IDENTIFY CREDIT WORTHINESS – GERMAN CREDIT DATASET - DECISION TREES WITH RAPIDMINER

Preparing the data for modelling

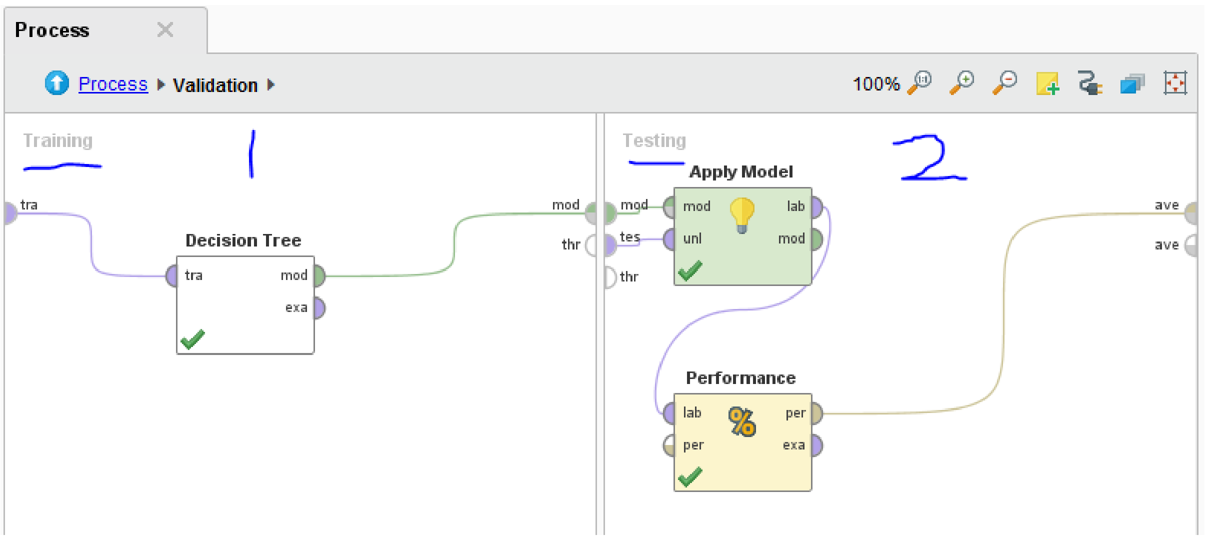
Before applying any model to the German Credit data in its raw form the data had to be tidied up, cleaned and a Role had to be set for the model to use. The following operators were used in order to complete these tasks:

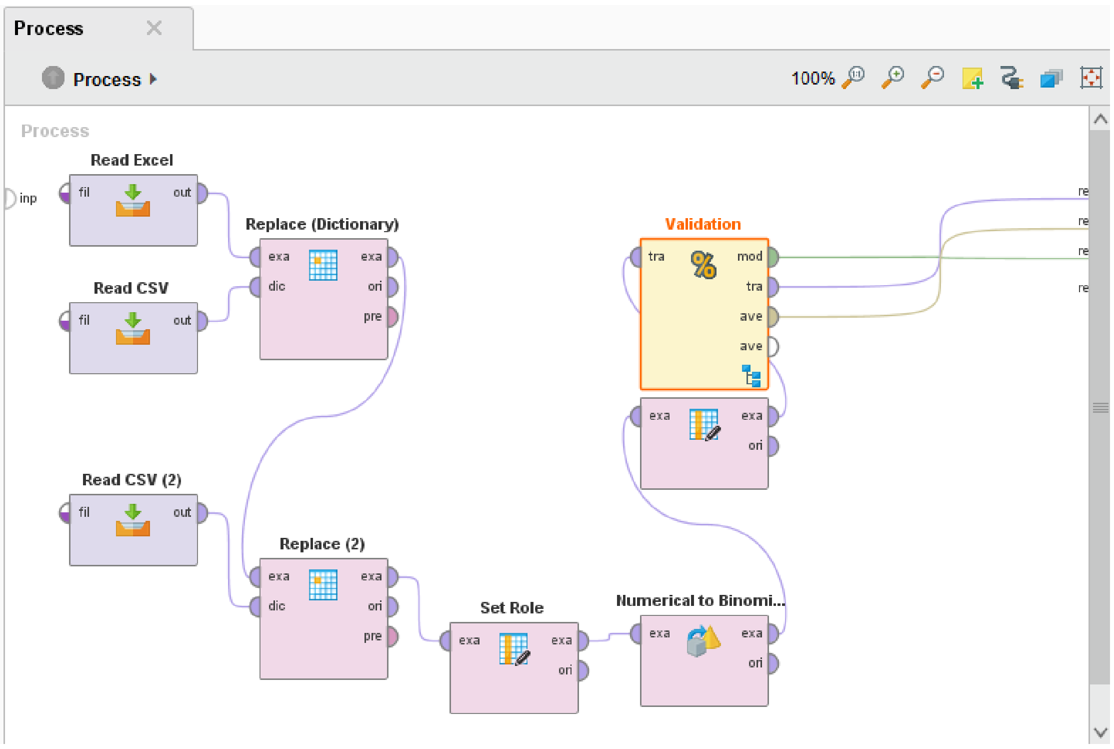
- Replace – to replace the original value with updated attributes
- Set Role – used to set the role to the "Credit Rating" column
- Numerical to Binomial – used to change the Credit Rating to binomial
- Rename – used to change "Credit Rating" to "Credit Rating = Good"

To apply a **Decision Tree** model to the dataset the **Split Validation** operator was used. This model performs a simple validation and classification process. The model separates the data into 2 different sets and evaluates each one: a training set and a testing set.

1. Learning/Training set: In this first step the data is analysed and uses the label attribute (in this case Credit Rating) as the basis for the model to learn from.

2. Testing set: In this step data is selected at random from the whole dataset. Here an evaluation takes place of the accuracy of the learnings from the training step.





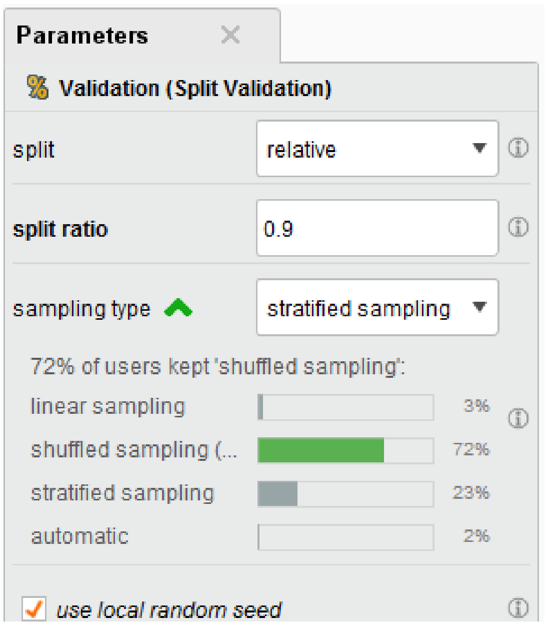
This model also allows you to use various split ratios when applying the model, for example 0.9 training and 0.1 testing or 0.7 training and 0.3 testing.

Sampling type

RapidMiner offers 3 main types of sampling for building the subsets. They are as follows:

1. **Linear sampling**: Linear sampling simply divides the data into partitions (training vs. testing) without changing the order of the instances i.e. subsets with consecutive examples are created.
2. **Shuffled sampling**: Shuffled sampling builds random subsets of a dataset. Instances are chosen randomly for making subsets.
3. **Stratified sampling**: Stratified sampling builds random subsets and ensures that the class distribution in the subsets is the same as in the whole dataset. For example, in the case of a binominal classification, stratified sampling builds random subsets such that each subset contains roughly the same proportions of the two values of class labels.

RapidMiner's' wisdom of crowds suggests that 72% of users use the Shuffled sampling option here. I experimented with each sample type and chose the one that produced the best accuracy.


Parameters

Validation (Split Validation)

split	relative
split ratio	0.9
sampling type	stratified sampling

72% of users kept 'shuffled sampling':

linear sampling	▯	3%
shuffled sampling (...	🟩	72%
stratified sampling	▨	23%
automatic		2%

✓ *use local random seed*

RapidMiner's' wisdom of crowds suggests that 45% of users kept Automatic setting for sampling type. Again, I experimented with each sample type and chose the one that produced the best accuracy.

Alternative Validation Method – Cross Validation

As an alternative I chose the **Cross-Validation** model and applied it to the same dataset to see what results I could achieve. I used 10 as the value of k (number of folds) in my testing.

Test Results

To see the full list of tests I carried out please see the included Excel file "Rapidminer Decision Tree tests". I have listed the best 4 models I achieved through my testing as follows, the results include the test setup and confusion matrix and accuracy score:

DT tests	**SPLIT**	**Sampling Type**	**Criterion**
Test 8	0.7 vs 0.3	Shuffled Sampling	gini_index

CONFUSION MATRIX

Accuracy: 71.00%

	true false/Bad	true true/Good	class precision
pred. false/Bad	58	52	52.73%
pred. true/Good	35	155	81.58%
class recall	62.37%	74.88%	

DT tests	**SPLIT**	**Sampling Type**	**Criterion**
Test 12	0.6 vs 0.4	Shuffled Sampling	information_gain

CONFUSION MATRIX

Accuracy: 70.50%			
	true false/Bad	true true/Good	class precision
pred. false/Bad	51	48	51.52%
pred. true/Good	70	231	76.74%
class recall	42.15%	82.80%	

DT tests	SPLIT	Sampling Type	Criterion
Test 17	0.9 vs 0.1	Shuffled	gini_index

CONFUSION MATRIX

Accuracy: 71.00%			
	true false/Bad	true true/Good	class precision
pred. false/Bad	15	9	62.50%
pred. true/Good	20	56	73.68%
class recall	42.86%	86.15%	

Test Number	Validation	Sampling Type	Criterion
Test 4	10 Cross	Shuffled Sampling	gini_index

CONFUSION MATRIX

Accuracy: 69.60%, +/- 4.61%			
	true false/Bad	true true/Good	class precision
pred. false/Bad	81	85	48.80%
pred. true/Good	219	615	73.74%
class recall	27.00%	87.86%	

Evaluation and findings

As you can see from the 4 test results listed above they all return a similar Accuracy score somewhere between 69.60 to 71%. It would be easy to settle for the highest Accuracy score as this tells you how often the model/classifier is correct. However in my tests I was also looking at the Type I and Type II errors, specifically the False Class scores as afterall in the real world wouldn't a lender want to know who would default on a loan more than who wouldn't default!

So overall I would choose **Test 8** as the most efficient model for this task of predicting Credit Worthiness. The False/Defaulter Class (determing bad credit ratings) score of 62.37% is significantly higher than the other tests I ran and it also has a respectable 74.88% score for the True/Non-Defaulter Class (determing good credit ratings) .

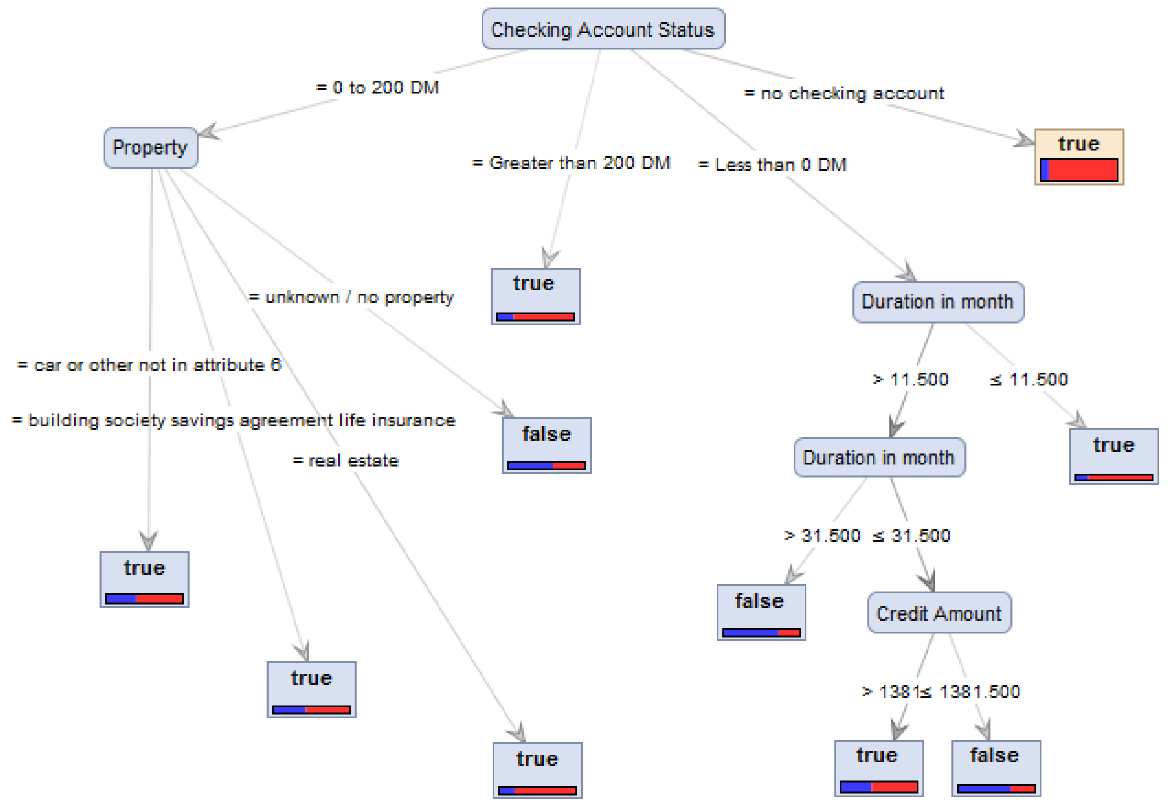
DT tests	**SPLIT**	**Sampling Type**	**Criterion**
Test 8	0.7 vs 0.3	Shuffled Sampling	gini_index

CONFUSION MATRIX

Accuracy: 71.00%

	true false/Bad	true true/Good	class precision
pred. false/Bad	58	52	52.73%
pred. true/Good	35	155	81.58%
class recall	**62.37%**	**74.88%**	

The Decision Tree



Checking Account Status is the root node that was produced by the Decision tree for Test 8. It classified 100% of the data, all 1000 rows. You can then easily read the leaf results, for example if the Checking Account Status was = no checking account then there is a 88% chance of them having a True score or good credit rating. Similarly if the Checking Account Status was = Greater than 200 DM then there is a 78% chance of them having a good credit rating. However if the Checking Account Status = 0 to 200DM then the Property parameter needs to be considered. Similarly if the Checking Account status = Less than 0 DM the Duration in month needs to be considered.

For the various tests I ran I found that the most common Root Node that was produced was the Checking Account Status, there were only 2 other nodes that were produced, namely Saving Account/Bonds and Credit Amount.