Predicting the Success of a Bank Marketing Campaign using Machine Learning

Nitya Rudaraju
Boston University

Rahul Srivatsa
Boston University

Andrés Rivera
Boston College

Kabir Oberoi
Boston University

June 28, 2019

Abstract

Using data from a Portuguese Bank's telemarketing campaign which sold Term-Deposits, multiple machine learning algorithms were implemented in order to classify whether or not a given customer would subscribe to a term deposit. The benefits of such a classification include better targeted marketing campaigns and therefore cost reductions as well minimal negative externalities to those that would not like to subscribe to a term deposit. Through data cleaning, feature selection and feature engineering the dataset is developed and run through the algorithms(Support Vector Machines, Decision Trees, Naive Bayes, K-nn and Logistic regression. We conclude that Naive Bayes works best at this classification task when measured on Accuracy scores, F-scores and Run-times.

1 Introduction

The importance of marketing in today's capitalistic society is undebated. The key to successful marketing is simply understanding who the target market is. Based on this, different strategies to reach the consumer base can be evaluated. It is difficult to identify a successful marketing strategy because each customer is different and there are often many different features which affect customer behavior. Usually there are simply too many customers and variables to analyze for a person to predict the success of a marketing strategy- which is where machine learning steps in.

This project deals with a dataset containing information from a Portugese bank's direct telemarketing campaign, including several details about the customers they contacted. The customers were classified based on whether or not they subscribed to a term deposit at the end of the phone call - either yes or no. The features of each customer included a number of social (marital status, education), economic (job, consumer price index) and personal (age) details. The training data set contained 20 different customer attributes/features plus the classification of whether or not they signed up for the term deposit. The dataset was subject to several machine learning techniques to build a data-driven prediction model to predict the success of bank telemarketing. This will help determine how financial institutions can improve effectiveness for future marketing campaigns.

2 Data cleaning

2.1 Value replacement

The first step in data preprocessing is to get rid of any null values. In order to check the number of null values for each feature, dataframe.isnull().sum() can be used. Fortunately, this data set was complete and did not contain any null values to clean. Although there were no null values to replace, there was a replacement of the binary classification values in the feature 'y'. The string 'yes' was replaced with 1 and 'no' was replaced with 0 to create a more manageable data.

2.2 Feature selection

Next, an exploratory data analysis (EDA) was conducted to better understand the features theoretically and intuitively deduce if any of them should be removed from the data set. The code used to return the range of values, value counts, mean and other important statistical features for our EDA is included in Appendix A.

The 'default' feature was dropped, because out of the entire dataset there were only 3 people who were credit defaulters. For the 'education' feature, it was assumed that there wasn't much of a difference between having a 4 year, 6 year or 9 year basic education, so

these three categorical values were combined into a single one. The other categories such as a college degree, high school GED etc were left alone. Finally, the last feature that was dropped was 'duration'. According to UCI, this attribute highly affects the output target (e.g., if duration=0 then classification='no'/0). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model. Furthermore, Several features in the dataset were categorical variables. These categorical variables were binarized through One hot encoding. One hot encoding creates new (binary) columns for each category of a categorical feature.

2.3 Normalization

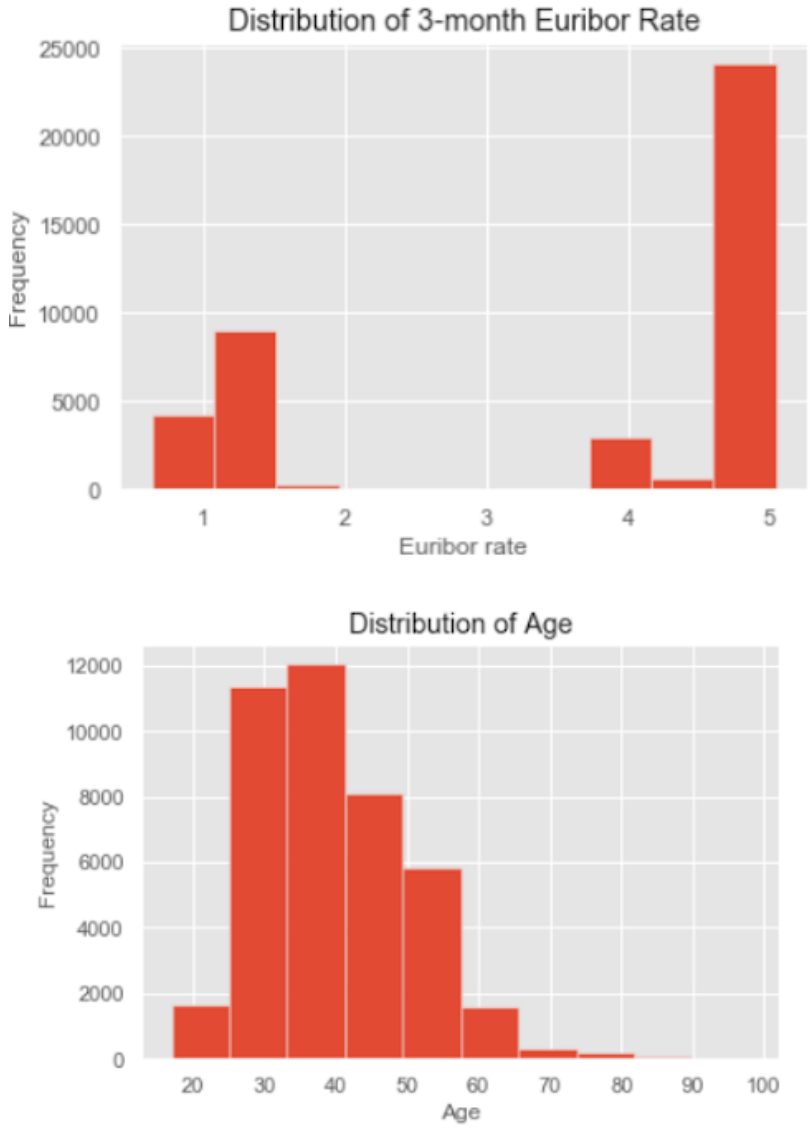
Once the categorical values were cleaned with one hot encoding, the numerical features were normalized. There were 9 different numerical features(the duration feature was dropped from the dataset), and each one of them had values in dissimilar ranges. In order to compare these features these values had to be adjusted to a notationally common scale, by having the range of values be 0-1 for all of the numerical features.

3 Data visualization

The initial purpose of the data visualization component is to intuitively identify and explain features that may have predictive power over a customer's decision on whether or not to purchase term-deposits. Intuitively, age plays an important role in determining investment decisions. Usually individuals who are well established into their careers would have the sufficient funds to invest in term deposits. Furthermore, older individuals would be less likely to invest in term-deposits as they value cash now rather than later. Young professionals on the other hand could be willing to invest in term-deposits however, they would face capital constraints. These intuitions were indeed captured by a simple histogram of age. The histogram indicates that mean age is approximately 40 years with a standard deviation of approximately 10 years. Young professionals (aged 20-30) do tend to invest less as they are entering their careers and need to accumulate capital before deciding to invest in long term facilities. There is again a drop after the 50 age band as it makes little sense to invest in long term deposits at that stage.

The 3-month Euribor (European Interbank Offer rate) is another important feature that is a critical factor in financial decision making. The Euribor rate is akin to the Libor (London Interbank Rate) and is the rate at which banks in Europe lend to each other. The Euribor rate is also similar in effect to the Federal Funds Rate in America but is computed differently. A higher rate would imply a greater "bang for buck" through compounding over the years. Since term-deposits are usually negotiated at fixed rates, it would make sense that customers who were called when the rates were higher would have invested more. This hypothesis is also partially conveyed by the histogram of Euribor rates. The Euribor rate at 5 percent has the highest frequency of individuals. However, there is a concernable frequency of individuals under 1 and 1.5 percent Euribor rates as well. We hypothesized about other economic indicators such as consumer price index (captures inflation) and Consumer confidence index (captures optimism in the markets) in a similar manner.

Figure 1: Histograms of the '3-month Euribor Rate' and 'Age'



3.1 Correlation heatmap

The correlation heatmap (Figure a.1: Appendix A) was created using the seaborn package and the drawheatmap function to include the entire dataframe. The pearson correlation coefficient was used to evaluate the relationship between features. We observe in the heatmap that most of the features do not have strong correlations. However, in the bottom right quadrant of the heatmap, we observe that there are a few features that have high correlation. These features are namely; No. of people employed, Employee variability rate, Consumer confidence index and the 3-month Euribor. Generally, we would drop highly correlated variables to prevent overfitting the model. However, upon running our Decision trees model, we observed that these variables have high feature importance, especially No. of people employed as it appears on the first layer of the Decision Tree. Moreover, as we have discussed above, Euribor rates the other economic indicators are very influential factors in financial decision making. Therefore, none of the variables were dropped in this stage of the feature selection.

3.2 Principal component analysis

As a result of the One-hot encoding, all the categorical features were converted into numerical ones, where each category of a categorical feature was converted to dummy variable. While this eases computation, it leads to a significant increase in the number of features in the model. The One-hot encoding procedure increased the number of features from 21 to 57, increasing the dimensionality more than twofold. Therefore, there were concerned about overfitting in the model. Using the Scikit Learn package PCA, the entire dataframe was fit into the PCA function. Thereafter the Explained variation ratio for each was component and plot it onto the Cumulative variation Table a.1 (Appendix A). The graph shows that the first 46 features explain approximately 100% of the variation in the model. Therefore, these variables were dropped and the model was rerun. However, this yielded negative results. In fact, taking a look at Table 2 (Appendix D), we see that accuracies and f-scores decrease for all algorithms when compared to table 1 which does not drop these features. Therefore, these features were not dropped from the final model.

4 Model building

The dataset was split into a training set and testing set (70% - 30%). The training set was used to train a particular model, and then cross-validate the errors to find optimum parameters.

SVM:

Python provides the package sklearn.svm.SVC. The support vector machine algorithm finds a hyperplane in an N-dimensional space that distinctly classifies the data points. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using the 'kernel trick' and implicitly mapping their inputs into high-dimensional feature spaces.

Naive Bayes:

Python provides the package sklearn.naivebayes.GaussianNB. Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle. It's relatively simple to understand and build . A Naive Bayes model is easily trained and is not sensitive to irrelevant features. It assumes every feature is independent, which isn't always the case. Due to the simplicity of the naive assumption, this method was chosen as a benchmark to compare with other hyper parameterized models.

KNN:

Python provides the package sklearn.neighbors.KNearestClassifier. K-Nearest Neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure and in this instance, it is the Euclidean distance.

Decision Tree:

Python provides the package sklearn.tree.DecisionTreeClassifier for the decision tree classifier. Decision trees are a simple yet effective method for classification. Using a tree structure, this algorithm splits the data set based on one feature at every node until all the data in the leaf belongs to the same class. The criterion used for splitting is called information gain, which is based on a purity measure called entropy, a measure of disorder. The set with the highest impurity will have higher entropy whereas the set which has higher purity will have lower entropy. Information gain measures the change in entropy due to the amount of information

added. The higher the information gain, the more information that feature provides about the target variable.

Logistic Regression:

Python provides the package sklearn.linear model.LogisticRegression for Logistic Regression. It is a well known classification model.The linear model fits the training data to the equation: $y = w_0 + w1_{x1} + w2_{x2} + \ldots w_n x_n$ (where y stands for the target variable, w_0 stands for the y intercept, $x_1, x_2, x_3 \ldots x_n$ are feature vectors, and $w_1, w_2, w_3 \ldots w_n$ are their corresponding weights) while the logistic regression algorithm uses the same decision boundary with bit modifications as shown:Logistic regression is used because classification is not exactly a linear function and using linear regression produces an output within $[-\infty, \infty]$ while the probability has to be within $[0, 1]$. The logistic function itself does output the probability of an instance belonging to the positive class. This output probability does indeed have a range of $[0, 1]$, hence overcoming the drawbacks of classification using a linear model.

4.1 Model evaluation

Figure 2: Accuracy for models implemented

	train_time	pred_time	acc_test	f_test
SVM	121.082808	15.922179	0.898624	0.872053
NB	0.074832	0.058744	0.860001	0.864083
KNN	2.474418	41.934086	0.891268	0.874722
DT	0.118184	0.015378	0.898624	0.872131
LR	0.595937	0.007347	0.899213	0.875101

From the table above we can conclude that all the different classifiers achieve a good accuracy. We then evaluated a model based on it's training and testing time shown in Figure 2. Kernelized SVMs require the computation of a distance function between each point in the dataset, which is the dominating cost of (features2observations)O(nfeaturesnobservations2). The storage of the distances is a burden on memory, so they have to be recomputed. The KNN algorithms takes a significantly high testing time since it needs to compute the similarity of each testing sample with that of all the training samples.

4.2 Hyper-parameter tuning

For each model implemented, the hyper-parameters were tuned to obtain the optimal performance of the classifier. Logistic Regression: For Logistic Regression, two hyper- parameters were tuned: the penalty type ('L1' or 'L2' penalty) and the regularization coefficient ('C': 104 to 105 on the log scale).

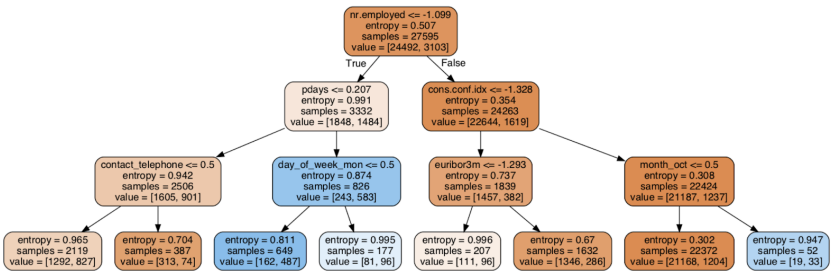
KNN:

The elbow method of plotting the error function against different values of K was used to find the optimal choice for K.

Decision Tree:

A decision tree is a relatively fast model to implement. By using sklearn.modelselection.GridSearchCV, multiple combinations of parameter max depth and minimum sample split were to give the optimal F1-score.

Figure 3: Decision Tree with Max Depth of 3



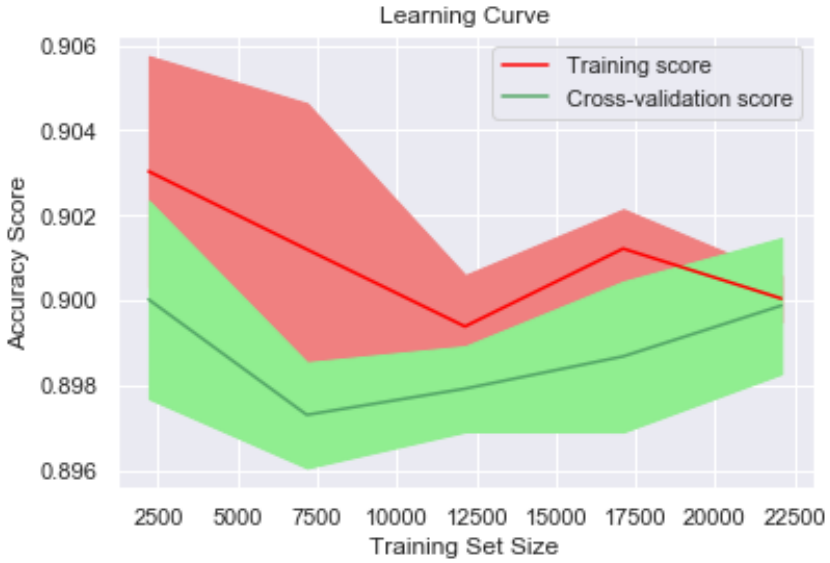
Support Vector Machines:

The parameters 'C' and 'kernel' were tuned using GridSearchCV. An initial run on our entire dataset took more than 12 hours and had to be shut down. Therefore, the tuning was done on a smaller version of the dataset from the same UCI database which contained fewer samples and was quicker to compute

Moreover, the validation and learning curves for each model was created to check where the model was underfitting or overfitting. In the example of decision trees below, the optimal depth is between 3 and 4. Starting at a depth of 5, the split between the training score and the cross-validation score rises rapidly.

Figure 4: Validation Curve for Decision Tree

Figure 5: Decision Tree with Max Depth of 3



4.3 Results

After optimizing the parameters to fit the data, the accuracy scores showed slight improvements. It is important to note that so far we have compared models based on accuracy. Due to the nature of the given dataset, only about 12% of samples were classified as with label of 1. Therefore, even if a model predicted 0's , the model would have a good accuracy. We thus considered the recall metric to compare our models. The Decision Tree model scored the highest on recall for amongst our models but was still outperformed by our benchmark Naive Bayes model. The data described is tabulated as follows:

```
[ ]: 1 print( '\n',report_DT)
```

```
                precision    recall  f1-score   support

           0       0.91      0.99      0.95     12056
           1       0.69      0.19      0.30      1537

   micro avg       0.90      0.90      0.90     13593
   macro avg       0.80      0.59      0.62     13593
weighted avg       0.88      0.90      0.87     13593
```

```
[ ]: 1
     2 print('\n', report_NB)
```

```
                precision    recall  f1-score   support

           0       0.93      0.91      0.92     12056
           1       0.40      0.45      0.42      1537

   micro avg       0.86      0.86      0.86     13593
   macro avg       0.66      0.68      0.67     13593
weighted avg       0.87      0.86      0.86     13593
```

Conclusion

Although the Naive Bayes model compared slightly low in accuracy, it had the best recall for the label 1. This metric is more important in the domain of telemarketing since generally the objective is to correctly identifying a person who will make a term deposit even at the cost of incorrectly identifying 0's. Therefore, Naive Bayes is the most appropriate model to use in this case.

The implications of such a classification are endless as long as accurate and extensive feature data is collected. These classification models can be applied to customer acquisition issues in several fields such as Finance, where there is great variability in customer acquisition. Furthermore, kaggle competitions like the yelp review classification can also use these models, although they involve Natural Language Processing.

References

"Using Categorical Data with One Hot Encoding." Kaggle, www.kaggle.com/dansbecker/using-categorical-data-with-one-hot-encoding.

"Bank Marketing Data Set ." UCI Machine Learning Repository: Bank Marketing Data Set, archive.ics.uci.edu/ml/datasets/Bank+Marketing.

Appendix A

Figure a.1 : Correlation Heatmap of the Pearson Correlation Coefficients

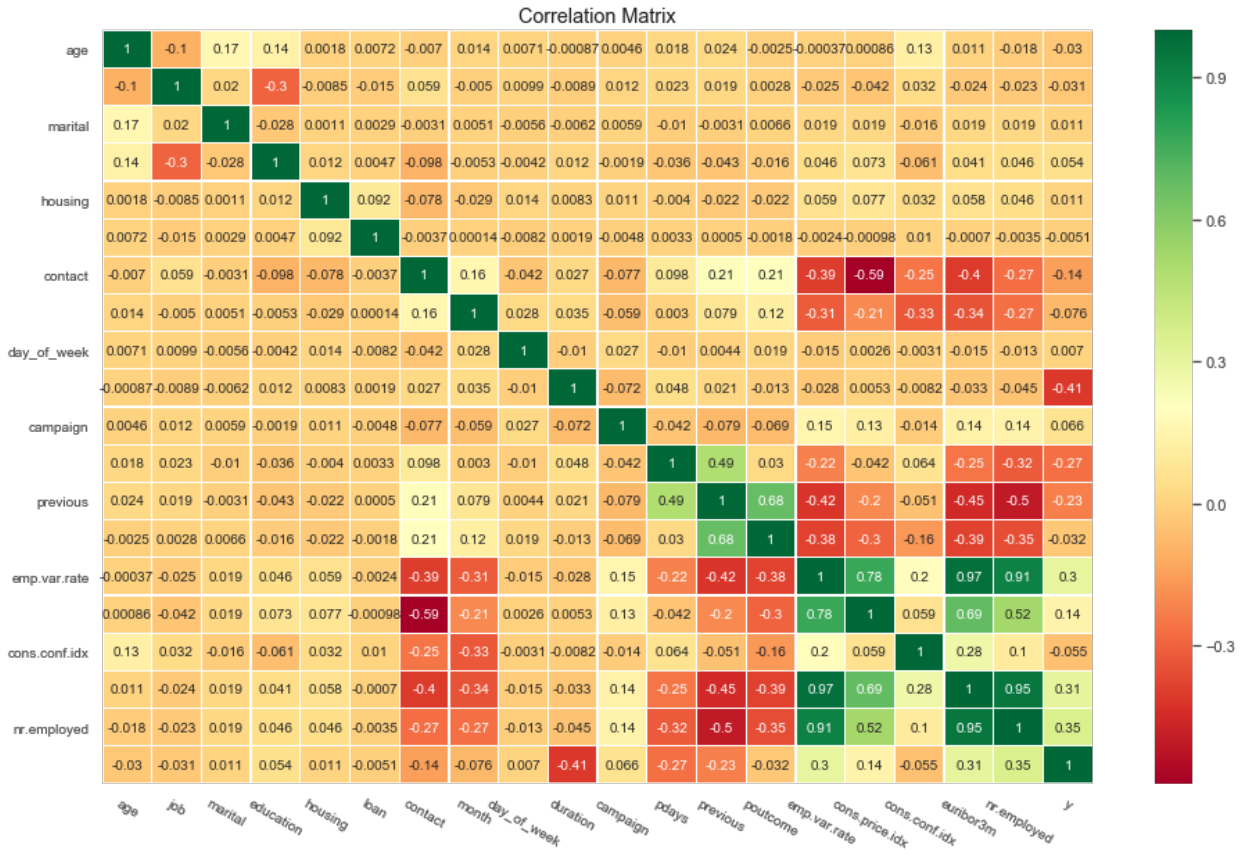


Table a.1: Results when Features are dropped based on PCA analysis

	train_time	pred_time	acc_test	f_test
SVM	104.460306	13.897518	0.898477	0.871557
NB	0.076867	0.043700	0.853822	0.860708
KNN	0.798692	18.428260	0.893401	0.876861
DT	0.077158	0.008623	0.898477	0.871790
LR	0.473227	0.009273	0.898919	0.875662