Evaluation of Bank Marketing Models

# Introduction

Marketing campaigns are a common practice within the banking industry. As such, our research uses a dataset provided from the UCI Machine Learning repository (<http://archive.ics.uci.edu/ml/datasets)> about a Portuguese bank. The purpose of this research paper is to provide an effective telemarketing strategy to sell term deposits to the bank’s clients. To do that, we will gather Precision, Recall and F-1 measures for each model and for different tests.

Classification models (Naïve Bayes, Decision Tree, Repeated Incremental Pruning to Produce Error Reduction (JRip), Logistic Regression, K-Nearest Neighbours, a Neural Network, and a Support Vector Machine) are used to predict the outcome that a bank customer will accept a term deposit. The performance evaluation of these algorithms will be verified with Accuracy, Precision, Recall and F-1 measures as the data is imbalanced.

# Literature Review

We wanted to review literature that discussed models that were used to solve a binary classification problem (in particular, bank data), and the methods used to compare them. Then, other secondary issues included what models or processes were used to select features (dimensionality reduction) before using the models, how was the imbalanced data handled (since our data has 87%/13% ratio), and how was the data transformed from numeric to categorical and vice/versa, and finally how was the data ‘normalized’ or scaled?

The initial dataset was provided by Moro *et al.* [1], and they have written a number of papers on the Portuguese bank data, such as [1-4]. In paper [1] Moro *et al.* discussed how to use data mining techniques (the Cross-Industry Standard Process for Data Mining or “CRISP”) when formulating models that could explain the success of a contact i.e. if the client subscribes to the deposit. This article is one of the earliest ones on the Portuguese bank marketing dataset. The data used was larger, and had more outcomes, than the dataset provided to us. Data was discarded that had missing values (we had none) and by reviewing the data alone, about half of the attributes were deleted. Three different models were used (Naïve Bayes, Decision Trees and Support Vector Machines). All three models were tested and validated with a holdout split of a training set (2/3) and test set (1/3). Twenty iterations were run for each model. ROC curves and LIFT analysis were performed to compare the models. The results were good, with a minimum AUC of 87%. However, rebalancing, normalizing, discretizing, outliers and “unknown” categories were not discussed.

In paper [2] Moro *et al.* use the same dataset with several classification models (Logistic Regression (LR), decision trees (DTs), neural networks (NNs), and support vector machines (SVMs). It is well known that the LR and DT provide an understandable model while providing good predictions whereas NNs and SVMs are hard to understand but are much more flexible. The dataset also included economic data which our dataset does not. Imbalanced data is discussed but left as is. Feature selection was done in two steps. In the first step, business experts were used to filter down the attributes to 22, then an adapted forward selection method was used. The results were compared for the 4 models using AUC and ALIFT and the results were good for all four models. The minimum time for feature selection to run one of the methods was 53 hours. We may not be able to run similar methods or models with a MacBook Air. For the NN model, a highlight is a Table of the relative importance of the 22 attributes, along with an extracted decision tree from the NN model.

In paper [5] four classification models are used on the Portuguese bank data to determine which models work well, and what factors are the most important. The models used are multilayer perception neural network (MLPNN), augmented Naïve Bayes (TAN) or Bayesian Networks, logistic regression, and decision tree model C5.0. The dataset seems to be left it as is as there is no more discussion on modifying the data. Further, the author did not discuss rebalancing, normalizing, discretizing, or what to do with any outliers and “unknown” categories. The author used classification accuracy, sensitivity and specificity to measure the performance of the models. The models performed well, with a minimum accuracy of 89.16% on the training set and 88.75% on the testing set for TAN, and the other 3 models performed even better.

In paper [6], the same Portuguese bank data is used with models in a WEKA package. The two classification models used are Bayes Net algorithm and Naïve Bayes algorithm. The dataset attributes are discussed but then the discussion moves on quickly to the models. 10-fold cross validation is used for the training and test sets. The authors did not discuss rebalancing, normalizing, discretizing, or what to do with any outliers and “unknown” categories. Further, there was no discussion on what attributes worked the best. Although the authors use WEKA to run the models, it was useful to know what models were actually used. The evaluation metrics were the Confusion Matrix, Kappa, TP Rate, FP Rate, Recall, Precision, F-Measure and ROC Area. Both models performed well with Bayes Net having 81.5% accuracy and Naïve Bayes with 78% accuracy.

Feature selection (dimensionality reduction) is covered in [7-9]. In particular, paper [7] Ladyzynski *et al.* discusses dimensionality reduction in that it can fall into two categories: feature extraction (i.e. Principal Component Analysis) and feature selection. Feature selection is then used by selecting a subset of the original features and an algorithm that does this is called ‘Boruta’. In using Boruta, the number of attributes decreased from 616 to 164. The purpose was to identify models for identifying customers interested in credit products for a large bank in Poland. Three machine learning algorithms were used: 1) classification trees - CART, 2) Random forests and 3) Deep belief networks – DBN. We intend to try Boruta on our data.

It is well-known that there are four type of methods used to rebalance the data when the minority class is overwhelmed by the majority class: 1) undersamping i.e. EasyEnsemble algorithm; 2) oversampling, or a combination thereof; 3) Synthetic (i.e. Random Over Sampling Examples or ‘ROSE’, Synthetic Minority Oversampling Technique or ‘SMOTE’); and 4) cost sensitive or “error correct” learning. We intend to try all four methods to see which one performs best. There are quite a few articles on imbalancing [10 to 21]. In particular, [15] discusses undersampling, [15-18] the SMOTE technique and results, and [19-21] cost-sensitive methods. We will discuss these techniques in greater detail in the next section.

Finally, the review of literature that discusses data transformation and data scaling is now discussed.

Crone et al [22] did a high-level review of 16 research papers and detail various pre-processing steps done by the various authors, including the methods used (i.e. CART), data reduction (feature selection), standardization and/or discretization of the continuous attributes. Further, standard methods of scaling is discussed which we will explore and perhaps use in our data. Discretization, or ‘binning’, of categorical data is also mentioned and this is a method we will use on our data.

# Dataset

There are two datasets in the University of California at Irvine Machine Learning Repository [5]. The bank dataset was collected by S. Moro et al [1]. The end result is to obtain a model or models that will provide an effective telemarketing strategy to sell long-term deposit accounts. Data from old and current telemarketing campaigns were obtained as part of this dataset. Multiple contacts were often needed to determine whether a customer would subscribe to a long-term deposit account. The data is described below in Table 1.

There are two datasets provided. The ‘full’ dataset has 45,211 examples and is ordered by date from May 2008 to November 2010. The second bank dataset has only 10% of that data (4,521), randomly selected from the full bank dataset to allow for testing more computationally demanding machine learning algorithms (such as a Support Vector Machine model). We have found that we could only use the smaller dataset to run all the models and tests.

The classification goal is to predict (“yes”) if the client will subscribe to a term deposit. There is a total of 16 attributes with 7 numeric, 6 categorical, and 3 binary. The dataset presents a binary classification problem, and an imbalance problem - one key point is that the data is highly imbalanced (88%/12% ratio), which is a major reason for reviewing data reduction methods in the later steps. Because we want results for the minority class, we will use Precision, Recall, and the F1- measure from the *minority class perspective*.

Data from old and current telemarketing campaigns were obtained as part of this dataset. Multiple contacts were often needed to determine whether a customer would subscribe to a long-term deposit account. The dataset attributes are shown below in Table 1.

Table 1: Data List

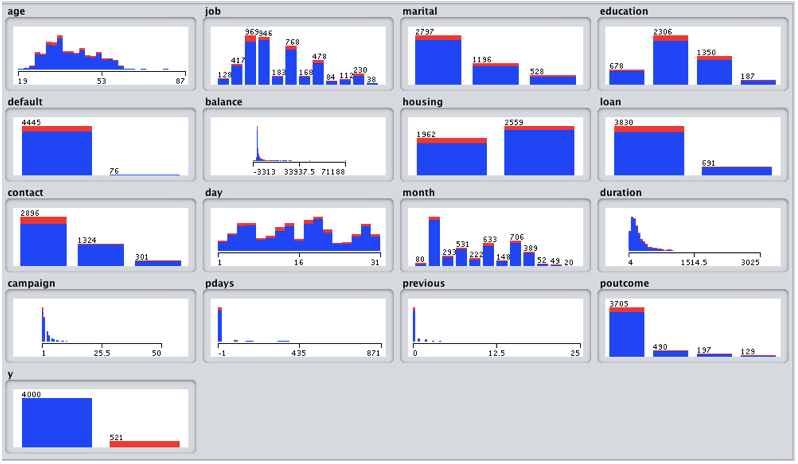
|  |  |  |  |
| --- | --- | --- | --- |
| # | Attribute | Description | # of classes for categorical data |
| 1 | Age | Age | Numeric / discrete |
| 2 | Job | Type of job | 12 categorical classes: ‘admin’, ‘blue-collar’, ‘entrepreneur’, ‘housemaid’, ‘management’, ‘retired’, ‘self-employed’, ‘services’, ‘student’, ‘technician’, ‘unemployed’, ‘unknown’ |
| 3 | Marital | Marital status | 3 categorical/(ordinal): ‘single’, ‘married’, ‘divorced’ (divorced includes widowed) |
| 4 | Education | Level of Education | 4 categorical/(ordinal): ‘primary’, ‘secondary’, ‘tertiary’, ‘unknown’ |
| 5 | Default | Has credit in default or not | 2 binary: yes, no |
| 6 | Balance | Average yearly balance in Euros | Numeric / discrete |
| 7 | Housing | Has housing loan or not | 2 binary: yes, no |
| 8 | Loan | Has personal loan or not | 2 binary: yes, no |
|  |  | ***Related to current campaign:*** |  |
| 9 | Contact | Type of communication | 3 categorical: ‘cellular’, ‘telephone’, ‘unknown’ |
| 10 | Day | Last contact day of the month | numeric |
| 11 | Month | Last contact month of the year | 12 categorical: ‘jan’, ‘feb’, ‘mar’ …. ’nov’, ‘dec’ |
| 12 | Duration | last contact duration in seconds | Numeric / discrete |
| 13 | Campaign | # of contacts for this customer | Numeric / discrete |
|  |  | ***Related to previous campaign:*** |  |
| 14 | Pdays | # of days since previous campaign (-1 means client was not previously contacted) | Numeric / discrete |
| 15 | Previous | # of contacts before this campaign | Numeric / discrete |
| 16 | Poutcome | Outcome of the previous campaign | 4 categorical: ‘failure’, ‘success’, ‘other’, ‘unknown’ |
| 17 | Y | **Class attribute showing whether the client has subscribed a term deposit or not** | 2 binary: yes, no |

There is a total of 17 attributes with 7 numeric features and 10 nominal / categorical features. The purpose of each feature is described in Table 1. The numeric features are Age, Balance, Day, Duration, Campaign, Pdays and Previous. The categorical features are Job, Marital, Education, Contact, Month, Poutcome, and the four binary features are Default, Housing, Loan and the “Y” target. There was no missing data and no obvious incorrect data.

Table 1 shows the categorical attributes and it is interesting to note that there is an “unknown” category in four of these attributes (job, education, contact, and poutcome). The ‘job’ category has 12 classes, which is a lot to discern from. The ‘married’ class has only three classes. Education has four, but one of them is ‘unknown’. Contact is either by ‘cellular’ or regular ‘telephone’, and also ‘unknown’. Month has the 12 months of the year. Poutcome has ‘failure’ and ‘success’ but also ‘other’ and ‘unknown’ again. And finally, the binary attributes are either ‘yes’ or ‘no’ for Default, Housing, and Loan.

Figure 1 below shows the breakdown at a high level of the attributes and their classification. For the numeric data, it appears that almost all are skewed right (age, balance, duration, campaign, pdays, previous). For the categorical data, the data is somewhat split across the categories (like job, marital, education, contact, month). As for the binary data, default and loan are very high in the “yes” category whereas housing is more evenly split. Even though the numerical data is highly skewed right, and there are many outliers on the high end, the data itself is not “incorrect”. Note the “blue” colour is for the “no” answer and red is for the “yes” answer.

*Figure 1: Data Pre-Processing – WEKA output of attributes*

****

The next two Tables further detail the attributes to get a better understanding of the data.

Table 2 shows the numeric data with the minimum value, Q1, median, mean, Q3, the maximum value and the standard deviation. We will also see plots that show that all the numeric data is not normal and is right-skewed. The outliers noted will **not** be taken out of the dataset, because technically they are not ‘outliers’- the data just has some values that are on the higher ends. For example, customers can be 85 years old, and others can have a high balance in their banking account. However, we will use the 97.5% confidence interval as a way to keep the numeric data within this limit. This will be discussed in the ‘Approach’ section, along with other methods.

Also, Pdays contains a “-1” and even though it doesn’t make sense in the definition, and “0” doesn’t either, *all values of “-1” will be replaced with zero* (which currently doesn’t exist in this attribute) knowing that 0 means NO CONTACT and not 0 days*.* This attribute can then be more easily normalized (0 to max) since all values are now positive. Also, when ‘pdays’ = -1, ‘previous’ = 0, which makes sense, but this means that the two variables are correlated. However, we will see the correlation table results later, and it will be noted that there are no large correlations.

Table 2: Numeric Attributes Information

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Attribute | # outliers | Min | Q1 | Median | Mean | Q3 | Max | Std Dev. |
| 1 | Age | 487 | 18 | 33 | 39 | 40.94 | 48 | 95 | 10.6 |
| 6 | Balance | 4729 | -8019 | 72 | 448 | 1362.3 | 1428 | 102127 | 3044 |
| 10 | Day |  | 1 | 8 | 16 | 15.8 | 21 | 31 | 8.3 |
| 12 | Duration | 3235 | 0 | 103 | 180 | 258.2 | 319 | 4918 | 257.5 |
| 13 | Campaign | 3064 | 1 | 1 | 2 | 2.8 | 3 | 63 | 3.1 |
| 14 | Pdays | 8256 | -1 | -1 | -1 | 40.2 | -1 | 871 | 100.1 |
| 15 | Previous | 8256 | 0 | 0 | 0 | 0.6 | 0 | 275 | 2.3 |

Table 3: Categorical Attributes Information

|  |  |  |  |
| --- | --- | --- | --- |
| # | Attribute | # of types | Attribute Types |
| 2 | Job | 12 | ‘admin’, ‘blue-collar’, ‘entrepreneur’, ‘housemaid’, ‘management’, ‘retired’, ‘self-employed’, ‘services’, ‘student’, ‘technician’, ‘unemployed’, ‘unknown’ |
| 3 | Marital | 3 | Marital status – ‘married’, ‘divorced’, ‘single’ (divorced includes widowed) |
| 4 | Education | 4 | ‘primary’, ‘secondary’, ‘tertiary’, ‘unknown’ |
| 5 | Default | 2 | ‘yes’ or ‘no’ |
| 7 | Housing | 2 | ‘yes’ or ‘no’ |
| 8 | Loan | 2 | ‘yes’ or ‘no’ |
| 9 | Contact | 3 | ‘cellular’, ‘telephone’, ‘unknown’ |
| 11 | Month | 12 | ‘jan’, ‘feb’, ‘mar’ …. ’nov’, ‘dec’ |
| 16 | Poutcome | 4 | ‘failure’, ‘success’, ‘other’, ‘unknown’ |
| 17 | Y | 2 | ‘yes’ or ‘no’ |

In Table 3, there are several attributes that have the “unknown” category (job, education, contact, poutcome). That data will be kept and left as is.

The visualization of the data is next. All tests and plots were performed in R code.

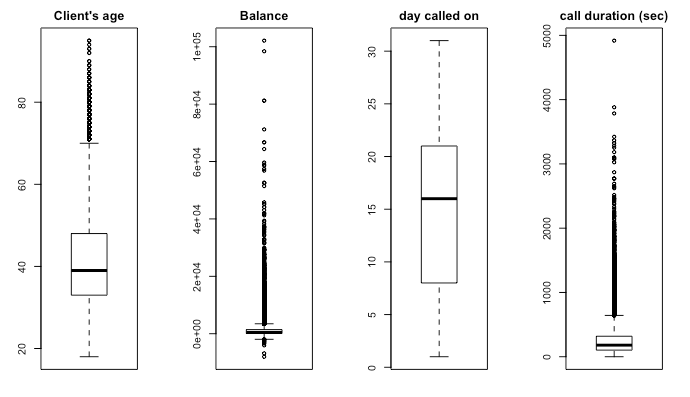
Visualization of Numeric data

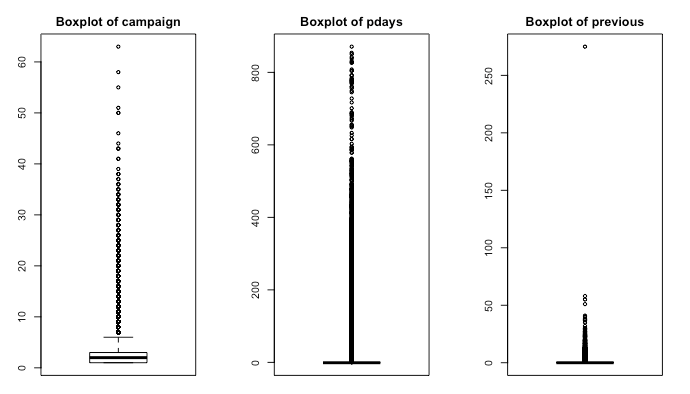
In order to perform numerical analysis, the data is presumed “normal”. This implies that the data has a bell-shaped curve (not skewed), and does not have multicollinearity (the dependent variables are not correlated). In reviewing the data in R, we did normality tests, boxplots, histograms, and a correlation map for the numeric data. Q-Q plots were also run in R (not shown here) and they distinctly show that the data is not normal.

As well, the normality of each numeric attribute was tested using the Shapiro-Wilk test (not shown), and all tests failed, meaning that the numeric attributes were not normal (any value above 0.05 indicates normality). Therefore, the data will be scaled and that version will be used for all the models.

The boxplots of the numeric data are shown below in Figure 2. From these plots, we can see that the data is not normal, based on the size and location of the boxes. One can see the “outliers” such as in ‘previous’. The histograms in Figure 3 also show the skewness of all the numeric attributes.

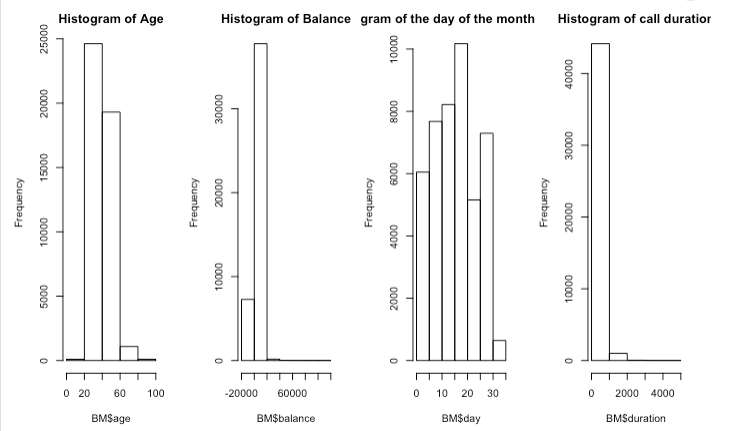
*Figure 2: Boxplots of Numeric data*

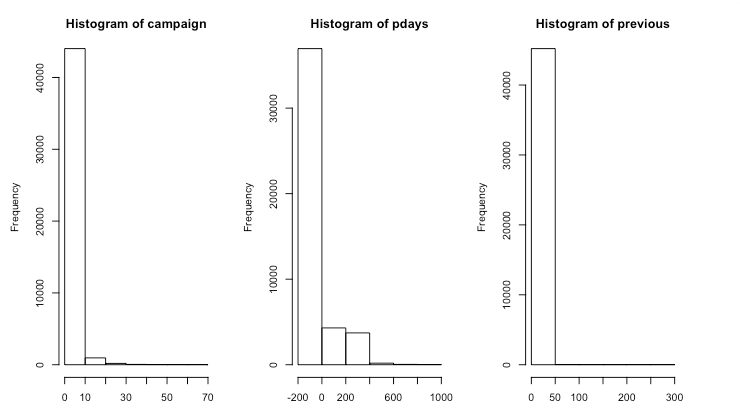




The histograms of the numeric data are shown below in Figure 3. Notice that for many of the attributes, a large portion of the instances are on the far left.

*Figure 3: Histograms of Numeric data*



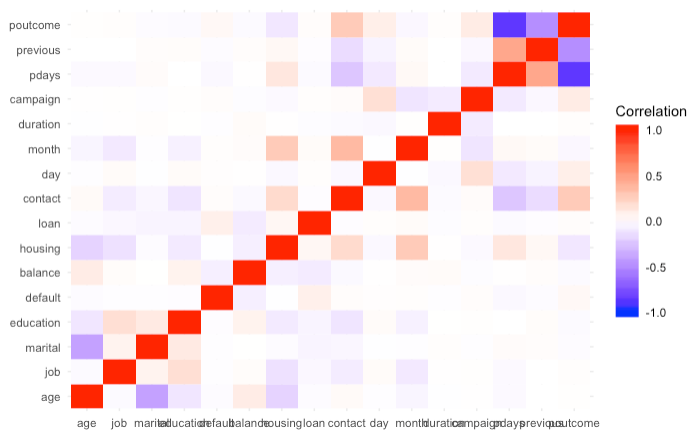


Categorical data

As well, correlations were performed for the categorical data using Pearson’s Chi-squared test and the tests showed that the data was relatively unrelated to each other. The work was done in R and is not shown here.

A correlation heat map of all the data was also generated for a visual representation of the correlations (see Figure 5 below). Upon review, the data correlations showed that they were all very low, except for three stronger correlations where two were negative and one mildly positive (not shown but in R code): i) ‘poutcome’ to ‘previous’ of -0.49, ‘poutcome’ to ‘pdays’ of -0.86, and ‘previous’ to ‘pdays’ of 0.45.

*Figure 4: Correlation Heat Map of all data*



Discussions on the dataset data being changed are discussed in the Approach section. The R source code is stored in Github for all the sections [see link – <https://github.com/njeanius/bank1/tree/draft> ].

# Approach

The steps we took are shown below. We used a modified version of the CRISP [2,3] methodology. The data is a binary classification problem in that (some of) the inputs (both numeric and nominal) should be able to predict the outcome of the class variable (Yes or No) based on the respective models. The steps are broken down as follows.

Step 0: Understanding the Performance Measures

Step 1: Pre-Processing the Data

Step 1a: 95% confidence interval

Step 1b: Change the categorical variables to ‘dummy’ variables.

Step 1c: Scale the numeric data

Step 2: Rebalance the data

Step 2a: Undersampling

Step 2b: Oversampling

Step 2c: Oversampling - ROSE

Step 2d: Oversampling - SMOTE

Step 2e: Cost matrix

Step 3: Data Reduction

Step 3a: principal component analysis (PCA)

Step 3b: RFE + Random Forest,

Step 3c: Boruta

Step 3d: Artificial Neural Network (ANN - LVQ)

Step 3e: Regression Methods – ANOVA and AIC

Step 3f: Regression Methods - AIC stepwise

Step 4: Model Parameters

Step 5: Models

5A: Model 1 – Naïve Bayes (NB)

5B: Model 2 – C5.0 Decision Tree (DT)

5C: Model 3 – JRip

5D: Model 4 – Logistic Regression (LR)

5E: Model 5 - K- Nearest Neighbours (KNN)

5F: Model 6 – Neural Networks (NN)

5G: Model 7 – Support Vector Machine (SVM)

Step 6: Results

Step 7: Conclusion

**Step 0: Understanding the Performance Measures**

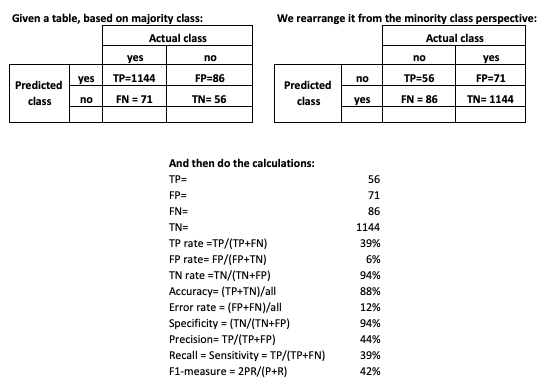
Note that even with the results of the models shown in the Results section, we will be showing highlights of our findings along the way, as the two go together. The performance measures we will show in the results will be Accuracy, Precision, Recall, and the F1 measure, all from the *minority class perspective.* However, in the next sections, we will show the F1 measure as a proxy of how we made decisions on the data methods used, on the rebalancing, and on the data reduction. The results need to be shown as we go along, and the then the final model results will be shown. Therefore, we will explain below how the metrics were obtained, given a confusion table output from any of the models used.

Because of the highly imbalanced nature of the dataset, and we want to find what attributes will provide a positive minority value, it has been recommended to use precision and recall [16] in order to obtain proper performance measures, from the minority perspective. We will add to that the F1 measure and Accuracy.

In order to compare the performance of the models, ***based on the minority class***, the statistical measures required are: True Positives (TP) rate, False Positives (FP) rate, Accuracy, Precision and Recall, and the F1 measure. These measures are defined using a confusion matrix as shown below right. The percentage of the Correct/Incorrect classification is the difference between the actual and predicted values of variables. Note that since we are provided the confusion matrix as it is shown below, but we really want to see the calculations based on the minority class, we switched the formulas around to make it work. The calculations are shown below in Figure 6 with the two confusion tables.

* True Positives (TP) irate s the number of correct predictions that an instance is true out of total positives. Also known as Sensitivity, it is the true positives predicted out of total actual positives in the dataset.
* False Positives (FP) is the number of incorrect predictions that an instance is true out of the total negatives. It is the negatives predicted as positives out of total actual negatives in the dataset.
* True Negatives (TN) is presenting a number of correct predictions that an instance is false.
* False Negatives (FN) is the number of incorrect predictions that an instance is false.
* Accuracy (A):
* Precision (P): Precision is the true positives predicted out of all the positives predicted.
* Recall (R): Recall is essentially the Sensitivity rate and is the True Positives rate.
* F1 measure (F1) combines the Precision and Recall and is 2PR/ (P+R).

*Figure 5: Confusion Matrix Calculations*



## Step 1: Pre-Processing the Data

The first thing that we did was to change all the “-1” in ‘pdays’ to zero, and divided the call ‘duration’ by 60 to get the data in minutes, and thereby be more in line as a data range with the rest of the data. However, this step would be done after the confidence interval step1a below.

We did try deleting data that had “unknown” as in ‘contact’ but we got worse results. Therefore, all the data was kept. We also tried many other configurations but the results were not good so we dropped those ideas, however, we did thoroughly test the three choices noted below.

Upon review of various papers noted above, for example [21, 22] as discussed below, many methods were discussed on how to pre-process the data before running models. We chose in the end to keep the nominal data as is, and to use methods 1 and 3 below only. The piecemeal results will be shown below and the full set in the Results section.

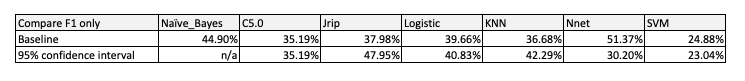
We tried various datasets with the models, but only fully tested the following setups:

1. Using the 95% confidence interval (2.5% to 97.5%) to minimize/maximize the numbers in four of the features: balance, duration, campaign, and previous.
2. Change the categorical variables to ‘dummy’ variables.
3. Scale the numeric data.

**Step 1a: 95% confidence interval**

In article [17], the Authors mention the very out-of-range data in the same banking dataset, and discuss using the 95% confidence interval as a way to keep the data in a specific range – regardless of scaling or not. Article [29] mentions using the 98% confidence interval. We used the 95%. They also discussed equal frequency distribution using k percentiles on certain features. We set the data up but did not pursue this option.

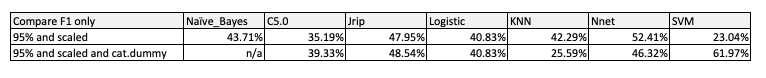
For five of the features, we used the 95% confidence interval (2.5% to 97.5%) to minimize/maximize the numbers in ‘balance’, ‘duration’, ‘campaign’, ‘pdays’ and ‘previous’. The features ‘duration’, ‘pdays’, and ‘previous’ each had over 100 records on the upper end that was out of range. We compared the model results of this effect before and after, and there was a large improvement of 10% for JRip, large drop of 21% for Nnet, and other minor changes for the rest, so the results are inconclusive. The Table 4 below highlights the differences, and a complete comparison is discussed at the end of Step 1.

*Table 4 - Comparison of Baseline to 95% confidence interval method* 

**Step 1b: Change the categorical variables to ‘dummy’ variables.**

In articles [15, 22, 25, 28] the changing of categorical variables to dummy variables was discussed with four methods. The first was “N encoding”, which is the method we will use, “N-1” encoding, “Thermometer encoding” and “Ordinal encoding”. The authors did a significant amount of testing, and for the models of neural net, SVM and DT, changing the categorical variables to ‘dummy’ variables was positive. Using ordinal data was negative for NN and SVM. The authors went into a lot of detail that won’t be repeated here, but suffice to say, there is no “one method fits all models” situation.

Therefore, we chose to try one of the methods (although we did quickly test other versions) and changed the categorical variables into dummy “1” variables. This meant that ‘job’ now had 12 features, not one, the ‘marital’ feature had three now, ‘education’ four, ‘contact’ three, ‘month’ 12, ‘poutcome’ four and the remaining three features (default, housing and loan) stayed as binary 1/0. This version was tested on the models to see if there was a difference or not on the model performance. As we are also testing other changes, we compared the “95% and scaled and categorical dummy” vs. the “95% and scaled” version and the results are shown below in Table 5. Notice there is a big improvement for SVM, large drop for KNN, but minor differences for the others - so the results are inconclusive again. Scaling is discussed in the next section, and a complete comparison of the different choices is discussed at the end of Step 1.

*Table 5 - Comparison of categorical ‘dummy’ to ‘95% and Scaled method’* 

**Step 1c: Scale the numeric data**

In article [22, 28], scaling was a method used on the authors’ models.… and interestingly enough, it depended on the models used that informed which method was best. And because scaling was found to be sensitive to extreme values, that is why we chose the 95% change noted above.

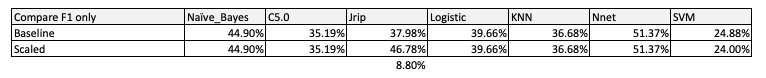
We will limit our discussion on scaling to two different methods: 1) the standardized method using z-scores; and 2) min-max scaling (0 to 1). These methods are quite common. Scaling is used when data has feature ranges that are very different from one another and/or when a model requires distance in its calculations such as in PCA or K-NN. Scaling is not normalizing data. Normalizing data changes the shape of the data to a bell curve whereas scaling just scales down the numbers so that they are between 0 and 1 (for example). The formulas are noted below.

z-score formula is (-1 to 1):

min-max scaling is (0 to 1):

We coded both methods in R and briefly checked the z-score version against the min-max version and found the min-max version better so the min-max scaling version was used going forward (results not kept). We compared the model results of the min-max scaling effect before and after running the models, and there was a big improvement for JRip of 8.8%, a drop in 0.88% for SVM, and the rest remained the same - so the results are minimal but considered positive. Table 6 below highlights the differences, and a complete comparison is discussed at the end of Step 1.

*Table 6 - Comparison of Baseline to Scaled method*



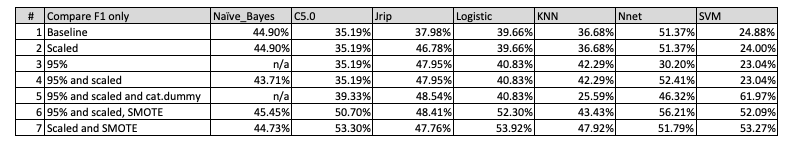
Although we haven’t discussed rebalancing yet, the results in Table 7 below use the ‘SMOTE’ rebalancing method (as it was found to be the best!) which will be discussed in Step 2. The different methods are highlighted below in Table 7. As noted above, we have already compared (1) to (3) in step 1a, and (4) to (5) in step 1b, and (1) to (2) in step 1c.

In summary, scaling the data (2 vs. 1) (vs. baseline) gave JRip an 8.8% advantage but reduced SVM’s by 4.88% and the rest did not change. The 95% over baseline (3 vs. 1) gave JRip a 9.97% advantage but reduced NNet’s by 21.17%, Logistic and KNN benefited by 1.17% and 5.61% respectively. The 95% and scaled in (4 vs. 3) improved (or was even) for C5.0, JRip, Logistic, KNN and NNet, and only a minor decrease for Naive Bayes (1.19%) and SVM (1.84%, 0.96%, 0.96%) over choices 1, 2 and 3, respectively. Therefore, method 4 is a good choice.

With (4) vs (5), categorical as dummy variables works very well for SVM model but not well at all for KNN and minor differences for the rest. Comparing (4) to (6) gives SMOTE a huge advantage in the models and we will compare the other balancing models in the next section. Comparing (7) vs. (2) also shows that SMOTE is a very good choice for 6 models and only a minor negative change for Naïve Bayes. Finally, comparing (6) to (7) with the difference being the 95% change, the results are inconclusive as the difference works better for some models but not for others.

In summary, the imbalanced method (4) (95% and scaled) is a good technique, but if we can over-balance with SMOTE in (6), then the model results are even better. Going forward the data will use the 95%, is scaled, and may or may not use SMOTE for further tests, depending on the test.

*Table 7 – Comparison of various pre-processing methods*



The next section discusses rebalancing of the dataset four different ways so that we can get a better response with the minority category.

## Step 2: Rebalance the data

Because the data was imbalanced, with a 89% majority “no” and only 11% minority “yes” in the target, rebalancing should be considered. Various articles discuss this topic [10-22] and the most popular methods are discussed below.

##### Therefore, we chose to test four sampling methods (and using the test dataset with numeric scaled + categorical data): 1) Undersampling the majority class; 2) Oversampling the minority class; 3) Generation Of Synthetic Data By Randomly Over Sampling Examples (ROSE); 4) Synthetic Data Generation or SMOTE; and 5) a Cost Sensitive Learning or “error correct” situation with a cost error matrix used on the confusion matrix. No special parameters were used for these first four rebalancing techniques.

**Step 2a: Undersampling**

Undersampling is a technique used to under-sample the majority class. It does this by creating a subset of the original dataset by (randomly or selectively) disregarding some of the samples of the majority class while keeping the original population of the minority class. In general [15] states that undersampling provides better results than oversampling. It also mentions that the combination of undersampling and oversampling does not provide better results than simply undersampling the majority class. An issue with undersampling is that there may be critical examples missing that could have had vital information contrasting the two classes. In the future, as discussed in [17], one option would be to try using the Tomek Links under-sampling method for the imbalanced dataset.

**Step 2b: Oversampling**

As mentioned in [15], oversampling methods produce a superset of the original dataset by replicating some of the samples from the minority class or by creating new samples from the original minority class samples. Basically, oversampling increases the minority class. The sampling can be done randomly or informatively (just as in undersampling), but with the second method, overfitting can result along with reduced accuracy. Synthetic minority oversampling technique or SMOTE is a widely used oversampling technique and discussed further in step 2d. A drawback of this method is that oversampling can cause model overfitting.

**Step 2c: Oversampling - ROSE**

The ROSE (Random Over Sampling Examples) package generates a synthetic balanced sample of data simulated according to a smoothed-bootstrap approach. This package has well defined accuracy functions to do the tasks quickly.

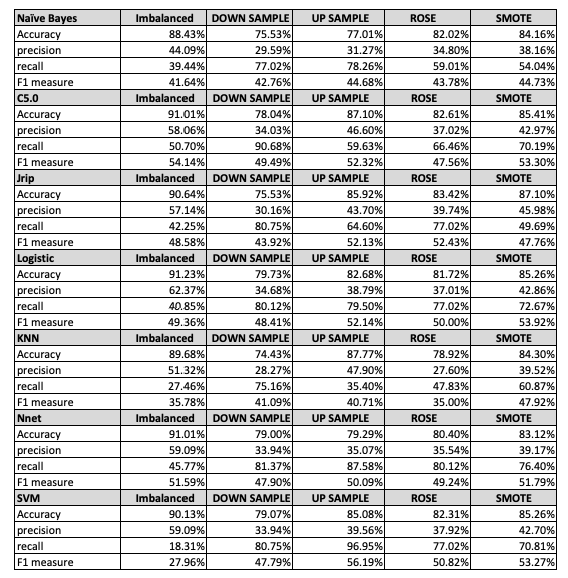
**Step 2d: Oversampling - SMOTE**

For synthetic data generation, a well-known technique is SMOTE (synthetic minority oversampling technique) in which more samples of the minority class are created by finding the k-nearest neighbours in the minority class for each of the samples in the class.

The results of the first four rebalancing methods are shown below in Table 8. The data used was the scaled numeric (and categorical was kept) version. For the Naïve Bayes model, the rebalancing methods were overall slightly better than the imbalanced version based on the F1 measure, and SMOTE was best overall – based on Recall (54.04%) and the F1 measure (44.73%). For the C5.0 model, the Imbalanced version was the best overall with a F1 of 54.14%, although the down sample had a 90.68% recall. For the JRip model, ROSE performed the best for the F1 measure at 52.43% but its precision was lower than the imbalanced at 57.14%. For logistic regression, SMOTE came out on top at 53.92% for F1, but again, the imbalanced had a higher precision of 62.37%. For KNN, SMOTE was best with F1 at 47.92%, however, its precision was lower than that of the imbalanced precision at 51.32%. Nnet was only 0.20% higher than the imbalanced version and although its recall was much higher at 76.40% (vs. 45.77% for imbalanced), its precision was down by about 20% to 39.17%. Finally SVM’s up-sample was the best with 56.19% F1 then SMOTE came in second at 53.27%. So, the final best results: imbalanced – C5.0, up-sample – SVM, ROSE – Jrip, and SMOTE: NB, logistic, KNN, nnet.

SMOTE won for four out of the seven models, therefore, even though SMOTE isn’t the best balancing method for *all the models*, we will be using SMOTE in future model testing.

*Table 8 – Comparison of four rebalancing methods for the 7 models*

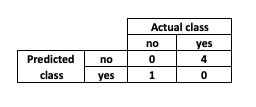


**Step 2e: Cost matrix**

Cost-sensitive matrices are discussed in [20] for a neural network model in which the authors found the Meta-Cost method gave better results over the usual cost matrix method. We did not try the Meta-Cost method, however we will use a neural network model, and add a cost matrix to it.

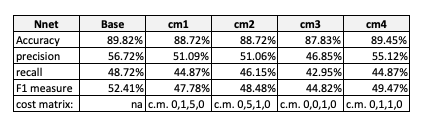
For cost-sensitive learning, we don’t create any new balanced data, but basically “punish” the model using a cost matrix. We usually punish for getting a False Positive and this is the worst case because we wouldn’t want to give a client any credit if they end up bankrupt. On the other hand, if we miss giving a client the credit or a False Negative, who would use the money wisely, is not as big a deal. If we use an example and suppose that a loan default costs the bank four times as much as a missed opportunity than the error matrix would look like in Figure 6 below:

*Figure 6 –cost matrix example*



The Cost Matrix is a method for adjusting the weight assigned to misclassifications. The model tries to avoid classification errors with a high error weight. We tried four different cost matrices, partly to see “what if” and we were surprised that none of the matrices proved better than the base case for the Neural Net model. The cost matrix results are shown in Table 9 below. We see that for the four measures used – Accuracy, Precision, Recall and F1 – *all the answers* were better for the Base case of no cost matrix. Therefore, we will not be using a cost matrix in the models section, however this could be an area of further research.

*Table 9 – Comparison of cost matrix for Nnet model on imbalanced data (97.5% & Scaled)*

**

In the above Table 9, we see that the cost matrices used were ineffective – for the model we chose to use which was nnet. Perhaps if we had more time, we could try all sorts of various combinations, but given the four cost matrices above gave similar or worse results than the base case, we will not use the cost matrix for the models. Finally, given the results in Table 8 and discussed above, SMOTE is the best balancing method and will be used for the model testing in step 5.

**Step 3: Data Reduction**

In [24], the authors discuss seven techniques to reduce dimensionality. The steps that are valid for us are: i) Step 3 - Reduce highly correlated items (we checked this in the Dataset step and we don’t have any highly correlated features); ii) step 4 - Run PCA on numeric data and use the results to reduce the features; iii) Step 5 - run Random Forest model; iv) steps 6 and 7 - use backward feature elimination i.e. Boruta and forward feature construction. In [7] the authors discuss various methods and Boruta is one of them, thus we will use Boruta as one of our methods.

We will use PCA, RFE + Random Forest, Boruta, Nnet LVQ and regression methods to see if we can reduce features in the dataset that will improve the models. We ran a test of each of the results of these features on the models, and compared the model results before and after. The results of these ‘data reduction’ models will be shown at the end of this section.

The dataset’s categorical data was transposed into numeric data using a somewhat ordinal process. For example, in the ‘marital’ feature, single would be a ‘1’, married a ‘2’, and divorced a ‘3’. The original numeric data was scaled, and so now we use a numeric dataset for this section.

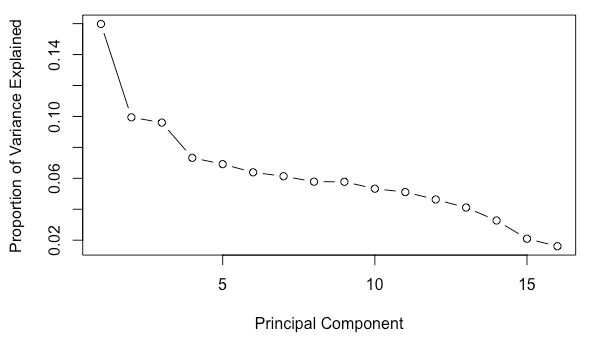
**Step 3a: Principal Component Analysis (PCA)**

PCA uses a coordinate system to transform the data features in 2-, 3-, or an n-dimensional space. The first axis is in the direction of the greatest variance of the points to maximize the variance along that axis. Then, the second axis is perpendicular to it. The third axis can lie anywhere, but it has to be perpendicular to the first axis as well, and so on. In short, covariance matrices are calculated to find eigenvectors and the eigenvectors are the axes of the transformed space.

The ‘principle components’ (pc) are now orthogonal (perpendicular) data made up of pieces of the original data. Normally we would like to see the first component, pc1, make up for, say 50% of the variation in the target answer, then pc2 could be, say, 30%, then pc3 10% then pc4 5%, for a total of 95% which is a decent percentage to forecast an answer. And, we now only have 4 principal components to explain all the data, down from 16 original features.

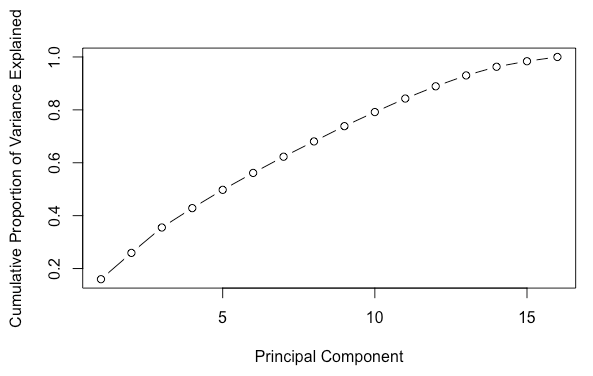
Unfortunately, when the data was run, we did not get good results. The first diagram in Figure 7 shows that the first component explains for, at most, 15% of the total variation of the dataset, then pc2 is 10%, pc3 is 10%, pc4 7%, pc5 7% - so that the first five principal components only account for ~50% of the total variation. Then it just gets worse, and the last component #16 accounts for under 1%.

*Figure 7 – PCA - Proportion of Variance Explained*



The next figure 8 shows that, at 5 pc’s, only 50% of the variation is explained, and then at 10 pc’s 80% of the variation is explained, and then at 15 pc’s close to 100% of the variation is explained. The whole point of using pca is to reduce the number of components to a smaller number, like 5, to explain over 80% of the variation. As this has not happened in this case, we will not be using these results on the models.

*Figure 8 – PCA – Cumulative Proportion of Variance Explained*



**Step 3b: Recursive Feature Elimination (RFE) + Random Forest**

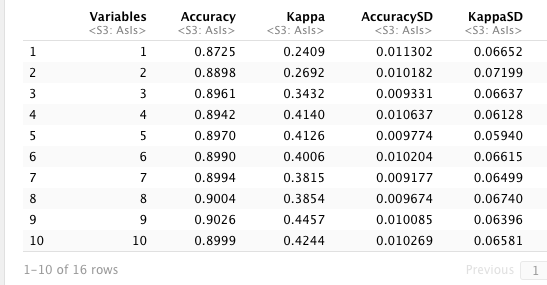
In [15] the authors mention that the benefit of the Random Forest method is its readiness to show the importance of the variables. This model is similar to the LVQ model in that it readily ‘spits out’ the top-most factors, along with the second-best factors.

Our results showed the top 5 variables (out of 9): duration, month, poutcome, day, contact.

Then all the best set of 9 variables: "duration", "month", "poutcome", "day", "contact", "pdays", "previous", "age", "housing".

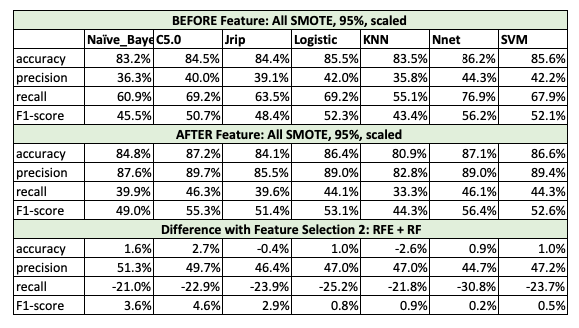
The model also provided a table of the Accuracy, Kappa, etc. of the variables, in the order that they are in the dataset. For example, in Table 10, we see the results for Age (#1), job, marital, education, etc.

*Table 10 – PCA Results*



In using the results of this model, we kept only the first 5 variables mentioned above and deleted the rest. The results of this test are shown in Table 11 below. The F1 and precision results are all better after the feature has been implemented on the data, and tested with the models than before - but there are large negative differences in Recall. Overall, the F1 scores average 1.91% better. This feature could be effective in reducing the number of attributes – down to 5! One note of caution is that, from what we have seen with PCA, because the data as a whole seems to be required to represent all the data, we may not want to delete so many features.

*Table 11 – RFE + RF Results*

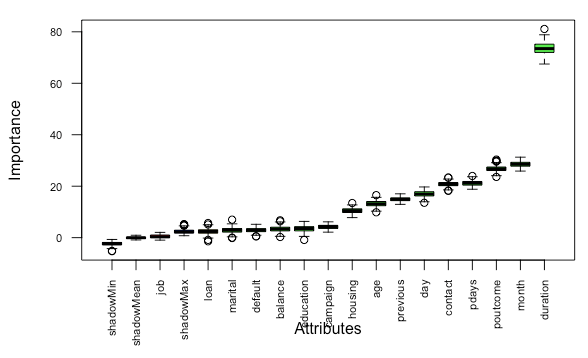


**Step 3c: Boruta**

Boruta is a feature selection algorithm for finding all relevant variables. The algorithm is designed as a wrapper around a Random Forest classification algorithm. There were no special parameters used.

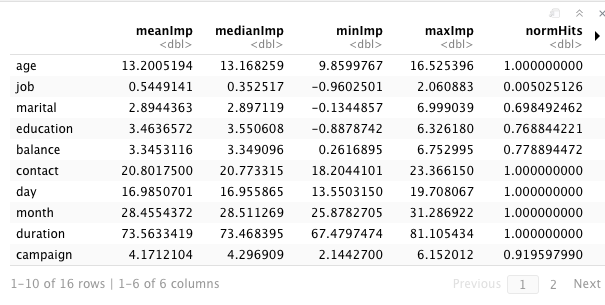
With Boruta, it only gave up ‘loan’ and ‘job’ as the only features we could delete. Figure 9 below shows how important it designated each feature. Again, we see that ‘duration’ is very important, and then the importance drops substantially to the next feature – in this case, ‘month’ and so on down to the last feature.

*Figure 9 – Importance Plot of the Variables*



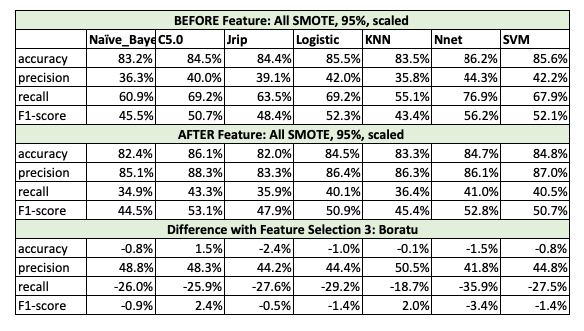
The Boruta model also provided statistics on all the features, in order of the dataset, as shown in the Table 12 below.

*Table 12 – Table of Statistical Data of the Variables*



So in using the results of this model, we only deleted 2 variables mentioned above (‘loan’ and ‘job’). The results of this test are shown in Table 13 below. The F1 and precision results are mainly worse after the feature has been implemented on the data, and tested with the models than before - but C50 and KNN models show a positive 2.4% and 2.0% respectively. Overall, we would not recommend using this feature on this particular dataset.

*Table 13 – Boruta Results*

****

**Step 3d: Artificial Neural Network (ANN - LVQ)**

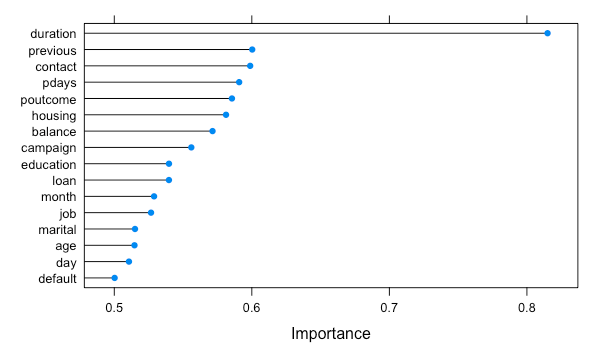
We used an ANN called Learning Vector Quantization (LVQ). LVQ is a classification algorithm for binary and multiclass problems. LVQ model creates codebook vectors by learning with the training dataset. Codebook vectors represent class regions. They contain elements placed around the respective class according to their matching level. If the element matches, it comes closer to the target class, and if it does not match, it moves farther from it. With codebooks, the model classifies new data.

The parameters for this model were size and k. We let the model decide the optimal parameters (size=50, k=10), then we used those parameters for any further testing if needed. Size is the codebook size and k is the number of prototypes.

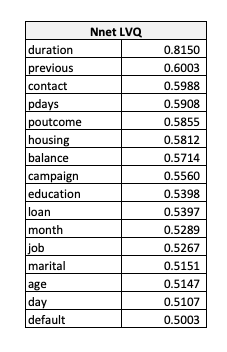
LVQ is a quick way of gathering features that are important. When we ran the results, we obtained the Importance of each of the 16 factors. As this model was used more for information on the relevant factors, we didn’t do any testing or obtain ‘results’ on how good the model is.

Our results in Figure 10 and Table 14 below show that ‘duration’ is extremely relevant and is at 0.815, but then the other factors drop substantially from ‘previous’ down to ‘default’ and all between 0.5 to 0.6. Therefore, we know that call ‘duration’ is a relevant factor, but it’s not too clear on the other factors, however, we decided to delete the factors below 0.57 cutoff and tested this new dataset of 7 features on the models. The dataset used was the 95% CI, scaled numeric plus nominal. The results of this test are shown at the end of this section.

*Figure 10 - Importance Plot of the Variables*

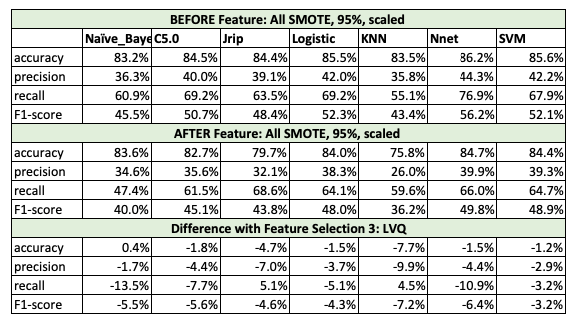


*Table 14 – Importance Values of the Features*



As shown in Table 15 below, we see that the results of deleting all but 7 features gave poorer numbers for each of the models, based on the F1 measure, and the other three measures. For example, Nnet has a negative difference of 6.4% with the drop in the number of features. We therefore do not recommend this feature on this particular dataset.

*Table 15 – LVQ Results*

****

**Step 3e: Regression Methods- ANOVA and AIC**

Multiple Regression is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. We have sixteen explanatory variables in this dataset, and the ‘response’ is the Y variable, being the answer “yes” or “no”. We are combining the results of ANOVA and AIC in this section as they gave the same results.

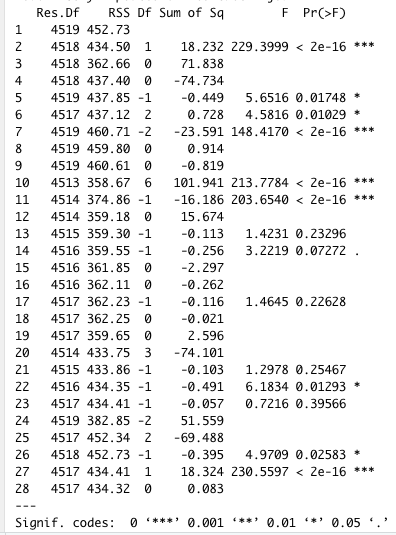
ANOVA can be used on a set of models to obtain the “RSS” number for each model, and then the model with the lowest RSS is the best model of the group of models that was created by the author. Other similar methods to ANOVA include the Akaike Information Criterion (AIC) and BIC (Bayesian Information Criterion).

A residual sum of squares (RSS), which is a measure of error, is a statistical technique used to measure the amount of variance in a data set that is not explained by a regression model. So, for example, if RSS were zero then the model obtained explains 100% of the dataset. The lower the RSS the better, and if comparing models together, the lowest RSS would be the best.

There may still be better models out there – it’s just a matter of creating different models with different components – the variables. For example, model 1 could just include ‘job’ and model 2 could include ‘job’ and ‘duration’ and the ANOVA process reviews these models and provides the RSS and other numbers for each model.

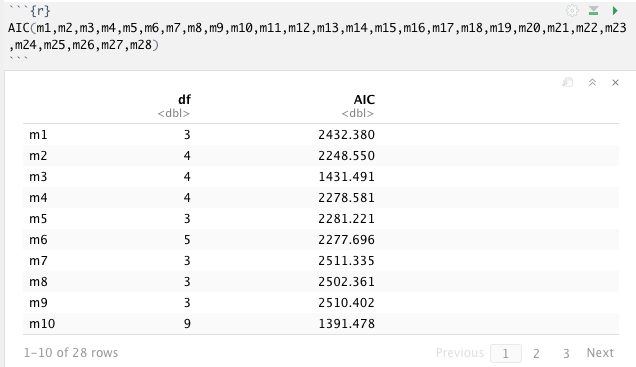
In trying a set of 28 models, Model 10 had the lowest RSS number ( y ~ poutcome + month + contact + default + education + job + duration) so these 7 attributes only were testing on the 7 models. A screenshot of the results is shown below in Figure 11 with Model 10 showing the lowest RSS number, AND being a significant model.

*Figure 11 –ANOVA output showing Model 10 Importance*



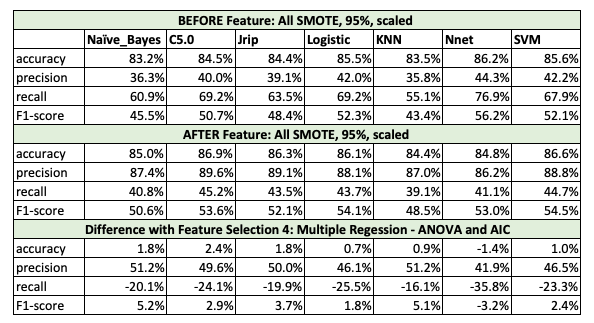
The output using AIC of the models shows a table of all the attributes, with the AIC numbers for each model, and again we see that model 10 has the lowest AIC number of all 28 models (not all data shown).

*Table 16 - AIC Results of 28 models – 10 model results showing*



As shown in Table 17 below, we see that the results of deleting all but 7 features gave better numbers for each of the models, based on the F1 measure, except for Nnet model. All the precision values increased by at least 41.9% but recall decreased by 16.1% to 25.5%. We therefore recommend this feature on this particular dataset, save for Nnet.

*Table 17 – Regression Results – ANOVA and AIC*

****

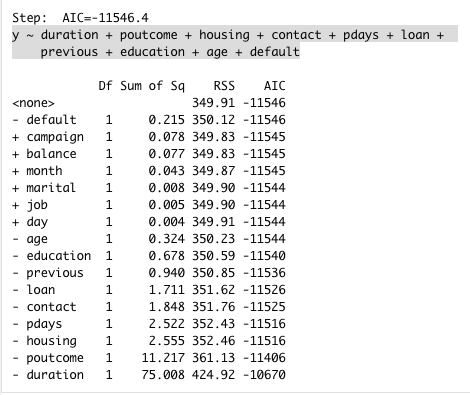
**Step 3f: Regression Methods - AIC stepwise**

AIC stepwise gives RSS numbers like the methods above, except that we make AIC start with essentially a NULL model, and then the AIC function adds variables one at a time to see what the AIC value is again. If it is lower, the function keeps going, if it is higher it stops. The AIC went through 11 iterations until it ended up with the lowest AIC shown – at AIC = -11546.4.

The final equation was: y ~ duration + poutcome + housing + contact + pdays + loan + previous + education + age + default.

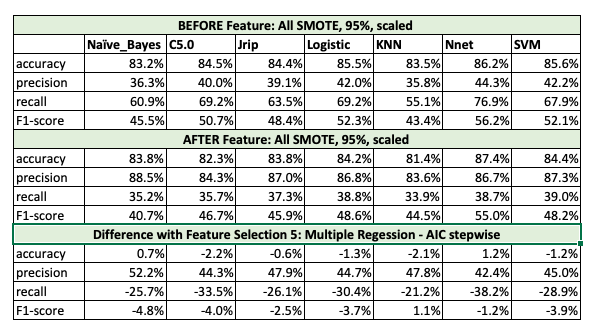
The final step is shown below in Figure 12, with the + sign in front of the variable meaning it should not be used and minus sign means these variables are significant.

*Figure 12 - AIC Stepwise Results*



With these 10 variables, it means we should delete 6 of them. We deleted the six, and ran the models before and after. As shown in Table 18 below, we see that the results of deleting all but 7 features gave worse numbers for each of the models, based on the F1 measure and Recall. All the precision values increased by at least 42.4% but recall decreased by 21.2% to 38.2%. We therefore do not recommend this feature on this particular dataset.

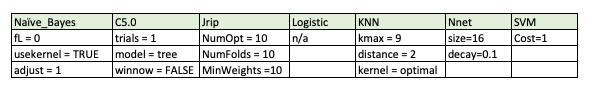
*Table 18 – Regression Results – AIC Stepwise results*



## Step 4: Model Parameters

In [22] the authors show a study of research papers and 14 out of 16 authors use parameter tuning as a method to get better model results. It was noted for the neural net that up to 10 million iterations would be performed! It should be noted that we let the model run ‘on its own’ to discover the best parameters and then we used those for further testing. Sometime we would try again to see if the parameters would change – and sometimes they did. However, for test purpose, we used the same set of parameters as noted below in Table 19. The table highlights the parameters available, and used, for the models in our study. The parameters will be discussed in the Models step.

*Table 19 – Model Parameters*



For SVM, the only option was the cost function, and we kept it to 1. On further thought, we would switch it to zero as we found the cost matrix for the neural net in the Step 2e did not improve the model. There were no options for Logisitic, except of course the “family=binomial(link="logit")” option for a glm model.

## Step 5: Models

Various papers [1-6] discuss the classification models that work well for this binary classification problem. Therefore, we chose models in line with these papers.

Before discussing the models, the testing process started with an imbalanced dataset. We created a data partition for the training and test sets (70%, 30% split). Then, if the balancing was performed on the data, a new dataset would be created from the training dataset. Either way, the training set was used in a 10-fold cross-validation, and repeated 10 times, to obtain a model. The model was then tested with the remaining test set (30% of the initial data, always imbalanced, noted above) to estimate the model performance. Further, a 10-fold cross-validation was run with the model and the test dataset to get an average F1-measure.

The models we want to use are called classification models, and with supervised training. The supervised training means that we know the ‘target’ response and we want to train the data to that target so as to obtain an algorithm that works well. The models used are: a) Naïve Bayes, b) Decision Tree, c) JRip, d) Logistic Regression, e) K-Nearest Neighbours, f) A neural network, and a g) a Support Vector Machine.

## 5A: Model 1 – Naïve Bayes (NB)

Naïve Bayes classifiers are a family of simple probability-based classifiers by using Bayes’ theorem with strong naïve independence assumptions between the features. A Naive Bayes classifier considers each of these attributes to contribute independently to the probability that the answer is a “Yes” regardless of any possible [correlations](https://en.wikipedia.org/wiki/Correlation_and_dependence) between the attributes. For categorical data, the probabilities computed use a discrete probability distribution function, while for numeric features, a continuous probability distribution function is used. The model’s final choices were *fL= 0, usekernel=TRUE, adjust=1.* We can tune the few hyperparameters that a naïve Bayes model has:

* *Usekernel=TRUE*parameter allows us to use a kernel density estimate for continuous variables versus a guassian density estimate,
* *adjust*allows us to adjust the bandwidth of the kernel density (larger numbers mean more flexible density estimate),
* *fL=*allows us to incorporate the Laplace smoother (1 yes, 0 no).

One additional issue to be aware of - since naïve Bayes uses the product of feature probabilities conditioned on each class, we run into a serious problem when new data includes a feature value that never occurs for one or more levels of a response class. What results is P(xi|Ck)=0 for this individual feature and this zero will ripple through the entire multiplication of all features and will always force the posterior probability to be zero for that class. A solution to this problem involves using the *Laplace smoother*. The Laplace smoother adds a small number to each of the counts in the frequencies for each feature, which ensures that each feature has a nonzero probability of occuring for each class. Typically, a value of one to two for the Laplace smoother is sufficient. The model chose fL=0 (No smoother) as the best choice.

## 5B: Model 2 – C5.0 Decision Tree (DT)

Decision trees (DT) are supervised classification algorithms which create a tree structure during the training phase. The algorithm can handle both numeric and categorical data. It builds DTs from a dataset using the concept of entropy, and in the tree induction phase separate mechanisms are used for handling numerical and categorical data: for numbers, a binary threshold is chosen, giving birth to two branches of the tree; in the latter, one branch is generated for each possible value of the categorical attribute. The algorithm then chooses at each node the attribute of the pattern that most effectively splits its set of samples into subsets. Or in another way of saying this, the decision tree algorithm splits up the data samples into two or more subsets so that the samples within each subset are more homogeneous than in the previous subset. This is a recursive process and the resulting two subsets in the binary tree decision are then split again, and so on, until the homogeneity criterion is reached.

The criterion of splitting is the difference in entropy (normalized information gain). The attribute with the highest normalized information gain value is chosen. The benefits of a Decision Tree model is that it can generate understandable rules, and as noted above, can handle both numeric and categorical data.

The parameters used are: *Trials=1, model=tree, winnow=FALSE*. Meaning run with one trial, model=tree is used instead of rules, and winnowing=FALSE means do not ‘winnow’ or remove predictors. This usually gives better accuracy.

The output from C5.0 shows the importance of the attributes, similar to the Random Forest in the Features section in Table 20 below. Note that ‘duration’ and ‘poutcome’ are 100% important then month at 77.5% and so on.

*Table 20 – C5.0 Model output*



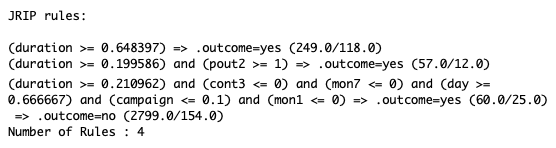
**5C: Model 3 – JRip – a Rules-based model**

JRip implements a propositional rule learner, called “Repeated Incremental Pruning to Produce Error Reduction” or “RIPPER”. It is based on association rules with reduced error pruning (REP), a very common and effective technique found in decision tree algorithms.

The options we had for this model included *NumOpt=10, NumFolds=10, MinWeights=10*. The NumFolds is used for REP, and 1 fold is used as a pruning set (default is 3). The MinWeights is to set the minimal weights of instances within a split (default is 2.0). The NumOpt is the number of runs of optimizations (default is 2).

One of the examples we obtained (95%, scaled numeric, and ‘dummy’ variables), was a simple set of 4 rules shown in Figure 13 below. The JRip produces rules that are fairly simple - if we look at the example below in Figure 13 – we see for example that as long as duration is over 0.64 minutes, then the outcome is a yes.

*Figure 13 – Jrip Rules*



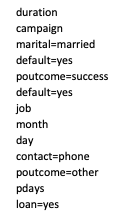
## 5D: Model 4 – Logistic Regression (LR)

Logistic Regression is similar to Multiple regression noted above, but there is are specific differences in that logistic regression expects a binary output for the target answer ‘yes”. Another advantage of logistic regression is that the independent variables do not have to be ordinarily distributed, or have equal variances in each group, and nonlinearity is assumed. Also, the independent variables can be numeric or categorical.

LR uses maximum probability estimation rather than the least squares estimation in traditional multiple regression. Starting values of the predicted parameters are used and the probability that the sample came from a population with those parameters is computed. The values of the estimated parameters are adjusted iteratively until the greatest probability value of them is obtained. That is, maximum probability approaches try to find estimates of parameters that make the data observed “most likely”.

The logistic model provides statistically significant attributes or components of the attributes as shown in Table 21 below.

*Table 21 – Logistic Model output*



## 5E: Model 5 - K- Nearest Neighbours (kKNN)

K-NN is a very simple supervised learning algorithm. For classification tasks, k-NN classifies a given instance based on the majority vote of its closest k instances. Close instances can be determined using some distance measurement or function like the Euclidean, Minkowsky or mini- max. K-NN does not need any model to fit therefore it is categorized as lazy learner, where the learner waits for the provided test dataset before doing any model generating. Lazy learners such as k-NN make less work when a training dataset is provided, and extra work when making a numeric prediction or classification [30].

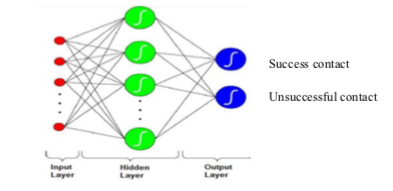
The parameters we had for this model were: 1) ‘*Distance=2’* which means to use the Minkowski distance, 2) ‘*Kernel= optimal’* is to decide what type of kernel function to use like “rectangular”, “Gaussian”, “optimal” etc. Optimal just chooses the best method, and 3) ‘*kmax =* 9’ is the number of nearest neighbours considered.

## 5F: Model 6 – Neural Networks (nnet)

In Figure 14 below the MLPNN structure is organized into layers of neuron's input, output and hidden layers. There is at least one hidden layer, where the actual computations of the network are processed. Each neuron in the hidden layer sums its input attributes xi after multiplying them by the strengths of the respective connection weights wij and computes its output yj using activation function (AF) of this sum. AF may range from a simple linear threshold function, or a sigmoid, hyperbolic tangent, or radial basis function [5].

Back-propagation (BP) is a common training technique for MLPNN. The available data set is normally divided into training and test subsets. BP works by presenting each input sample to the network where the output is computed by performing weighted sums and transfer functions.

*Figure 14 – Artificial Neuron illustration*



Weights wij adjusted to finding the minimum error E as fast, quickly as possible. BP applies a weight correction to reduce the difference between the network outputs and the desired ones; i.e., the neural network can learn, and can thus reduce the future errors. The performance of MLPNN depends on network parameters, the network weights and the type of transfer functions used [5].

The ‘*size=16’* is the number of units in the hidden layer, and ‘*decay=0.1’* is the weight decay threshold. This prevents the weights from growing too large. The default is zero.

## 5F: Model 7 – Support Vector Machine (SVM)

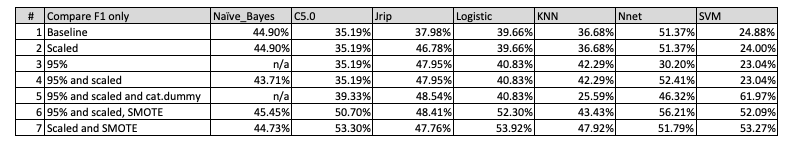
SVM is a supervised machine learning algorithm which can be used for classification or regression problems. It uses a technique called the kernel trick to transform the data into a higher dimension space and then based on these transformations it finds an optimal boundary between the possible outputs, which are points of data plotted in a multidimensional space representing the examples and their feature values. The algorithm involves a search for the maximum margin hyperplane (MMH) that creates the greatest separation between the two classes. The data may be linearly or nonlinearly separated.

The only parameter for this model is the Cost function (Cost=1). We had it defaulted to 1 – although we would make it zero if we used this model again because the cost matrix results for the ANN in Step 2e did not benefit the model (the cost matrix results were done later than other tests).

## Step 6: Results

In Step 1 we reviewed changes to the data itself and tested before, and after, these changes on the models, based on the F1 measure (although we do have the other data on Precision, Recall and Accuracy as well but not shown). The Table is shown again below as Table 22. We compared the 95% confidence interval (CI), scaled, and 95% + scaled to the base case. The scaled results were flat, except that JRip had an 8.8% improvement, and SVM a minor 0.88% drop. With the 95% CI, the results were flat for C5.0, JRip had a 9.97% increase, Logistic and KNN improved 1.17% and 5.61% respectively, Nnet dropped by 21.17% and SVM dropped by 1.84%. With 95% CI and scaling, the results were similar to just 95% CI, however NNet went back up to an F1 of 52.41%, even better than baseline. Naïve Bayes and SVM were slightly negative again, but overall the results were good. Then, using 95% and scaled, before and after, changing the categorical data to numeric, the categorical category increased slightly for C5.0 ( 4.14%) and JRip ( 0.59%) but worked very well for SVM model – there was an improvement of 38.94%. However, there was a large drop for KNN of 16.7%, and Nnet dropped 6.09%, so overall not good results as a whole. Although we added SMOTE in the tests, we will discuss the rebalancing results in Step 2.

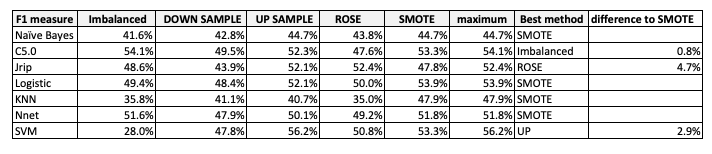
*Table 22 – Highlights of Step 1 – Pre-Processing the Data*



Therefore, the results showed that scaling on its own was flat for most and positive for JRip, 95% CI on its own was positive for most but negative for NNet with a 21.17% drop, 95% plus scaling had positive results, categorical ‘dummy’ variables worked very well for SVM model.

In Step 2 we reviewed rebalancing the data and tested before, and after, these changes on the models. Table 8 in Step 2 highlights the metrics used, however, we have replicated just the F1 measures below in Table 23 to highlight the differences.

*Table 23 -* *Highlights of Step 2 – Rebalancing Methods*



We notice that SMOTE rebalancing tends to be the best rebalancing method overall, however Imbalanced remains slightly higher for C5.0 (0.8%), ROSE worked well for JRip (+4.7% over SMOTE), and SVM worked best with up-sampling (2.9% better than SMOTE). However, since we want to use the same rebalancing method for all the models, we chose SMOTE for all the latter tests.

In Step 3 we reviewed and showed the results of the reduction algorithms and will summarize the results here.

The PCA analysis did not give us any insights. We think that because the data itself is somewhat convoluted, it would take almost all the features to represent the ‘answer’, especially for the minority class, and so PCA just didn’t work well. The Figure 8 shows the PCA response well, in that over 10 variables are required to explain 80% of the response, which doesn’t reduce the number of variables by much.

With Recursive Feature Elimination (RFE) + Random Forest, and using only 5 variables (duration, month, poutcome, day, contact) in the models, then comparing to a model with all the attributes, the results were all positive with an overall increase of 1.9% to the F1-score as shown in Table 11. We recommend using this feature – and deleting 11 variables!

In using Boruta, and deleting only 2 variables (job and loan), five of the F1 model results were negative, and there was an average loss to the F1-score of 0.46% overall as shown in Table 13. We do not recommend this algorithm.

With the ANN called LVQ, we saw the importance of the values in Table 14, and decided to only use the 7 highest important variables (duration, previous, contact, pdays, poutcome, housing, balance). The results were all negative for the models as shown, as compared to running the same models with all the variaibles, in Table 15, and we therefore do not recommend using this method.

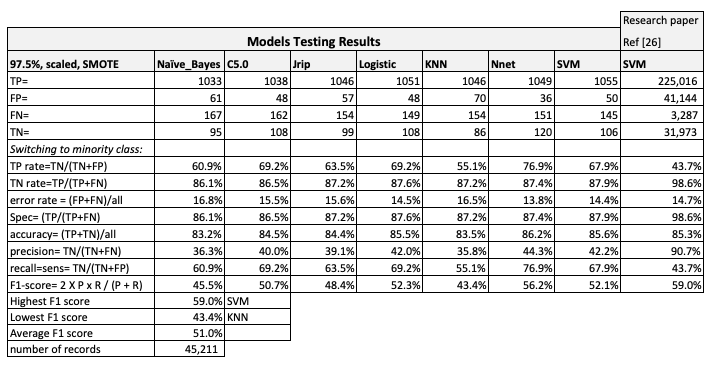
In testing with the regression methods ANOVA and/or AIC (same results in this case), we found that Model 10 was the best with 7 variables (poutcome, month, contact, default, education, job, and duration) and the results were all positive for the models – except for Nnet at -3.2% - as shown in Table 17. However, we would recommend this method.

Finally, using AIC stepwise, the model showed the top 10 variables (duration, poutcome, housing, contact, pdays, loan, previous, education, age, and default). In using those variables in the model, and comparing the ‘before’ data (with all 16) and the ‘after’ data using just 10 variables, all the test results were negative, except for KNN with +1.1% as shown in Table 18. We therefore do not recommend using this method.

In Step 4, we highlighted the model parameters used. We ran a set of possible parameters and then the model chose the best parameters. With those chosen parameters, we then ran all the tests with the same parameters.

In Step 5 we discussed the models and the parameters but did not show any model results – we will do that in this section. All the work in the other sections was to get to this stage in comparing the models. We therefore used the 95% CI, scaled the data and used SMOTE as the rebalancing technique for *all the models*, to be consistent in comparing them. The results are shown in Table 24 below.

*Table 24 – Model Results*



Note that we ‘switched’ to calculating the metrics based on the minority class having the “True Positive” instead of the majority class. The calculations were shown in Step 0, since we needed to show results along the way, and used the F1 measure to highlight the differences in step 1. However, we do note that all the four metrics – Accuracy, Precision, Recall, and the F1 measure – are equally important in figuring out if the metrics are “good” or not.

Reviewing the metrics above, we see that the accuracy of the models is all in the 80% range – from 83.2% for Naïve Bayes to 86.2% for Nnet. Precision is low – from 35.8% for KNN to 44.3% for Nnet. Recall runs from 55.1% for KNN up to 76.9% for Nnet. And final, the F1-score is 43.4% for KNN up to 56.2% for Nnet. Therefore, running NNet model seems to be the best model.

We would further add that we copied in the results of a confusion matrix from reference [26] in Table 24 on the far right, using the same data provided by the Portuguese bank. Although the researchers used over 5 times the number of data points, our results for the SVM model compared to theirs is not that much worse. IN our model, the accuracy was better (+0.3%), recall was better (24.2%), but the precision was worse (48.2%) but the final F-score was only 6.9% lower. And in [27] different authors provided F-measures, again for the same dataset, without normalization, of 44.52% for Naïve Bayes (ours is +0.98%), 28.25% for C4.5 algorithm, 60.12% for the ANN (ours is -3.92%), and 42.60% for the SVM (ours is +9.5%). Therefore, in comparison, our results are good here as well.

## Step 7: Conclusion

We have been able to show that, combined, using the 95% CI with scaling and with the SMOTE rebalancing method, all provided overall good results for the various models. We would also recommend using ‘dummy’ variables for the categorical features, if SVM model is going to be used.

We also highlighted that using feature selection methods such as Recursive Feature Elimination (RFE) + Random Forest, multiple regression with ANOVA or AIC all show positive results in reducing the number of variables down to 5 or 7, respectively, with the first three variables being the same, as shown below.

* RFE + RF: duration, poutcome, month, contact, day
* ANOVA: duration, poutcome, month, contact, default, education, job

Finally, in using the knowledge carried out in steps 1 to 5, we obtained the metric results of the 7 models, and although all had respectable results, the Neural Net model had the best results to use in banking applications for lending products. The user might want to also use a second model along with this model, such as a Decision Tree or Rules-based algorithm, so that the results can be better understood. As much as ANNs and SVMs in general perform better than other algorithms, their results are harder to understand.

## References

1. Moro, S., Laureano, R., and Cortez, P. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. P. Novais et al (Eds.), Proceedings of the European Simulation and Modelling Conference – ESM’2011, pp. 117-121, Guimaraes, Portugal, October, 2011.
2. Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. Decision Support Systems, 62, 22-31.
3. Moro, S., Cortez, P., & Rita, P. (2015). Using customer lifetime value and neural networks to improve the prediction of bank deposit subscription in telemarketing campaigns. Neural Computing and Applications, 26, 131–139.
4. Moro, S., Cortez, P., & Rita, P. (2018). A divide and conquer strategy using feature relevance and expert knowledge for enhancing a data mining approach to bank telemarketing. Expert Systems, 35(3).
5. Elsalamony, H. Bank Direct Marketing Analysis of Data Mining Techniques. International Journal of Computer Applications, V85, 12-22, 2014.
6. M. Purnachary, B. Kumar and H. Shaziya. Performance Analysis of Bayes Classification Algorithms in WEKA Tool using Bank Marketing Dataset. International Journal of Engineering Research in Computer Science and Engineering (IJERCSE), V5, Issue 2, February 2018.
7. P. Ładyzynski, K. Zbikowski and P. Gawrysiak. Direct Marketing Campaigns in Retail Banking with The Use of Deep Learning and Random Forests. Expert Systems with Applications, V134, 28-35, 2019.
8. J. Li, K. Cheng, S. Wang et al. Feature Selection: A Data Perspective. ACM Computing Surveys, Vol. 50, No.6 Article 94. Dec 2017.
9. M. Alibeigi, S. Hashemi, A. Hamzeh. Unsupervied Feature Selection Based on the Distribution of Features Attrbiuted to Imbalanced Data Sets. International Journal of Artificial Intelligence and Expert systems (IJAE), vol.2, issue 1. 2011.
10. S. del Rio, V. Lopez, J. Manuel et al. On the use of MapReduce for imbalanced big data using Random Forest. Information Sciences, 11/2014, Volume 285
11. S. Cateni, V. Colla, M. Vannucci. A method for resampling imbalanced datasets in binary classification tasks for real-world problems. Neurocomputing, 07/2014, Volume 135
12. C. Beyan, R. Fisher. Classifying imbalanced data sets using similarity based hierarchical decomposition. Pattern Recognition, 48 (5) (2015), pp. 1653-1672
13. Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. Expert Systems with Applications, 73(Complete), 220–239.
14. V. Lopez, A. Fernandez, S. Garcia, V. Palade, F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics.Information Sciences, 250 (2013), pp. 113-141
15. Miguéis, V.L., Camanho, A.S. & Borges, J. Predicting direct marketing response in banking: comparison of class imbalance methods. Serv Bus 11, 831–849 (2017).
16. Sáez J.A., Krawczyk B., Woźniak M. Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets.Pattern Recognition, 57 (2016), pp. 164-178
17. Marinakos, G., & Daskalaki, S. (2017). Imbalanced customer classification for bank direct marketing. Journal of Marketing Analytics, 5(1), 14-30.
18. Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence, 5(4), 221–232.
19. Jena, P. C., Ojha, A. C., & Pani, S. K. (2015). Cost-sensitive modeling to predict bank telemarketing campaign. IUP Journal of Information Technology, 11(2), 39-47.
20. Ghatasheh, N., Faris, H., AlTaharwa, I., Harb, Y., & Harb, A. (2020). Business analytics in telemarketing: Cost-sensitive analysis of bank campaigns using artificial neural networks. Applied Sciences, 10(7), 2581.
21. F. Cheng, J. Zhang, C. Wen, Z. Liu, Z. Large cost-sensitive margin distribution machine for imbalanced data classification.Neurocomputing, 224 (2017), pp. 45-57
22. S. F. Crone, S. Lessmann, and R. Stahlbock. The impact of preprocessing on data mining: an evaluation of classifier sensitivity in direct marketing. In European Journal of Operational Research, vol. 173, no. 3, pp. 781–800, 2006.
23. <http://archive.ics.uci.edu/ml> (The UCI Machine Learning Repository collection of datasets).
24. Silipo, R. Adae, I., Hart, A., Berthold, M. Seven Techniques for Dimensionality Reduction. KNIME, 2014.
25. Kawasaki, Y., Ueki, M., (2015). Sparse Predictive Modeling for Bank Telemarketing Success Using Smooth-threshold Estimating Equations. In Journal of Japan Society of Computational Statistics, V28, pp.53-66.
26. Moro, S., Laureano, R., Cortez, P. Enhancing Bank Direct Marketing through Data Mining. EMAC 2012 Conference.
27. Koumétio, C. S. T, Cherif, W., & Hassan, S. (2018, October). Optimizing the prediction of telemarketing target calls by a classification technique. In 2018 6th International Conference on Wireless Networks and Mobile Communications (WINCOM) (pp. 1--6). IEEE.
28. Jin, W., He, Y. Three data mining models to predict bank telemarketing. In IOP Conf Series: MAerials Science and Engineering. V490 (2019).
29. M. S. Islam, M. Arifuzzaman and M. S. Islam, "SMOTE Approach for Predicting the Success of Bank Telemarketing," 2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), Bangkok, Thailand, 2019, pp. 1-5, doi: 10.1109/TIMES-iCON47539.2019.9024630.
30. D. W. Aha, D. Kibler, and M. K. Albert, Instance-based learning algorithms, Machine learning, vol. 6, no. 1, pp. 3766,1991.