Credit Risk Analysis using Neural Networks

Manal Mohania

Australian National University, Canberra, Australia
manal.mohania@anu.edu.au

Abstract. Credit risk analysis is a hot area of research in the financial communities. This paper proposes a solution for the credit risk assessment problem by creating a neural network that is trained on pre-processed data, with a genetic algorithm applied to obtain the optimal hyperparameters. The data pre-processing techniques used are largely influenced from the ones used in "Bustos and Gedeon's, Decrypting Neural Network Data: A GIS Case Study". These general techniques are applied to the Credit Approval dataset obtained from UC Irvine Machine Learning Repository. A significant boost in performance is observed when the neural network is trained and tested on the pre-processed data as compared to the raw data. Finally, these results are compared to those obtained from other prediction methods applied on the same dataset.

Keywords: neural networks, data pre-processing, genetic algorithms, credit risk assessment

1 Introduction

Real world data is often incomplete, inconsistent and noisy [1]. Predictions made from such data are usually very ordinary because the quality of the accuracy is directly influenced by that of the data. Raw data is generally pre-processed to make it more usable. It helps one to extract underlying features and patterns much more easily and faster on application of any machine learning algorithm. We follow some data processing techniques from Bustos and Gedoen's GIS Case Study [5], and also use some of our own while analysing the performance of the neural network. Furthermore, we make use of a genetic algorithm to select the optimal hyperparameters.

1.1 The Dataset

Credit risk assessment is an important area of research for many financial institutions. Analysing the customers' background and history goes a long way in imposing soft and hard limits on the amounts that can be credited to them. Such research has direct impacts on the profitability of these institutions. For this reason, we test our neural network on a dataset from the financial domain.

The Credit Approval Data Set [2] obtained from the UCI Machine Learning Repository describes positive and negative instances of people who were granted credit by a particular financial institution in Australia.

This dataset is quite interesting to use because it contains a good mix of attributes- continuous variables, categorical variables with small number of values and categorical variables with larger number of values. The dataset itself isn't very large (690 instances and 15 attributes). It also contains rows with one or more missing values (5% of the rows).

The biggest challenge while using this dataset is the fact that all attribute names and values have been changed to meaningless symbols (in order to protect confidentiality of the data). This poses a major challenge because anonymising the data and the attributes makes it harder to perform pre-processing as ideally one would like to have as much information as possible about the attributes and their data. Nevertheless, we are able to obtain superior accuracy using some preprocessing techniques and a simple neural network to other more sophisticated classification algorithms.

The aim of the experiment is to classify people into two categories- those whose credit application was approved, and those whose was not. The target variable in the dataset is under the attribute $A16$. It has two values $(+)$ and $(-)$. Although this information is not explicitly used, it is safe to assume that $(+)$ denotes the category of people whose credit application was accepted, and $(-)$ is the category of people whose credit application was rejected.

1.2 Genetic Algorithm

A genetic algorithm (GA) is an optimization and search technique based on the principles of genetics and natural selection. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" [6]. Since the space of all possible combinations of hyperparameters is quite large, we use a genetic algorithm to narrow down the search and obtain the combination hyperparameters for the neural network that provide the best performance on the test set. We discuss the exact details of the algorithm used in a later section.

2 Data Pre-Processing

We employed various techniques to process the raw data depending on the type and distribution of the variables. Figure 1 depicts the frequency plots for each attribute in the dataset. The pre-processing consists of three different stages: Filling missing values, transformation of attributes, and converting categorical variables to one-hot vectors. We talk about the different stages in the remainder of this section.

2.1 Filling Missing Values

37 rows in the dataset (5%) contain 1 or more missing values. Since, the reason for the missing data is not mentioned, we do not make any assumptions about it.

In the case of missing values from continuous variables, our first approach was to replace the missing the data with the attribute mean. However, on experimentation, it was found that the neural network performs better if

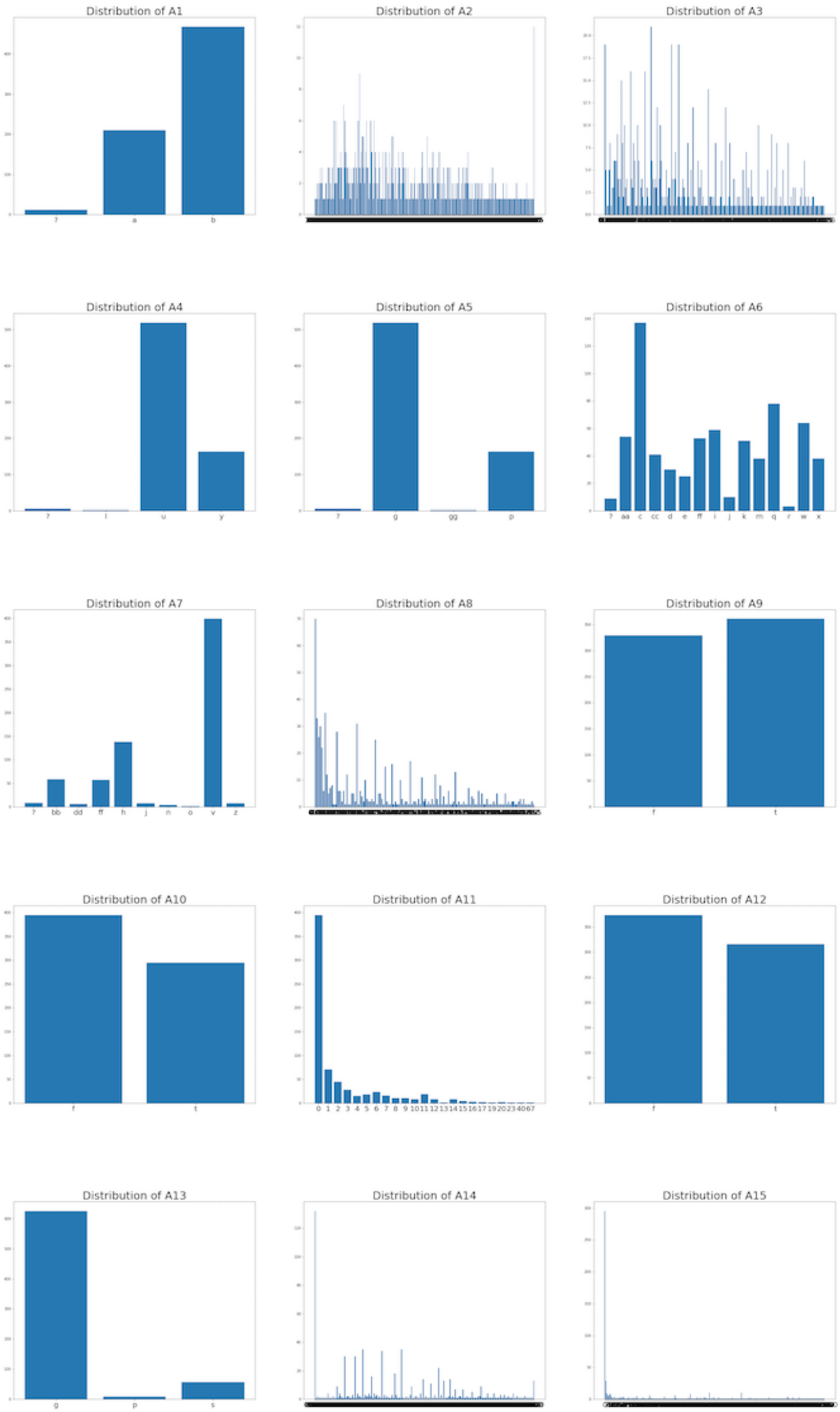

Fig. 1: Graphs showing the frequency plot for each attribute

there is some noise in the missing values. So instead, we sampled the missing values from a gaussian distribution centred at the mean and with variance equal to that of the attribute. The better performance in this case can be attributed to the fact that when we replace the missing values to the mean, we are introducing bias into the data. Instead, when we sample the missing values from the gaussian distribution, we are still introducing bias but much less so.

Another method could have been to simply discard the rows where there was missing data. We chose not to do so because the number of instances in the dataset is not very high.

Methods such as predictive modelling using linear regression could also have been used to infer the missing data. However, these methods were not used because the percentage of missing data was not very high, and also that these methods introduce bias of their own.

For categorical variables, in the instances where there weren't enough variables with low counts, we simply replaced the missing values with the mode [7]. Otherwise, we clubbed the missing values with other attributes of lower count into a category of their own.

2.2 Data Transformation

Based on what the frequency plot looked like for each attribute, we apply different techniques.
Attribute A1: We replace the missing values with the mode. This leaves us with two major categories for this variable.
Attribute A2: This attribute is a continuous variable. Upon plotting it, we see that it is unimodal. It moderately resembles a gaussian plot. We scale all values in the $\mathcal{N}(0, 1)$ range for the logistic cost.
Attribute A3: Just like A2, this attribute is a continuous variable too. After replacing the missing values using the algorithm described in the previous section, this attribute also moderately depicts a gaussian. Just like was done in A2, we scale all values in the $\mathcal{N}(0, 1)$ range.
Attribute A4: The fist thing that we noticed from its plot is that the frequencies of variables in A4 and A5 are identical. After further inspection, we observe that there is a one-to-one mapping between the variables in A4 and A5. Thus, A4 does not provide us with any further information than what A5 already does. Hence, we remove this attribute entirely.
Attribute A5: The insignificant variables are replaced with the mode because there aren't sufficient instance of those variables.
Attribute A6: This variable is composed of numerous categorical variables. The distribution is mostly uniform, with a few variables with low frequencies and one variable with comparatively high frequency. Since the number of categories is too high (15), we combine the categories so as to form four major categories. Since in the next stage we convert categorical features to one hot encodings, this ensures that this attribute isn't over-represented among the input neurons.
Attribute A7: We only have two major categories in this instance- v and h. The rest of the variables (including the missing values) are combined together to form a category of their own.
Attribute A8: This variable is continuous with high skew towards the right. To lower the effect of the skewness we consider the logarithm of every variable in this attribute when performing further analysis.
Attribute A9: As is evident from its graph, this attribute has a fair distribution of its variables. We do not process it futher.
Attribute A10: This attribute, too, is fairly distributed. We do not process it futher.
Attribute A11: This attribute consists of integers only. It, too, is highly right-skewed. For further analysis, we consider the logarithm of every variable in this attribute.
Attribute A12: Just like A9 and A10, this attribute too, has an equitable distribution. It is not processed further.
Attribute A13: A13 is a categorical variable. It has one major and two minor attributes. We combine the two variables with the lowest counts into one for our neural network.
Attribute A14: This attribute weakly resembles a gaussian distribution with two spikes at its ends. However, the number of instances which are zero at this attribute easily outnumber those at any other value. For this reason, we simply consider this to be a categorical value which is either zero or non-zero.
Attribute A15: This attribute is highly skewed towards the right. We transform this attribute by applying the logarithm function on it.

2.3 One Hot Encodings

The input variable to the neural network needs to be numerical. For this reason, we need to convert the categorical string variables to integers. A simple way to do so is to map every string to an integer. However, we avoided this approach when building the network because this may create unnecessary ordinal relationships between the variables. The drawback of converting to one-hot encodings is that the number of input neurons would increase significantly which could make the network performance poorer due to the curse of dimensionality.

After converting categorical variables to one hot vectors, the number of attributes increased to 26.

3 Hyperparameter Selection

We used a genetic algorithm to decide the hyperparameters of the network. The hyperparameters that were selected by the use of the genetic algorithm were the number of neurons in the hidden layer, the learning rate and the number of epochs. We use real valued representations for representing these three quantities. As with all genetic algorithms, the process consisted of the following stages.

Population Initialisation. Our population consisted of 20 different combinations of neural network parameters. These parameters were randomly chosen at the first iteration.

Fitness Calculation. Our fitness function was a weighted sum of the specificity and the accuracy. The weight assigned to the specificity was double the weight of the accuracy. This is because specificity is a better measure for performance as compared to the total accuracy. We discuss this in more detail in the next section. The choice of

the fitness function suggests that both the total accuracy and specificity need to be optimised, and yet optimising the specificity is more important than optimising the total accuracy.

Selection. We select two-fifths of the "fittest" population at every iteration. This subset is determined by running the network five times for every chromosome in the population. This subset is then used to produce offsprings. We also randomly retain some of the chromosomes in the subset of the population that were not the fittest (each with uniform probability of one-tenth). This creates variety in the genes of the population for the next generation.

Crossover. Parents are randomly selected from the subset that was created above. We use a uniform crossover method so that there is equal chance of the child inheriting the each of the parents' alleles. This process is continued until the desired population size is reached.

Mutation. We use uniform mutation to maintain genetic diversity between generations. The rate of mutation falls linearly with the number of generations. Thus, it is more likely that mutations would occur during the first few generations as opposed to the last few generations. This creates more diversity at the beginning and ensures that we have the best characteristics in the subsequent generations. The step sizes are dependent on the actual gene that is being mutated. For the number of epochs, it is 500; for the learning rate, it is 10; and it is 0.001 for the learning rate.

Termination. We do not create an explicit termination condition in terms of convergence of the fitness. This is because we are keeping a significant fraction of the chromosomes that are not among the fittest. Thus, at the expense of maintaining genetic diversity, the average fitness does not converge fast with each generation.

However, it was also observed that by the sixth or seventh generation, the fitness of the fittest individual does not change significantly. Thus, we stop the process after 10 generations, and choose the fittest individual for our neural network.

The choice of the values of the different percentages and step sizes in the above process was based upon experimentation. After ten generations, it was found that the optimal number of hidden neurons were 49, the optimal learning rate was 0.0021 and the optimal number of epochs were 5280.

4 Neural Network Architecture

We built three different neural networks- one that trains on the raw dataset, one that trains on the processed dataset with the hyperparameter values being set manually (NNP), and one that trains on the processed dataset in which the hyperparameter values were obtained from the genetic algorithm (NNPG).

NNP was built as a two layer feed-forward network with 40 hidden units, trained with back-propagation. The neural network is trained with Stochastic Gradient Descent as an optimiser, and its performance is measured using the cross-entropy loss. The choice of the loss function was mainly due to the reason that it lends itself well to classification problems. Varying the number of hidden neurons was experimented with; however, it did not bring about any significant change in the performance. Similarly, a three layered network was also tested with. It, too, did not yield a change in the accuracy of classification.

When the network was trained on the raw, unprocessed data instead, a hidden layer with 30 neurons was used instead. This is mainly because the unprocessed dataset consists of fewer input features than the processed one.

The architecture of NNPG differed from NNP in the number of hidden neurons. 49 hidden neurons was found to be more optimal (with a different combination of the learning rate and number of epochs trained) for NNPG.

4.1 Evaluation

The overall evaluation on the test set was performed by calculating accuracy which is calculated as follows

$$\frac{tp + tn}{tp + fp + tn + fn}$$

where
tp = Number of true positives
fp = Number of false positives
tn = Number of true negatives
fn = Number of false negatives

Accuracy was chosen for evaluating the network because its invariance under exchanges between positives and negatives makes it a robust measure to evaluate performance of the network [3].

We also evaluate the Type I accuracy (specificity) in the output obtained from the output of the network. When analysing credit risk, it is important to have a high specificity because lending credit to a person/institution who is unable to repay the amount creates much more of a financial burden than not lending credit to person who would be able to repay it from the point of view of the credit issuer. Specificity is calculated using the following formula

$$\frac{tn}{fp + tn}.$$

The symbols have the same meaning as before.

5 Results and Discussion

5.1 Setup

The dataset was randomly split into 80% training set and 20% testing set. NNP was run for a total of 4000 epochs with a learning rate of 0.005. Increasing the number of epochs any more than that led an increase in the training loss which is indicative of over-fitting.

Inspired by the results of the genetic algorithm, we trained NNPG for 5280 epochs with a learning rate of 0.0021. It was also manually verified that the training the network for 5280 epochs led to no overfitting.

5.2 Results

Wang et al. in [4] create a fuzzy Support Vector Machine to evaluate credit risk. As the divide for good and bad creditors is often arbitrary, they treat every sample as positive and negative, but with different memberships. This way, their prediction model is expected to have a better generalisation ability. In order to compare the performance of their fuzzy support vector machine, they implement a few other prediction models that are run on various Credit Risk datasets. We compare the performance of our neural network on pre-processed data against our neural network on the raw dataset (with minimal pre-processing), and also against two of the implementations in their work- their neural network and their B-FSVM Logit Regression which uses a polynomial basis function.

The results of the four different experiments can be summarised below. We use NNW to denote Wang et al.'s neural network.

	Type I (Specificity)	Accuracy
NNU	80.69%	73.48%
NNP	90.28%	84.78%
NNPG	92.15%	85.32%
NNW	80.32%	81.45%
B-FSVM	82.70%	83.94%

Table 1: Performance comparisons using various classifiers averaged over 20 runs

5.3 Discussion

It can be observed from Table 1 that there was a tremendous increase in the overall accuracy and the Type-1 Performance when the neural network was run on the pre-processed data as compared to the raw data. The better performance in this case can be solely attributed to the data pre-processing stage. The increase in performance is indicative of the effectiveness of data-preprocessing.

It was also noticed that our neural network that was run on pre-processed data also performed significantly better than NNW and the B-FSVM.

NNW and NNU differ in the fact that the categorical variables in NNW are encoded using the one-hot representation whereas, those in the NNU are represented using integer encoding. It can be seen that this alone creates a big difference in the overall accuracy but less so in the Type I accuracy. The transformation of attributes that was performed while data pre-processing contributed a great deal in improving accuracy and the specificity, as is evident from the comparison of NNP and NNW.

It should also be noted that even though NNP outperformed B-FSVM, the latter achieved a comparable accuracy without much pre-processing (converting categorical variables to one hot encodings was the only pre-processing performed). This shows that there is a possibility of the B-FSVM outperforming NNP if it, too, was trained on pre-processed data. Moreover, this also indicates the fact that NNP is heavily tied to a particular dataset and does not generalise well (i.e. every dataset needs to pre-processed differently whereas, B-FSVM can produce impressive results with minimal pre-processing).

Lastly, it was also observed that NNPG performed better than all other classifiers both with respect to the total accuracy and specificity. Specifically, after comparing to NNP, approximately a 2% increase in specificity and 0.5% increase in total accuracy was observed. This demonstrates that the use of a genetic algorithm was helpful in determining the hyperparameters for the neural network. The higher increase in the specificity as compared to the total accuracy can be attributed to the choice of the fitness function wherein the specificity was given higher importance than total accuracy.

6 Future Work

A number of different techniques were used when pre-processing the data. Although the results obtained from these techniques have been better than existing result, there is still some scope for improvement in the data pre-processing front.

Reverse Engineering Attributes: By comparing similar existing datasets where the information is not anonymised, it may be possible to reverse engineer the attrbiutes and their varaibles. Obtaining the original attribute and variable names could be immensely useful as the transformations that would be applied to attributes need not be "guessed" by analysing their distributions. Instead, one would be able to apply standard transformations for those particular attributes. This would result in lesser bias and potentially higher accuracy.

Predicting Missing Values: Missing values may be predicted using either linear regression or clustering methods such as k-NN. Although, not quite accurate, the results from these would likely be less biased than the existing approach of replacing them with the mode.

Collecting More Data: Although it is outside the scope of this study, the results could potentially improve if more data is gathered. The right skewness in some of the attributes seem to suggest that majority of the instances in the dataset belong to a single income stratum. Instances from other income strata seem to be under-represented in this dataset.

Introducing More Hyperparameters: It is also possible to introduce more hyperparameters so that they can be optimised by the genetic algorithm. Some of them could include the number of hidden layers, the choice of optimiser etc. It is possible that optimising these quantities would lead to a better performance for the network.

7 Conclusion

This paper explored the effects of the performance of a neural network when it is trained on processed and unprocessed dataset. When experimented on a credit risk dataset, there was almost an 18% increase in overall accuracy and an 11% increase in specificity.

Some of the pre-processing techniques were quite standard such as applying the logarithm in a right-skewed distribution to reduce the effect of outliers. Many of them, such as combining categorical variables with low frequencies were inspired by [5], and some of the techniques used (such as replacing missing values with a sample from a gaussian distribution) were novel. Altogether, they were able to perform better than other existing techniques as well.

It was also observed that choosing hyperparameters using genetic algorithm brought about an increase in the performance of the network.

References

1. Hernandez, M.A., Stolfo, S.J.: Data Mining and Knowledge Discovery (1998). https://doi.org/10.1023/A:1009761603038
2. Dua, D., Karra Taniskidou, E. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science (2017).
3. Sokolova M., Lapalme G.: A systematic analysis of performance measures for classification tasks, Information Process Management. 45, 427–437 (2009)
4. Wang Y., Wang S., Lai K. K.: A new fuzzy support vector machine to evaluate credit risk. IEEE Transactions on Fuzzy Systems. 13, 820-831 (2005)
5. Bustos R.A., Gedeon T.D.: Decrypting Neural Network Data: A Gis Case Study. Artificial Neural Nets and Genetic Algorithms. Springer, Vienna (1995)
6. Haupt, Randy L. Practical Genetic Algorithms 2nd ed. A Wiley-Interscience publication., 1998
7. Magnani, M.: Techniques for dealing with missing data in knowledge discovery tasks. Available from http://magnanim.web.cs.unibo.it/data/pdf/missingdata.pdf, Version of June 2004.