Predicting Client Response to Marketing Campaign using Machine Learning

Thinakone Louangdy

February 2024

1 Introduction

Given a set of data of clients from a certain banking details and features, we want to be able to predict if the client will response to a bank term deposit or not. The data that we will used is related to direct marketing campaigns of banking institutions.

However, why resolve this issue? During a phone campaign, the marketing center frequently has to make a lot of calls to consumers in order to get a reaction. The amount of time spent on the phone is frequently far greater than the quantity of favorable answers. To save time, we will first identify those who will respond in the affirmative before contacting them.

We will employ machine learning techniques to find a solution to this issue. Since there are many clients and hence many distinct circumstances, it is impossible to create a rule or piece of code that can be customized for every client. It will take too much time to maintain and write. We are looking for an algorithm that can quickly generalize to new data after learning from a set of existing data. We are employing machine learning techniques for this purpose.

We will primarily utilize conventional models such as Logistic Regression and Random Forest from the sklearn[1] library for our tasks. Our approach will be grounded in the use of these more traditional, interpret-able models rather than complex neural networks.

2 Data Selection

We have a dataset that contains various variables in it. We have:

- 20 features
- 1 target value (binary)

The objective for this task is to check which variables are meaningful to our study. First, we can safely remove **duration** as it will results our target as 'no' because the duration is not known before the call is performed. The objective of our study is to put a label on the customer responses before the call has been performed.

After our first observation on our dataset, we can separate our data into two main groups:

- qualitative variables (categorical)
- quantitative variables

[1]Scikit-learn, often abbreviated as "sklearn", is a popular Python library for machine learning. It provides a wide range of supervised and unsupervised learning algorithms via a consistent interface

2.1 The qualitative variables

For each variable we will take a look at their distribution toward our target variables. We have to check if we have any imbalanced variables that could contribute to our model performance or not. But in our dataset, we will not discard any categorical variables as we can see that there is a small proportion toward 'yes' (figure 1). Our target variable can therefore be explained by the variable having a label with a limited population.

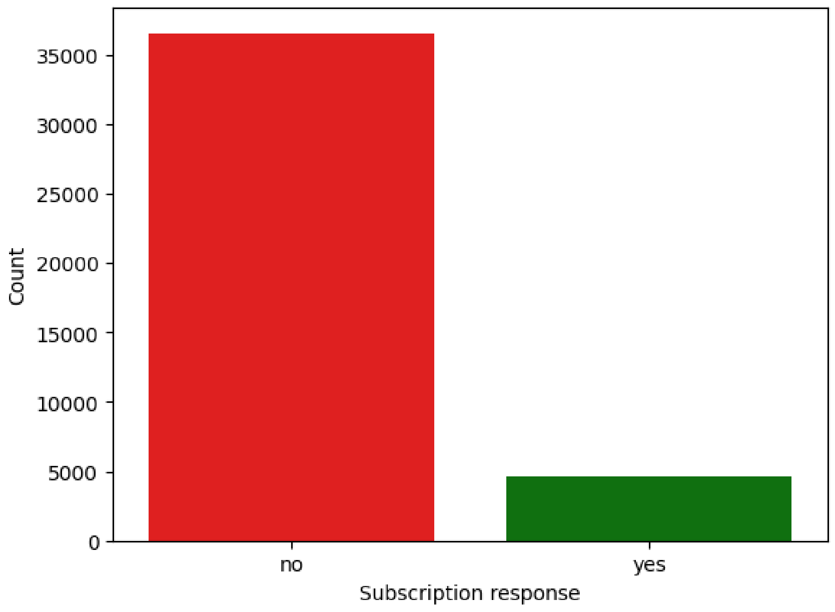


Figure 1: Target Distribution

2.2 The quantitative variables

We will take a look at a variances of each quantitative variable that is related to the information that provided from the dataset. We then can discard variables that are not significant for our target.

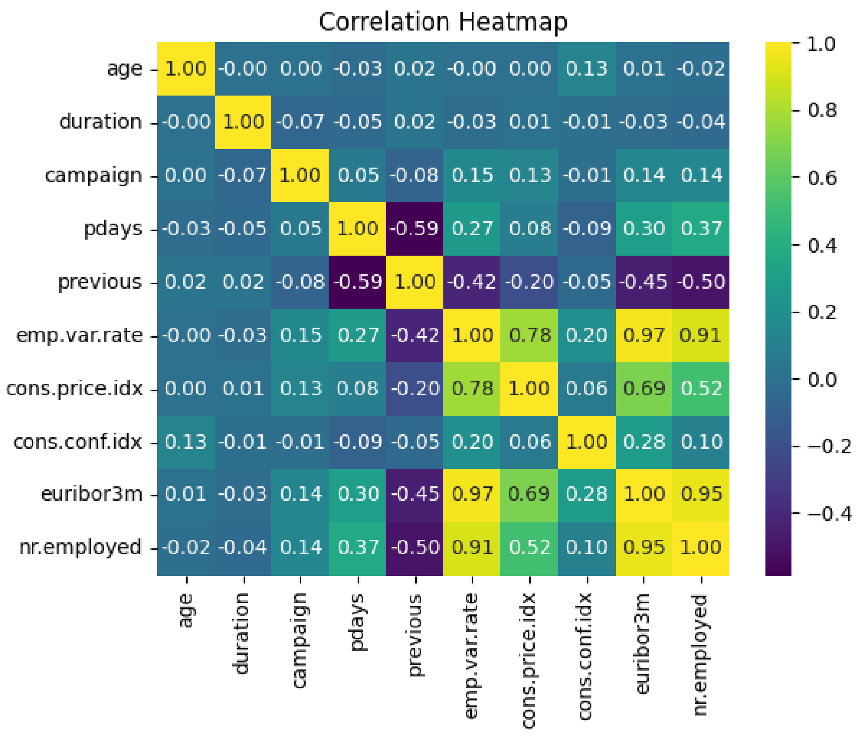


Figure 2: Heatmap Correlation

We also check the correlation of our data and we can see from figure2 that there is not enough correlation from **cons.price.idx**, **emp.var.rate**, **euribor3m**, **nr.employed** so we decided to discard those variables.

3 Data Preprocessing

After we selected our variables we will do some pre-processing. While not strictly distance-based, we picked model that uses gradient descent optimization, which can benefit from standardized features. The normalization applied by StandardScaler ensures features with large variations don't dominate the optimization process, leading to quicker convergence and a more stable training process.

We also have to deal with the categorical variables and convert it into numerical data before we can proceed to fit into our machine learning model. This is when we apply **one hot encoding** to transform those value into numerical data, but pay close attention to this technique, it might increase our data's demensionality.

After those steps above, we will now have a data in numerical format:

- 41188 individuals
- 50 variables

4 Model Selection

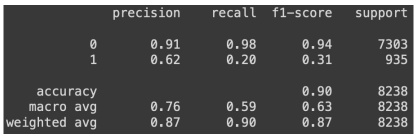
Determining whether a marketing offer will be accepted (”yes”) or rejected (”no”) by customers is a difficulty. We decided to look into Random Forest and Logistic Regression to implement our model due to these models seems to be the best fit for our dataset, which has intricate patterns that are difficult for straightforward prediction techniques to handle.

Specifically, contrary to what simpler models assume, customer decisions don't always follow clean, obvious trends. Furthermore, a multitude of distinct bits of client data influence their decisions. Because they are made to perform well in precisely these kinds of circumstances, logistic regression and random forest are attractive options for producing accurate forecasts.

For parameters of each model we will use the classic and initial value from the model itself to see performance, but we will adjust some random state for our model reproducibility and add "max_iter" to logistic regression to make the iteration involves updating the model's coefficients based on the training data.

5 Model Evaluation

After constructed the model, we have to evaluate their performance. There are different ways to do this, we will start by looking at the confusion matrix of an overall and each class. First attempt, we got a scores as below:

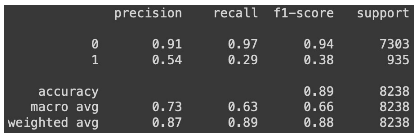


```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      7303
           1       0.62      0.20      0.31       935

    accuracy                           0.90      8238
   macro avg       0.76      0.59      0.63      8238
weighted avg       0.87      0.90      0.87      8238
```

(a) Score of Logistic Regression



```
              precision    recall  f1-score   support

           0       0.91      0.97      0.94      7303
           1       0.54      0.29      0.38       935

    accuracy                           0.89      8238
   macro avg       0.73      0.63      0.66      8238
weighted avg       0.87      0.89      0.88      8238
```

(b) Score of Random Forest

Figure 3: Grouped Classification Report

As we can see that in figure 3 and figure 4 the score are pretty high toward class "0" and making it really bad for class "1" that was the imbalanced of two classes.

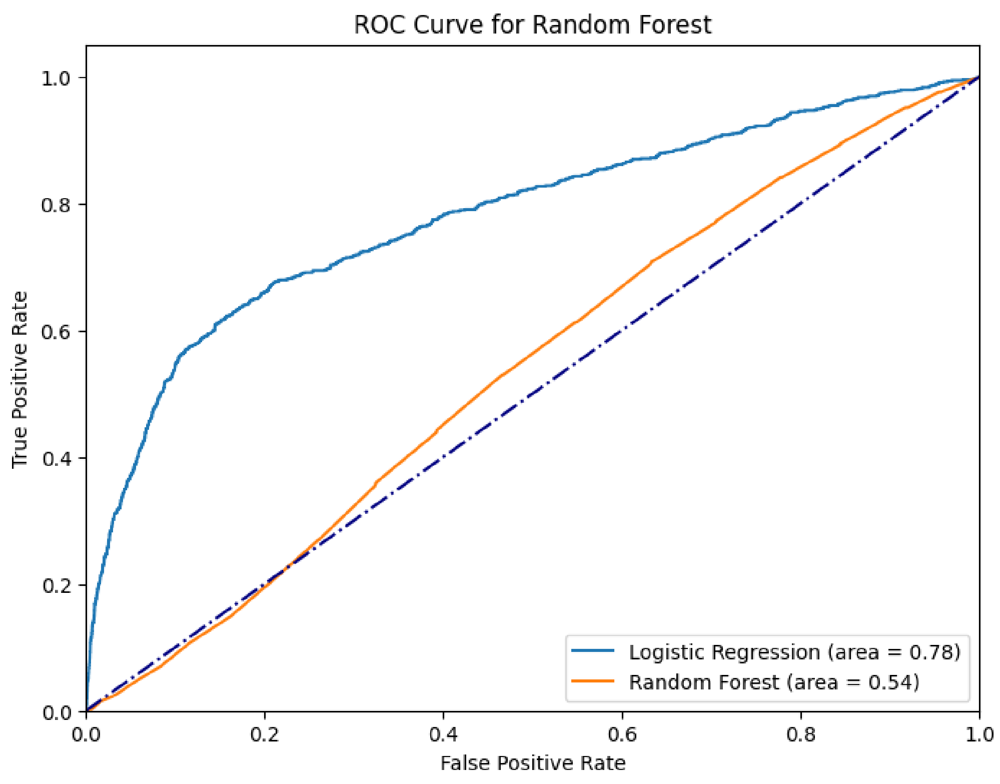
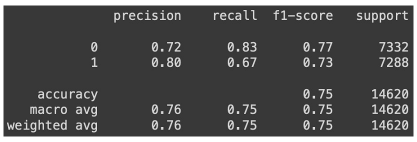


Figure 4: ROC Graph for both Models

Next, we have applied the oversampling method to our dataset to generated the balance data for both classes. Oversampling is a general term for techniques that address class imbalance by creating new samples for the minority class(es). We will use **SMOTE** which stand for Synthetic Minority Oversampling Technique to generated the new, pausible sample for the minority class instead of duplicating the existing sample. Then we simply go through the pre-processing step again which is scaling the features and then retrain our models. We can see that our model has improved in terms of classification score and ROC score as you can see at figure 5 and figure 6 below:

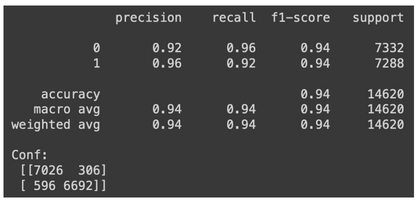


```
              precision    recall  f1-score   support

           0       0.72      0.83      0.77      7332
           1       0.80      0.67      0.73      7288

    accuracy                           0.75     14620
   macro avg       0.76      0.75      0.75     14620
weighted avg       0.76      0.75      0.75     14620
```

(a) Score of Logistic Regression after Resampling



```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      7332
           1       0.96      0.92      0.94      7288

    accuracy                           0.94     14620
   macro avg       0.94      0.94      0.94     14620
weighted avg       0.94      0.94      0.94     14620

Conf:
 [[7026  306]
 [ 596 6692]]
```

(b) Score of Random Forest after Resampling

Figure 5: Grouped Classification Report after applied oversampling

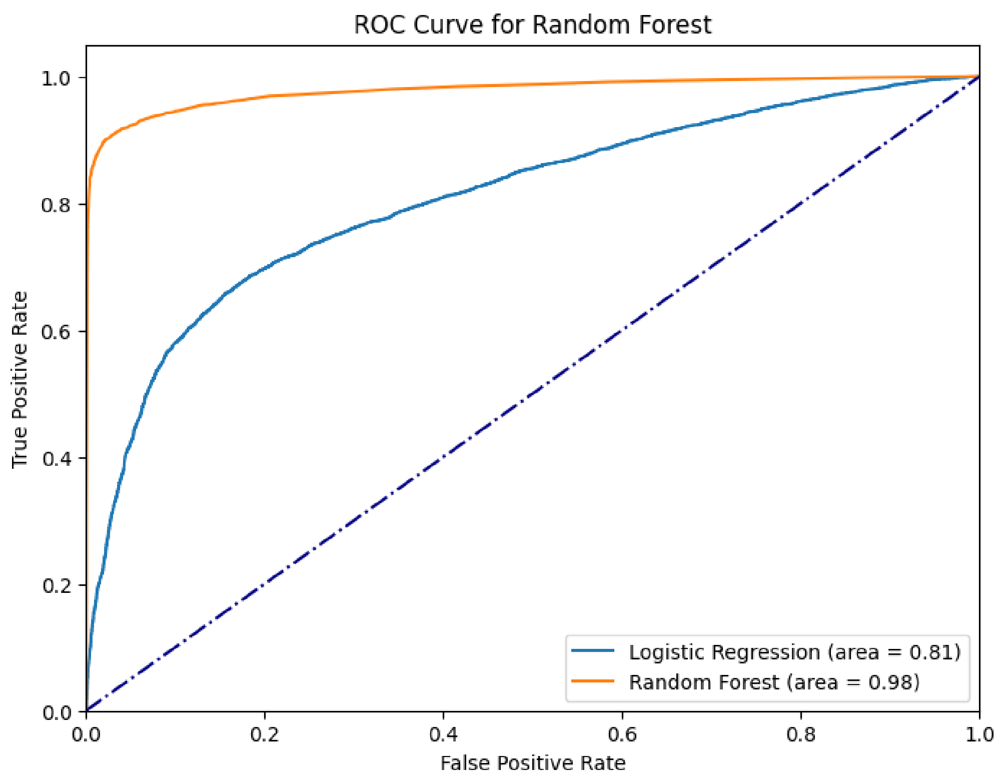


Figure 6: ROC Graph for both Models after applied ovrsampling

6 Conclusion

This study explored the application of logistic regression and random forest models to a dataset exhibiting a class imbalance. Key scientific bottlenecks and resolutions included: **Class Imbalance**, the inherent class imbalance posed a challenge to achieving optimal model performance. This can lead to models biased towards the majority class. **Resolution**, we applied oversampling technique to mitigate this, the Synthetic Minority Oversampling Technique (SMOTE) was employed. SMOTE effectively balanced the dataset by generating synthetic samples for the minority class, enhancing the models' ability to learn from its patterns. Following data pre-processing and the application of SMOTE, both logistic regression and random forest models were trained and evaluated. Experimental results demonstrated that the random forest model excelled in overall performance metrics. This superiority is likely attributed to random forests' inherent ability to handle non-linear relationships and their robustness to over-fitting.