Prediction of Marketing Campaign

Name:	**Devashish Tripathi**
Roll No.:	21093
Institute:	IISER Bhopal
Program/Stream:	Data Science and Engineering
Problem Release date:	August 17, 2023
Date of Submission:	November 19, 2023

1 Introduction

The problem statement "Prediction of Marketing Campaign" concerns the prediction of campaigning results on customers evaluated by a company evaluated on various attributes. It is a binary classification problem, with the results being 0 and 1, indicating the failure and success of the campaign, respectively.
The provided training and test datasets consist of 2016 and 224 instances, respectively. The dataset has date-time variables like 'Dt_Customer', Categoricals like 'Marital_Status', 'Complain' and 'Education' and Continuous variables like 'Income'. Only 301 training instances are classified as success, meaning a class imbalance is present. Additionally, in both the training and test datasets, there are 21 and 3 missing Income values, respectively.
The basic plan involves imputing the missing Income values using a suitable imputation technique, converting the DateTime variables, One Hot Encoding the String Categoricals and doing rigorous analysis of the selected classifiers on the training data while using various hyperparameter configurations along with scaling and feature selection to get the best working model for the problem.
The results were achieved and evaluated. Adaptive Boosting with No Scaling, No Feature Selection and Logistic Regression as the base estimator gave the best results.

2 Methods

The following classifiers were used and evaluated rigorously for their performance on the given problem: Decision tree, Random Forest, Support Vector Machine, Naive Bayes, Logistic Regression, K Nearest Neighbors, Adaptive Boosting and Gradient Boosting. For all of the classifiers, various combinations of Hyperparameters, Scalers and Feature Selectors were used and compared for the best results.
For Scaling, the following techniques were used: Standard Scaling, Max Absolute Scaling and MinMax Scaling. For Feature Selection, the following techniques were used: Select K Best and Sequential Feature Selection. Principal Component Analysis(PCA) was also implemented. Hyperparameter tuning was implemented for Scaling and Feature Selection techniques, too.
All the techniques were implemented from their respective libraries in Scikit-Learn[1]. Additionally, for the initial run, no scaling was done.
The GitHub link for the project is: https://github.com/Devashish-Tripathi/Project_DSE317

3 Experimental Setup

The different models were evaluated primarily on the F-Measure of class 1, i.e. the minority class. For imputation, it was found that imputing the missing values using mean gave the best results. For all evaluations, the models were trained using 5-fold Stratified Splits to have an almost equal percentage of samples from both classes. This gave, for each training set, 1372 features in Class 0 and 240 features

in Class 1 and for each validation set, 343 features in Class 0 and 61 features in Class 1. The hyperparameter tuning was done, and results on the best parameters were evaluated using GridSearch.
For the first step, all models except Adaptive Boosting and Gradient Boosting were directly evaluated on all the features. This was followed by assigning balanced weights to the classes and training the classifiers, which can account for balanced class weights(i.e. not Naive Bayes or K Nearest Neighbors Classifiers).
In the second step, Adaptive Boosting and Gradient Boosting classifiers were trained. The base estimators for Adaptive Boosting were selected from all the estimators used in the previous step, and the parameters used for these estimators were the ones given by GridSearch to be the best ones. The best results for each classifier to evaluate validation data in the first two steps can be found in Table 1.
From the results obtained from the above steps, the model's Decision Tree, Logistic Regression, Adaptive Boosting, and Gradient Boosting with hyperparameters predetermined from the above steps were selected for evaluation under feature selection and Scaling. For the third step, GridSearch was used again to apply the scalers and Feature Selection. The best results for each classifier to evaluate validation data for this step can be found in Table 2. Based on these results, the best estimator was found, and the test data labels were predicted and stored.
A variety of hyperparameters had to be tuned during the process. The important ones included: 'solver' for Logistic Regression, no of estimators and splitting criterion for Random Forests, splitting criterion and cost complexity pruning constant, 'ccp_alpha' for Decision Trees, 'kernel' for SVM, no of neighbors for kNN, loss and splitting criterion for Gradient Boosting and the base estimator for Adaptive Boosting. For feature selection, the major parameter to be tuned was related to the number of features to keep in all cases.

4 Results and Discussion

Table 1: Best results(For Class 1) on validation set for Different Classifiers after Step 2.

Classifier	Precision	Recall	F-measure
Adaptive Boosting	0.84	0.52	0.64
Decision Tree	0.42	0.70	0.52
Gaussian Naive Bayes	0.36	0.59	0.44
Gradient Boosting	0.79	0.52	0.63
K-Nearest Neighbor	0.65	0.28	0.40
Multinomial Naive Bayes	0.24	0.64	0.35
Logistic Regression	0.45	0.80	0.57
Random Forest	0.82	0.45	0.58
Support Vector Machine	0.34	0.51	0.41

Analysis for Table 1: k-Nearest Neighbors and Random Forest classifiers were showing overfitting. The Support Vector Machine and Naive Bayes performed poorly on most runs. As such, based on overall results, Adaptive Boosting(best parameters: 'base_estimator': LogisticRegression(random_state=1, solver='newton-cg'), 'n_estimators': 100), Gradient Boosting(best parameters: 'ccp_alpha': 0, 'criterion': 'friedman_mse', 'loss': 'log_loss', 'n_estimators': 500), Decision Tree(best parameters:'ccp_alpha': 0.001, 'criterion': 'gini', 'max_features': 'sqrt', 'splitter': 'random'), and Logistic Regression(best parameters: 'solver': 'newton-cg') were selected for further steps.

Analysis for Table 2: Gradient Boosting was showing overfitting. Adaptive Boosting with No Scaling and Feature Selection gave the best results and, thus, was selected as the final model.

The best estimator turned out to be Adaptive Boosting using 100 estimators, with the base estimator of a Logistic Regression having the solver 'newton-cg' and no Scaling or Feature Selection needed. The confusion matrices for the validation set for all 5 runs for this model are in Table 3.

The most surprising finding was the fact that the Adaptive Boosting classifier did not require any

Table 2: Best results(For Class 1) on the validation set for the selected Classifiers after Step 3.

Classifier	Precision	Recall	F-measure	Scaler	Feature Selection
Adaptive Boosting	0.50	0.83	0.62	None	None
Decision Tree	0.44	0.83	0.58	None	SelectKBest(10 features)
Gradient Boosting	0.68	0.47	0.55	MaxAbs	SequentialFeatureSelection(forward, auto)
Logistic Regression	0.44	0.72	0.55	MinMax	SelectKBest(10 features)

Table 3: Confusion Matrices for the validation set for all 5 runs for the final model

Actual Class	Predicted Class	
	Class 0	Class 1
Class 0	279	64
Class 1	10	51

Run 1

Actual Class	Predicted Class	
	Class 0	Class 1
Class 0	295	48
Class 1	16	44

Run 2

Actual Class	Predicted Class	
	Class 0	Class 1
Class 0	285	58
Class 1	15	45

Run 3

Actual Class	Predicted Class	
	Class 0	Class 1
Class 0	283	60
Class 1	15	45

Run 4

Actual Class	Predicted Class	
	Class 0	Class 1
Class 0	293	50
Class 1	10	50

Run 5

Scaling or Feature Selection. It also gave the most consistent results on the training and validation samples across all the splits and all the runs.

5 Conclusion

From the analysis, it can be concluded that Adaptive Boosting with the base estimator as Logistic Regression can provide satisfactory results for binary classification problems like the one tackled in this project, even in case of class imbalance. Future works in this project can be focused on increasing the F-Score for the minority class more, possibly by increasing their effect on the training set.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.