A Comprehensive Workflow for Enhancing Business Bankruptcy Prediction

Rui Portocarrero Sarmento
LIAAD- INESC TEC, Portugal

Luís Trigo
LIAAD- INESC TEC, Portugal

Liliana Fonseca
FEP, University of Porto, Portugal

ABSTRACT

Enterprise bankruptcy is a problem that evolves and is due to several reasons. The bankruptcy problem can cause severe damage to financial institutions. If predicted accurately and ahead of time, it may enable the company to react and change the course of history. This problem may be asserted as a typical classification problem, as several financial ratios are considered attributes. Widespread software tools offer a broad spectrum of Artificial Intelligence algorithms, and the most challenging task may be the decision of selecting that algorithm. Testing solutions to support this decision, on the relatively large amount of available literature in this area with so many options, advantages and pitfalls may be as informative as distracting. In this work, we present an empirical study with a comprehensive Knowledge Discovery and Data Mining (KDD) workflow. With a classifier selection automation, we select an algorithm with better prediction performance than the most widely documented in the literature.

INTRODUCTION

Accurately predicting bankruptcy of a company may be impossible, given the market uncertainty and other adverse situations to which all companies are subjected. However, it is possible to identify those that have a higher chance of not being able to survive and therefore take preventive measures. Bankruptcy prediction is an essential area for decision making of investors, suppliers, clients and even the company itself. Thus, by integrating the latest Artificial Intelligence trends, in an easy to use decision support tool for this area, can further empower a Business Intelligence (BI) system. Hence, Michalewicz et al. (2007) expanded the traditional approach to BI systems, introducing Artificial Intelligence in its scope. They presented the Adaptive Business Intelligence concept, exploring self-learning adaptive systems that learn with previous decisions and recommend the best actions.

Ratio analysis on bankruptcy prediction dates back in the 1930's (Bellovary et al, 2007). Beaver (1966) focused on getting statistical evidence from individual ratios with Discriminant Analysis. Altman Z-score, published in 1968, made this analysis multivariate and is still very popular today. In the 1980's decade, Artificial Intelligence algorithms started to appear. They grew with the ever increasing massive machine processing resources. Today, they outnumber statistical methods in the literature about financial distress.

The goal of this work is to present a structured method for estimating a prediction for the bankruptcy hypothesis. It traverses a comprehensive workflow of Knowledge Discovery and Data Mining (KDD), to explore different classification algorithms.

First, the workflow will address the preprocessing stage. The exploratory analysis of the dataset takes a statistical snapshot of all the variables in the model, precedes the application of various techniques to "clean" the data and select the most relevant attributes for the creation of classification models. The data will then be ready to be processed by some classification algorithms.

The most challenging decision could be choosing the most appropriate algorithm. Thus, several experiments were performed to know which of the classifiers have the best predicting test values. In the end, the best of those classifiers should be selected to forecast the value of the target class of the test data set.

Finally, we benchmark our best classifier performance (J48 classifier), with Artificial Neural Networks, the most widely used algorithm nowadays, in this area.

BACKGROUND

Empirical models to predict corporate bankruptcy and bankruptcy theories have been different strands of research. However, different paths have a substantial amount of overlap (Scott, 1981).

The literature in the field dates back to the 1930's, with the analysis of single financial ratios for specific purposes and industry (Bellovary, Giacomino, & Akers, 2007). With no advanced statistical methods, analysts simply compared failed and non-failed companies, and noticed that failed companies had worst ratio performances.

Beaver (1966), introduced a statistical perspective in univariate ratio analysis. From the 30 selected ratios, only six were significant:

cash flow / total debt
net income / total assets
(current + long-term liabilities) / total assets
working capital / total assets,
current ratio
no-credit interval

Altman (1968), went a step further and introduced Multivariate Discriminant Analysis (MDA), to find the linear function of financial ratios that discriminates failed from non-bankrupted companies. The idea is to classify new sample data with the historical sample used as a training set (Li, Liu, Xu, & Shi, 2004). The result was a combination of five ratios known as Altman Z-score:

$$Z = 0.012X1 + 0.014X2 + 0.033X3 + 0.006X4 + 0.999X5$$

X1 = Working Capital / Total Assets
X2 = Retained Earnings / Total Assets
X3 = Earnings Before Interest and Taxes / Total Assets
X4 = Market Value of Equity / Book Value of Total Liabilities
X5 = Sales/ Total Assets

with the discrimination zones:

$Z > 2.99$ -"Safe" Zones
$1.81 < Z < 2.99$ -"Grey" Zones
$Z < 1.81$ -"Distress" Zones

These first statistical driven studies assumed that bankruptcy could send some signals five years early – prediction accuracy decreases with temporal distance.

The original Altman model, designed for publicly held manufacturing companies with over 1 million dollars in assets, was latter adapted with new values to private firms, as well as non-manufacturers and emerging markets (Altman, 2000). Li (2012) updated Altman Z-scores to predict company bankruptcy in the 2008-2011 crisis.

In the 1980's decade, the first AI approaches started to explore company bankruptcy. These new models were the answer to several critiques to some rigid statistical analysis assumptions, like predefined distributions.

Chen et al. (2011), identified several statistical methods in the literature (simple univariate analysis, multivariate discriminant analysis, logistic regression and factor analysis) as well as AI approaches (artificial neural networks (ANN), rough set theory, support vector machines (SVM), k-nearest neighbour (KNN), Bayesian network, as well as hybrid and ensemble methods), that were successfully applied in the Bankruptcy prediction problem. These authors also note that Artificial Neural Networks became one of the most used techniques since the 1990's decade, as computer processing power started to become a trivial issue.

Classification Algorithms

Classification is a task of supervised learning, that conveys one or more attributes in order to group subpopulations into different labels or classes. The function that maps the attribution of these classes is called a classifier and its output is therefore discrete. The output is what differentiates classification from the other supervised learning method. Regression output is otherwise continuous, and its mapping function is called estimator.

Knowing the underlying assumptions of the most used classification algorithms may be very useful, when the analyst wants to deepen its data understanding. Prediction accuracy for each problem is usually the most critical feature. However, it may also be useful to have better computational efficiency. Transparency is another issue that may arise because, sometimes algorithms that provide the most accurate models, do not reveal how their models are generated.

In this section, some of the most popular classification algorithms available in the Weka software application - and that will be used in this work - are presented.

The most straightforward algorithm to be tested is the OneR method. “OneR” stands for one rule, i.e., based on a unique attribute. Attributes are ranked based on the training set error rate (Holte, 1993). Even considering its simplicity, it can be a relatively accurate method.

Rule sets have several advantages. They are easier to understand and may be used as a first order logic. However, they have some disadvantages too, like poor scaling and noisy data susceptibility. JRip implements the propositional rule learner “Repeated Incremental Pruning to Produce Error Reduction” (RIPPER) proposed by (Cohen, 1995). It is an improved method that grows rules to 100% accuracy and then prunes over-fitting rules until accuracy starts to decrease.

In a decision tree, each node is either a decision node for an attribute or a leaf node corresponding to a classification. In contrast to rules setting based on error rates, decision trees rely on entropy-based measures (Holte, 1993). The decision stump is merely a one level tree resulting in the same configuration of the OneR algorithm over a single attribute.

Logit Regression has a misleading name and is, in fact, a classification algorithm (Bishop, 2006). It uses a logistic function to linearly split the outcome in a binary class. Logistic Model Trees (LMT) is a regression tree that uses logistic regression in their decision nodes.

Haykin (1999) defines an Artificial Neural Network (ANN) as a massively parallel processor, distributed, consisting of simple processing units, which have a natural propensity for storing experiential knowledge and making it available for use.

Sequential Minimal Optimization (SMO) is an optimization of the Support Vector Machines (SVM) algorithm. SVM represents instances as points in space creating clear gaps between different classes (Chandan Kolvankar et al., 2012).

Contrary to the previous algorithms, Instance-based learning (IBl) does not record abstractions from instances but predicts based on specific instances (Aha, Kibler & Albert, 1991). This is why it is straightforward to interpret its results.

The other methods that were used in this study are also known as ensemble methods and assume that a combination of classifiers has better results than each classifier isolated. Bagging is an acronym for bootstrap aggregation (Breiman, 1996). It generates multiple versions of a predictor in order to use them into an aggregated average to predict a class. Boosting “works by sequentially applying a classification algorithm to re-weighted versions of the training data and then taking a weighted majority vote of the sequence of classifiers thus produced” (Jerome Friedman, Hastie & Tibshirani, 2000). It improves the performance of any weak learning algorithm by running on various distributions over the training data and combining the resulting models in a composite classifier (Freund & Schapire, 1996). Stacking differs from boosting because it combines different learning algorithms and tries to balance their strengths and weaknesses.

KDD FOR BANKRUPTCY PREDICTION

This section presents a comprehensive workflow for extracting the best classifiers that will enable accurate and understandable bankruptcy predictions.

Fayyad et al. (1996), identify five sequential stages in the Knowledge Discovery and Data Mining (KDD) process, that is usually taken as a reference. In the first stage – Selection -, the data set is extracted from the Knowledge Bases as a subset of variables or data samples. In the next stage – Preprocessing - , the data set is cleaned and verified for redundancies. In the Transformation stage, data can be submitted to dimensionality reduction or other transformation methods. The present work skipped this step because data was already well fitted for the next step. The Data Mining (DM) stage searches for relevant patterns that fit a particular goal. These authors distinguish two high-level primary goals of DM. The Prediction goal involves searching for unknown values or variables of interest. The Description objective is concerned with finding human-interpretable patterns in this process. The final step is the Interpretation/Evaluation stage.

Software Applications

The purpose of this analytical exploration is to deal with a real context, and for that, we used tools widespread in the academy context, but with increasing expansion in the BI world.

R language is an open source version of the S language developed in the Auckland University. R may be categorized in the same software segment of other well known commercial tools, such as SPSS and SAS. It has an exponentially growing number of packages that cover a wide variety of subjects, from text mining to social network analysis and data visualization. Major BI players like Oracle, SAP, and IBM are integrating their systems with R.

Providing user-friendly analytical tools is a vital software feature to effectively implement BI systems, because it closes the gap with the majority of decision makers that are less aware of IT low-level details. A part of the broad development community is working to provide more intuitive interfaces for this language. Examples of those intuitive GUI packages are Rcommander and Rstudio, and there are also several packages dedicated to generating reports.

Weka is an open source software developed by the University of Waikato in New Zealand. It is developed with an intuitive and more visual user interface than R. It has as an advantage with the possibility of being driven by a specific R package (RWeka), and from R. It has a highly intuitive visual interface called Knowledge Flow that enables the user to conveniently design a workflow to apply to his or her dataset. Finally, it also serves as a basis for a mature open source Business Intelligence software named Pentaho.

Data Selection

The dataset studied in this chapter was kindly provided by Dr. Wieslaw Pietruszkiewicz and is available online, from Pietruszkiewicz (2004). The dataset contains 240 cases, 112 from failed companies and 128 from non-failed ones. Each enterprise data comes from a period of 2 years in a row, so there are observations for 120 unique companies. Observations come from 2 up to 5 years before bankruptcy took place. This dataset was divided into two different groups, for classification prediction purposes, and to build our model. Therefore, it was established the division of dataset in 168 instances (observations or companies) for the training set, and the remaining instances for the test set.

The given training set consists of 31 attributes, 30 are input attributes (called X1, X2, X3, ..., X30) and the rest is an output attribute/class (called STATUS). The input attributes are considered numeric variables, given the considerable variability of values. Moreover, the class has only two possible values ("-1" and "1") which correspond to two discrete nominal values, since it is a classification problem. The value "-1" means bankruptcy, while "1" represents a company's solvency.

Financial ratios try to summarize a company accounted for the use of resources, and are always interpreted considering the benchmarks for the company in its economical sector. Analysis with these ratios can have some pitfalls, because most frequently it is based not on market values (except for sales) but on accounted values – this can be trickier when dealing with a company operating in several economical sectors or with large internal transferences of resources that surpasses market exchange.

These ratios can be divided into several categories. In this dataset, we have identified four groups of ratios. Liquidity ratios represent the company's ability to pay off debts in the short-run. Activity ratios are indicators of the company's efficiency as it measures its ability to generate sales from some specified asset. Profitability ratios measure the ability to generate results from assets or other resources. Financial Leverage ratios entail how the firm has financed its assets and its ability to pay its long-term liabilities.

Liquidity ratios

- X1 - cash/current liabilities
- X2 - cash/total assets
- X3 - current assets/current liabilities
- X4 - current assets/total assets
- X5 - working capital/total assets
- X6 - working capital/sales

Activity ratios

- X7 - sales/inventory
- X8 - sales/receivables
- X16 - sales/receivables
- X17 - sales/total assets
- X18 - sales/current assets
- X19 - (365*receivables)/sales
- X20 - sales/total assets

Profitability ratios

- X9 - net profit/total assets
- X10 - net profit/current assets
- X11 - net profit/current receivables
- X12 - gross profit/sales
- X13 - net profit/liabilities
- X14 - net profit/equity
- X15 - net profit/(equity + long term liabilities)
- X24 - net profit/sales
- X29 - EBIT/total assets

Financial leverage ratios

- X21 - liabilities/total income

X22 - current liabilities/total income
X23 - receivables/liabilities
X25 - liabilities/total assets
X26 - liabilities/equity
X27 - long term liabilities/equity
X28 - current liabilities/equity
X30 - current assets/sales

Preprocessing of data – from exploratory analysis to attribute selection

Proper preparation of the data before applying classification algorithms is crucial to improving the quality of data. It also enables the use of techniques that lead to the construction of models that are more faithful to the actual distribution of data, reducing its computational complexity. Thus, the data characteristics that can harm the analysis may be divided into three categories: incomplete data, redundant data, and data with noise. These elements will be essential for selecting the most relevant attributes to build the model.

Given that there are no missing values or lack of attributes of interest in the dataset, the incomplete data issue is absent.

In contrast, several redundant data issues can be found. The first one is data inconsistency:

- X18 is the inverse of the X30 and must be removed.

- Some of the attributes can be obtained from others: X5 = X6*X20 , X9= X10*X4 , X21 = X22*X25*X1/X2 . This means that some attributes that reference a particular aspect of the data are present more than once. This redundancy makes the induction process slower due to the higher number of attributes to be analyzed by the algorithm.

Correlation between variables is another kind of this problem. The following correlation matrix was prompted by the cor() R command.

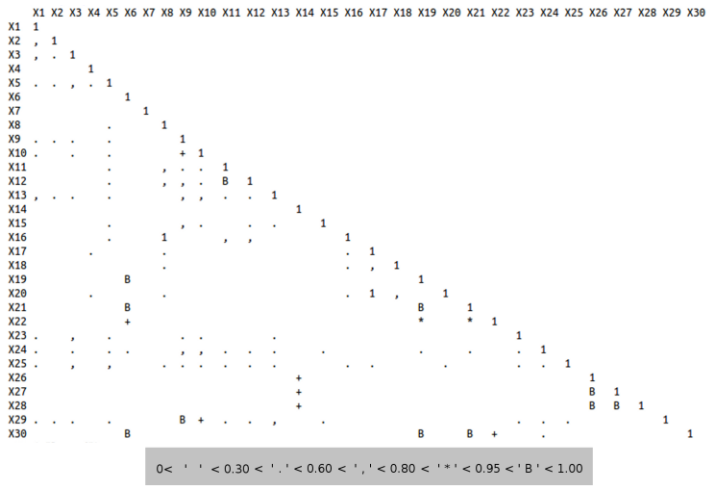

Figure 1. Correlation Matrix of attributes.

Several correlation issues can be found on this matrix:

- Perfect correlation means that X8 = X16 and X17 = X20, therefore, one attribute of each pair might be eliminated.

- Some pairs of attributes are strongly correlated (X12-X11, X19- X6, X6- X21, X21-X19, X27-X26, X28-X26, X28-X27, X29-X9, X30-X6, X30-X19, X30- X21). For each of these pairs, the attribute with less information gain is removed. The attributes are removed deemed irrelevant, since correlated attributes provide a similar profile variation.

The last issue to be solved is data with noise. An indicator of the possible presence of noise is the existence of outliers. The generation of boxplot graphs revealed that almost all of the attributes have several outliers. However, it is essential to clarify that it is not possible to be sure whether some of these values are or are not a result of the presence of noise.

Weka provides several attribute selection algorithms that consider the issues above in order to evaluate attributes relevance.

- Correlation Feature Selection (CFS) involves a search through all possible combinations of data attributes and for attributes of the subset found to enhance the prediction task. The goal is to assess the importance of a subset of the attributes, given an individual's ability to predict, considering the degree of redundancy between them.

- Gain Ratio determines the information gain that an attribute can provide, based on entropy. The attributes that enhance the most substantial information gain are the most interesting, because they are the ones that allow the reader to better separate/distinguish classes.

Table 1. Attribute evaluation and selection regarding class separation with two Weka methods.

AttributeEvaluator	CfsSubsetEval	GainRatioAttributeEval	
SearchMethod	BestFirst -D 1 -N 5	Ranker	Score
Selected attributes	X1	X1	0.2375
	X6	X24	0.2226
	X9	X10	0.2199
	X10	X11	0.2190
	X11	X9	0.2189
	X13	X15	0.2159
	X14	X14	0.2026
	X15	X13	0.1764
	X21	X12	0.1733
	X24	X29	0.1618
	X29	X21	0.1157
		X2	0.1124

		X22	0.0864
		X3	0.8500
		X5	0.0802
		X6	0.0777
		X8	0
		X30	0
		…	…

The table shows that attributes X1, X24, X10, X11, X9, X15, X14, and X13 had the best position in the Gain Ratio ranking, and also belong to the variables selected by CFS - those are the most relevant, the attributes that best discriminate class.

On the other hand, attributes X8, X30, X4, X7, X17, X16, X20, X18, X19, X27, X28, X26, X25, X23 presented lower positions in the ranking with an information gain of zero, and also do not belong to the list of attributes selected by CFS. It means that they are irrelevant, or dispensable for building the model. According to the CFS methodology, attributes relevant to the construction of the model are X1, X6, X9, X10, X11, X13, X14, X15, X21, X24, and X29.

Note that although 16 attributes present information gain greater than zero, the highest value found is only 0.2375, no attributes are clearly better from others regarding information gain. After several experiments, the criterion for selecting the attributes subsequently was CFS. The next stage will consist of the Data Mining stage, and 11 attributes are used.

Data Mining - finding the best classifiers

Before selecting the classifiers, a small piece of R code was developed to discover the best classification models for the preprocessed data obtained with the methods described before. Rweka is a R package that enables R interaction with Weka. R has its software packages with a comprehensive set of classification methods. The decision of using Weka through the R console was binded by the need of being consistent with the evaluation stage that will be performed with the Weka.

In this section, R code to prepare the classification methods, and for testing with a ten folder cross-validation sampling method will be presented.

The code starts with the declaration of the used package, RWeka, the reading of preprocessed data previously created with Weka software, and also the initialization of a dataframe, later used to store the results of calculations for all classifiers:

```
library('RWeka')
w <- read.arff("TRAB1 ECD WEKA-VF.arff")

#prepare dataframe for results
df <- data.frame(classifier = c(0), pctCorrect = c(0),
pctIncorrect=c(0), pctUnclassified= c(0), kappa= c(0),
```

```
meanAbsoluteError=c(0), rootMeanSquaredError=c(0),
relativeAbsoluteError=c(0), rootRelativeSquaredError=c(0))
```

A set of 12 different classifiers was used. Therefore, each of the corresponding functions for each classifier was initialized with the representation of the class (STATUS), and the input data represented by the *w* variable, previously set with the preprocessed data.

```
c1 <- J48(STATUS ~., w)
c2 <- IBk(STATUS ~., w)
c3 <- AdaBoostM1(STATUS ~., w)
c4 <- Bagging(STATUS ~., w)
c5 <- DecisionStump(STATUS ~., w)
c6 <- JRip(STATUS ~., w)
c7 <- LMT(STATUS ~., w)
c8 <- Logistic(STATUS ~., w)
c9 <- LogitBoost(STATUS ~., w)
c10 <- OneR(STATUS ~., w)
c11 <- SMO(STATUS ~., w)
c12 <- Stacking(STATUS ~., w)
```

The adopted sampling method was cross-validation with ten folders, which enables all data to be used for training and testing. In cross validation the initial data are randomly divided into K subsets D_1, D_2, ..., D_k approximately equal in size. Then a model is estimated using K-1 subsets, and tested with the remaining subset. The above procedure is repeated K times, always using a different subset for the test in each iteration.

Note that the cross-validation is considered superior to validation dividing the sample to small sets of training examples, as is the case. The final error of generalization is achieved by averaging the obtained validation errors in K iterations.

After initiating classifiers, a cross-validation method gets results from classifiers, and the results are recorded to the *df* dataframe with a final ordering of results, descending from the classifier with lower error percentage to the one most error prone.

```
for(i in 1:12){
      name <- paste("e", i, sep = "")
      classifier <- paste("c", i, sep = "")
      assign(name,evaluate_Weka_classifier(get(classifier),
      cost = matrix(c(0,2,1,0), ncol = 2),
      numFolds = 10, complexity = TRUE,
      seed = 123, class = TRUE))

      name_aux <- paste("e",i,sep="")
      aux <- get(name_aux)
      res <- c()
      for(j in 1:8){
            res <- c(res,unlist(aux[2])[j])
      }
```

```
#fill the dataframe with classifiers result
     df <- rbind(df,c(paste(classifier),res))
}


#finish dataframe composing and order decreasing by
#percentage of correct classification
df <- df[-1,]
df <- df[order(df[,2], decreasing=TRUE),]
```

From the generated dataframe, an accuracy table was extracted for all the tested classifiers.

Table 2. Classifier ranking for correct classes.

Classifier	% Correct
J48	91.67
Ibk	89.29
Bagging	87.50
LMT	86.90
JRip	80.95
LogitBoost	79.17
OneR	77.98
DecisionStump	73.81
Logistic	73.81
AdaBoost	72.02
SMO	72.02
Stacking	50.60

From the available results, J48 classifier with the standard parameters was selected, because it was the classifier with more accuracy among the group of 12 available classifiers.

A closer performance analysis

Neural Networks classifier has been widely used to build predictions of bankruptcy models (Bellovary, Giacomino & Akers, 2007). Therefore, we will compare that algorithm performance with J48, the algorithm that provided the best result in Table 2.

Artificial Neural Networks

Neural networks are computational mechanisms that attempt to mimic the human brain, and are a popular technique in Data Mining that uses the optimization of some particular function.

According to (Haykin, 1999), an Artificial Neural Network (ANN) is a massively parallel processor, distributed, consisting of simple processing units, which have a natural propensity for storing experiential knowledge and making it available for use. It is similar to the brain, once the knowledge is acquired by the network from its environment, through a learning process, and the connection strengths between neurons, known as synaptic weights which are used to store the acquired knowledge.

Usually, layers are classified into three groups. In the input layer, variables are presented to the network. Intermediate or hidden layers, where most of the processing through weighted connections is made, can be considered as an extractor of characteristics. The output layer, is where the end result is completed and submitted as output.

A neural network is specified mainly by its topology, the characteristics of the nodes, and the rules of training.

The nodes have an activation function that makes a comparison of the sum of weighted inputs with a threshold value. If the sum of the inputs is greater than the threshold, the output neuron is activated, remaining turned off otherwise. If there are *n* entries for the unit, the output or activation will be equal to

$$a = g\left(w_1 x_1 + w_2 x_2 + ... + w_n x_n\right),$$

where g is an activation function, usually non-linear.

The arrangement of neurons in the layers and the pattern of connections between them define the architecture of the ANN. In networks without feedback (feedforward), neurons are grouped into layers, and the signal traverses the network in a single direction, from input to output. Neurons in the same layer are not connected.

For networks with feedback or recurrent, the output of some neurons in the same layer feeds others (including himself) or previous layers; the signal traverses the network in both directions, has dynamic memory and ability to represent states in dynamic systems.

Chains with more than one layer are called multilayer perceptron (MLP), which represent a generalized single-layer perceptron (composed around a nonlinear neuron, i.e., the model of a McCulloch-Pitts neuron). The MLP is applied through his training as supervised by an algorithm known as back-propagation algorithm. It consists of the iteration of two phases – one phase to the front and a stage backwards. In step forward, each object entry is received by each of the first hidden layer neuron from the network when weighted by the weight associated to its corresponding input connections. Each neuron layer then applies the activation function and produces an output value (value of the input neurons of the next layer). This process continues until reaching the output layer neurons, and each ends producing its output value. The error on the network to the displayed object is the difference between the output values produced and the desired for each neuron of the output layer. The error value of each neuron of the output

layer, is then used in backward steps, to adjust the weights in the activation functions. The adjustment continues from the output layer to the first intermediate layer.

There are some positive aspects on ANN, that made them very popular as an Artificial Intelligence algorithm:

- Applied to tasks where sets of examples on a given problem are available, performing the automatic acquisition of knowledge;

- Good capabilities for discovering pattern associations between inputs and outputs;

- Good for supervised or unsupervised classification patterns;

- Applicable to an approximation of unknown functions through samples of these functions;

- Can work with approximate, incomplete and inaccurate data;

- They are prone to parallelism, generalization, robustness;

- They are also applied to "complex tasks usually performed by humans";

ANN also has some disadvantages that may condition its application:

- Their predictive capacity is still far below the capacity of the human brain;

- They are known for some difficulty in working with high-level symbolic knowledge;

- Their results imply some difficulty to achieve explicitness of knowledge;

- There is some difficulty in defining the network structure, its parameters, and the database;

- They lack guaranteed convergence of the algorithm to an optimal solution;

Configuration parameters in Weka's perceptron classifier

The parameter *autoBuild* adds and activates the hidden layers in the network and was considered active (True). *HiddenLayers* defines the neural network hidden layers quantity and its 'a' value correspond to (class + attributes)/2. The *Learningrate* parameter indicates the learning rate, which should not be too small (many cycles may be required) nor too high (can hinder convergence). *Momentum* has a value of 0.2 by quantifying the degree of importance of weight change of the previous cycle concerning the current cycle. *Reset* is considered active, allowing the network to restart with a reduced learning rate. The *trainingTime* considered was 500. The *validationThreshold* parameter takes the value 20, used to complete the validation test.

Decision Trees (J-48)

A decision tree uses the divide and conquer strategy to solve a decision problem, i.e., a complex problem is divided into simpler problems. The same strategy is recursively applied. The solutions can be combined in the form of a tree, to produce a solution for the problem. It is a predictive model based on demand, in

which each node is either a decision node or a leaf node. Each leaf node is labeled with a function, and a leaf represents a class. Each decision node contains a conditional test based on attribute values – each branch as a rule with a conditional part and a conclusion.

The aim is to group the examples recursively, minimizing the entropy, which is a measure of randomness of the target attribute. At each decision node, the attribute chosen to divide the data is one with the purpose to reduce the randomness. This choice is based on gain information, which measures the decrease in entropy in the partitions in accordance with the obtained attribute values.

The J-48 algorithm is a variant of the C4.5 algorithm, in Java programming language, which can, through the discretization of numeric values using own heuristics, process attributes of any kind at the entrance. The pruning mechanisms, of great importance, in that switching deep nodes by leaves, helps minimize the problems caused by noise.

The positive aspects of decision trees are:

- Easy interpretation because one sees the reason for the classification decision. The model is understandable as a tree;

- They have good flexibility because no particular distribution is assumed for the data;

- They have robustness as univariate trees are insensitive to monotonic transformations of input variables;

- They have an automatic choice of the most relevant attributes. This means that in each case most relevant attributes appear higher up the tree.

Decision trees also have some negative aspects:

- They have a high sensitivity to small disturbances in the training set. Consequently, they tend to generate very different networks;

- They are prone to replication by duplicating a sequence of tests in different branches;

- Missing values in the data can cause trouble deciding which branch to follow in instance classification;

- Regarding processing time, continuous attributes can consume much time. In this case, discretization of these attributes should be considered for faster results.

Configuration parameters in Weka's J48 classifier

Weka software features parameters customization, to use with the J48 classifier. These are shown in Figure 2.

Figure 2. Weka's J48 classifier parameters.

The option *binarySplits*, set to false, enables the use of binary divisions on nominal attributes when building the decision tree. The parameter *confidenceFactor*, assumed as 0.25, is used to perform pruning in decision tree in order to remove any branch that does not meet the ratio between examples correctly and incorrectly classified – the lower, more pruning.

The *minNumObj* option is set to the value 2, which corresponds to the minimum number of leaf level examples of the decision tree. The *numFolds* parameter, which takes the value 3, determines the amount of data used to reduce error pruning. The option *reducedErrorPruning*, set to False, is the most important since it allows choosing the proper algorithm for error reduction – Weka pruning algorithm instead of the standard specification of the classifier C4.5. The parameter *saveInstanceData*, set to False, allows the reader to save the data for display on the screen. The *subtreeRaising* assumed to True, allows the birth of subtrees on the time of pruning. The *unpruned* option disables pruning and is therefore set to False.

Error rates: ANN vs. J48

Several experiments were performed, in order to find the best possible results. After the selection of attributes (CFS), classifiers selection and of parameters definition, tools such as Normalize, Discretize and resample were tested.

In this section, the evaluation process with Weka Experimenter will be described.

In the "*Preprocess*" tab, we apply the filter *Unsupervised -> Attribute -> NumericToNominal* and indicate that this will be applied to the last attribute - the class (which was placed in the last place of the data table).

In the "*Preprocess*" tab, we apply the filter *Supervised -> attribute -> AttributeSelection* using *CfsSubsetEval* evaluator, and *BestFirst* search method.

After previous settings, the "*resample*" tool (filter: *Supervised -> instance -> resample*) was used. It proved to reduce the percentage of classification error conveniently. Once there is a sample with relatively few instances, the difference between class majority and minority class can be significant (90-78). This is detrimental to the classification, especially for decision trees, because the classifier tends to classify and give results for the majority class, leaving the minor class with low percentage of classified cases. In the context of predicting bankruptcy of companies, the target class is '-1' (failed), which in this case corresponds to the minority class. The use of "*resample*" provides a balance between classes by considering the parameter *BiasToUniformClass* = 0.5 – it gives a more balanced sample: class' -1 '-> 83 instances and class '1' -> 85 instances. Therefore, it produces a random sample of the sub-dataset using sampling replacement.

Results for each classifier are presented in the following table.

Table 3. J48 and Neural Network classifiers results for the used data.

Classifier	**Error Rate (%)**	**AUC ROC**
Neural Networks	23.8095 %	0.859
J-48	7.7381 %	0.945

From the J-48 parametrization, the following decision tree was generated.

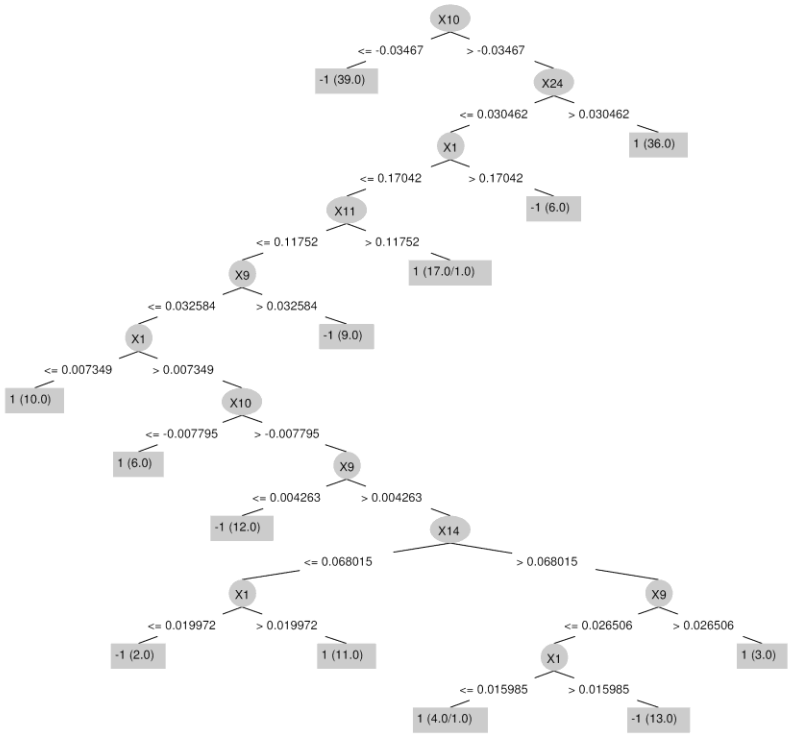

Figure 3. J48 decision tree.

This decision tree only uses six financial ratios from the 11 input ratios after preprocessing. Most of them (five) were profitability ratios and one isolated liquidity ratio. Leverage and Activity ratios are not considered important by this model to predict bankruptcy, even considering these ratios are somewhat related to each other – this relation is more visible for the X24 attribute that has a high correlation with Activity ratios has can be seen on the previously shown correlation matrix. The decision tree also allows us to have a description of how the algorithm will classify future values. For example, an observation that has X10 (net profit/current asset) attribute with values exceeding -0.03467, X24 (net profit/sales) less than or equal to 0.030462 and X1 (cash/current liabilities) greater than 0.17042 will be classified as failing ('-1'). The tree shows the generated output attribute, to the leaf level, following numerical information (in parentheses), indicating the number of samples that reached each leaf, and existing incorrectly classified examples. For example, (4.0/1.0) on the Weka tree leaf means that for four examples of the reached leaf level one is incorrectly classified.

In the following confusion matrix, generated with the J-48 Weka's classifier results, we split the results into four categories - TP - True Positive (hit); TN - True Negative (hit); FN - False Negative (error); FP - False Positive (error)

Table 4. Confusion Matrix for J48 classifier.

-1	**1**	**<- classified as**
75 (TP)	8 (FN)	**-1**
5 (FP)	80 (TN)	**1**

Some relevant statistics can be inferred from the previous table:

FP Rate = FP / (TN + FP) = 5 / (5 + 80) = 5.9 %
TP Rate = TP / (TP + FN) = 75 / (75 +8) = 90.4 %
Rate of incorrectly classified instances = (FN + FP) / (TP + TN + FN + FP)
= (8 +5) / (75 + 80 + 8 + 5) = 7.738 %

Thus, the error rate of the classifier with this database is 7.738%, meaning that the probability of this classifier makes the type I and II errors in predicting is 7.738%. Another way to obtain the rate error is by reading the Weka model output.

Applying the same method to the Multilayer Perceptron classifier confusion matrix, it can be inferred an error rate of 23.810%.

Analyzing the results obtained for each classifier, it is concluded that the J-classifier 48 is more useful to estimate, for this data set. Thus, it has an error rate much lower than Multilayer Perceptron classifier, the amount of 7.7381%, which can be considered an excellent result, with regard to the low number of training instances (168).

Further parameter tuning for J48

The definition of the configuration parameters of a classifier is a critical stage in its outcome behavior. Thus, the change in values can lead to significant changes in the results. As such, a study of the parameter "*reducedErrorPruning*" for the J-48 classifier was made.

Previously set to False, has now been changed to True, which means that it was chosen the standard specification of the algorithm C4.5 classifier to reduce error pruning. It allows obtaining error estimates from a validation set independent of the training set. This method reduces the volume of information available to grow the tree, changing the decision tree that has been reduced. As a result, it was found that the error rate increased.

By observing the output, the error rate is 25% and the ROC AUC area 0.821. Therefore, the reader might conclude that this change in the parameter of the classifier had a negative influence since the error rate increased considerably and the ROC area decreased. In this case, the error rate began to be higher than the error rate of the Multilayer Perceptron classifier, outlined above, again emphasizing the importance of choosing appropriate parameters for best results.

Evaluation with ROC curve analysis

The ROC analysis (Receiver Operating Characteristic), is a powerful tool to measure and evaluate problems of prediction. Using a robust graphic method allows the study of variation of sensitivity and specificity for different values of cut. The area under the ROC curve is associated with the discriminating power of a classification algorithm.

Geometrically, the ROC curve corresponds to a graph of "x" and "y" in a plane called the ROC plane unit. The ROC plane designation unit is because the coordinates of this graph are probability measures, and thus vary between zero and one. The point (0,1) is perfect ratings, in which all positive and negative examples are classified correctly. The point (1,1) is always positive ratings, and the point (0,0), always negative ratings. Thus, classifiers in the region near the point (1,1) almost always labeled examples as positive and classifiers in the region near the point (0,0) label the majority of examples as negative. A classifier is considered better than another if a point in space is above the ROC curve and left of the point corresponding to the second classifier.

The ROC curves were generated with the Knowledge Flow Weka feature.

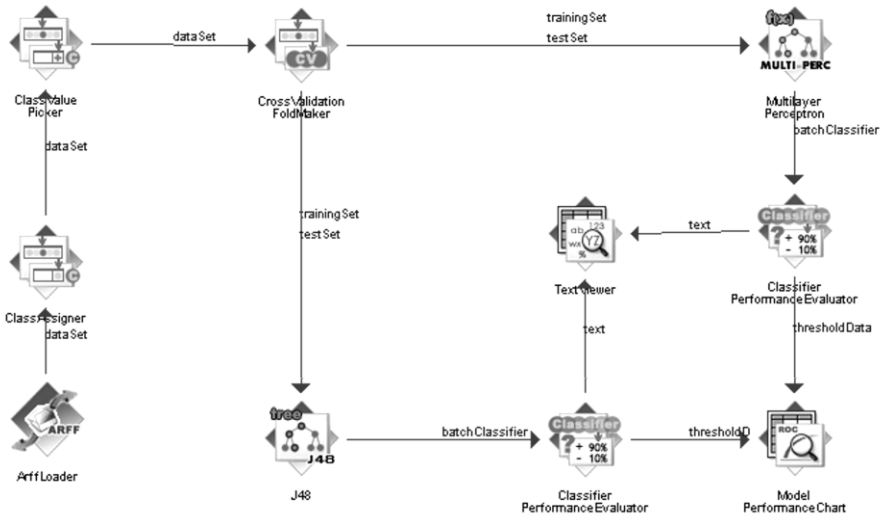

Figure 4. Workflow for ROC analysis.

The generated ROC curves, with the previous "knowledge flow", are in the following image.

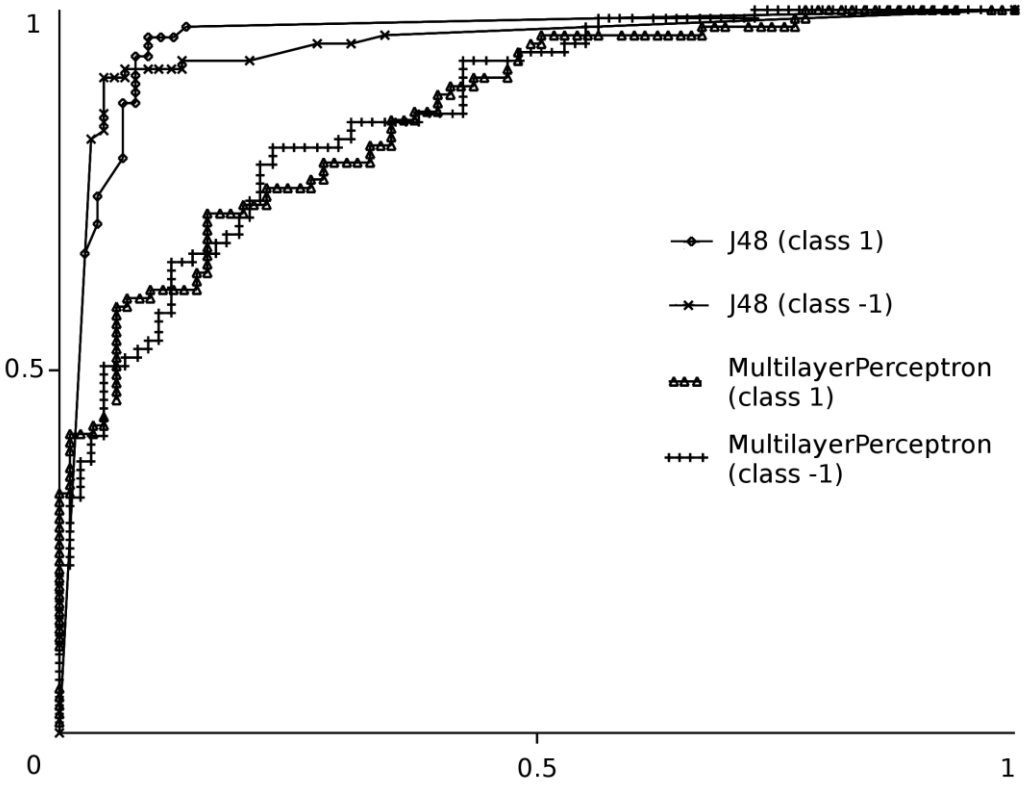

Figure 5. ROC curves for J48 and Perceptron ANN classifiers.

From Weka's model outputs, the reader can also obtain ROC Area Under Curve (AUC), and compare both classifier models:

[AUC (J-48)] 0.945 > 0.859 [AUC (Multilayer Perceptron)]

Figure 5 and AUC measures, enable to conclude that the classifier with the best performance is J-48, since its curve is almost always above the Multilayer Perceptron curve, for both classes.

Test set predictions

The prediction of the test model was based on Decision Trees (J-48), because it was the one who had the lowest error rate (7.7381%), and majored areas of the ROC (0.945). After the selection of attributes that would enter the classification models, new native training and testing WEKA files (ARFF files format) were created, to allow us to predict the test set. With some R programming, a new ARFF test file was created, and only with previously selected attributes.

A 'ghost' column was also added, filled with all the values '1' in order to maintain compatibility, except the first line in training class that was set to '-1'. This is done to have two classes in the test set as otherwise, Weka does not accept the new test set.

Test data was then imported with WEKA. With the "Supplied test set" option, we select the new ARFF test file, and make the model prediction, activating the option "Output predictions". In the generated output, the predicted values are those in column 'predicted'.

Table 5. Sample Predictions on test set.

instance	actual	predicted	error	probability distribution
1	1: -1	1: -1	0	1
2	2:1	2:1	0.058	0.941
3	2:1	2:1	0	1
...	...	...	...	...

The "actual" column, which can be ignored, states that each class belongs to two different classes (1 and 2 corresponding to class marked as '-1' and '1' respectively and on this dataset). The "predicted" column shows that instance 1 is predicted to be of class '-1' and instance 2 through 3 are predicted to be of class '1'. The error field indicates it is 0 for instances 1 and 3 and 0.059 for instance 2; being predictions performed on a labeled test set, each instance where the prediction fails to match the label would contain a "+". As an example, the probability that instance 2 belongs to class '1' is estimated at 0.941.

FUTURE RESEARCH DIRECTIONS

The automation to find the best classifier with the standard parameters, could be extensively expanded to provide more insight on the classifiers behavior, with different input parameters. This could provide better combinations of input parameters, for the available classifiers. It could also lead to finding of even better accuracy in the early stages of the KDD process with this data, though attention should be taken to avoid over-fitting the data.

Other exciting application of this data would be the study of social network constraints that would influence the spread of failure conditions, over an industry or cross industries.

CONCLUSION

The correct classification of the data depends not only of a suitable classifier, but also a right combination of the data preparation and use of suitable algorithms. We automated the classifier selection process, in order to go beyond theory assessments about the most suited methods.

Stream-lining a complete and specific KDD workflow enables some work automation and reduces human effort. Frequently, the most human power consuming issue lies in the data selection, the preprocessing stages and its data consistency checking. A robust data and business awareness are fundamental to have the best working results.

Moreover, workflow automation can be most of the times extended to replicate the workflow to other spatial-temporal data. This is a self-reinforcing human-machine interaction that can increase dramatically BI systems strength. The better the input a human decision maker provides, the better the output the machine provides. In the next iteration, the previous machine output will mean better data and business awareness as well as a better selection and preprocessing of data – a better input into the system.

AI tools are becoming more intuitive and affordable for proper integration with BI systems. The Weka tool was chosen because it is a powerful tool for analysis and forecasting of data, besides having many algorithms, allows for flexibility in testing. R language was also considered useful for its smooth and fast programming statistical capabilities, with an excellent interface to other tools like Weka – it was fundamental to run the analysis of the variables of the training set and preliminary selection of classification algorithms.

Regarding specific conclusions derived from this work, we had better performance of the algorithm J-48 (Decision Trees), when compared to Multilayer Perceptron (Artificial Neural Network). A smaller error rate (7.7381%) and larger areas of the ROC curve (0.945) were achieved. The results obtained were confirmed with significance tests underlining better overall results for J-48, on this single classification problem. It is also apparent how crucial it is to accurately define the parameters of the model, for improving accuracy. Moreover, and for this dataset, J48 classifier is simultaneously more accurate and highly comprehensible, due to its tree model representation, unlike ANN.

REFERENCES

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, (6), 37–66.

Altman, E. I. (2000). Predicting Financial Distress of Companies: Revisiting the Z-Score and Zeta® Models.

Beaver, W. H. (1966). Financial Ratios As Predictors of Failure. *Journal of Accounting Research*, *4*(Empirical Research in Accounting: Selected Studies 1966), 71–111.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140.

Chandan Kolvankar, Trivedi, J., Mani, B., Ramanathan, R., & Kadam, S. (2012). Support Vector Machine for Learning in Artificial Intelligence Systems. *International Journal of Engineering Research and Applications (IJERA)*, 409–412.

Chen, H.-L., Yang, B., Wang, G., Liu, J., Xu, X., Wang, S.-J., & Liu, D.-Y. (2011). A novel bankruptcy prediction model based on an adaptive fuzzy k-nearest neighbor method. *Knowledge-Based Systems*, *24*(8), 1348–1359.

Cohen, W. W. (1995). Fast Effective Rule Induction. In *In Proceedings of the Twelfth International Conference on Machine Learning* (pp. 115–123). Morgan Kaufmann.

Fayyad, U. M., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery: an overview. In *Advances in knowledge discovery and data mining* (pp. 1–34). American Association for Artificial Intelligence.

Freund, Y., & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *Proceedings of the Thirteenth International Conference* (pp. 148–156).

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (2nd ed.). Prentice-Hall.

Holte, R. C. (1993). Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. In *Machine Learning* (pp. 63–91).

Jerome Friedman, Hastie, T., & Tibshirani, R. (2000). Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics*, *28*(2), 337–407.

Li, J., Liu, J., Xu, W., & Shi, Y. (2004). Support Vector Machines Approach to Credit Assessment (Vol. 3039, pp. 892–899). Presented at the International Conference on Computational Science, Springer.

Pietruszkiewicz, W. (2004). *Application of Discrete Predicting Structures in an Early Warning Expert System for Financial Distress*. Szczecin University of Technology. Retrieved from http://www.pietruszkiewicz.com

Scott, J. (1981). The probability of bankruptcy: A comparison of empirical predictions and theoretical models. *Journal of Banking & Finance*, *5*(3), 317–344.

ADDITIONAL READING SECTION

Aktan, S. (2011a). Application of machine learning algorithms for business failure prediction. *Investment Management and Financial Innovations*, *8*(2).

Aktan, S. (2011b). *Early Warning System for Bankruptcy: Bankruptcy Prediction*. Karlslruhe.

Assessing the Systemic Implications of Financial Linkages. (2009). In *Global Financial Stability Report Responding to the Financial Crisis and Measuring Systemic Risks*. International Monetary Fund. Retrieved from http://www.imf.org/external/pubs/ft/gfsr/2009/01/pdf/chap2.pdf

Bellovary, J., Giacomino, D., & Akers, M. (2007). A Review of Bankruptcy Prediction Studies: 1930 to Present. *Journal of Financial Education, 33*.

Cessie, L. S., & Houwelingen, J. C. van. (1992). Ridge Estimators in Logistic Regression. *Applied Statistics, Vol. 41(1)*, pp. 191–201.

Fawcett, T. (2006). ROC Analysis in Pattern Recognition. *Pattern Recognition Letters*, *27*(8), 861–874.

Greene, W. H. (n.d.). Econometric Analysis. In *Econometric Analysis* (7th ed.).

Hand, D. J., & Till, R. J. (2001). A simple generalization of the area under the ROC curve for multiple class classification problems. *Machine Learning*, *45*(2), 171–186.

Iba, W., & Langley, P. (1992). Induction of One-Level Decision Trees. In *In Proceedings of the Ninth International Conference on Machine Learning* (pp. 233–240). Aberdeen: Morgan Kaufmann.

Kainulainen, L., Yu, Q., Miche, Y., Eirola, E., Séverin, E., & Lendasse, A. (2010). Ensembles of Locally Linear Models: Application to Bankruptcy Prediction. In *In proceeding of: Proceedings of The 2010 International Conference on Data Mining*.

Khan, R. A. (2012). KDD for Business Intelligence. *Journal of Knowledge Management Practice*, *13*(2).

Kohavi, R., & Quinlan, R. (1999). Decision Tree Discovery. In *IN HANDBOOK OF DATA MINING AND KNOWLEDGE DISCOVERY* (pp. 267–276). University Press.

Landwehr, N., Hall, M., & Frank, E. (2004, June 10). Logistic Model Trees. Kluwer Academic Publishers.

Michalewicz, Z., Schmidt, M., Michalewicz, M., & Chiriac, C. (2006). *Adaptive Business Intelligence*. Springer-Verlag New York, Inc.

Pai, G. A. R., & Pai, G. A. V. (2010). Bankruptcy Prediction Using Principal Component Analysis Based Neural Network Models. In *Business Intelligence in Economic Forecasting: Technologies and Techniques* (pp. 138–140).

Pietruszkiewicz, W. (2008). Dynamical Systems and Nonlinear Kalman Filtering Applied in Classification (pp. 1–6). Presented at the 7th IEEE International Conference on Cybernetic Intelligent Systems, 2008, London.

Platt, J. C. (n.d.). Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods* (pp. 185–208). MIT Press Cambridge, MA, USA ©1999.

Ravi, V., & Ravi Kumar, P. (2007). Bankruptcy prediction in banks and firms via statistical and intelligent techniques – A review. *European Journal of Operational Research*, 1–28.

Wang, Y., Wang, S., & Lai, K. K. (2005). A New Fuzzy Support Vector Machine to Evaluate Credit Risk. *IEEE Transactions on Fuzzy Systems*, *13*(6), 820–831.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*, 241–259.

Yoav Freund, & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, (55), 119–139.

Yu, Q., Miche, Y., Séverin, E., & Lendasse, A. (n.d.). Bankruptcy Prediction with Missing Data. In *In proceeding of: Proceedings of the 2011 International Conference on Data Mining*.

Zhou, Z.-H. (2009). Ensemble Learning. In *Encyclopedia of Biometrics* (pp. 270–273). Springer US.

KEY TERMS & DEFINITIONS

Bankruptcy Analysis: Finance and accounting research area covering the analysis of financial distress measures of companies.

Financial Ratios: A magnitude relating relevant financial statement items.

Artificial Intelligence: Computer Science research field covering self-learning adaptive systems aiming to solve complex problems.

KDD Workflow: The set of sequential data mining steps needed to extract meaningful knowledge from data.

Classification Models: Logical procedures of supervised learning that conveys one or more attributes in order to group subpopulations into different labels or classes.

Feature Selection: Part of the preprocessing stage aiming to select relevant features for the classification stage.

Classifier Selection: Evaluation task of choosing the appropriated classification models that best suits the comprehension or prediction of a given problem.