Predictive Model on Campaign Impacts

Final Report for DSC148-WI24

Donald Taggart
dtaggart@ucsd.edu

Andrew Peng
apeng@ucsd.edu

Introduction

Direct marketing campaigns allow you to target the ideal customer by understanding their needs and wants in order to gain customers in the most cost-effective way. However, while this method does result in a much higher customer-response rate, direct marketing and sale campaigns come with very high costs regarding management and administrative overhead. These costs can be reduced if the marketing campaign targets individuals and groups that are much more likely to purchase said product. In this project, we will build a model that will predict an individual's response to a direct marketing campaign.

Github link:

https://github.com/DonaldTaggart/dsc148-campaign-impact/tree/main

1. Dataset

1.1 Identify Dataset

The dataset used will be data collected from a Portuguese banking institution based through phone calls (bank-full.csv). This **bank** [1] dataset was found on UCI's machine learning repository and was obtained as a .csv file. The dataset contains 45,211 instances with 16 features as well as a corresponding variable, referred to as **y**, for each instance to denote if the marketing campaign succeeded in leading the individual to subscribe to a new term deposit. This data ranges from May 2008 to November 2010 and has no missing values.

Columns within the dataset focus on the basic features of each individual contacted (age, job category, marital status, education). It also included a few more in-depth measurements of each individual's financial status (loan, housing, balance) as well as data summarizing if the individual had been contacted before for a similar campaign and the outcome of said campaign (last contacted, number of contacts, previous outcome).

1.2 Basic Exploratory Data Statistics

The most basic thing to acknowledge is that, after searching through this dataset, there are no NaN or missing values in the dataset. However, for columns where the data is categorical, data that does not belong to any of the categories is saved as "unknown" instead.

To develop a better understanding of the individuals who were contacted during the marketing campaign, we explored most of the columns within the dataset.

age records the numeric ages of each individual contacted during the direct marketing campaign. While a majority of those contacted are condensed around the ages of 30 to 50, a very small number of responses come from those over 60+ (Figure 1).

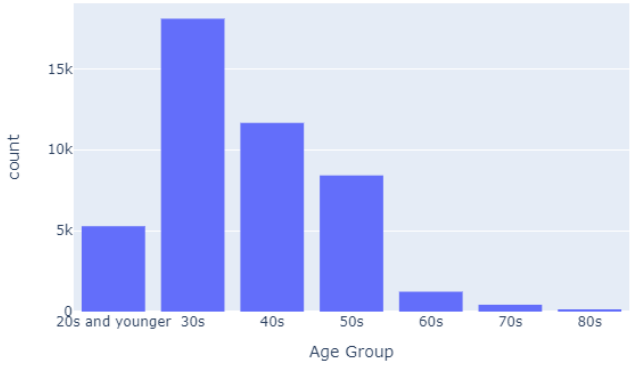


Figure 1: Frequency Distribution of Age Groups

balance describes the average yearly balance in euros (a numeric value). However, here we see that some people have negative yearly balances (Figure 2). This is very likely due to overdrafts where individuals spend more than what they have in their account. This can also be attributed to having overdraft fees and other late payments[2]. This will prove very useful and

[1] https://archive.ics.uci.edu/dataset/222/bank+marketing

[2] https://www.self.inc/blog/bank-balance-negative

interesting to see if those with lower average account balances are less likely to subscribe a term deposit.

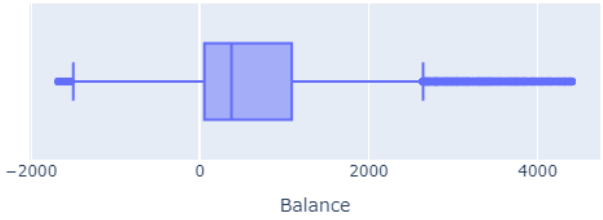


Figure 2: Box plot of balance accounting for a single standard deviation

job has 12 predetermined categories for jobs. These include the following (Figure 3).

```
blue-collar      9732
management       9458
technician       7597
admin.           5171
services         4154
retired          2264
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: job, dtype: int64
```

Figure 3: Frequency of Job Categories

From this we can gleam a few basic statistics. Firstly, the most common job categories are “blue-collar”, “management”, and “technician”, suggesting that the dataset does indeed cover a diverse range of professional backgrounds, preventing too much skewness from having a predominant job category available. Additionally, the wide spread of jobs is positive as different professions have varying financial situations, influencing their decision to get a subscription to a term deposit or not. Thus, by including **job** in the classification prediction model, we can develop a more effective marketing strategy to tailor the marketing campaign to consider the finances and interests of those employed in different industries.

1.3 Interesting Exploratory Data Analysis

When doing a simple linear regression of age on average balance (standardized with z-scores) in euros, a very interesting relationship comes to light (Figure 4). While this could be conditional to the lack of individuals contacted for ages 60 and up, the spread of data increasingly narrows. Interestingly enough, after the age of 50, the number of individuals with negative bank account balances gradually tapers off. However, a little after the age of 60, that number sharply drops to nearly zero. A likely source of this can be seen in the OCED database regarding retirement ages[3]. In the years 2008 to 2010, the average age of retirement (and thusly the age most people accessed their pension and retirement funds) was 62 years old. This graph displays that feature as many do not overspend their retirement pensions once they receive them.

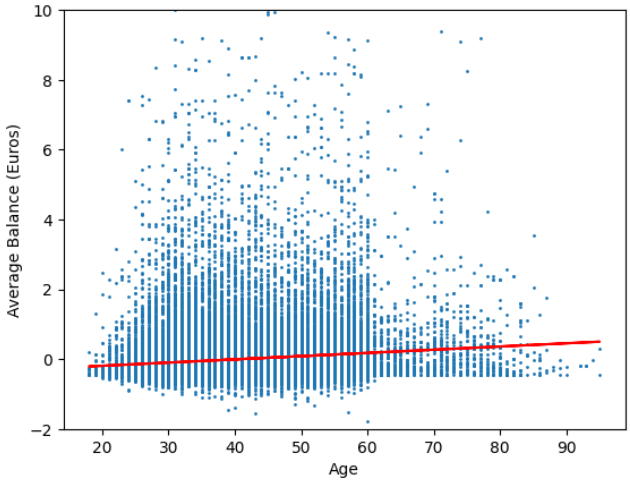
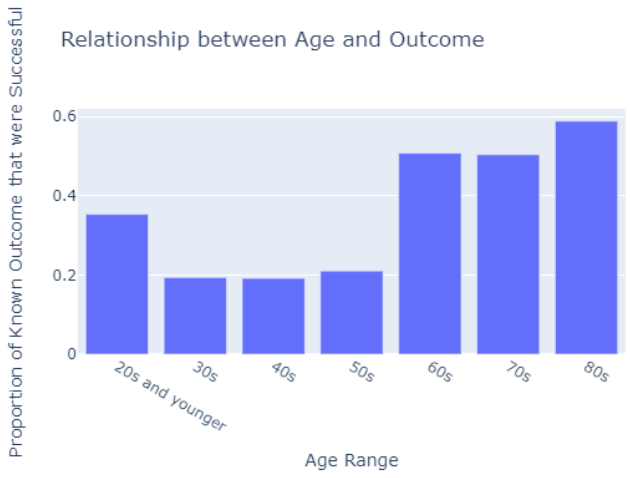


Figure 4: Relationship between Age and Balance (Standardized)

Looking further into if those who have already retired may be more likely to accept a subscription to a term deposit, we observed the relationship between age and the proportion of recorded interactions that were successful (from previous marketing campaigns). Here, we can see a clear jump in the outcome proportion for those ages 60 and up (Figure 5).



[3] https://stats.oecd.org/Index.aspx?DataSetCode=PAG#

Figure 5: Relationship between Age and known outcome of those previously contacted

The large jump in success rates of prior contact records is evident. Therefore, it's fair to say that there will be a substantial correlation between someone's age and the chances of a them purchasing a subscription upon contact.

Unsurprisingly, this trend is noted by the banks themselves. Working with the data they collected, we can visualize the average time since last contacted for each age (Figure 6).

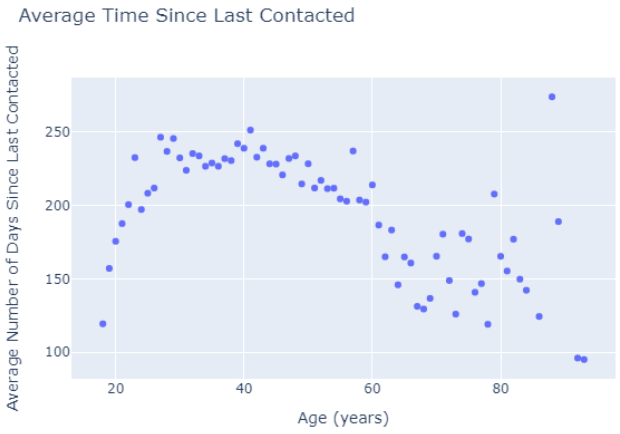


Figure 6: Relationship between Age and average last known contact

The disparity and frequency in ages contacted along with each age group's propensity to accept a subscription will be a key component of the machine learning process. To visualize this with another method, we plotted the relationship between those who were acquired a subscription upon being contacted and the category of their job (Figure 7).

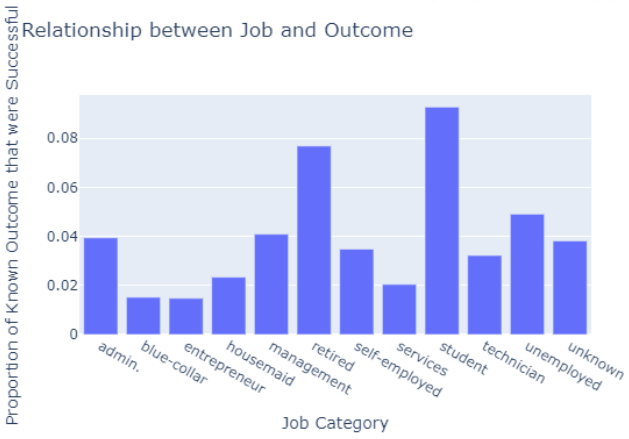


Figure 7: Relationship between outcome and job category

Here, we see further evidence of the above along with a surprising fact. The two categories of people that had the highest proportion of those with the propensity to accept a subscription were students and those that were retired. Unsurprisingly, retirees once again had some of the highest proportions regarding outcome of prior contact: this makes it yet again a large point of contention within our final model and therefore something to focus heavily on. However, surprisingly, students also had a tendency to accept bank marketing campaigns for term deposits based on prior data. As seen in figure 6, the age range of a student was also one of the most contacted groups as well. This is likely because these are the two groups that are the most financially free (or starting out in life) and are the ones most capable of pursing a term deposit for basic interest.

Using these features (while heavily relying on those who are retired as well as their ages), we will determine and improve the dataset and thusly the model mostly through focusing on age, occupation, as well as other factors where people might be financially free enough to pursue a subscription to a term deposit.

2. Predictive Task

2.1 Objective

Predicting if an individual will subscribe a term deposit or not (the column labeled as **y**) based on the provided features is a binary classification problem. We will conduct a train test split on our chosen dataset in order to train each model on a randomly-selected training set made up of eighty percent of all instances and test each model with the remaining twenty percent.

2.2 Evaluation

Of the 45,211 instances, only 5289 contained a y value of "yes", meaning that a response of "no" was nearly eight times as common. This indicates that we are conducting an imbalanced classification and would benefit heavily from avoiding using a metric such as accuracy to determine the merit of our predictive models because it weighs the two classification options equally whereas we would like to prioritize correctly predicting the positive class (responses of "yes"). Thus, using **recall** would be the most applicable tool in this particular use case because it simply depends on the predictions for the positive class, only taking into account the number of true positives (TP) and number of false negatives (FN).

$$\text{Recall} = TP / (TP+FN)$$

Additionally, we will be using every feature here as they all deal with one of two objectives key to determining if an individual will subscribe a term deposit. The first category deals with an individual's financials, that being from if they have a loan (housing or person) and other factors that may affect their financial stability and capability to afford a term deposit (such as job, education, and if they have a credit default). The second category deals with prior contact with an individual regarding previous marketing campaigns. These details include the amount of time it has been since they were last contacted and the outcome assuming they were contacted. For our use case, including all of the sixteen features did not lead to noticeable occurrences of over-fitting for any baseline model, so we decided on training all of them with every available feature for the sake of consistency.

2.3 Baseline Models

Before perfecting any models that are to become our final product, we will calculate the recall of five different options in order to determine which one shows the most potential once optimized by tweaking various hyperparameters. Each of these model options will work with all sixteen features contained in the dataset, but with the categorical data one-hot encoded so it can be treated as if it were quantitative. Additionally, the features that are already quantitative will all be standardized so they are all of the same scale.

Logistic Regression is one of the fastest models to train which makes it intriguing for a large dataset like ours. It also only classifies binary outcomes, which is what our dependent variable is. One drawback, however, is that it assumes a linear relationship between the independent and dependent variables, which is not the case with our dataset given many of the features are categorical and some of the features that are quantitative (such as age) likely have a relationship with y that is far from linear as seen during the exploratory data analysis where there was a strong correlation between age and willingness to subscribe a term deposit. Thus, while an interesting model to train, it likely will not put enough weight on certain variables and will underfit contrary to what we desire.

K-Nearest Neighbors is a model that can classify an instance based on the classifications of the instances with the most similar sets of attributes, especially when our EDA proved that similar groups (regarding age and occupation) tend to generate a positive y. KNN works well for non-linear data, which is beneficial for our use case. However, negatives of this model include it taking up a large amount of storage and running relatively slow. Additionally, a larger issue could attribute to the curse of dimensionality, especially when we train the model with every feature.

A trained **Decision Tree Classifier** results in a decision tree that can be used to make predictions from both categorical and numeric data. Due to the large number of hyperparameters that can be tweaked, it can have great potential if carefully developed to avoid over-fitting, something it can be prone too if made to be too complex.

A **Random Forest Classifier** considers the predictions of many different decision tree classifiers which are trained on different random subsets of the data. This reduces variance and the risk of over-fitting that is common with individual decision tree classifiers, but leads to a relatively slow runtime and less interpretability compared to a singular tree. This is an example of a bagging algorithm (an ensemble method) given the training is done on samples of the initial dataset. Datasets with imbalance class distributions, such as ours, may see that random forests are biased towards the majority class and therefore reduce the performance quality for the minority class. This is unfortunate given the minority class in our case (y is "yes") is the one we care about most.

Another ensemble method that can be applied to the decision tree classification model is boosting or **XGBoosting**. Instead of training many decision trees simultaneously, boosting train them one-by-one in order to modify the weights of each training instance between models in order to bring greater attention to the ones that were incorrectly classified. Similar to bagging, boosting still considers each of the models when making an aggregated prediction. This process helps in reducing the bias that can occur with bagging, but unfortunately does not reduce variance to the same degree.

Results of the Baselines

	Training Recall	Testing Recall
Logistic Regression	0.366581	0.339494
K-Nearest Neighbors	0.490026	0.343385
Decision Tree	1.000000	0.480545
Random Forest	1.000000	0.378405
XGBoost	0.703356	0.500000

Figure 8: Outcomes of training and testing recall for each of the baselines

Both the decision tree and random forest models excelled at predicting their training data perfectly, but did not hold up so well when it came to the testing recall which indicates they were over-fitting. Due to the boosting model resulting in the highest testing recall despite a much lower training recall, it appears to possess the greatest potential and will be what we optimize in order to create our final model.

3. Model

Initial Setting

Increasing the testing recall substantially from that of the baseline XGBoost model (exactly 0.5) is the objective of our final model.

3.1 Basic Hyperparameter Tuning

To gauge which model has the most potential While not over-fitting to the same degree as some of the other models, there is still some progress to be made with XGBoost that can be accomplished by tweaking some of the values of its hyperparameters.

3.2 Basic Feature Engineering

Our final model will continue with the same sixteen features as the baseline. As there was no missing data prior to testing, these features will be engineering in the same way. Qualitative data will be one-hot encoded in order to ensure that each of the categories they describe have weights associated with them. Quantitative data will instead be passed through a standard scaling procedure to reduce the effect of outliers affecting the final model.

3.3 Model Optimization

The most effective method for improving our XGBoost model is by having it focus more on the minority class category ("yes" in our case). This can be accomplished through modifying the scale_pos_weight[4] ("pos" being short for positive) hyperparameter to match the imbalanced nature of our binary dependent variable. A response of "no" being about eight times as prevalent as "yes" means setting this hyperparameter to 8 would be a great value to start out with as it is the ratio of negative class responses to positive ones. Setting it a greater value could possibly improve performance further, but runs the risk of over-fitting as well. This value being anything but its default value of 1 leads to a class-weighted version of XGBoost.

Another modification that can be made to XGBoost is the traits of each tree that contributes to its aggregated prediction. Altering how "deep" each tree can be with max_depth can change the complexity of each, but also lead them to over-fit if they become too complex. Therefore, providing lower max_depth values could prevent XGBoost from over-fitting as much as it is currently.

Ideal values for these hyperparameters can be calculated with the help of cross validation as this procedure will determine which combination of values allows the XGBoost model to best predict on unseen data.

3.4 Final Optimization Result

Cross validation indicated that scale_pos_weight being set to 8 and max_depth being kept at 6 was what led our XGBoost model to perform best on unseen data. This combination of hyperparameter values resulted in a training recall of about 0.977 and a testing recall of 0.813, a much greater value than that of the baseline XGBoost model.

4. Literature

This dataset has been used by past studies, but mainly in the context of improving algorithmic fairness in machine learning, specifically in the case of clustering algorithms. The following studies collectively highlight the versatility of the dataset and its ease of use by applying it to various challenges related to machine learning.

However, the main difference between articles discussing said studies and our own work with this dataset simply lays in what the dataset was used towards. Rather than a primary feature, the UCI bank dataset was more of a subsidiary to prove fairness in clustering algorithms in those articles. Thus, the state-of-the-art method was more akin to clustering as compared to classification. This dataset was likely used for that application due to its well-organized nature and lack of NaN values, resulting in minimal work needed to prepare it for use. Thus, the main

[4] https://machinelearningmastery.com/xgboost-for-imbalanced-classification/

novelty of our work is more so regarding the fact that we are using recall as an evaluation metric rather than proving a hypothesis based on a new form of clustering or fairness.

Clustering with Fairness Constraints: A Flexible and Scalable Approach[5]

The authors (Ziko, Granger, Yuan, and Ayed) do not intend to predict the outcome variable (term deposit subscription), but instead use the bank dataset as a control to contrast a synthetic dataset in order to demonstrate the effectiveness of their algorithm in handling real-world data with various demographic proportions. The main focus of the dataset is far different from our investigation. Instead of maximizing recall to correctly classify instances of data, the authors were more focused on comparing their method with existing fair-clustering methods to show the advantages of the method they had theorized regarding balancing fairness and clustering objectives.

While we focused on a more general overview (also focusing on past interactions with prior marketing campaigns, these authors constrained their dataset to create two initial clusters based on marital status to test fairness by dividing the data into three equal-sized groups, which was very different from the subset of data we used as we used every instance available.

Similar to one of our baselines, Ziko, Granger, Yuan, and Ayed proposed a clustering algorithm that involved K-means clustering.

Fair Algorithms for Clustering[6]

The authors (Bera, Chakrabarty, Flores, and Negahbani) in this article also do not intend to predict the outcome variable. Similar to the previous article, these authors used the UCI bank dataset as a way to focus on metrics to solve the problem of finding low-cost fair clustering methods. This dataset was one of several used to evaluate their algorithm. They focused on fair clustering, ensuring that each cluster had fair representation of different groups in the data. For clustering, the authors employed algorithms based on the lp-norm objective, which includes k-means, k-median, and k-center objectives.

While somewhat similar to the XGBoosting and k-nearest neighbors we tuned parameters for during our baseline, these authors were less concerned about the performance of the model based on predicting the outcome variable and more concerned about how the model they had created stacked up against additional, state-of-the-art models that were normally used. While these authors ended up focusing more on the k-means objective, our project looked more in-depth into XGBoost. Additionally, we also focused on tuning hyperparameters to increase the recall of our model while the authors of this article were more concerned with allowing groups to lie in multiple protected groups.

ToPs: Ensemble Learning with Trees of Predictors[7]

This article (Jinsung Yoon, William R. Zame, and Mihaela van der Schaar, Fellow, IEEE) was the most similar to the model that we created. Here, the authors test the difference in two instantiations of ToPs (trees of predictors) that included adaboost, linear regression, logistic regression, logitboost, and random forests. Many of which we used in our baseline and final model. These instantiations allowed them to explore improvement of models, and also allowed them to compare the performance of their models to create a new approach to ensemble learning and prove that their method was better than other casual uses of state-of-the-art-methods in terms.

These authors employed a similar prediction model in predicting for the same y-variable (which allowed for a comprehensive evaluation of the ToPs algorithm's effectiveness in comparison to other machine learning algorithms in predicting the acceptance of bank marketing offers based on clientele features. However, the main difference regarded on how they used their new approach. Unlike our project (which used recall), the authors here utilized area under the ROC curve as the loss function due to the unbalanced nature of the dataset that we also noticed.

5. Result

5.1 Comparison

When compared to the five baseline models, the final XGBoost model outperformed them all by 60% on the low end and well over 100% in other cases. Thus, it was settled on as the final product. The primary factor at play are the hyperparameter adjustments made for XGBoost in particular because we were able to focus more heavily towards the minority class category and consequently improve the testing recall of what came to be our final model.

[5] https://arxiv.org/pdf/1906.08207v1.pdf

[6] https://proceedings.neurips.cc/paper_files/paper/2019/file/fc192b0 c0d270dbf41870a63a8c76c2f-Paper.pdf

[7] https://arxiv.org/pdf/1706.01396.pdf

Choosing XGBoost out of the five baseline models in the first place was a safe choice given it already possessed the highest testing recall and had a training recall that did not indicate severe over-fitting was at play. These two hyperparameters were chosen primarily because of the wide difference in the ratio of positives to negative responses as well as how "deep" a tree could be being a major factor in complexity, thus possibly limiting how overfit the final model was. Both of these hyperparameters increased the efficacy of the final model. Additionally, by having too low of a scale_pos_weight, one would be able to sabotage the recall of the model as it would assume equal importance between positive and negative values.

5.2 Significance

In any situation, the testing recall jumping from 0.5 to over 0.8 from the baseline model to the final model is significant improvement, especially when the respective baseline was already competitive relative to the four other options. Additionally, the final model having a recall of 0.813 shows great efficacy in being able to determine true positives and predict if an individual with given features is likely to subscribe a term deposit. That is a substantially larger value than 0.113, the recall that would come from making predictions based solely off of the proportion of response values that are positive.

5.3 Conclusions

Various takeaways about the data mining process can be extracted from the work conducted on this particular final model.

- The ideal hyperparameter to tune is the one that directly addresses the current weakness of the model as it relates to the characteristics of the dataset at hand. This was seen in scale_pos_weight having such a substantial impact in remedying the imbalanced class issue.
- It is very possible that a hyperparameter's default value is the ideal one for the job at hand. This was the case for max_depth given cross validation did not show that changing it from the baseline's value of 6 resulted in a greater testing recall.
- Taking into consideration how the data for each feature was generated can help in gauging how much impact it will have on the model. This is shown in how the campaign contacted middle-aged individual less recently than their younger and older counterparts.