Aston University
Birmingham B4 7ET

Data Mining Coursework Outline Solution

Ian T. Nabney

Abstract

In this report we provide a solution to a credit rating problem using data from a German financial institution. the training data consisted of 950 examples with about 70% labelled as a good risk and the remainder as a bad risk. Data exploration and visualisation showed that the predictability of the classes was poor. Attribute selection techniques were applied to the original 20 input variables and two reduced sets, of four and ten attributes were created. A number of machine learning algorithms were applied to the data and experiments to optimise the meta-parameters were performed. The best model was a voting committee of four models, which achieved 76.4% (under ten-fold cross-validation). We also assessed these models (with the same meta-parameter settings) under an unequal cost regime (misclassified bad risks cost five times as much as misclassified good risks) and found that a naive Bayes model trained on the complete dataset performed best, at a cost of 495.
Both models were then applied to an unseen test dataset containing 50 examples. The default rule has an accuracy of 68% and a cost of 34. These models achieved an accuracy of 76% and 31 respectively. The drawbacks of such a small test set are discussed.

1 Introduction

This document presents an outline solution to the coursework. It is not meant to be a complete report, but highlights some of the issues and results that arise when following a systematic data mining process.

The *goal* is to develop models to classify good and bad credit risks under both equal and unequal costs. In the following sections, we report on data exploration, data preprocessing, model building, and results.

2 Data Exploration

The dataset was supplied as an ARFF file ready for use with the Weka data mining toolkit.

Initial exploration of the training dataset showed the following features.

- Approximately 70% of the data belongs to the 'good' class. This is the accuracy of the default classifier.
- There are no missing values.
- There are 20 input attributes and 950 labelled examples. This suggests that it may be beneficial to reduce the number of attributes to avoid overfitting.
- Viewing the histograms for each variable showed that there were no variables that were strongly predictive of the class.
- Some attributes have quite imbalanced distributions of values.
  - Credit history: two values cover 780 examples, and the other three values cover just 170 examples.
  - Purpose: there are 10 values, but six of them cover just 96 examples.
  - Credit amount: numeric attribute with a strongly skewed distribution.
  - Other parties: one value covers 860 examples, while the other two values cover just 90 examples.
  - Existing credits and duration: numeric attributes with a very strongly skewed distribution (see Figure 2). These strongly non-Gaussian distributions may affect the naive Bayes model (which fits a Gaussian distribution to numeric variables) so results for this model may be improved by processing the variables (for example, by discretisation).
- Two-dimensional scatter plots don't show strong class separation; this suggests that several attributes will be needed to separate the two classes.

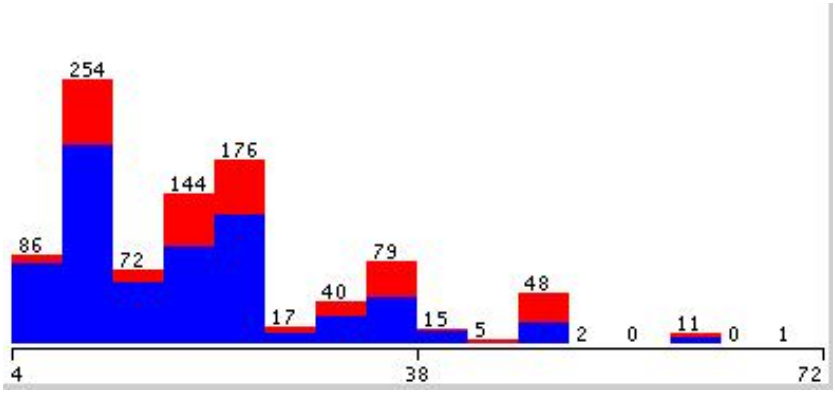

Figure 1: Histogram of duration variable.

3 Data Preprocessing

Numeric variables: duration (a2), credit amount (a5), installment commitment (a8), residence since (a11), age (a13), existing credits (a16), num dependents (a18).

- Applied standardisation (to make all numeric attributes have zero mean and unit variance) and then PCA. First PC: $0.716a5 + 0.649a2 - 0.25a8$ (plus negligible terms). Second PC: $0.625a13 + 0.548a11 + 0.44a16 - 0.33a2$.
- Created a scatter plot of PC1 versus PC2 (see Figure 2). This shows that there is poor class separation in this space, so it is not worth using principal components as variables. There are a few outlying points, but they are not far enough from the body of the data to be a problem. For example, the red cross at $(2.5, 3.0)$ is instance 515.

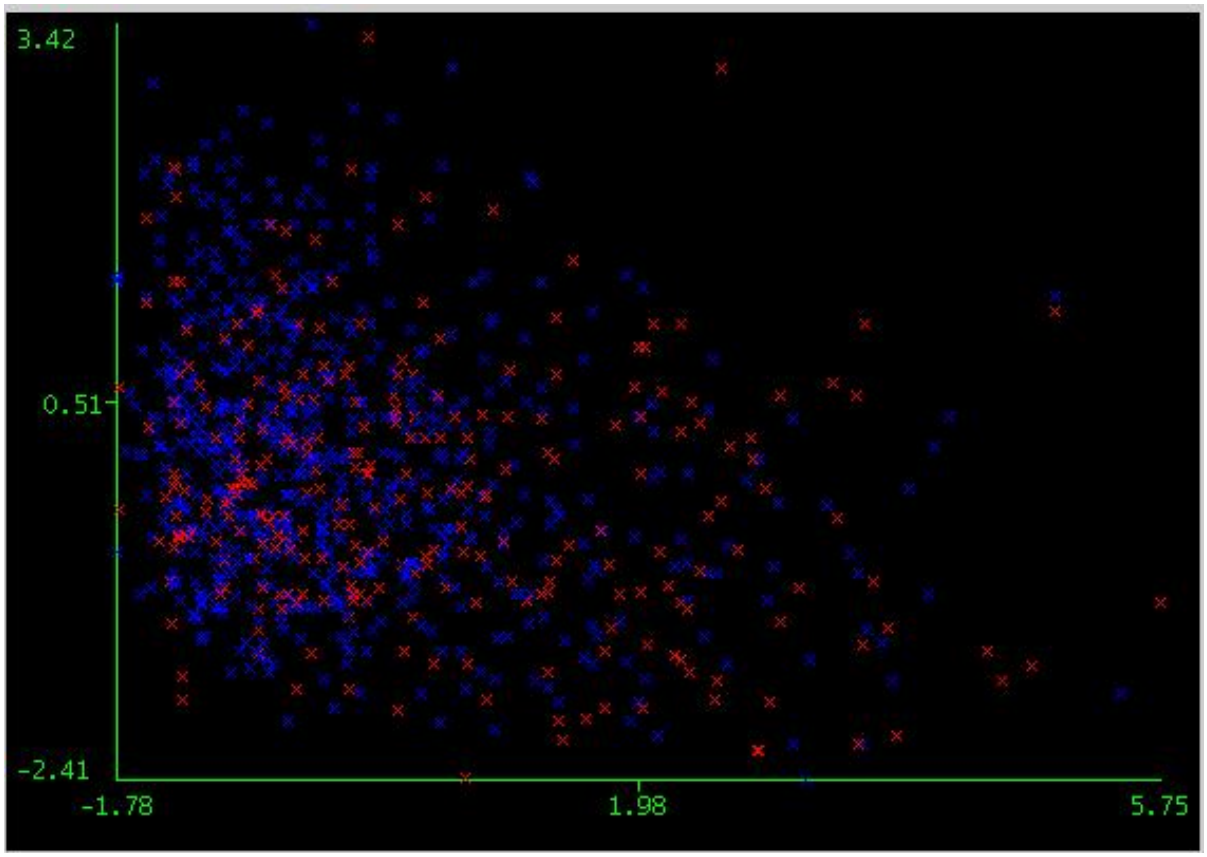

Figure 2: PCA visualisation plot.

4 Classification Models

In this section we develop models of varying complexity in order to select those we think are the best for this problem. This is carried out in five phases: benchmark models; attribute selection; model development; combining models; cost-based modelling.

4.1 Benchmark Models

As basic benchmarks on the dataset, naive Bayes, k-nearest neighbour, logistic regression and J4.8 were applied without any preprocessing and with their default parameters. 10-fold cross-validation was used to increase the reliability of the error estimate. The value of k was chosen by cross-validation and the value selected was 9.

Model	Accuracy
naive Bayes	74.7%
k-nearest neighbour ($k = 9$)	73.7%
logistic regression	74.4%
J4.8	71.9%

Table 1: Benchmark results on whole dataset.

4.2 Attribute Selection

Given the limited amount of data, it was felt to be important to select a subset of important attributes.

- Inspecting the decision tree, we found that 'checking status' was the variable at the root node; most other attributes were used; most leaf nodes covered a small number of examples (typically 15 or less).
- Attribute selection using the Cfs subset evaluation picked four attributes: checking status (a1), duration (a2), credit history (a3) and savings status (a6).
- Attribute selection using the chi-squared evaluator chose the following attributes (in order of importance): 1, 3, 2, 6, 4, 5, 12, 7, 15, 13, 9, 14, 10, 20, 17, 19, 8, 16, 18, 11.
- Attribute selection using the information gain measure chose the following attributes (in order of importance): 1, 3, 2, 6, 4, 5, 12, 7, 15, 13, 9, 14, 20, 10, 17, 19, 18, 8, 16, 11. This is almost identical to the chi-squared measure.
- The symmetric uncertainty measure chose the following attributes (in order of importance): 1, 3, 2, 6, 5, 13, 4, 15, 12, 20, 7, 14, 10, 9, 17, 19, 18, 8, 16, 11.

There is a clear indication that the four most important attributes are a1, a3, a2 and a6. Using just these attributes, the performance of the benchmark models is shown in Table 2. They show that reducing the number of variables improves the performance of k-nearest neighbour, logistic regression and J4.8 and slightly reduces the performance of the naive Bayes model.

Model	Accuracy
naive Bayes	74.2%
k-nearest neighbour ($k = 9$)	75.2%
logistic regression	74.6%
J4.8	74.8%

Table 2: Benchmark results on four-attribute dataset.

As an intermediate selection, we chose the ten most important attributes according to information gain: 1, 2, 3, 4, 5, 6, 7, 12, 13, and 15. This gave the results in Table 3, which shows that reducing the number of attributes to ten is not as effective at improving the performance of k-nearest neighbour and J4.8 as reducing the number to four.

Model	Accuracy
naive Bayes	74.6%
k-nearest neighbour ($k = 10$)	72.1%
logistic regression	74.1%
J4.8	71.3%

Table 3: Benchmark results on ten-attribute dataset.

4.3 Model Development

The goal of this phase is to explore the modelling process in more detail and select the optimal parameters for each model.

4.3.1 Naive Bayes

This model performed consistently well, but was at its best with the full dataset. The main issue to explore is the representation of numeric variables: some have few values (so don't look Gaussian), for example installment commitment, while others are skew, such as credit amount. There are two ways to address this: either to use a kernel density estimator to approximate the non-Gaussian distributions, or to discretise the attributes. We chose the second alternative for simplicity.

Attributes 18–20 were removed on the basis that they had very little predictive power. Those attributes with a small number of values were discretised into the same number of bins; those with a large number of values were discretised into 10 bins of equal frequency. The naive Bayes model trained on this dataset had an accuracy of 74.2%, which is little different from the benchmark.

4.3.2 k-nearest Neighbour

We continued using the four-attribute dataset and 9 neighbours. We experimented with distance weighting (1/distance and 1-distance). The results are in Table 4.3.2.

Weighting	Accuracy
Standard	75.2%
1/distance	72.2%
1-distance	75.5%

Table 4: Results of varying kNN parameters.

These results show that there is a small decrease in performance using 1/distance weighting and a very slight increase in performance using 1-distance weighting.

4.3.3 Logistic Regression

We continued using the four-attribute dataset. We experimented with varying the ridge estimator parameter (that controls the model complexity: small values allow very flexible models). The results are shown in Table 4.3.3.

Ridge parameter	Accuracy
1×10^{-8}	74.6%
1×10^{-4}	74.7%
1	74.7%
10	73.7%

Table 5: Results of varying the ridge parameter in logistic regression.

There is little variation, but increasing the ridge estimator from 1×10^{-8} (the default value) to 1 does lead to a slight improvement in performance.

4.3.4 Decision Trees

The key algorithm parameters are those that control the complexity of the model. All experiments are carried out on the four-attribute dataset (see Table 4.3.4).

Complexity Control	Parameter Value	Accuracy
Post-pruning	0.35	75.1%
Post-pruning	0.30	75.1%
Post-pruning	0.25	74.8%
Post-pruning	0.15	74.9%
Post-pruning	0.10	74.5%
Reduced error pruning	—	71.9%
Minimum number of objects	10	73.5%
Minimum number of objects	20	73.6%
Minimum number of objects	30	73.6%

Table 6: J4.8 results on four-attribute dataset.

The variation in results is quite small (apart from reduced error pruning, for which the reduced size of training set does seem to have a significant impact). The best model uses post-pruning at quite a light level (0.30).

4.4 Combining Models

We have now identified four strongly performing models: instead of choosing one of them, we have combined them in a committee with voting used to make decisions. These models are naive Bayes (default parameters), k-nearest neighbours (9 neighbours, 1-distance weighting), logistic regression (ridge parameter set to 1.0), J4.8 (pruning at 0.3). Because of the difficulties of setting up different datasets for different models in a committee, I have made the simplification that the four-attribute dataset was

used for all the models (although this is not optimal for naive Bayes). The overall performance, as estimated using 10-fold cross-validation, was **76.4%**, which is the best we have achieved so far.

4.5 Cost-Based Modelling

While it would be possible (and probably preferable) to redo a lot of the former analysis using an unequal cost matrix, we shall take a simpler approach, normally reusing parameter settings from the equal cost scenario.

In the unequal cost scenario, the cost of misclassifying a bad credit risk (as good) is five times that of misclassifying a good credit risk (as bad). The cost matrix has the following structure:

```
% Rows  Columns
2       2
% Matrix elements
0.0     1.0
5.0    0.0
```

The default minimal cost classifier assigns every example to the bad class with a cost of 666. This is our very basic benchmark to beat.

4.5.1 Naive Bayes

We ran naive Bayes with the full set of attributes since it worked best with that in the original problem. Because the naive Bayes model provides reasonably good estimates of class posterior probabilities, we ran it with the CostSensitiveClassifier meta-model and 'minimize Expected Cost' set to true (so it trained the same model as before, but just evaluated the cost. The resulting confusion matrix was as follows:

$$\begin{matrix} 416 & 250 \\ 49 & 235 \end{matrix}$$

which corresponds to a cost of $49 \times 5 + 250 \times 1 = 245 + 250 = 495$.

4.5.2 Logistic Regression

We ran logistic regression on the reduced four-attribute dataset with the same ridge parameter (1) as before. Again, as this model outputs probabilities we used the same meta-model as for naive Bayes. The confusion matrix was as follows:

$$\begin{matrix} 315 & 351 \\ 38 & 246 \end{matrix}$$

which corresponds to a cost of $38 \times 5 + 351 \times 1 = 541$. This is significantly greater than that for naive Bayes.

4.5.3 k-nearest Neighbour

Since k-nearest neighbour does not output sensible probability values (though with 9 neighbours, a very approximate probability could be generated, but Weka does not support this), we shall use the reweighted training examples option for the CostSensitiveClassifier. The four-attribute dataset was used. The confusion matrix was as follows:

$$\begin{matrix} 315 & 351 \\ 40 & 244 \end{matrix}$$

which corresponds to a cost of $40 \times 5 + 351 \times 1 = 551$. This is greater than that of logistic regression.

4.5.4 Decision Trees

Like k-nearest neighbour, decision trees do not provide useful approximations of class posterior probabilities, so we use the same meta learner. The four-attribute dataset was used. The confusion matrix was as follows:

$$\begin{matrix} 287 & 379 \\ 42 & 242 \end{matrix}$$

which corresponds to a cost of $42 \times 5 + 379 \times 1 = 589$, which is the greatest cost of all.

4.5.5 Model Selection

From the results reported here, it is clear that naive Bayes is the best model for the unequal cost problem. It is worth noting that the setup we used is actually the same model as was developed for the equal cost scenario, just with a different way of making predictions that takes the cost matrix into account. This demonstrates graphically one of the advantages of models that output estimates of class probabilities; they can be used much more flexibly (e.g. with different cost structures) than models that just output classifications. We also note that for the naive Bayes model the misclassification costs for good examples (250) was very nearly the same as that for bad examples (245). This is intuitively a good property for the model to have.

With more time, it might have been worth evaluating the MetaCost meta-learner. It might also have been worth developing a voting model. However, given that in our setup the naive Bayes model comfortably outperformed all the other models, it may well be best to use it by itself.

5 Evaluation and Conclusions

The performance of the voting model on the test data with equal costs was relatively mediocre, at 76%, but consistent with the results achieved on the training set. The confusion matrix for this classifier was:

$$\begin{matrix} 31 & 3 \\ 9 & 7 \end{matrix}$$

The maximum achieved by anyone was 82%. The test dataset contains 34 good examples and 16 bad examples, so the default rule (classifying every example as good) has an accuracy of 68%.

The performance of the naive Bayes model on the test data with unequal costs was given by the following confusion matrix:

$$\begin{matrix} 23 & 11 \\ 4 & 12 \end{matrix}$$

which corresponds to a cost of $4 \times 5 + 11 \times 1 = 31$. The default classifier (assigning everyone to the bad class) would have a cost of 34. The best achieved by anyone was a cost of 27.

It is worth noting that the relatively small size of the test dataset makes the results on it rather unreliable, in a statistical sense. If I could have been sure that everyone would supply results in an electronic format, I would have chosen a larger set. However, given that many students only submitted paper copies in a readable form, and these had to be marked by hand, I am quite glad that it only contained 50 examples!