Modeling Customer Response (Auto Classifier) Model

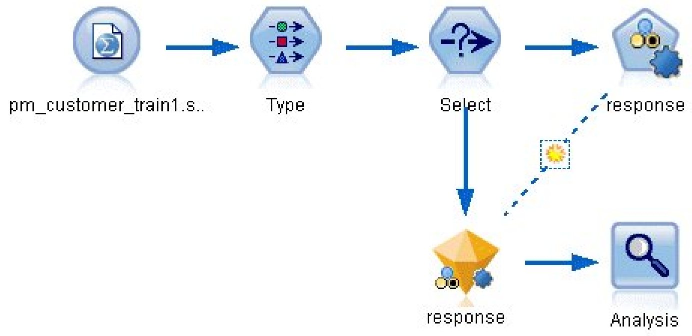
The Auto Classifier node enables you to automatically create and compare a number of different models for flags (such as whether or not a given customer is likely to default on a loan or respond to a particular offer) targets.

In this example we'll search for a flag (yes or no) outcome. Within a relatively simple stream, the node generates and ranks a set of candidate models, chooses the ones that perform the best, and combines them into a single aggregated model. This approach combines the ease of automation with the benefits of combining multiple models, which often yield more accurate predictions than can be gained from any one model.

This example is based on a fictional company that wants to achieve more profitable results by matching the right offer to each customer.

This approach stresses the benefits of automation. For a similar example that uses a continuous (numeric range) target, see Property Values (Auto Numeric).

Figure 1. Auto Classifier sample stream



The file *pm_customer_train1.sav* has historical data tracking the offers made to specific customers in past campaigns, as indicated by the value of the *campaign* field. The largest number of records fall under the *Premium account* campaign.

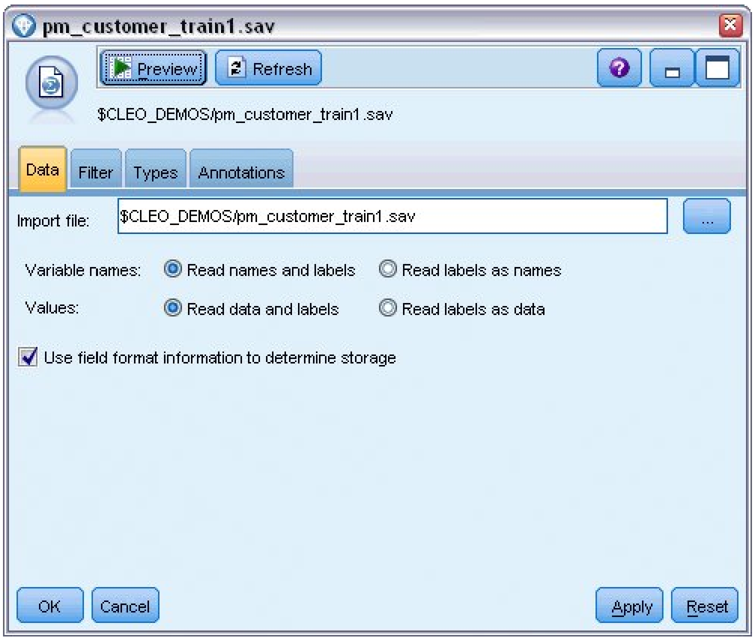
The values of the *campaign* field are actually coded as integers in the data (for example *2 = Premium account*). Later, you'll define labels for these values that you can use to give more meaningful output.

The file also includes a *response* field that indicates whether the offer was accepted (0 = *no*, and 1 = *yes*). This will be the **target field**, or value, that you want to predict. A number of fields containing demographic and financial information about each customer are also included. These can be used to build or "train" a model that predicts response rates for individuals or groups based on characteristics such as income, age, or number of transactions per month.

Building the Stream

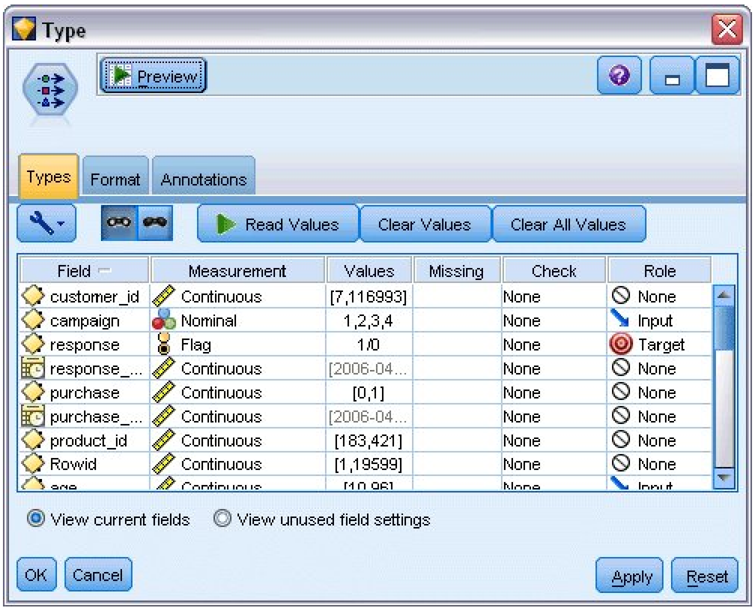
1. Add a Statistics File source node.

   Figure 1. Reading in the data



2. Add a Type node, and select *response* as the target field (Role = Target). Set the Measurement for this field to Flag.

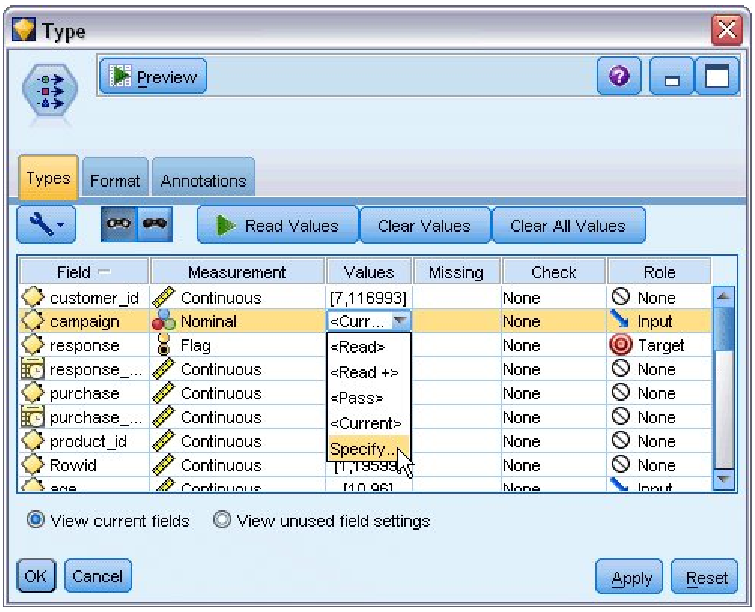
Figure 2. Setting the measurement level and role



3. Set the role to None for the following fields: *customer_id*, *campaign*, *response_date*, *purchase*, *purchase_date*, *product_id*, *Rowid*, and *X_random*. These fields will be ignored when you are building the model.
4. Click the Read Values button in the Type node to make sure that values are instantiated.

   As we saw earlier, our source data includes information about four different campaigns, each targeted to a different type of customer account. These campaigns are coded as integers in the data, so to make it easier to remember which account type each integer represents, let's define labels for each one.

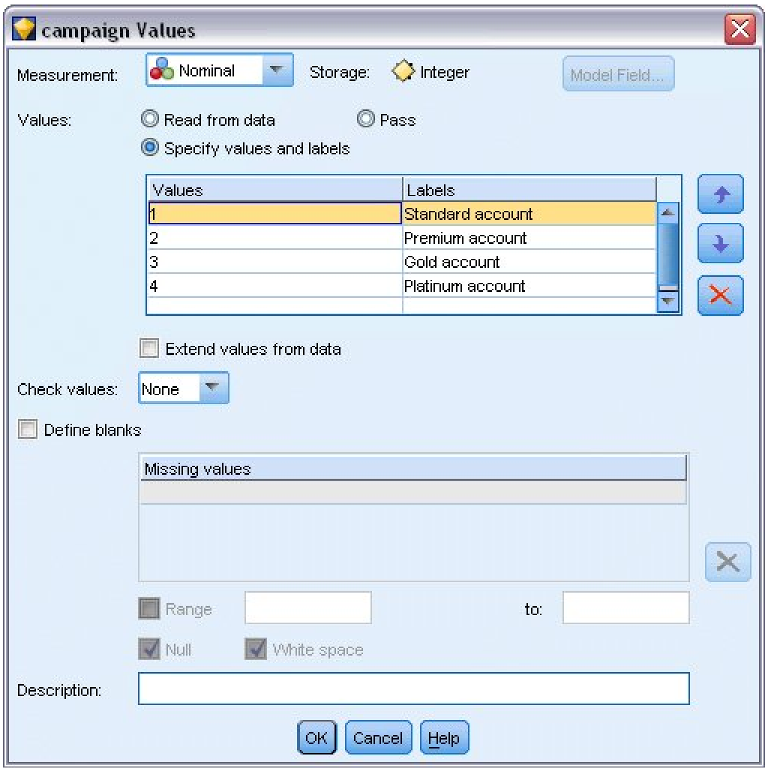
Figure 3. Choosing to specify values for a field



5. On the row for the campaign field, click the entry in the Values column.

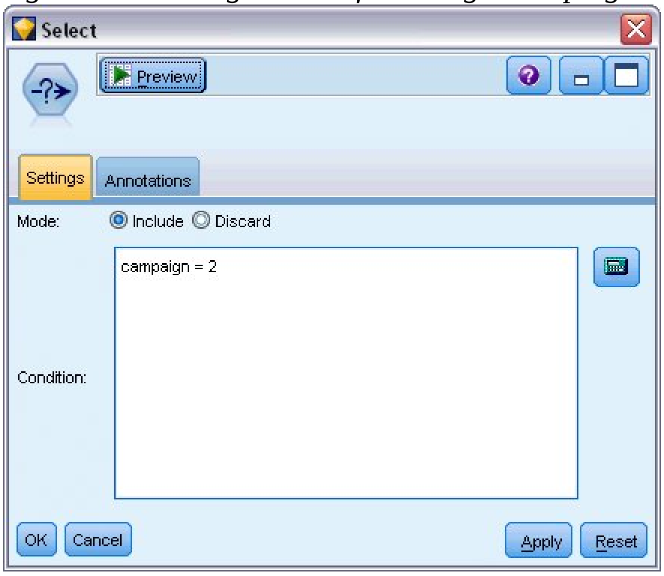
6. Choose Specify from the drop-down list.

Figure 4. Defining labels for the field values



Although the data includes information about four different campaigns, you will focus the analysis on one campaign at a time. Since the largest number of records fall under the Premium account campaign (coded *campaign=2* in the data), you can use a Select node to include only these records in the stream.

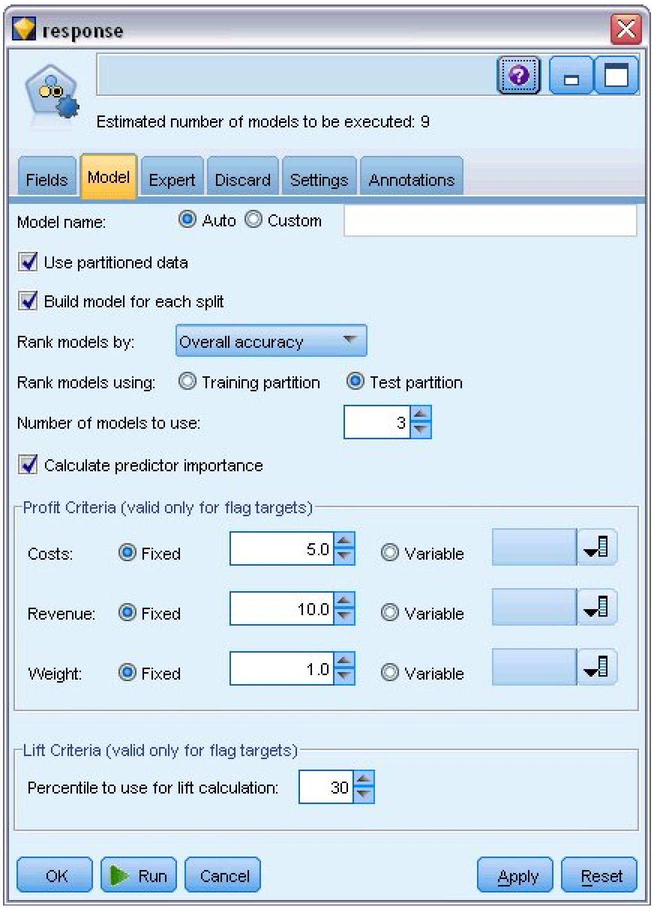
Figure 6. Selecting records for a single campaign



Generating and Comparing Models

1. Attach an Auto Classifier node, and select Overall Accuracy as the metric used to rank models.
2. Set the Number of models to use to 3. This means that the three best models will be built when you execute the node.

Figure 1. Auto Classifier node Model tab

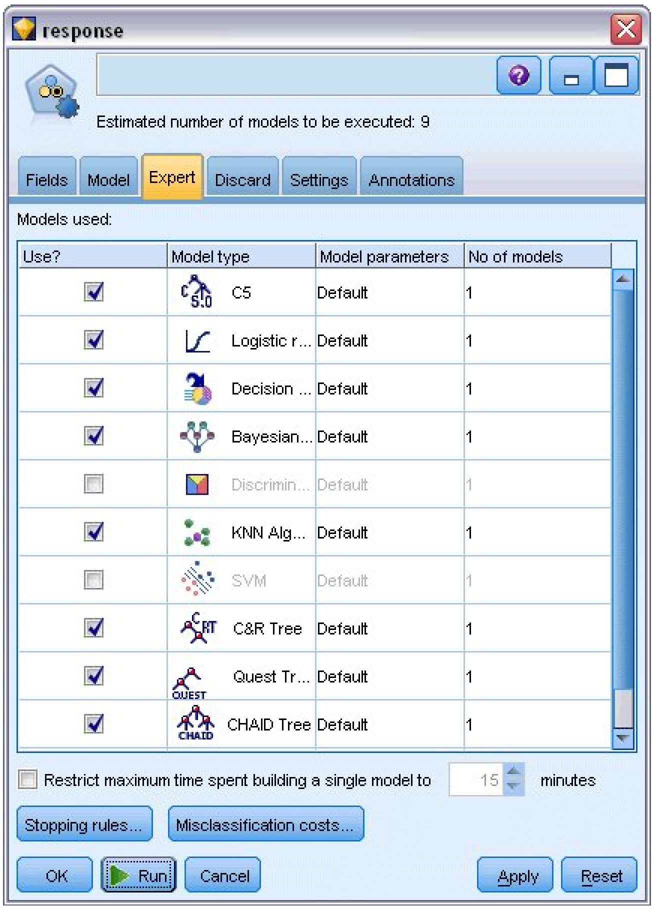


On the Expert tab you can choose from up to 11 different model algorithms.

3. Deselect the Discriminant and SVM model types. (These models take longer to train on these data, so deselecting them will speed up the example. If you don't mind waiting, feel free to leave them selected.)

   Because you set Number of models to use to 3 on the Model tab, the node will calculate the accuracy of the remaining nine algorithms and build a single model nugget containing the three most accurate.

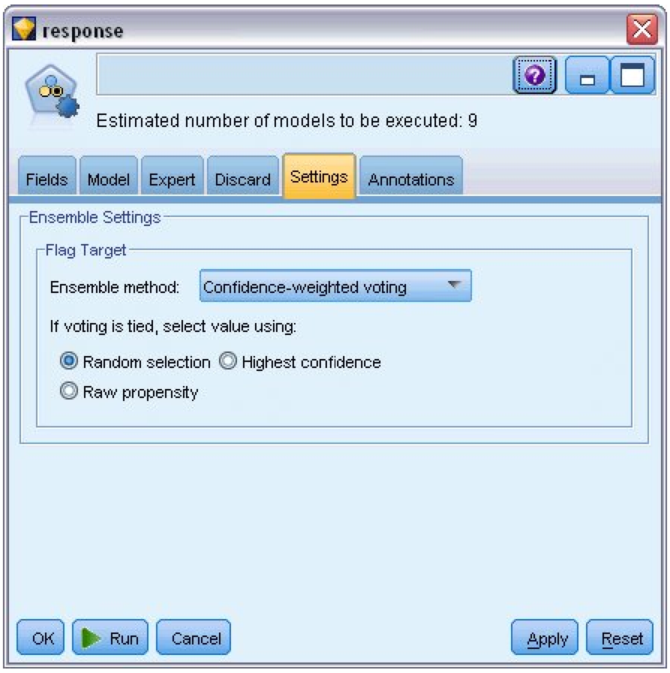
Figure 2. Auto Classifier node Expert tab



4. On the Settings tab, for the ensemble method, select Confidence-weighted voting. This determines how a single aggregated score is produced for each record.

   With simple voting, if two out of three models predict *yes*, then *yes* wins by a vote of 2 to 1. In the case of confidence-weighted voting, the votes are weighted based on the confidence value for each prediction. Thus, if one model predicts *no* with a higher confidence than the two *yes* predictions combined, then *no* wins.

Figure 3. Auto Classifier node: Settings tab



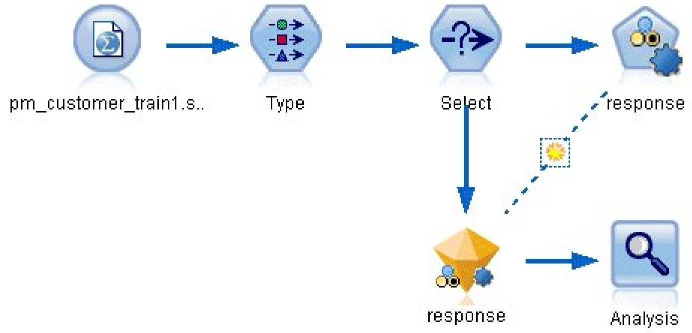
5. Click Run.

   After a few minutes, the generated model nugget is built and placed on the canvas, and on the Models palette in the upper right corner of the window. You can browse the model nugget, or save or deploy it in a number of other ways.

   Figure 4. Generated model displayed in the palette

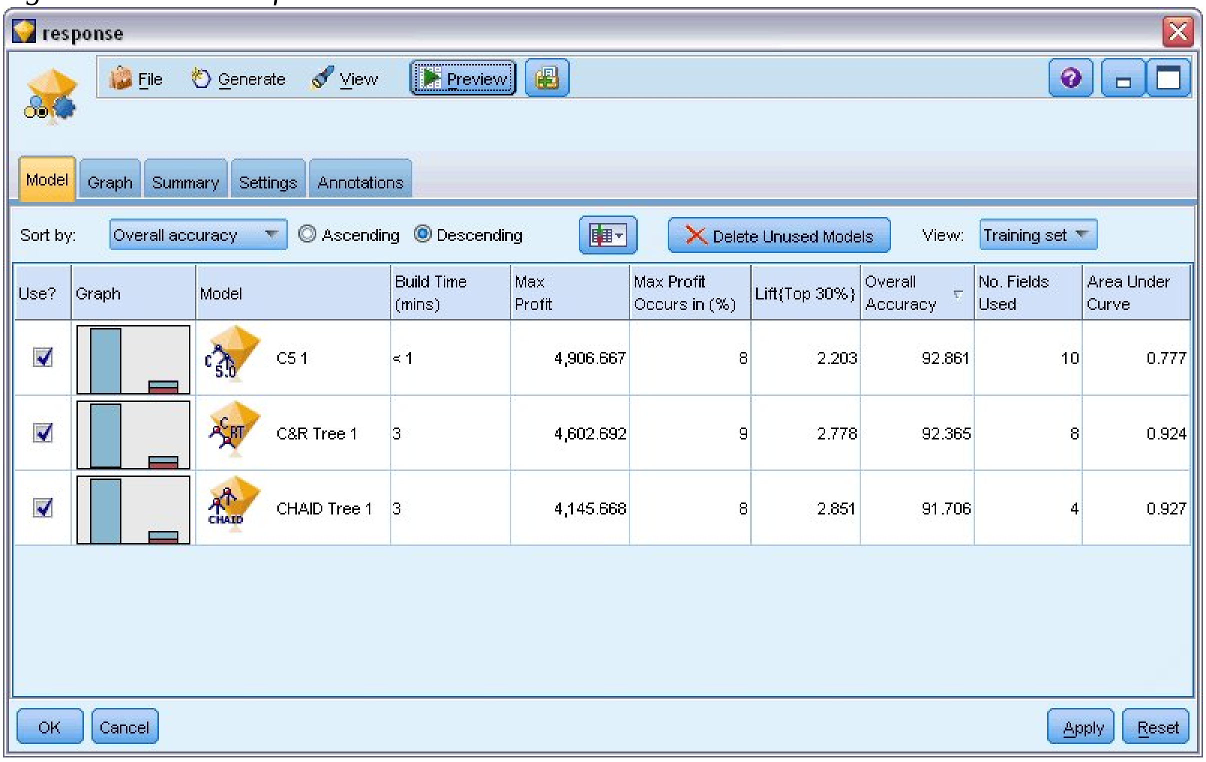
   Open the model nugget; it lists details about each of the models created during the run. (In a real situation, in which hundreds of models may be created on a large dataset, this could take many hours.)

Figure 5. Auto Classifier stream with model nugget



If you want to explore any of the individual models further, you can double-click on a model nugget icon in the Model column to drill down and browse the individual model results; from there you can generate modeling nodes, model nuggets, or evaluation charts. In the Graph column, you can double-click on a thumbnail to generate a full-sized graph.

Figure 6. Auto Classifier results



By default, models are sorted based on overall accuracy, because this was the measure you selected on the Auto Classifier node Model tab. The C51 model ranks best by this measure, but the C&R Tree and CHAID models are nearly as accurate.

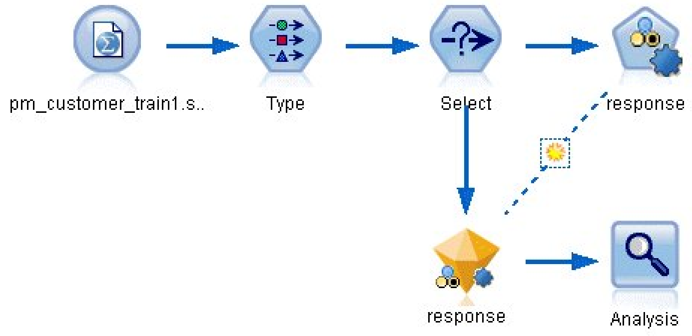
You can sort on a different column by clicking the header for that column, or you can choose the desired measure from the Sort by drop-down list on the toolbar.

Based on these results, you decide to use all three of these most accurate models. By combining predictions from multiple models, limitations in individual models may be avoided, resulting in a higher overall accuracy.

In the Use? column, select the C51, C&R Tree, and CHAID models.

Attach an Analysis node (Output palette) after the model nugget. Right-click on the Analysis node and choose Run to run the stream.

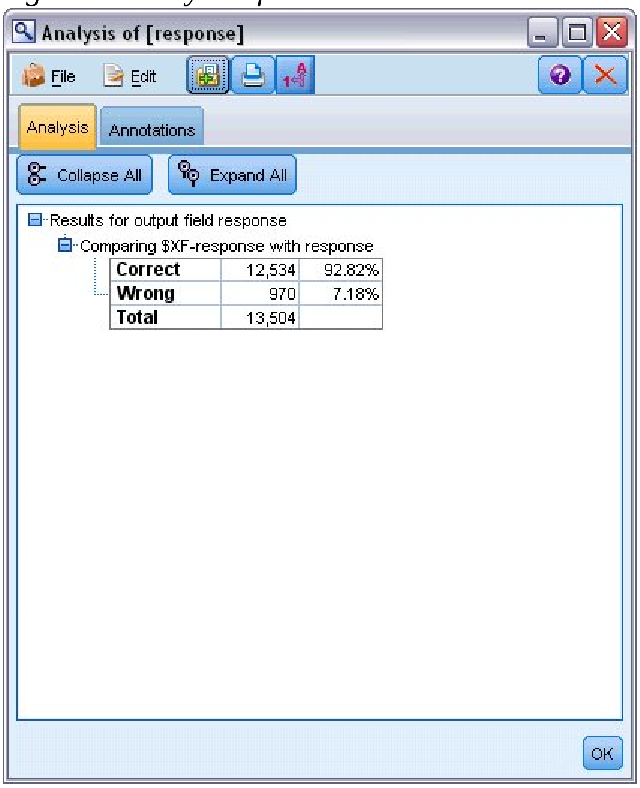
Figure 7. Auto Classifier sample stream



The aggregated score generated by the ensembled model is shown in a field named *$XF-response*. When measured against the training data, the predicted value matches the actual response (as recorded in the original *response* field) with an overall accuracy of 92.82%.

While not quite as accurate as the best of the three individual models in this case (92.86% for C51), the difference is too small to be meaningful. In general terms, an ensembled model will typically be more likely to perform well when applied to datasets other than the training data.

Figure 8. Analysis of the three ensembled models



Correct	12,534	92.82%
Wrong	970	7.18%
Total	13,504	