Predicting Outcome of Marketing Campaign

Name:	**Satyajeet Shashwat**
Registration No./Roll No.:	21243
Institute/University Name:	IISER Bhopal
Program/Stream:	DSE
Problem Release date:	August 17, 2023
Date of Submission:	November 19, 2023

1 Introduction

The aim is to build a machine-learning model that predicts the success or failure of the marketing campaign based on the customer's data. The data set contains 2016 training instances, and 25 different features. The feature vectors are classified into two different classes either 1(success) or 0(failure).301 feature vectors are labeled as 1(success) and 1715 data points are labeled as 0(failure) out of the given 2016 data points. The overview of the Outcome is shown in Figure 1. The set have only two features that have categorical value for which i have applied one-hot encoding, other than that all the features contains numerical value. The data set also has a feature named Income that has 21 missing values. There is 224 test instances for which i have to predict the labels. I have used different classifiers to get the best one and then predicted the labels.

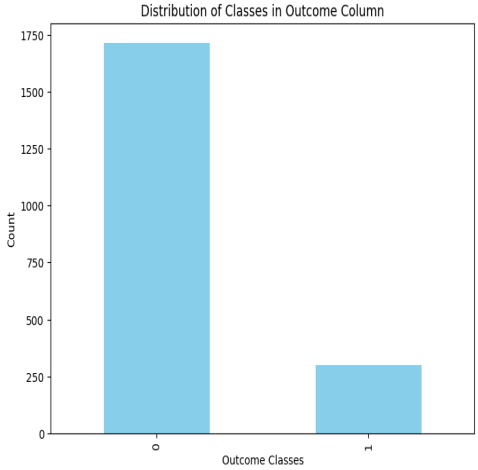

Figure 1: Overview of the Outcome

2 Pre-processing

One of the feature named "Income" had 21 missing values, I have imputed those missing values by mean based on their education. My data-set had two features namely "Education" and "Marital Status" with categorical values, I have applied one-hot encoding on them. I have dropped two features named "Year Birth" and "Dt Customer" which was not much relevant for predicting the labels. I have also dropped two feature because it was positive for only one data point and negative for all other. My data-set was imbalanced as it had 1715 feature vectors for label 0 and 301 feature vectors for label 1. Therefore I have applied oversampling technique to generate some random feature vectors for label

Table 1: Best Parameters for all the classifiers

Naive Bayes	Alpha: 0, Fit Prior: True
Decision Tree	ccp-alpha: 0.01, criterion: entropy, max-depth: 10
K-Nearest Neighbor	n-neighbours: 11, P: 1, Weights: Distance
Logistic Regression	C: 0.001, Penalty: l2, solver: newton-cg
Random Forest	Criterion: Gini, Max-depth: 10, n-estimators: 100
Support Vector Machine	C: 100, Kernel: rbf
Adaptive Boosting	base-estimator: LR(class weight= balanced, solver= liblinear)

Table 2: Performance Of Different Classifiers without oversampling

Classifier	Precision	Recall	F-measure
Naive Bayes	0.5706	0.6242	0.5618
Decision Tree	0.7201	0.6186	0.6448
K-Nearest Neighbor	0.7277	0.5467	0.5505
Logistic Regression	0.6772	0.7799	0.6994
Random Forest	0.7924	0.6546	0.6917
Support Vector Machine	0.7834	0.5077	0.4762
Adaptive Boosting	0.6747	0.7381	0.6953

1. I have trained all the classifiers for Over sampled data and without over sampled data. After all this preprocessing and data cleaning my total number of features became 32.

3 Hyperparameter Tuning and Feature Selection

For hyperparameter tuning, I have used GridSearchCV[0] and did not used RandomizedSearchCV because Grid search is a deterministic method, and explores all possible combinations of parameter values, while random search is a probabilistic method that only explore a random subset of possible combinations. I have used f1-micro as the selecting criteria for the grid search and have done cross validation 10 times. I have created a pipeline for feature selection and grid search and for feature selection I have used SelectKBest. I have trained the model using all features(without feature selection) and with best 10 features. The selection criteria used for selecting K best feature is chi square.

4 Methods

I have split the training data into training and validation set using 8:2 as the split ratio in a stratified fashion and train the model using the best features obtained from SelectKBest and using the best parameters I got from hyperparameter tuning. I have used 7 differnt classifiers namely Adaptive Boosting[3], Decision Tree[1], K-Nearest Neighbour[0], Naive Bayes[2], Logistic Regression[5], Random Forest[4] and Support Vector Machine[0]. The best parameters I got for all the classifiers is given in table 1.

5 Experimental Setup

Macro-averaged precision, recall, and F-measure was used to determine the performance of the different classifiers using different parameters to evaluate the best performing framework to predict the class labels. The results of classifiers without Oversampling (on original data) and with all features is given in table 2. The results of classifiers with oversampling and with 10 best feature is given in table 3 and the results of classifiers with oversampling and with all features is given in table 4.

Table 3: Performance Of Different Classifiers Using Oversampling and 10 best Features

Classifier	Precision	Recall	F-measure
Naive Bayes	0.7629	0.7592	0.7584
Decision Tree	0.8038	0.8037	0.8037
K-Nearest Neighbor	0.8597	0.8594	0.8594
Logistic Regression	0.8195	0.8195	0.8194
Random Forest	0.7922	0.7914	0.7912
Support Vector Machine	0.8289	0.8285	0.8284
Adaptive Boosting	0.7945	0.7943	0.7942

Table 4: Performance Of Different Classifiers Using Oversampling and All Features

Classifier	Precision	Recall	F-measure
Naive Bayes	0.7626	0.7588	0.7579
Decision Tree	0.7997	0.7996	0.7996
K-Nearest Neighbor	0.8600	0.8598	0.8598
Logistic Regression	0.8644	0.8619	0.8616
Random Forest	0.8187	0.8186	0.8186
Support Vector Machine	0.8298	0.8293	0.8292
Adaptive Boosting	0.8370	0.8359	0.8358

6 Results and Discussion

All the codes and Output is also uploaded on GITHUB Click here. The results by using Oversampling with 10 features and by using Oversampling with all features are almost same for all the classifiers except for Logistic Regression and Adaptive Boosting which got a boost of 3 to 4 percent but it also increases the time complexity of the code. The best result(F1 score= 0.8616) that I got is by using Logistic regression classifiers with parameters c=0.001, penalty= None and solver= netown-cg. This result I got by Oversampling of the traning data and by using all the features.

References

[1] 1.10. Decision Trees — scikit-learn.org. `https://scikit-learn.org/stable/modules/tree.html`. [Accessed 19-11-2023].

[2] 1.9. Naive Bayes — scikit-learn.org. `https://scikit-learn.org/stable/modules/naive_bayes.html`. [Accessed 19-11-2023].

[3] sklearn.ensemble.AdaBoostClassifier — scikit-learn.org. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html`. [Accessed 19-11-2023].

[4] sklearn.ensemble.RandomForestClassifier — scikit-learn.org. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`. [Accessed 19-11-2023].

[5] sklearn.linear$_model.LogisticRegression - - - scikit - learn.org..$ [Accessed 19-11-2023].

sklearn.model$_selection.GridSearchCV - - - scikit - learn.org..$ [Accessed 19-11-2023].

sklearn.neighbors.KNeighborsClassifier — scikit-learn.org. `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`. [Accessed 19-11-2023].

sklearn.svm.SVC — scikit-learn.org. `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`. [Accessed 19-11-2023].