‘BANK MARKETING’ ANALYSIS

Chris Boatto

Justin Carver

Ekaete Czub

Brendan Dagys

Nirali Dave

SUMMARY

The ability to correctly identify potential customers is a critical skill for any business to possess. A Portuguese bank is looking to effectively target potential clients that are candidates to open long-term deposit accounts through a telemarketing campaign. Long-term deposit accounts usually require a term length of at least twelve months to qualify as long-term. Examples of long-term deposit accounts include bonds and RRSPs.

We trained several models to correctly classify customers into binary ‘Yes’ and ‘No’ responses in regard to opening a long-term deposit account. The first algorithm we ran was Naïve Bayes, which yielded an encouraging Recall value on the ‘Yes’ class at 0.685, however, this was at the expense of a great deal of Precision. Furthermore, we used the J48 (Decision Tree) algorithm, which yielded a Recall value for ‘Yes’ of 0.652, but managed to keep Precision relatively high, as indicated by an F-Measure score of 0.512. Finally, we used Logistic Regression as a final algorithm for classification and obtained our top F-Measure score of 0.525, while still maintaining a desirable 0.638 Recall score.

After identifying Logistic Regression as our best model, we performed unsupervised learning to try to discover any rules or trends we could find through Cluster Analysis and Association Rules. Due to the unbalanced dataset, both of these elements of post-predictive analysis yielded few insights of significant interest in regard to identifying potential long-term deposit account clients.

Overall, we suggest that a classification model based around Logistic Regression is the best way for the bank to identify future customers using the same data variables provided.

METHOD

Pre-Processing in R

The first step in our R pre-processing was inputting the data. We did this using the read.csv( ) function. We saved the resultant data frame in a variable called ‘data’, and selected numeric columns into a new data frame: ‘int_data’. This was for use in some functions that required only numeric attributes.

These columns were: ‘age’, ‘balance’, ‘day’, ‘duration’ , ‘campaign’, ‘pdays’, ‘previous’.

We then implemented a for-loop to simply print the class of each attribute, in order to become more familiar with them.

Another for-loop was then calculated some descriptive statistics for the attributes, if they were of a numeric data type. The statistics calculated were the minimum, maximum, median, mean, and standard deviation. We also plotted a box plot in each run of the loop, and a histogram to assess normality.

Looking at the outputs of this for-loop, we learned what range of values we should expect for each attribute. For example, average age was around 40 years, with a standard deviation of 10.6. The maximum age was 87. We also noticed that the distribution was fairly normal, albeit with a slight right skew.

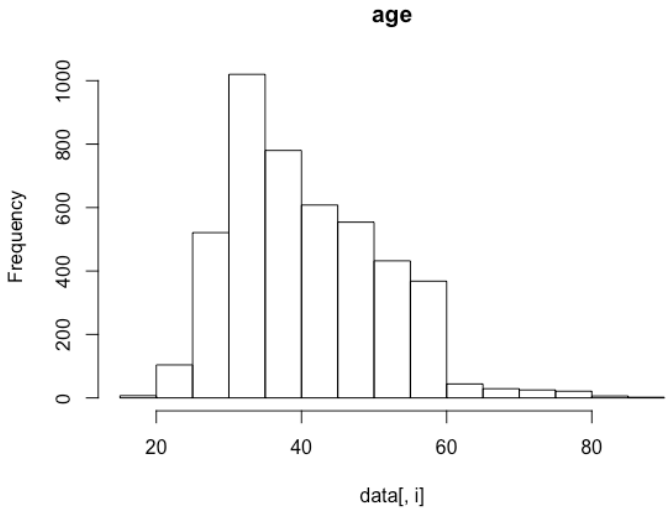

Figure 1: 'Age' histogram

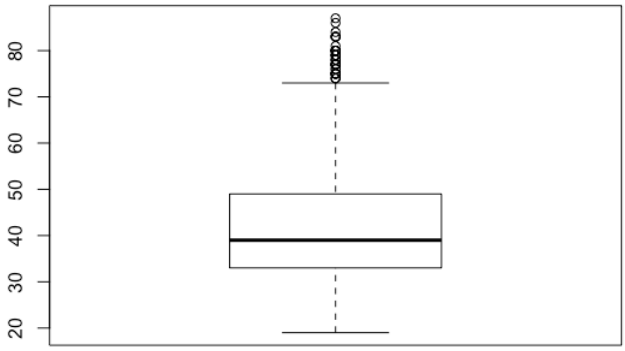

Figure 2: 'Age' boxplot

At this point, we moved on to analysis in Weka, and applied various machine learning algorithms to the dataset. We ran into some problems while doing this, and used what we learned to adapt and add to our R code, based on our experiences. We noticed in Weka that three variables could be done away with. These were 'day', which measures day of the month from 1 to 31; 'pdays', which measures the number of days since last contact between the customer and any campaign; and 'poutcome', a nominal attribute that indicated the outcome of any previous campaign the customer was involved in.

From our work in Weka, we saw that 'day' was not strongly correlated with the class attribute. Because of this, and because the attribute itself was not very descriptive (it did not provide any information about day of the week), we decided to get rid of it.

We removed ‘pdays’ and ‘poutcome’ because both were very similar to another attribute, ‘previous’, in what they measured *and* in their histograms, which all looked alike. The attribute ‘pdays’ contained a possible value, -1, that indicated that no previous contact had been made between the customer and any campaign. This was essentially a nominal category in a numeric attribute. This skewed our descriptive statistics. As well, 80% of observations had this value for ‘pdays’! The attribute ‘poutcome’ was similar as well, with 80% of observations taking a value of ‘unknown’, which meant that no previous contact had been made. It was for these reasons that we decided to remove ‘pdays’ and ‘poutcome’ and use ‘previous’ instead.

We made two more notable alterations to our dataset. We binned ‘age’ into five groups with a range of ten each (with the exception of 65+ and 0 – 29). We also categorized ‘previous’ into three possible values: ‘No Contact’, ‘One Contact’, and ‘Multiple Contacts.’

We then used three variables to calculate the Mahalanobis distance (‘duration’, ‘campaign’, and ‘balance’) as they contained notable outliers in the histograms and boxplots and were quite skewed. We used the mahalanobis( ) function with the column means and covariance of these attributes to generate a vector of Mahalanobis distances. We then used the Chi-square distribution to calculate the cumulative probability of observing these values. Subtracting these values from 1 gave us the probability of obtaining these Mahalanobis distances given the degrees of freedom. We then proceeded to remove outliers with a probability of less than 0.001, which removed approximately 150 observations.

Next, we generated a correlation matrix for all numeric variables to assess how dependent they were on one another:

	age	balance	day	duration	campaign	pdays	previous
age	1.000000000	0.083820142	-0.017852632	-0.002366889	-0.005147905	-0.008893530	-0.003510917
balance	0.083820142	1.000000000	-0.008677052	-0.015949918	-0.009976166	0.009436676	0.026196357
day	-0.017852632	-0.008677052	1.000000000	-0.024629306	0.160706069	-0.094351520	-0.059114394
duration	-0.002366889	-0.015949918	-0.024629306	1.000000000	-0.068382000	0.010380242	0.018080317
campaign	-0.005147905	-0.009976166	0.160706069	-0.068382000	1.000000000	-0.093136818	-0.067832630
pdays	-0.008893530	0.009436676	-0.094351520	0.010380242	-0.093136818	1.000000000	0.577561827
previous	-0.003510917	0.026196357	-0.059114394	0.018080317	-0.067832630	0.577561827	1.000000000

Figure 3: Correlation matrix of numeric variables

We can see that the greatest correlation is between previous and pdays.

We also plotted a histogram of the class variable, to observe the frequency distribution of ‘Yes’ and ‘No’. We saw that approximately 8/9 observations have a class value of ‘No’!

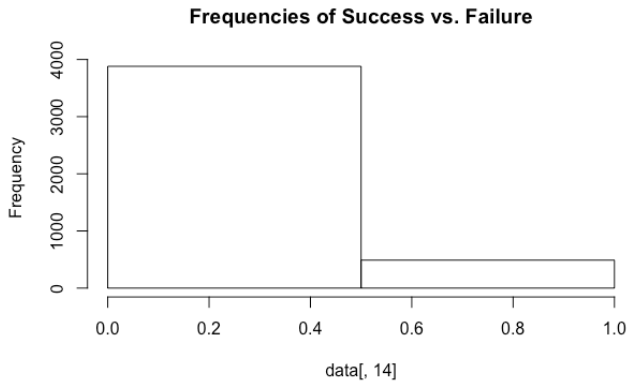

Figure 4: Histogram of the class variable

Finally, we used the ggcorr( ) function to plot a visually-friendly representation of the correlation between variables. It echoed, in a visual way, what we saw in the correlation matrix.

ZeroR

ZeroR, which stands for 'Zero Rule', is a classification algorithm that only classifies based on the most common class variable, which in this case is the 'No' class. ZeroR is the simplest algorithm used for classifying data. The output of ZeroR allows us to see how many results there are for each outcome and what percentage of Confidence we will have from the baseline data. It focuses only on the result rather than the factors that lead to the result, and often can be used as a good baseline algorithm for classification when accurate classification of all classes is of equal importance.

For our research purposes, classification of both classes is *not* of equal importance, and therefore ZeroR serves us very little purpose. More specifically, ZeroR yields a correctly classified instances rate of ~88% due to the unbalanced nature of the data set, but also has a 0.0% Recall rate for our most important class, 'Yes'. Our future models will focus more on the Recall value of the 'Yes' class as opposed to the percentage of correctly classified instances, and our best classification models may not reach the overall C.C.I. of ZeroR, but that is nonetheless acceptable.

```
Correctly Classified Instances        3881               88.8101 %
Incorrectly Classified Instances       489               11.1899 %
Kappa statistic                          0
Mean absolute error                      0.1989
Root mean squared error                  0.3152
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances             4370

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    1.000    0.888      1.000    0.941      ?        0.499     0.888     0
                 0.000    0.000    ?          0.000    ?          ?        0.499     0.112     1
Weighted Avg.    0.888    0.888    ?          0.888    ?          ?        0.499     0.801

=== Confusion Matrix ===

    a    b   <-- classified as
 3881    0 |    a = 0
  489    0 |    b = 1
```

Figure 5: ZeroR output

Naïve Bayes

Naïve Bayes is a classification algorithm that classifies based on probabilities, classifying an observation by evaluating its probability of belonging to one class vs. belonging to another. The main assumption needed for the use of Naïve Bayes is the independence of the variables. In this case, it is nearly impossible to believe that the variables in this dataset are not related in some way, but that does not mean that this algorithm will have poor predictive power in the case of this dataset.

During the original modeling using Naïve Bayes, we obtained an overall C.C.I. of ~ 88%, but managed an F-Measure score for the 'Yes' class of only 0.440. An F-Measure score is the harmonic mean between Precision and Recall for a class, and gives us a numeric value that measures the optimal balance between Precision and Recall. To adjust for the importance of identifying observations belonging to the 'Yes' class, we introduced a cost matrix that penalized any False Negatives, and therefore coerced the classification of observations to the 'Yes' class. After adjusting the cost matrix, we reached an optimal point, resulting in a Recall of 'Yes' of 0.685 and an F-Measure score of 0.497. Although the C.C.I. fell to 84% (4% below the baseline set by ZeroR), this model performed better in relation to our research goal of identifying 'Yes' class observations as thoroughly as possible.

```
Cost Matrix
 0    0.8
 3    0


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3692               84.4851 %
Incorrectly Classified Instances       678               15.5149 %
Kappa statistic                          0.4134
Mean absolute error                      0.2262
Root mean squared error                  0.3463
Relative absolute error                113.7276 %
Root relative squared error            109.8532 %
Total Number of Instances             4370

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.865    0.315    0.956      0.865    0.908      0.436    0.863     0.978     0
               0.685    0.135    0.390      0.685    0.497      0.436    0.863     0.427     1
Weighted Avg.  0.845    0.295    0.893      0.845    0.862      0.436    0.863     0.916

=== Confusion Matrix ===

    a    b   <-- classified as
 3357  524 |    a = 0
  154  335 |    b = 1
```

Figure 6: Naïve Bayes output

J48 (Decision Tree)

A Decision Tree algorithm, such as J48, is used to visualize the classification of observations. It works by evaluating the Information Gain of each variable at each level of the tree and splitting the tree on the variable that provides the most gain, therefore minimizing the amount of Entropy (uncertainty). The major challenge we faced in creating the Decision Tree with the dataset provided was overfitting the tree. Running a 10-fold classification with the 'Confidence' and 'minObject' settings at their default values resulted in a tree with almost 100 leaves and 100 stems. This type of tree can lead to overfitting of the model and also gave a very poor performance in regard to the 'Yes' class F-Measure score of 0.380. In order to improve the performance of our model, we decreased the 'Confidence' level to 0.10 and changed the 'minObjects' value to 50.

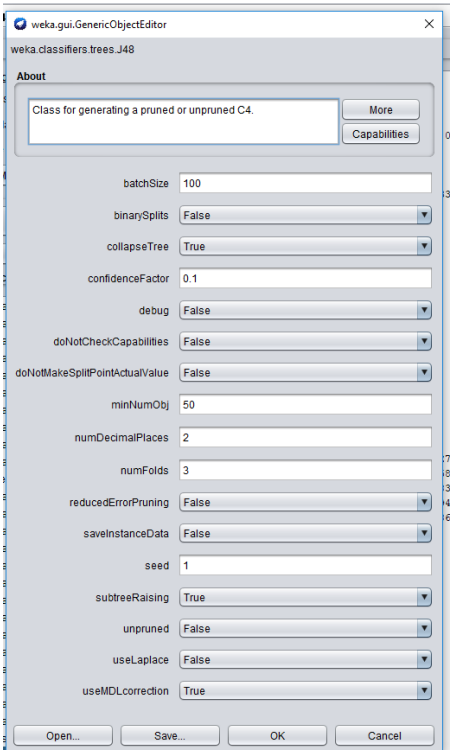

Figure 7: J48 decision tree settings

With these adjustments we got manageable tree size of 13 with 8 leaves. The percent of correctly classified instances was 89.31%, but we the Recall value for the 'Yes' class was still poor.

Our next adjustment to the model was to introduce the cost matrix to the Decision Tree. Once again, we punished False Negatives, sacrificing overall Precision of the algorithm for better classification of the 'Yes' class. After identifying the ideal weighting of the cost matrix, we managed to obtain a Recall of 0.652 and F-Measure score of 0.512 for the 'Yes' class. The resulting tree had 23 leaves (slightly more than the original), but still managed to perform better for our purposes. The F-Measure score of our J48 model was stronger than that of Naïve Bayes, and that gave us optimism going forward into our third model.

Logistic Regression

Logistic Regression is a probabilistic algorithm that can be used for classification purposes. It obtains probabilities of belonging to the 'Yes' class and classifies the instance as 'Yes' if the probability exceeds 0.50. An advantage of Logistic Regression is that it does not require the data to be linearly related.

Logistic Regression operates under a few key assumptions. Firstly, and most importantly, Logistic Regression requires the class variable to be binary in nature. In our dataset, the class variable is indeed binary, containing the values 'Yes' and 'No', respectively.

Another important assumption of the Logistic Regression algorithm is the independence and absence of multicollinearity among the variables. To check for multicollinearity, we observed a correlation matrix of the numeric variables which showed no concerning correlations between variables.

Last, a large dataset is required to run Logistic Regression, and this dataset is sufficiently large to run this algorithm. For the purposes of our research goal, it is imperative that we focus on the Recall value and F-Measure of the 'Yes' class in order to evaluate our model. We will sacrifice overall Accuracy in order to obtain the best F-Measure possible.

Running a 10-fold cross validation test without a cost matrix, Logistic Regression performed well in terms of overall classification of instances, reaching a C.C.I. of 89.6%. However, the Recall and Precision in terms of the 'Yes' class both fell well under 0.4, which is very poor in terms of our research. To correct this issue we once again utilize the cost matrix, and after reaching an optimal point in the cost matrix, we were able to obtain our best F-Measure score of 0.525.

In summary, in terms of identifying possible 'Yes' observations using the dataset provided, this model gives us the most predictive power.

```
Cost Matrix
 0    1
 3.8  0


Time taken to build model: 0.14 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3805               87.0709 %
Incorrectly Classified Instances       565               12.9291 %
Kappa statistic                          0.4527
Mean absolute error                      0.2067
Root mean squared error                  0.3128
Relative absolute error                103.8947 %
Root relative squared error             99.2134 %
Total Number of Instances             4370

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.900    0.362    0.952      0.900    0.925      0.462    0.885     0.983     0
                 0.638    0.100    0.446      0.638    0.525      0.462    0.885     0.467     1
Weighted Avg.    0.871    0.333    0.895      0.871    0.880      0.462    0.885     0.925

=== Confusion Matrix ===

    a    b   <-- classified as
 3493  388 |    a = 0
  177  312 |    b = 1
```

Figure 8: Logistic regression output

POST-PREDICTIVE ANALYSIS

Association Rules

The first step of our Association Rules analysis was to apply a filter to our dataset. This is a requirement of the Apriori algorithm: the class attribute must be nominal. After doing this, we did a preliminary run with default settings. This generated 10 rules after 8 cycles, and produced itemsets up to a size of 4. We noticed that it did not provide any predictions for the class value of 'Yes', and that all the rules overwhelmingly contained the attributes 'contact' and 'default'.

The default settings have 'car' as False, minimum Confidence set to 90%, and the number of rules generated set to 10. We proceeded to adjust some of these for our next run of the algorithm. We changed 'car' to True to generate predictions of the class variable. We also increased the number of rules to 20 to see more information. These changes generated rules with a lower, more realistic Confidence of around 90%, on average. However, our results were still not exactly as we had hoped. Specifically, we strongly wanted to get rules that predicted a class value of 'Yes', and to have more variety in the attributes present in our rules.

We ran the algorithm some more times, while experimenting with the settings in hopes of improving our results. Some changes we tried were increasing the minimum Support from 0.1 to 0.5, and decreasing the minimum Confidence from 0.9 to 0.65. Both of these alterations alone, and together, did not significantly help us achieve our goals for this algorithm. We had similar Confidence values, not a single 'Yes' on the right side of our rules, and a lack of variety in the attributes present.

We concluded that it would be difficult to produce rules in an unbalanced dataset such as this one, as the support for all rules of 'Yes' would be extremely low, based on the nature of the class variable. The class variable was predominantly 'No', with a ratio of 'No':'Yes' of 3881:489.

Finally, we tried removing various combinations of attributes, but none of this helped us. Ultimately, we couldn't get rules that led to a class variable of 'Yes', or that provided any valuable insights.

Cluster Analysis

When performing cluster analysis, we were mostly concerned with Clustering to classes, so we made sure to enable the 'Classes to Clusters' function. The issue with this dataset in terms of Clustering is that it is so imbalanced that it is hard to correctly cluster to classes with a high Accuracy percentage. When running the Simple K-means Clustering algorithm, we obtained two clusters identified by the two binary classes. The correctly clustered instances were only ~53% which is extremely poor, and the Recall value for the 'Yes' class was in the neighbourhood of 30%. These numbers offer us very little information in terms of predicting the 'Yes' class. There are some small inferences that could possibly be made using Clustering, such as that a greater mean 'balance' being present in the 'No' class, and a longer mean 'duration' being present in the 'Yes' class.

Overall, Clustering does not offer much predictive power in terms of identifying possible 'Yes' observations.

CONCLUSION

Using this Bank Marketing dataset, we were left with one overriding problem of an unbalanced dataset. Utilization of a cost matrix allowed us to create models that were more in-line with the purposes of this analysis. Unsupervised learning through Association and Clustering did not provide us with much in terms of predictive power, however, we enjoyed moderate success in creating a model during the classification stage of the analysis. Furthermore, Logistic Regression was the machine learning algorithm that produced the best F-Measure score for our 'Yes' class.

Despite the promising results of Logistic Regression and other algorithms, there is still plenty of room for improvement in future analyses.

First, the further analysis of other variables could introduce a new way of approaching our business problem and could lend more predictive information to future models. Additionally, other machine learning algorithms could be used for classification in hopes of finding a model that better fits the data and gives a better F-Measure for the 'Yes' class than we were able to obtain in this project.

Finally, and more specifically, adding information about the year to the dataset, so that we have the ability to decipher the days of the week for last contact, could certainly yield valuable information going forward.

Our recommendation to the bank would be to focus on customers who have previously had long-duration calls in previous campaigns, as it appears to signal a possible interest in long-term deposit accounts.

Additionally, it may be prudent to focus on those who have a small bank balance as it appears a lower balance favours the use of long-term deposit accounts. However, we believe that the best way for the bank to move forward is to use our Logistic Regression model, input future candidate variables into the model, and attempt to identify telemarketing candidates through that method.