Model Fitting of the South German Credit Dataset

Gregory Pollard

April 16, 2021

1 Model Fitting Goals

1.1 Specific Aims

We'd like to find the optimal type of classifier and an appropriate model (that provides the greatest degree of accuracy) for each of the following goals:

- Model A should classify 'good' and 'bad' customers as accurately as possible with no constraints applied.
- Model B should accept at least 85% of all 'good' customers and reject as many 'bad' customers as possible.
- Model C should classify as many people as possible to be 'good' customers, but only if at least 98% of 'bad' customers have been correclty identified.

1.2 Methodology & Variables to Consider

From previous knowledge, we know that *personal_status_sex* and *people_liable* are useless variables; in addition we know that the following pairs of variables should not be used in the same model together unless we add an interaction term:

(*telephone* & *job*), (*telephone* & *number_credits*), (*duration* & *amount*), (*property* & *housing*).

To forward these ideas, the *housing* and *employment_duration* variables were seen to have a small amount of erroneous entries and thus may not be included in models. Our final list of variables to consider are therefore:

status, amount, credit_history, number_credits, telephone, property, purpose, savings, installment_rate, other_debtors, present_residence, other_installment_plans, foreign_worker.

2 Fitting A Decision Tree

2.1 Variable Selection

When building a decision tree (DT), we are **only** concerned with variables that **directly** influence the response variable, each covariate helps us to make rules on how to segment the data into groups (or leaves), the relationships between them do not effect the tree construction. Thus, we can omit all variables that do **not** cause a statistically significantly effect on *credit_risk*; from our initial list described in section 1 and previous χ^2 tests, these are *installment_rate, present_residence,* and *number_credits.* In addition, we should use the same exact set of variables in order to construct models A, B and C. This is because the relationships and importance of the variables in relation to *credit_risk* and to each other doen't change depending on what our goals are. We should note that our final models may include different variables due to the constraints and complexity parameters (CP) applied in order to limit model complexity to achieve these goals.

2.2 Construction & Visualisation of Trees

It's important to note before we perform any tree construction that we will be utilising the Gini index as our misclassification method instead of information entropy. This is because the only variables we will be using are incredibly impactful on credit risk and so the errors will be low anyway, and (perhaps a more important point to make is that) the range of entropy lies between 0 and 1, whereas the range of Gini impurity lies between 0 and 0.5. Hence, the Gini impurity measure may be more beneficial when identifying the best models.

2.2.1 Model A

To find the best overall DT for classifying 'good' and 'bad' customers we need to evaluate the cost of increasing tree size and its effect on the misclassification error so that we do not over-fit our model. A good approach would be to select the CP for pruning the tree which gives the lowest size such that the K-fold cross validation [1] (CV) error is below 1-SD of the overall relative K-fold CV error. It is also important to use the 1-SD rule with respect to **variable importance** when constructing a model; the 1-SD rule tends to isolate the most influential variables by design, reducing the chance of over-fitting. We use cross validation in this way in order to utilise the full extent of our data, in doing so, all records are both used for testing **and** training, the average of the errors can then be taken and a generalised error can be calculated. To do this, we simply construct a complexity table of the **full tree** that involves **all** variables. Leading from this, as an extra step to aid in reducing our model size, we employ the 1-SD rule to find the first relative CV error that lies under 1 standard deviation of the overall relative K-fold CV error. We need to also decide on the number of folds K, it should be a **divisor** of our sample size (1000) so we have folds of equal size, and it shouldn't be too large so that a high number of fold combinations are possible (thus allowing the iterations to have more diversity). A fold number of 10 should be fairly **unbiased**. We can now plot the relative errors an additive model of all variables previously stated would give us, however we receive different values each time due to the nature of the **randomised** folds. The two main plots that were produced over many iterations of this method are displayed in Figure 1.

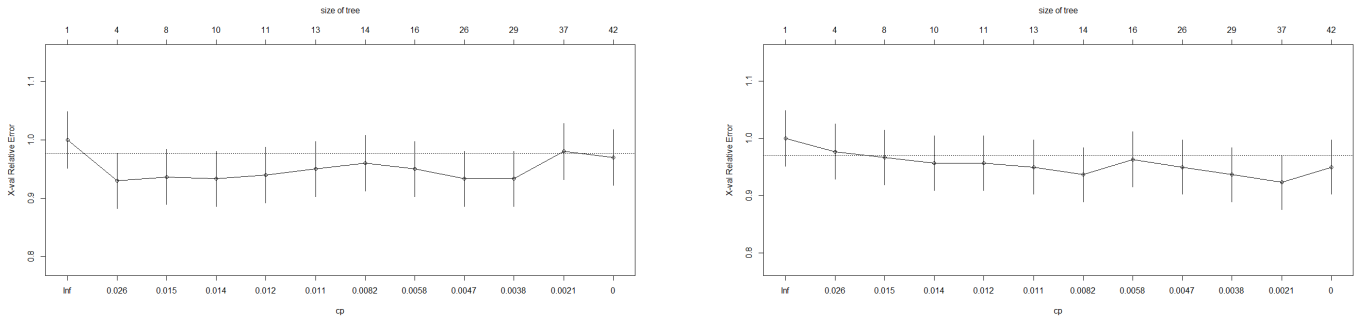
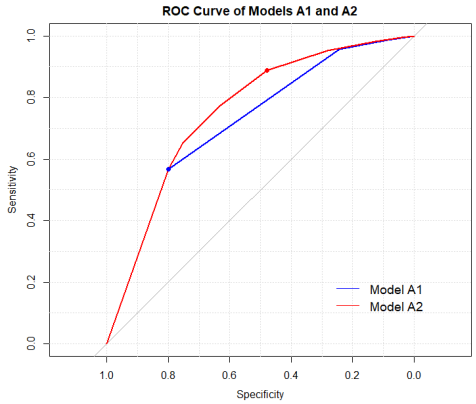


Figure 1: The relative 10-fold CV errors of each tree size compared to the SD of the overall relative 10-fold CV error. After plotting this many times, the two figures above were seen most prominently.

The CP values we receive from these plots are 0.01556 and 0.0150 respectively, giving us two candidate trees of size 4 and 8 (which we will denote as A1 and A2). We can asses the performance of these models with ROC curves; the area under each curve will show us how effective each model is at **maximising sensitivity and specificity** with various probability thresholds. After deciding between A1 and A2, we can find the optimal threshold by observing the point along its ROC curve that is furthest from the diagonal line. The figures below show the ROC curves for both models, as well as the truth tables at the optimal thresholds.



Model A1	Prediction: 'bad'	Prediction: 'good'
True: 'bad'	226	74
True: 'good'	243	457

Figure 2: Truth table for A1 at a threshold of 0.839759.

Model A2/B	Prediction: 'bad'	Prediction: 'good'
True: 'bad'	144	156
True: 'good'	78	622

Figure 3: Truth table for A2 at a threshold of 0.669052.

Figure 4: The coloured dots are the locations of the furthest point from the diagonal line for the set of models A1 and A2, this diagonal line is the ROC curve produced by a model that simply assigns new points randomly.

[1] All cross validation will utilise an 80% / 20% split for all training and testing sets respectively throughout this report. This is the industry-standard ratio that balances the validity of model evaluation and training quality.

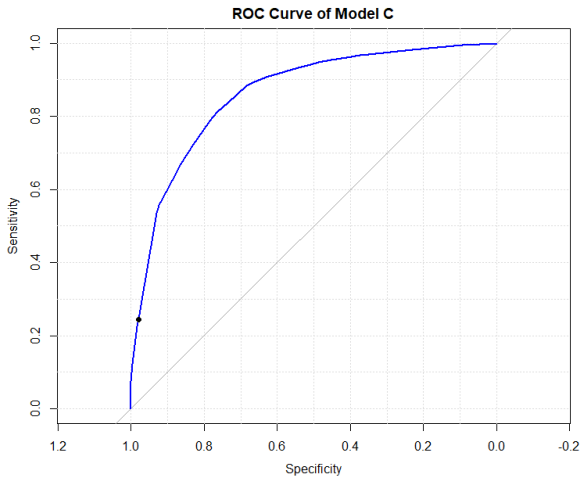
We can safely conclude that the A2 model is superior, this is clear because the curve for A1 is always above the curve for A2 at **all** thresholds; and since both models are **relatively small**, we know over-fitting has not occurred. We can also conclude this because the misclassification rates for our final A1 and A2 models are **31.7%** and **23.4%** respectively, the rate for A2 is lower meaning it performs better on the training data. Although A2 performs adequately and classifies 'good' customers well, it classifies 'bad' customers poorly which is much more dangerous financially for the bank and thus may not be the best model in practice. In reality, we would want a model that classifies 'bad' customers very well.

2.2.2 Model B

For model B, we'd like at least 85% of all 'good' customers to be correctly identified while also making our model as accurate as possible. We can simply use the same approach as we did for model A and utilise the **red** (A2) ROC curve to find the set of models produced by thresholds where the **sensitivity** or true positive rate is above 0.85 (since the A2 model were better than the A1 model). Luckily for us, the threshold that gives the highest accuracy for A2 under these constraints is the same one we calculated for model A, meaning **our choice for model A and B are the same**. The reason this is true, is because the exact threshold that produces the lowest error rate and still correctly identifies 85% of all 'good' customers is the very same tree seen in Figure 3.

2.2.3 Model C

In order to correctly identify 98% of all 'bad' customers, but also construct a model as accurate as possible, we need to turn our attention to **specificity**. We need a model (once again along the A2 ROC curve) with a threshold at 0.98 specificity or higher that is as accurate as possible. Unfortunately, when we construct a DT in the same fashion as before with these constraints, the only model that can achieve this harsh true negative rate is one that either classifies **all** customers as 'bad', **or has more than 8 leaves**. However, if we construct a tree that is larger than this we run the risk of over-fitting. Nonetheless, if we disregard our methods to control tree size and focus simply on satisfying the constraint, we produce an entirely new tree with **37** leaves, misclassification error of **53.3%**, and truth table as in Figure 5. The ROC curve for this tree is also shown in Figure 6, the AUC is **unsurprisingly** higher than the AUC for models A1 and A2 since we have constructed a more complex tree that would naturally perform well on the training data. Unfortunately, it would perform poorly on new and unseen test data due to its complexity and lack of generalisation, which is precisely why we would prefer a tree constructed using the 1-SD rule (such as models A1 and A2).



Model C	Prediction: 'bad'	Prediction: 'good'
True: 'bad'	294	6
True: 'good'	527	173

Figure 5: Truth table for C at a threshold of 0.9350000.

Figure 6: We can see again that the black dot shows us where along the line we extracted our threshold.

Unsurprisingly, this tree with this threshold performs quite **poorly**. It categorises many 'good' customers to be 'bad', however in reality this might be **beneficial** for the bank since providing a loan to someone who is likely to default is more financially risky than missing out on the opportunity to provide a loan to a customer who will pay back the loan successfully. This is because the loss of the money loaned due to a 'bad' customer defaulting will be **more** than the interest the bank earns from a 'good' customer (since the interest is a small faction of the loan itself).

3 Fitting A GLM

3.1 Variable Selection

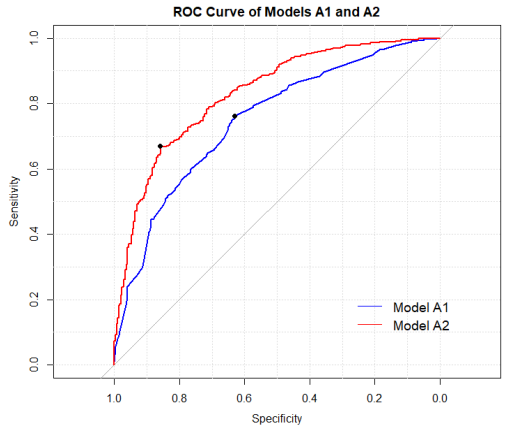
As we did for decision trees, we can select the best subset of our initial list of variables to construct a model with, this time however, our final choice of model will depend on the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), and log-likelihoods (LL). This is because we can't use the same methods to assess optimal model size as we did before due to how decision trees are designed. Instead we do this with the AIC, BIC and LL which balance model accuracy with model size (however we can still utilise ROC curves and AUC values to compare models). To begin our variable selection, we need to decide on **how** to find the best subset of variables from our initial list as in section 1. We could use the '**best subset selection**' method which compares all **possible** models of size $k = 1, 2, \ldots, 13$ and outputs the one with the best evaluation metrics. This is **incredibly computationally expensive** since this approach would consider $2^{13} = 8192$ models; it's also likely that one of these models would perform **too** well on the training data, so much in fact that over-fitting is a possibility.

Due to these observations, it makes sense to use '**stepwise selection**' instead, a method of model selection that can handle more explanatory variables than its counterpart, but also generalises our final model because is doesn't consider as many candidates. Since **forward** stepwise selection starts with the null model and iteratively increases our model size, and 13 variables is still a lot to consider, it would seem the more intuitive approach to take than **backward** selection. Because of this, we'll be using the AIC and AUC to assess model performance since BIC is more generally used for backward stepwise selection. We'll also be using **10-fold CV and the 1-SD rule** with our forward stepwise selection to aid us in deciding a model. Although it introduces randomness to the process, this will help us limit the number of parameters while also **generalising** our model. Our choice of 10 folds is made for the same reasons as pointed out in section 2.

3.2 Construction & Visualisation of Models

3.2.1 Model A

We can utilise the approach as described above to find a GLM that is accurate but is also relatively simple (and thus is not exposed to over-fitting). As an output we get 20 models to choose from. Figure 9 shows the ROC curve for two of these 20 models, one with the highest AUC (A2), and one with the lowest AIC (A1) for **all** probability thresholds. We can also see the truth tables for these models to the right of the plot, these tables were generated with the thresholds found furthest from the diagonal line (just as we did in section 2) giving us the best overall specific model.



Model A1	Prediction: 'bad'	Prediction: 'good'
True: 'bad'	135	165
True: 'good'	139	561

Figure 7: Truth table for A1 at threshold of 0.6062267.

Model A2	Prediction: 'bad'	Prediction: 'good'
True: 'bad'	252	48
True: 'good'	231	469

Figure 8: Truth table for A2 at threshold of 0.7736108.

Figure 9: The black dots are where we've decided our optimal thresholds to be (furthest from the diagonal line as possible). Models A1 and A2 have AIC values of **993.78** and **1061.9** respectively, implying at first glance that A1 balances model size with accuracy to the training data better than model A2.

It's important to decide between these models based not only on their performance, but also their structure. Although model A2 had a higher AUC, it was a much larger model than A1, containing 46 terms! The reason this is so high is due to how many categorical variables we have in our data; for all N states (or levels) of every categorical variable, the model must include N-1 **indicator variables** with their own coefficients to

accommodate. Model A1 simply received a smaller model size after CV and the 1-SD rule were used; (as we can see from Figure 9) the result of this gives us a model that only includes essential variables that are all **statistically significant** (4 variables and an intercept term to be exact), which explains why its curve is lower. In other words, model A2 receives **noise** because it includes so many covariates that aren't actually significant. In fact, 27 terms in model A2 are **not** statistically significant, that's over half!

Another note of interest is that since we used 10-fold CV and the 1-SD rule to generate our models, how did we manage to receive a model with so many terms? This isn't that strange if we remember that **we generated 20 models**, it's not surprising that at least one of them is very large (possibly due to the randomness introduced by CV). In addition, it makes sense that the model with the highest AUC is also the most complex, as this would naturally be the better model at **predicting the training data** (in fact, we can see that A2 is superior than A1 in this regard for **all** thresholds). This is exactly why the misclassification rates for model A1 and A2 are **30.4%** and **27.9%**.

However as mentioned before, it fits the data **too** well due to its complexity and thus would perform poorly when attempting to classify **unknown** data; and so we can't simply use AUC to decide our model like we did for decision trees. In conclusion, model A1 is fairly accurate **if we have no specific goal in mind**, and since it balances model complexity with model accuracy better than A2, it's our preferred GLM candidate for our choice of model A.

3.2.2 Model B

Just like we did for decision trees, we want to use model A's ROC curve to find a threshold furthest away from the diagonal line that gives us a **sensitivity** of at least 0.85 (we use sensitivity since we've labelled the 'bad' customers as our negative values, and 'good' customers as our positive values, just as we did with the decision trees). Travelling along the ROC curve for the GLM A1, the threshold for our desired model is 0.5604393. The truth table for model B can be seen in Figure 10.

Model B	Prediction: 'bad'	Prediction: 'good'
True: 'bad'	139	161
True: 'good'	101	599

Figure 10: Here we can see that around **85.6%** of 'good' customers have been correctly identified. This model has a misclassification rate of **26.2%**.

The problem with this model in practice is that it classifies around **53.7%** of all 'bad' customers **incorrectly**. As discussed earlier, the bank would want this to be a lot lower since the financial loss from misidentifying a 'bad' customer is greater than the loss of misidentifying a 'good' customer. We'll discuss these issues again later when we decide on our final models.

3.2.3 Model C

In order to correctly identify 98% of all 'bad' customers while also accepting as many 'good' customers as possible, we can use the same methods as before but with **specificity**. Once again we can find the point along the curve that is furthest from the diagonal line where the specificity is greater than 0.98. We receive a threshold of 0.9176648, resulting in a GLM that gives the truth tables as described in Figure 11.

Model C	Prediction: 'bad'	Prediction: 'good'
True: 'bad'	295	5
True: 'good'	628	72

Figure 11: Here we can see that around **98.3%** of 'bad' customers have been correctly identified. This model has a misclassification rate of **63.3%**.

Although model C can accurately identify 98% of 'bad' customers, the issue with this model is perhaps that most 'good' customers are misidentified as 'bad'. Most of our erroneous classifications come from when our model attempts to predict 'good' customers correctly.

4 Model Choice & Recommendations

4.1 Variable Importance

A final metric to compare **all** of our models discussed in this report would be through **variable importance**, this will directly shed light on **the most impactful factors that determine repayment behaviour**. Figure 12 details the **relative importance** of each variable for every model in isolation as a **percentage** and rounded for all decision trees. It also gives the p-values that **determine variable significance** for the GLM we constructed as well as the smallest level of significance that each covariate satisfies when performing hypothesis tests for each term.

Relative Importance (%)	DT		
Variable	A	B	C
`status`	47	47	27
`credit_history`	14	14	14
`purpose`	13	13	14
`savings`	12	12	13
`amount`	9	9	19
`property`	5	5	5
`other_debtors`	0	0	3
`foreign_worker`	0	0	2
Misclassification Rate (%)	23.4	23.4	53.3

P-Value	GLM	Sig Level
Covariate	A, B & C	α
`Intercept`	$1.01 \cdot 10^{-11}$	0.001
`status: DM >= 200`	$7.91 \cdot 10^{-14}$	0.001
`status: 0 <= DM < 200`	0.06187	0.1
`status: no account`	0.00632	0.01
`duration`	$1.09 \cdot 10^{-9}$	0.001
Misclassification Rate (%)	A = 30.4, B = 26.2, C = 63.3	

Figure 12: For decision trees, variable importance is effected by the **gain** in accuracy (or change in impurity) provided by the use of the variable in question, and also by how many elements are assigned to each node. For GLMs, the significance of a term is related to how much it impacts the final predictions of the model.

We can see that, through the approach we have used in this report to model building, that **our specific goals can effect our optimal choice of decision tree**. However when selecting a GLM, we can simply find a **singular** and **fairly accurate** model, and then select a threshold accordingly. This is due to the structure of decision trees, and their relationship to thresholds. A decision tree groups data points together and classifies them all with the same prediction based on what the majority is for that group, meaning all predictions are viewed in groups and **not** considered **individually** like in a GLM. The complexity of a tree determines how smooth the ROC curve is too. When a tree is small, lots of thresholds will correspond to the exact same set of predictions, see Figure 4 for a visualisation of this. However, GLMs give the probabilities of each prediction being correct **individually** and not in groups, meaning that changing your choice of threshold slightly is likely to effect at least **some** predictions.

Our decision trees considered 8 variables and (for the most part) concluded that *status, credit_history, purpose,* and *savings* were **relatively** important. Whereas our GLM only consisted of 5 terms, 3 of which are indicator variables for the variable *status* (meaning overall the model only considered 2 variables from our initial list). A good point to mention is that most of the terms in our GLM were all incredibly significant, this means that all terms (excluding `status: 0 <= DM < 200`) were very impactful on our prediction of the *credit_risk* variable, possibly more significant in fact than any of our decision trees.

4.2 Model Choice

Because of these reasons, and since the misclassification error for our GLM is **only** 7% and 2.8% higher for models A and B respectively, we can utilise **Occam's razor** here and pick the simpler GLM to describe new data points. This makes sense because most of the variables in our GLM are important, it is the smaller model overall, and performs better. In addition to these observations, our choice for model C should also be through our GLM approach to model building. This is because, although the misclassification error for our GLM is 10% higher, the decision tree for model C is very large, boasting 37 leaf nodes! With such a complicated model, it's very clear **that our GLM is indeed the more parsimonious model**. In fact, it's very clear to see from these points that the more **parsimonious** choice for all 3 models is our GLM since it has great explanatory predictive power while also keeping the number of parameters to a minimum.