Aalto University
CS-C3240 - Machine Learning

Using Logistic Regression and Decision Tree to analyze how customers responds to the last marketing campaign

October 2022

1 Introduction

The machine learning application for this project is very straightforward to business perspective. This project is excellent evidence of how a marketing analyst can benefit from using customer data to predict their behaviors, their needs as well as the trending marketing channels, they frequently interact.

The report is constructed into four sections: problem formulation, method, result and conclusions. The first section starts with problem formulation and then presents dataset as well as the meaning of features and labels. The data preprocessing, feature selection, and modeling is discussed in the second section. The next section is to explain the result where the outputs of the used models are evaluated and then suggest the best fitting model. The conclusion as the last section gives summary of the results and discuss limitations of the different models and improvements.

2 Problem formulation

2.1 Project aim

The goal of this machine learning project is to help the marketing manager to predict customer response to the recently implemented marketing campaign based on their personal information. The Scikit learn package are used to build different models while Pandas as an effective python data analysis library for processing data into correct form and data is visualized using. Matplotlib library.

2.2 Dataset and sources

The dataset collected from Kaggle [1] contains 2240 rows/ observations with 28 columns/variables related to marketing data, which is sufficient for the analysis. Each datapoint corresponds to a customer identified by a unique ID. More specifically, the properties of this dataset present customers personal information, products they purchased and campaign results.
The features recorded for each datapoints are explained in the Table 1. There are 23 numeric and 5 categorical candidate variables. The label for each observation is binary value “Response”, which is 1 if customer accepts the offer in the last campaign, 0 otherwise.

3 Methods

3.1 Preprocessing and feature selection

The purpose of EDA (exploratory data analysis) is to better understand the datapoint and what need to be preprocessed. Some significant issues are pointed out as following:

- The figure [1] showed that the property ‘Income’ has 24 missing values, which are replaced by the median() value. The column name ‘Income’ has extra space that should be striped for better searching in the future.

Moreover, the datapoints in 'Income' which are string data type contains '$' symbol that need to be removed and convert into numerical data type.

- In the 'Marital Status' and 'Education' in figure [2], there are some categories with different names but the same meaning in real life. For example, 'YOLO', 'Alone', or even 'Absurd' is technically fall into the category 'Single'; the '2n Cycle' are the same as 'PhD'.
- In the "Year_Birth" and "Dt – Customer", according to the meaning explained, it is more practical to tranform them into Age and Years that the customers have enrolled with the company.
- When checking some significant numerical variables, such as 'Age' and 'Income' , there are outliers found. Those outliers need to be replaced by median value or removed.
- There are five columns with object datatype. The pd.get_dummies function is used to transform those categorial values into binary values.

Regarding feature selection, the data visualization contributes significantly to the feature selection process. The specific process is explained as below:

- The correlation between the features is visualized by Heatmap matrix using the Seaborn library [2] . It makes it easy to identify which features are highly related to the target variable. As a result, the variables 'Age', 'Kidhome', 'Teenhome', 'Recency', 'NumWebVisitsMonth', and 'Complain' shows the negative correlation with the target variable, and thus are removed from the list of features.
- For the categorical variables that are unable to be visualized in correlation matrix, the simple bar charts are used to reveal how those variables relate to the target variable. For example, the Figure [3] shows how the response to the last campaign depends on the education level of the customer. In the other word, education level could be a good predictor. Similar visualizations are implemented to 'Country', 'Marital Status' variables in figure [4] and figure [5].

3.2 Construction of Training, Testing, and Validation Sets

The common 70/30 split ratio is chosen as 70% of the entire data set will be allocated to the training set and the remaining 30% for the test set using the Train test split function from the Scikit library [3]. Since the data is highly imbalanced and thus the amount of data in the minority class is limited for testing. Setting a low proportion for testing could cause the test accuracy being higher than the train accuracy (it happened when the 80/20 ratio was tried).

The training set, as its name is trained using GridsearchCV from Scikit-learn library (Pedregosa et al., 2011). It tunes the hyperparameters by using a different combination of all the specified hyperparameters to select the best values from them. While applying GridsearchCv, cross-validation StratifiedFold is also performed. It ensures that each fold of dataset has the same proportion of samples with a given label. It is the iterative process that divides training data into k-parts and the algorithm runs k– times. In each interaction, one partition is saved for validation and the remaining of training data. It also records the performance of model and give the average of all performance at the end. Since the process consumes time and more cost based on the number of hyperparameters involved, it is sufficient to split data into 5 folds instead of 10.

3.3 Methods

3.3.1 Oversampling without SMOTE

The two classes are highly imbalanced with 85% in the class 'No' and 15% in the class 'Yes'. Therefore, it is necessary to rebalance the data . The algorithm works in a way that it creates synthetic samples from the minority class ('Yes') instead of making copies and randomly choose one of the nearest neighbors to create a similar new observation.

The SMOTE was applied only to the training set to ensample the minority class ('Yes' class in this case) with the function SMOTE in Imbalanced learn library [4].

3.3.2 Logistic regression without SMOTE

The first model chosen for this project is Logistic Regression [5] for the following reasons:

1. The label is binary value, making it classification problem and logistic regression is the go-to model that allows to probabilistically model binary variables.
2. The objective of this task is not only to predict the value of variable but also to explain the impact of each predictator. Logistic regression as an interpretable model helps in a way that a predictor's coefficient quantifies the impact of each feature via odd ratio. The odd ratio is defined as the probability that the output for a given input is positive compared to the probability against the event. It is formulated in a form:

$$\boldsymbol{Odd\ ratio} = \frac{\boldsymbol{P(y=1|x;w)}}{\boldsymbol{P(y=0|x;w)}} = \frac{\boldsymbol{P(y=1|x;w)}}{\boldsymbol{1-P(y=1|x;w)}} = \boldsymbol{e^{-w^t x}}$$

Logistic Regression is a supervised machine learning algorithm that uses the hypothesis space of linear hypothesis maps $\boldsymbol{h(x) = w^T x}$ to classify a datapoint into classes where w is the weight vector. It models the probability distribution $\boldsymbol{p(y|x)}$ where x is its input representation and y is the class label (either 0 or 1). Once the distribution is learned, the model will classify a new datapoint based on the decision boundaries (commonly 0.5) or threshold as belonging to class 1 if $\mathbf{p(y=1|x)}$ is greater than a chosen threshold, commonly 0.5, and class 0 if $\mathbf{p(y=1|x)}$ is less than a chosen threshold**.** The logistic regression finds a hypothesis in the form: **y = 1 if y = 1 if** $\mathbf{p(y=1|x) > 0.5}$**, 0 otherwise**. The weighted input features (plus a bias term) $\hat{y} = w_0 x_0 + w_1 x_1 + \ldots + w_n x_n = W^T X$ is plugged into the sigmoid function**:** $\boldsymbol{f(x)}$**:** $\boldsymbol{p(x) = 1 / (1 + \exp(-f(x))}$ to calculate the probability $p(y = 1|x)$. The logistic loss was used as the loss function because is quick to optimize and implemented as the log-loss function in the Scikit-learn library (Pedregosa et al., 2011). The logistic loss is formulated as:

$$\boldsymbol{L_{log}(y,p) = -\big(ylog(p) + (1-y)log(1-p)\big)}$$

The tuned hyperparameters for this model are: 'C': 1.1, 'random_state': 42, "solver": 'liblinear'

3.3.3 Decision tree with SMOTE

The second model selected to solve this machine learning problem is Decision Tree Classifer using gini impurity [6]. The hypothesis space of all decision trees is a complete space of finite discrete-value functions . Decision Tree is not only a powerful but efficient algorithm for classification problem. Comparted to the Random Forest, Decision Tree is much simpler, less time and cost consuming even on a large and complex data while a Random Forest combines more several decision trees as its name also reveal. Moreover, GridsearchCV application makes the process even much longer, probably last more than one hour to run.

At its core, it includes a root node, some branches (to connect these different nodes) and leaf nodes. The root node is always at the top of the tree while the tree leaves represent the outcome of the problem (in this case 0 or 1). The algorithm will split the tree until the data is sufficient homogeneous. The algorithm as a supervised approach uses a tree - based data structure to predict the outcome. The process is iterative starting from the root node. In each iteration, it iterates through the very unused attribute (features) and calculate the selected criteria of attribute selection measures in each node, commonly Entropy, Gini Impurity, or Information Gain to select attribute for the node. When the training set gone through Gridsearch cross validation, the criteria along with other parameters that altogether came out with the best accuracy score is Gini Impurity. Decision Tree use loss function that evaluates the split based on the purity of the resulting nodes, in this case Gini Impurity measured as:

$$G(node) = \sum_{k=1}^{c} p_k(1 - p_k)$$

while p_k is probability that the output is in class 1. The lower Gini Impurity is preferred to select the attribute for the node. The tuned hyperparameters for decision tree model are: 'criterion': 'gini', max_depth': 6, 'random_state': 42.

4 Results

Since it is a classification problem, the accuracy score is preferred to evaluate the model performance, instead of training and validation errors . Moreover, since the data has the large amount of "negative" sample, it is better to also use F_score to compare the two model performance along with the accuracy score. The f_score is defined as a harmonic means of the model's precision and recall. The formula for the f score is:

$$F_1 = 2\,x\,\frac{precision\ x\ recall}{precision + recall}$$

The two score are computed using the Classification_report() function in Scikit-learn library [7]. The performance of each model is visualized and summarized via confusion matrix. The obtained accuracy scores of both models are in Table 2. The logistic regression has a higher test accuracy and higher f- score. That also means that when applying SMOTE on the data set, it fits better on the logistic regression model. The test error for Logistic Regression is computed by training the training data on the model and then calculating the test error on the test set, using log loss function Scikit-learn library (Pedregosa et al., 2011). The test error is ≈ 5.551. (Lower than the one in Decision Tree).

The confusion matric visualized in figure [7] reveals how the unbalanced Logistic Regression is essentially always predicting the majority class '0'. Due to the highly imbalanced class distribution, it results in a high testing accuracy of 85.91% but such a classifier has clearly not learned anything and useless in practice. In contrast, the balanced Logistic Regression classifier performs much better: it classifies 48% of the "yes" cases correctly while the figure is only 12% in imbalanced data model.

As a result, the Logistic Regression with SMOTE fits much better and thus is chosen to be the final model.

5 Conclusion

This project used two classifier Logistic Regression and Decision Tree to classify customers responses to the last marketing campaign. The Logistic Regression model fits better and thus was chosen as the final model. It achieves 83.93% accuracy in the final test. Additionally, using SMOTE to balance on the training data allows the model to predict the class 1 more accurately. It is also worth mentioning that the test accuracy is very close to the training accuracy as interpreted the model is not overfitting.

Though the results are promising, the chosen model shows some rooms that could be improved or need further inspection. Firstly, the dataset is highly imbalanced with only 15% in negative samples, and thus and thus more samples would be necessary to further test the model, especially the samples in the negative class. Secondly, when applying SMOTE model to rebalance data, it has tendency to create noise and confusing samples that might reduce the performance of the model. There are more tactics to handle imbalanced data such as Near-Miss Algorithm application to eliminate samples from larger class, better feature selection, clustering the abundant class. Finally, trying different models or changing the loss function should also be taken into consideration for improvement. However, it is highly recommended that collecting more data is the priority for further work.

6 Reference

[1] Marketing Campaign. Kaggle. Uploaded by Rodolfo Saldanha. Accessed 12.09.2022. Available at: https://www.kaggle.com/datasets/rodsaldanha/arketing-campaign

[2] seaborn learn Machine Learning in Python library documentation: Heatmap. Accessed 21.9.2022. Available at: seaborn.heatmap — seaborn 0.12.0 documentation (pydata.org)

[3] scikit learn Machine Learning in Python library documentation: Train test split. Accessed 17.9.2022. Available at: sklearn.model_selection.train_test_split — scikit-learn 1.1.2 documentation

[4] imbalanced learn Machine Learning in Python library documentation: SMOTE. Accessed 10.9.2022. Available at: SMOTE — Version 0.9.1 (imbalanced-learn.org)

[5] scikit learn Machine Learning in Python library documentation: Logistic Regression Classifier. Accessed 10.9.2022. Available at: sklearn.linear_model.LogisticRegression — scikit-learn 1.1.2 documentation

[6] scikit learn Machine Learning in Python library documentation: Decision Tree Classifier. Accessed 15.9.2022. sklearn.tree.DecisionTreeClassifier — scikit-learn 1.1.2 documentation

[7] scikit learn Machine Learning in Python library documentation: Classification Report. Accessed 18.9.2022. sklearn.metrics.classification_report — scikit-learn 1.1.2 documentation

7 Appendix

7.1 Table and Figure

Features	Explaindation	Data type
ID	Customer's unique identifier	int64
Year_Birth	Customer's birth year	int64
Education	Customer's education level	Object
Marital_Status	Customer's marital status	Object
Income	Customer's yearly household income	Object
Kidhome	Number of children in customer's household	int64
Teenhome	Number of teenagers in customer's household	int64
Dt_Customer	Date of customer's enrollment with the company	Object
Recency	Number of days since customer's last purchase	int64
MntWines	Amount spent on wine in the last 2 years	int64
MntFruits	Amount spent on fruits in the last 2 years	int64
MntMeatProducts	Amount spent on meat in the last 2 years	int64
MntFishProducts	Amount spent on fish in the last 2 years	int64
MntSweetProducts	Amount spent on sweets in the last 2 years	int64
MntGoldProds	Amount spent on gold in the last 2 years	int64
NumDealsPurchases	Number of purchases made with a discount	int64
NumWebPurchases	Number of purchases made through the company's web site	int64
NumCatalogPurchases	Number of purchases made using a catalogue	int64
NumStorePurchases	Number of purchases made directly in stores	int64
NumWebVisitsMonth	Number of visits to company's web site in the last month	int64
AcceptedCmp3	1 if customer accepted the offer in the 3rd campaign, 0 otherwise	int64
AcceptedCmp4	1 if customer accepted the offer in the 4th campaign, 0 otherwise	int64
AcceptedCmp5	1 if customer accepted the offer in the 5th campaign, 0 otherwise	int64
AcceptedCmp1	1 if customer accepted the offer in the 1st campaign, 0 otherwise	int64
AcceptedCmp2	1 if customer accepted the offer in the 2nd campaign, 0 otherwise	int64
Response	1 if customer accepted the offer in the last campaign, 0 otherwise	int64
Complain	1 if customer complained in the last 2 years, 0 otherwise	int64
Country	Customer's location	Object

Table 1: Dataset candidate variable and explanations

	Logistic Regression	Decision Tree
Training Accuracy	84.54%	85.05%
Test Accuracy	83.93%	80.51%
F_score	0.48	0.34
Test error	6.528	6.772

Table 2 : Accuracy Score, F score, and test Error

```
pd.DataFrame(data.isnull().sum(), columns=['#Null values']).T
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumStorePurchases	NumWebVisitsMonth
#Null values	0	0	0	0	24	0	0	0	0	0	...	0	0

Figure 1: Check missing values

Marital Status and Education

```
print(data['Marital_Status'].unique())
print(data['Education'].unique())
```

```
['Divorced' 'Single' 'Married' 'Together' 'Widow' 'YOLO' 'Alone' 'Absurd']
['Graduation' 'PhD' '2n Cycle' 'Master' 'Basic']
```

Figure 2: Checking the categorical variables – Marital Status and Education

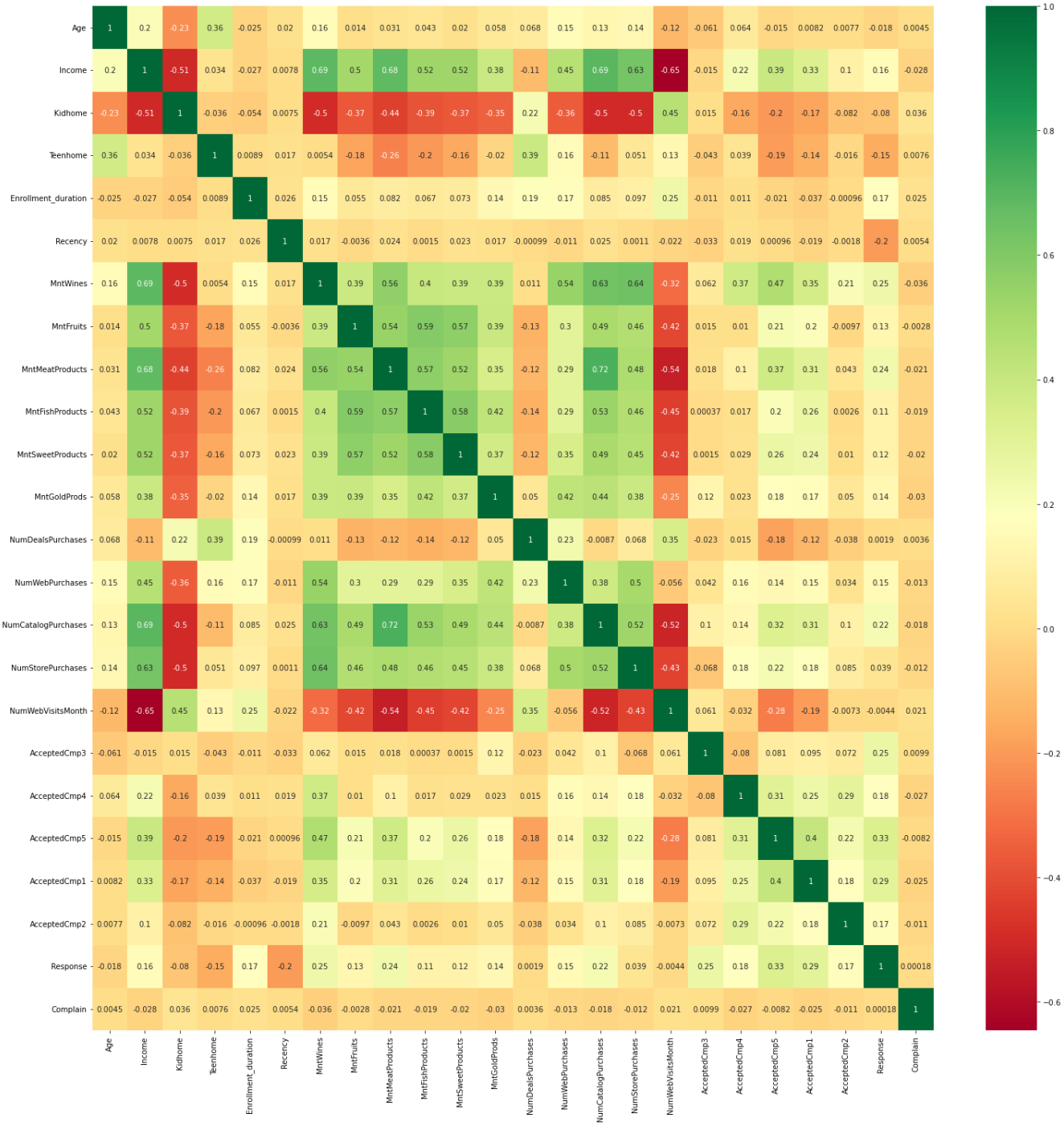

Figure 3: Correlation matrix between features

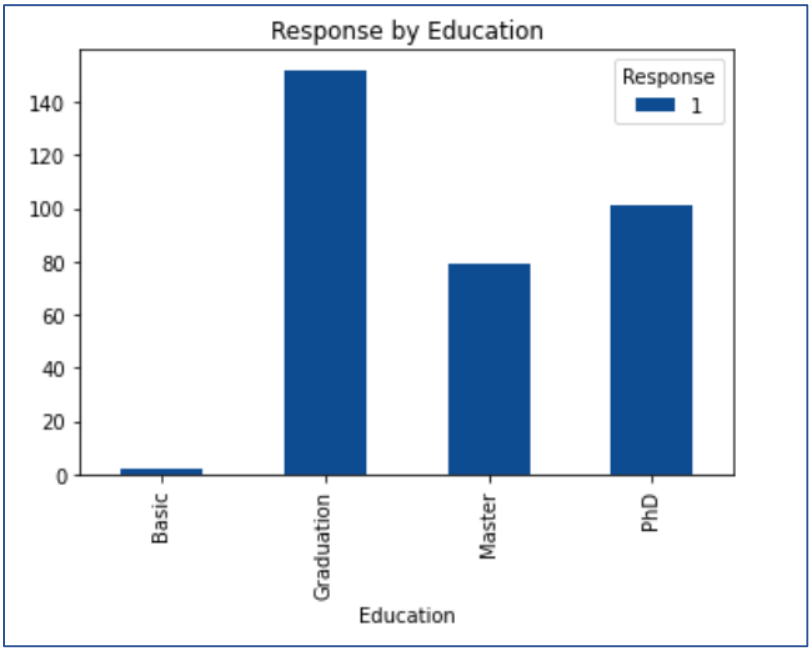

Figure 4: The response to the last campaign by Education

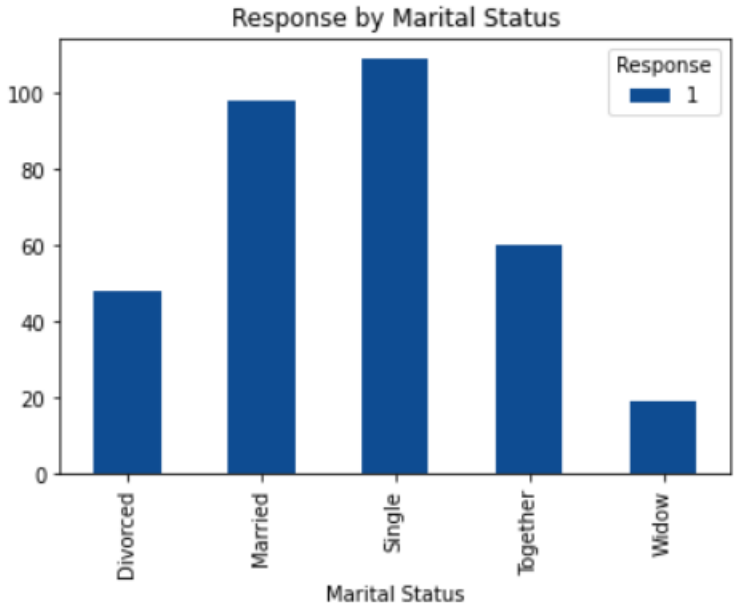

Figure 5: The response to the last campaign by Marital Status

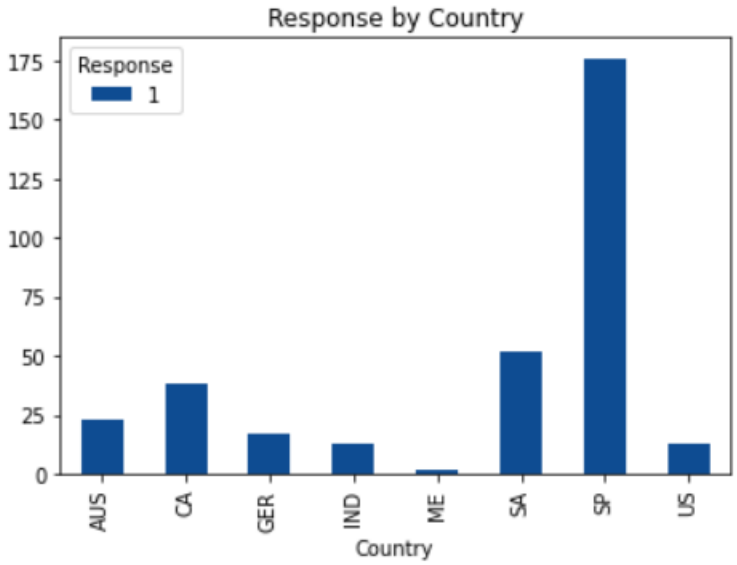

Figure 6: The response to the last campaign by country

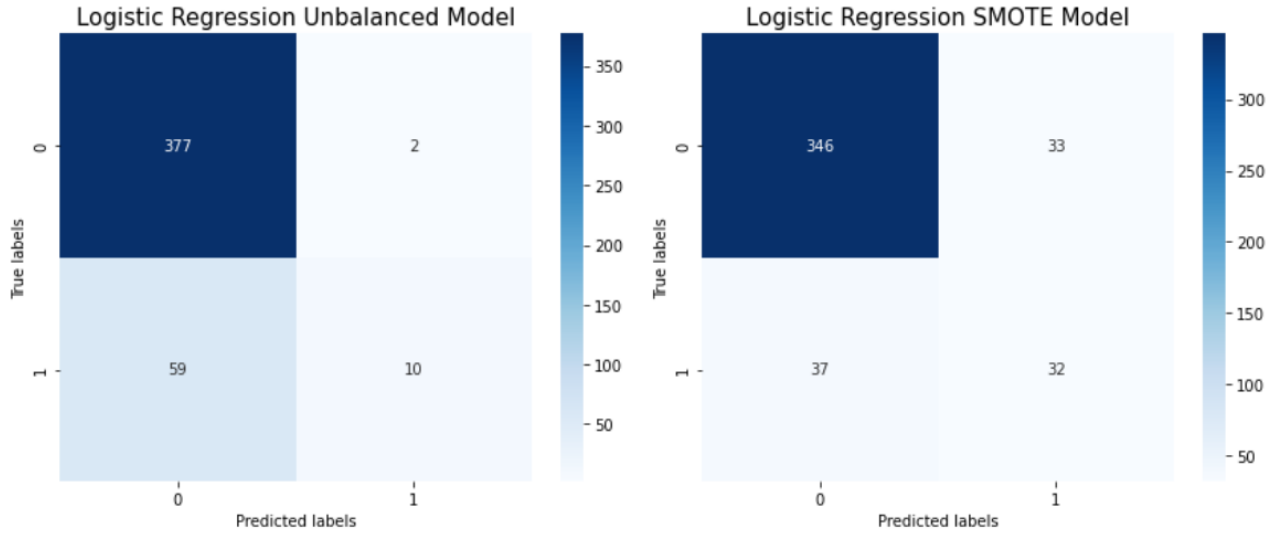

Figure 7: The confusion matrix

7.2 Source Code