1. Introduction

We are looking at building a model to predict "good" and "bad" customers given a set of features $X = [X_1, X_2, \ldots, X_p]$, in our case a "bad" customer is a positive classification. The goals of the bank concern two different metrics (1) true positive rate (TPR = TP/(TP+FN)) which is also known as sensitivity; what percentage of the "bad" customers did the model correctly identify and (2) true negative rate (TNR = TN/(FP+TN)) which is also known as specificity; what percentage of the "good" customers did the model correctly identify. The bank is looking into two goals:

1. Building a model that has a TPR of at least 0.98 which maximises the TNR
2. Building a model that has a TNR of at least 0.85 which maximises the TPR

Both goals involve models that have two parts, a statistical component, and a decision component. In the statistical component we output a probability of being "bad" and in the decision component, we take this probability and decide what to do with it. It is unwise to try and finalise statistical and decision components at the same time, instead, we find the optimal statistical component to some criteria and then adjust the decision threshold afterwards to meet our goals.

Variable encoding

We choose to encode all our categorical variables using a one-hot encoding method, even if they are ordinal. This is because whilst the ordinal variables may be monotonic in probability of "bad" they are not linear; for example, number of credits has the categories "1", "2-3" and "4-5", therefore, there is no constant increase between categories.

Choosing a performance metric

To measure performance, we need to choose a metric. Two possible candidates are misclassification rate and area under the receiver operating characteristic curve (AUC), these are very different metrics, and we will explain what they mean before we explain which one is most appropriate given the task at hand.

Misclassification rate assumes a decision threshold of 0.5 to make calculations about the performance of a given classifier. This means that if our statistical model outputs a probability greater than 0.5, then we predict positive, if not we predict negative. Given our very specific objectives relating to achieving relatively high TPR and TNR it is almost certain that we will not restrict ourselves to using a decision threshold of 0.5 in our models; for example, the decision threshold for objective 1 may be below 0.5 and the decision threshold for objective 2 may be above 0.5 to achieve the goals.

AUC on the other hand considers many alternative decision thresholds or trade-offs between TNR and the false positive rate (FPR; TP/(TP+FP)); in this sense it is said to be threshold-invariant. Whilst AUC does not directly measure the specificity vs sensitivity trade-off (it uses FPR instead of TNR) it is still a summary of the trade-off because it considers the consequences of increasing and reducing the threshold.

We use AUC because it allows us to find candidate models that have leeway to adjust the decision threshold to achieve our objectives.

Cross Validation

In this coursework we use tenfold cross-validation on a training set of 80% of the data to create an unbiased estimate of performance, tune our model parameters and set our decision thresholds. We leave the remaining 20% as a holdout set for the purposes of model assessment, which is to say estimating the prediction error of a selected model on new data. The 80/20 split was created by randomly shuffling the data subject to stratification constraints, this ensured that the training and test set both had a ratio of roughly 3 bad to 7 good. This ensures that each split is representative of the bigger picture which means that models created using this data will generalise well to the realities facing the bank.

We use tenfold cross-validation as this will lead to less variability AUC across our folds compared to fivefold cross-validation, this is preferable when using the one-standard-error rule which we will explain later. In each of our folds, the training size will be 720 samples and the test size will be 80 samples.

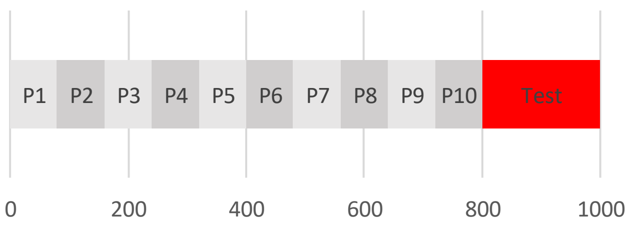

Figure 1.1. *An illustration of our test train split, showing the size of a partition for cross validation and the relative size of our test set.*

The form of cross validation we will use is stratified cross validation, in stratified cross validation each fold keeps the same ratio of classes as seen in the whole of the training data, this means that each fold is more representative of the sample which gives a more general estimation of performance, rather than data changing significantly between folds thus creating a high variance in performance. The parameters inferred from analysis of such folds should generalise well to the underlying class densities.

One-standard-error rule

Across our cross validations we will be using the "one-standard-error" rule (Friedman, Hastie & Tibshirani, 2001), thus it is worth explaining it in advance. When we evaluate a given model across our ten folds, we get ten measurements of AUC, we then take the mean, this gives us an indication of average performance. However, this mean value comes with some variance, so we also measure the standard error, which is the standard deviation of the ten AUC scores, divided by the square root of ten: the number of observations of the AUC score. Once we have done this, we look for the least complex model that is within one standard error of the best AUC score (highest). We measure complexity in terms of number of variables for logistic regression and tree size (number of leaf nodes) for decision trees.

As such, when looking for a trade-off between complexity and cross-validation performance, the one-standard error rule acknowledges that our mean AUC score is estimated with error which makes it a more conservative method of model selection.

Choice of decision threshold

We will also use cross-validation to decide proposed decision thresholds to take into the model assessment stage, so it is important here to describe how and why we do this. The reason we cross-validate our decision threshold is so that we don't overfit our training data; we want to create decision thresholds that will provide satisfactory results on un-observed data.

Once we have created a statistical model, we use the following algorithm for each objective:

Algorithm 1.1 Decision threshold selection

```
• For $k = 1, 2, \ldots, 10$ (where $k$ is the fold in our cross-validation)
    1. Fit a statistical model on the training partitions and output a vector of $\Pr(Class = 'Bad')$ for the validation partition
    2. Comparing probabilities with the class labels, find the set of all thresholds which meet the specific goal of the bank (for objective 1 it is 98% TPR) by searching across the interval [0,1] with a step of 0.005
    3. Out of this set, choose the threshold which maximises the other metric (for objective 1 this is TNR), then add this threshold to a list
• Take the mean of the list of thresholds
```

By taking the mean, we avoid overfitting any fold of the training data and we generate a general estimate of the required threshold to achieve optimal results for the bank.

2. Decision Tree Model

Tree-based classification methods partition the feature space into a set of rectangles and then fit a simple model in each one. In a node m representing region R_m with N_m observations, the probability of the class being k is the class density of the datapoints that end up at that node where $I(x)$ is the indicator function.

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

Choice of impurity measure

To build a new split from R_t to $R_{t,1}$ and $R_{t,2}$, we have the criteria that we must maximise the reduction of impurity. Two measures of impurity which we will evaluate are the Gini index and cross-entropy, these are preferable to misclassification rate as they are differentiable and more sensitive to changes in node probabilities.

Gini index: $\sum_{k=0}^{1} \hat{p}_{mk}(1 - \hat{p}_{mk})$

Cross-entropy: $-\sum_{k=0}^{1} \hat{p}_{mk} \log(\hat{p}_{mk})$

The Gini index computes the expected misclassification error after predicting the majority class with probability $\hat{p}_{mk}$ at node m. The cross-entropy quantifies how unpredictable a random variable is.

How large should the tree grow?

A large tree might overfit the data, whilst a small tree might not capture the important structure. It is of fundamental importance to ensure our tree is not too complex to avoid overfitting our data.

Whilst we intend on performing post pruning, which is to say reducing the size of the tree after it has been built, there are also parameters we can use to restrict the growth of our tree in the first instance. However, it is essential that we avoid the horizon effect (Russell & Norvig, 2002), restricting a seemingly useless growth at a certain stage to the detriment of preventing important growth further down the tree. For example, we could stop growth after some arbitrary maximum number of leaf nodes is achieved, but with just 3 more leaves there could be a split of immense value, this restricts the possible subtrees our post pruning routine can consider, to the detriment of finding a very well performing subtree that may not be that large overall.

One parameter, which we view as important which does not suffer from the horizon effect is minimum samples split. This parameter restricts nodes that contain a small number of samples from growing leaves. By default, this parameter is set to 2 by the Scikit Learn Decision Tree Classifier, this is bad because this represents less than 1% of our training data for each fold (n=720); theoretically a split could occur that is very specific to a very small part of the training data thus overfitting. Minimum samples split is not subject to the horizon effect because we state a priori that no split that small will create value for our classifier because it is simply overfitting.

By increasing the minimum samples split, we ensure that we do not make any decisions that are extremely specific to our training data.

Cost-complexity pruning

Once we have created tree T we then prune it using cost-complexity pruning. Cost-complexity pruning analyses the average purity of the leaves of the tree and then compares this with the size of the tree (in number of leaf nodes) to create a measure of cost. We change the weight that is given to the size of the tree by changing the tuning the parameter $\alpha \geq 0$, this governs the trade-off between tree size and goodness of fit to the data.

Large values of α result in smaller trees and small values of α result in larger trees. For each value of α it can show that there is a unique smallest subtree T_α of the tree T which minimises $C_\alpha(T)$.

Grid search

To perform cross validation, we create the following parameter grid:

- 'ccp_alpha': 25 evenly spaced floats over the interval [0, 0.01]; the value of α
- 'criterion': ['gini', 'entropy']; the criterion to build the tree
- 'min_samples_split': 10 evenly spaced floats over the interval [0.01,0.1]; the minimum samples split – 0.01 means 1% of the training size is required to split a node

We consider the range of α because in preliminary analysis 0 translated to very large trees (around 100 leaf nodes) and 0.01 translates to very small trees (around 2 leaf nodes), values outside of this range would not be appropriate candidates for creating an interpretable yet suitably complex model with strong prediction performance.

Parameter results

Using the one-standard-error rule which we explained in the introduction, we are offered a model with Gini criterion, 1% minimum sample split and alpha 0.007; this yielded a mean size of 2 leaf nodes across the folds and a mean AUC of 0.70 with a standard error of 0.018. This tree, despite being a suitable candidate according to the one-standard-error rule is overly simplistic. Therefore, we look across our grid search for a model with an average tree size larger than 5, that is within one standard error of the best AUC and select a model with alpha 0.006, Gini criterion and minimum samples split 0.1, this model had an average size of 6.6.

Comparing results between trees constructed using entropy and Gini criteria, Gini trees had a marginally higher mean AUC across the folds (0.68 vs 0.65) and a smaller mean size of trees (16 versus 30).

The effect of our minimum samples split parameter was to decrease the size of trees when it was increased, 1% produced a mean size of 45, whereas 10% produced a mean size of 14.

Decision thresholding

For objective 1, we estimate a threshold of 0.0005 using the method described in the introduction, on our cross validation this results in a mean TNR of 0.002% and a TPR of 100%.

For objective 2, we estimate a threshold of 0.433, on our cross validation this results in a mean TNR of 87% and a TPR of 32%.

These two models; each with the same statistical component but different decision components, are the models we take forward for a final comparison after we have created a logistic regression model which we can compare against our tree model.

3. Logistic Regression Model

Logistic regression models the log-odds of being “bad” as a linear combination of the variables in our dataset, it then converts these log-odds to probabilities using the logistic function. To find the coefficients for the model, logistic regression uses maximum likelihood.

Subset selection

Unlike decision trees which offer no guarantee that all variables in the training data will have some presence in the final estimated tree, all variables shown to a logistic regression model will receive a non-zero coefficient.

Performing a logistic regression on all 69 of our variables and an intercept results in an AUC of 0.77 and a 0.021.

We may want to reduce the variables included for two reasons, firstly we want to avoid overfitting our data, this will improve out-of-sample performance and secondly, we would like a more interpretable model, this is easier to do when we focus on a smaller subset that exhibits the strongest effects, rather than all 69 features we have available.

Our number of variables dictates that searching through all possible subsets will be extremely computationally expensive. Instead, we can use stepwise selection, which is more constrained, this means that it will exhibit lower variance, but we accept the potential of increased bias as a trade-off for computation. Forward stepwise selection starts with the intercept and then sequentially adds the predictor that most improves the fit into the model. We measure fit with cross-entropy loss, which decreases as fit increases.

We use a cross-validation form of forward-stepwise selection which is shown in algorithm 3.1. we use cross-validation to choose a final subset, this is preferable as we can use the one-standard-error rule.

Algorithm 3.1 Forward stepwise feature selection for logistic regression

```
• $S_0$ is the subset containing only the intercept
• For $k = 1, 2, \ldots, p$ (where $p$ is the number of variables)
    1. For each candidate variable, create a proposed subset consisting of our existing subset $S_{k-1}$ and the candidate variable, for this proposed subset calculate the cross-entropy loss on our training data
    2. Set $S_k$ to the proposed subset with the lowest cross-entropy loss
• For each subset $S_k$, evaluate performance using average AUC across the folds
```

Using algorithm 3.1 on our training data leaves us with the path shown in figure 3.1. The one-standard-error rule selects a model with 2 variables and an intercept with mean AUC 0.74 with a standard error of 0.017.

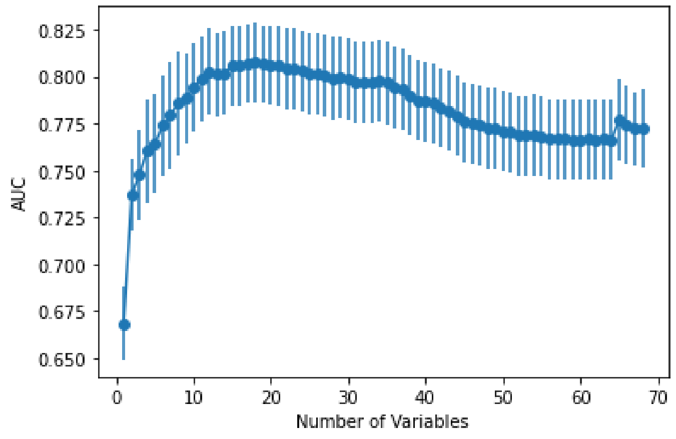

Figure 3.1 *Forward stepwise selection of variables for logistic regression. Points indicate the mean AUC of the subset on the cross validation with error bars indicating* $\pm$*1 standard error. The left-most point contains the subset of 1 variable (status_... >= 200 DM / salary for at least 1 year) and the intercept, then as the algorithm continues, variables are added.*

An interesting thing to see is that at a certain point around 20 variables, the AUC starts to drop, which is an argument for subset selection; some variables serve to decrease cross-validation performance because they are not informative. These models had a lower cross-entropy loss than the previous models (as shown by figure 3.2) which goes to show that the quality of fit on the training data is not always a good indicator of general performance.

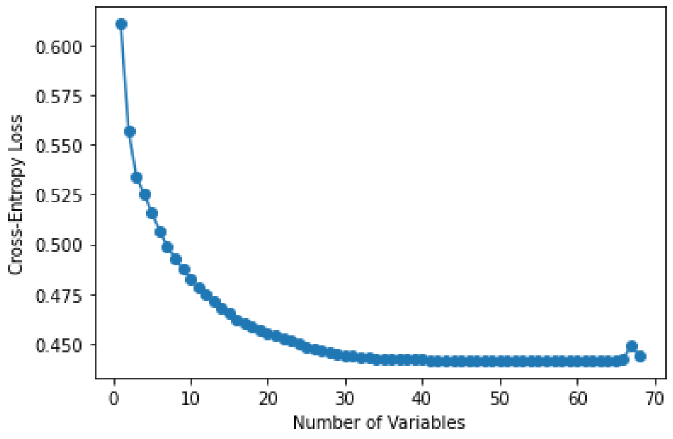

Figure 3.2 *Cross-entropy loss; the metric used to decide what variables to add to the subset plotted again the number of variables, as we can see fit generally increases as variables are added.*

We can also use a similar algorithm which works in reverse. We start with all variables and the intercept, then at each step we remove the variable whose removal would produce the lowest cross-entropy loss out of all possible removals. This method results in the selection of a subset of 6 variables with mean AUC 0.77 and a standard error of 0.029. The intercept remains throughout all subsets.

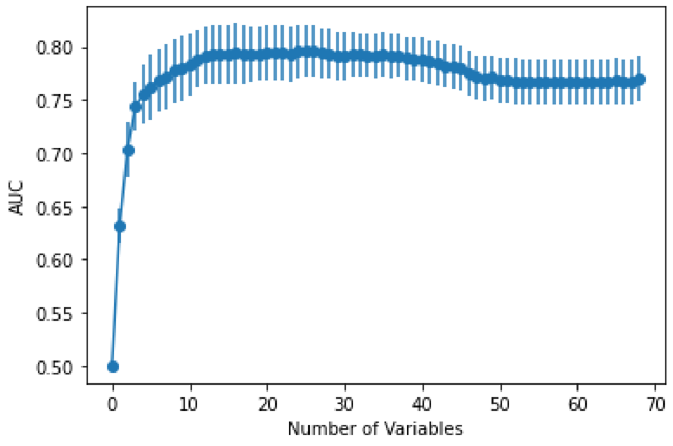

Figure 3.3 *Backward stepwise selection of variables for logistic regression, the same as Figure 3.1 only the journey of the algorithm can be seen from right to left.*

We choose the subset obtained through backward stepwise selection over the subset obtained through forward stepwise selection because it had a higher AUC score and was not too simplistic, 2 variables is too little. Furthermore, the backwards method selected all the variables in the forward subset. This subset obtained the same AUC average as the full model, with a significant reduction in complexity so we choose to take this subset forward.

Feature transformation

With our feature selection completed, we can now use basis functions on our features and compare models, however we are limited by the fact we only have one continuous variable (duration) in our subset. We can transform this by squaring it and cubing it to create two new features. This increases the complexity of our model and evaluating it across our cross-validation we observe an AUC score of 0.771. As this is a more complex model with a negligible increase in performance, we choose to keep our final model as a logistic regression on the 6 variables with no transforms.

Decision thresholding

For objective 1, we estimate a threshold of 0.077, on our cross validation this results in a mean TNR of 17.5% and a TPR of 97%.

For objective 2, we estimate a threshold of 0.43, on our cross validation this results in a mean TNR of 83% and a TPR of 53%.

4. Model Assessment

How well do our models meet the objectives when used on the test set?

Now that we have decided parameters, we take two statistical models, each with two different decision components to the final stage where we assess them in relation to the goals specified.

1. A logistic regression model which uses 6 variables and an intercept
   a. For objective 1 use a decision threshold of 0.077
   b. For objective 2 use a decision threshold of 0.43
2. A decision tree model with α = 0.006, Gini criterion, minimum sample split 10%
   a. For objective 1 use a decision threshold of 0.0005
   b. For objective 2 use a decision threshold of 0.433

Using these two models trained on the entire training set and tested on the test set we obtain the following results.

- Objective 1:
  a. Logistic regression (a): 100% TPR, 0% TNR
  b. Decision tree (a): 100% TPR, 0% TNR
- Objective 2
  a. Logistic regression (b): 76% TNR, 52% TPR
  b. Decision tree (b): 75% TNR, 50% TPR

Final selection

With equal results for either model for objective 1, we select the decision tree model as it is more interpretable (we will discuss this later in our conclusion). For objective 2 we use the logistic regression model as it had a higher TNR and TPR.

If the objectives weren't explicit, which model would you choose

The bank needs to make a return, if it rejects everyone it will have no income, if it accepts everyone, the losses caused by "bad" customers could render it bankrupt.

Both models for objective 1 would probably not work well for the bank, as the results on the test data shows that all customers were rejected, whilst the exact same result is not guaranteed for other data, there is a high probability this will happen again. The bank would have no defaults, but it would also have negligible or zero interest income, thus it would have to shut down.

The logistic regression model from objective 2 would be my choice. Considering the test set was 200 samples large and our TPR was 52% we would accept 29 bad customers and with a TNR of 76% we would have accepted 106 good customers. In this case the bad customers would have to cost the bank more than 3.6 times as much as good customer benefits the bank for the bank to make a loss.

The bank takes some risk as it knows it will most likely accept some bad customers, however it needs to take this risk to accept good customers. Therefore, I believe this model is the most appropriate for the task at hand, it is not perfect, but it represents an opportunity for the bank to make a return.

How to use this model

To use this model, the bank needs to extract the information of customers, encode them numerically and then multiply the features by the coefficients estimated by the model on the entire training set. This will create a log-odds figure, which we can use the logistic function to turn into a probability. The bank will then combine this probability with the threshold (0.43) to classify the customers; anyone with a probability of default of over 0.43 gets classified as "bad", anyone below is "good".

How and which are the most important variables that determine repayment behaviour?

Variable importance differs depending on objective because we decided on a different statistical model for each objective; the logistic regression model for objective 1 and the decision tree for objective 2. Variable importance does not vary with decision threshold.

To measure variable importance for our logistic regression model, we look at the order of our stepwise selection process. In the case of backward selection, the variables that most improved the fit were discarded last, this can be said to be a measurement of importance, this is indicated in figure 4.1 below. The "no checking account" category dummy for status was the most important variable.

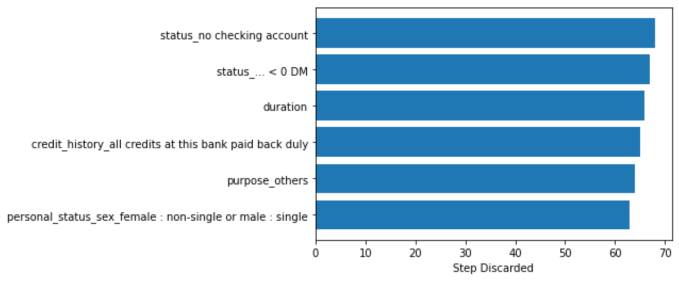

Figure 4.1 *The variables in the logistic regression and the step at which they were discarded in the stepwise selection, high steps indicate variable importance*

To measure feature importance for our decision tree model, we can look at its structure, this is shown in figure 4.2 below. Our decision tree makes use of only two variables, a status dummy, and the continuous variable duration. The most important variable is the status dummy for the category ">= 200 DM / salary for at least 1 year" as it is the highest up the tree and automatically creates a leaf node. The other variable duration splits based on whether the loan has duration greater than or equal to 20.5.

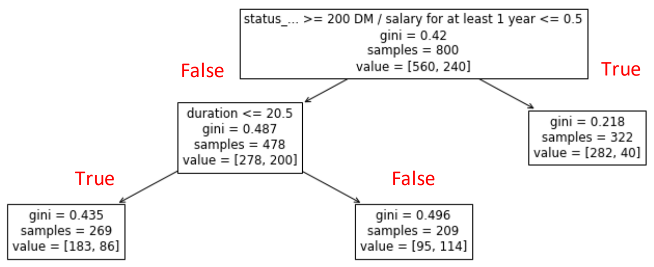

Figure 4.2 *The tree structure created after training on the training set with our parameters*

As shown by our analysis, both models valued the variable duration, however all other variables used in the models had no importance in the other model, for example the "others" purpose dummy was included in the logistic regression subset but didn't feature in the decision tree.

5. Conclusion

Key findings

For objective 1, the bank should use a decision tree model with α = 0.006, Gini criterion and a minimum sample split of 10%, the bank should then combine this with a decision threshold of 0.0005.

For objective 2, the bank should use a logistic regression model with 6 variables, an intercept, and a decision threshold of 0.43.

Logistic regression and decision trees: relative merits and disadvantages

Decision trees for this task were less computationally expensive because we did not have to perform subset selection. Subset selection in this instance was computationally expensive because we had 69 different variables after feature pre-processing.

Our decision tree model was also arguably more interpretable to a finance audience who may not be well versed in probability theory. For example, many people have never heard of log-odds or the logistic function. At each internal node in the tree, there is a simple rule with a binary response. It is more like a process map, which are common in most businesses. Decision trees may lose their interpretability when they become copiously large, but as discussed, we can restrict their size and our decision tree only had an average of 6.6 leaf nodes across the folds of our cross validation.

The benefit of logistic regression in this context was a more continuous output of probability. The probability output range of decision trees are limited by the number of leaf nodes, if you have 2 leaf nodes then you only have 2 possible probabilities of "bad"; one for each leaf node a data point can end up at. There were 200 samples in our test set, the logistic regression model outputted 125 unique probabilities, the decision tree model outputted 3. This would mean that if the bank used the logistic regression model, it would have considerably more freedom to perform meaningful adjustments to the decision threshold, giving it more options as to how it classifies prospective customers.

6. References

- Friedman, J., Hastie, T. and Tibshirani, R., 2001. The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.
- Russell, S. and Norvig, P., 2002. Artificial intelligence: a modern approach.