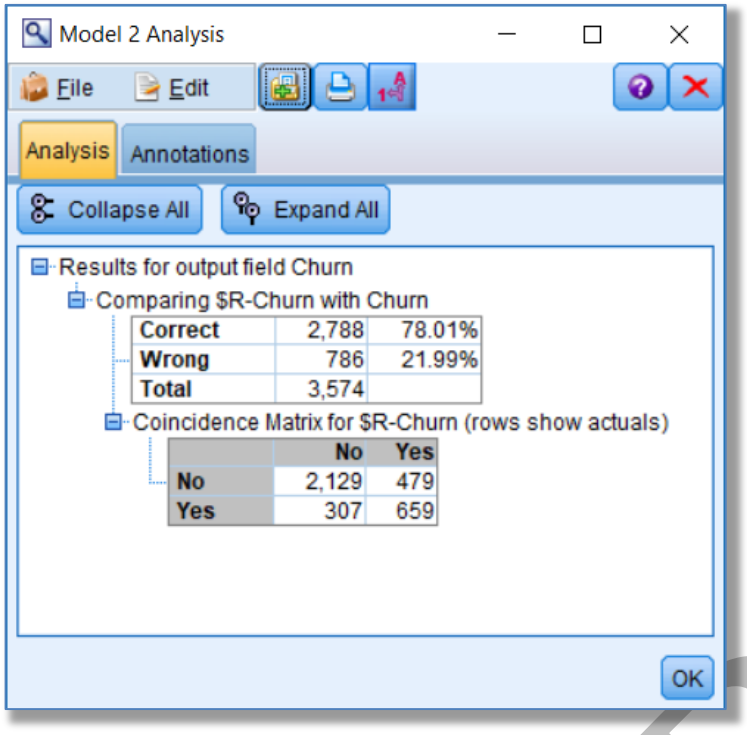


Figure 9.7b Analysis node output with coincidence matrix for Model 2

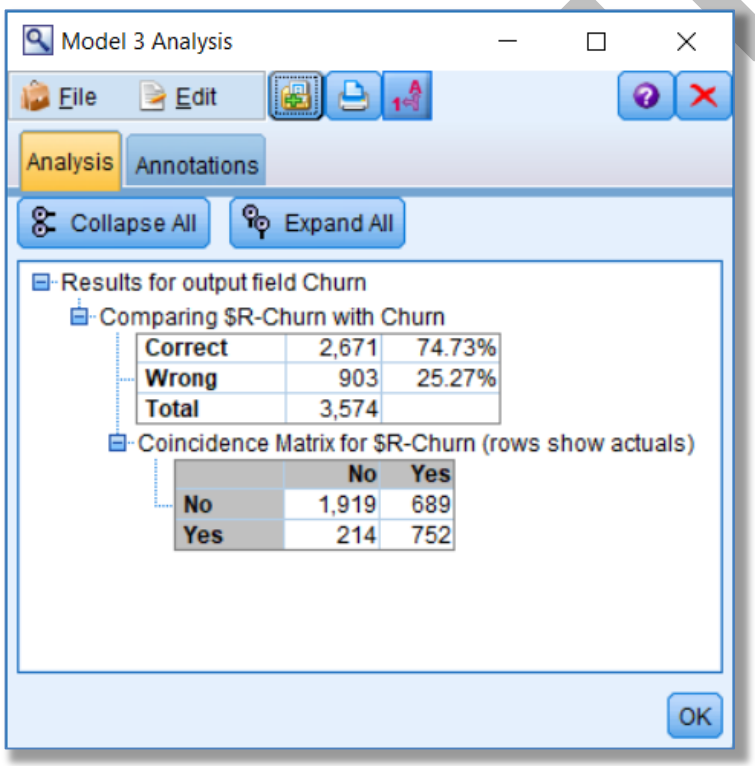


Figure 9.7c Analysis node output with coincidence matrix for Model 3

Although the overall model accuracy varies slightly with each Analysis node output. Looking at each of model's performance evaluations, we can see that a model with increased accuracy in correctly identifying the customers who have churned, tends to have a *decreased* accuracy in correctly identifying the current customers and vice-versa. The question as to which is the

best model relates back to our Business Understanding and the costs associated with each outcome. What are the costs associated with erroneously predicting someone to have a high risk of defecting compared to the cost of mistaking someone who in reality is likely to cancel their contract, for one who will remain a customer? Before we consider some example success criteria, let's take a look at how we can use the Analysis node to compare the performance of multiple models. From the Section 9 folder, open the following stream:

Section 9 Two Model Comparison.str

Figure 9.8 shows the stream.

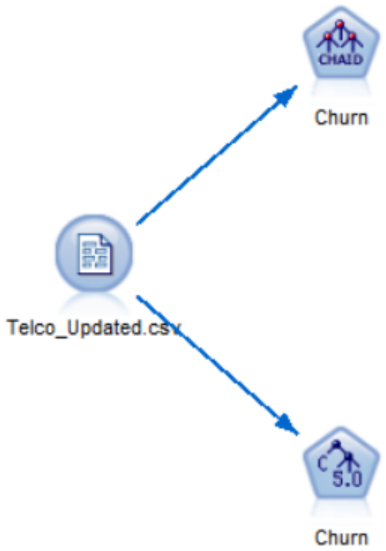


Figure 9.8 The Modeler stream 'Section 9 Two Model Comparison.str'

The stream shows two decision tree modelling nodes that have been configured to predict the same target field. To generate both models, from the main toolbar click the stream run button:

Both model nuggets are generated. To compare their respective performance with the same Analysis node, connect the two models in the same stream branch as shown in figure 9.9.

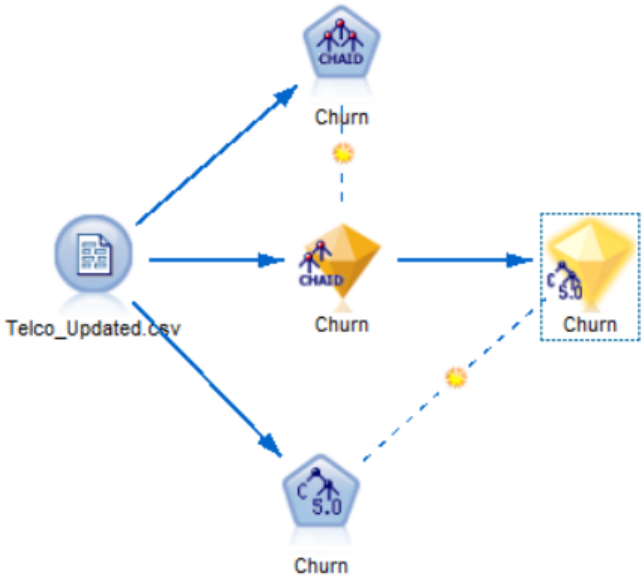


Figure 9.9 Connecting two model nuggets in the same stream branch

Having connected the two model nuggets, we can attach the analysis node to compare their performance. From the Output palette,

Attach an Analysis node to the last model nugget

Right-click on the analysis node and request 'Coincidence matrices (for symbolic targets)'

Figure 9.10 shows the completed stream.

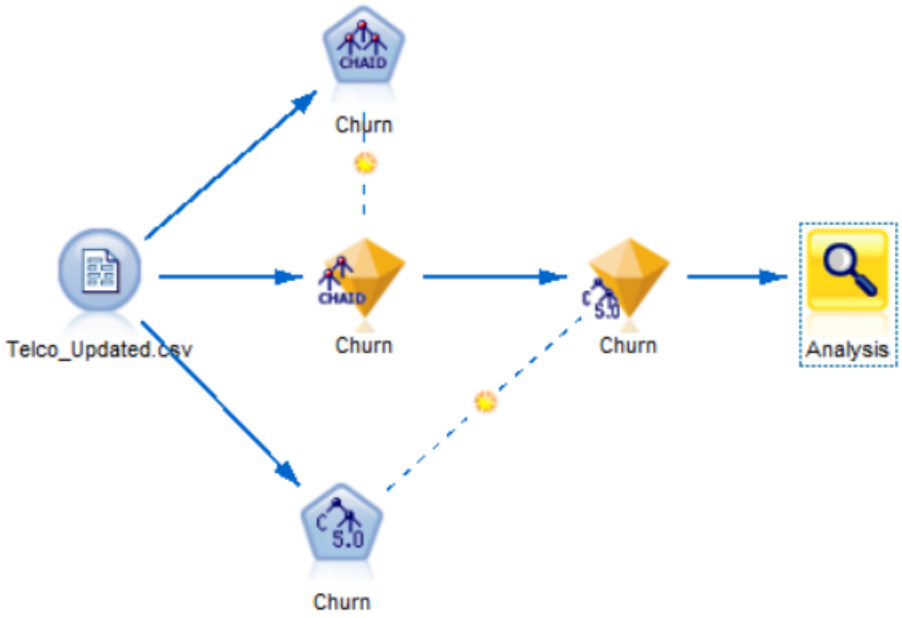


Figure 9.10 Analysis node attached to stream branch containing two model nuggets

Now:

Right-click and on the Analysis node and select run

Figure 9.11 shows the Analysis node output.

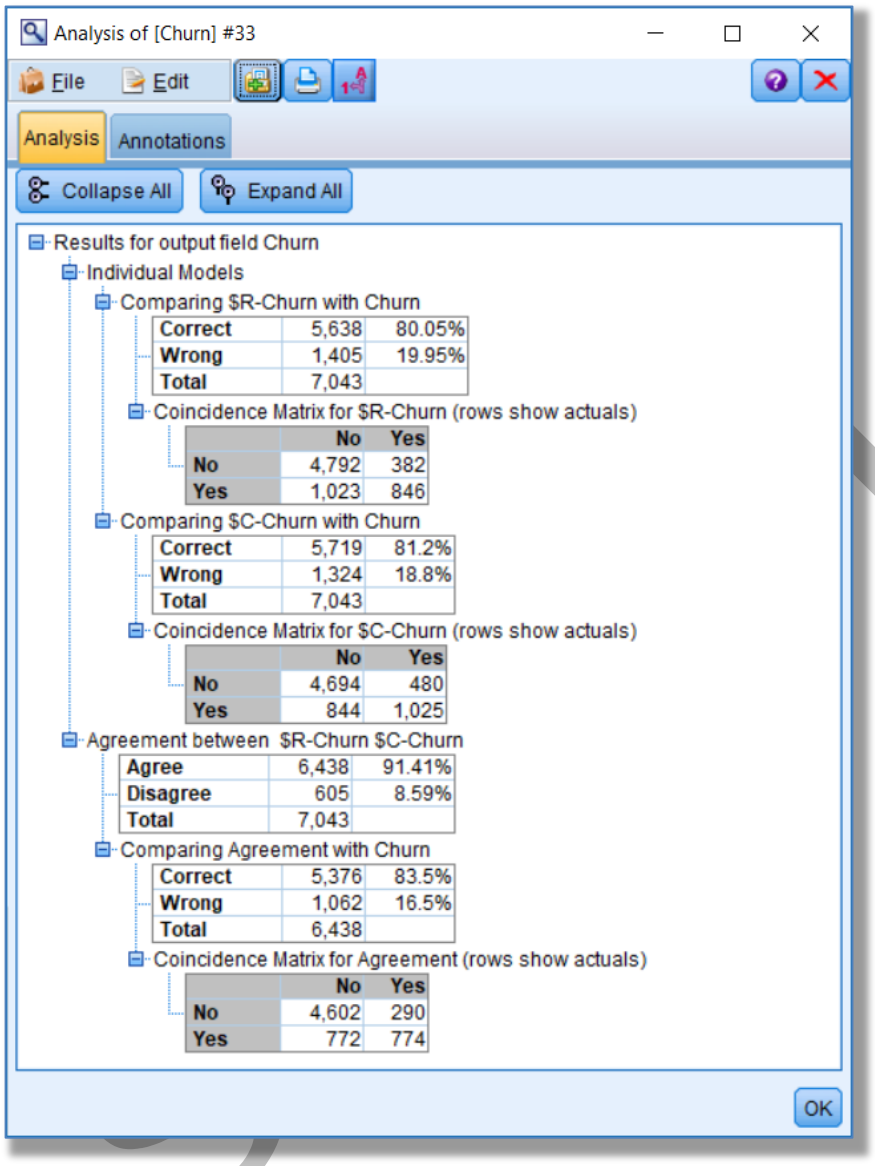


Figure 9.11 Analysis node showing performance comparison for two predictive models

The first part of the output shows the classification accuracy for the CHAID model (as denoted by the $R-Churn variable). The second section shows the classification accuracy for the C5 model (as denoted by the $C-Churn variable). The overall accuracy for the C5 model is slightly higher than the CHAID model (81.2% vs 80.05% respectively). Note that the C5 model is better at accurately predicting customers who churned compared to the CHAID model but worse at predicting the current customers. The third section of output looks at the degree of agreement between the two models: showing that both models gave the same predictions in 91.41% of cases. This represents 6,438 out of a total 7,043 records. Lastly, the output shows

that in the 6,438 records where both models agreed, the predictive accuracy was 83.5%. The final table shows a crosstab (or coincidence matrix) of these predictions against the actual outcomes.

Success Criteria Scenarios

So what success criteria might the decision makers within the telco organisation have established to choose a final model? In all of the following examples, the organisation's aims rest on its ability to proactively identify customers with a high risk of churning.

Scenario 1:

- ***Currently around 100K customers cancel their contracts each month.***
- ***Previous tests have shown that sending offers to a random group of 100K current customers approaching contract end dates each month, reduces the churn amount by 7K customers (7%).***
- ***We would like to send 50K offers to customers with a high risk of churning with view to reducing churn by 14K (14%).***
- ***The model should therefore identify the 50K current customers approaching their contract end date who have the highest likelihood of churning with a view to retaining at least 14K of them.***

Scenario 2:

- ***Currently we have a monthly outbound email campaign that targets 400K customers approaching their contract end.***
- ***Tests indicate that this retains around 15K customers who would otherwise have churned.***
- ***However, the total cost of the offers is very expensive as we suspect many customers with a low likelihood of churning also redeem them.***
- ***We would like to reduce the outbound mailing to 100K customers and still retain the 15K customers who would otherwise have churned.***

Scenario 3:

- ***Currently we have a monthly outbound email campaign that targets 60K key customers approaching their contract end.***

- ***The campaign offers the next month's standard call costs for free if they extend their contract by two months.***

- ***However, the campaign makes a net loss of $130K each month as many customers churn after the additional two months anyway.***

- ***We would like to make a net profit of $70K per month by targeting only those with a low likelihood of churning after accepting the offer.***

The first scenario directs the analyst towards finding the 50K customers with the highest likelihood of churning. The success criterion here is that of these 50K customers, the subsequent offer should enable the company to retain at least 14K customers. We should bear in mind the models we have built thus far have been focused on predicting which customers will churn, not which of those customers will respond to the retention offer. However, it's reasonable to assume that if an offer to a random selection of 100K customers succeeds in retaining 7% of churners, then the same offer to a highly targeted group should do much better.

In the second scenario, the aim is to reduce the number of offers being made whilst still retaining 15K customers. This is more about reducing the cost of the campaign as measured by the number of customers contacted with an offer (400K). Here the criterion is that the model should be four times better than a random approach as the organisation only wants to send the offer to 100K people without losing any additional customers.

The third scenario directly focuses on the profitability of the campaign. Specifically, the profits associated with offering free services to customers who do decide to churn anyway. This of course requires a data sample where high risk customers have already been identified (perhaps through an earlier churn model) and have then been made a retention offer. The aim here is to only make the retention offer to those customers that are likely to remain long enough with the organisation that the offer costs can be recouped, and a net profit realised. As we shall see in the next section, it's possible to make this selection by entering a few estimated costs and revenue values to identify the selection of customers where the business has the best chance of maximising the profit from a campaign.

The Evaluation Node

We have already seen how the analysis node will allow us to assess the accuracy of a classification model using crosstabs (or coincidence matrices). The Evaluation node is stored in the Modeler Graphs palette however and therefore allows us to visualise the model performance via a portfolio of charts. To see an example of this we can return to the currently open stream 'Section 9 start.str' as shown in figure 9.12.

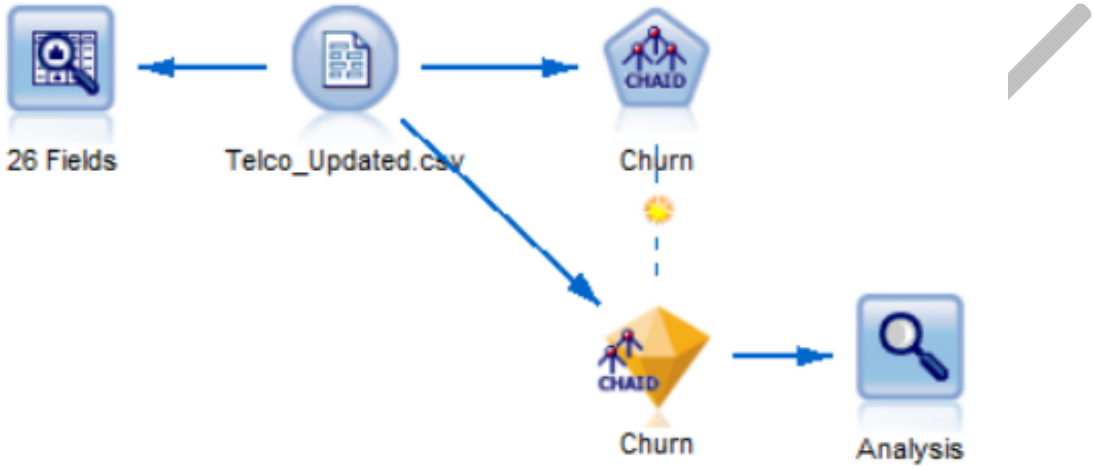


Figure 9.12 The currently open Modeler stream 'Section 9 start.str'

To attach an Evaluation node:

Click on the CHAID nugget in the stream to select it

From the Graphs palette within SPSS modeler:

Double click the Evaluation node

The Evaluation node will automatically attach itself to the model nugget. Figure 9.13 shows the updated stream.

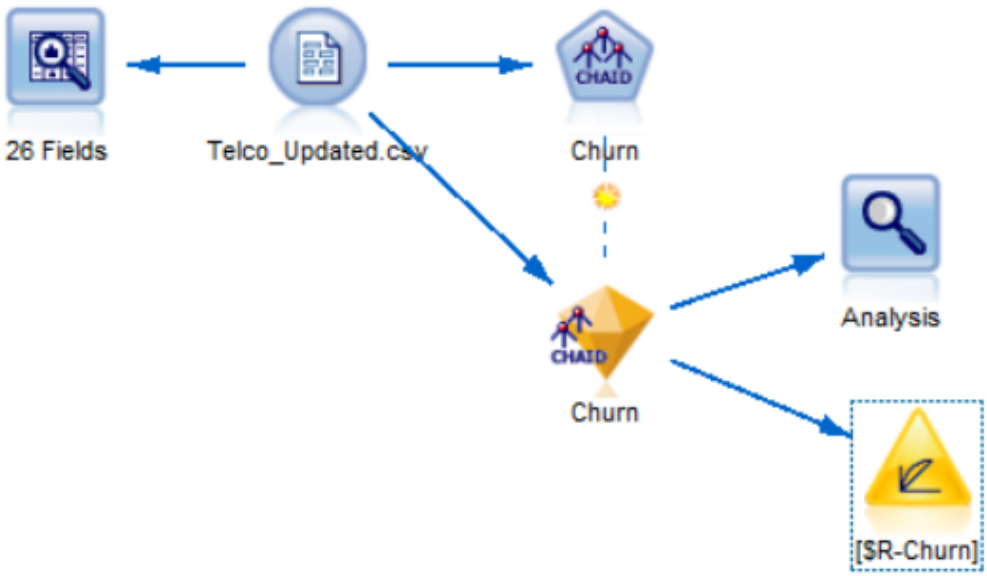


Figure 9.13 Evaluation node added to the current stream

Before running the node itself, it's useful to make one alteration to the default settings.

Right-click on the Evaluation node and edit it

Within the control dialog for the Evaluation node, check the box marked:

Include Best Line

You may notice at this stage that the default chart type in the Evaluation node is set to 'Gains'. Gains charts are a common way to show how well a classification model performs. Figure 9.14 shows the edited Evaluation node.

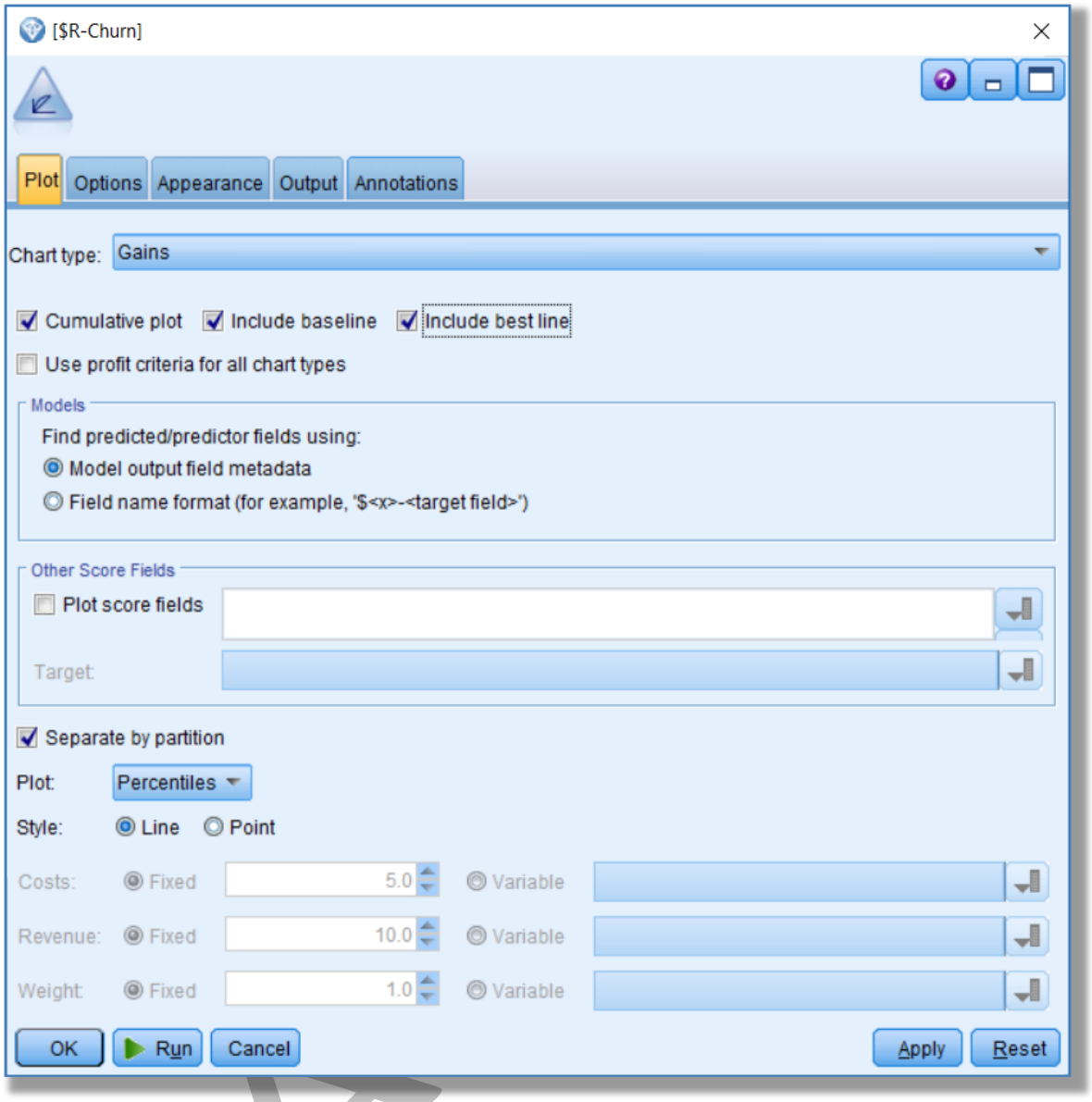


Figure 9.14 The edited Evaluation node with the 'Include Best Line' option selected

To execute the Evaluation node and view the Gains chart, click:

Run

Figure 9.15 shows the resultant Gains chart.

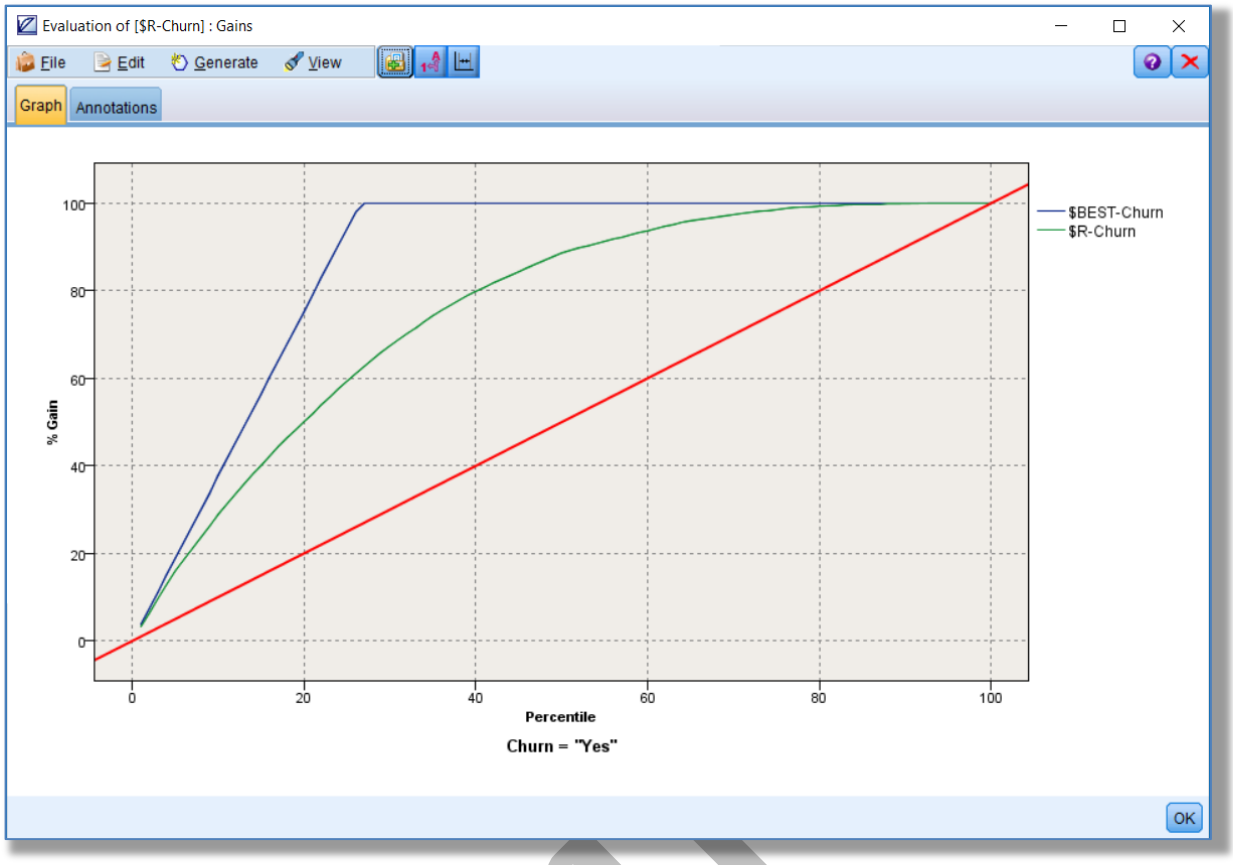


Figure 9.15 The Gains chart as generated by the Evaluation node with the 'Best Line' option displayed

The purpose of the Gains chart is to show the proportion of records in a target group that the we can 'gain' by using the model. Perhaps the easiest way to make sense of it is to note that the chart itself only can only focus on one group within the target field at a time. Here the default group of interest is those customers who have churned (as evidenced by the label on the horizontal axis indicating that 'Churn= "Yes"'). The diagonal red line within the chart simply tells us what proportion of records within this group we might expect to find if we were to randomly sample the data. The line therefore simply indicates that for example, using a random approach, we could only expect to find (or 'gain') 20% of the customers in the Churn group from 20% of the data, or indeed 50% of the churners by sampling 50% of the file. The top line (or 'Best line') however, represents what a perfect predictive model would like. Here we could gain 100% of the churners from 27% of the data. This is simply because within the sample dataset, 27% of the records belong to those customers who have churned. Having established what a random model and a perfect model would look like, the middle line shows us how many customers we might expect to find using the predictive model itself. Or alternatively, how much better the model is than random and how much worse it is than perfect. By moving the cursor along any of these lines, a pop-up label appears telling us what proportion of churners (the 'gain') we might detect as we increase our sample size. In this case, the model indicates that we could expect to find about 63% of the people who churned from 27% of the data. This is extremely useful, because if we wished to reduce the number of customers that we intend to contact to reduce churn, the gains chart will tell us what

proportion of the churners the model could detect. In this this case, by using the chart, we could select the 50% of customers with the highest estimated risk and still expect to capture 88% of the customers who churn (see figure 9.16).

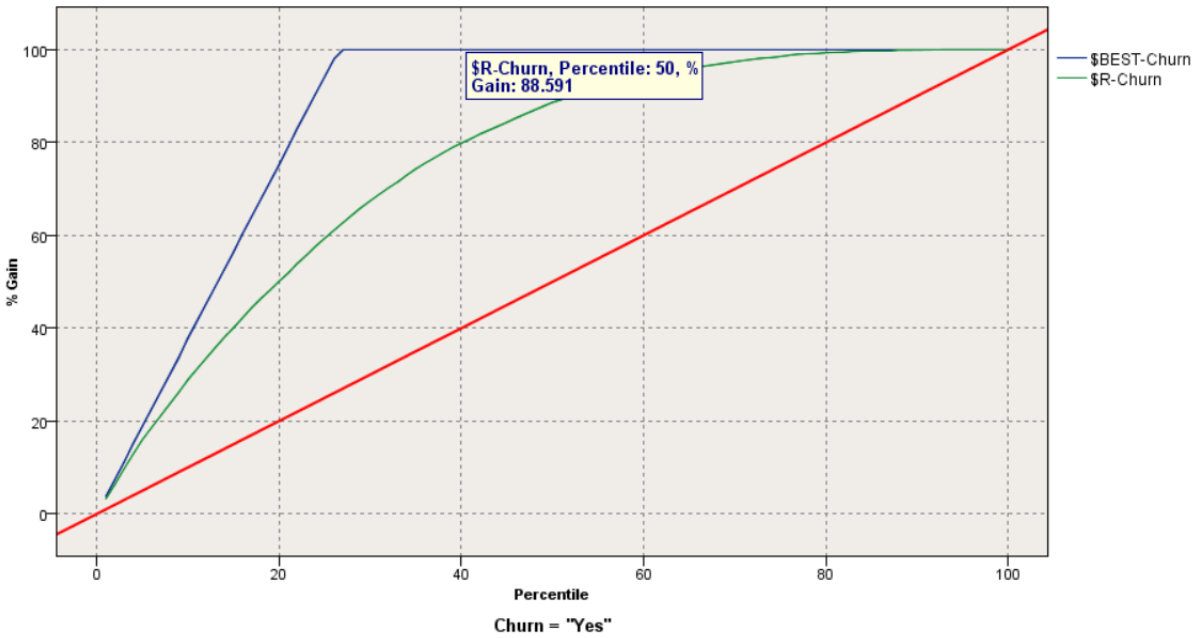


Figure 9.16 Using the predictive model to select the highest risk 50% in the sample we would expect to detect around 88% of the customers who churn

We can also use the Evaluation node to compare multiple models. To show this, within the currently open stream, 'Section 9 start.str':

Right-click on the Evaluation node and select 'Copy Node' from the pop-up menu

Return to the previously opened stream 'Section 9 Two Model Comparison' and paste the Evaluation node into the stream

Attach the pasted Evaluation node to the second model nugget

Figure 9.17 shows the updated stream.

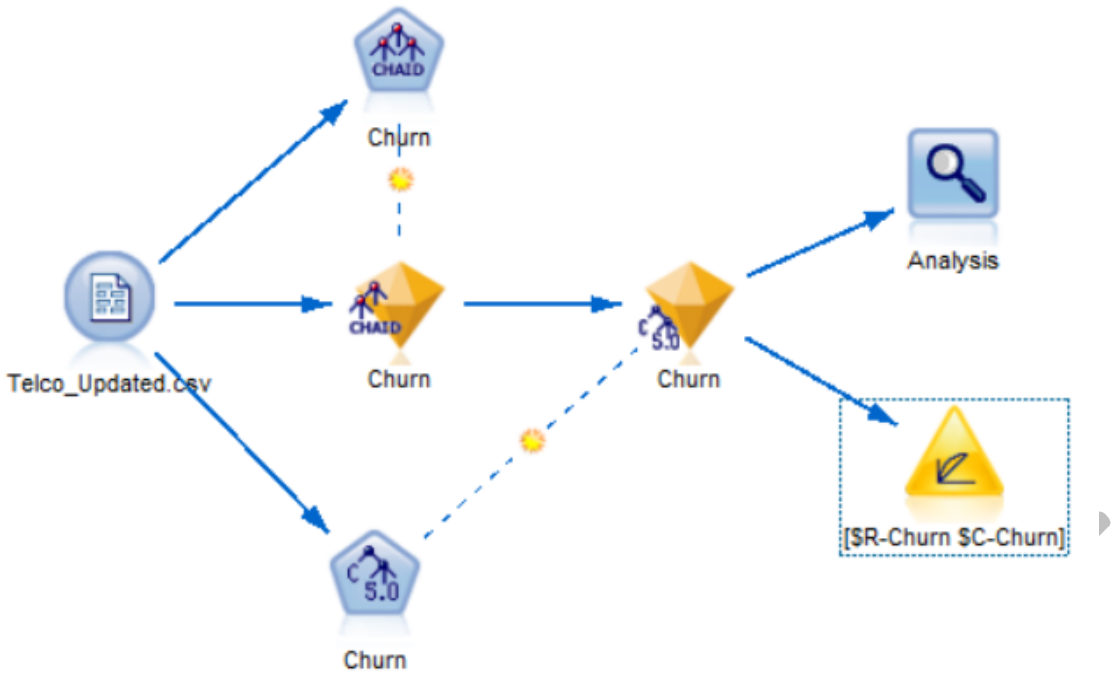


Figure 9.17 Attaching an Evaluation node downstream of two model nuggets

Right-click and run the node

Figure 9.18 shows the resultant chart

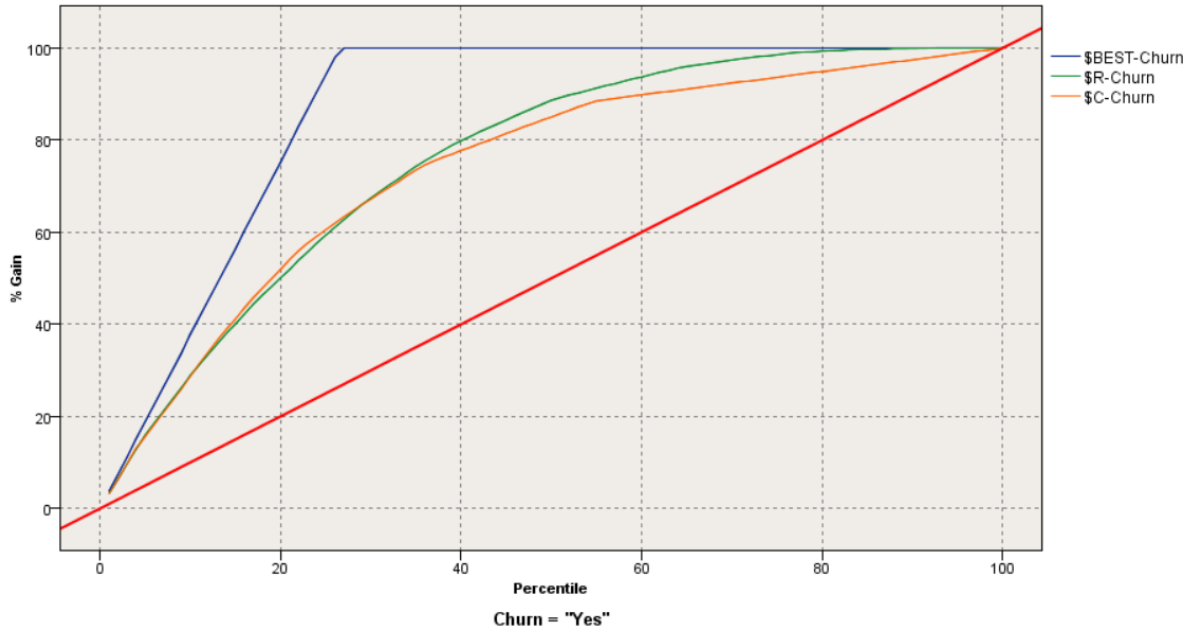


Figure 9.18 Gains chart generated by an Evaluation node displaying the performance of two classification models simultaneously

We can see from the Gains chart that the two models exhibit subtle differences in their classification accuracy. If the goal of the analyst was to select the top 20% of the file

containing customers with the highest risk of churning, the C5 model ($C-Churn) has a slightly better performance than the CHAID model ($R-Churn). If however the analyst wanted to select the 40% of cases with the highest risk, the CHAID model seems to be slightly better than the C5.

To show how this can be done, within the chart itself, from the main menu, click:

View

Interactions

Figure 9.19 illustrates this process.

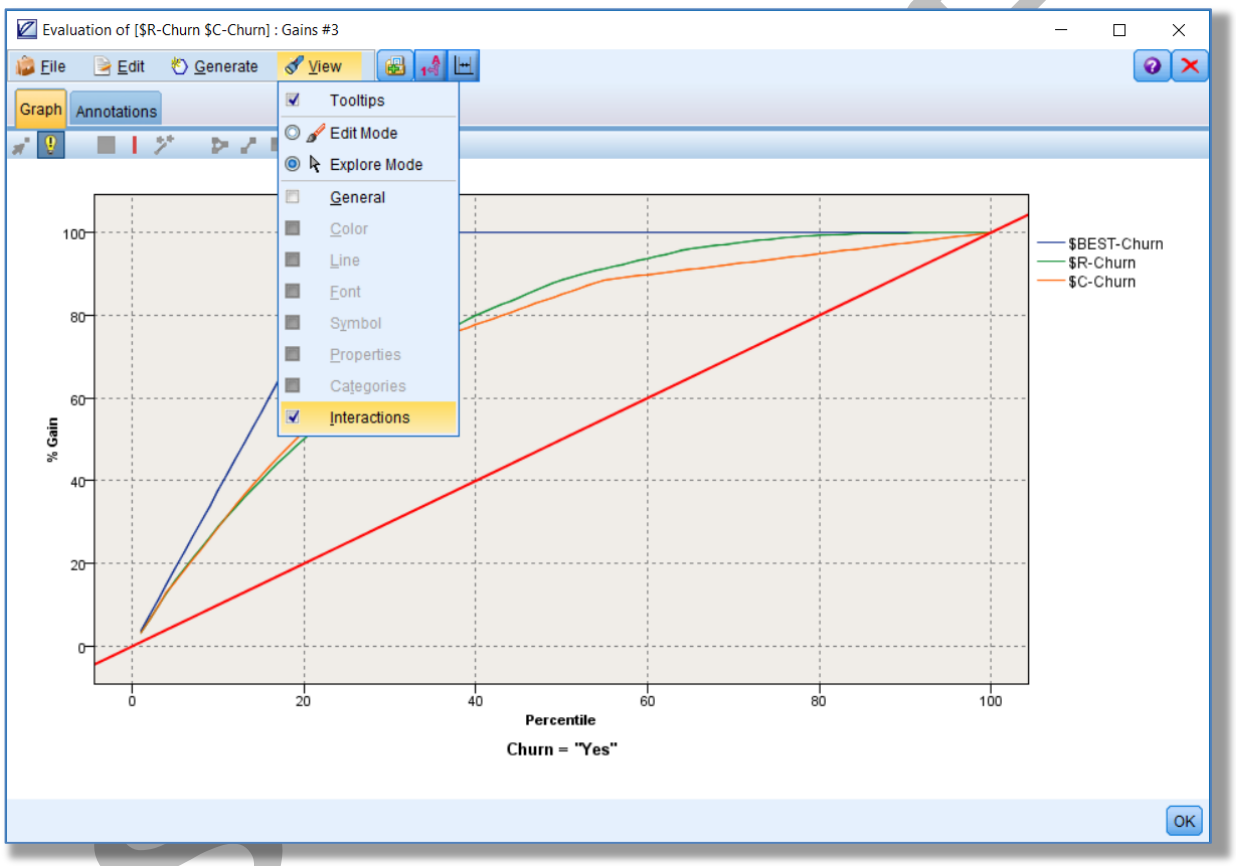


Figure 9.19 Switching on Interactive mode in a Gains chart.

On the chart toolbar click the band selection tool:

Using the tool:

Move the cursor to around the 40th percentile on the horizonal axis and click

Figure 9.20 illustrates this.

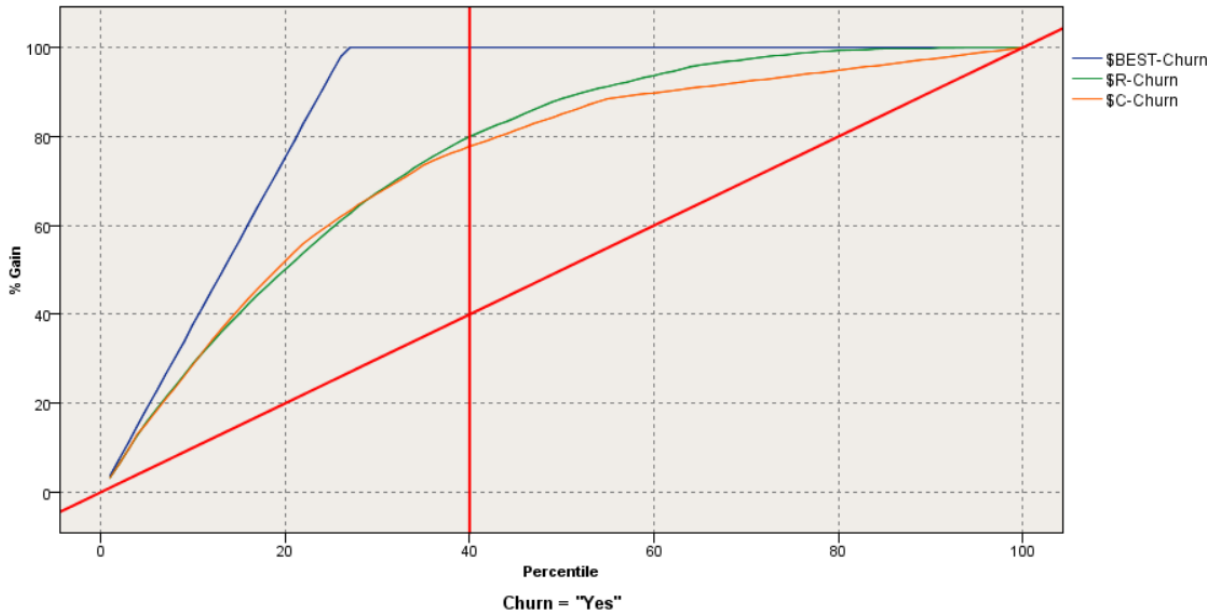


Figure 9.20 Using the Band Selection tool to interact with the Gains chart

Now:

Right-click on the area to the left of the red selection line

From the pop-up menu choose:

Generate Select Node for Band

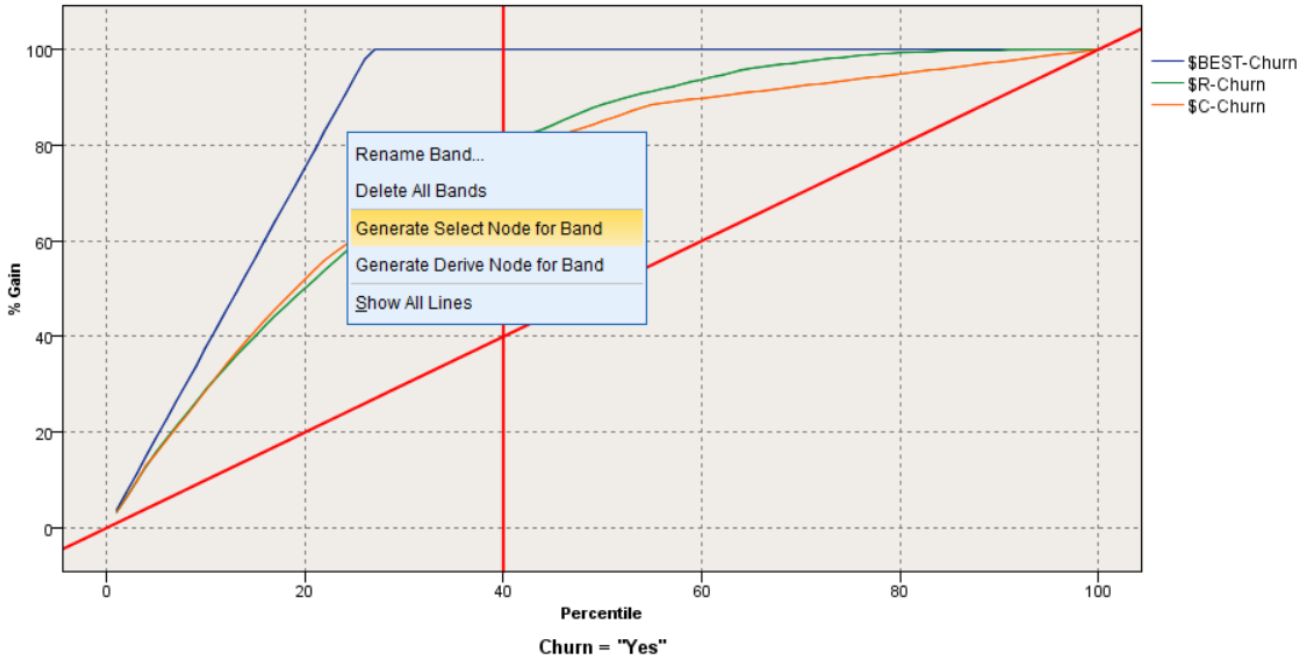


Figure 9.21 Generating a Select node for the highest risk 40% of data

A pop-up menu appears asking you to choose which model the selection should refer to (see figure 9.22).

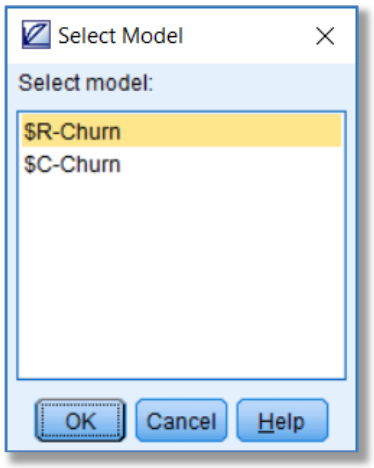


Figure 9.22 Select model pop-up menu

From the pop-up menu choose the variable containing the CHAID model's predictions:

$R-Churn

OK

A new Select node (labelled 'Band 1') is added to the stream. This node will use the CHAID model scores to select the 40% of records that contain 80% of the customers who churned in the sample data. We can attach the node downstream of the CHAID nugget as shown in figure 9.23 and display the records in a Table node as shown in figure 9.24.

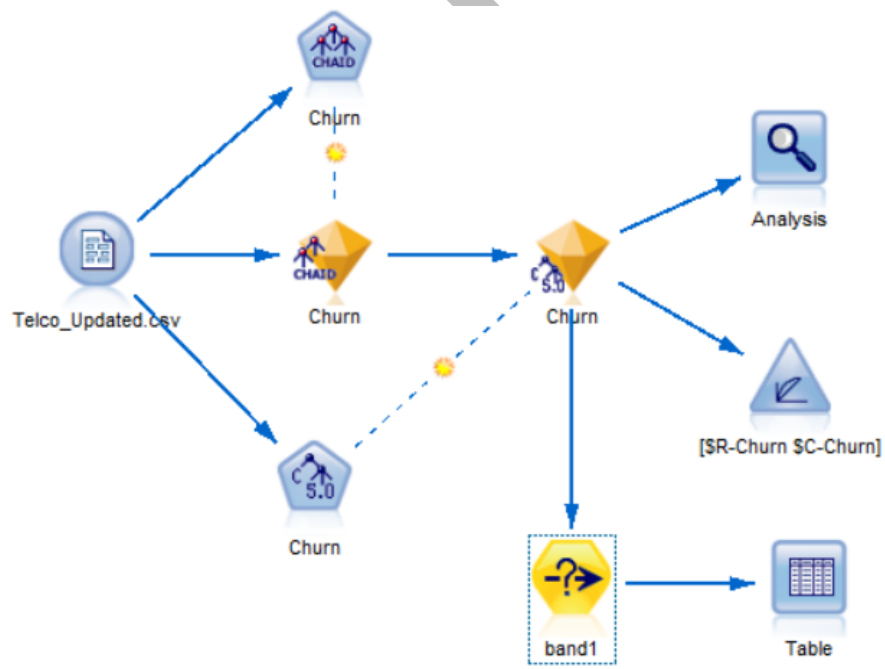
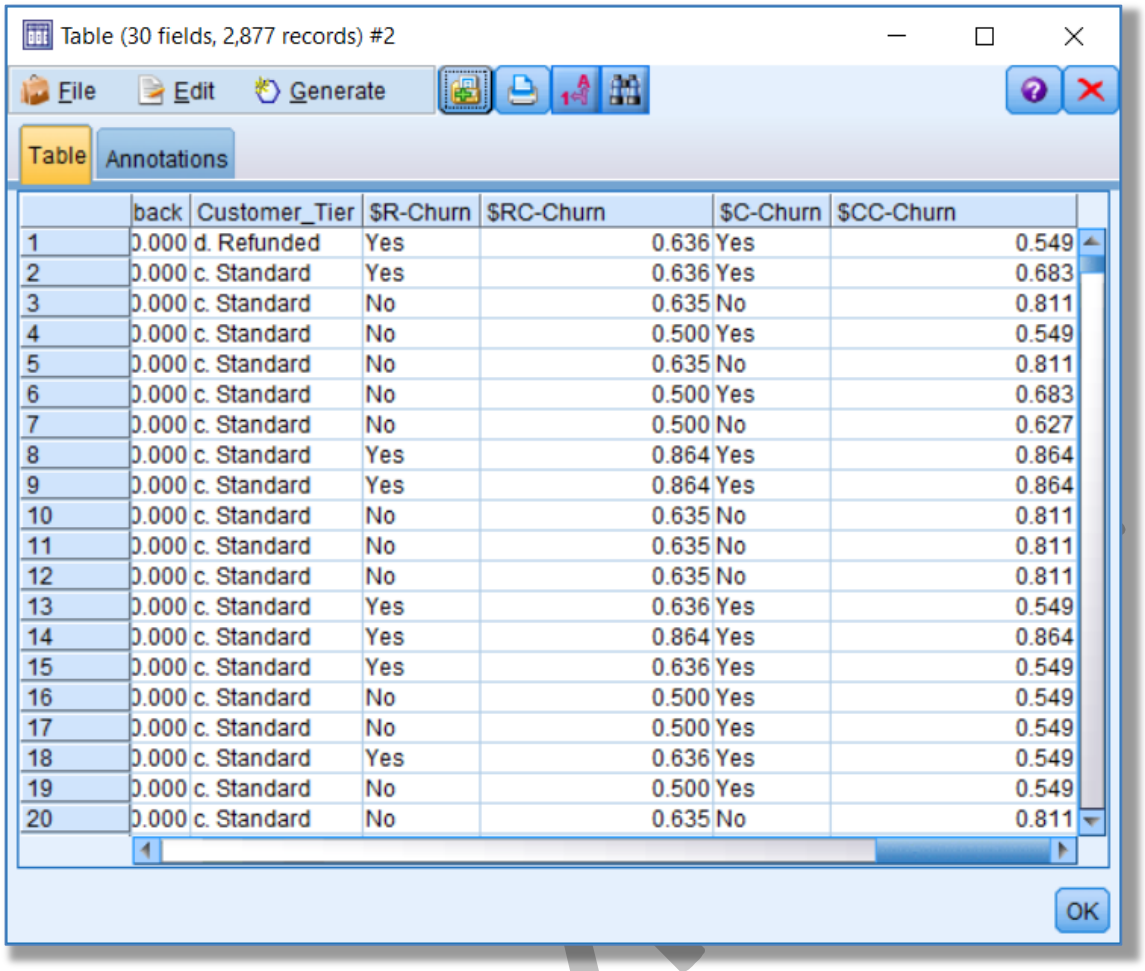


Figure 9.23 The generated Select node that selects the 40% of customers containing 80% of the churners



	back	Customer_Tier	$R-Churn	$RC-Churn	$C-Churn	$CC-Churn
1	0.000	d. Refunded	Yes	0.636	Yes	0.549
2	0.000	c. Standard	Yes	0.636	Yes	0.683
3	0.000	c. Standard	No	0.635	No	0.811
4	0.000	c. Standard	No	0.500	Yes	0.549
5	0.000	c. Standard	No	0.635	No	0.811
6	0.000	c. Standard	No	0.500	Yes	0.683
7	0.000	c. Standard	No	0.500	No	0.627
8	0.000	c. Standard	Yes	0.864	Yes	0.864
9	0.000	c. Standard	Yes	0.864	Yes	0.864
10	0.000	c. Standard	No	0.635	No	0.811
11	0.000	c. Standard	No	0.635	No	0.811
12	0.000	c. Standard	No	0.635	No	0.811
13	0.000	c. Standard	Yes	0.636	Yes	0.549
14	0.000	c. Standard	Yes	0.864	Yes	0.864
15	0.000	c. Standard	Yes	0.636	Yes	0.549
16	0.000	c. Standard	No	0.500	Yes	0.549
17	0.000	c. Standard	No	0.500	Yes	0.549
18	0.000	c. Standard	Yes	0.636	Yes	0.549
19	0.000	c. Standard	No	0.500	Yes	0.549
20	0.000	c. Standard	No	0.635	No	0.811

Figure 9.24 Table containing the 40% of customers comprising the 80% of churners

The Partition Node

A fundamental problem with the creation of predictive models is the uncertainty as to how the model will perform when applied in the real world. There are number of techniques that analysts employ to simulate how a model might behave when applied to new data and one of the most popular is the use of training and test samples. Training samples are simply extracts of data (usually randomly chosen) that are used to develop the model. As we have already seen, Modeler's predictive algorithms are able to automatically build models using default settings against representative historical data where the outcome of interest is known. Depending on the nature of the algorithm, this model 'training' process can be achieved using statistical techniques or machine learning methods to select, transform and incorporate variables into a final model. The model itself might be expressed as a mathematical formula, a set of rules or a composite of hidden transformations, but the overall goal is to predict the outcome with the highest degree of accuracy. It's essential therefore that the sample training dataset is sufficiently large and unbiased so that the resultant model can then be applied in a

real-world context to generate accurate estimates. Nevertheless, every data sample is unique in its own way, and as such, there is always a danger that the training process results in a model which is overly specific to the idiosyncrasies of the training data. In analytics, this is known as 'overfitting'. An overfitted model will tend to perform poorly when applied to a different sample of data from the one it was trained on. To get around this, many researchers retain a sub-sample of the main dataset for testing purposes. If the analyst can test the model to see if it performs well on both the training and the test samples, then they will have greater confidence that it will perform well when deployed against data where the outcome to be predicted is not *yet* known. The Partition node within SPSS Modeler allows us to do just this.

To see how we can use the partition node to compare model performance on separate training and test samples:

Return to the stream 'Section 9 start.str'

Delete the existing CHAID model nugget from the stream

From the Field Ops palette:

Select and add the Partition node to the stream

Drag the connection between the Data Source node and the CHAID node so that passes through the Partition node

Figure 9.25 shows the edited stream at this stage.

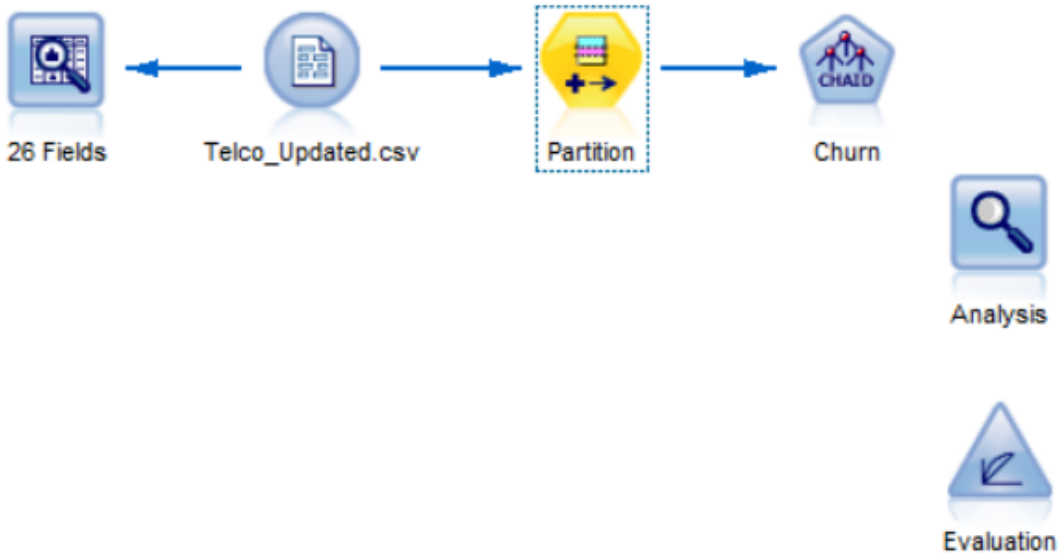


Figure 9.25 Adding a Partition node to the existing stream file 'Section 9 start.str'

Before running the CHAID node, we can take a look at how the Partition node is configured. To do so:

Right-click on the Partition node and edit it

Figure 9.26 shows the edited Partition node.

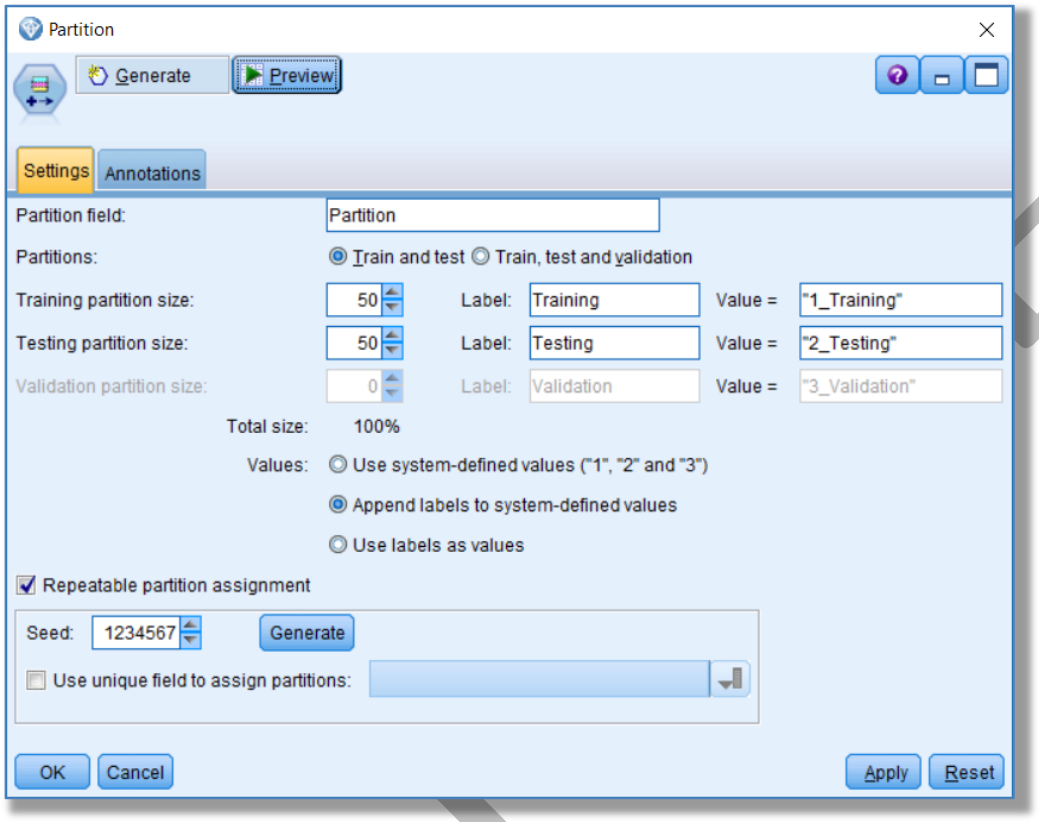


Figure 9.26 The Partition node dialog showing its default settings

By editing the Partition node, we are able to define what proportion of the sample data will be assigned as the 'Training' group and the 'Testing' group. Depending on the size of the main dataset, analysts often assign a smaller proportion for testing purposes than the training set. Note that you can also define a third group here: a Validation sample. Some analysts like to create this additional random partition when they are trying to select the best performing model from a number of candidates and they suspect that a particular model might be performing well on the Testing sample as the result of chance. You can also see that the node allows the analyst to define or generate a random seed number. Using the same seed number allows us compare one model to the next as the same cases will be randomly assigned to the training and testing samples. To illustrate how we might use the Partition node:

Change the Training partition size to 70

Change the Testing partition size to 30

Change the Random Seed number to 7654321

Figure 9.27 shows the edited Partition node control dialog.

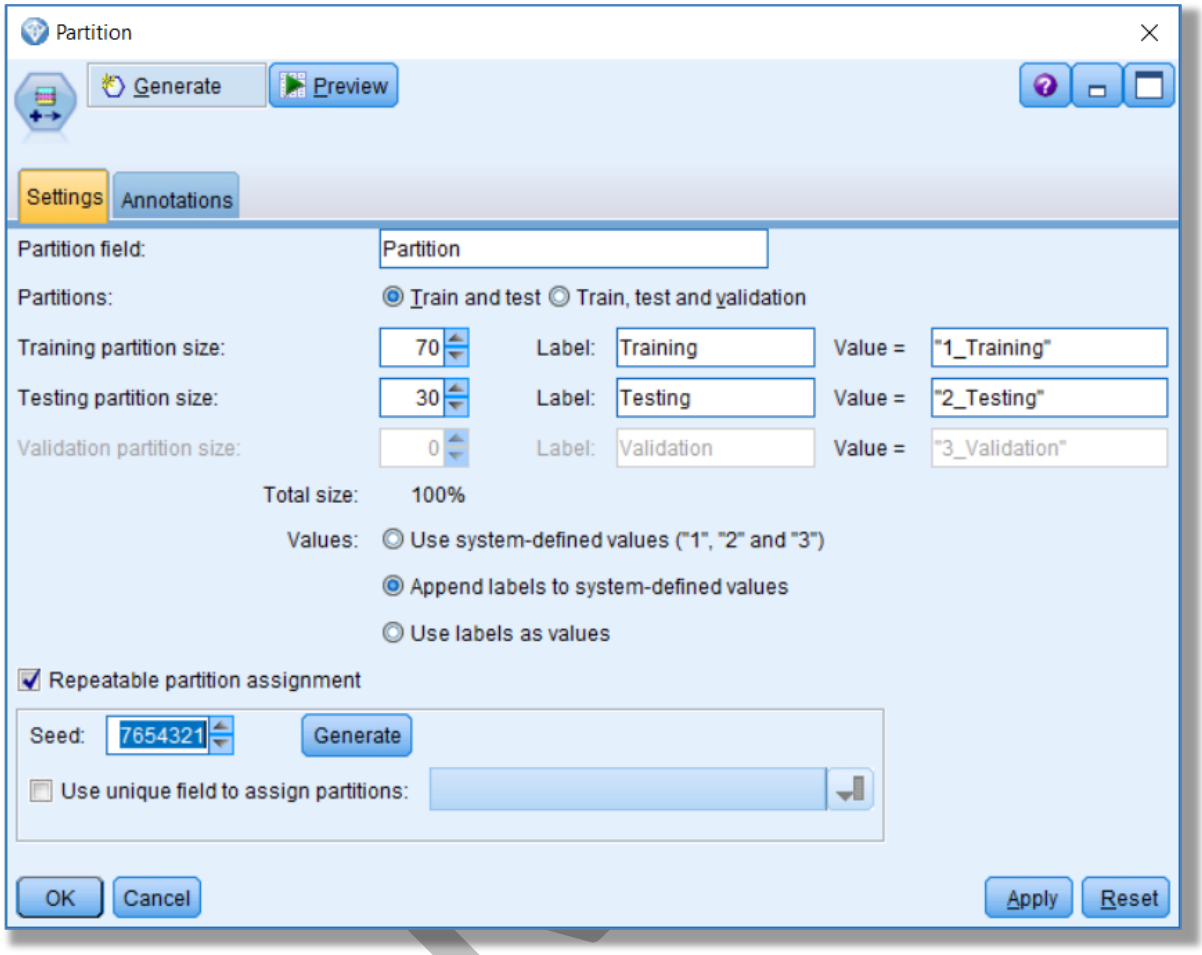


Figure 9.27 The edited Partition node dialog

Click OK and run the CHAID node

The CHAID node once again generates a CHAID model nugget.

Attach the CHAID model nugget to the Analysis node and the Evaluation node

Figure 9.28 shows the updated stream.

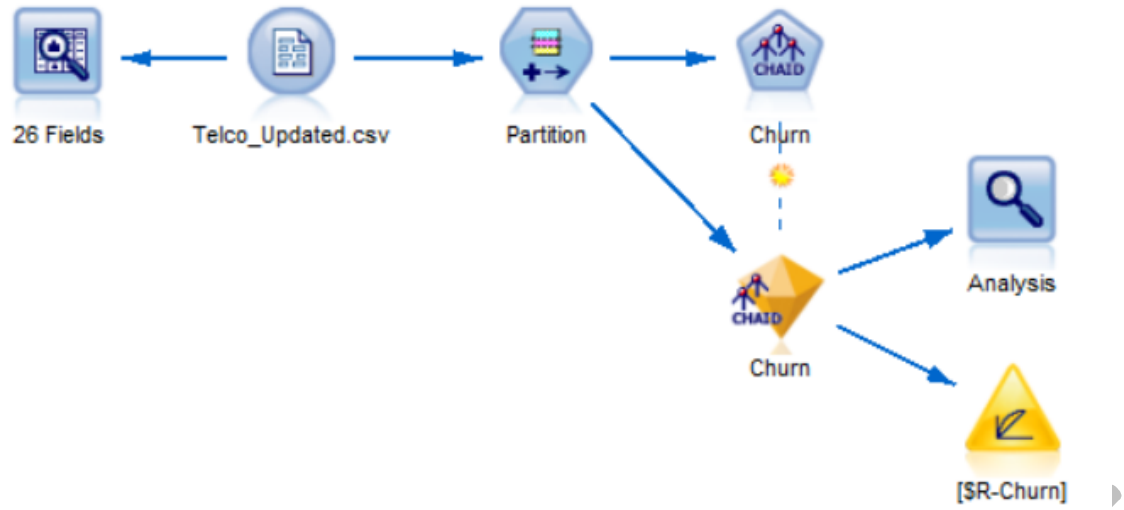


Figure 9.28 The newly generated CHAID nugget added to the Analysis node and Evaluation node

To see the effect of building a model downstream of a Partition node:

Run the Analysis node

Figure 9.29 shows the Analysis node output.

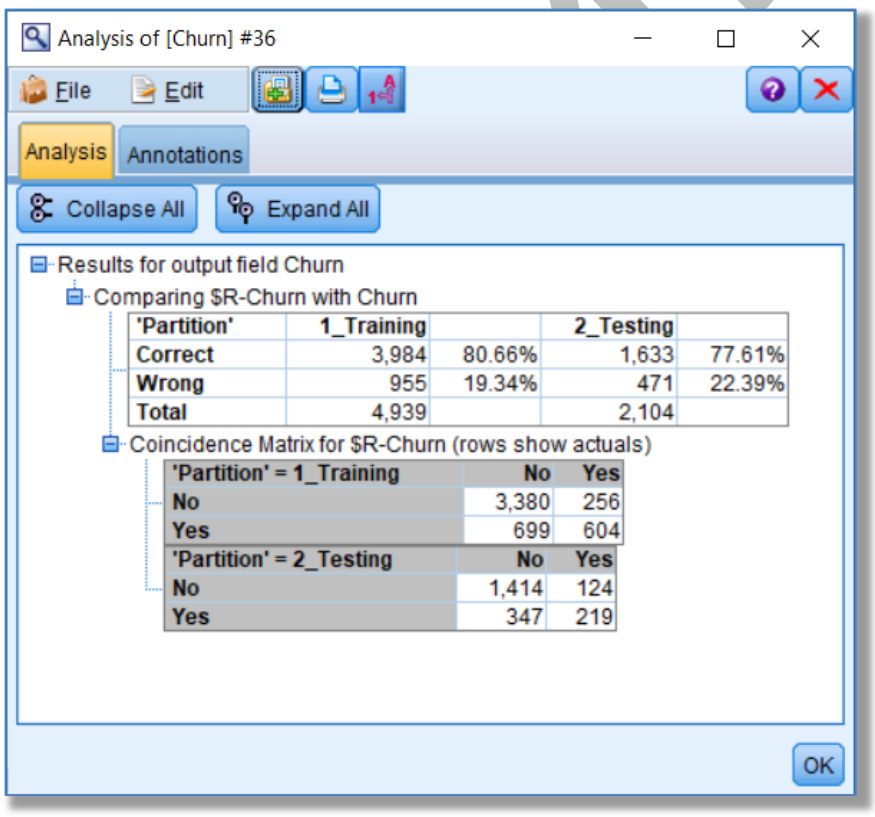


Figure 9.29 The Analysis node showing the relative performance of a partitioned model

As we can see from the results, not surprisingly, the model appears more accurate when applied to the Training sample (80.66% correct) than the Testing sample (77.61% correct). However, the difference is quite small so on this evidence we might conclude that the model shows little sign of overfitting. At this stage we could try to improve the model or test to see if it continues to give similar results by re-building it with different random splits (by generating new random seed numbers). For now though, we can:

Return to the stream and run the Evaluation node

Figure 9.30 shows the output from the Evaluation node.

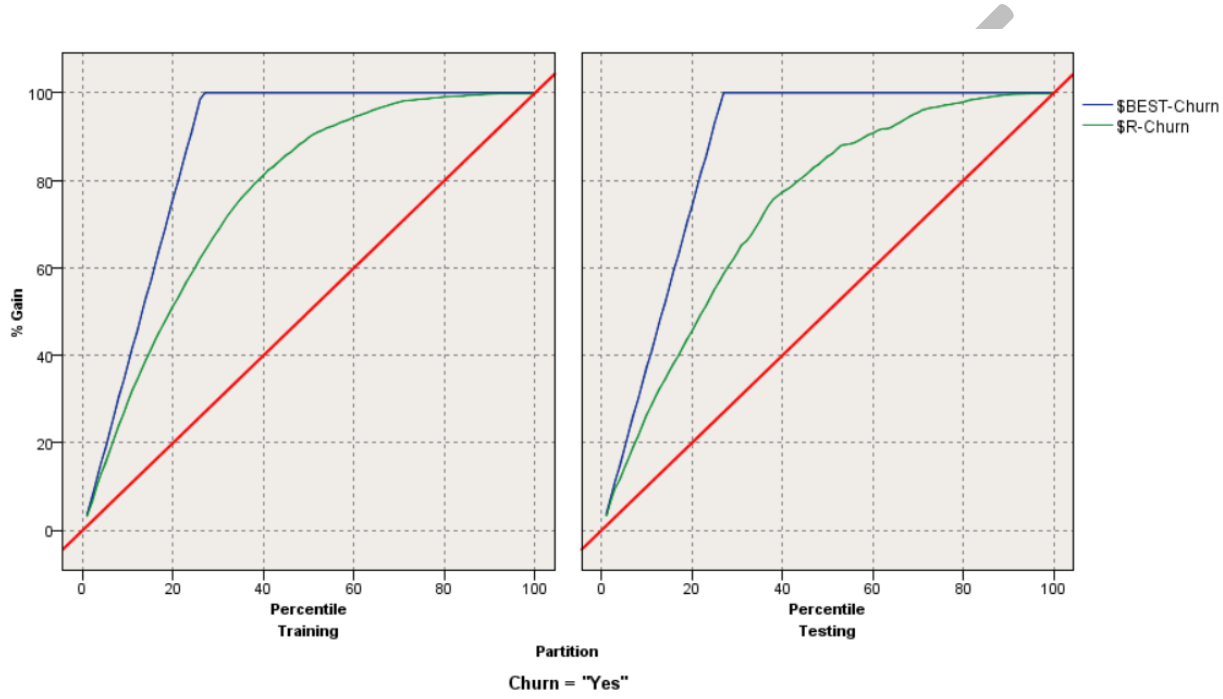


Figure 9.30 The output from the Evaluation node displaying the relative performance of a partitioned model

As we can see, the Evaluation node has now created two Gains charts for the testing sample and the training sample respectively. The charts show a similar story to Analysis node in that the model performs slightly more poorly on the Testing sample. The partition node has created this split in the data by creating a special partition field in the data. To view the field:

From the Output palette, add a Table node to the CHAID model nugget

Figure 9.31 shows the Table node added to the stream.

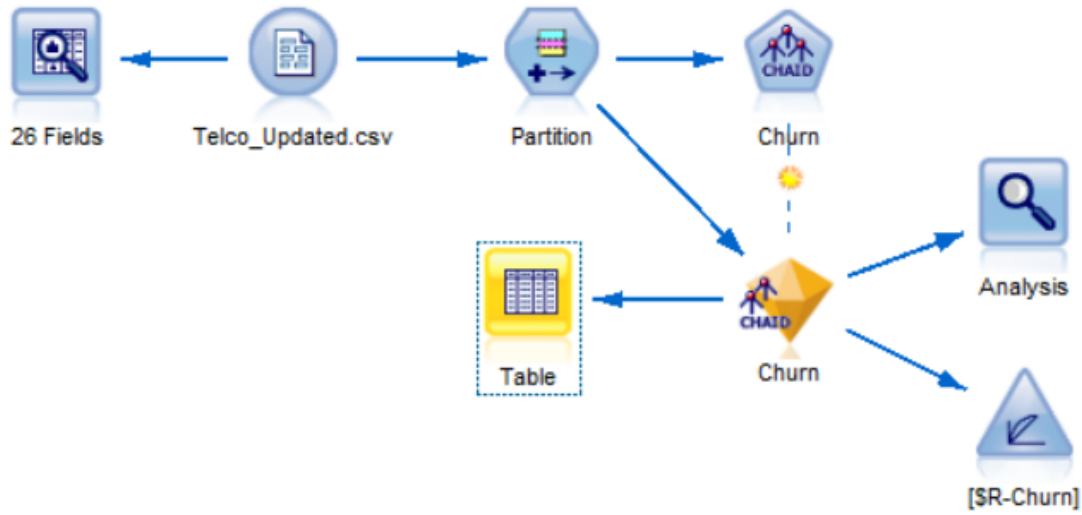


Figure 9.31 Table node added to the stream to illustrate the Partition field

Run the Table node

Figure 9.32 shows the output from the Table node and the generated Partition field.

Table (29 fields, 7,043 records)

File Edit Generate

Table Annotations

	ayment_Type_3Grp	Average_Monthly_Bill	Very_Loyal	Cashback	Customer_Tier	Partition	$R-Churn	$RC-Churn
2527	ard_Payment	95.188	F	0.000	c. Standard	1_Training	Yes	0.613
2528	count_Debit	82.000	F	0.000	c. Standard	1_Training	Yes	0.613
2529	ard_Payment	66.688	F	0.000	c. Standard	1_Training	No	0.892
2530	voice	78.062	F	0.000	c. Standard	2_Testing	Yes	0.613
2531	ard_Payment	78.812	F	0.000	c. Standard	1_Training	Yes	0.633
2532	voice	18.875	F	0.000	c. Standard	2_Testing	No	0.994
2533	ard_Payment	85.938	F	0.000	c. Standard	1_Training	Yes	0.613
2534	ard_Payment	79.250	F	0.000	c. Standard	1_Training	Yes	0.613
2535	ard_Payment	69.688	F	0.000	c. Standard	1_Training	No	0.983
2536	count_Debit	99.250	F	0.000	c. Standard	2_Testing	Yes	0.613
2537	ard_Payment	90.125	F	0.000	c. Standard	1_Training	Yes	0.613
2538	ard_Payment	73.000	F	0.000	c. Standard	1_Training	No	0.534
2539	count_Debit	96.250	F	0.000	c. Standard	1_Training	Yes	0.613
2540	ard_Payment	20.625	F	0.000	c. Standard	2_Testing	No	0.908
2541	ard_Payment	82.562	F	0.000	c. Standard	2_Testing	Yes	0.613
2542	count_Debit	92.688	F	0.000	c. Standard	1_Training	Yes	0.613
2543	voice	17.250	F	0.000	c. Standard	1_Training	No	0.908
2544	ard_Payment	18.438	F	0.000	c. Standard	1_Training	No	0.908
2545	ard_Payment	98.375	F	0.000	c. Standard	1_Training	Yes	0.613
2546	voice	17.250	F	0.000	c. Standard	1_Training	No	0.908

OK

Figure 9.32 Output from the Table node showing the random values of the Partition field

As we can see, the Partition node simply creates a field called 'Partition' and randomly assigns the values '1_Training' or '2_Testing', whilst honouring the requested proportions to the records in the dataset. The relevant nodes within Modeler detect the presence of this field and take account of it when building or evaluating models downstream of it.

The Auto Classifier Node 

Within Modeler there are a large number of algorithms that can be used to predict classification outcomes such as customer churn. The Auto Classifier node can help to narrow down which technique (or combination of techniques) will yield the best model. Assuming that the modelling fields have been correctly configured in the stream Type node (or Type tab in the Data Source node) and that the target has been correctly typed as a flag or nominal field, the Auto Classifier node can automatically execute several model-building algorithms at once. The goal of this procedure is usually to retain the resulting best models according to a pre-specified criterion such as overall accuracy, lift or profit. To see this procedure in action, from the Section 9 folder open the Modeler stream:

Section 9 Auto Classifier.str

Figure 9.33 shows the stream.

Figure 9.33 The Modeler stream 'Auto Classifier.str'

From the Modelling palette:

Select and attach an Auto Classifier node to the Partition node in the stream

(If you can't see the Auto Classifier node, make sure the node filter tab marked 'All' is highlighted on the left side of the palette).

Figure 9.34 shows the updated stream.

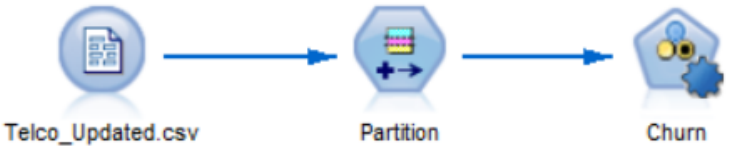


Figure 9.34 The Modeler stream 'Auto Classifier.str' with the Auto Classifier node added

To see the options associated with this node:

Double-click the Auto Classifier node to edit it

Figure 9.35 shows the edited Auto-Classifier node.

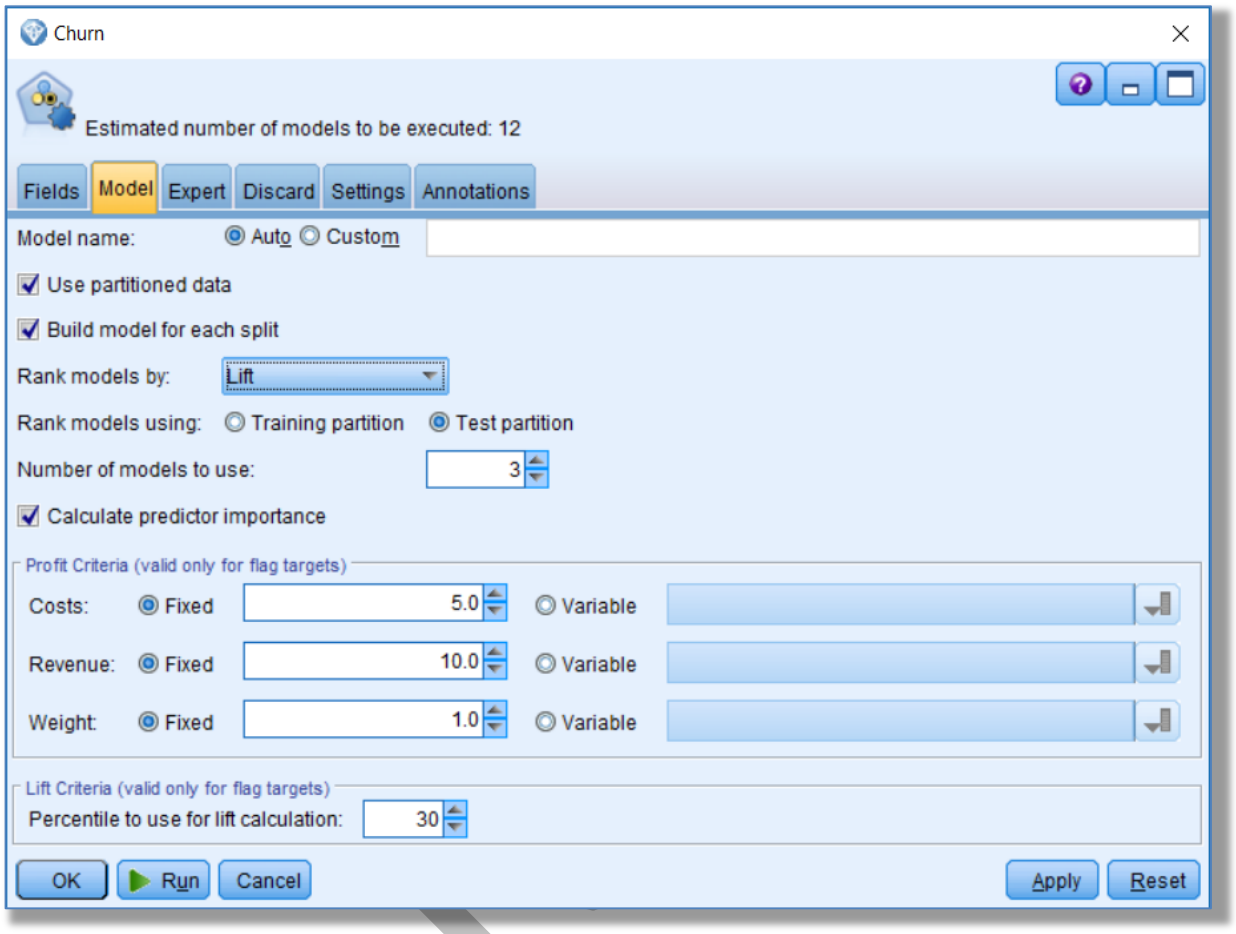


Figure 9.35 The edited Auto Classifier node showing the options within the Model tab

Within the edited node, some of the more interesting options in the Model tab include how the procedure ranks the Models it builds. Here we can see the default option is to rank models by 'Lift'. In fact, the options for ranking models include:

- Overall Accuracy – This is simply the overall (or average) accuracy the model exhibits when considering all the groups in the target field. This measure is more useful when the group sizes are close to equal.

- Area Under the Curve – The Area Under the Curve (or AUC) measure looks at the false positive vs true positive rate. Recall that the Gains chart displays a diagonal line representing a random classifier. The diagonal line simply indicates that using a random model, you would expect to find 50% of the responders (or churners) from randomly sampling 50% of the data. The AUC score for a model that followed the random line would therefore be 0.5 (scores lower than 0.5 would indicate the model was worse than random). A score for a perfect classification model would be 1. As you

might expect, most models are somewhere between these two values with higher values associated with higher accuracy in predicting the outcome of interest in the target group (usually the category denoted with values such as 'Yes' or 'T' or '1').

- Profit – As the dialog shows, it's possible to include costs and benefit parameters when the target field is binary (flag). This is useful if we can associate numerical values with the outcomes such as when trying to select a model that will drive the most profit for a marketing campaign. Alternatively, it could be used to select a model that minimises the financial loss associated with outcomes such as asset failures. Note that the user can enter specific parameter values or identify variables that record the revenue or costs associated with the outcomes.

- Lift – The lift value is a measure of how much better the model does at classifying the data (i.e. predicting the outcome) compared to a random model (or the baseline proportion). Here the proportion of churners is about 26%. So, a random model predicting every case to be a churner would be right about 26% of the time. This would generate a lift value equal to 1.0. If however the model was better than random, and was able to predict the proportion of churners with 52% accuracy it would be twice as accurate as the random approach and would generate a lift value of 2.0. The higher the lift value, the better than random the model is. Here the Lift measure is based on top 30% of data where the model is most confident.

- Number of Fields – The last method to rank the models is simply by the number of fields. Models based on fewer fields are regarded as more desirable as they are likely to be simpler and more efficient. In Statistics, such models are sometimes described as having greater 'parsimony'.

Two other things to note from this tab are that firstly, by default, the procedure creates a nugget based on the best three models and that this value can be altered. Secondly, models are ranked based upon their performance on the Testing group as identified by the Partition node.

Before we run the procedure, let's examine the range of algorithms the Auto Classifier uses by default. Click the tab marked:

Expert

Figure 9.36 shows the Expert tab in the edited Auto Classifier node.

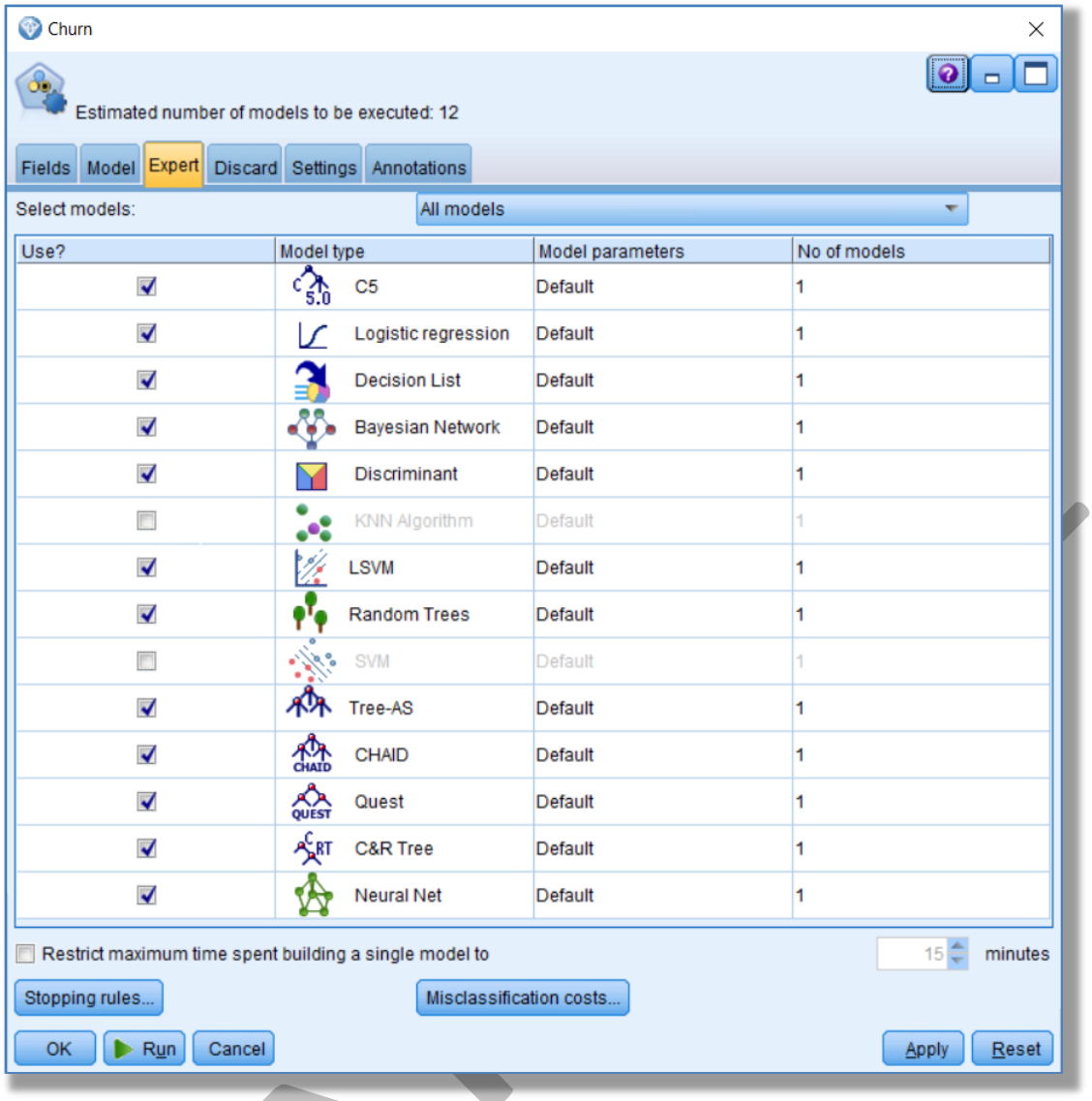


Figure 9.36 The edited Auto Classifier node showing the options within the Expert tab

Not only can we decide to include or drop individual algorithms from the Auto Classifier process, but we can also request that multiple models are built within the same techniques by editing an algorithm and specifying that additional parameters or settings are tried. To illustrate this, within the Model Parameters column:

Click the cell marked Default next to the C5 model type

Click:

Specify

Figure 9.37 shows the resulting sub-dialog.

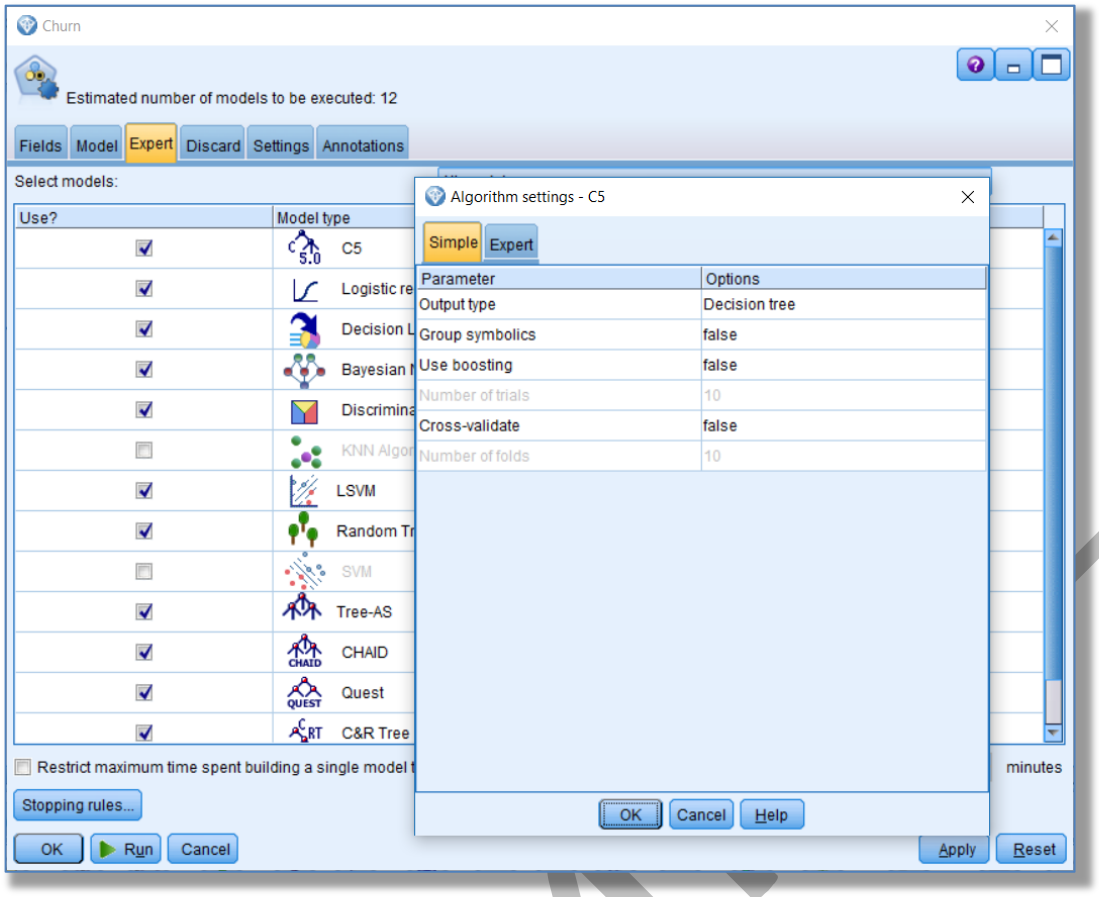


Figure 9.37 The Model Parameters sub-dialog for the C5 model type

Edit the cell next to the Output type parameter

From the drop-down menu request that the procedure also includes a C5 Ruleset by clicking:

Both

Figure 9.38 shows this:

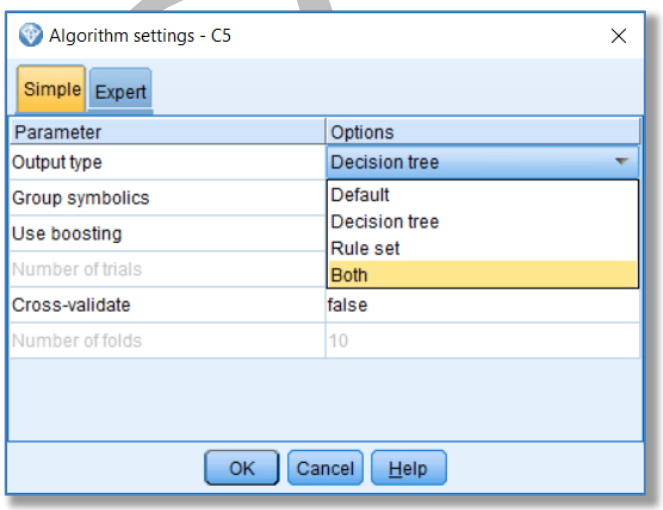


Figure 9.38 Requesting that a Ruleset model is built as a decision tree within the C5 model type

Click:

OK

We are returned to the main Expert tab within the Auto classifier. The cell corresponding to the C5 model type under the column header 'No. of models' now indicates that two C5 models will be built. We can also de-select some of the algorithms. To illustrate, uncheck the boxes in the column marked 'Use?' next to the following algorithms:

LSVM

Random Trees

Tree-AS

Figure 9.39 shows the updated Expert tab.

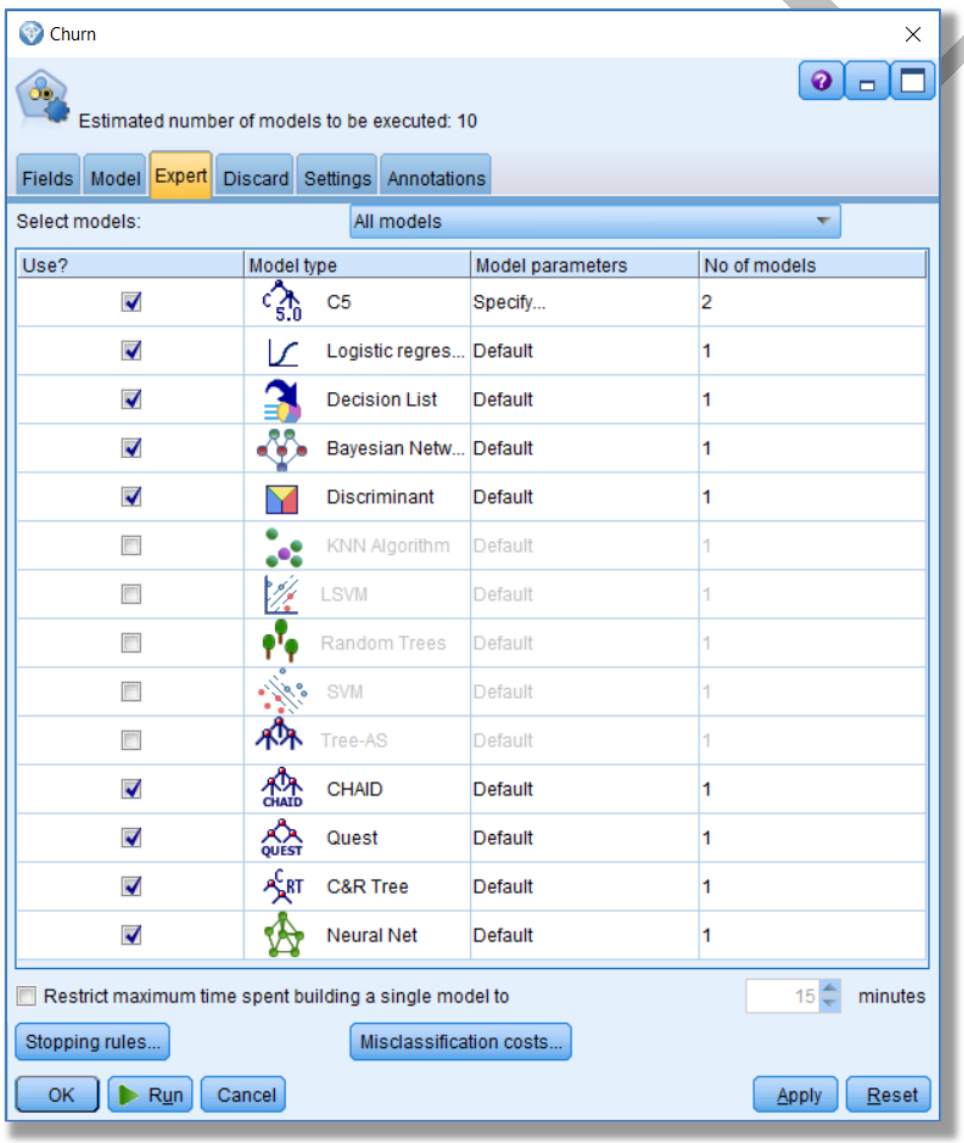


Figure 9.39 The updated Expert tab within the edited Auto Classifier node

Now if we run the procedure, Modeler will attempt to build 10 separate models retaining the 3 completed models with the highest lift value when applied to the Test partition group. Click:

Run

A progress window appears showing how many of the models are to be built and any which fail to complete or which are discarded.

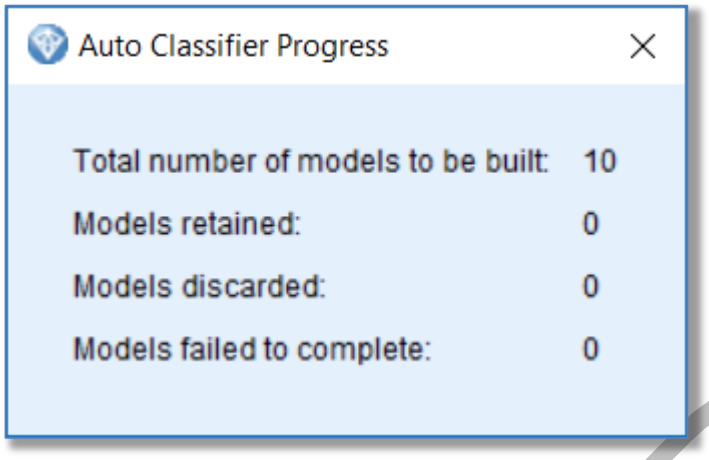


Figure 9.40 Auto Classifier Progress window

The Auto Classifier model is nugget is created as shown in figure 9.40.

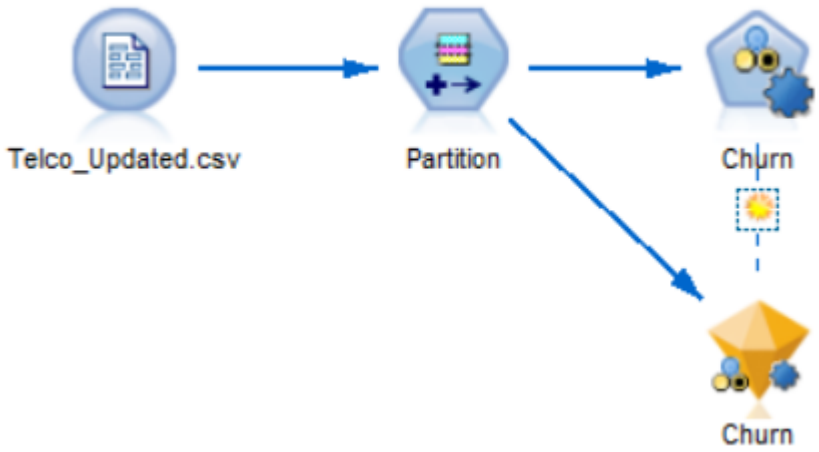


Figure 9.41 Auto Classifier model nugget added to the stream

To view the its contents:

Double-click the Auto Classifier model nugget

Figure 9.42 shows the contents of the nugget.

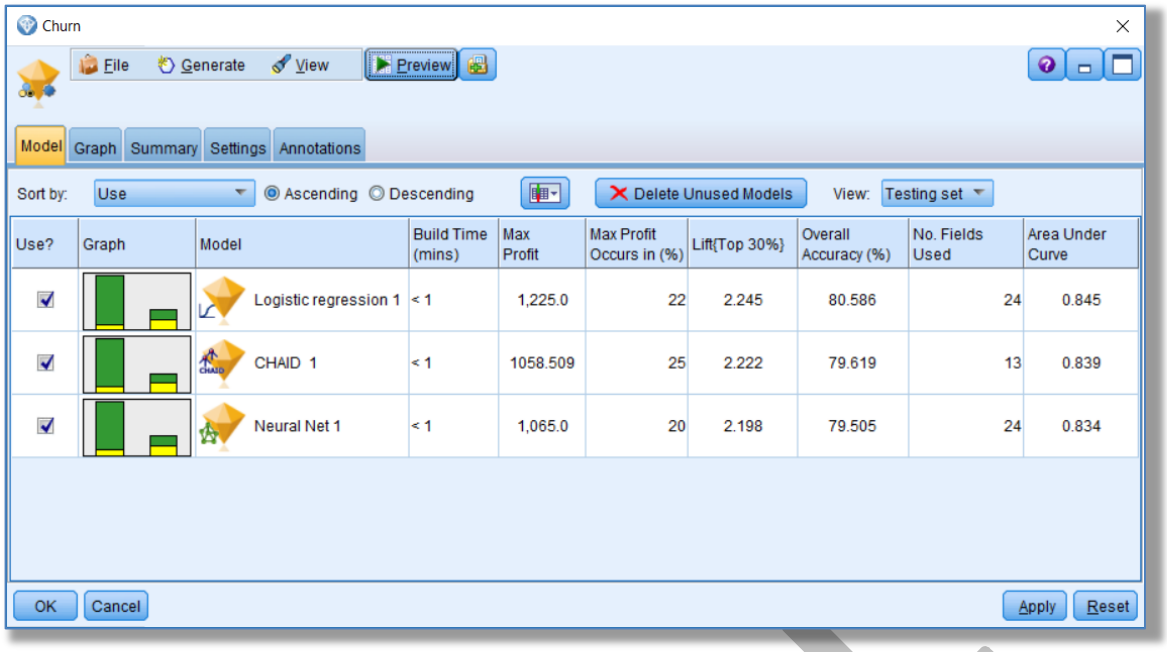


Figure 9.42 The results of the Auto Classifier procedure based on the Testing set

In our example, the procedure has selected three model types, Logistic Regression, CHAID and Neural Network (different results may occur if using a different build version of Modeler). This initial output screen contains a wealth of information that we may summarise here:

- A series of check boxes under the column marked 'Use?' indicating whether or not the model is to be retained or dropped.
- A clustered bar chart that shows the misclassification rate for each model. This can be double-clicked so the user can get a more detailed view.
- A model nugget for the individual model type that the user can double-click and browse to see more details.
- A series of performance measures showing, for example, the model build time, Lift value, overall accuracy, number of fields used and AUC values. In this case, the Logistic Regression model has the highest lift value, so it is shown at the top of the list.
- The option to click on a drop-down view box and switch between the statistics for the models' performance on the Testing set or the Training set

To see if the model performed equally well on the Training set, click the drop-down menu next to the view label and choose:

Training Set

Figure 9.43 shows the resulting model performance output,

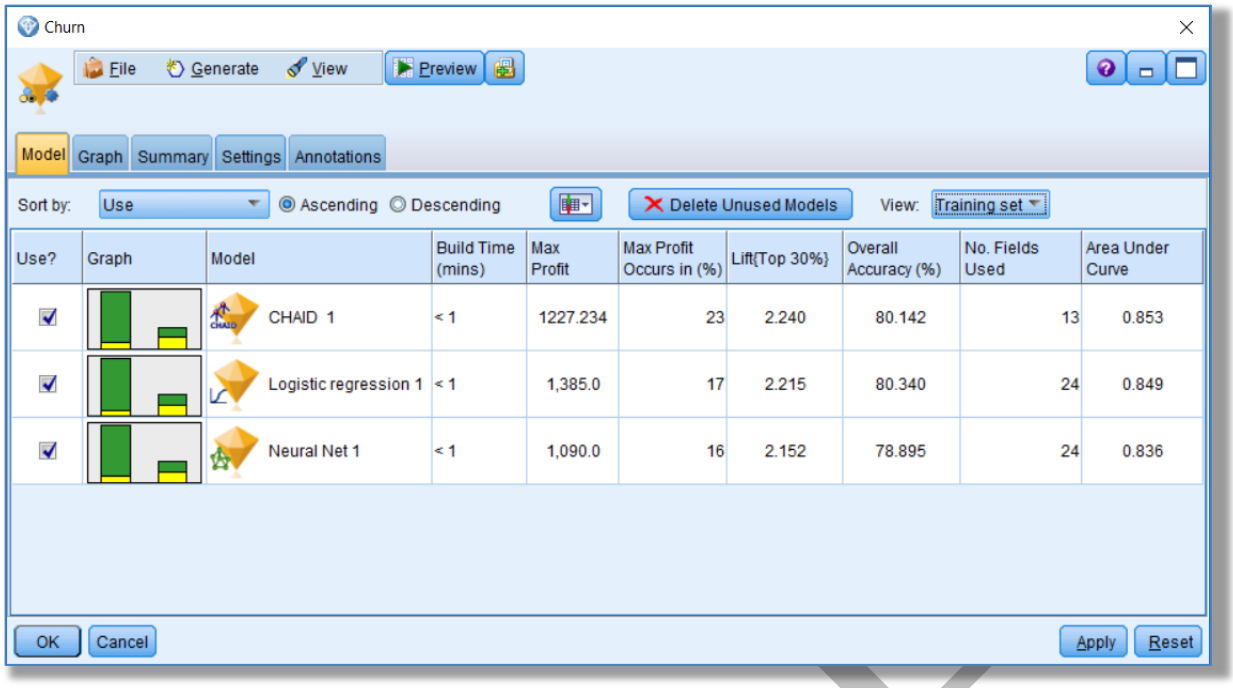


Figure 9.43 The results of the Auto Classifier procedure based on the Training set

We can see that in actual fact, the CHAID model performed better than the Logistic Regression model in terms of the lift value when applied to the Training set. However, the Logistic Regression model gave superior results on the Testing set. To view the contents of any model nugget we need only double-click it. As an example:

Double-click the Logistic Regression nugget

Click the Advanced tab

As figure 9.44 shows, Logistic Regression models are displayed as a series of statistical coefficients and as such are quite different from the rule or tree-based models such as C5 and CHAID.

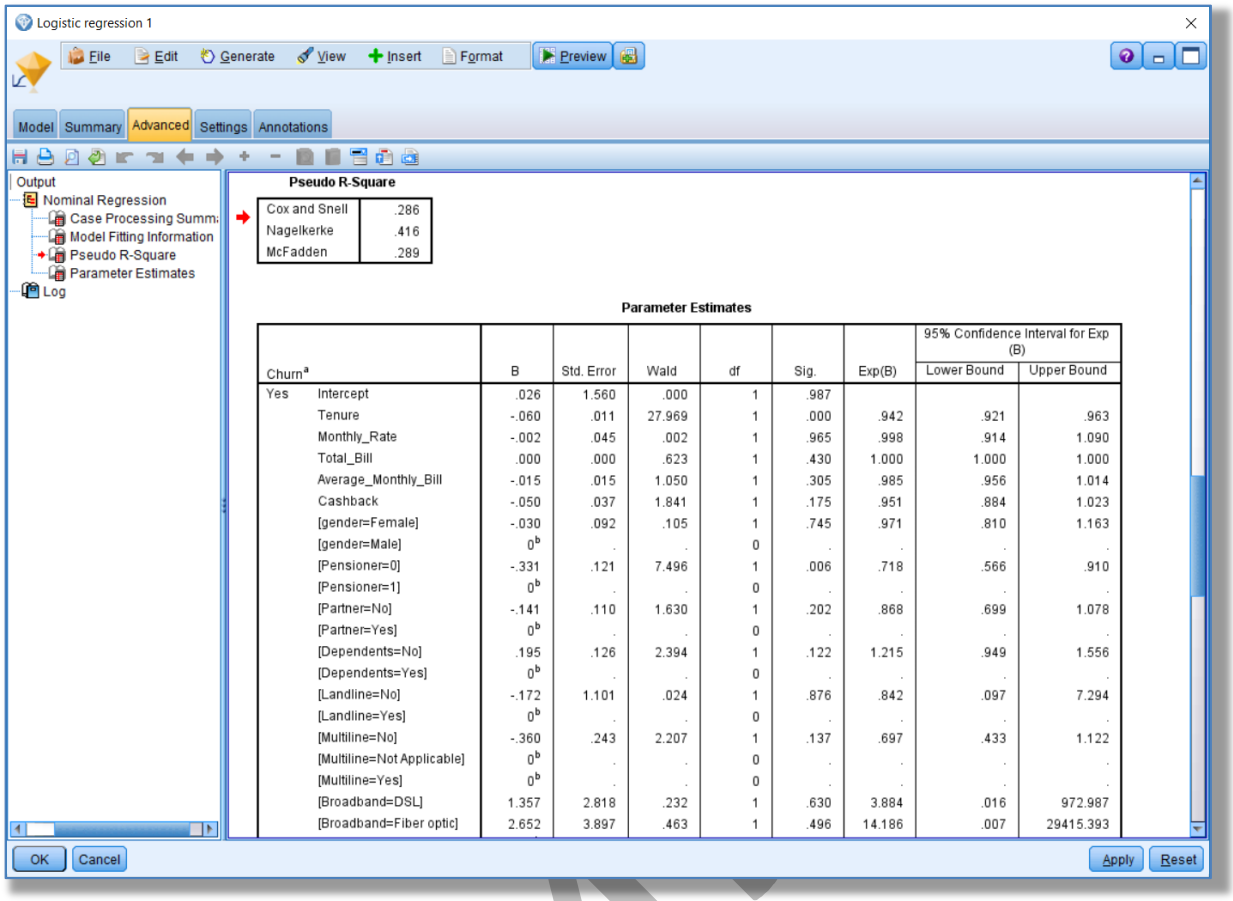


Figure 9.44 Logistic Regression output from the Logistic model in the Auto Classifier node

To return to the Auto Classifier output, click:

Cancel

As well as clicking on the individual clustered bar charts next to each generated model, we can view the performance of the auto-classifier nugget using all three models combined in a single clustered bar chart. To do so, click the tab marked:

Graph

Figure 9.45 shows the output.

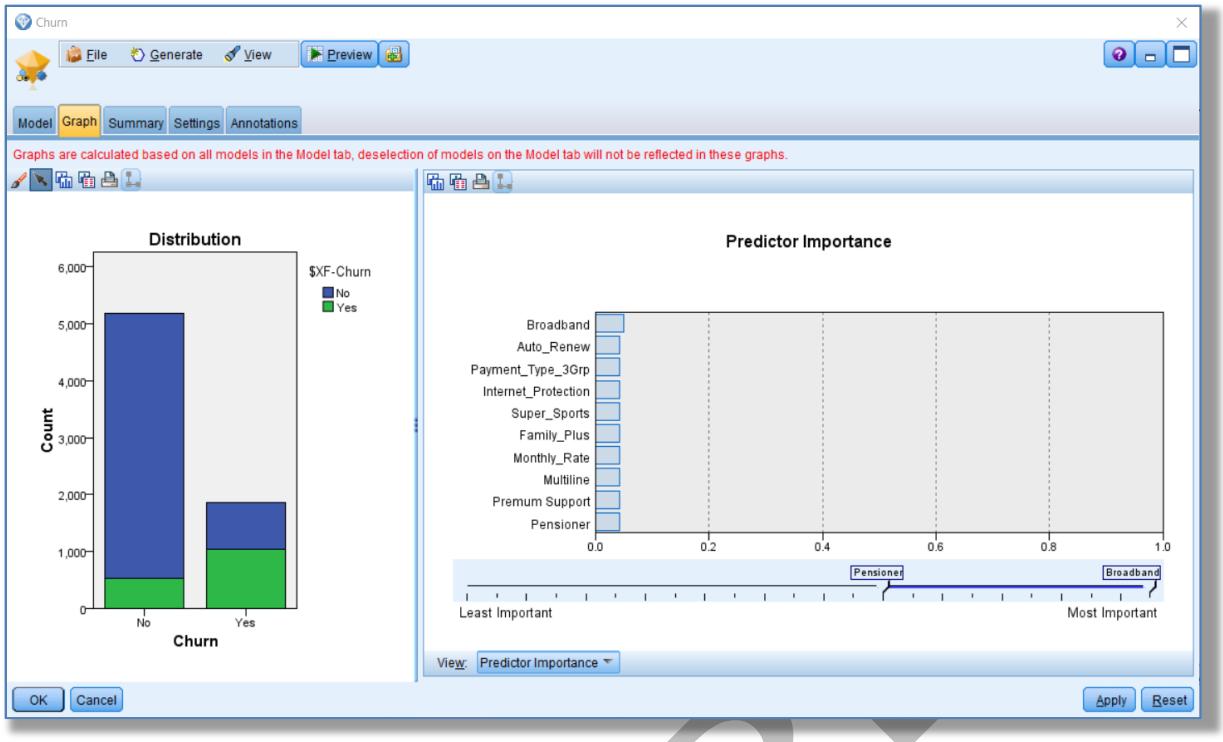


Figure 9.45 The Graph output from the Auto Classifier model nugget

The graph tab shows the overall classification chart for the combined models. A combination of models such as this are often referred to as an Ensemble model. The bars show the actual status of the customers whilst the colours show the predicted outcomes. You can see that the combined models do a better job of predicting the customers who have not defected compared to those who have churned. You can also see that the predicted outcome variable the Auto Classifier produces is labelled '$XF-Churn'. The '$X' indicates that this is an ensemble model. The predictor importance chart shows which variables are most important in the three models that the procedure has chosen. A note at the top of the screen reminds us that the charts are based on all the models in the nugget and will not be affected or updated by changing whether an individual model is selected or not in the Model tab. We can of course evaluate the ensemble model performance by attaching an Analysis or Evaluation node to the nugget. From the output palette:

Choose an Analysis node and attach it to the Auto Classifier nugget

Edit the node so that the 'Coincidence Matrix' is included

Now choose an Evaluation node and attach it to the Auto Classifier nugget

Edit the node so that the 'Best Line' is included

Figure 9.46 shows the updated stream.

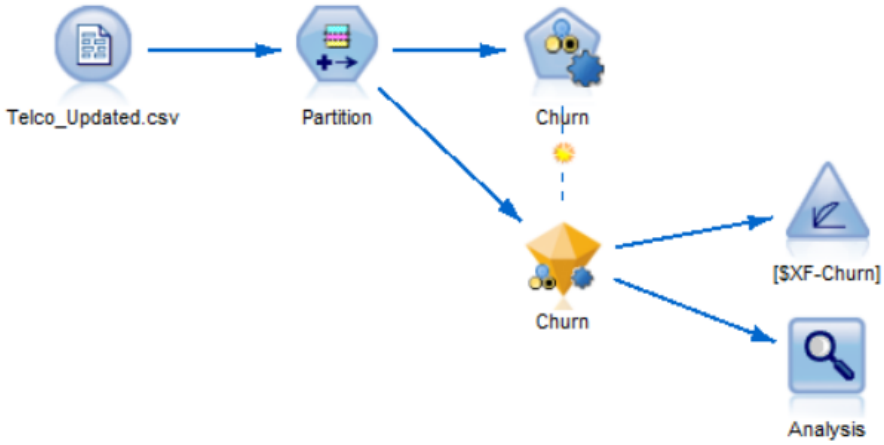


Figure 9.46 The Auto Classifier ensemble model with an Evaluation and Analysis node attached

Run the Evaluation node

Figure 9.47 shows the resulting output.

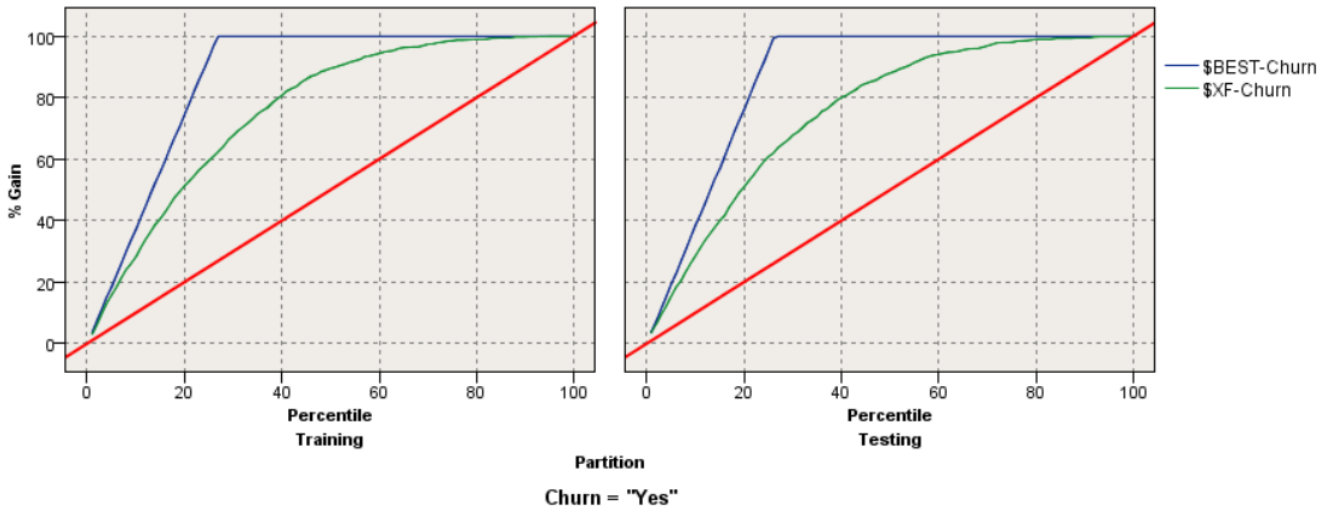


Figure 9.47 Gains charts from the Evaluation node showing the performance of the combined (ensemble) models in the Auto Classifier nugget

By default, Modeler combines the scores from the different models using 'Confidence-Weighed Voting'. This method enables each model to 'vote' as to whether or not they predict the customer to churn. With confidence-weighted voting, each vote is weighted based on the confidence value for that prediction. In a situation where one model predicts a customer to be a churner and the other two models predict them to be remain a current customer, the model that predicts 'churn' will win if its confidence value is greater than other the two predictions combined. Here the Gains chart only shows a single curve representing the ensemble model. In this case, it indicates a good performance for predicting churn compared to the random classifier model line (the diagonal). To continue:

Run the Analysis node

Figure 9.47 shows the Analysis node output.

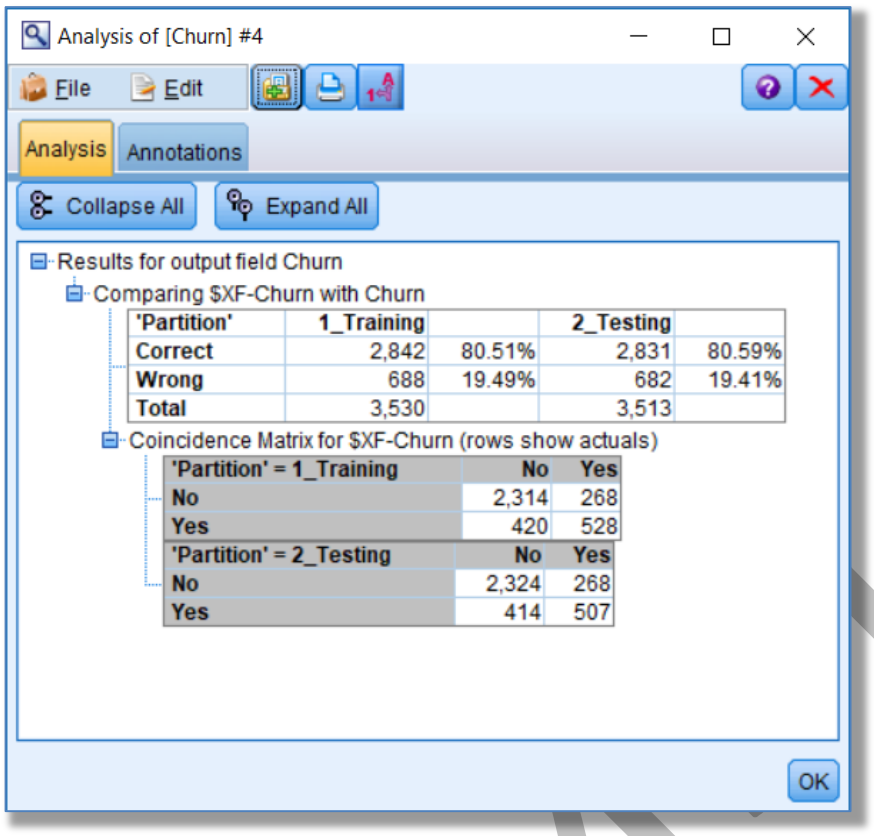


Figure 9.48 Analysis node output showing the overall accuracy for the Auto Classifier ensemble model and coincidence matrices

The ensemble model performs well, with the majority of the churners in the testing set correctly classified. This time we can actually affect the model Evaluation and Analysis node results by selecting or deselecting models within the Auto Classifier nugget. To illustrate:

Double-click the Auto Classifier nugget

Uncheck the last model type in the 'Use?' column (the Neural Network)

Figure 9.49 shows this window after the model is de-selected

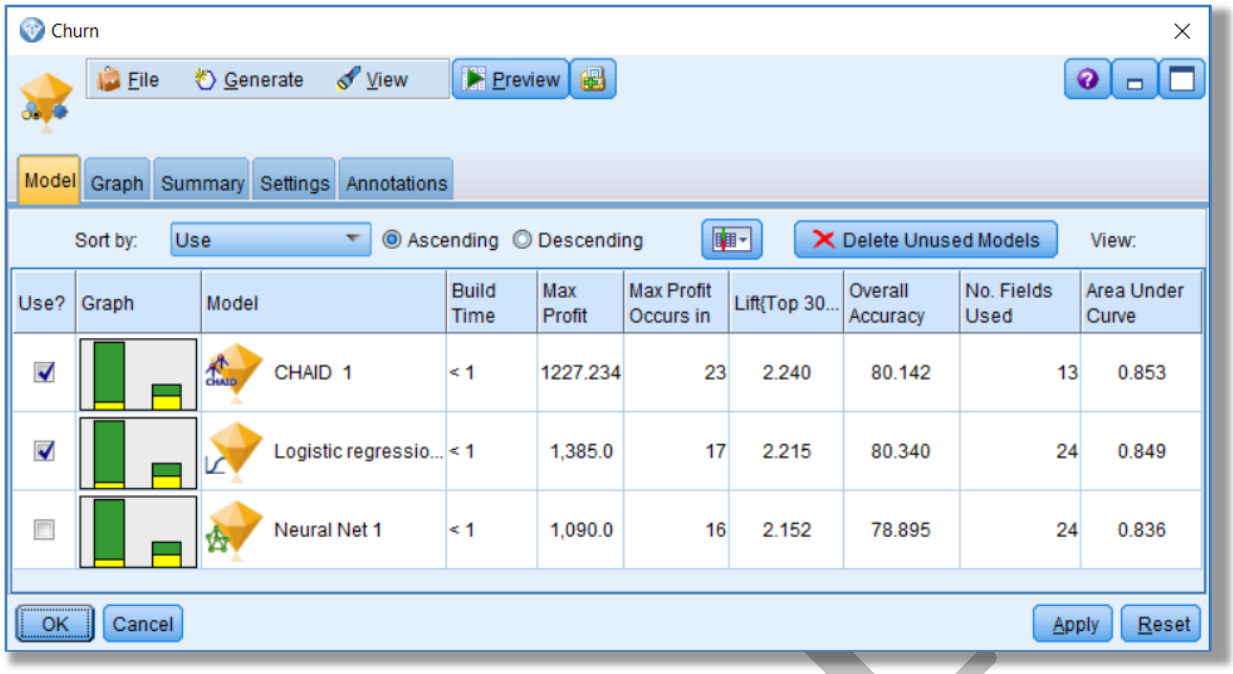


Figure 9.49 De-Selecting the Neural Network model type in the Auto Classifier nugget

Click OK and re-run the Analysis node

Figure 9.50 shows the results.

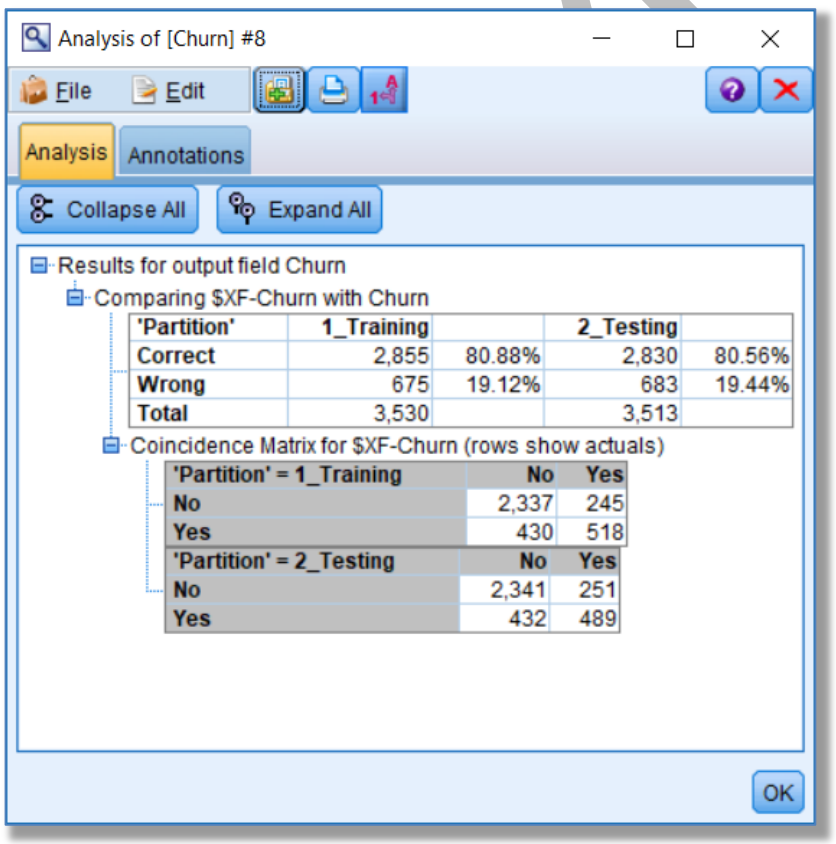


Figure 9.50 The Analysis node results based on the Logistic Regression and CHAID models in the Auto Classifier nugget

The results are quite interesting as they illustrate one of the reasons why ensemble models can be powerful. Even though the Neural Network model was the weakest classification model, by removing it, the ensemble model is less accurate at predicting the outcomes in the testing set. Perhaps the model was more accurate than the other two model types in a particular sub-set of cases and because of this the confidence-weighted voting was enough to ensure it made a positive contribution to the accuracy of the ensemble model.

In the next section we will turn our attention to the Deployment phase of the CRISP-DM process by looking at how we can use these models to score data about current customers.