Using Supervised Machine Learning Algorithm to Predict Customer Response

Problem Definition and Exploratory Data Analysis

Sony JUFRI, Sydney, 8 April 2020

Problem Definition

Marketing spending in the banking industry is massive. The big four banks each spent around $1-10 million on Google Ads between April and October last year. Other banks like AMP and Macquarie spent between $100,000 and $1 million during the same period[1]. The big spending however, does not always translate to higher sales. In marketing, if you try to reach everyone, you can spend a lot of money and resources but still not getting the right outcome for your business. Therefore, it is important for the banks to optimize their marketing strategies and improve effectiveness by targeting the right customers.

The Bank Marketing dataset relates to direct marketing phone campaigns of a Portuguese bank. It has reasonably good size (41,188 examples x 20 attributes) and complexity, with many missing values, outliers and imbalanced dataset. Whilst many topics could be explored from this dataset, for this project, we will focus on three things: (1) to have a better understanding of the customer base and identify which attributes are critical to the campaign success, (2) build machine learning models to predict customers' response (whether they will subscribe to a term deposit as a result of a campaign), and (3) determine the best model to use (Research Question: Is the recommended model better than the alternatives considered? H_0: the recommended model's F1-score is not better than the alternatives).

It is hoped that the analysis would be useful not only for the Portuguese bank, but also for other banks around the world, with some key learnings could potentially be applied to other financial companies.

Exploratory Data Analysis

The data selected was publicly available and downloaded from the UCI Machine Learning Repository[2]. It contains historical data of a Portuguese bank's marketing campaigns from May 2008 to November 2010, promoting their term deposit products to existing customers. Upon loading the dataset onto Jupyter notebook, an exploratory data analysis was performed using numpy, pandas, matplotlib and seaborn libraries in python. There are 41,188 examples in this dataset, each represents an existing customer that was contacted during the campaigns. For each example, it records seven attributes relating to customer demographics (such as age and education), nine relating to the campaigns (contact type, duration, etc.) and five to socio-economic features (CPI, employment rate, etc). Ten of them are numericals, whilst the rest are categoricals, with one output variable indicating if a customer subscribed to a term deposit.

In the categorical variables, some missing ('unknown') values were noted and a number of different approaches were used to tackle these issues: (1) the small number of unknowns in 'marital' and 'education' (less than 0.8%) were removed from the dataset; (2) the unknowns in 'default' status were quite high (20%) because many customers might not want to disclose this information to the bank. Because removing a large amount of data can lead to loss of information and poor predicting outcomes, it was decided to keep them in the dataset as a separate category; (3) the other significant ones were 'housing' (984), 'loan' (984) and 'education' (1,722). For these variables, different imputation strategies were applied to replace the missing values (see appendix B for detail). Lastly, a small amount of outliers in 'education' (18 illiterate customers) were removed to further reduce noise and improve the dataset.

For the numerical variables, statistical analysis was performed using pandas and seaborn. It can be seen that the distribution of the customer 'age' is fairly normal with small standard deviation. The majority of customers contacted were between age 30-40, but some outliers were identified (age 60+ with the

oldest age 98). Because they were mostly retirees with high level of subscription rate, it was decided to keep them in the dataset and a new 'age group' column was created to address this issue (see below).

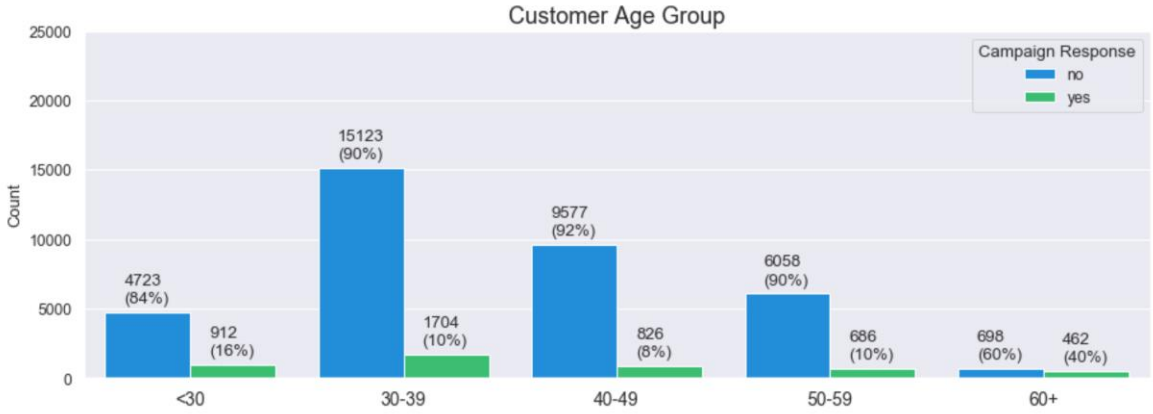

Figure 1. Number of contacts made and its success rate based on the customer age group

Outliers were also identified in the 'campaign', 'previous' and 'duration' variables. As they were only a small amount (but can negatively impact the model), it was decided to remove them from the dataset.

From the exploratory analysis, it seemed that students (31% success rate) and retirees (25%) were more likely to subscribe to a term deposit. This is consistent with the above chart, where customers under age 30 (16%) and 60+ (40%) showed higher success rate. From the campaign data, it appeared that cell phone calls (15%) were more effective than home phone calls (5%), and customers contacted in the month of March (51%), September (45%), October (44%) and December (49%) responded better to the campaigns. Lastly, from the heat map, call duration (0.42) and number of days since last contact (-0.32) appeared to have higher correlation to customer response compared to other numericals in the dataset. Further analysis into this seemed to suggest that when calls made were less than six times and last contact was less than 16 days ago, higher success rate was achieved (see Appendix A for more detail).

Once the dataset was cleaned, the LabelEncoder class from SciKit Learn was used to convert the categorical variables into numbers, so the predictive model can better understand them. Subsequently, the sklearn MinMaxScaller[4] was used to normalize the data to address the scaling issue in the numerical variables. Lastly, because the dataset is imbalanced (only 11% customers subscribed to a term deposit), the SMOTE[3] sampling method from imblearn library was used to synthesize new samples from the minority class ('Yes') to balance the data (so the number of 'Yes' are equal to the number of 'No'). This will help the model learn both classes ('Yes' and 'No') better and yield better predictions on new data.

Approach

In stage two of the project, different statistic and modelling techniques would be used to answer the research question outlined earlier. Machine learning algorithms such as logistic regression, k-nearest neighbor and random forest classifier would be used to fit the data and create models to predict customer response to the campaigns. Valuation methods such as Cross-validation and Grid Search would be implemented to evaluate and improve the models, and performance measures such as Precision, Recall and F1-score, as well as statistical tests would be utilised to compare their performance. It is expected that the resulting model would not only help the bank predict new customers' response in future campaigns, but also help them understand which attributes are critical to the campaign success.

References

1. Bank Marketing Data Set, UCI Machine Learning Repository, viewed 14 March 2020, https://archive.ics.uci.edu/ml/datasets/Bank+Marketing
2. "How much are big banks spending on ads in the post-Royal Commission era?", Marketing Magazine, Josh Loh, 4 December 2019, https://www.marketingmag.com.au/news-c/news-semrush-banking-search-spend/
3. "SMOTE Oversampling for Imbalanced Classification with Python", Machine Learning Mastery, Jason Brownlee,17 January 2020, https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
4. "Scale, Standardize, or Normalize with Scikit-Learn", Towards Data Science, Jeff Hale, 5 March 2019, https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02
5. "5 Ways To Handle Missing Values In Machine Learning Datasets", Analytics India Magazine, Kishan Maladkar, 9 February 2018, https://analyticsindiamag.com/5-ways-handle-missing-values-machine-learning-datasets/

Appendix A – Examples of Data Visualisation

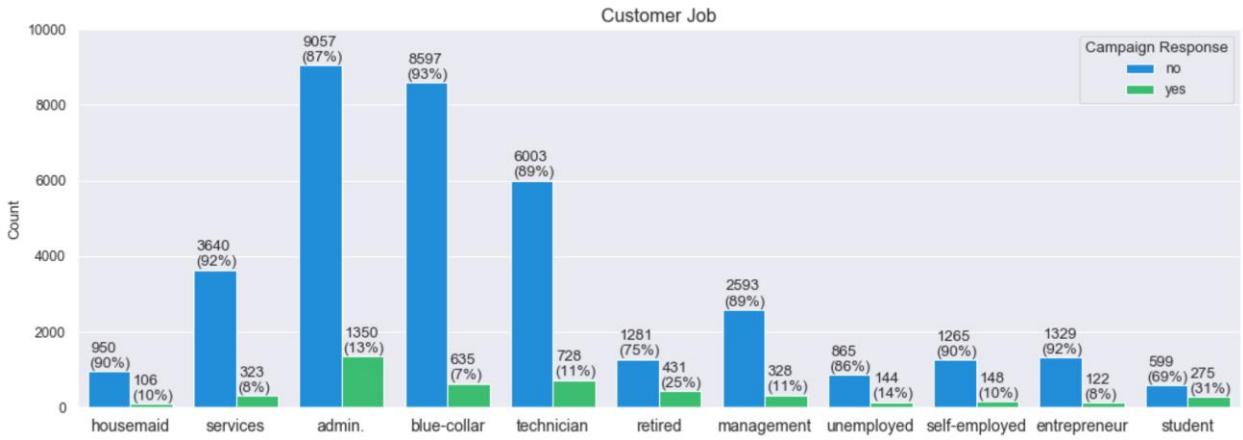

Figure 2. Number of contacts made and its success rate based on customers' jobs

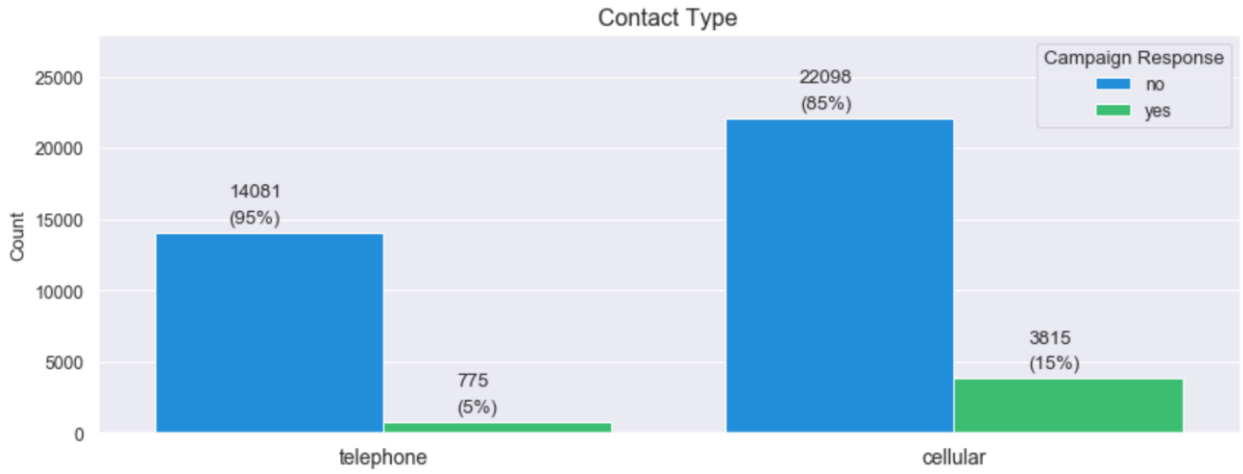

Figure 3. Number of contacts made and its success rate based on contact types

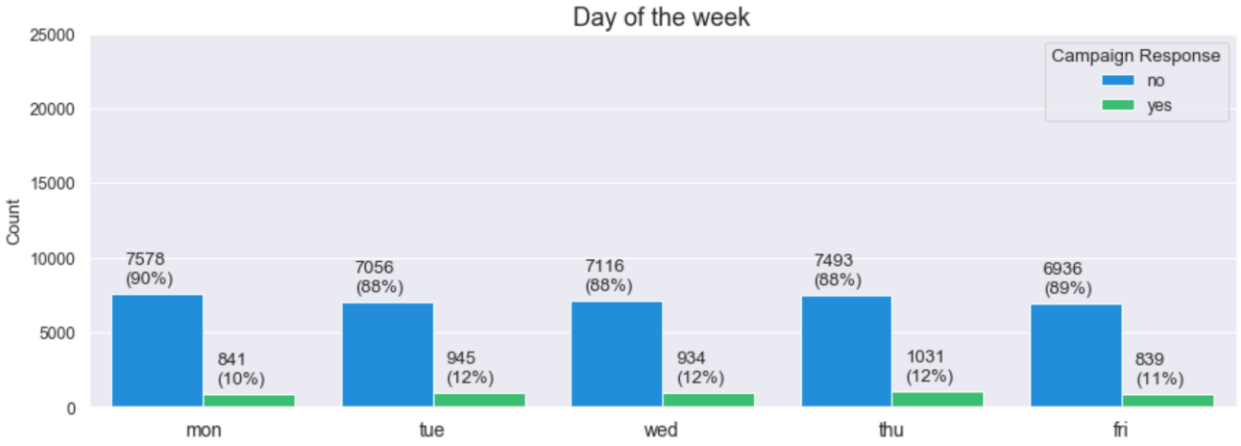

Figure 4. Number of contacts made and its success rate based on the day the customer was contacted

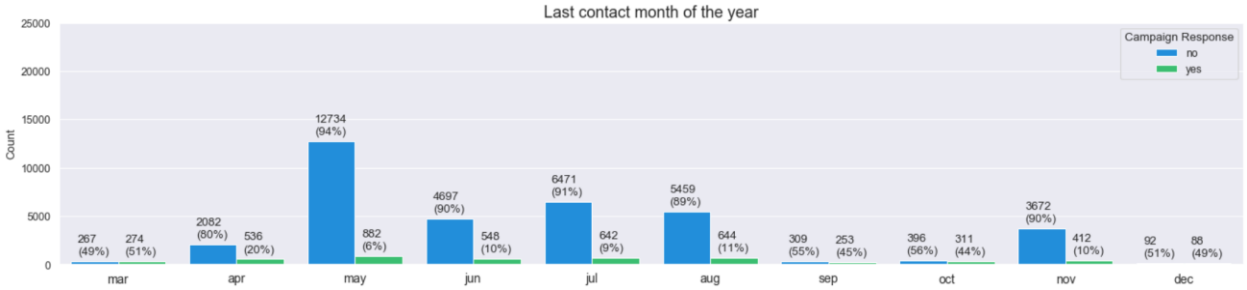

Figure 5. Number of contacts and its success rate based on the month the customer was contacted

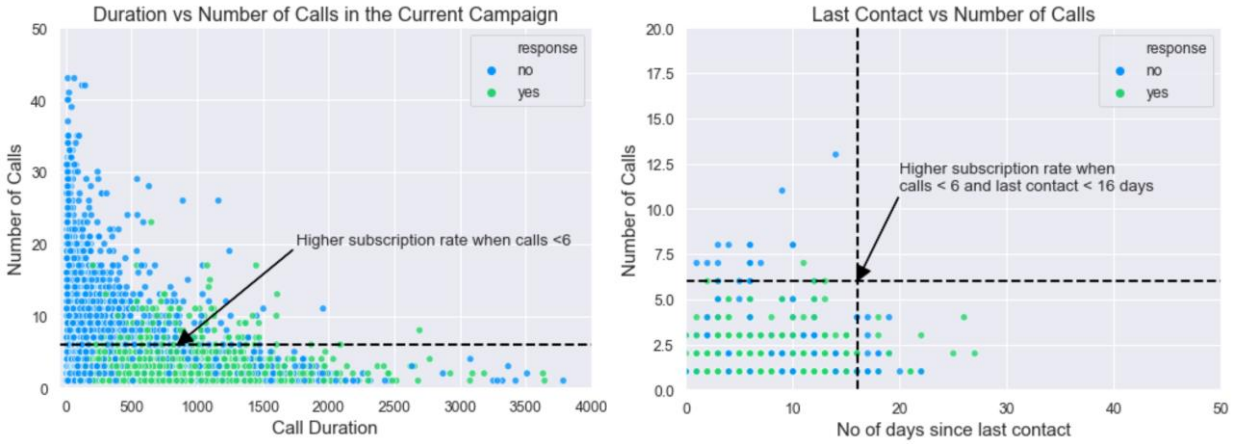

Figure 6. Call duration vs Number of calls vs No of days since last contact

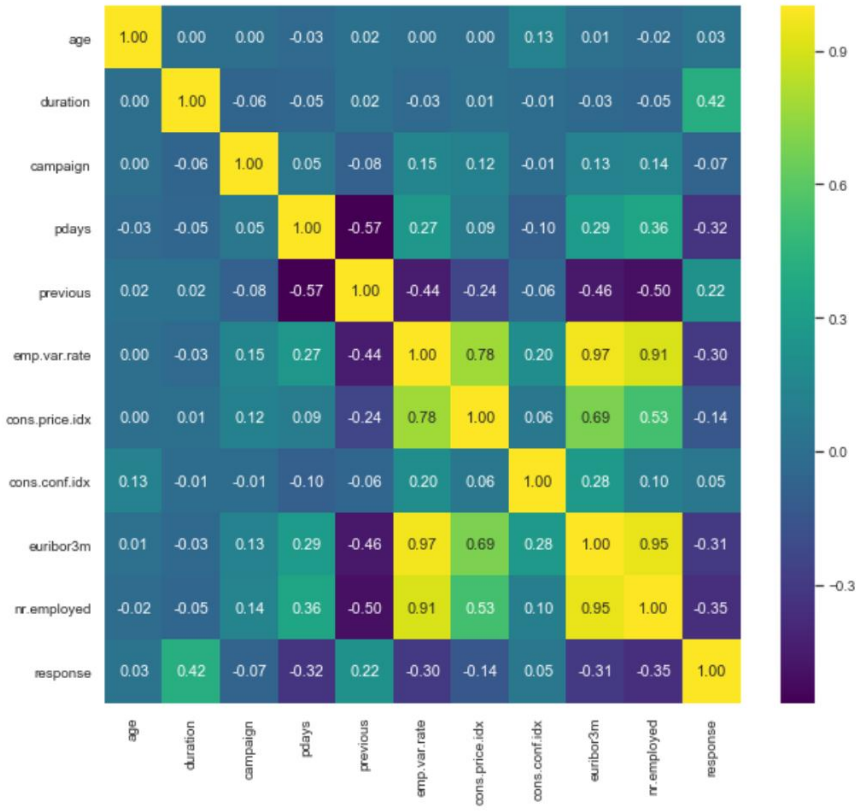

Figure 7. Heat map showing correlation coefficient of numerical variables and the response variable

Appendix B – Imputation Strategies to Replace Missing Values

For 'education', cross-tabulation between 'education' and ‘job’ were used to impute the missing values. The assumption here is that 'job' is influenced by the 'education' of a person, and therefore, someone’s education can be inferred by their job (eg. people in management jobs usually have a university degree, hence all customers with management jobs were assigned to ‘university.degree’).

A similar approach were applied to the ‘housing’ and ‘loan’ variables. The unknowns here were assigned to ‘Yes’ and ‘No’ so that the proportion in each job category remains the same (eg. 55% technicians in the dataset have a housing loan, so 55% of the unknowns that are technicians were assigned to ‘Yes’).

Appendix C - Examples of Python code

Example 1. Describing statistics for the numerical variables

```
In [6]: # check the statistics of the numerical attributes (mean, std, min, max, etc)
        df.describe()
```

Out[6]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

See the python code submission for complete coverage of the codes