Statistical and Machine Learning
Individual Project

IÉSEG School of Management – March 31st, 2022

AZAR Nour

Introduction

In this report, I will explain a set of machine learning models and set up a benchmark experiment for those models. The report consists of two parts. For the first part, five different machine learning models will be explained. For the second part, a benchmark experiment will be set up in R using the credit card default dataset, and the results of each model will be explained and compared with the rest. The five models chosen for this project are: logistic regression, linear discriminant analysis, decision trees, random forests, and KNN.

Task 1: Explaining the models:

Logistic regression

The logistic regression model is a classification technique that predicts the probability that a certain outcome belongs to a certain class, using the following logistic function that only returns values between 0 and 1:

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

After doing some manipulation and after taking the logarithm of the equation, we arrive at the following equation:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

We notice that the right-hand side looks like a linear regression equation, and while comparing it to the linear regression model, we see that for the logistic regression an increase in one unit of X changes the logit (the left-hand side) by β_1 whereas in linear regression the change in one unit of X gives the average change in Y. However, if we look at the change of value of Y following a one unit increase in X in Logistic regression, we see that the change depends on the value of X.

β_0 and β_1 are both unknown coefficients, and in order to estimate them we maximize the following likelihood function such as plugging the coefficients into the logistic model would yield an outcome as close as possible to the actual outcome:

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}} (1 - p(x_{i'}))$$

Assumption of the model:

- The target variable is binary, it only takes on two possible outcomes.
- The observations are independent of each other: they should not be related to each other in anyway.
- Explanatory variables are not highly correlated.
- There is no extreme outliers.

- There is a linear relationship between explanatory variables and the logit of the target variable (as discussed before).
- The sample is large enough to train.

LDA (Linear Discriminant Analysis):

Unlike logistic regression which is used in case of only 2 response classes, LDA is used in the case of more than two response classes.

π_K represents the prior probability that a randomly chosen observation belongs to class k of the target variable Y, and $f_k(x) = \Pr(X = x|Y = k)$ represents the density function of X for an observation that belongs to k class. If the density function is high, it means that there is a high probability that an observation in the k class has X=x.

We plug in estimates of $f_k(x)$ and π_K in Bayes' theorem:

$$\Pr(X = x|Y = k) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

$\Pr(X = x|Y = k)$ is the posterior probability that observation X=x belongs to class k given the predictor value of that observation.

The bayes classifier involves assigning an observation X=x to the class for which the posterior probability is larger.

Calculation of density function in case of one predictor:

Calculation of density function is complicated. If we assume that there is only one predictor and assuming that the distribution of this predictor is normal then:

$$f_k(x) = \frac{1}{\sqrt{2\pi\sigma_k}} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

Where μ_k is the mean for class k

σ_k^2 is the variance for class k

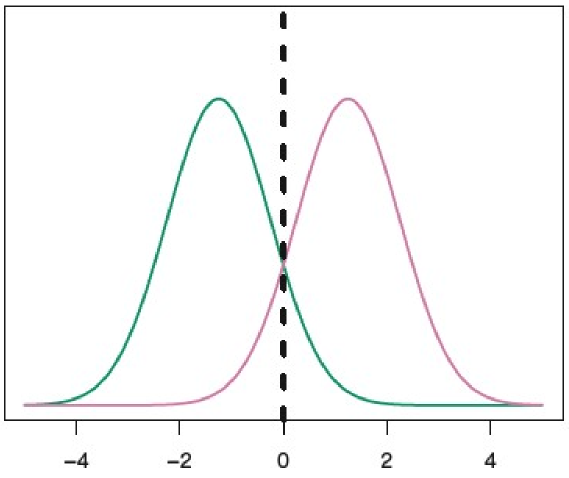
- By plugging the previous equation to the bayes' theorem we get:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi\sigma}} exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right)}$$

- As mentioned before, the bayes classifier involves assigning an observation to the class for which the posterior probability is larger, which is equivalent to the log of the bayes' theorem which gives us the following function:

$$\delta_k(x) = x.\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

- Example:



In this example we assume two classes (k=2) and one variable with normal distribution. The bayes decision boundary is shown in the dashed vertical line: if x<0 then the bayes classifier assigns the observation to class 1 and if x>0 it assigns it to class 2.

Since the Bayes classifier takes the largest probability, it is subject to a decision boundary where:

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2}$$

In our discussion, we assumed that the distribution is normal, however even by assuming that we need to estimate the parameters of the previous formula and that is done by the LDA model which calculates the Bayes classifier by plugging the mean and the variance using the formulas below:

$$\begin{aligned} \hat{\mu}_k &= \frac{1}{n_k} \sum_{i:y_i=k} x_i \\ \hat{\sigma}^2 &= \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 \end{aligned}$$

Where n_k is the total number of training observations in the kth class.

<u>Calculation of density function in case of more than one predictor:</u>

X is assumed to be drawn from a multivariate distribution and , the density function is defined as follows:

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

If we do the same steps as the steps done previously in case of one predictor, we get to the following formula where we assign an observation to the class for which the value is largest:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

Assumptions of LDA:

- Each variable is normally distributed
- Variances among group variables are the same across levels of predictors
- The observations are independent of each other.

<u>Decision Tree:</u>

Regression trees can be applied both to regression and classification models, they are easy to interpret and have nice graphical representation.

1- Regression trees

Trees are split into *terminal nodes* or *leaves* (at the bottom of the tree), we refer to the points along the tree where the predictor space is split as internal nodes and to the segments of the trees that connect the nodes as branches.

There are two steps for building a regression tree:

1) We divide the predictor space into distinct and non-overlapping regions.
2) For every observation in a certain region the same value is predicted, which is the mean value for that region.

In the first step, the goal is to find the regions that minimize the RSS given by:

$$\sum_{j=1}^{R} \sum \left(y_i - \hat{y}_{R_j}\right)^2$$

Where $\hat{y}$ is the mean value of the observations present in region j.

Using the top-down approach, we start splitting from the top of the tree with the best split by minimizing the RSS. The top-down approach is also called the *greedy* approach because it only takes the best split of as region in a particular step rather than picking a split that will give better splits in the next steps. More specifically, for any region g and for any cut-off value on the internal node we seek the value of j and s that minimizes the equation:

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

We keep doing this process on each step, however unlike the first step where we split the first whole region into two, on the next steps we split the regions that were already split in the previous steps. The process continues until the stopping criterion is met, for example until 10 observations per region is met.

Once all the regions are created, we predict the target variable for a given observation by taking the mean of the region in which it falls.

The process described above might overfit the data because of its high complexity, therefore in order to reduce the complexity of the decision tree we can apply the *Tree Pruning* technique, which reduces the size of the tree by removing some sections only so long as the decrease of the RSS is higher than a certain threshold that we set. This technique is called pre-pruning since it involves building the tree before it had completed. However, this technique is short-sighted since there is a possibility that a small decrease in the RSS in a certain step might be followed by a very large decrease in RSS in the following step but we can't anticipate that in the pre-pruning technique. Therefore, there is another technique called post-pruning that consists of growing a very large tree and then pruning it back. *Cost Complexity pruning*, also called *weakest link pruning* is a technique that allows us to select a small set of subtrees from which we select the subtree that leads to the lowest error rate. In this technique, we consider a series of trees indexed by a nonnegative tuning parameter α where the following equation is as small as possible:

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Where |T| indicates the number of terminal nodes of the tree T, R_m is the subset of the predictor space, corresponding to the mth terminal node and $\hat{y}_{R_m}$ is the predicted response associated with R_m. When α=0 then the subtree is T_0 which is the large tree, and as we increase α the branches get pruned from the tree.

2- Classification Trees

Classification tree is similar to Regression tree but the difference is that the response is qualitative instead of quantitative, and instead of taking the mean of the region where a certain observation falls like in the regression tree, we take the most occurring class in that region. Also, in classification tree the RSS is not used for splitting criterion but instead other indexes are used, like the classification error rate, which is the fraction of observations that do not actually belong to the most common class:

$$E = 1 - \max(\hat{p}_{mk})$$

Where $\hat{p}_{mk}$ represents the proportion of the observations in the *m*th region that are from the *k*th class.

Apart from the classification error rate, we also use *Gini index* which is a measure of total variance across K classes:

$$G = \sum_{k=1}^{K} \hat{p}_{m_k} (1 - \hat{p}_{m_k})$$

A small value of *Gini index* indicates that a node contains predominantly observations from a single class.

Another measure of error is *entropy*, which like Gini index, takes a small value of the node contains predominantly observations from a single class:

$$D = -\sum_{k=1}^{K} \hat{p}_{m_k} \log \hat{p}_{m_k}$$

The *Gini index* and *entropy,* which are a measure of purity, are usually preferred over the classification error rate because they are more sensitive to split purity, however if we want to measure the final accuracy rate of the final pruned model, then classification error rate might be better to use.

Assumptions of decision tree model:

We don't need to make assumptions in this model since it is not a probabilistic model and consists only of binary splits. However, of course, it is assumed that the sampling is representative.

Random Forest

If we run the decision tree many times, we will get different output each time, which means that the decision tree model suffers from high variance. Now we will discuss the Random Forest model that solves this issue of high variance. In order to reduce the variance, Random Forest takes several sample from the training dataset, called bootstrapped training datasets, and runs the model on these samples, and then, for the quantitative response, takes the average of the obtained predictions as the prediction for the observations of the training set as a whole, and for the qualitative response it takes the most commonly occurred class in the predictions of the bootstrapped training datasets. However, if the training dataset has a predictor that is much stronger than the other predictors, most of the trees will likely use this predictor in the first split, which will produce similar subtrees, therefore averaging the predictions of these subtrees will not lead to lower variance. Random Forests solves this issue by forcing the model to choose a sample of predictors in each split instead of all predictors, therefore there is a probability that (p-m)/p of the splits will not consider the strong predictor (p indicates all predictors and m indicates a sample of the predictors).

Assumptions of Random Forests model:

Like the Decision Tree, the Random Forests the only assumption is that the sampling is representative.

K-Nearest Neighbor (KNN)

KNN assumes that similar things are close to each other. In other words, KNN assumes that observations that are close to each other in a graph are similar.

The KNN can be explained on the basis of the below steps:

1- Select the number k of the neighbors.
2- Calculate the Euclidean distance of k number of neighbours.
3- Take the K nearest neighbors as per the calculated Euclidean distance.
   Example: Euclidean distance between point A(X_1, Y_1) and point B(X_2, Y_2) is:
   $$\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$
4- Among these k neighbors, count the number of the data points from the training set in each category.

5- Assign the test observations data points to that category for which the number of the neighbor is maximum.

The value of K is random and there is no particular way in determining it, however in order to get the best accuracy we need to try different K values and determine the best one. It is worth noting that taking a small value of K will lead to noisy data and will cause outliers,, and taking a high value of K will yield better accuracy.

Just like decision tree and random forests, KNN calculates the mean of the selected neighbor observations in case of regression, and the class with the most training observations in case of classification.

Assumptions of KNN:

- Clusters are spherical.
- Clusters are of similar sizes.

Task 2: Benchmarking

Steps we did before running the models:

1- Error Correction:
   - First, in order to avoid information leakage from unseen data, we split the data into train and test before starting the preprocessing.
   - I imputed the missing values of the categorical columns with the mode.
   - I imputed the missing values of continuous values with the mean.
   - I created new variables to track the missing values of all the columns.
   - I removed 302 outliers.
2- Feature engineering and Value transformation:
   - I changed the categories of the columns 'PAY_0', 'PAY_2','PAY_3', 'PAY_4', 'PAY_5', 'PAY_6'.
   - I created a new column ‘age_range’ that contains the ranges of the ages.
   - I scaled the numerical variables.
3- Variable Selection:

- I used fisher score for variable selection and ended up selecting – variables.

Interpretation of the models:

	Train accuracy	Test accuracy	Train AUC	Test AUC
Logistic Regression	0.82	0.81	0.76	0.75
LDA	0.81	0.81	0.75	0.74
Decision Tree	0.82	0.81	0.69	0.68
Random Forest	0.99	0.81	0.99	0.75
KNN	0.84	0.79		

We notice that the accuracy scores are all approximately equal across all models, but AUCs differ. The highest AUC we got are from both Decision Tree and Random Forest, but since Random Forest is overfitting, I choose the best model as AUC because it has the best combination of AUC and accuracy score.

<u>References:</u>

Book: *An Introduction to statistical learning – Gareth James*

Website*:* *https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning*