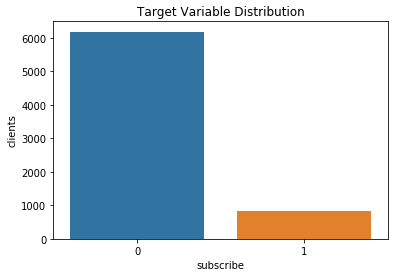
**Banking Telemarketing – Individual Assignment**

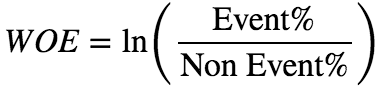
The main purpose of this project is to build a data driven model to predict if a bank client subscribes or not to a term deposit from several input variables. The target variable is binary, with a value of 1 corresponding to a client having subscribed and 0 to a client with no subscription. Because of this characteristic, classification models were used, with the final 5 models being chosen according to their performance under the AUC metric and their ability to be understood, considering business setting conditions. The following document details the final iteration of the process and final 5 models used, although the accompanying code also includes additional techniques tested, such as undersampling and other classification models.

To evaluate the models, the original train dataset provided was randomly split into train (60%), validation (20%) and test set (20%). The original test set provided without target variable information was left as a holdout dataset to be unseen by the models. The purpose of this was to select models trying to avoid overfitting, especially considering the unbalance of the target variable, as seen on the figure below, created from the full train set.



To improve the model’s classification power, data pre-processing was applied consistently on all sets, starting from the split train set and later applying the same transformations to all other datasets. The first step was to check for missing values, with ‘unknown’ being considered missing in categorical variables. The columns *loan*, *housing*, *job*, *education* and *marital* displayed less than 5% missing values. These observations were dropped, with the exception of *job*, as they were treated later on using a weight of evidence approach, described in the next paragraph. The variable *default* showed around 20% of observations with ‘unknown’ as the value, so these observations were kept and used as a dummy variable. No missing values were found for numerical variables.

Once the missing values were cleaned, variables with too many values were binned using weight of evidence (WOE) and information value (IV) with respect to the target variable of *subscribe*. The purpose of this technique is to establish a monotonic relationship (either positive or negative) between the independent and dependent variable and to ensure selection of relevant variables. Plus, the transformation is based on logarithmic value of the distributions, as shown in the formulas below, aligning it with the logistic regression output.



https://miro.medium.com/max/600/1*9Gi0fGyTpxfwM2TpV4GZQQ.png

IV was calculated for all columns, with existing literature suggesting to focus on the variables that are medium or strong predictors. Out of the variables that turned out to be medium or strong predictors, *age* and *job* were selected for binning, because they have enough values to do so, while the other variables could be converted into dummy variables.

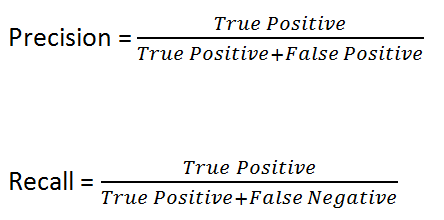
|  |  |  |
| --- | --- | --- |
| Information Value (IV) | Predictive Power | Variable Name |
| <0.02 | Useless for prediction | campaign, marital, loan, housing |
| 0.02 to 0.1 | Weak predictor | day\_of\_week, education |
| 0.1 to 0.3 | Medium predictor | default, job, contact |
| 0.3 to 0.5 | Strong predictor | age, previous |
| >0.5 | Suspicious value | euribor3m, emp.var.rate, nr.employed, pdays, poutcome, month, cons.price.idx, cons.conf.idx |

Categorical variables deemed useless or weak for prediction, such as *day\_of\_week*, *education*, *marital*, *default*, *housing* and *loan*, were transformed to n-1 dummy variables, while *campaign* was transformed into *n\_campaign*, with 1 for multiple campaign contacts, else taking a value of 0.

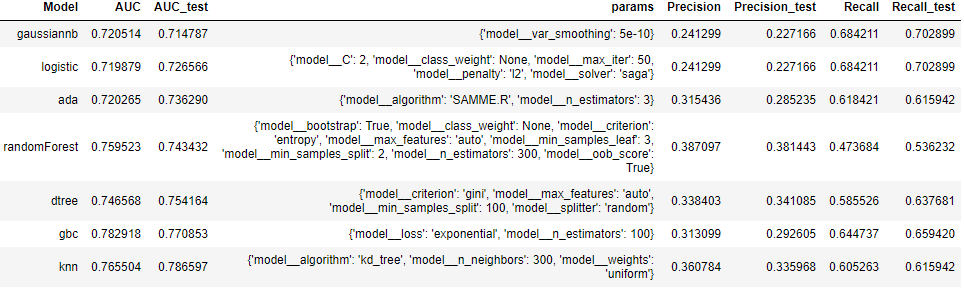
Finally, new variables were created. The variable *month* displays a suspicious predictive power and so was grouped by seasons -*winter*, *summer*, and *spring*, to create n-1 dummy variables. The variable *pdays* also results in suspicious predictive power and so was transformed to a binary variable called *contacted*, taking a value of 1 for clients not contacted (a value of 999 for *pdays*), and 1 for contacted (all other *pdays* values). Other numerical variables with suspicious predictive power were applied a standard scaling transformation.

The final data sets resulted in 36 independent variables, detailed in the appendix, plus the target binary variable and *client\_id* identifier. However, only the top 20 independent variables ranked by Fisher score were used in the models.

For model selection, a pipeline was built including Synthetic Minority Over-sampling Technique (SMOTE) and a grid search with cross validation. The oversampling was used in an attempt to avoid overfitting when building a model from an unbalanced dataset. The purpose of the grid search was to select the best parameters, measured by AUC. Additionally, Precision, Recall and overall Confusion Matrix were also looked at to get a better understanding of the models predictive power and to ensure a maximization of predicting positive events (clients subscribing).



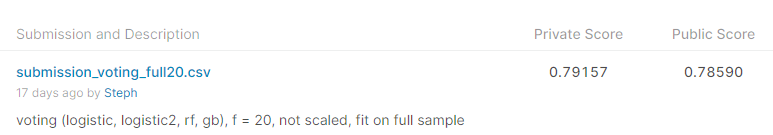
The experimental setup consisted of training the grid search cross validation for each model on 60% of the training data, validating on 20% and finally evaluating on the remaining 20%. AUC scores were compared to avoid selecting overfitted models by checking the scores did not go down too much on the test set. The same comparison was done on Precision and Recall metrics. Overall, the grid search using this split setup resulted in relatively stable models when judged by AUC, with the output displayed below:



However, when looking at precision and recall scores, the models by themselves seemed to be more volatile. Because of this, the models were refit, using the best parameters for each model saved from the grid search and different combinations of models were tested using a Voting Classifier. The idea behind this was to stabilize predictions, again combat overfitting and also to fight one model’s weaknesses with another model’s strengths.

To maximize the Voting Classifier’s power, a loop was built which added a new model to the classifier only if it increased the test AUC. Multiple submissions were made using this approach. In the end, the top performing submission (looking at the Kaggle private score) was a voting classifier consisting of two logistic regressions, one random forest and one Gaussian bayes model, fit on the full training data (60% training, 20% validation and 20% test set). One of the logistic regressions used liblinear as solver with penalty l1, while the other used newton-cg as a solver and no penalty, both set with 500 max iterations. Random forest classifier used bootstrap, 100 estimators, entropy criterion, no class weight, no oob score and sqrt max features. Gaussian bayes used the default parameters. The reason for using these parameters was because they were consistently popping up after multiple runs of the grid search. Additionally, they were consistently achieving decent scores on the Kaggle public leaderboard.

Unfortunately, due to an excessive amount of submissions (oops!), this entry did not make it to the final score.



The top 5 performing classification algorithms, over all the different voting classifier combinations attempted, are described below:

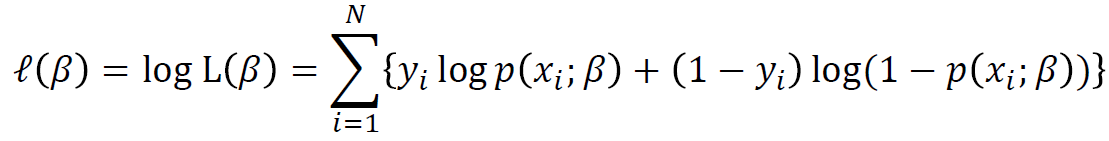
**Binary Logistic Regression:** a classification algorithm that models the probability that an observation belongs to a particular category. To do this, the probabilities are modelled using the logistic function, which gives outputs between 0 and 1 for all values of the independent variables. The coefficients are chosen to maximize the likelihood equation, which in this case means that predicted probabilities of subscribing for each client, using the logistic function below, match as closely as possible to the client’s true subscription status. This means probabilities are calculated using the logistic function and then replaced into the maximum likelihood function. However, maximum likelihood is unstable when differentiating to maximize, which is why log likelihood is used instead (monotonically increasing function).

Overall, logistic regression is a well-known algorithm and therefore it is easy to implement. Its coefficients have a linear relationship with the target variable, which is easy to explain in a business context.

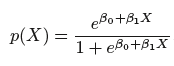
Maximum likelihood function:

****

Log Likelihood Function:

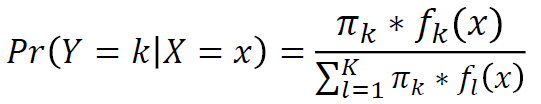
****

Logistic function:



**Random Forest:** this model build several decision trees on bootstrapped (or replaced) training samples and decorrelates them by using different predictors. By using different subsets of predictors, other predictors have a chance to be part of a tree, even if there is a very strong predictor. Out of a total of p predicts, m are selected at random at every tree split, with 𝑚≃√𝑝. By averaging many uncorrelated trees, a large reduction in variance can be seen, which is one of the downsides of using single decision trees. This is very useful in relatively small datasets, like the one used in this project. Another advantage of random forest is its ability to fit unknown non-linear and complex interactions of features with minimal feature engineering. Additionally, random forest can also be easily explained, by relying on their basic units of decision trees.

**Gaussian Naïve Bayes Classifier:** this algorithm consists of the Naïve Bayes theorem, which assumes independence among predictors, and a Gaussian (or approximately normal) distribution. This algorithm fits a model by converting the data set into a frequency table, grouped by clients who subscribed and clients who haven’t subscribed. Then, it creates the likelihood table by finding the probabilities for each class and variable, as shown in the function below. The advantage of this algorithm is that it has been proven to work numerous times in different classification scenarios, requiring small amounts of training data. Disadvantages are that it is better used in cases when all predictors are continuous and its predictive power is not too strong.



**AdaBoost Classifier:** this algorithm is an example of a boosted classification tree. It begins by fitting a weak classifier on the original dataset, such as a small decision tree. Then, additional copies of the classifier are fit on the same dataset, but adjusting where it made incorrect classification by applying different weights to the training sample. This allows for incorrect samples to have a higher weight, while samples correctly predicted are given a lower weight. Therefore, subsequent classifiers focus more on those challenging cases. The final prediction is calculated by combining all these weighted votes. This usually means that using more estimators decreases the error, but it can also quickly lead to overfitting.

**KNN Classifier:** this method attempts to estimate the conditional distribution of Y given X, and then classify an observation to the class with highest estimated probability. To do this, the KNN classifier first identifies the K points in the training data that are closest to an observation (nearest neighbors, N0) and then estimates the conditional probability for a class (for example, subscription) as the fraction of neighbour observations with subscriptions, as represented by the function below. Finally, the Bayes rule is applied and the observation is classified to the class with the largest probability. The advantages of this approach is that overall it has a good performance, with existing literature showing it often results in classifications very close to the optimal Bayes classifier. Additionally, the right amount of nearest neighbours can make the model robust to noisy data. Unfortunately, one of the disadvantages of KNN is how important it is to select the correct level of flexibility by selecting the amount of nearest neighbours. A number that is too low can lead to overfitting by allowing too much flexibility, a number that is too high can lead to underfitting by not allowing enough flexibility.



**Appendix**

|  |  |  |
| --- | --- | --- |
| **N** | **Variable (bold for top 20 in Fisher score)** | **Definition** |
| 1 | subscribe | Target variable, 1 for term deposit subscription else 0 |
| 2 | client\_id | Unique client id, for identification purposes only |
| 3 | **euribor3m** | Numeric daily indicator of euribor 3 month rate |
| 4 | nr.employed | Numeric quarterly indicator of number of employees |
| 5 | **emp.var.rate** | Numeric quarterly indicator of employment variation rate |
| 6 | **previous** | Number of contacts before this campaign |
| 7 | **cons.price.idx** | Numeric monthly indicator of consumer price index |
| 8 | **cons.conf.idx** | Numeric monthly indicator of consumer confidence index |
| 9 | job\_binned\_technician + management + admin. | Binned dummy variables (n-1), using WoE, from original job variable |
| 10 | **job\_binned\_blue\_collar + services + misc. level pos** |
| 11 | marital\_married | Dummy variables (n-1) from original marital variable, showing marital status as a categorical variable |
| 12 | **marital\_single** |
| 13 | **education\_basic.6y** | Dummy variables (n-1) from original education variable, showing education level as a categorical variable |
| 14 | **education\_basic.9y** |
| 15 | education\_high.school |
| 16 | education\_illiterate |
| 17 | education\_professional.course |
| 18 | **education\_university.degree** |
| 19 | day\_of\_week\_mon | Dummy variables (n-1) from original day of week variable |
| 20 | day\_of\_week\_tue |
| 21 | day\_of\_week\_wed |
| 22 | day\_of\_week\_thu |
| 23 | **poutcome\_nonexistent** | Dummy variables (n-1) from original poutcome variable, showing previous campaign outcome as a categorical variable |
| 24 | **poutcome\_success** |
| 25 | age\_dist\_Age\_31\_to\_36 | Binned age dummy variables (n-1), using percentiles, from original age variable |
| 26 | **age\_dist\_Age\_36\_to\_41** |
| 27 | **age\_dist\_Age\_41\_to\_50** |
| 28 | **age\_dist\_Age\_gt\_50** |
| 29 | age\_mean | Categorical variable with 1 if client age is above mean, else 0 |
| 30 | **winter** | Dummy variables (n-1) grouping original month variable by season |
| 31 | spring |
| 32 | **summer** |
| 33 | **contact\_telephone** | Dummy variable (n-1) from original contact variable |
| 34 | **default\_unknown** | Dummy variable (n-1) from original default variable |
| 35 | housing\_yes | Dummy variable (n-1) from original housing variable |
| 36 | loan\_yes | Dummy variable (n-1) from original loan variable |
| 37 | **contacted** | Categorical variable from pdays, with 0=999 (no contact), else 1 |
| 38 | n\_campaign | Categorical variable from campaign, with 1 for multiple contacts, else 0 |

**References**

* <https://medium.com/@sundarstyles89/weight-of-evidence-and-information-value-using-python-6f05072e83eb>
* <https://towardsdatascience.com/attribute-relevance-analysis-in-python-iv-and-woe-b5651443fc04>
* <https://scikit-learn.org/>
* A data-driven approach to predict the success of bank telemarketing. 2014. Sérgio Moro, Paulo Cortez, Paulo Rita.
* An Introduction to Statistical Learning: With Applications in R