UNIVERSITÀ CATTOLICA DEL SACRO CUORE

MILANO

Facoltà di Economia

Corso di Laurea in: Economia delle Imprese e dei Mercati

Ensemble machine learning methods for bankruptcy prediction

Relatore: Prof. Gabriele Cantaluppi

Tesi di Laurea di:
Davide Alberto Lupis

Matricola: 4701017

1 Abstract

"There is a realistic probability that we could indeed see a 'pandemic' of bankruptcy filings in the near future. The pandemic analogy is particularly apt, in that if the number of new filings is sufficiently high, the bankruptcy courts, like hospitals treating COVID-19 patients, could be overwhelmed." [1].(Stuart C. Gilson ,2020).

Bankruptcy prediction is the task of predicting bankruptcy and various measures of financial distress of firms. It is a vast area of finance and accounting research. In this work, the predictive strength of data and machine learning techniques will be analyzed with the final purpose to discuss a novel approach to bankruptcy prediction. "Extreme Gradient Boosting for learning an ensemble of decision trees." The initial part will be a summary of the most important concepts taken by different sources (books papers and git-hub repositories). The second part will present some statistical theory and the usage of Machine Learning with a deep dive on Tree-Based Models and Ensemble, especially on XGB. The third part is a presentation of the specific use-case with a personal comment related to the most innovative approaches. It will contain the complete pipeline, starting by methodology and concluding with a Deployed model on IBM Watson Studio (A cloud platform for Data Science).

Table of Contents

2 Introduction

The last financial crisis has led to a higher number of bankruptcies, which is a prime example of how environment can influence financial health. These financial failures can have a large impact on investors, employees and even society.The need of successfully predict a corporation's financial health is evident. Corporate failures and bankruptcy are a result of financial and/or economic distress. Firms in financial distress experience a shortfall in cash flow needed to meet their debt obligations.

The Business does not necessarily have fundamental problems and products are often attractive. In contrast ,firms in economic distress often have unsustainable business models.

Many factors contribute to the high number of corporate failures. The most common are listed below:

- Poor operating performance and high financial leverage.
- Lack of technological innovation.
- Liquidity and funding shock.
- Relatively high new business formation rates in certain periods.
- Deregulation of key industries.
- Unexpected liabilities.

[2].

Corporations do not fail instantaneously, objective measures and rigorous analysis of qualitative and quantitative data can help identify a company's financial risk. Gathering data about a corporation has become less difficult with recent advancements in communication and information technology. However, there are different issues associated with the bankruptcy prediction:

- The indicators describing the firm's condition are proposed by domain experts and measuring financial ratios is not always equivalent to observing "real market characteristics".
- The hacking of Financial Ratio by the Management
- The need of many variables often causes instability in traditional rule-based models.
- The historical observations used to train a model are usually influenced by imbalanced data phenomenon.

Every method presented is a Binary Classification Problem, which involves an input vector of features X in R^n space where n is the number of features. The possible outcomes Y are always:

- 1 for Bankruptcy
- 0 for Non-Bankruptcy

3 Previous Work in Bankruptcy Prediction

3.1 Altman Z score

In 1966 one of the first models for predicting bankruptcy is created through the use of financial ratios by William H. Beaver then other authors followed this quantitative path. Edward I. Altman in the 1968 extended the work with the Z score. The Altman Z-score is the output of a credit-strength test that gauges a publicly-traded manufacturing company's likelihood of bankruptcy. The model is the result of a Multiple Discriminant Analysis or MDA.

Assumptions of MDA:

- The variables $X_1, X_2, \ldots, X_k$ are independent of each other
- Groups are mutually exclusive and the group sizes are not grossly different.
- Errors (residuals) are randomly distributed
- The variance-covariance structure of the independent variables are similar within each group of the dependent variable

It is used primarily to classify and/or make predictions in problems where the dependent variable appears in categorical form, for example, male or female, bankrupt or non-bankrupt.Therefore, the first step is to establish explicit group classifications.

The number of original groups can be two or more. Some analysts refer to discriminant analysis as "multiple" only when the number of groups exceeds two. After the groups are established, data are collected in the groups; MDA in its most simple form attempts to derive a linear combination of these characteristics which "best" discriminates between the groups. If a particular object, for instance, a corporation, has characteristics (financial ratios) which can be quantified for all of the companies in the analysis, the MDA determines a set of discriminant coefficients. The MDA technique has the advantage of considering an entire profile of characteristics common to the relevant firms, as well as the interaction of these properties. A univariate study, on the other hand, can only consider the measurements used for group assignments one at a time. Another advantage of MDA is the reduction of the analyst's space dimensionally.This analysis is concerned with two groups, consisting of bankrupt and nonbankrupt firms.

According to Altman, bankruptcy could be explained successfully through the use of five financial ratios:

- Working capital/total assets,
- Retained earnings/total assets,
- Earnings before interest and taxes/total assets,
- Market Value of Equity/Book Value of Total Liabilities
- Sales/total assets.

Working capital/total assets (WC/TA) The working capital/total assets ratio, frequently found in studies of corporate problems, is a measure of the net liquid assets of the firm relative to the total capitalization. Working capital is defined as the difference between current assets and current liabilities. Liquidity and size characteristics are explicitly considered. Ordinarily, a firm experiencing consistent operating losses will have shrinking current assets in relation to total assets.

Retained earnings/total assets (RE/TA) Retained earnings is the account which reports the total amount of reinvested earnings and/or losses of a firm over its entire life. The account is also referred to as earned surplus. It should be noted that the retained earnings account is subject to "manipulation" via corporate quasi-reorganizations and stock dividend declarations.
Firm will probably show a low RE/TA ratio because it has not had time to build up its cumulative profits. Therefore, it may be argued that the young firm is somewhat discriminated against in this analysis, and its chance of being classified as bankrupt is relatively higher than that of another older firm, ceteris paribus. But, this is precisely the situation in the real world. The incidence of failure is much higher in a firm's earlier years.

Earnings before interest and taxes/total assets (EBIT/TA) This ratio is a measure of the true productivity of the firm's assets, independent of any tax or leverage factors. Since a firm's ultimate existence is based on the earning power of its assets, this ratio appears to be particularly

appropriate for studies dealing with corporate failure. Furthermore, insolvency in a bankrupt sense occurs when the total liabilities exceed a fair valuation of the firm's assets with value determined by the earning power of the assets.

Market Value of Equity/Book Value of Total Liabilities (MVE/TL) Equity is measured by the combined market value of all shares of stock, preferred and common, while liabilities include both current and long term. The measure shows how much the firm's assets can decline in value (measured by market value of equity plus debt) before the liabilities exceed the assets and the firm becomes insolvent

Sales/Total assets (S/TA) The capital-turnover ratio is a standard financial ratio illustrating the sales generating ability of the firm's assets. It is one measure of management's capacity in dealing with competitive conditions. This final ratio is quite important because it is the least significant ratio on an individual basis. In fact, based on the univariate statistical significance test, it would not have appeared at all. However, because of its unique relationship to other variables in the model, the sales/total assets ratio ranks second in its contribution to the overall discriminating ability of the model.

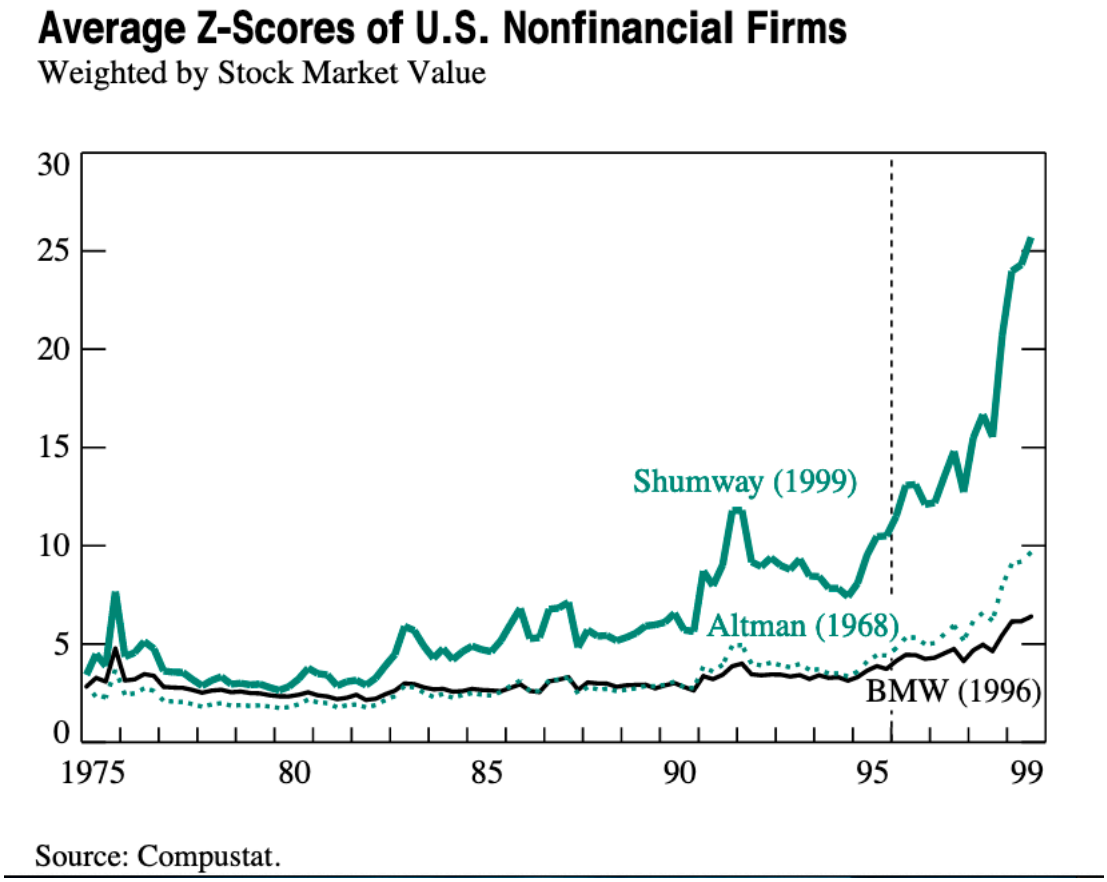
Final formula of the Altman Z-Score:

$$Zscore = \beta_1 X_{WC/TA} + \beta_2 X_{RE/TA} + \beta_3 X_{EBIT/TA} + \beta_4 X_{MVE/TL} + \beta_5 X_{S/TA} \tag{1}$$

Due to the original computer format arrangement, variables WC/TA through MVE/TL must be scaled as absolute percentage value For instance, the firm whose net working capital to total assets (WC/TA) is 10% should be included as 10.0% and not 0.10. Only variable S/TA should be expressed in a different manner: that is, a S/TA ratio of 200% should be included as 2.0. [3].

3.1.1 Interpretation of the Output

The original cutoff was a Z score below 1.1 but recent tests shown the average Z-Score increasing significantly with the average rising from the 4-5 level in 1970-1995 period to almost 10 in 1999. [4].



The majority of increase in average Z-Scores was due to the dramatic climb in stock prices and its impact on Market Value of Equity/Book Value of Total Liabilities (MVE/TL). The lower bond of the zone-of-ignorance (1.81) is a more realistic cutoff Z-Score. There are different versions of the model that are more cross-industry and involve different ratios and modification to the original model and to the final cut-off area.

3.1.2 Adapting the Model for Non-Manufacturers

In the model for Non-Manufacturers the Sales/Total assets (S/TA) is removed in order to minimize the potential industry effect which is more likely to take place when such an industry-sensitive variable as asset turnover is included. The "new" Z score model is :

$$Z_{NM}score = \beta_1 X_{WC/TA} + \beta_2 X_{RE/TA} + \beta_3 X_{EBIT/TA} + \beta_4 X_{MVE/TL} \tag{2}$$

3.1.3 The modern Zeta Score

After an iterative process of reducing the number of variables, the more advanced model selected 7-variables that are the most reliable in various validation procedures and they were defined as follow:

X1 : Return on assets

Measured by the earnings before interest and taxes/total assets. This variable has proven to be extremely helpful in assessing firm performance in several past multivariate studies.

X2 : Stability of earnings

Measured by a normalized measure of the standard error of estimate around a five to ten-year trend in X1. Business risk is often expressed in terms of earnings fluctuations and this measure proved to be particularly effective. It has been assessed the information content of several similar variables which attempted to measure the potential susceptibility of a firm's earnings level to decline which could compromise its ability to meet its financial commitments. These variables were quite significant on a univariate level but did not enter into the final multivariate model.

X3 : Debt service

Measured by the familiar interest coverage ratio, i.e., earnings before interest and taxes/total interest payments (including that amount imputed from the capitalized lease liability). This measure has been transformed by taking the log 10 in order to improve the normality and homoscedasticity of this measure.

X4 : Cumulative profitability

Measured by the firm's retained earnings (balance sheet)/total assets. This ratio, which imputes such factors as the age of the firm, debt and dividend policy as well as its profitability record over time, was found to be quite helpful in the Z-Score model, discussed earlier. The cumulative profitability measure is unquestionably the most important variable-measured univariately and multivariately.

X5 : Liquidity

Measured by the familiar current ratio. Despite previous findings that the current ratio was not as effective in identifying failures as some other liquidity measures, it is slightly more informative than others, such as the working capital/total assets ratio.

X6 : Capitalization

Measured by common equity/total capital. In both the numerator and the denominator, the common equity is measured by a five-year average of the total market value, rather than book value. The denominator also includes preferred stock at liquidating value, long-term debt and capitalized leases. In the more recent studies a 5-year average was used to smooth out possible severe or temporary market fluctuations and to add a trend component.

X7 : Size

Measured by the firms' total assets. This variable, as is the case with the others, was adjusted for financial reporting changes. No doubt, the capitalization of leasehold rights has added to the average asset size of both the bankrupt and nonbankrupt groups. The size variable has been transformed with a logarithmic transformation to help normalize the distribution of the variable due to outlier observations.

$$Zetascore = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 \tag{3}$$

A final consideration about the Altman Z score is that it is not very effective with financial companies due to the particulare nature of the balance sheets.

Summarizing

The first conclusion of the study is that the Altman-Z score cut-off range $1.81 < Z < 2.99$ should be updated regularly instead of being used as a fixed range.

The ZETA model for assessing bankruptcy risk of corporations demonstrates improved accuracy over existing failure classification model (Z-Score) and, perhaps more importantly, is based on data more relevant to current conditions and to a larger number of industrial firms. Recall, to use of the Z" model for non-manufacturer.

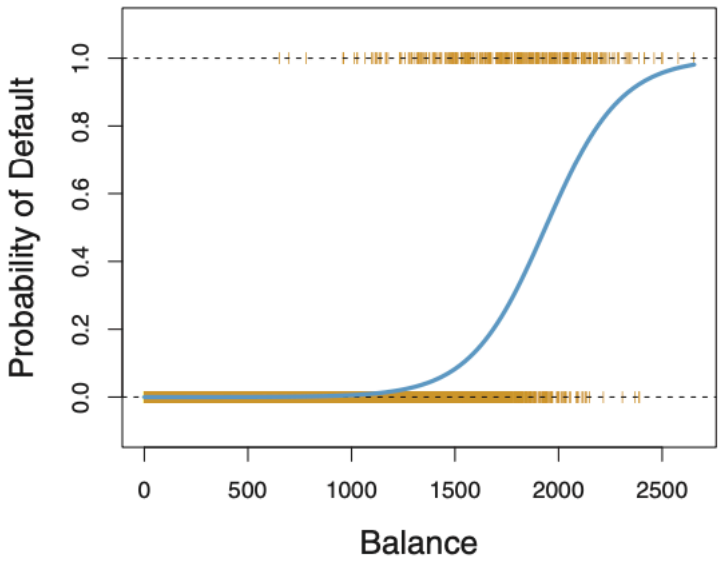
The Z-score isn't very effective for new companies with little to zero earnings. Regardless of their actual financial health, these companies will score low. Moreover, the Z-score doesn't address the cash flows of a company. Rather, it only hints at it through the use of the net working capital-to-asset ratio.

Even if this approach was theoretically successful, real scenarios often stressed the need to overcome imbalanced distribution of classes and the ability to adapt to emerging companies.

3.2 Logistic Regression

Logistic regression maps the probability of belonging to a specific class according to some input values (as specified in the abstract all the problems statement will involve a binary classification with Y outcome 1 for Bankruptcy and 0 for Non Bankruptcy).

In scientific terms, the goal is to predict the chance $P(Y = 1)$ using the logistic function.



Source: Introduction to Statistical Learning

Assumptions of Logistic Regression

- Response variable should be belong to a Bernoulli distribution with n= 1 and the 2 classes must be mutually exclusive

- Statistical independency of the n observations
- No strong correlation between the regressors
- Linear relationship between any continuous independent variables and the logit transformation of the dependent variable

According to [Hardin JW, Hardin JW, Hilbe JM, Hilbe J. Generalized linear models and extensions. Stata press; 2007.].

The logistic regression model arises from the desire to model the posterior probabilities of the K classes via linear functions in x, while at the same time ensuring that they sum to one and remain in [0,1]. (in this case we restrict the model to K =2 classes but in other cases the model is built using the One-VS-Rest Algorithm)

The model has the form:

$$Pr(Y_i = 1|X_1..X_n) = \frac{1}{1 + \exp(-[\beta_0 + \beta_1 X_i + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_n X_n])} \tag{4}$$

Logistic regression models are usually fit by maximum likelihood, using the conditional likelihood of G given X. Since Pr(G|X) completely specifies the conditional distribution, the multinomial distribution is appropriate. The log-likelihood for N observations is

$$\boldsymbol{\ell}(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i, \theta) \tag{5}$$

where

$$p_k(x_i, \theta) = Pr(G = k, X = x_i; \theta) \tag{6}$$

It is convenient to code the two-class g_i via a 0/1 response y_i, where $y_i = 1$ when $g_i = 1$,and $y_i = 0$ when $g_i = 2$.

Let $p_1(x; \theta) = p(x;)$, and $p_2(x; \theta) = 1-p(x; \theta)$. The log-likelihood can be written as:

$$\boldsymbol{\ell}(\beta) = \sum_{i=1}^{N} \{y_i log p(x_i, \beta) + (1 - y_i) log(1 - p(x_i, \beta))\} \tag{7}$$

To maximize the log-likelihood, we set its derivatives to zero. These score equations are

$$\frac{\partial \boldsymbol{\ell}(\beta)}{\beta} = \sum_{i=1}^{N} x_i(y_i - p(x_i; \beta)) = 0 \tag{8}$$

[5].

To solve the score equations there are several aglorithms that are well explained in "The Elements of. Statistical Learning. Data Mining, Inference, and Prediction.Trevor Hastie. Robert Tibshirani. Jerome Friedman" page 120.

The advantages of this type of models are the efficiency in terms of results with respect to computational resources and also the interpretability; in fact in many industries logistic regression is used as a benchmark.

However artificial intelligence and machine learning have become a major research direction in the bankruptcy prediction. In the era of increasing volumes of data it turned out that the linear models like the logistic regression or logit (probit) models are unable to reflect non-trivial relationships among economic metrics.

[6].

3.3 Support Vector Machines

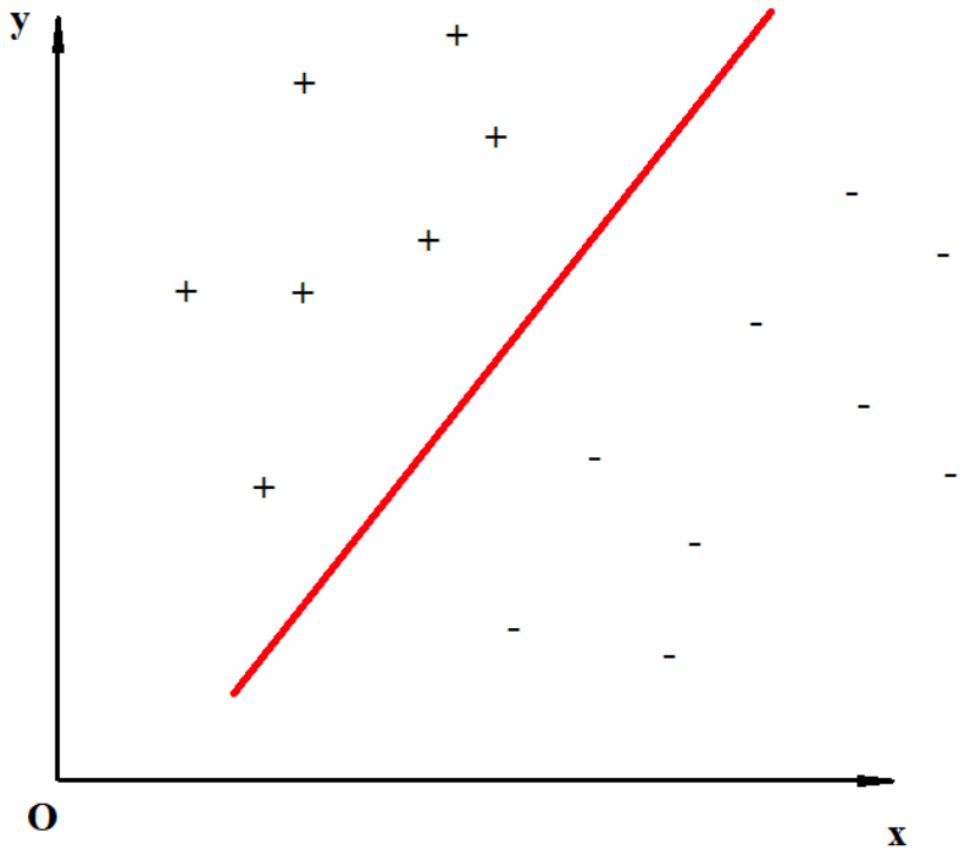
In order to study the non linear relationships between features and bankruptcy, artificial intelligence is a disruptive change in the industry. The main advantage to step over traditional statistical methods is the possibility to learn directly from data. One of the most successful model was the support vector machine. Support Vector Machine (SVM) is a supervised learning algorithm that classifies data. The final goal of the SVM is to find out the best hyperplane that can successfully divide data into n regions.

Given:

$D = (x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n); y_i \in \{-1, +1\}$

The SVM Hyperplane might look like this:

SVM Hyperplane



In math terms, the hyperplane can be represented as:

$$\omega^T X + b = 0 \tag{9}$$

where

$$\omega = (\omega_1, \omega_2, \cdots, \omega_n)^T \tag{10}$$

ω denotes the slope of the hyperplane. The letter b denotes the distance between the hyperplane and the origin. Now it's required and necessary to ensure that all the inputs of these positive points above the hyperplane can give positive results, while all the negative ones below the hyperplane correspond to negative results.

$$\omega^T X + b > 1; y_i = 1 \tag{11}$$

$$\omega^T X + b < -1; y_i = -1 \tag{12}$$

the target of SVM is to find out the minimum value of $0.5|\omega|^2$ so that

$$y_i \omega^T X + b > 1 \tag{13}$$

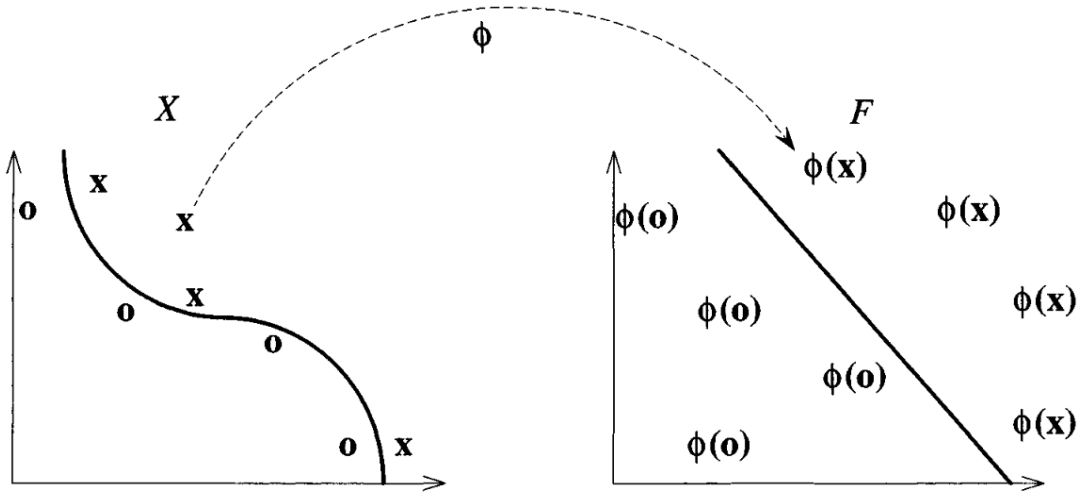
and so by using Lagrange we obtain

$$L = 0.5|\omega|^2 + \sum_{i=i}^{n} l_i(1 - y_i \omega^T X + b) \tag{14}$$

Source: [7].

The linear model constructed in the new space can represent a non linear decision boundary in the original space.In the new space, an optimal separating hyperplane (OSH) is constructed. Thus, SVM is known as the algorithm that finds a special kind of linear model, the maximum margin hyperplane. The maximum margin hyperplane gives the maximum separation between decision classes. The training examples that are closest to the maximum margin hyperplane are called support vectors. The complexity of the target function to be learned depends on the way it is represented, and the difficulty of the learning task can vary accordingly. Ideally a representation that matches the specific learning problem should be chosen. So one common preprocessing strategy in machine learning involves changing the representation of the data:

$$X = (x_1, x_2, x_3 \cdots x_n) \mapsto \phi(X) = (\phi(x_1), \phi(x_2), \phi(x_3), \cdots \phi(x_n)) \tag{15}$$



In order to learn non-linear relations with a linear machine, we need to select a set of non-linear features and to rewrite the data in the new representation. This is equivalent to applying a fixed non-linear mapping of the data to a featurespace, in which the linear machine can be used.

The process described previously is defined as Kernel trick , the algorithm itself is well described in [8].

SVM still one of the most competitive methods in bankruptcy prediction, however the disadvantages of SVM are that the kernel function must be carefully hand-tuned and it is impossible to obtain a comprehensible model. [9].

3.4 Modern approach to bankruptcy prediction

Given the previous approaches it is evident that generalized linear models struggle to perform ; by the other hand more complex methods are a good solution in terms of performance but they lack in interpretability. That is why it has been advocated to take advantage of different learning paradigm, namely, the ensemble of classifiers. The idea of the ensemble learning is to train and combine typically weak classifiers to obtain better predictive performance. First approaches but still very successful were bagging and boosting [10]. [11]. [12].

The idea of boosting was further developed to the case of unequal classification costs and imbalanced data. [13]. [14].

Recently, the boosting method was modified to optimize a Taylor expansion of the loss functions, an approach known as Extreme Gradient Boosting that obtains state-of-the-art results in many problems on Kaggle competitions . Recently, it has been shown that the ensemble classifier can be successfully applied to the bankruptcy prediction and it significantly beats other methods. [15]. [16]. [17].

In order to understand how an ensemble works, the following lines will firstly define the concept of Decision Tree and the related ensemble techniques.

3.4.1 Decision Tree

Decision Tree methods are simple and useful for interpretation. The smaller decision trees can be understood by both computer scientists and domain experts, providing a valuable communication means between the two parties. Decision trees can be applied to both regression and classification problems. The main idea is to make a recursive partitioning of data based on variables that maximize or minimize an attribute selection measure to separate the classes.

Key Terminology of Tree Based Models:

- Top Down Structure
- Attribute selection measure ASM
- Binary Split
- Root Node
- Leaves
- Depth
- Pruning

Top Down Structure The approach is top-down because it begins at the top of the tree(at which point all observations belong to a single region and they are all classified as elements of a particular class.) Each split is indicated via two new branches further down on the tree. It is greedy because at each step of the tree-building process, the best split is made at that particular step.

Attribute selection measure ASM:

- Entropy
- Information Gain
- Gini Index
- Sum of Squared Residuals

Entropy

$$H = -\sum_{k=1}^{K} p_k log(p_k) \tag{16}$$

Where p is the probability of a class for K classes

Entropy is the measures of impurity, disorder or uncertainty and it can be use to evaluate the quality of the partition. The entropy will take on a small value if the node is pure. The goal is to minimize the Entropy.

Information Gain It is a measure to maximize and it depends by the entropy of the previous split and the current. Entropy need to be minimized and it calculated as shown before. If the current split does a better job compared to the previous situation before the split the information gain is increased. Formal definition:

$$IG = Entropyprevious(X) - WeightedAvg * EntropyCurrent(X) \tag{17}$$

Where X are the data belonging to the regions.

The weighted average is used because there are 2 distinct regions after the split, the average will give a summary of the entropy in both. It is possible to find an imbalanced situation where in the first region there are much more or much less data compared to the complementary region, so the weights are the percentage of total data falling into the region.

Gini Index It is very similar to Entropy since both measures the randomness among K classes in the region. Formal Definition:

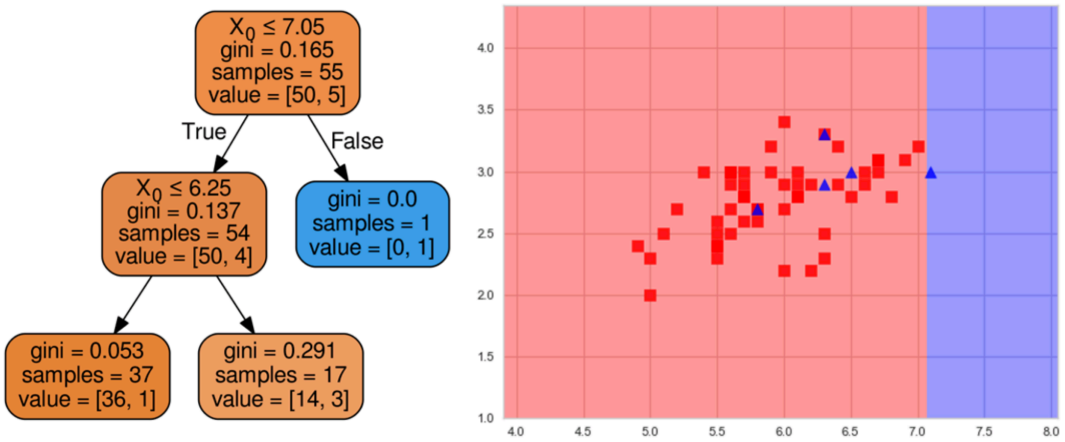
$$GI = 1 - \sum_{k=1}^{K} p_k^2 \tag{18}$$

Just like the entropy, the goal is to minimize the Gini Index.

Sum of Squared Residuals It is the sum of squared residuals for each i element belonging to the j region the J region space, it is used for continuous data and it must be minimized.

$$RSS = \sum_{j=1}^{J} \sum_{iR_j} (y_i - \hat{y_j})^2 \tag{19}$$

Binary Split The binary split is the system that separates data in two different regions according to a specific value of the feature, the value is chosen using the ASM. The node is the part of the algorithm where these separations take place



Source: sklearn documentation

Root Node The first split of the data is called root node, and it is found according to the ASM. Among all the feature the root is the most efficient to split the classes and it is used at the top of the tree. This is a practicle application of a Greedy structure ,meaning that every step its made with the most optimal decision.

Leaves The Leaves are the terminal region created after the split, the final result is the average (continuous) or the mode (categorical) of the data belonging to the region. The algorithm creates leaves when is not allowed to split. It is part of the greedy structure to limit the number of partitions in order to avoid an overfitting of the training set. The limit of partitions is set through parameters related to the number of samples belonging to the region.

Depth The Depth of the tree is the number of splits allowed, is often decided according to the training, the first training session should arrive to an overfitting point and then the algorithm start pruning the tree.

Pruning It is a technique that resolves the problem of overfitting and it helps decreasing the computational complexity of the algorithm. To prune a tree there are several parameters that measures the quality of the split according to the ASM. If the split does not generate some utility (above a chosen threshold) the split is not valid and the region become a leaf. This task is performed using another Algorithm called Weakest link Pruning or Cost Complexity Pruning. It is a more complex measure that takes count a penalty element proportionally to the number of terminal nodes (leaves). To compute the pruning the trees are tested using different data, that didn't belong to the training set.

Let the Tree Score of the algorithm be:

$$TreeScore = ASM + alpha|T| \tag{20}$$

In case of a regression tree it is:

$$\sum_{m=1}^{T} \sum_{x_i} (y - \hat{y})^2 + \alpha |T| \tag{21}$$

Here |T| indicates the number of terminal nodes. Alpha is a parameter that need to be tuned. It follows that the more complex tree will be the best in terms of training error; but it will not be chosen since the number of leaves, and so the number of unnecessary splits, will penalize it. This is the first situation that makes evident the well-known trade-off Bias-Variance in tree-based models. There are different algorithms for Decision trees but one of the most common is CART.

CART (Classification and Regression Tree) uses the Gini index method to create splits and decide the root node as well.

Summarizing:

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.
   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.
   Average the results for each value of α, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Source: Introduction to statistical Learning. Trevor Hastie Robert Tibshirani. page 309. Tree Based Models.

Another important detail is that Decision trees are really versatile in terms of preprocessing. The need of Dummy Variables is not required and in some cases it will make the algorithm underperform due to sparsity in the branches. The need to scale data is also not required and the multicollinearity problem is not significant.

Advantages and drawbacks of the Decision Tree Algorithm.

- Easy Interpretability
- Low Complexity
- Efficient in Preprocessing
- Effective learning with small datasets

Drawbacks

- Compared to other algorithms they under-perform
- Very High Variance and easy overfitting

[17].

By aggregating many decision trees, using methods like bagging, random forests, and boosting, the predictive performance of trees can be substantially improved. In many fields the aggregation of trees that is part of the ensemble learning methods is among the most effective methods. It is taken to production by the majority of big industries and takes the first places in many Kaggle Competition. (Kaggle is an open community for Data Science Competitions).

3.4.2 Ensemble Methods

Ensemble learning is a machine learning paradigm where multiple models (often called "weak learners") are trained to solve the same problem and combined to get better results. The main idea is that when weak models are correctly combined they lead to a more accurate and/or robust models.

Regardless of the model, there is always a trade off between the variability of results in the training set and the one in the test set. In other words a model can adapt very easily to the training data but performs very poorly on new instances, in this case the model has a low bias, if not already overfitting, and high variance.

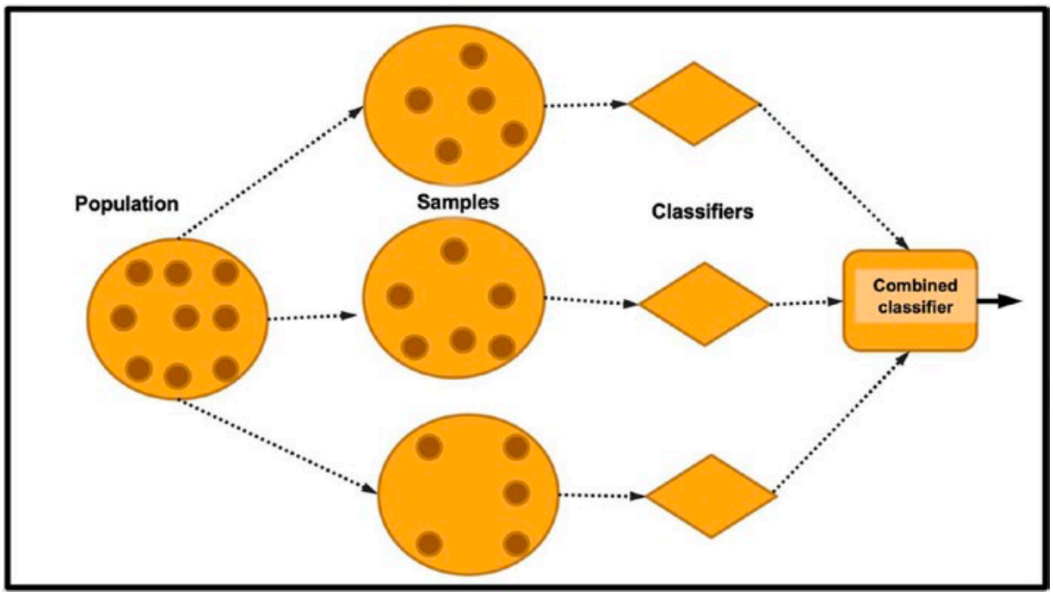
There are several techniques to deal with this trade-off; one of the most effective is to exploit the ability of weak learners to "overfit" a portion of data on which they have been trained. The combination of the n models leads to a situation where the noise of the single models is reduced, while the ability to perform on the portion of data is preserved and combined along with the n weak learners. The result is a single strong learner with lower variance and bias.

Ensemble learning techniques can be divided under a broad umbrella of three classes:

- mixing training data
- mixing combinations
- mixing models.

Mixing Training Data Instead of training a single large classifier on all the training data, turns out that dividing the training data into multiple chunks and train separate classifiers on each subset of training is more effective. In the end, the outputs of all of these classifiers are combined.

This approach ensures that classifiers capture sufficient diversity as they are trained on a subset of the population. By combining the output of these diverse learners, is possible to achieve superior accuracy compared to a case in which we trained a single learner on a whole population (training data). This kind of division of training data is called Bagging.



Source: Ensemble Learning for Ai Developers

[18].

Bagging The decision trees discussed previously suffer from high variance. This means that if we split the training data into two parts at random, and fit a decision tree to both halves, the results could be quite different. In contrast, a procedure with low variance will yield similar results if applied repeatedly to distinct data sets. Linear regression tends to have low variance, if the ratio of n to p is moderately large. Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method; it is particularly useful and frequently used in the context of decision trees. Averaging a set of observations reduces variance. Hence a natural way to reduce the variance and hence increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. In other words, we could calculate $f_1(x), f_2(x), \cdots, f_B(x)$ using B separate training sets, and average them in order to obtain a single low-variance statistical learning model.

$$\hat{f_{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x) \tag{22}$$

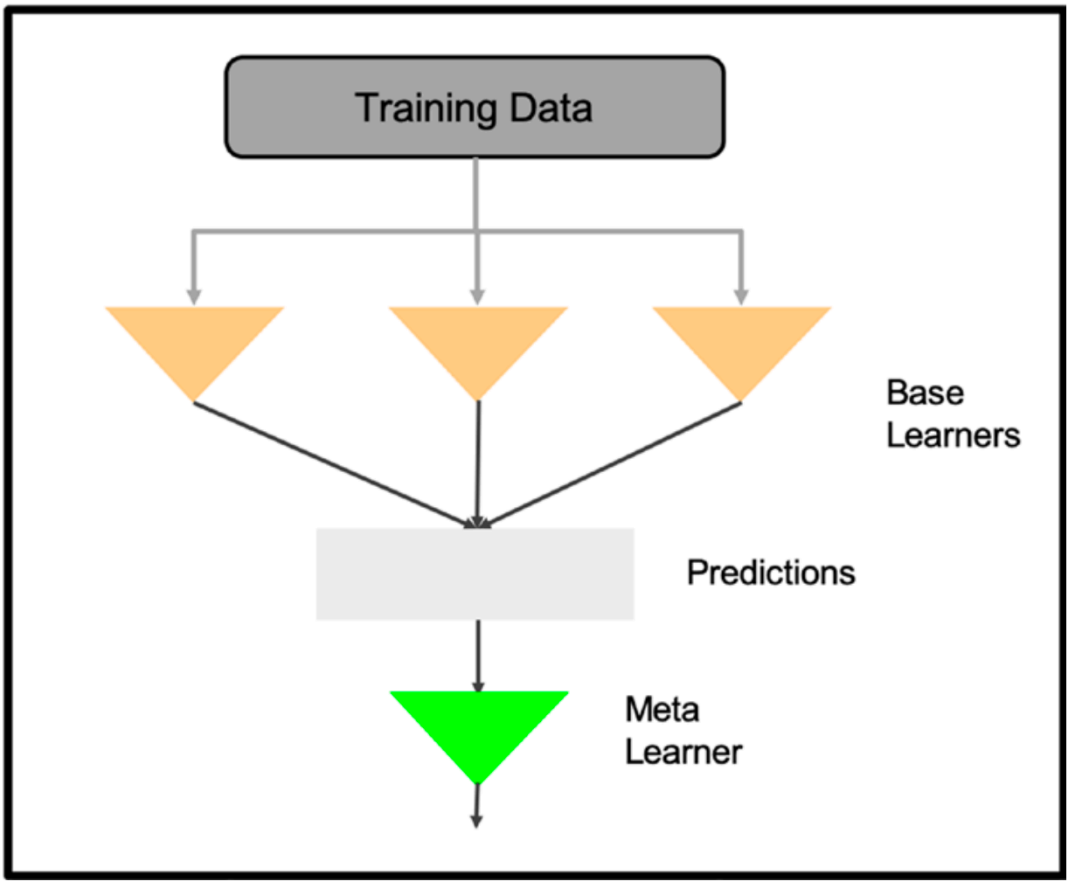
Instead, we can use the bootstrap techinque by taking repeated samples from the (single) training data set. In this approach we generate B different bootstrapped training data sets, then train our method on the bth bootstrapped training set in order to get $f_b * (x)$ and finally average all the predictions to obtain:

$$\hat{f_{bagging}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b * (x) \tag{23}$$

This is called bagging. While bagging can improve predictions for many regression methods, it is particularly useful for decision trees. To apply bagging to regression trees, B regression trees are used with B bootstrapped training sets, and the average result of predictions is the output. These trees are grown deep, and they are not pruned. Hence each individual tree has high variance, but low bias , so averaging these B trees reduces the variance. Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure.

[19].

Mixing Combinations In mixing models combination we start out with a collection of learners, in which each learner is trained on a particular subset of training objects. If the model learner has a weak performance, we could give more weight to that particular learner with increased emphasis where previous learner had weak performance. This combination of learners is known as boosting.



Source: Ensemble Learning for Ai Developers

[20].

Boosting Boosting refers to a family of algorithms that are able to convert weak learners to strong learners. The main principle of boosting is to fit a sequence of weak learners−models that are only slightly better than random guessing, such as small decision trees to weighted versions of the data. More weight is given to examples that were misclassified by earlier rounds.

Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. Here the discussion is restricted to boosting in the context of decision trees.

Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.

Another example of mixing combination is Stacking

Stacking The idea of stacking is to alternate through many rounds of training session a base weak learner and a meta model in a sequencial manner; this meta learner will try to make up for the error of the previous model and improve the results via update of parameters. The stacking method is often used to predict residual or pseudo residuals of the base learners. The intuition is that stacking often considers heterogeneous weak learners, learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models predictions.

The training process For Regression problems. The first step is to fit a decision tree and then fit another tree using the current residuals rather than the outcome Y as the response. Next , a new decision tree is added into the fitted function in order to update the residuals. Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm. By fitting small trees to the residuals, the model slowly improve f in areas where it does not perform well.

The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals. In general, statistical learning approaches that learn slowly tend to perform well. Note that in boosting, unlike in bagging, the construction of each tree depends strongly on the trees that have already been grown. This is the main reason that defines Stacking and Boosting as "sequential train methods".

Boosting classification trees proceeds in a similar way but the concepts of residual is different due to the nature of the response variable , instead of pure residuals the meta model works with "pseudo-residuals".

Boosting has three tuning parameters

- The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large. Although this overfitting tends to occur slowly if at all is better to use cross-validation to select B.
- The shrinkage parameter λ ,a small positive number.This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small can require using a very large value of B in order to achieve good performance.
- The number d of splits in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split. In

this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

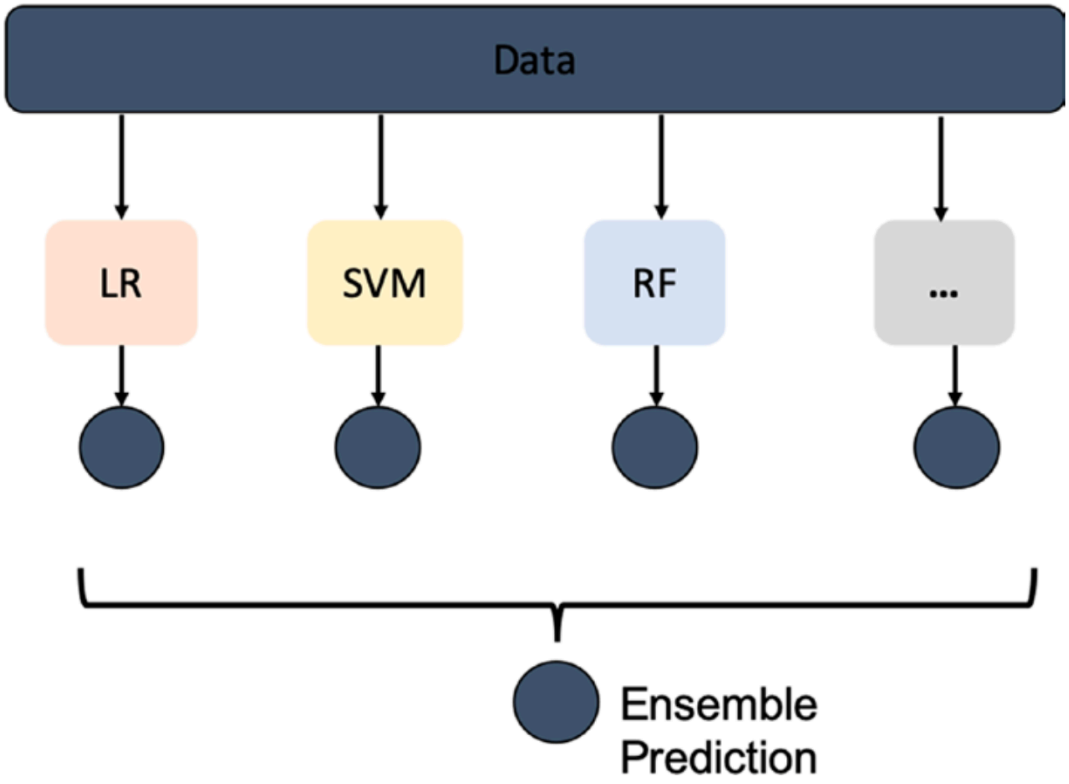
Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Source : An Introduction to Statistical Learning with Applications in R

[21].

Mixing Models The third type of ensemble learning methods involve varying models. It is possible to use different models and even in the case of a single machine learning model, is possible to use settings/hyperparameters that differ between training runs. Instead of relying on only a single model or single hyperparameter settings for the machine learning tasks, a combination of n models that diffear for model type or tuning is indeed a stronger predictor . This leads us to have better accuracy and lower bias when compared to using a single model or single settings.



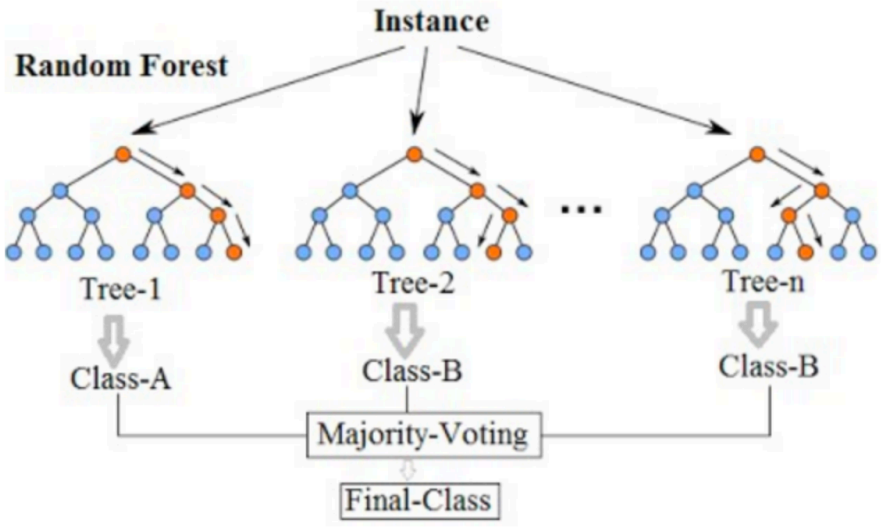
Source: Ensemble Learning for Ai Developers

3.5 Ensemble Tree Based Models

3.5.1 Random Forest

Key Concepts:

- The random forest is a classification/regression algorithm consisting of many decisions trees.
- It uses bagging and feature randomness to create uncorrelated trees ,trained on a slightly different set of the observations and considering a limited number of features for each tree.
- The final predictions of the random forest are made by averaging the predictions of each individual tree in regression and by majority vote in classification.



Source: Mathematics behind Random forest and XGBoost-Article

The essential idea in bagging is to average many noisy but approximately unbiased models, and hence reduce the variance. Trees are ideal candidates for bagging, since they can capture complex interaction structures in the data, and if grown sufficiently deep, have relatively low bias. Since trees are notoriously noisy, they benefit greatly from the averaging. Moreover, since each tree generated in bagging is identically distributed , the expectation of an average of B such trees is the same as the expectation of any one of them. This means the bias of bagged trees is the same as that of the individual trees, and the only hope of improvement is through variance reduction. This is in contrast to boosting, where the trees are grown in an adaptive way to remove bias, and hence are not i.d. The algorithm consist in averaging B random variables, each with variance σ^2, each tree has variance $\frac{1}{B}\,\sigma^2$. If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation ρ, the variance of the average is

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{24}$$

As B increases, the second term disappears, but the first remains, and hence the size of the correlation of pairs of bagged trees limits the benefits of averaging. The idea in random forests is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much. This is achieved in the tree-growing process through random selection of the input variables.

Here the final Algorithm:

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Source: The Elements of. Statistical Learning: Data Mining, Inference, and Prediction. Second Edition. February 2009. Trevor Hastie · Robert Tibshirani.[22].

3.5.2 Extreme Gradient Boosting

XGB is the most efficient version of Boosting Algorithms for both regression and classification problems. It is a supervised machine learning algorithm that belongs to the ensemble methods and it has been introduced in 2014 by Tianqi Chen. The structure of the algorithm is Sequential , since we start by a single weak learner and the prediction is updated via metalearners for each iteration.

The main idea is to build a sequence of learners $f(x)$ which prediction are updated by a metalearner $h(x)$, the combination of the two is the new predictor $f'(x)$ that must minimize the objective function L according to the nature of the problem.

In this case :

Let us denote by $x \in X$ a vector of features describing an enterprise, where $X \subseteq R^D$ and by $y \in \{0, 1\}$, remind by the abstract that 1 stands for Bankruptcy in a supervised machine learning problem. XBG uses decision trees as discriminative models , for reasons that have been discussed previously. The specific version of tree is the CART (Classification and Regression Tree).A CART tree can be represented by the weights associated with the leaves in the tree structure:

$$f_k(x_n) = w_{q(x)} \tag{25}$$

where $q(x_n)$ is the function that returns the path id in the structure of the tree , $q : R^D \to \{1, \ldots, T\}$ with T number of leaves (or simply path). A path is ended with a leaf that cointains a weight w_i.

The goal is to create an ensemble of K decision trees

$$h_K(x) = \sum_{k=1}^{K} f_k(x) \tag{26}$$

where $f_k \in F$ for $k = \{1, \ldots K\}$ and F is a space of possible decision trees. The prediction of the decision tree will be :

$$p(y = 1|x) = \sigma(h_K(x)) \tag{27}$$

The term σ will refer to the logistic function. Once the logistic function is applied the standard cuttoff will be 50% and so every value over 50 will be classified as 1 (Bankrupt).

Given the training data $D = \{x_n, y_n\}_{n=1}$ for N instances , the model will minimize:

$$L_\Omega(\Theta) = L(\Theta) + \Omega(\Theta) \tag{28}$$

Where :

- Θ are the parameters of the model
- Ω is the regularization term
- $L(\Theta)$ is the loss function

The previous expression can be further reviewed as :

$$\sum_{n=1}^{N} l(y_n, h_K(x_n)) + \sum_{k=1}^{K} \Omega(f_k) \tag{29}$$

The definition of Loss function is set according to the nature of the problem, being classification the case we define L as :

$$L(\Theta) = \sum_{n=1}^{N} \{y_n log(1 + exp(-h_k(x_n))\} + \{(1 - y_n) log(1 + exp(h_k(x_n))\} \tag{30}$$

This expression of the loss function is known as LogitBoost.The learning problem can be solved iteratively by adding new weak learners f_k sequentially.

XGB uses in the optimization problem a Taylor approximation with respect to the learner $h_{k-1}(x_n)$ so :

$$L_\Omega(\Theta) \approx \sum_{n=1}^{N} \{l(y_n, h_{k-1}(x_n)) + g_n f_k(x_n) + 0.5 h_n f_k^2(x_n)\} + \Omega(f_k) + constant \tag{31}$$

Note that g is the first derivative with respect to $h_{k-1}(x_n)$.

h_n is the second derivative , or Hessian , and it must not be confused with the learner notation $h_{k-1}(x_n)$.

The optimal output value is given according to :

$$g_n = \sigma(h_{k-1}(x_n) - y_n) \tag{32}$$

$$h_n = \sigma(h_{k-1}(x_n) - y_n)(1 - \sigma(h_{k-1}(x_n) - y_n)) \tag{33}$$

There are different possible regularization terms. However, we focus on the regularizer in the following form:

$$\Omega(f_k) = \gamma T + 0.5\lambda \sum_{t=1}^{T} w_t^2 \tag{34}$$

λ and γ are the parameters that help to prune and avoid overfitting in the trees

The optimal value of the weight in the t-th leaf is as follows:

$$\dot{w}_t = -\frac{G_t}{H_t + \lambda} \tag{35}$$

G is the sum of all the g derivatives and H the sum of all the h derivatives + λ parameter. h derivatives is also called Cover. In regression problems the Cover is just equal to the number of instances.

The optimal value of the approximated objective function is given by:

$$L_\Omega(\Theta) \approx -0.5 \sum_{i=1}^{T} \frac{G_t^2}{H_t + \lambda} + \gamma T + constant \tag{36}$$

The key problem in the above consideration is that the structure of the tree is not given in advanced and searching all possible structures is computationally infeasible. To overcome this issue the tree

is being constructed starting from the root and further the best attribute to be located in the root is selected and the best split point for the attribute is chosen. The splitting process is performed until the quality of the model is improved. As the splitting criterion we take the information gain just like a regular decision tree. The more leaves are present , the more the parameter γ will penalize the algorithm.

The model can be also regularized by:

- setting minimal number of examples combined with each of the leaves
- setting maximal depth of the tree by setting the percentage of features randomized for each iteration of constructing the tree
- adding the new tree with corrected influence of the trees in the committee:

$$h_k(x_n) = h_{k-1}(x_n) + \varepsilon f_k(x_n) \tag{37}$$

$\varepsilon \in \{0, 1\}$ and it is defined as learning rate or eta and it is used as shrinkage parameter on the metalearner. [23].

4 Experimental and Innovative Approaches in Bankruptcy prediction

One of the most intersting work in Bankruptcy prediction was performed in Poland by the Department of Operations Research in the University of Wrocław ,2016. The study considered financial ratios of Poland manufacturing companies which have faced Bankruptcy in a time range from 1 to 5 years. Data are available at the UCI Machine Learning Repository under the Business session. link: https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data

ID	Description	ID	Description
X1	net profit / total assets	X33	operating expenses / short-term liabilities
X2	total liabilities / total assets	X34	operating expenses / total liabilities
X3	working capital / total assets	X35	profit on sales / total assets
X4	current assets / short-term liabilities	X36	total sales / total assets
X5	[(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365,	X37	(current assets - inventories) / long-term liabilities
X6	retained earnings / total assets	X38	constant capital / total assets
X7	EBIT / total assets	X39	profit on sales / sales
X8	book value of equity / total liabilities	X40	(current assets - inventory - receivables) / short-term liabilities
X9	sales / total assets	X41	total liabilities / ((profit on operating activities + depreciation) * (12/365))
X10	equity / total assets	X42	profit on operating activities / sales
X11	(gross profit + extraordinary items + financial expenses) / total assets	X43	rotation receivables + inventory turnover in days
X12	gross profit / short-term liabilities	X44	(receivables * 365) / sales
X13	(gross profit + depreciation) / sales	X45	net profit / inventory
X14	(gross profit + interest) / total assets	X46	(current assets - inventory) / short-term liabilities
X15	(total liabilities * 365) / (gross profit + depreciation)	X47	(inventory * 365) / cost of products sold
X16	(gross profit + depreciation) / total liabilities	X48	EBITDA (profit on operating activities - depreciation) / total assets
X17	total assets / total liabilities	X49	EBITDA (profit on operating activities - depreciation) / sales
X18	gross profit / total assets	X50	current assets / total liabilities
X19	gross profit / sales	X51	short-term liabilities / total assets
X20	(inventory * 365) / sales	X52	(short-term liabilities * 365) / cost of products sold)
X21	sales (n) / sales (n-1)	X53	equity / fixed assets
X22	profit on operating activities / total assets	X54	constant capital / fixed assets
X23	net profit / sales	X55	working capital
X24	gross profit (in 3 years) / total assets	X56	(sales - cost of products sold) / sales
X25	(equity - share capital) / total assets	X57	(current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X26	(net profit + depreciation) / total liabilities	X58	total costs /total sales
X27	profit on operating activities / financial expenses	X59	long-term liabilities / equity
X28	working capital / fixed assets	X60	sales / inventory
X29	logarithm of total assets	X61	sales / receivables
X30	(total liabilities - cash) / sales	X62	(short-term liabilities *365) / sales
X31	(gross profit + interest) / sales	X63	sales / short-term liabilities
X32	(current liabilities * 365) / cost of products sold	X64	sales / fixed assets

The innovative component was the idea of Synthetic Features that toke place in the Algorithm. The central idea in their approach is to generate synthetic features that may have better influence on prediction than typical economic factors. The synthetic features are generated by random selection of two existing features and random selection of arithmetical operation to be performed on them. The complete algorithm is the following:

Algorithm 1: Ensemble of boosted trees with synthetic features

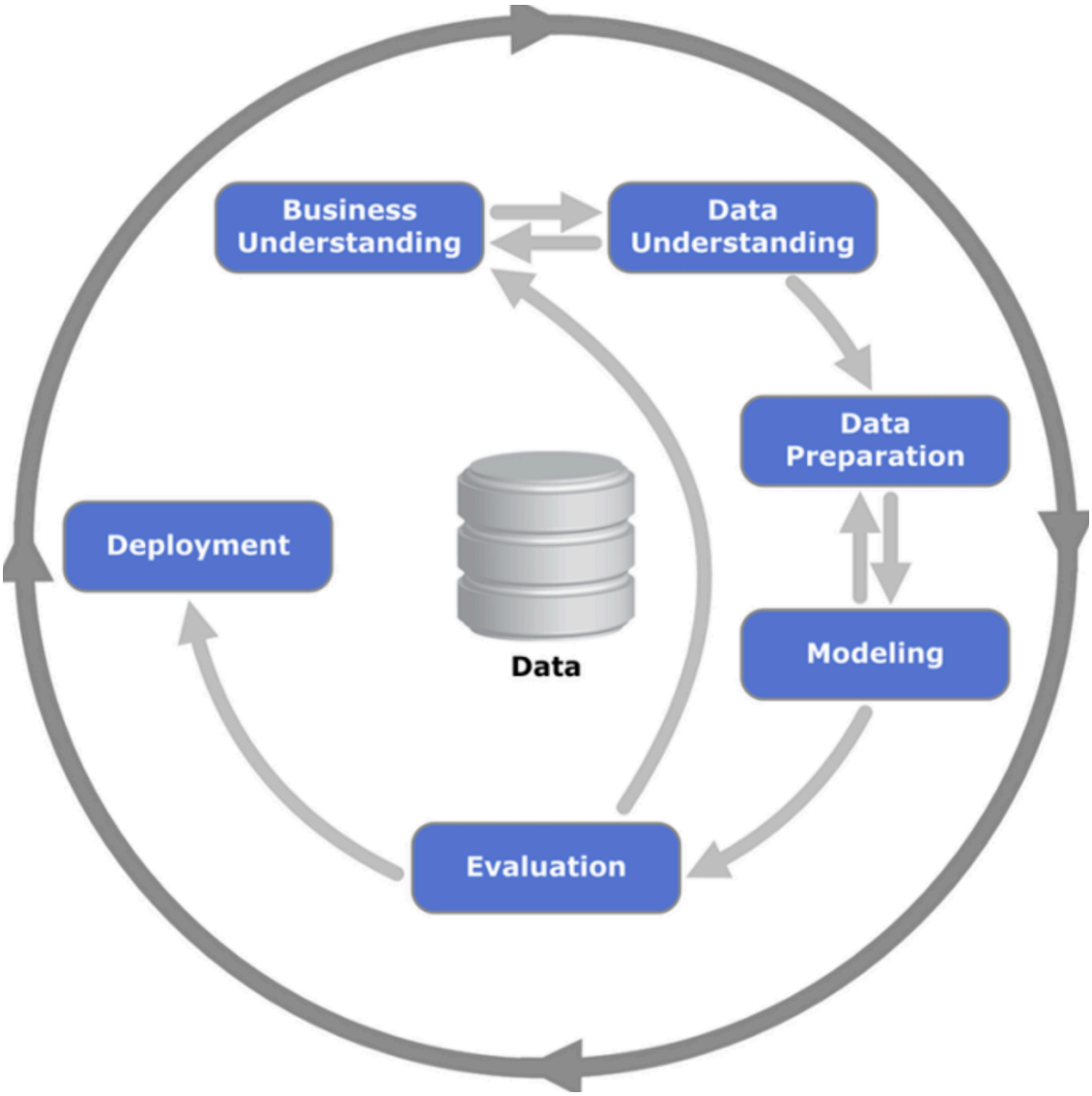
Input : $\mathcal{D}$: training set, D_{new}: number of synthetic features,
L: number of base learners, η: features acceptance threshold

Output: $H = \{h_1, \ldots, h_K\}$: set of base learners

for $k = 1, \ldots, K$ **do**
  Train h_k using $\mathcal{D}$;
  Remove features from $\mathcal{D}$ for which $m_d < \eta$;
  Estimate $\boldsymbol{\theta}_F$ from model h_k;
  for $d = 1, \ldots, D_{new}$ **do**
    Sample features f_1 and f_2 from distribution $\boldsymbol{\theta}_F$;
    Sample operation $\circ$ from $\{+, -, *, /\}$;
    Generate new feature $f_{new} = f_1 \circ f_2$;
    Extend $\mathcal{D}$ with new values of f_{new};
  end
end
return $H = \{h_1, \ldots, h_K\}$;

5 Creating the Ensemble Model for Bankruptcy prediction

5.1 Methodology



5.2 Business Understanding

The Dataset is a collection in CSV format of Financial Ratios belonging to manufacturing companies in Poland; the same used in the Synthetic Features study. The Label is the Defualt status in a range by 1 to 5 years. The goal is to develop an Ensemble Learning Model on IBM Watson Studio.

According to the previous considerations about the nature of financial ratios.Traditional linear methods will not be efficient in terms of performance and preprocessing.

We need to adress also the following problems:

- Dirty Data
- Imbalanced Dataset

- Possible multicollinearity issues
- Explainability

5.3 Data Understanding

Recalling the polish study, we have the following financial ratios

ID	Description	ID	Description
X1	net profit / total assets	X33	operating expenses / short-term liabilities
X2	total liabilities / total assets	X34	operating expenses / total liabilities
X3	working capital / total assets	X35	profit on sales / total assets
X4	current assets / short-term liabilities	X36	total sales / total assets
X5	[(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365,	X37	(current assets - inventories) / long-term liabilities
X6	retained earnings / total assets	X38	constant capital / total assets
X7	EBIT / total assets	X39	profit on sales / sales
X8	book value of equity / total liabilities	X40	(current assets - inventory - receivables) / short-term liabilities
X9	sales / total assets	X41	total liabilities / ((profit on operating activities + depreciation) * (12/365))
X10	equity / total assets	X42	profit on operating activities / sales
X11	(gross profit + extraordinary items + financial expenses) / total assets	X43	rotation receivables + inventory turnover in days
X12	gross profit / short-term liabilities	X44	(receivables * 365) / sales
X13	(gross profit + depreciation) / sales	X45	net profit / inventory
X14	(gross profit + interest) / total assets	X46	(current assets - inventory) / short-term liabilities
X15	(total liabilities * 365) / (gross profit + depreciation)	X47	(inventory * 365) / cost of products sold
X16	(gross profit + depreciation) / total liabilities	X48	EBITDA (profit on operating activities - depreciation) / total assets
X17	total assets / total liabilities	X49	EBITDA (profit on operating activities - depreciation) / sales
X18	gross profit / total assets	X50	current assets / total liabilities
X19	gross profit / sales	X51	short-term liabilities / total assets
X20	(inventory * 365) / sales	X52	(short-term liabilities * 365) / cost of products sold)
X21	sales (n) / sales (n-1)	X53	equity / fixed assets
X22	profit on operating activities / total assets	X54	constant capital / fixed assets
X23	net profit / sales	X55	working capital
X24	gross profit (in 3 years) / total assets	X56	(sales - cost of products sold) / sales
X25	(equity - share capital) / total assets	X57	(current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X26	(net profit + depreciation) / total liabilities	X58	total costs /total sales
X27	profit on operating activities / financial expenses	X59	long-term liabilities / equity
X28	working capital / fixed assets	X60	sales / inventory
X29	logarithm of total assets	X61	sales / receivables
X30	(total liabilities - cash) / sales	X62	(short-term liabilities *365) / sales
X31	(gross profit + interest) / sales	X63	sales / short-term liabilities
X32	(current liabilities * 365) / cost of products sold	X64	sales / fixed assets

Some values are missing, so it will be necessary to impute or remove missing values. We will test different pipelines about :

- Imputation techniques
- Imbalance Learning
- Hyperparameter tuning

[6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats  as stats


from sklearn.impute import KNNImputer

import hdbscan



from imblearn.over_sampling import SMOTE
from imblearn.pipeline import make_pipeline, Pipeline
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split



import xgboost as xgb
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report, f1_score, precision_score, recall_score␣
 ↪,roc_auc_score,average_precision_score,roc_curve,auc
```

5.4 Data Import

[7]:
```
#Uploading default in 5 years
df_5 = pd.read_csv('Polish_data/1year.csv',low_memory=False,na_values=['?'])
df_4 = pd.read_csv('Polish_data/2year.csv',low_memory=False,na_values=['?'])
df_3 = pd.read_csv('Polish_data/3year.csv',low_memory=False,na_values=['?'])
df_2 = pd.read_csv('Polish_data/4year.csv',low_memory=False,na_values=['?'])
df_1 = pd.read_csv('Polish_data/5year.csv',low_memory=False,na_values=['?'])

# preview
df_5.head()
```

[7]:

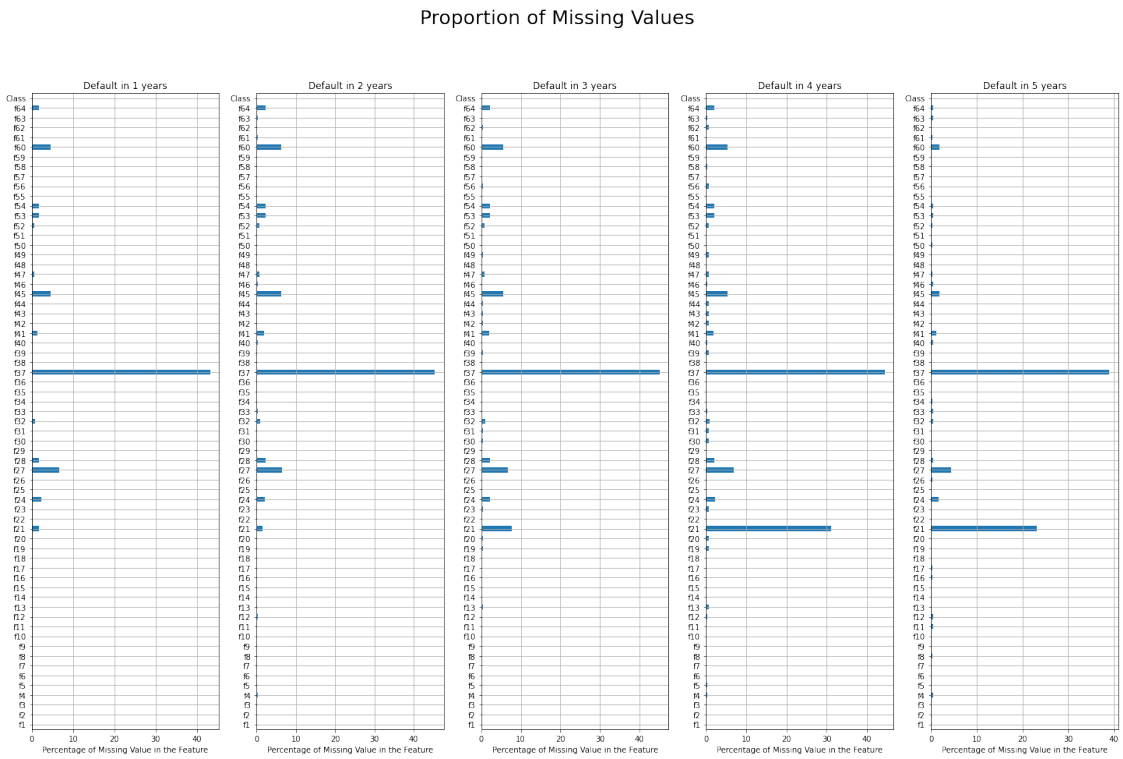
	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	\
0	0.200550	0.37951	0.39641	2.0472	32.3510	0.38825	0.249760	1.33050	
1	0.209120	0.49988	0.47225	1.9447	14.7860	0.00000	0.258340	0.99601	
2	0.248660	0.69592	0.26713	1.5548	-1.1523	0.00000	0.309060	0.43695	
3	0.081483	0.30734	0.45879	2.4928	51.9520	0.14988	0.092704	1.86610	

```
4  0.187320  0.61323  0.22960  1.4063  -7.3128  0.18732  0.187320  0.63070

    Attr9   Attr10  …    Attr56   Attr57   Attr58    Attr59  Attr60  Attr61  \
0  1.1389  0.50494  …  0.121960  0.39718  0.87804  0.001924  8.4160  5.1372
1  1.6996  0.49788  …  0.121300  0.42002  0.85300  0.000000  4.1486  3.2732
2  1.3090  0.30408  …  0.241140  0.81774  0.76599  0.694840  4.9909  3.9510
3  1.0571  0.57353  …  0.054015  0.14207  0.94598  0.000000  4.5746  3.6147
4  1.1559  0.38677  …  0.134850  0.48431  0.86515  0.124440  6.3985  4.3158

    Attr62  Attr63   Attr64  class
0   82.658  4.4158   7.4277      0
1  107.350  3.4000  60.9870      0
2  134.270  2.7185   5.2078      0
3   86.435  4.2228   5.5497      0
4  127.210  2.8692   7.8980      0

[5 rows x 65 columns]
```

5.5 Data Cleaning and Preparation

[8]:
```
# Rename Features
df_1.columns  = ['f'+str(i) for i in range(1,66)]
df_2.columns  = ['f'+str(i) for i in range(1,66)]
df_3.columns  = ['f'+str(i) for i in range(1,66)]
df_4.columns  = ['f'+str(i) for i in range(1,66)]
df_5.columns  = ['f'+str(i) for i in range(1,66)]

# Rename Response
df_1 = df_1.rename({'f65': 'Class'}, axis=1)
df_2 = df_2.rename({'f65': 'Class'}, axis=1)
df_3 = df_3.rename({'f65': 'Class'}, axis=1)
df_4 = df_4.rename({'f65': 'Class'}, axis=1)
df_5 = df_5.rename({'f65': 'Class'}, axis=1)
```

[9]:
```
# Plot multiple missing value proportion
fig = plt.figure(figsize = (25,15))
fig.suptitle('Proportion of Missing Values',fontsize = 25)

for y,df in enumerate([df_1,df_2,df_3,df_4,df_5],start = 1):
    ax = fig.add_subplot(1,5,y)
    (df.isna().sum()/df.shape[0]*100).plot(kind = 'barh',ax = ax).
 ↪set_title('Default in '+str(y)+' years')
    ax.set_xlabel('Percentage of Missing Value in the Feature')
    ax.grid(True)
```

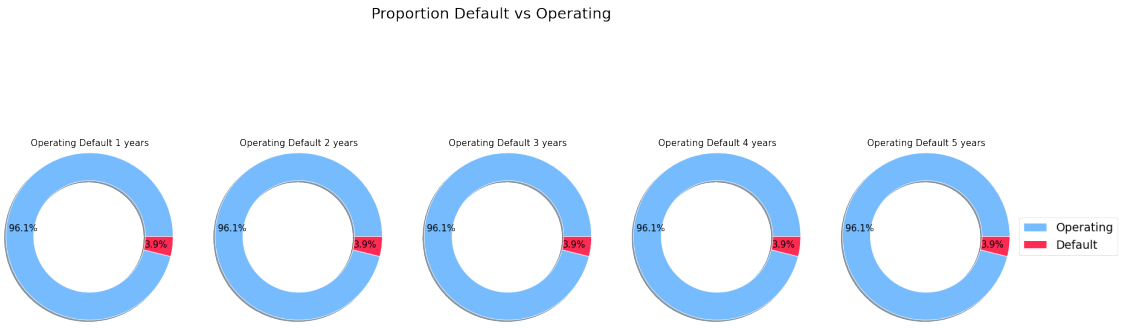


All features have at least one missing value, but some features are missing in a systematic way. There is some pattern in the distribution of missing values. For example the variable 21 missing rate increases with the period of default. X21 represents the difference in sales. It might be very important in order to evaluate the operational management and so how business is going in general.

5.5.1 Response Distribution

[14]:
```
#Create figure
fig1 = plt.figure(figsize = (30,25))
fig1.suptitle('Proportion Default vs Operating',fontsize = 25)

# Plot Proportion
for y,df in enumerate([df_1,df_2,df_3,df_4,df_5],start = 1):
    ax1 = fig1.add_subplot(3,5,y)
    ax1.set_title('Operating Default '+str(y)+' years',fontsize = 15)
    pieplot,_,_ = ax1.pie(df.Class.value_counts().values ,radius = 1.2,
                          colors=['xkcd:sky blue','xkcd:reddish pink'],
                          autopct="%0.1f%%",
                          shadow=True,
                          wedgeprops=dict(width=0.4, edgecolor='white'),
                          pctdistance=0.8,textprops={'fontsize': 15})
```

```
ax1.legend(['Operating','Default'], loc="center left",bbox_to_anchor=(1, 0, 0.
↪5, 1),fontsize = 19);

plt.show()
```



According to the previous statements about the various issues of Bankruptcy prediction , there's an extreme difference in the operating and default proportion. As such the models response will suffer of proportionary bias. To adress the issue there are several techniques:

- Undersampling
- Oversampling
- Synthetic Sampling (in this case SMOTE)

Their pro and con will be discussed in the upcoming steps.

5.5.2 Imputation

In statistics, imputation is the process of replacing missing data with substituted values.

Some variables might not be "missing completely at random", so in order to preserve their predictive power there are some techniques. However it must be stressed that trying to impute variables with high missing rate will indeed make the analysis more discretional,since the relative preference of imputation techinques will affect the final result.

In this precise case the variable X37 should be eliminated,since its missing rate is almost 50%, every imputation technique will make the model too much discretional.

On the other hand for X21,there is a good chance that imputation techinques will help to preserve the predictive power of the feature.

There are many imputation techniques,the simplest is the mean imputation. In this case however using the mean will do more harm than good. The reasons are :

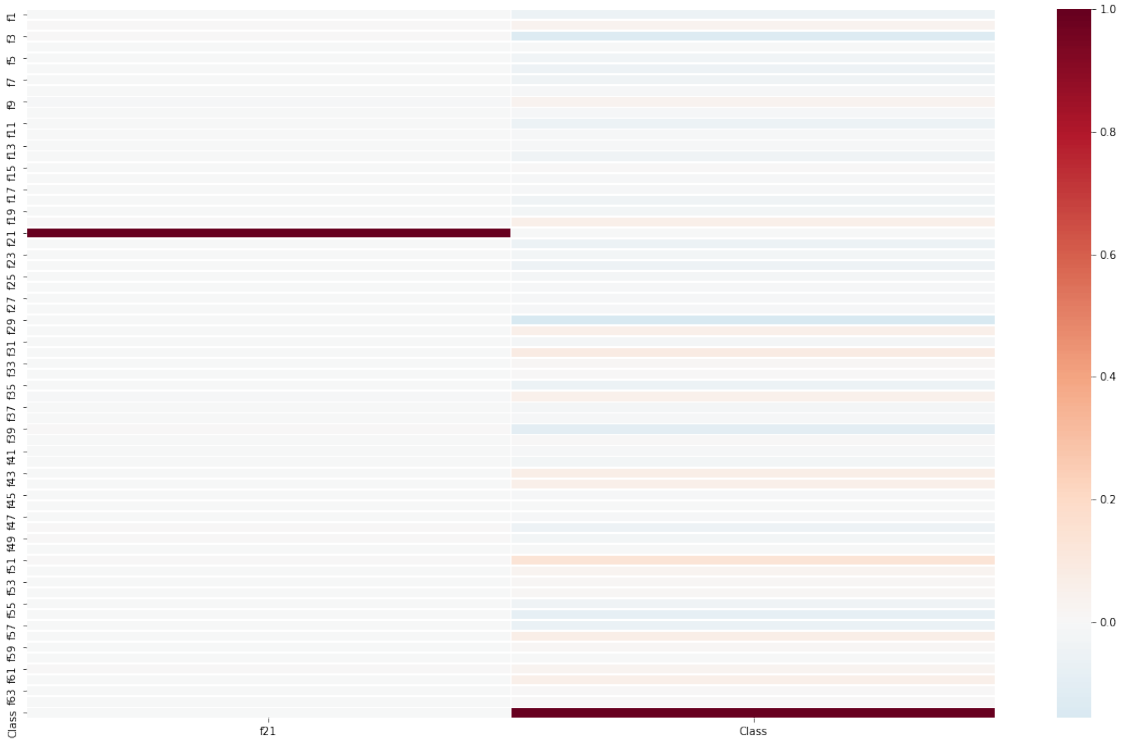
- The imbalance porportion will make a lot of confusion among variables and will make look default companies more competitive
- Some variables are indeed correlated with each other and so in some dimensions the imputed values can be discording

A more robust way to impute values is by using linear regression or machine learning techinques like Random Forest Imputation or KNN imputation. The more variables are correlated with the

response variable ,the more useful the linear regression will be. Before doing any imputation techinque it is important to visualize data and study the various correlations in a balanced scenario.

Unbalanced Correlation

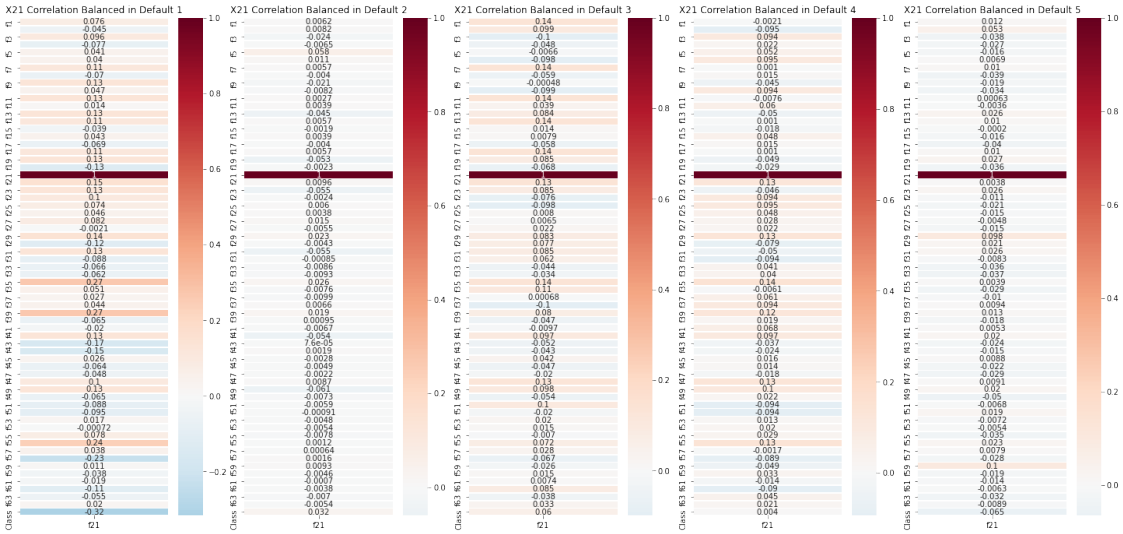
[11]:
```
plt.figure(figsize = (20,12) )
sns.heatmap(df_1.corr()[['f21','Class']], center=0 ,linewidths=.8,cmap =␣
↪'RdBu_r');
```



Correlation in balanced scenario Given the random sampling of the operating companies that must match the number of failures , a more fair evaluation of correlation can be given averaging the n sampled folders.

[12]:
```
def sample_correlation(df):
    # Downsampling
    ND = df.loc[df['Class'] == 0 , :].sample(n = df.loc[df['Class'] == 1 , :].
↪shape[0])
    D = df.loc[df['Class'] == 1 , :]
    ND_D_Balanced = pd.concat([ND,D], 0 )

    return ND_D_Balanced.corr()

def corr_folder(df):
    # Correlation across 10 random folds
```

```
    Corr_folds = pd.DataFrame( pd.concat([sample_correlation(df)['f21'] for i␣
 ↪in range(3)],axis = 1).mean(axis = 1) ,columns = ['f21'])
    return Corr_folds

f = plt.figure(figsize = (25,40) )

for y,df in enumerate([df_1,df_2,df_3,df_4,df_5],start = 1):
    ax2 = f.add_subplot(3,5,y)
    # Correlation Heatmap
    sns.heatmap(corr_folder(df), center=0 ,linewidths=.8,annot = True,cmap =␣
 ↪'RdBu_r' , ax = ax2).set_title('X21 Correlation Balanced in Default␣
 ↪'+str(y));
```



KNN imputation The KNN algorithm is a supervised machine learning method that is often used in Imputation , the general idea is that similar classes shares some common region in the feature space , and so it is reasonable to state that two instances with a low distance or high belong to the same class. There are different definition of distance , the most common used is the Euclidean Distance. The name "K" stands for a parameter K of the algorithm that consider the K nearest-neighbor in order to fill the missing values. The disadvantage of this approach is the so called curse of dimensionality. In a large dimension space the KNN algorithm might be unstable. According to a previous consideration,imputation techniques make the model discretional and so they must be tuned in order to maximize the benefits of such discretional method.

[13]:
```
#KNN imputer

def Imputation(df,k = 100):

    KNN_imputer = KNNImputer(n_neighbors=k)
```

```
    df_KNNimputed = pd.DataFrame( KNN_imputer.fit_transform(df) , columns = df.
 ↪columns)

    return df_KNNimputed

for n,df in enumerate([df_1,df_2,df_3,df_4,df_5],start = 1):
    print('Missing values in Default after '+str(n) + ' years :')
    print(df.isna().sum())
    print('#'*50)

print('='*50)
print('Imputation complete!')
print('='*50)

print('Missing values in Default after '+str(n) + ' years :')
df_1 = Imputation(df)
df_2 = Imputation(df)
df_3 = Imputation(df)
df_4 = Imputation(df)
df_5 = Imputation(df)

print('#'*50)
for n,df in enumerate([df_1,df_2,df_3,df_4,df_5],start = 1):
    print('Missing values after KNN Imputation in Default after '+str(n) + '␣
 ↪years :')
    print(df.isna().sum())
    print('#'*50)
```

```
Missing values in Default after 1 years :
f1          3
f2          3
f3          3
f4         21
f5         11
         ...
f61        15
f62         0
f63        21
f64       107
Class       0
Length: 65, dtype: int64
##################################################
Missing values in Default after 2 years :
f1          1
f2          1
f3          1
f4         43
```

```
f5        21
         …
f61       32
f62       21
f63       43
f64      231
Class      0
Length: 65, dtype: int64
##################################################
Missing values in Default after 3 years :
f1         0
f2         0
f3         0
f4        18
f5        25
         …
f61       17
f62       43
f63       18
f64      228
Class      0
Length: 65, dtype: int64
##################################################
Missing values in Default after 4 years :
f1         1
f2         1
f3         1
f4        22
f5        24
         …
f61       16
f62       63
f63       22
f64      212
Class      0
Length: 65, dtype: int64
##################################################
Missing values in Default after 5 years :
f1         3
f2         3
f3         3
f4        30
f5         8
          ..
f61       22
f62        0
f63       30
f64       34
```

```
Class     0
Length: 65, dtype: int64
##################################################
==================================================
Imputation complete!
==================================================
Missing values in Default after 5 years :
##################################################
Missing values after KNN Imputation in Default after 1 years :
f1       0
f2       0
f3       0
f4       0
f5       0
        ..
f61      0
f62      0
f63      0
f64      0
Class    0
Length: 65, dtype: int64
##################################################
Missing values after KNN Imputation in Default after 2 years :
f1       0
f2       0
f3       0
f4       0
f5       0
        ..
f61      0
f62      0
f63      0
f64      0
Class    0
Length: 65, dtype: int64
##################################################
Missing values after KNN Imputation in Default after 3 years :
f1       0
f2       0
f3       0
f4       0
f5       0
        ..
f61      0
f62      0
f63      0
f64      0
Class    0
```

```
Length: 65, dtype: int64
################################################
Missing values after KNN Imputation in Default after 4 years :
f1       0
f2       0
f3       0
f4       0
f5       0
        ..
f61      0
f62      0
f63      0
f64      0
Class    0
Length: 65, dtype: int64
################################################
Missing values after KNN Imputation in Default after 5 years :
f1       0
f2       0
f3       0
f4       0
f5       0
        ..
f61      0
f62      0
f63      0
f64      0
Class    0
Length: 65, dtype: int64
################################################
```

5.5.3 Balancing Data

It's once again stated the importance of analyzing data in a balanced scenario , in order to create a unbiased model, the possible techinques are:

- Downsampling
- Oversampling
- Other (SMOTE)

Pro and cons of each:

Downsampling is the reduction of the majority class in order to match the number of values of the minority class. It is the most effective and correct method but is almost impossible to find a real situation where this technique is suitable. In fact most of the times data are dirty and imbalanced. In such case it will be extremly difficult to relate on just 6% (3% failures matched with other 3% operating) of a Dataset. Plus each time we sample data it is important to understand that the the final conclusion might not be significant.

So Downsampling is :

- Efficient
- Non discretional
- Impossible to apply in real situation

Oversampling is the opposite concept of Downsampling. The minority class instances are repeated until there's no difference between the classes. This approach is very common but it is very unefficient in terms of information gained by "new data". In other words there's just more computational expenses for each new replicated data. It is also very prone to overfitting.

So Oversampling is :

- Effective
- Increase the computational complexity
- Leading to overfitting

Another way to deal with the imbalance problem is generating a set of synthetic data that should belong to the same feature space. The advantage is that there are no copies in the dataset and that there are no useless data. This approach is named SMOTE.

Synthetic Minority Oversampling TEchnique (SMOTE) is a very popular oversampling method that was proposed to improve random oversampling and its behavior on high-dimensional data.

SMOTE, as well as many other methods, applies the k-nearest neighbor approach during the sample generation phase. In order to generate an artificial instance x_{gen},the kNN approach randomly selects another minority class sample $x^{'}$ from the k-nearest neighbors of x.

Then x_{gen} is generated by using a linear interpolation of x and x_n which can be expressed as

$$x_{gen} = x + \alpha \cdot (x\prime - x) \tag{38}$$

Where a is a random number in the [0, 1] interval . Thus x_{gen} lies in the line segment between x and $x^{'}$. The limitation of SMOTE is that in a noisy sceneraio, even if k in the knn framework is small, the new instances might be unrealistic. [24].

5.6 Model

It has been discussed so far the main reason why an Ensemble framework is more convinient than other traditional methods. In order to create a XGBoost model the steps are the following:

- 1. Extract a Train set ,a validation set and also a test set of data
- 2. Apply any required modification to the train set in order to create a fair and representative set of data for the model
- 3. Test the model with Randomized Grid Search Cross validation

1. Extracting Train Validation and Test is nothing more than a random shuffled split of data , we relay on the quality of Knn Imputer to obtain a reasonable fill of missing values and on Cross Validation Framework.
2. In order to creata a fair model it is important to create a balanced scenario using SMOTE and Cost Sensitive parameters that will enchance the importance of the minority class(Bankrupt) . It is not allowed to upsample or modify the test set , so the pipeline will be :

- Sample data
- Using SMOTE on the Training fold

3. To optimize the parameters research we will use a Randomized Grid Search Cross Validation.The main advantage of the Randomized Grid is that we do not need to test every signle combination of parameters , using n iteration of the k parameters will lead to a more efficient hyperparameter tuning in terms of performance and computational resources.

The parameter that will be tested are :

For Imputation:

- K neighbors

For SMOTE:

- Sampling strategy
- Number of K

For XGB:

- Learning Rate
- Max depth of Trees
- Min Child Weight
- Regularization Gamma
- Columns samples in Decision Trees
- Scale pos weight for Imbalanced Cost sensitive Optimization

6 Bankruptcy Prediction Model on IBM Watson Studio

[ ]:
```
# @hidden_cell
# The project token is an authorization token that is used to access project␣
 ↪resources like data sources, connections, and used by platform APIs.
from project_lib import Project
project = Project(project_id='b05d3f54-9010-4d42-a871-17b01d5e241a',␣
 ↪project_access_token='p-f99253bba3d03dc0c44bc63d2759bc25bcea1de1')
pc = project.project_context
```

[2]:
```
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# accesses a file in your IBM Cloud Object Storage
client_4a5bfdc511d34c0ba54d8d626dcd8751 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='____',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
```

```
    endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')
```

[ ]:
```
#! pip install scikit-learn==0.22.
#! pip install -U ibm-watson-machine-learning
#! pip install imbalanced-learn
```

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats  as stats

import sklearn
from sklearn.impute import KNNImputer

from imblearn.over_sampling import SMOTE
from imblearn.pipeline import make_pipeline, Pipeline
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split



from xgboost.sklearn import XGBClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,␣
↪classification_report, f1_score, precision_score, recall_score␣
↪,roc_auc_score,average_precision_score,roc_curve,auc
```

[3]:
```
#from sklearn.preprocessing import FunctionTransformer

def Data_Import(key,path = 'Polish_data/5year.csv',test_sample_class = 25 ):
    try:
        # Read csv
        df = pd.read_csv(path,low_memory=False,na_values=['?'])

        # Rename Features
        df.columns  = ['f'+str(i) for i in range(1,66)]
        # Rename Response
        df = df.rename({'f65': 'Class'}, axis=1)

        # Create Train fold  and hide Test fold

        df_dropNA = df.dropna() #Drop Nan
        Test_Operating  = df_dropNA[df_dropNA.Class == 0].
↪sample(n=test_sample_class)
        Test_Default    = df_dropNA[df_dropNA.Class == 1].
↪sample(n=test_sample_class)
```

```
        Test = pd.concat([Test_Default,Test_Operating] , 0) # Create the Hide␣
↪pure Test Data

        Train = df.loc[[ i for i in df.index if i not in  Test.index], :] #␣
↪Remove the index of Test and create the Train Block from original df

        Test.sample(frac=1) # Shuffle randomly the Test Block

        return Train , Test

    except:


        body = client_4a5bfdc511d34c0ba54d8d626dcd8751.
↪get_object(Bucket='ensemblemachinelearningmethodsfor-donotdelete-pr-rk2ik14r8qxhyu',Key=key
        # add missing __iter__ method, so pandas accepts body as file-like␣
↪object
        if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType(␣
↪__iter__, body )

        df = pd.read_csv(body,na_values=['?'])
        # Rename Features
        df.columns  = ['f'+str(i) for i in range(1,66)]
        # Rename Response
        df = df.rename({'f65': 'Class'}, axis=1)

        # Create Train fold  and hide Test fold

        df_dropNA = df.dropna() #Drop Nan
        Test_Operating  = df_dropNA[df_dropNA.Class == 0].
↪sample(n=test_sample_class)
        Test_Default    = df_dropNA[df_dropNA.Class == 1].
↪sample(n=test_sample_class)


        Test = pd.concat([Test_Default,Test_Operating] , 0) # Create the Hide␣
↪pure Test Data

        Train = df.loc[[ i for i in df.index if i not in  Test.index], :] #␣
↪Remove the index of Test and create the Train Block from original df

        Test.sample(frac=1) # Shuffle randomly the Test Block

        return Train , Test
```

```
def Model_Pipeline(df,target,pipeline,param_dist,n_iter = 10,scoring =␣
↪'roc_auc',n_splits = 10):

    X = df.drop(target,1)
    y = df[target]

    GridS = RandomizedSearchCV(pipeline,
                               param_distributions = param_dist,
                               cv =␣
↪StratifiedKFold(n_splits=n_splits,shuffle=True),
                               n_iter = n_iter,
                               scoring = scoring,
                               error_score = 0,
                               verbose = 0,
                               n_jobs = -1)
    GridS.fit(X, y)


    return GridS , GridS.best_estimator_


def Evaluation_Pipeline(X_test,y_test,model,w = 20,h = 25):

    predicted = model.predict(X_test)

    fig = plt.figure(figsize = (h,w))
    ax  = fig.add_subplot(2,2,1)
    ax1 = fig.add_subplot(2,2,2)


    predicted = model.predict(X_test)
    print(classification_report(y_test,predicted,))


    feat_importances = pd.Series(model.feature_importances_, index=X_test.
↪columns)
    feat_importances.nlargest(20).plot(kind='barh',ax = ax1)

    ax1.set_title('Top 20 Feature Importance')

    sns.heatmap(confusion_matrix(y_test,predicted),annot = True, cmap='Blues'
                ,ax = ax ,annot_kws={"fontsize":13}).set_title('Confusion␣
↪Matrix');

    ax.set_ylabel('True Label ')
```

```
    ax.set_xlabel('Predicted Label')

    plt.grid()

    print('='*50)
    print('Pred Count:',
            pd.Series(predicted).value_counts(), sep = '\n' )

    print('AUC Test :', round( roc_auc_score(y_test, predicted),2) )


    print('='*50)


def Final_Pipeline(key ,path_csv,year):
    Train , Test  = Data_Import(path = path_csv ,key = key)

    X_train  = Train.drop('Class',1)
    y_train  = Train['Class']

    X_test = Test.drop('Class',1)
    y_test = Test['Class']

    #Initialize Functions for Pipeline
    SM  = SMOTE()
    Clf = XGBClassifier()
    Kf  = StratifiedKFold(n_splits=10,shuffle=True)
    Imp = KNNImputer()

    #Grid of possible combinations of parameters
    param_dist = {

                  "Imputation__n_neighbors" : [30,50,100],

                  "SMOTE__sampling_strategy": [1,0.9,0.7],
                  "SMOTE__k_neighbors"      : [3,5,10],

                  "Clf__learning_rate"      : [0.01, 0.1, 0.3] ,
                  "Clf__max_depth"          : [ 3, 5, 6],
                  'Clf__n_estimators'       : [300,800,1000],
                  "Clf__min_child_weight"   : [ 2,5],
                  'Clf__scale_pos_weight'   : [1,5,8,13,15],
                  "Clf__gamma"              : [ 0.0, 0.1, 0.2],
                  "Clf__colsample_bytree"   : [ 0.5,0.7,1 ]
    }

    # Pipeline Initialization
```

```
    pipeline = Pipeline([
              ('Imputation',Imp),
             #('SMOTE', SM),
              ('Clf', Clf)
        ])

    # Running the Full Pipeline
    Grid_Default_1year ,Model_Default_1year =␣
↪Model_Pipeline(Train,target='Class',pipeline=pipeline,param_dist=param_dist,n_iter␣
↪= 15,scoring = 'roc_auc')

    AUC_mean = Grid_Default_1year.cv_results_['mean_test_score'].mean().round(2)

    AUC_std  = Grid_Default_1year.cv_results_['std_test_score'].mean().round(2)

    AUC_best = Grid_Default_1year.best_score_.round(2)

    Result = pd.DataFrame( pd.Series( [AUC_best,AUC_mean,AUC_std], index  =␣
↪['AUC_best','AUC_mean','AUC_std'] ,name = 'XGB_'+str(year)+'Default')  ).T

    return Model_Default_1year, Result ,Grid_Default_1year
```

[7]:
```
Model_Default_1year, Result_1yD ,Grid_Default_1year = Final_Pipeline(path_csv =␣
↪'Polish_data/5year.csv',key = '5year.csv',year=1)
Model_Default_2year, Result_2yD ,Grid_Default_2year = Final_Pipeline(path_csv =␣
↪'Polish_data/4year.csv',key = '4year.csv',year=2)
Model_Default_3year, Result_3yD ,Grid_Default_3year = Final_Pipeline(path_csv =␣
↪'Polish_data/3year.csv',key = '3year.csv',year=3)
Model_Default_4year, Result_4yD ,Grid_Default_4year = Final_Pipeline(path_csv =␣
↪'Polish_data/2year.csv',key = '2year.csv',year=4)
Model_Default_5year, Result_5yD ,Grid_Default_5year = Final_Pipeline(path_csv =␣
↪'Polish_data/1year.csv',key = '1year.csv',year=5)
```

7 Deployment

Deploying a machine learning model simply means to integrate a machine learning model into an existing production environment, the model is already trained and it can take in an input and return an output. The purpose of the deployment is to make the model available to others. Model deployment is closely related to ML systems architecture, which refers to the arrangement and interactions of software components within a system to achieve a predefined goal. The proposed Environment for deployment is IBM Watson Studio.

[21]:
```
from ibm_watson_machine_learning import APIClient
```

[5]:
```
#https://cloud.ibm.com/iam/apikeys
secret_api_key_WML= '-----'
```

```
location = 'eu-gb'
wml_credentials = {
                    "apikey": secret_api_key_WML,
                    "url": 'https://' + location + '.ml.cloud.ibm.com'}

client = APIClient(wml_credentials)

print(client.version)
```

```
1.0.10
```

[8]:
```
#Create deployment space
space_id = '6920c261-5c60-416f-a090-21a83d0c41ff'
client.set.default_space(space_id)




# cleanup of previously created models and deployments
client.repository.list_models()
client.deployments.list()

#client.repository.delete('GUID OF stored model')
# client.deployments.delete('GUID of deployed model')
```

```
--  ----  -------  ----
ID  NAME  CREATED  TYPE
--  ----  -------  ----
----  ----  -----  -------
GUID  NAME  STATE  CREATED
----  ----  -----  -------
```

[9]:
```
n = str(1)
model_ = Model_Default_1year['Clf']
```

[10]:
```
Model_Default_1year['Clf']
```

[10]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bytree=0.7, gamma=0.2, learning_rate=0.1,
              max_delta_step=0, max_depth=6, min_child_weight=2, missing=None,
              n_estimators=800, n_jobs=1, nthread=None,
              objective='binary:logistic', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=13, seed=None, silent=True,
              subsample=1)
```

[11]:
```
software_spec_uid = client.software_specifications.get_uid_by_name('xgboost_0.
↪90-py3.6')
```

```
# store the model in WML
meta_props={client.repository.ModelMetaNames.NAME: "Model_"+n+'_Def',
            client.repository.ModelMetaNames.TYPE: "scikit-learn_"+sklearn.
 ↪__version__,
            client.repository.ModelMetaNames.SOFTWARE_SPEC_UID :␣
 ↪software_spec_uid,
              }

published_model = client.repository.store_model(model=model_,␣
 ↪meta_props=meta_props)


# new list of models
client.repository.list_models()

# get UID of our just stored model
model_uid = client.repository.get_model_uid(published_model)
print("Model id: {}".format(model_uid))
```

```
------------------------------------  -----------  ------------------------
-----------------
ID                                    NAME         CREATED
TYPE
04ded5a9-61cd-4754-a86c-13708307854e  Model_1_Def  2020-09-11T10:24:33.002Z
scikit-learn_0.22
------------------------------------  -----------  ------------------------
-----------------
Model id: 04ded5a9-61cd-4754-a86c-13708307854e
```

[12]:
```
# Create the deployment.
meta_props = {
    client.deployments.ConfigurationMetaNames.NAME:␣
 ↪"Bankruptcy_Model_1year_Default",
    client.deployments.ConfigurationMetaNames.ONLINE: {}
}




deployment_details = client.deployments.create(model_uid,meta_props)
```

```
#########################################################################################
#######

Synchronous deployment creation for uid: '04ded5a9-61cd-4754-a86c-13708307854e'
started

#########################################################################################
#######


initializing
ready


------------------------------------------------------------------------------------
----------------
Successfully finished deployment creation,
deployment_uid='d70f3cff-37e9-4940-8bbf-10193daf77a2'
------------------------------------------------------------------------------------
----------------
```

[17]:
```
# List the deployments.
client.deployments.list()

#Show deployments details
deployment_uid = client.deployments.get_uid(deployment_details)
client.deployments.get_details(deployment_uid)
```

```
------------------------------------  ------------------------------  -----
------------------------
GUID                                  NAME                            STATE
CREATED
d70f3cff-37e9-4940-8bbf-10193daf77a2  Bankruptcy_Model_1year_Default  ready
2020-09-11T10:24:37.701Z
------------------------------------  ------------------------------  -----
------------------------
```

[17]:
```
{'entity': {'asset': {'id': '04ded5a9-61cd-4754-a86c-13708307854e'},
  'custom': {},
  'hardware_spec': {'id': 'Not_Applicable', 'name': 'S', 'num_nodes': 1},
  'name': 'Bankruptcy_Model_1year_Default',
  'online': {},
  'space_id': '6920c261-5c60-416f-a090-21a83d0c41ff',
  'status': {'online_url': {'url': 'https://eu-gb.ml.cloud.ibm.com/ml/v4/deploym
ents/d70f3cff-37e9-4940-8bbf-10193daf77a2/predictions'},
```

```
   'state': 'ready'}},
 'metadata': {'created_at': '2020-09-11T10:24:37.701Z',
  'id': 'd70f3cff-37e9-4940-8bbf-10193daf77a2',
  'modified_at': '2020-09-11T10:24:37.701Z',
  'name': 'Bankruptcy_Model_1year_Default',
  'owner': 'IBMid-5500015JBB',
  'space_id': '6920c261-5c60-416f-a090-21a83d0c41ff'}}
```

[94]:
```
#Get scoring url
deployment_id = client.deployments.get_id(deployment_details)

# Prepare scoring payload.
payload_scoring = {client.deployments.ScoringMetaNames.INPUT_DATA:
    [
        {
        'values': [[ 7.5494e-02, 7.7537e-01, 5.2367e-02, 1.3347e+00, 1.4025e+00,
                     5.9421e-02, 9.2271e-02, 2.8970e-01, 7.7555e-01, 2.2463e-01,
                     9.2718e-02, 5.8981e-01, 1.2156e-01, 9.2271e-02, 3.0019e+03,
                     1.2159e-01, 1.2897e+00, 9.2271e-02, 1.1897e-01, 2.3118e+01,
                     2.6368e+00, 9.1672e-02, 9.7342e-02, 1.1442e-01, 2.2133e-01,
                     9.9951e-02, 2.0482e+02, 6.6187e-02, 4.2225e+00, 9.6407e-01,
                     1.1897e-01, 8.0881e+01, 4.5401e+00, 9.1602e-01, 6.9503e-02,
                     8.0301e-01, 5.0572e+00, 2.5620e-01, 8.9617e-02, 1.8007e-01,
                     2.7590e-01, 1.1820e-01, 8.5014e+01, 6.1895e+01, 1.5369e+00,
                     4.9406e-01, 2.5244e+01, 8.9666e-02, 1.1562e-01, 2.6930e-01,
                     1.5644e-01, 2.2026e-01, 2.8391e-01, 3.2382e-01, 8.7400e+02,
                     8.9694e-02, 3.3609e-01, 8.8450e-01, 1.4057e-01, 1.5788e+01,
                     5.8970e+00, 7.3626e+01, 4.9575e+00, 9.8023e-01]]
        }
   ]
}
print(payload_scoring)
```

```
{'input_data': [{'values': [[0.075494, 0.77537, 0.052367, 1.3347, 1.4025,
0.059421, 0.092271, 0.2897, 0.77555, 0.22463, 0.092718, 0.58981, 0.12156,
0.092271, 3001.9, 0.12159, 1.2897, 0.092271, 0.11897, 23.118, 2.6368, 0.091672,
0.097342, 0.11442, 0.22133, 0.099951, 204.82, 0.066187, 4.2225, 0.96407,
0.11897, 80.881, 4.5401, 0.91602, 0.069503, 0.80301, 5.0572, 0.2562, 0.089617,
0.18007, 0.2759, 0.1182, 85.014, 61.895, 1.5369, 0.49406, 25.244, 0.089666,
0.11562, 0.2693, 0.15644, 0.22026, 0.28391, 0.32382, 874.0, 0.089694, 0.33609,
0.8845, 0.14057, 15.788, 5.897, 73.626, 4.9575, 0.98023]]}]}
```

[95]:
```
# Perform prediction and display the result.
response_scoring = client.deployments.score(deployment_id, payload_scoring)
print(response_scoring)
```

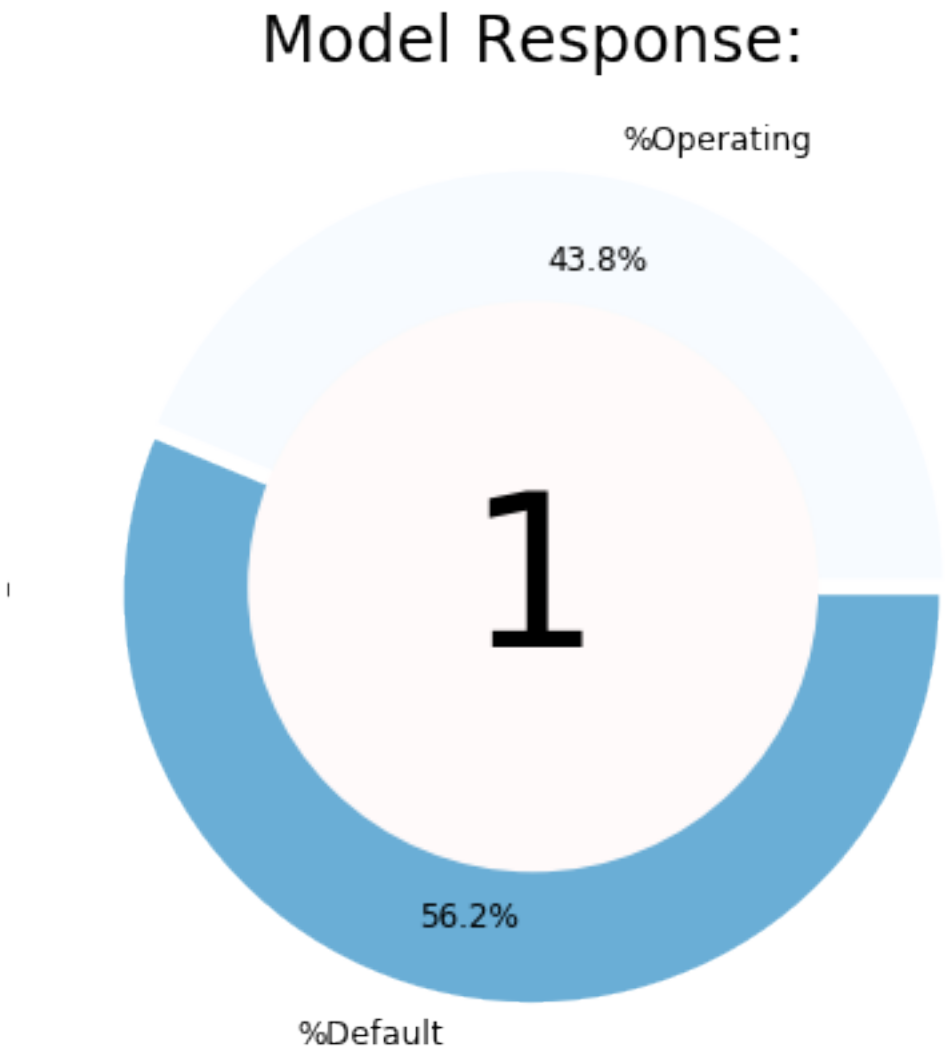
```
predictions = client.deployments.score(deployment_id, payload_scoring)
```

```
{'predictions': [{'fields': ['prediction', 'probability'], 'values': [[1,
[0.4376674294471741, 0.5623325705528259]]]}]}
```

[96]:
```
predictions['predictions'][0]['values'][0][1]
```

[96]:
```
[0.4376674294471741, 0.5623325705528259]
```

[122]:
```
from matplotlib import cm
import numpy as np


centre_circle = plt.Circle((0,0),0.70,fc='snow')
fig = plt.gcf()

fig.gca().add_artist(centre_circle)

plt.title('Model Response:',fontsize = 24)

cs=cm.Blues(np.arange(2)/2)
pd.Series(predictions['predictions'][0]['values'][0][1] ,
          index=['%Operating','%Default']).plot(kind  =␣
 ↪'pie',startangle=0,fontsize = 12,
                                                pctdistance=0.8,autopct='%1.
 ↪1f%%',
                                                colors=cs,figsize = (20,6),
                                                explode = tuple([0.02 for i in␣
 ↪range(0,2)]))


plt.ylabel('_');
#ax1.axis('equal')
plt.tight_layout()

label = plt.annotate(str(predictions['predictions'][0]['values'][0][0]),
                     xy=(0, 0),
                     fontsize=80,
                     verticalalignment='center',
                     horizontalalignment='center',)
plt.show()
```



8 Conclusions

The conclusion of the study shows the following results considering:

- LDA (Linear discriminant Analysis)
- MLP (Multilayer perceptron)
- JRip (decision rules)
- J48 (decision tree)
- LR (Logistic Regression)
- CLR (Cost sensitive LR)
- AB (AdaBoost)
- AC (Cost sensitive AdaBoost)
- SVM (Support Vector Machine)

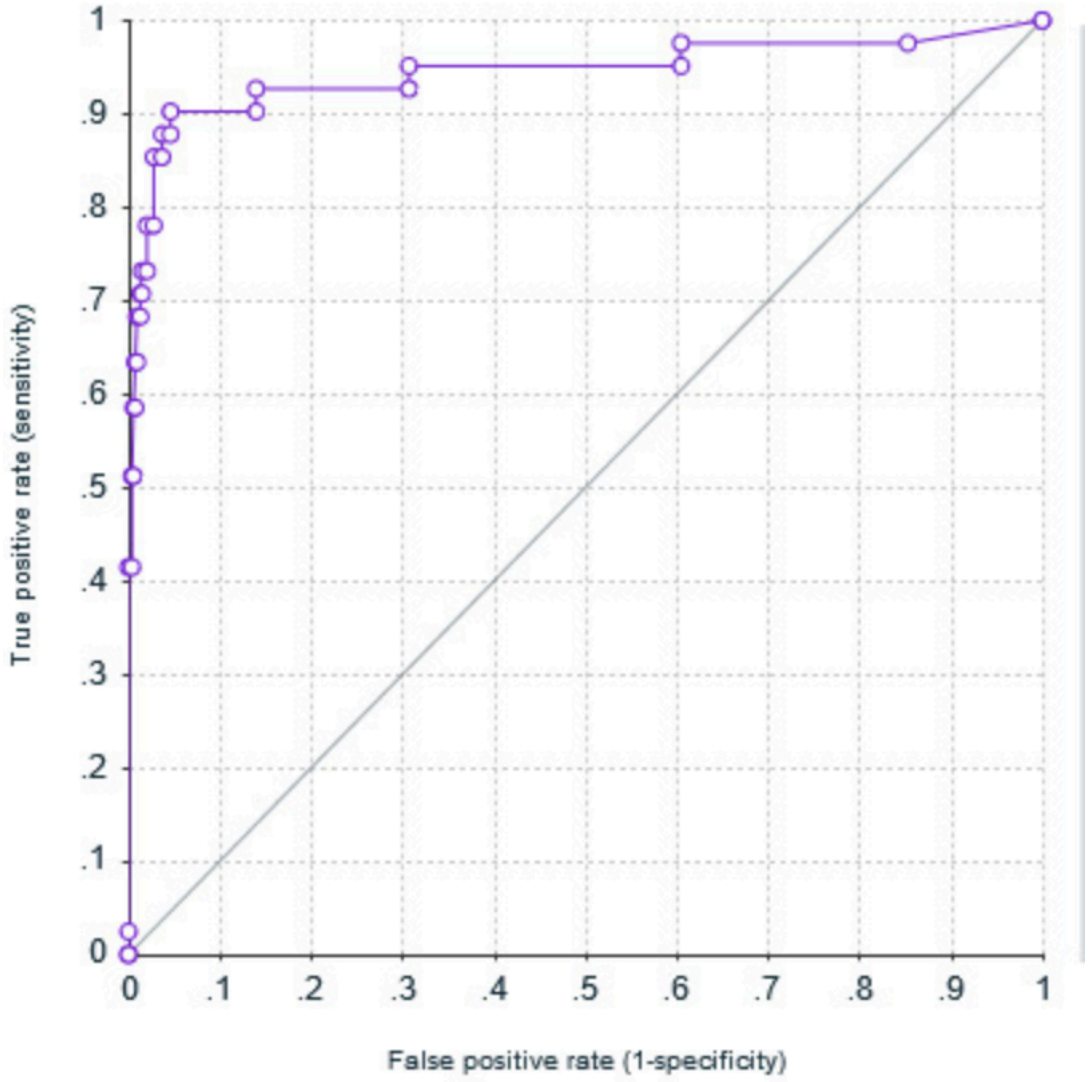
- RF (Random Forest)
- CSVM (Cost sensitive SVM)
- XBG (Extreme Gradient Boostin)
- EXGB (Synthetic Feature XGB)

Due to the imbalanced nature of training data, the Area Under ROC Curve (AUC) curve is used as criterion to evaluate the quality of the models.

For each of the considered periods and examined models is presented the mean (MN) and standard deviation (STD) for AUC measure that was calculated basing on 10 cross validation folds.

	1stYear		2ndYear		3rdYear		4thYear		5thYear	
	MN	**STD**	**MN**	**STD**	**MN**	**STD**	**MN**	**STD**	**MN**	**STD**
LDA	.639	.083	.660	.037	.688	.030	.714	.063	.796	.041
MLP	.543	.042	.514	.042	.548	.041	.596	.049	.699	.059
JRip	.523	.030	.540	.025	.535	.022	.538	.026	.654	.049
CJRip	.745	.112	.774	.073	.804	.054	.799	.070	.778	.035
J48	.717	.059	.653	.068	.701	.062	.691	.076	.761	.049
CJ48	.658	.047	.652	.047	.618	.061	.611	.025	.719	.046
LR	.620	.065	.513	.042	.500	.000	.500	.000	.632	.119
CLR	.704	.065	.671	.032	.714	.034	.724	.041	.821	.037
AB	.916	.020	.850	.029	.861	.023	.885	.031	.925	.026
AC	.916	.023	.849	.022	.859	.022	.886	.015	.928	.023
SVM	.502	.006	.502	.006	.500	.000	.500	.000	.505	.006
CSVM	.578	.040	.517	.064	.614	.040	.615	.034	.716	.039
RF	.851	.044	.842	.028	.831	.031	.848	.027	.898	.035
XGB	.945	.033	.917	.027	.922	.025	.935	.024	.951	.024
XGBE	.953	.024	.941	.019	.929	.049	.940	.027	.954	.018
EXGB	**.959**	.018	**.944**	.021	**.940**	.032	**.941**	.025	**.955**	.019

Here the ROC curve for the Deployed Model:



The AUC (Area Under the Curve) is a metric used in classification problems. A good classifier is able to maximize this Area that has a max value of 1 and a baseline level of 0.5.

The ROC curve is a way to summarize a different set of results in a classification problem. It indicates, for each possible threshold of a classifier, the corresponding schema of true positive rate and false-positive rate. A maximized AUC is created when the highest point of the curve is close to the upper left angle of the graph.

A False positive is a term which indicates a situation where the classifier gives a positive label but the true label is negative.

A True positive is a term which indicates a situation where the classifier gives a positive label and the true label is positive.

The same rule can be applied to negative labels.

A False negative is a term which indicates a situation where the classifier gives a negative label but the true label is positive.

A True negative is a term which indicates a situation where the classifier gives a negative label and the true label is negative.

In the axis of the ROC plot:

$$TruePositiveRate = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{39}$$

$$FalsePositiveRate = \frac{FalsePositives}{FalsePositives + TrueNegatives} \tag{40}$$

The diagonal line is a set of points where for a given threshold the False positive rate equals the True positive rate. This situation reflect a random classifier.

In the Polish study ,the following Synthetic Features were the most used in the feature importance:

ranking	**ID**	$\theta_F^{(d)}$
1	(((((((X18-X34)/X56)/X46)/(X24/X27))*(X11*X44))/(((X18-X34)/(X36*X58))+((X38/X30)-X22)))/X46)	.0121
2	(((((X38/X30)-X22)+((X46/X61)*(X61+X21)))*(X30/(((X22*X25)+X27)-((((X47/X27)-X46)*(X33*X39))*(((X38/X30)-X22)+((X18-X34)/(X36*X58)))))))+X46)	.0109
3	(((X18-X34)*((X2-X45)*X46))+X46)	.0106
4	(((((X18-X34)/X56)/X46)-X49)-((((X22*X25)+X27)/(X29*(X18-X34)))*X22))	.0102
5	((((X11-((X2-X45)/X25))-((X18-X34)/(X36*X58)))+(((X11*((X11*X44)-((X1/X61)-(X2-X45))))/(X46/((((X18-X34)/X56)/X46)/(X24/X27))))*X50))*(X11/((X46+X34)*(X29-X58))))	.0094
6	((((X22*X25)+X27)/(X29*(X18-X34)))*X22)	.0084
7	X46	.0077
8	((X29+X29)+(((X22*X25)+X27)/(X29*(X18-X34))))	.0074
9	(((X11*((X11*X44)-((X1/X61)-(X2-X45))))/(X46/((((X18-X34)/X56)/X46)/(X24/X27))))*X50)	.0074
10	((((X38/X30)-X22)+((X46/X61)*(X61+X21)))*(X30/(((X22*X25)+X27)-((((X47/X27)-X46)*(X33*X39))*(((X38/X30)-X22)+((X18-X34)/(X36*X58))))))))	.0074
11	(((((X56+X48)-X11)+X46)+((((X56+X48)-X11)+X46)-((X38/X30)-X22)))-X30)	.0074
12	(((((X22*X25)+X27)/(X29*(X18-X34)))*X22)+(X48+(X25/X31)))	.0072
13	(((X48+(X25/X31))/((X35/(X26/X57))-X29))+(X48+(X25/X31)))	.0069
14	(X25-(((X22*X25)+X27)/(X29*(X18-X34))))	.0067
15	((((((X22*X25)+X27)/(X29*(X18-X34)))*X22)+(X48+(X25/X31)))-X46)	.0067
16	(X46/((X46/((((X18-X34)/X56)/X46)/(X24/X27)))-((X13*(((X61+X21)+X41)/((X2-X45)*X46)))/((X25/X31)-(X47/X27)))))	.0067
17	X29	.0067
18	(((((X38/X30)-X22)+((X18-X34)/(X36*X58)))+((((X47/X27)-X46)*(X33*X39))*(((X38/X30)-X22)+((X18-X34)/(X36*X58)))))-((((X38/X30)-X22)+((X46/X61)*(X61+X21)))*(X30/(((X22*X25)+X27)-((((X47/X27)-X46)*(X33*X39))*(((X38/X30)-X22)+((X18-X34)/(X36*X58)))))))))	.0064
19	((X46/((((X18-X34)/X56)/X46)/(X24/X27)))*(X29+X29))	.0064
20	(X58*(((X18-X34)/(X36*X58))+((X38/X30)-X22)))	.0064

Summarizing:

We took under consideration the financial ratios of different Polish companies, built a data pipeline in Python and created an Extreme Gradient Boosting model with opensource frameworks. The model was then Deployed in the WatsonStudio Env and the results were summarized in the previous tables.

Bankruptcy prediction is a very interesting field that needs to be understood considering the full picture as well as the details. The usage of Artificial Intelligence, with critic supervision of domain and legal experts, might prevent an inappropriate allocation of resources.

We have seen the challenges of such a task as well as the possible approaches to the problem. It's reasonable to say that the performances of the Ensemble methods are indeed a big encouragement to persist with these techniques. It is important to remind the limitations presented at the very beginning. In more general terms, machines should be just a part of a more comprehensive analysis. For the future, it is very important to ensure a proper level of explainability that can match the level of the performances. In doing so, we avoid the Black Box solutions that often limits the adoption of AI solutions in the real world business. This is especially true when innovative approaches, like the one presented in Poland by "Zięba, M., Tomczak, S.K. and Tomczak, J.M" , lead to an improvement in performances.

9 References

[1].[Stuart C. Gilson ,2020 ,"Pandemic Bankruptcy". Article Harvard Business School] link: https://hbswk.hbs.edu/item/coronavirus-could-create-a-bankruptcy-pandemic

[2].[Altman, E.I., Hotchkiss, E. and Wang, W., 2019. Corporate financial distress, restructuring, and bankruptcy: analyze leveraged finance, distressed debt, and bankruptcy. John Wiley & Sons.]

[3].[Altman, E.I., 1968. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. The journal of finance, 23(4), pp.589-609.]

[4].[Osler, C.L. and Hong, G., 2000. Rapidly Rising Coporate Debt: Are Firms Now Vulnerable to an Economic Slowdown?. Current Issues in Economics and Finance, 6(7).]

[5].[Friedman, J., Hastie, T. and Tibshirani, R., 2001. The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.]

[6].[Zięba, M., Tomczak, S.K. and Tomczak, J.M., 2016. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction.]

[7].[Jizhen, C., 2019. Bankruptcy Prediction of a New Data Set of Companies in Norway (Master's thesis, NTNU).]

[8].[Nello Cristianini, John Shawe-Taylor - An Introduction to Support Vector Machines and Other Kernel-based Learning Methods (2013, Cambridge University Press)]

[9].[Zięba, M., Tomczak, S.K. and Tomczak, J.M., 2016. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction.]

[10].[Breiman, L., 1996. Bagging predictors. Machine learning 24, 123–140.]

[11].[Freund, Y., Schapire, R.E., et al., 1996. Experiments with a new boosting algorithm, in: ICML, pp. 148–156.]

[12].[Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics , 1189–1232.]

[13].[Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F., 2012. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 42, 463–484.]

[14].[Fan, W., Stolfo, S.J., Zhang, J., Chan, P.K., 1999. AdaCost: misclassification cost-sensitive boosting, in: ICML, pp. 97–105.]

[15].[Chen, T., He, T., 2015a. Higgs boson discovery with boosted trees, in: JMLR: Workshop and Conference Proceedings.]

[16].[Nanni, L., Lumini, A., 2009. An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. Expert Systems with Applications 36, 3028–3033.]

[17].[Alfaro, E., García, N., Gámez, M., Elizondo, D., 2008. Bankruptcy forecast- ing: An empirical comparison of AdaBoost and neural networks. Decision Support Systems 45, 110–122].

[18].[Alok Kumar, Mayank Jain - Ensemble Learning for Ai Developers_ Learn Bagging, Stacking, and Boosting Methods With Use Cases (2020, Apress)]

[19].[James, G., Witten, D., Hastie, T. and Tibshirani, R., 2017. An Introduction to Statistical Learning: with Applications in R, 7th Edition.]

[20].[Alok Kumar, Mayank Jain - Ensemble Learning for Ai Developers_ Learn Bagging, Stacking, and Boosting Methods With Use Cases (2020, Apress)]

[21].[James, G., Witten, D., Hastie, T. and Tibshirani, R., 2017. An Introduction to Statistical Learning: with Applications in R, 7th Edition.]

[22].[Friedman, J., Hastie, T. and Tibshirani, R., 2001. The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.]

[23].[Zięba, M., Tomczak, S.K. and Tomczak, J.M., 2016. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction.]

[24].[Douzas, Georgios, and Fernando Bacao. “Geometric SMOTE: Effective oversampling for imbalanced learning through a geometric extension of SMOTE.”]

[ ]: