IÉSEG SCHOOL OF MANAGEMENT

STATISTICAL AND MACHINE LEARNING FOR MARKETING

INDIVIDUAL ASSIGNMENT

Pedro Román Infante

Contents

Introduction

Statistical learning is a field of study that involves the application of statistics and machine learning algorithms to a set of data with the goal of obtaining either insight or foreseeing behaviors through the data. This discipline has been put to practice in the market for over 30 years. Hence, over time and with the rise of new challenges in its application multiple algorithms have been developed and proven to be successful. This project aims to deeply understand 5 of these common Machine Learning algorithms by explaining the science behind their implementation. At the same time, by creating a benchmark experiment, it will be demonstrated that every ML method has its purpose and that there is not an ultimate winner in the race of predictive analytics.

Project Description

Problem Definition

A campaign is launched within a financial institution, and it is of interest to foresee its success rate, measured by the subscription of clients to the product offered by the campaign. A data set belonging to the institution is given, with clients as granularity, and considering the following types of variables: general data about each client, last contact of an ongoing campaign, and results from previous campaigns.

Objectives

- Apply ML algorithms to predict the subscription of clients after a campaign is launched.
- Generate evaluation metrics to comprehend the performance of all ML models.
- Interpret the ML algorithms results and determine which is the best fit for this problem.

Machine Learning Algorithms

Logistic Regression

Considering that we are facing a classification problem (whether a client will subscribe or not) we can build a logistic regression model to predict the target variable. Logistic regression is an algorithm used to predict qualitative variables from a training set of observations. In comparison with the linear regression, where an ordinal predetermined order is assumed for the target variable and takes the least square to obtain the coefficients that minimize the RSS, the logistic regression generates the probability that the target variable could be identified as part of a certain category.

In a general context, given x as some feature, and y as the target variable of the prediction, the probability of y being 1 given the feature x can be represented as:

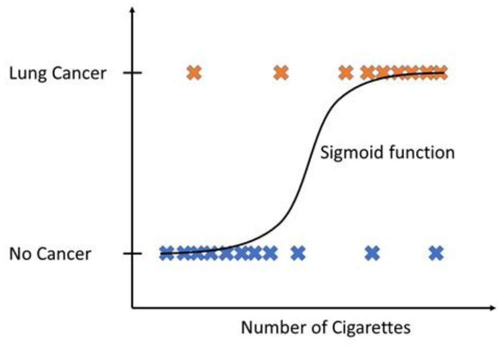
$$P(y = 1 \mid x)$$

The probability obtained from the relationship presented before would result in a value between 0 and 1. Then, based on the nature of the problem a threshold is set, and anything that crosses that threshold is considered to belong to the category in question.

Classifications obtained from logistic regression models are based on the following formula:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

This function is known as the Sigmoid function which allows partitioning of the data points due to its nature of S shape, within a probability range of (0,1). A representation of this function is shown in the image below:



To fit a Logistic Regression model a concept called maximum likelihood is applied. The Maximum Likelihood Estimation (MLE) is a method used to obtain parameters of a function such that the predicted probability for the target of each observation corresponds as much as possible to the real target of the observation. From the MLE we can obtain the coefficient estimates corresponding to each feature that makes the predicted probability as similar as possible to the real value of the target.

Once we have the coefficient estimates of each feature then we can calculate the probability of an observation belonging to the target class by substituting the values of the feature(s) and their corresponding coefficient estimates. In case we have a multiclass classification problem, we would apply the LR model n-1 times (being n the total number of classes that we want to predict), changing the target on every occasion and the remaining class would be $1 - \sum_{i=1}^{n-1} P(Y = y_i \mid x)$. This process can be applied to both single logistic regression and multiple logistic regression, which can be generalized with the following formula:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}$$

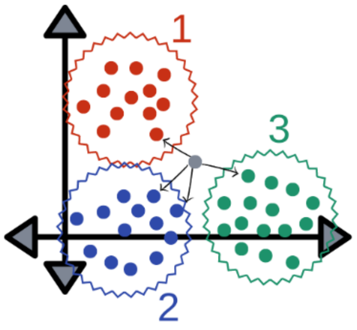
Applying this model can be beneficial when it is necessary to have a high interpretation of the output, feature importance, or obtain the direction of association of the predictors. It is also easily implemented and can be used to classify unknown records fast while not assuming the distribution of classes in feature space. Nevertheless, these advantages have some trade-offs such as the assumption of linearity between the target and the features. Therefore, non-linear problems can't be solved with the model and linearly separable data is not common in real life cases.

K-Nearest Neighbors

K-Nearest Neighbors algorithm is a supervised method applied to solve classification problems. The concept behind this algorithm lies in the assumption that similar observations can be found nearby. Based on this assumption, the KNN algorithm calculates the distance between points placed in an n-dimensions graph, being n the number of independent variables for the observations. This distance can be calculated through many distance calculation methods, although the most commonly applied for this algorithm is the Euclidean distance. The Euclidean distance calculates the squared root of the sum of the squared difference between each pair of values for each dimension, generally covered by the following formula:

$$d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

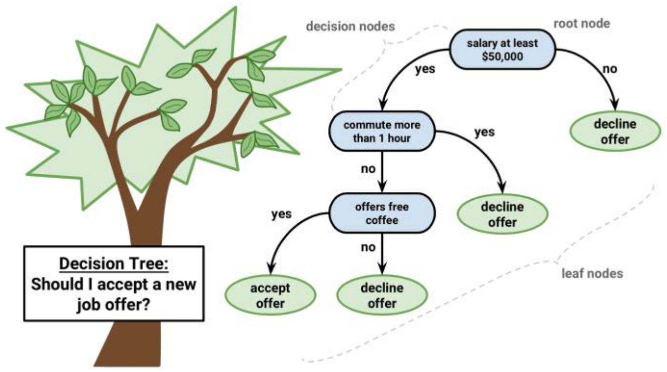
After calculating the distance between the test observation and the previously trained observations, the algorithm detects the number of observations belonging to each class within a K range and assigns to the new observation the class that has more neighbors within that range. To select the optimal number for the K range it is necessary to run the KNN algorithm multiple times with different values of K and chose the value that reduces the number of errors encountered while keeping the model from overfitting. A representation of the KNN model is shown below, where for the new observation given K, the assigned class would be the blue:



By applying this model for a classification problem, we can benefit from its high interpretability and easy implementation of it, since it is a simple model that does not need hyper-parameter tuning or additional assumptions. However, it is necessary to consider that this model could get significantly slower as the number of dimensions (predictors) increases.

Decision Trees

Decision trees are an ensemble of steps applied with the purpose of making a prediction based on historical data. This process is carried out subconsciously every day in all aspects of life, from a business decision up to a person's decision-making process. The basic concept of the decision tree is to start with an attribute (root node) and ask a decisive question about the attribute which could be true or false, and based on the binary answer the root note would be split into two decision nodes. This process is iteratively done with all decision nodes until the model arrives at the leaf nodes of the tree, which can be considered as the expected output of the process. A graphic representation of this process is shown below:



The decision trees algorithm is one of the most popular and well-known methodologies in the area of predictive analytics, and data science. This model can be applied to predict quantitative or qualitative targets; therefore, it is of use for both regression and classification problems. Given the nature of our problem, we will be explaining and applying the decision tree classifier since we have a qualitative target.

A DT classifier maximizes the information gain in a set of observations. The information gain is the measure that represents how much information a certain feature provides us for the given target. Hence, this algorithm makes splits into optimal subtrees that contain as much information and less randomness as possible, selecting the best attributes using the Attribute Selection Measures. After selecting the best attributes, it converts them into decision nodes and separates the dataset into smaller subtrees, repeating recursively until no more attributes remain.

The recursive process of creating decision nodes can lead to a big, complex decision tree that would generate overfitting in new observations. While growing a tree many measures can be used to determine if more decision nodes should be created. The first one is the classification error rate which measure provides the fraction of the training observations in the subset that do not correspond to the most common class. Another measure is the Gini Index, an evaluation of node purity indicating if a node contains predominantly observations from a single class. This index is obtained by computing the total variance across the K classes:

$$Gini(K) = 1 - \sum_{i \in N} P_{i,K}^2$$

Another process applied while growing a decision tree is pruning. Pruning refers to reducing the size of the tree by trimming the initial tree in a way that improves the generalization capability of the decision tree. This process helps to avoid overfitting and it's done by calculating either the classification error rate, the Gini index, or the entropy which measures the randomness contained in the dataset.

To predict the class of a new observation, the data point will go through the tree and associate the observation to the most commonly occurring class of training observations on the decision nodes until reaching the final leaf node which outputs a final class.

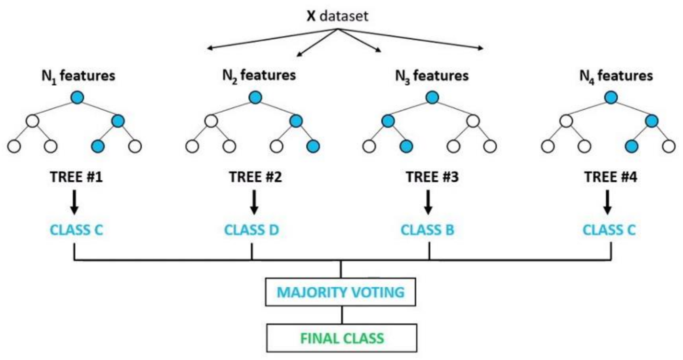
Decision Trees are widely used due to their intuitive implementation, and high interpretability which allows to explain the outcome to both technical users and all stakeholders. Also, decision trees can handle qualitative predictors without needing to create dummy variables. Another benefit of this algorithm is that it can be displayed graphically, which gives even more interpretability to parties that have no expertise in data analytics. On the other hand, trees are very susceptible to changes in the data, causing some instability and large changes in the structure of the decision tree. Also, a trade-off of the decision trees is that these models often take more time to train, and they generally do not have a high predictive accuracy as other classification methods.

Random Forest

Random forest is a commonly used Machine Learning algorithm that uses decision trees as the main factor for the decision-making process. As the decision trees algorithm, random forest can also be used to predict both quantitative and qualitative targets.

Before fully understanding the process of a prediction with an RF algorithm, it is necessary to introduce the concept of Bootstrap Aggregating or bagging. A regular decision tree has the inconvenience of high variance, meaning that with different splits from the same dataset the tree could throw very different results. However, by applying a process with low variance we would obtain similar results with different data sets. Bagging is a procedure used to reduce the variance of a method by taking many training sets from the population, building a separate prediction with each set and averaging the resulting predictions for a regression problem, and taking a majority vote for a classification problem. Those training sets are generated from our original training set by taking repeated samples of the set. This process has demonstrated to increase accuracy but at the expense of interpretability, due to the iteration with multiple training sets and average process.

With the intention of improving this process, the concept of Random Forest is created. This algorithm works on top of the concept of bagging, and it's sustained on the decorrelation of trees. In this case, not only the observations are randomly sampled but also the predictors are randomly chosen. Each time there is a split in a tree, a random sample of the predictors is chosen from the total number of variables. The number of predictors taken at each split must be less than the majority of the predictors available, and it is common to choose approx. the squared root of total predictors as the number of predictors in each split. This process is mainly applied to balance the splits between strong and weak predictors and avoid highly correlation between the trees. Therefore, using a small quantity of features is helpful when having many correlated variables. A graphical representation of the random forest algorithm is shown below:

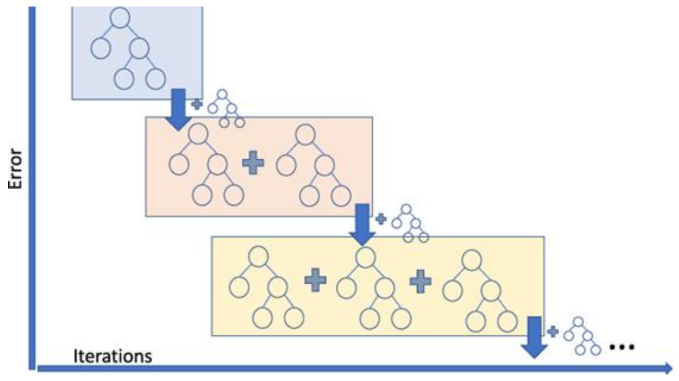


Within the advantages of using Random Forest it can be mentioned that it contributes to reducing the overfitting problem in decision tree based algorithms since it reduces significantly the variance of the model, and since it is based on decision trees, both categorical and continuous values can be used as well as it is not required to normalize the data. Nevertheless, the convenience of this model comes with some disadvantages to weight before using it. Some of these disadvantages are that the model requires much computational power since numerous decision trees are being modelled to determine a single output. At the same time, this model can be challenging to interpret compared to a simple decision tree.

Gradient Boosting

Gradient Boosting is one of the most powerful Machine Learning algorithms present in the industry. It is known for its prediction speed and high accuracy, specifically with large and complex datasets. The algorithm roots are found in boosting methods. Boosting is a methodology that uses a sequential ensemble of weak models seeking to improve the prediction power by compensating the weaknesses among one another. Boosting builds an initial decision and then assigns equal weight to all data points. In a following process weight is increased for all points that are misclassified and lowers the weight to all points correctly classified. This process is carried out iteratively until a strong model is obtained.

Gradient Boosting is an iterative functional gradient algorithm, therefore it is an algorithm that minimizes a loss function repetitively by choosing a function that points towards negative gradient. The model has three main components which are the loss function, the weak learner, and the additive model. The loss function estimates how good the model is at making predictions with the given dataset. The weak learner is used to classify our data poorly, with high error rate. Last, the additive model, is an iterative approach of adding the weak learners one step at a time, being closer to our final model after each iteration. A learning rate parameter is also commonly used, representing the impact of each decision tree in the model fitting. A graphical description of the model is shown below:



Gradient Boosting is a widely applied algorithm thanks to its benefits for both classification and regression problems, among which we can mention a high prediction accuracy, wide flexibility since the loss function can be modified as well as many other hyperparameters, and since it is mainly based on decision trees it often works great with categorical and numerical values as well as the handling for missing data. However, the Gradient Boosting model could tend to overfit since it continuously improves to minimize errors, and as the Random Forest this method is computationally expensive since it is working with a large amount of decision trees and challenging while interpreting its results.

Benchmark Experiment

Train – Test Split

To start with the benchmark experiment a train-test split was necessary. This process is done to guarantee that the model is evaluated in the data that it was fitted as well as unseen data. For this project, we split the data in 70/30 percent for train/test set, respectively.

Base-table Pipeline

In order to prepare the database to be used in all models a data transformation was done. First of all, we checked for null values in all variables. Then, for the variables that contained null observations we created a column to identify which of the observations had null values and then we filled the null values in the original categories, for both train and test sets with the aggregation of the train set, as follows:

- Age: Filled with median
- Job: Filled with category "unknown"
- Marital: Filled with category "unknown"
- Education: Filled with category "unknown"
- Default: Filled with category "unknown"
- Housing: Filled with category "unknown"
- Loan: Filled with category "unknown"

- Contact: Filled with category “missing”
- Day of week: Filled with category “missing”
- Month: Filled with mode
- Campaign: Filled with median
- Pdays: Filled with median. Changed 999 to -1
- Previous: Filled with median
- Poutcome: Filled with “unknown”
- Emp.var.rate: Filled with median
- Cons.price.idx: Filled with mean
- Cons.conf.idx: Filled with mean
- Euribor3m: Filled with median
- Nr.employed: Filled with mean

Then for the categorical variables, a One Hot Encoding was processed and removed the first column of the categories.

Since we have undersampled target=1 we created a process of oversampling by implementing the Imbalance library and increased the target = 1 by a ratio of 0.8 using Random Walk Algorithm.

Experimental Setup

Variable selection – Dimensional Reduction

For all models the variable selection process was done by variable importance. After fitting the models once, the variable importance was calculated, either by the Fisher Scoring or by the variable importance to reduce the selected variables of the model.

Models Implementation

Logistic Regression

The logistic regression model was implemented using the glm function of the stats library:

final_lr <- glm(subscribe ~emp.var.rate +job_student+job_retired + marital_unknown + contact_telephone + month_mar + month_may + month_jul + month_nov+ poutcome_nonexistent +poutcome_success +day_of_week_mon +education_university.degree + nr.employed + null_month, data= train_base, family = "binomial")

The results obtained with the parameters below were an Accuracy of 90%, however the AUC was 0.55.

Cross-validation methods

To apply cross-validation in the Logistic Regression function we used the makeResampleDesc function from the mlr library. The result for logistic regression after Cross Validation was an aggregated AUC of 0.78 after 10 iterations.

K-Nearest Neighbors

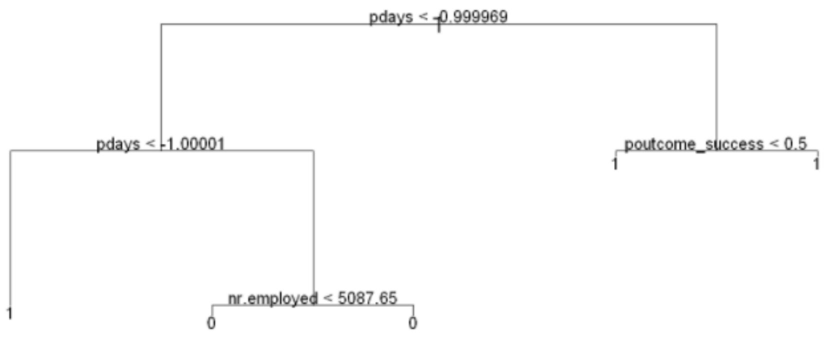
For the KNN algorithm we first used the trainControl method from the caret library to obtain the optimal k value to use in the KNN algorithm. This function suggested to use a K = 23. We then applied the knn function from the class library with the k suggested and created a confusion matrix to display the results. With this setup we obtain an accuracy of approx. 90% and an AUC of 0.75

Cross-validation methods

With the cross-validation algorithm, we obtain an AUC of 0.93. This result is significantly different from the normal train/test split because of the nature of KNN algorithm. Since the KNN tries to place new observations according to the training observation, this prediction's accuracy will fluctuate depending on the training set.

Decision Trees

For decision trees, we first ran a simple decision tree which resulted in the following tree:



This tree's evaluation suggested that for this model accuracy of approx. 90% was achieved and an AUC of 0.60. We tried implementing more sophisticated decision trees which caused an exponential increase in the variable's selection, therefore we decided to have a trade-off between interpretability against a small difference of accuracy.

Random Forest

For the random forest model, we first applied a simple random forest with all variables and 100 trees. That first model gave us a hierarchy on variable importance which allowed us to filter the variables for a next final model. For the final model we only used the variables that contained a node purity higher than 100 and set the number of trees to 200. The results obtained were an accuracy of approximately 90% and an AUC of 0.61

Gradient Boosting

The gradient boosting model also gives us variable importance. Therefore, we implemented a simple model without hyperparameter tuning, and then, after the first iteration, we fitted the model for a selected number of variables, 200 trees, and a learning rate of 0.01. We noticed that by selecting only a few of the variables the AUC of our model decreased. Hence, we chose the original model, with which we obtained an AUC of approx. 0.61

Conclusions & Recommendations

Throughout this process it was perceived how models behave differently towards the same dataset, and we could measure their performance based on different metrics. Hypertuning parameter is one of the advantages that some of these models offer and makes a difference in terms of prediction accuracy. By applying these algorithms to the bank dataset, the theory explained in the first part of this report was closely seen and put into practice.

For further studies of this dataset, we recommend to create different clusters for some categorical and numerical variables before applying the models. As well as enhancing the oversampling process in order to obtain a more balanced dataset that could benefit in the prediction process.

References

An Introduction to Statistical Learning with Applications in R. G. James, D. Witten, T. Hastie, R. Tibshirani

https://blog.finxter.com/wp-content/uploads/2019/03/grafik-17.png

https://towardsdatascience.com/probability-concepts-explained-maximum-likelihood-estimation-c7b4342fdbb1

https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/

https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

https://machinelearninghd.com/wp-content/uploads/2021/02/knn.png

https://www.aionlinecourse.com/tutorial/machine-learning/decision-tree-classification

https://medium.com/analytics-vidhya/decision-trees-for-dummies-a8e3c00c5e2e

https://www.aionlinecourse.com/uploads/tutorials/2019/07/18_2_decision_tree.png

https://www.analyticsvidhya.com/blog/2021/04/beginners-guide-to-decision-tree-classification-using-python/

https://www-users.cse.umn.edu/~kumar001/dmbook/ch4.pdf

https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a

https://www.mygreatlearning.com/blog/random-forest-algorithm/#AdvantagesandDisadvantagesofRandomForest

https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

https://ichi.pro/assets/images/max/724/1*85QHtH-49U7ozPpmA5cAaw.png

https://blog.paperspace.com/gradient-boosting-for-classification/