Statistical & Machine Learning

Individual Assignment Report

Submitted by: **Omar Abdelgelil (**omar.abdelgelil@ieseg.fr**)**

Prof: Minh Phan PhD

Introduction:

The main purpose of this report is to setup the benchmark experiment to compare 5 selected machine learning algorithms. For each algorithm the setup and the result will be explained in terms of cross-validation method (holdout, k-fold CV, etc.), evaluation metric (AUC, Accuracy, etc.), hyperparameter tuning, variable selection, resampling method (over-sampling, undersampling, etc.), etc.).

The dataset used is the same for the Kaggle competition with the aim to predict whether a customer will subscribe to a telemarketing campaign or not.

Models Selected:

This is a classification problem. But what is a classification problem ?

A classification problem is when independent variables are continuous in nature and dependent variable is in categorical form. Here is to categorize customers as they are going to subscribe or not. All these problem's answers are in categorical form i.e. Yes or No. and that is why they are two class classification problems.

Therefore the following ML models were used:

1. Logistic Regression
2. Random Forrest
3. XGBoost
4. Support Vector Machine
5. Linear Discriminant Analysis

Logistic Regression:

Logistic Regression is one of the basic and popular algorithm to solve a classification problem. It's underlying technique is quite the same as Linear Regression.

The problem with linear regression is that in classification problems where we need to predict values as 1 or 0, linear regression can't predict that due it's linear nature by using a line to predict values.

The term "Logistic" comes from the **Logit function** that is used in this method of classification.

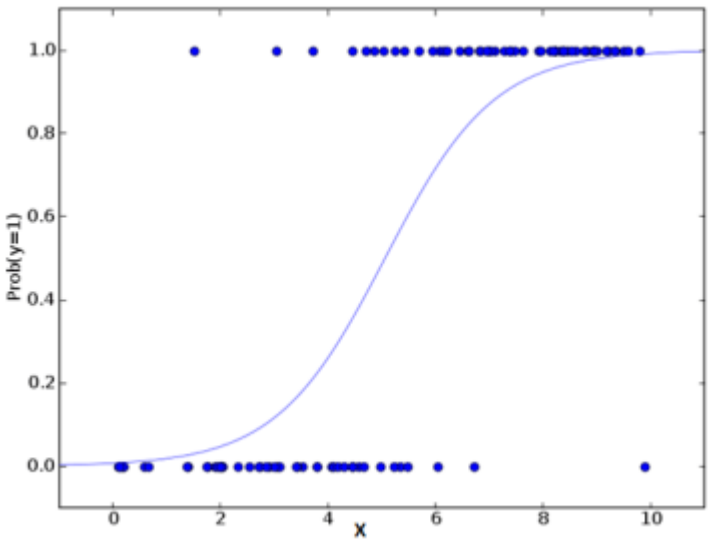
So basically what happens here is that the Logistic Regression measures the relationship between the dependent variable and the one or more independent variables (our features), by estimating probabilities using it's underlying logistic function.

These probabilities must then be transformed into binary values in order to actually make a prediction. This is the task of the logistic function, also called the sigmoid function. The Sigmoid-Function is an S-shaped curve that can take any real-valued number and map it into a value between the range of 0 and 1, but never exactly at those limits. This values between 0 and 1 will then be transformed into either 0 or 1 using a threshold classifier.

The logistic function also called Sigmoid function can be defined as:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

And if we plot it, the graph will be *S* curve,



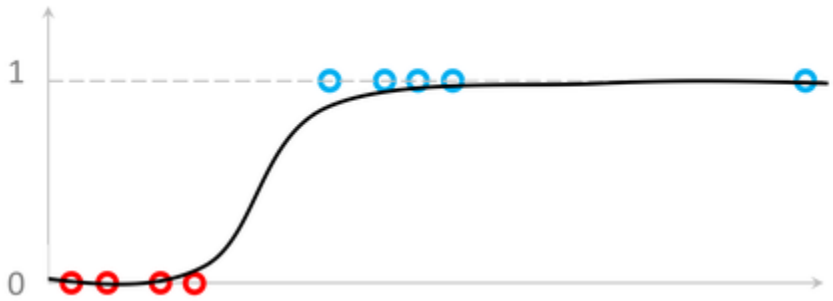
Let's consider *t* as linear function in a univariate regression model.

$$t = \beta_0 + \beta_1 x$$

So the Logistic Equation will become

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

Now, when logistic regression model come across an outlier, it will take care of it.



Important note regarding Logistic regression is that it does work better when you remove attributes that are unrelated to the output variable as well as attributes that are very similar (correlated) to each other. Therefore Feature Engineering plays an important role in regards to the performance of Logistic Regression.

Advantages of LR:

- Efficient, does not require too many computational resources
- Highly interpretable
- Doesn't require input features to be scaled as well as any tuning
- Outputs well calibrated predicted probabilities

Disadvantages of LR:

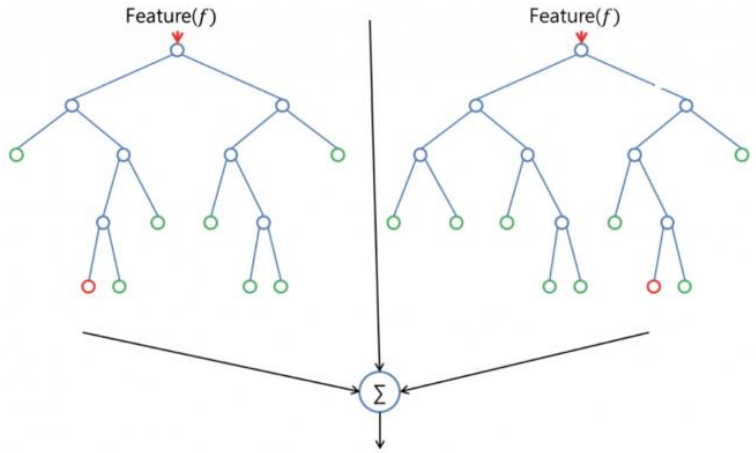
- Can't solve non-linear problems with logistic regression
- logistic regression is not a useful tool unless you have already identified all the important independent variables
- Can be easily overfitted

Random Forrest:

Random Forrest is a tree based method. Basically it builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).



So how a RF is trained?

Random Forests are trained via the bagging method. Bagging or Bootstrap Aggregating, consists of randomly sampling subsets of the training data, fitting a model to these smaller data sets, and aggregating the predictions. This method allows several instances to be used repeatedly for the training stage given that we are sampling with replacement. Tree bagging consists of sampling subsets of the training set, fitting a Decision Tree to each, and aggregating their result.

The Random Forest method introduces more randomness and diversity by applying the bagging method to the feature space. That is, instead of searching greedily for the best predictors to create branches, it randomly samples elements of the predictor space, thus adding more

diversity and reducing the variance of the trees at the cost of equal or higher bias. This process is also known as “feature bagging” and it is this powerful method what leads to a more robust model.

In the Random Forests algorithm, each new data point goes through the same process, but now it visits all the different trees in the ensemble, which are were grown using random samples of both training data and features. It uses the mode or most frequent class predicted by the individual trees (also known as a majority vote).

Advantages of RF:

- the default hyperparameters it uses often produce a good prediction results
- hard to overfit
- The same random forest algorithm can be used for both classification and regression task.

Disadvantages of RF:

- large number of trees can make the algorithm too slow and ineffective for real-time predictions.

In general, Random forest is a great algorithm to train early in the model development process, to see how it performs.

The algorithm is also a great choice for anyone who needs to develop a model quickly. On top of that, it provides a pretty good indicator of the importance it assigns to your features.

XGBoost:

XGBoost stands for Extreme Gradient Boosting; it is a specific implementation of the Gradient Boosting method which uses more accurate approximations to find the best tree model.

The beauty of this powerful algorithm lies in its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage.

XGBoost is an ensemble learning method. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The resultant is a single model which gives the aggregated output from several models.

The models that form the ensemble, also known as base learners, could be either from the same learning algorithm or different learning algorithms. Bagging and boosting are two widely used ensemble learners. Though these two techniques can be used with several statistical models, the most predominant usage has been with decision trees.

Let's have a deeper look at Boosting:

In boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals.

The base learners in boosting are weak learners in which the bias is high, and the predictive power is just a tad better than random guessing. Each of these weak learners contributes some vital information for prediction, enabling the boosting technique to produce a strong learner by effectively combining these weak learners. The final strong learner brings down both the bias and the variance.

In contrast to bagging techniques like Random Forest, in which trees are grown to their maximum extent, boosting makes use of trees with fewer splits. Such small trees, which are not very deep, are highly interpretable. Parameters like the number of trees or iterations, the rate at which the gradient boosting learns, and the depth of the tree, could be optimally selected through validation techniques like k-fold cross validation. Having a large number of trees might lead to overfitting. So, it is necessary to carefully choose the stopping criteria for boosting.

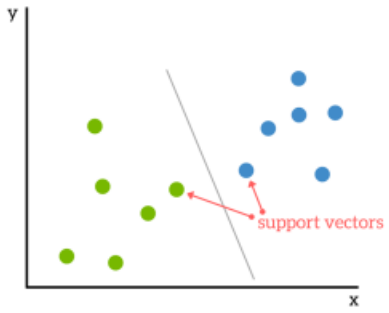
Unique features of XGBoost

- **Regularization**: XGBoost has an option to penalize complex models through both L1 and L2 regularization. Regularization helps in preventing overfitting
- **Handling sparse data**: Missing values or data processing steps like one-hot encoding make data sparse. XGBoost incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data
- **Weighted quantile sketch**: XGBoost has a distributed weighted quantile sketch algorithm to effectively handle weighted data

Support Vector Machine:

A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be employed for both classification and regression purposes.

SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes, as shown in the image below.

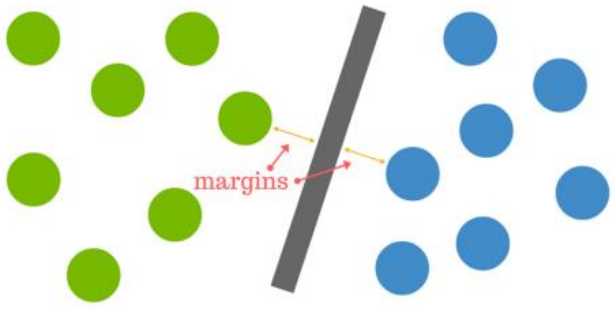


Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set. As a simple example, for a classification task with only two features (like the image above), you can think of a hyperplane as a line that linearly separates and classifies a set of data.

Intuitively, the further from the hyperplane our data points lie, the more confident we are that they have been correctly classified. We therefore want our data points to be as far away from the hyperplane as possible, while still being on the correct side of it.

So when new testing data is added, whatever side of the hyperplane it lands will decide the class that we assign to it.

The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.



Advantages of SVM:

- High accuracy
- Works well on smaller cleaner datasets
- High efficiency because it uses a subset of training points

Disadvantages of SVM:

- Isn't suited to larger datasets as the training time with SVMs can be high
- Less effective on noisier datasets with overlapping classes

Linear Discriminant Analysis:

LDA model consists of statistical properties of your data, calculated for each class. For a single input variable (x) this is the mean and the variance of the variable for each class. For multiple variables, this is the same properties calculated over the multivariate Gaussian, namely the means and the covariance matrix.

These statistical properties are estimated from your data and plug into the LDA equation to make predictions.

Learning LDA Models

LDA makes some simplifying assumptions about your data:

- That your data is Gaussian, that each variable is shaped like a bell curve when plotted.
- That each attribute has the same variance, that values of each variable vary around the mean by the same amount on average.

With these assumptions, the LDA model estimates the mean and variance from your data for each class.

The mean (mu) value of each input (x) for each class (k) can be estimated in the normal way by dividing the sum of values by the total number of values.

$$muk = 1/nk * sum(x)$$

Where muk is the mean value of x for the class k, nk is the number of instances with class k. The variance is calculated across all classes as the average squared difference of each value from the mean.

$$sigma^2 = 1 / (n-K) * sum((x – mu)^2)$$

Where sigma^2 is the variance across all inputs (x), n is the number of instances, K is the number of classes and mu is the mean for input x.

Making Predictions with LDA

LDA makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made.

The model uses Bayes Theorem to estimate the probabilities. Briefly Bayes' Theorem can be used to estimate the probability of the output class (k) given the input (x) using the probability of each class and the probability of the data belonging to each class:

$$P(Y=x|X=x) = (PIk * fk(x)) / sum(PIl * fl(x))$$

Where PIk refers to the base probability of each class (k) observed in your training data (e.g. 0.5 for a 50-50 split in a two class problem). In Bayes' Theorem this is called the prior probability.

$$PIk = nk/n$$

The f(x) above is the estimated probability of x belonging to the class. A Gaussian distribution function is used for f(x). Plugging the Gaussian into the above equation and simplifying we end up with the equation below. This is called a discriminate function and the class is calculated as having the largest value will be the output classification (y):

$$Dk(x) = x * (muk/siga^2) – (muk^2/(2*sigma^2)) + ln(PIk)$$

Dk(x) is the discriminate function for class k given input x, the muk, sigma^2 and PIk are all estimated from your data.

After explaining the theory behind the five chosen models, the following table will summarize the results in terms of the accuracy metric to evaluate the performance of each Model.

Results:

Model	AUC
Logistic Regression	0.812
Random Forrest	0.7368
XGBoost	0.7981
Support Vector Machine	0.57
LDA	0.8097

From observing the results, we see that the LR performed the best however its score was still very close to XGBoost model. The later could give a better score by further modifying the parameters.

The Coding structure:

1-Pre Processing:

Basic process, nothing special

2-Feature selection:

The DV was “subscribe” column.

63 columns were initially predictors. This is a big number and some of them might be highly correlated. Therefore a feature selection method was used: Boruta technique. By doing this, we obtained 30 predictors that were categorized as “confirmed” which means that the Boruta method think that they are relevant.

3-Modelling:

The model was built with the MLR package. To analyze performance the K fold cross validation technique with 10 iterations was computed.

4-Evalaution Metric:

The AUC metric was used as a comparison metric. The AUC score is a value between 0 and 1. The closer the score is to 1 the better is the Model.

5- MLR :

Basically the MLR package retrieve the model that have the best AUC score over the cross validation and keep the hyper-parameter obtained from the model that perform the best. Finally the best model with its parameters is used to compute predictions on test set.