Combining fuzzy clustering and artificial neural networks for financial performance predictions

Adrian Costea, Ph.D.
MSCA Digital Finance 2025
University of Twente, The Netherlands
05.02.2025

Presentation outline

1. A generic classification model
2. Preliminary steps (build the *class* performance variable)
   - 2.1. Fuzzy C-Means algorithm
   - 2.2. Modified Fuzzy C-Means algorithms
3. Determine the ANN architecture
4. RT-based and GA-based ANN training
   - 4.1. RT-based ANN training
   - 4.2. GA-based ANN training
5. The NFIs' performance dataset
6. Experiment
7. Conclusions

1. A generic classification model

- Research problem

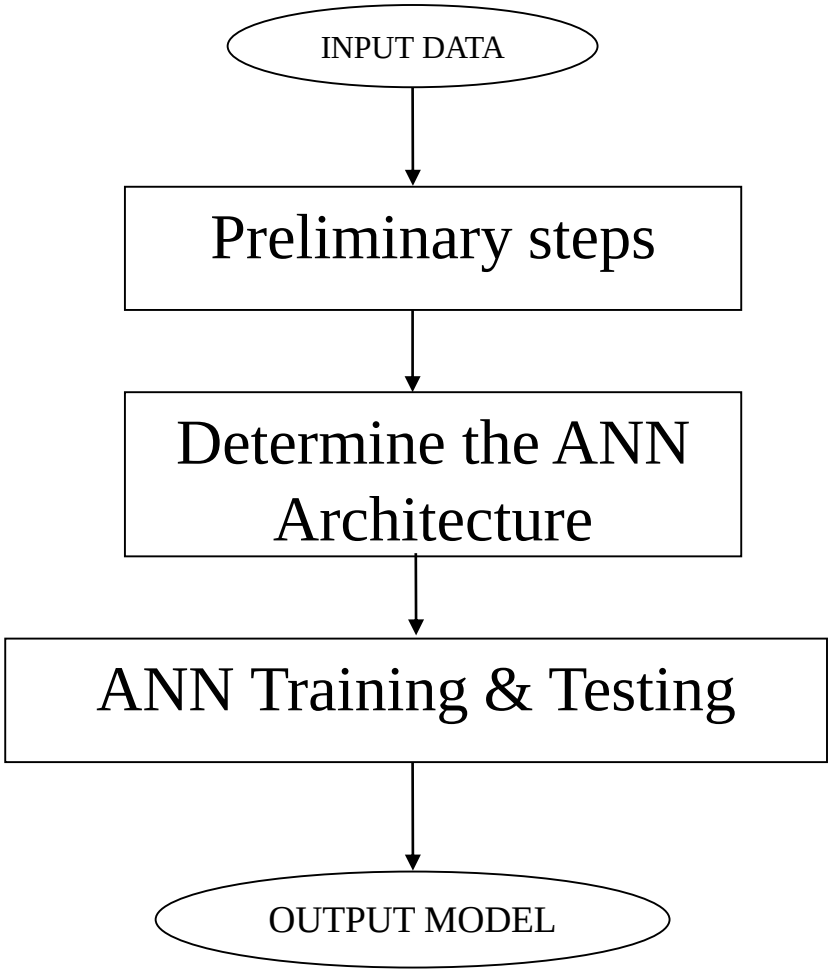
Evaluate comparatively the performance of non-banking financial institutions (NFIs) in Romania and make performance predictions

- Methodology:

We use Fuzzy C-Means (FCM) algorithm to describe the current situation of NFIs' sector (Data Mining description goal) and build the *performance class* variable.

We applied feed-forward neural networks trained by gradient-descent and genetic algorithms in order to map the input space to the newly created performance class variable. We want to be able to *predict* the performance of different NFIs as data become available.

1. The generic classification model based on neural approaches



2.1. Fuzzy C-Means clustering

- In contrast to traditional clustering methods, fuzzy clustering methods assign different membership degrees to the elements in the data set indicating in which degree the observation belongs to every cluster.
- One traditional method in fuzzy clustering is the fuzzy C-means clustering method (FCM). The algorithm partitions a multidimensional data set into a specific number of clusters, giving a membership degree for every observation in every cluster.

- The algorithm minimizes the following objective function:

$$J_m(U,v) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m (d_{ik})^2$$

$$d_{ik} = \|x_k - v_i\| = \left[\sum_{j=1}^{p}(x_{kj} - v_{ij})^2\right]^{1/2}$$

is the Euclidean distance between the cluster center v_i and observation x_k for p attributes (financial ratios in our case), $m \in [1,\infty)$ is the weighting exponent, and the following constraint holds

$$\sum_{i=1}^{c} u_{ik} = 1$$

2.1. Fuzzy C-Means clustering

If m and c are fixed parameters then, by the Lagrange multipliers, $J_m(U, v)$ may be globally minimal for (U, v) only if

$$u_{ik} = \frac{1}{\sum_{r=1}^{c} \left(\frac{d_{ik}}{d_{rk}} \right)^{2/(m-1)}}; \quad \forall 1 \le i \le c, 1 \le k \le n$$

and

$$v_{ij} = \frac{\sum_{k=1}^{n} (u_{ik})^m x_{kj}}{\sum_{k=1}^{n} (u_{ik})^m}; \quad \forall 1 \le i \le c, 1 \le j \le p$$

2.1. Fuzzy C-Means clustering

The algorithm follows the following steps:

-Step 1. Fix c, $2 \le c \le n$, and m, $1 \le m \le \infty$. Initialize $U^{(0)} \in M_{fc}$

Then, for s^{th} iteration, $s = 0, 1, 2, \ldots$:

-Step 2. Calculate the c fuzzy cluster centers $\{v_i^{(s)}\}$.

-Step 3. Calculate $U^{(s+1)}$.

-Step 4. Compare $U^{(s+1)}$ to $U^{(s)}$: if $\left\| U^{(s+1)} - U^{(s)} \right\| \le \varepsilon$ stop; otherwise return to Step 2.

2.1. Fuzzy C-Means clustering

- Problem:

  How to allocate in clusters the so-called "uncertain" observations. These are observations for which the two highest membership degrees are closed to each other (e.g., 0.45 and 0.48).

- Solution:

  The introduction in the objective function some kind of information about the characteristics of every cluster so that uncertain observations are better allocated.

  In order to characterize each cluster we construct so called "linguistic variables" out of initial quantitative ones.

2.1. Fuzzy C-Means clustering

- Linguistic variables are quantitative fuzzy variables whose states are fuzzy numbers that represent linguistic terms, such as *very small, medium*, and so on.

- In our study we model the eight financial ratios with the help of eight linguistic variables using five linguistic terms: *very low* (VL), *low* (L), *average* (A), *high* (H), *very high* (VH).

- To each of the basic linguistic terms we assign one of five fuzzy numbers, whose membership functions are defined on the range of the ratios in the data set.

2.1. Fuzzy C-Means clustering

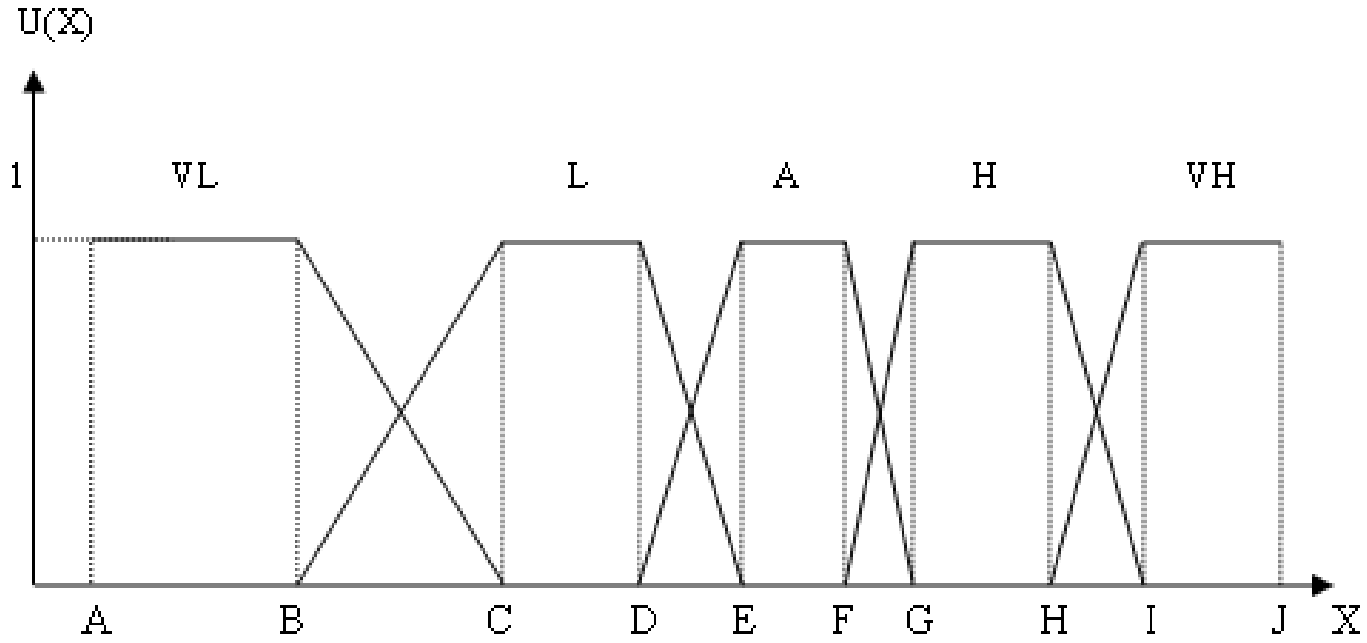
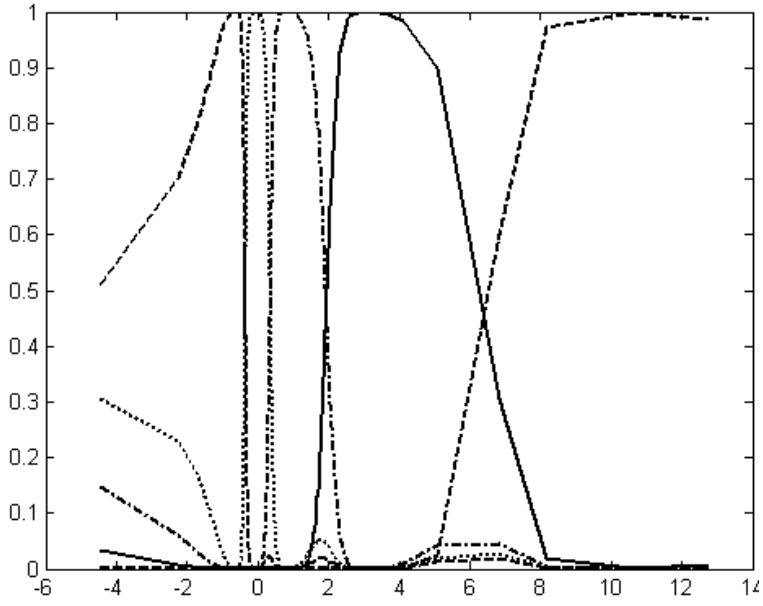
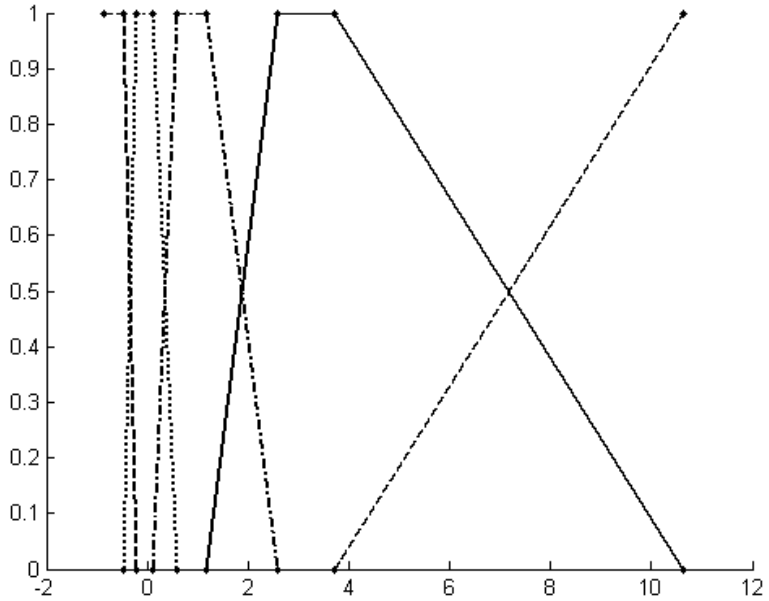


Figure 1. The trapezoidal representation of the five linguistic terms (VL, L, A, H, VH) for a generic variable X.

The minimum/maximum point for the linguistic term LT is defined as the minimum/maximum value for that ratio in the entire data set for which the membership degree in the class of linguistic term LT is greater or equal to 0.99.

2.1. Fuzzy C-Means clustering





Linguistic variable representation of activity cost and its trapezoidal approximation

2.2. Modified FCM

- Once we have the linguistic variables for all financial ratios in our data set, we can obtain an importance coefficient (weight) for every ratio in every cluster and introduce it in the clustering algorithm.

$$SVC_{ij} = \frac{VC_{ij}}{\sum_{j=1}^{p} VC_{ij}} \qquad VC_{ij} = \frac{standard\_deviation(perc_{ij})}{mean(perc_{ij})}$$

$$perc_{ij}(k) = \frac{nr\_of\ occurences\ of\ LT\ k\ for\ ratio\ j\ in\ cluster\ i}{nr\_of\ samples\ in\ cluster\ i}$$

$perc_{ij}(k)$ will denote the percentage of occurrences of linguistic term (*LT*) *k* for ratio *j* in cluster *i*.

2.2. Modified FCM

The previous weights are introduced in the Euclidean distance term of the FCM algorithm in the following form:

$$d_{ik} = \left[\sum_{j=1}^{p} (x_{kj} - v_{ij})^2 SVC_{ij} \right]^{1/2}$$

At each iteration *s* we should find the membership degrees that minimize the following objective function:

$$J_m(U, v) = \sum_{k \in I} \sum_{i=1}^{c} (u_{ik}^{(s)})^m (d_{ik}^{(s)})^2 \left(1 - u_{ik}^{(s-1)}\right)$$

where *I* is the set of certain observations in iteration *s* and $u_{ik}^{(s-1)}$ is the membership degree of the certain observation *k* for cluster *i* corresponding to the previous iteration.

2.2. Modified FCM

If m and c are fixed parameters then, by the Lagrange multipliers, $J_m(U, v)$ may be globally minimal for (U, v) only if

$$u_{ik}^{(s)} = 1 \Big/ \sum_{r=1}^{c} \left(\frac{(d_{ik}^{(s)})^2 \left(1 - u_{ik}^{(s-1)}\right)}{(d_{rk}^{(s)})^2 \left(1 - u_{rk}^{(s-1)}\right)} \right)^{(1/(m-1))}$$

and

$$v_{ij}^{(s)} = \frac{\sum_{k \in I} (u_{ik}^{(s)})^m \left(1 - u_{ik}^{(s-1)}\right) x_{kj}}{\sum_{k \in I} (u_{ik}^{(s)})^m \left(1 - u_{ik}^{(s-1)}\right)}$$

We use the above equations to update the centers and membership degrees in our algorithm.

3. Determine the ANN architecture

- The most widely used neural network is the Multilayer Perceptron (MLP) trained by back-propagation or other gradient-descent algorithms.
- number of output neurons: as many output neurons as the number of classes or just one neuron in the output layer. The latter approach would enable us to eliminate the risk that one class, corresponding to a higher value of the output variable (e.g., 4) is more important than classes with lower values (e.g., 1, 2 and 3).
- choosing the number of hidden layers and the number of neurons in each hidden layer: depend on “input/output vector sizes, size of training and test subsets, and, more importantly, the problem of non-linearity” (Basheer & Hajmeer 2000, p. 22);
- Lachtermacher & Fuller (1995): for one output: $0.11NTRN < NH(NI+1) < 0.30NTRN$
- Upadhyaya & Eryurek (1992): $Nw = NTRN\ log2(NTRN)$
- we followed Basheer & Hajmeer’s (2000, p. 23) advice that “the most popular approach to finding the optimal number of hidden nodes is by trial and error with one of the above rules”;
- Finally, we chose Lachtermacher & Fuller (1995) rule.

3. Determine the ANN architecture

- We did not take into consideration three hidden layer cases because of the restriction imposed by the number of cases per weights ratio;
- We have used the sigmoid and linear activation functions for the hidden and output layers respectively;
- There is a negative correlation between error rates and the convergence speed. One should seek a compromise between these two factors.
- Scaled Conjugate Gradient (SCG) algorithm (Moller, 1993) is not the fastest algorithm (as Levenberg-Marquardt in some situations), but works very efficiently for networks with a large number of weights.
- We apply the early stopping method (*validation stop*) during the training process in order to avoid *the over-fitting* phenomenon. The validation stop method implies separating the training set into two parts: an effective training set (*TRe*) and a validation set (*VAL*). The training process stops when the difference between the effective training error and the validation error exceeds a certain threshold.

4.1. RT-based ANN training

- Once we determine the ANN architecture (with the corresponding set of weights), the next step is to train the network. The first training mechanism is a retraining-based ANN (own ref., 2004), briefly described next:
- Start with a network with an initial set of weights from the previous step (Determining ANN architecture) as the reference network;
- Perform L runs to improve the ANN classification accuracy. After each experiment we save the best set of weights (the solution) in terms of classification accuracy. Each experiment consists of:
  - Reduction of the weights of the current best network with successive values of scaling factor γ ($\gamma = 0.1, 0.2, \cdots, 0.9$).

    Retrain the ANN with the new weights and obtain 9 accuracy rates.
  - Choose the best network from the above 9 in terms of classification accuracy.
  - Compare the accuracy rate of the current network with that obtained in the previous step and save the best one for the next run as the current best network.

4.1. RT-based ANN training

- Depending on the splitting of the training set (TR) in the effective training set (TRe) and validation set (VAL) we have 3 types of retraining mechanisms:
  1. RT1 where TRe and VAL are common for all of the L runs,
  2. RT2 where TRe and VAL are different for each run, but the same for all 9 reduction weights trainings (second step of the experiment),
  3. RT3 where TRe and VAL are distinct for each training.

- We have 4 types of accuracy rates: training accuracy rate (ACR_{TRe}), validation accuracy rate (ACR_{VAL}), total training (effective training + validation) accuracy rate (ACR_{TR}) and test accuracy rate (ACR_{TS}). Correspondingly, we calculate 4 mean square errors: MSE_{TRe}, MSE_{VAL}, MSE_{TR}, and MSE_{TS}. In total 5 runs (L = 5) were conducted resulting in 5*9 = 45 new trainings for each type of retraining mechanism.

4.2. GA-based ANN training

- A different way of learning the connection weights of an ANN is by using a genetic algorithm.
- In the GA-based ANN training framework, the GA's chromosome or solution is given by the set of ANN weights after training and it is represented as a vector.
- In our models, the size of the population represents a parameter and it was set to *PS* = 20.
- Even if we start with a small initial population composed by 20 chromosomes, one run of the GA-based ANN training mechanism corresponding to 1000 generations takes up to 1 hour.
- If this is firstly multiplied by 10 runs for each GA experiment and then by 4 different crossover operators, it results a total of 40 hours for training all GA-based ANNs.

4.2. GA-based ANN training

- Firstly, the elitism technique is applied, in order to insert the best N_{elite} chromosomes in terms of ACR_{TR} into the new population.
- The remaining 20-N_{elite} chromosomes are selected on the basis of the probability of selection for each chromosome, expressed by the *roulette wheel* procedure:

$$P_i = ACR_{TR}^{(i)} / \sum_{i=1}^{PS} ACR_{TR}^{(i)}$$

- The chance of a chromosome to be drawn into the new population is directly correlated to the probability P_i corresponding to this chromosome.

4.2. GA-based ANN training

- Next, as the probability of crossover is set to $P_c = 0.80$, it results that 80 percent of the chromosomes previously obtained are randomly selected for mating.
- In our approach three crossover operators proposed by Pendharkar & Rodger (2004), namely one-point, arithmetic, and uniform crossover will be used, in addition to the one proposed by Costea & Nastac (2005), namely multi-point crossover.
- Let us denote by l the length of the chromosomes and by C_1 and C_2 two parent chromosomes:

$$C_1 = c_{11}, c_{12}, \cdots, c_{1l}$$
$$C_2 = c_{21}, c_{22}, \cdots, c_{2l}$$

One-point crossover

For each pair of chromosomes a random integer $i, i \in \{1, l\}$ will be generated. The new born offsprings are constructed using the mechanism given by: $O_1 = c_{11}, c_{12}, \cdots, c_{1i}, c_{2,i+1}, \cdots, c_{2l}$ and $O_2 = c_{21}, c_{22}, \cdots, c_{2i}, c_{1,i+1}, \cdots, c_{1l}$.

Multi-point crossover

The chromosomes are split into n parts. The number of splitting points n is randomly generated so that $2 \leq n \leq maxsplit$. After that, n distinct random numbers $i_1, i_2, \cdots, i_n$ are generated, with $i_k \in \{1, l\}$ and $i_1 < i_2 < \cdots < i_n$.The two offsprings (for $n = 2$) are: $O_1 = c_{11}, c_{12}, \cdots, c_{1i_1}, c_{2,i_1+1}, \cdots, c_{2i_2}, c_{1,i_2+1}, \cdots, c_{1l}$ and $O_2 = c_{21}, c_{22}, \cdots, c_{2i_1}, c_{1,i_1+1}, \cdots, c_{1i_2}, c_{2,i_2+1}, \cdots, c_{2l}$.

4.2. GA-based ANN training

Arithmetic crossover

Firstly, the parent-chromosomes are split into n parts, as in the case of multi-point crossover. The children genes are given by a convex combination of the parents' genes. For $n = 2$, we have:

$$O_1 = \begin{cases} \alpha \times c_{1k} + (1-\alpha) \times c_{2k}, k \in \{1, \cdots, i_1\} \\ (1-\alpha) \times c_{1k} + \alpha \times c_{2k}, k \in \{i_1 + 1, \cdots, i_2\} \\ \alpha \times c_{1k} + (1-\alpha) \times c_{2k}, k \in \{i_2 + 1, \cdots, l\} \end{cases}$$

$$O_2 = \begin{cases} (1-\alpha) \times c_{1k} + \alpha \times c_{2k}, k \in \{1, \cdots, i_1\} \\ \alpha \times c_{1k} + (1-\alpha) \times c_{2k}, k \in \{i_1 + 1, \cdots, i_2\} \\ (1-\alpha) \times c_{1k} + \alpha \times c_{2k}, k \in \{i_2 + 1, \cdots, l\} \end{cases}$$

where $\alpha \in [0,1]$ represents a random number, generated for each chromosome-pair.

Uniform crossover

A random number $\alpha \in [0,1]$ is generated for each pair of genes of the parent-chromosomes. If $\alpha < 0.50$ the gene of the first parent will be given to the first child and the gene of the second parent will be given to the second child. Otherwise, the genes are inversed.

The offsprings (children chromosomes) are *added* to the population and the size of the population becomes $PS' > PS$.

4.2. GA-based ANN training

- Next, the uniform mutation operator is applied for all chromosomes in PS'.
- The probability of mutation is set to $P_m = 0.01$, which will be interpreted by the fact that approximately one percent of the genes will mutate for each chromosome. The choice of this probability is based upon "theoretically optimal value – 3/chrom length" (Tuson & Ross, 1998) since our chroms' length was 192 (8 financial ratios x 3 neurons in the first hidden layer x 2 neurons in the second hidden layer x 4 neurons on the output layer): $3/192 \approx 0.01$.
- Then a random number $\alpha \in [0,1]$ is generated for each gene of each chromosome.
- The new gene is randomly generated within the variable domain if the condition $\alpha \leq P_m$ is fulfilled. Otherwise, the gene will not be changed.
- The values are changed in the range of the existing values using linear interpolation: *new_value* = *min* + (*max* - *min*) x *rand*(1), where min and max represent the minimum (adjusted with -0.01) and maximum (adjusted with +0.01) values for that particular chromosome and *rand*(1) returns a random number between 0 and 1.
- The chromosomes which changed at least one gene are *added* to the population, obtaining $PS'' > PS' > PS$.

4.2. GA-based ANN training

- The final step for constructing the new population consists in reducing the size of the population to 20 chromosomes. The best 20 chromosomes are selected from *PS''* in terms of ACR_{TR} with the condition that one chromosome can have no more than *maxlim* duplicates. The mutation operator is used to generate more chromosomes if the number of best chromosomes which satisfy the above condition is less than 20.

- Except the training accuracy rate ACR_{TR}, in the literature there are different methods used to evaluate the performance of classification algorithms. In our experiments we used out-of-sample accuracy rate (ACR_{TS}) to evaluate more rigorously our models.

- In conclusion, excluding crossover operator, the parameters of our GA model are given by:
  - the number of generations (N_{gen}),
  - the population size (*PS*),
  - the number of elite chromosomes (N_{elite}),
  - the maximum number of splitting points (*maxsplit*) for the multi-point crossover operator,
  - the probability of crossover (P_c),
  - the probability of mutation (P_m), and
  - the maximum number of duplicates for the chromosomes (*maxlim*).

5. NFIs' performance dataset

The system of indicators for evaluating the performance of NFIs:

A. Assessing the degree of capitalization (C):

- Equity ratio (Leverage) = own capital / total assets (net value)
- Own capital / equity
- Indebtedness sources = borrowings / own capital

B. Assessing the assets' quality (A):

- Loans granted to clients (net value) / total assets (net value)
- Loan granted to clients (net value) / total borrowings
- Past due and doubtful loans (net value) / total loans portfolio (net value)
- Past due and doubtful claims (net value) / total assets (net value)
- Past due and doubtful claims (net value) / own capital

C. Assessing profitability (P):

- Return on assets (ROA) = net income / total assets (net value)
- Return on equity (ROE) = Net profit / own capital
- The rate of profit = gross profit / total revenues
- Activity cost = total costs / total revenues

5. NFIs' performance dataset

- We collected the data for these 65 NFIs quarterly for 4 years, obtaining a total of 769 observations.

- Then, the above twelve financial ratios are computed.

- Four ratios, namely *Leverage* (for the capital adequacy dimension), *Loans_to_Assets* and *Loans_to_Borrowings* (for the assets' quality dimension) and *ROA* (for profitability dimension) were discarded from our analysis, due to the high variation of their values or incorrect values, remaining with eight ratios.

- The data set composed by 769 observations multiplied by 8 ratios contains quarterly and yearly averages (16 quarterly averages and 4 yearly averages = 20 observations).

5. NFIs' performance dataset

The ratios used were:

- for capital adequacy dimension
  - *Own capital / equity (OC_to_EQ)*
  - *Indebtedness sources = borrowings / own capital (BOR_to_OC)*
- for assets' quality dimension
  - *Past due and doubtful loans (net value) / total loans portfolio (net value) (PDDL_to_Loans)*
  - *Past due and doubtful claims (net value) / total assets (net value) (PDDC_to_Assets)*
  - *Past due and doubtful claims (net value) / own capital (PDDC_to_OC)*
- for profitability dimension
  - *Return on equity = net profit / own capital (ROE)*
  - *The rate of profit = gross profit / total revenues (GP_to_REV)*
  - *Activity cost = total costs / total revenues (Costs_to_REV)*

.

6. Experiment – applying FCM

- Our data set was pre-processed by levelling the outliers to the domain [-50, 50] and also by normalizing each ratio, i.e. subtracting the mean from each value and dividing the result by the standard deviation of the ratio, in order to avoid our techniques results being affected by these extremes/abnormal financial ratios.

- FCM parameters: n=4; m = 1.5, no_of_iterations = 10000, the limit for the stopping criterion = 0.00001.

- After we characterize the clusters we re-allocated the uncertain observations within the clusters based on the procedure we presented in Section 2.2 and we obtained the structure of the clusters.

- cluster 1 - Best (54 observations), cluster 2 - Average (131 observations), cluster 3 - Bad (503 observations), and cluster 4 - Worst (81 observations).

- based on the clusterization we have constructed the class variable assigning a class value (1 - average to 4 - bad) to each observation within a cluster.

6. Experiment – applying FCM

	OC_to_EQ	*BOR_to_OC*	*PDDL_to_Loans*	*PDDC_to_Assets*	*PDDC_to_OC*	*ROE*	*GP_to_REV*	*Costs_to_REV*	***Order***	***No. of certain observations***
Cluster 1	A	VH	VL	VL	L	A	H	L	Average	113
Cluster 2	VL	L	VL	VL	L	A	H	L	Worst	62
Cluster 3	A	A	H	A	A	A	H	-	Best	44
Cluster 4	A	A	VL	VL	L	A	H	L	Bad	451

We considered that one linguistic term characterizes one cluster if it represents more than 40% out of total number of samples for that cluster. We chose 40% in order to allow maximum two linguistic term to characterize a cluster for each ratio. For example, for cluster 1, and ratio *OC_to_EQ*, we have one linguistic terms that has more than 40% of the occurrences (A). When all linguistic terms for one cluster and one ratio are under 40% we say that the ratio is not a good definer for that specific cluster. It seems that “Activity cost” ratio (*Costs_to_REV*) is not a good definer for the third cluster.

6. Experiment – applying FCM

- After Step 1 of the algorithm we obtained 99 uncertain observations, while the remaining 670 certain observations were distributed among different clusters as shown in the following Table (column “Step 1”).
- We can see that our algorithm had allocated all uncertain observations.

	Step 1	**Normal FCM**	**Modified FCM**
Cluster 1	113	142	131
Cluster 2	62	82	81
Cluster 3	44	52	54
Cluster 4	451	493	503
Total	670	769	769

6. Experiment – applying FCM

Table 4. The comparison of different clustering techniques for assessing NFIs' financial performance

Technique	Ease-of-implementation	Determination of attribute importance (relevance)	Linguistic characterization of clusters	Pattern allocation (in terms of correspondence to reality)
C-Means	Best	Not possible	Not possible	Worst
Fuzzy C-Means	Second Best	Possible	Objective: automatic characterization of the clusters based on the linguistic terms.	Second best
Self-Organising Maps	Worst. Highly parametrized.	Not possible	Subjective: based on the individual feature planes, the distances between neurons and analyst interpretation.	Third best
Modified Fuzzy C-Means	Third best	Possible	Objective: automatic characterization of the clusters based on the linguistic terms. A more robust linguistic characterization than Fuzzy C-Means: based only on the certain observations.	Best

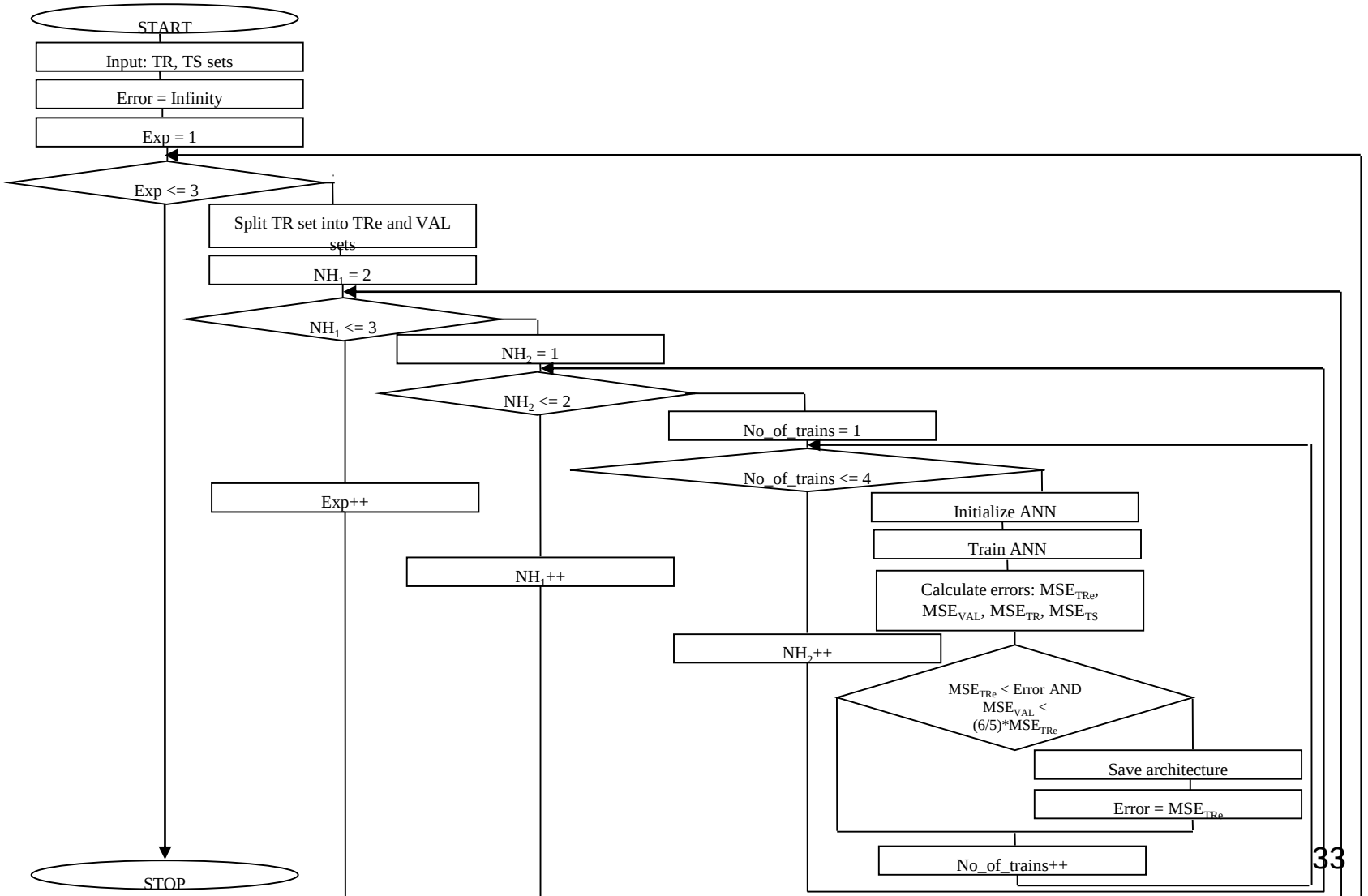
6. Experiment – applying ANNs

- We selected 54 observations (the number of observations in the smallest cluster - Best) from each cluster, totalling 216 observations for building the classification model.

- We split the data in testing and training (192 observations vs. 24 observations).

- The next step of the methodology was to determine the proper architecture for the ANN-based classification model:
  - ANN parameters: the learning algorithm (SCG), the performance goal of the classifier (1e-5), the maximum number of epochs (30000)
  - We vary the number of hidden neurons in the first (NH1: 2-3) and second (NH2: 1-2) hidden layers.
  - 192 observations further divided: 168 effective training and the rest (24) for validation.

6. Experiment – applying ANNs



6. Experiment – applying ANNs

- We have applied the Lachtermacher and Fuller (1995) rule and varied *NH* from 3 (this value has to be greater than 0.11*168/(8+1)=2.05) to 5 (this value has to be lower than 0.30*168/(8+1)=5.60).

- For the two-hidden layer case we have selected more neurons on the first hidden-layer than on the second hidden-layer, simulating a rough and a fine-tunning training phases (we varied NH_1 from 2 to 3 and NH_2 from 1 to 2).

- Then, the network was trained for each ANN architecture, based on the effective training dataset. The best ANN architecture in terms of mean squared error for the effective training dataset (MSE_{TRe}) was saved if the mean squared error based on the validation set (MSE_{VAL}) is less than 6/5*MSE_{TRe}. This restriction has been imposed in order to avoid saving ANN architectures with large differences of the mean squared error between effective training and validation.

- The final ANN architecture consists of 3 neurons on the first hidden layer and 2 neurons on the second hidden layer. The following accuracy rates were obtained: effective training dataset accuracy rate (ACR_{TRe}) = 93.41 percent, validation dataset accuracy rate (ACR_{VAL}) = 96.00 percent, total training dataset accuracy rate (ACR_{TR}) = 93.75 percent and testing dataset accuracy rate (ACR_{TS}) = 91.67 percent.

6. Experiment – applying GA-based ANNs

- Next, the GAs were applied for training the previously obtained network, in order to improve the accuracy rates. The values of the parameters used are given by: the number of generations $N_{gen} = 1000$, population size $PS = 20$, number of elite chromosomes $N_{elite} = 3$, maximum number of splitting points $maxsplit = 5$, probability of crossover $P_c = 0.80$, probability of mutation $P_m = 0.01$, and maximum number of duplicates for the chromosomes $maxlim = 1$.

- In our experiments we used both parental and offspring mutation by applying mutation to both parents and their offsprings.

- We run our GA algorithm 10 times, each time saving the best solution in terms of training dataset accuracy rate (ACR_{TR}). Each of the 10 runs consisted of 1000 iterations of the algorithm. Each iteration started with the application of the elitist approach (the first 3 chromosomes in terms of ACR_{TR} were selected into the new population). Then, we used one of the four crossover operators (one-point, multi-point, arithmetic and uniform) and $P_c = 0.80$ to obtain new offsprings (usually, we obtained 0.8x20=16 new offsprings), which were added to the initial population, obtaining PS'.

- Then, we applied the mutation operator ($P_m = 0.01$) to change 1% of the values of the chromosomes and the new chromosomes are added to the new population, obtaining PS''. Next, we evaluate all the chromosomes in PS'' and check whether the solution has been improved. Finally, we select into the new population the best 20 chromosomes from PS'' in terms of the highest ACR_{TR}, not allowing into the new population any duplicate chromosome ($maxlim = 1$).

6. Experiment – applying GA-based ANNs

- We performed four experiments by applying the four crossover operators presented in the previous sections, to check the findings in related literature (e.g., Pendharkar, Rodger 2004; Costea, Nastac 2005) which state that there is no or low correlation between the type of crossover operator and the GA performance. As reported elsewhere, we found no correlation. After a total of 40000 iterations (10 runs x 1000 iterations x 4 crossover operators), our genetic algorithm could not improve the initial solution, both training and testing accuracy rates remaining the same (ACR_{TR} = 93.75 percent and ACR_{TS} = 91.67 percent).

- However, all the chromosomes on the final populations presented the same performance: the genetic algorithm converged, but converged to the *initial* best solution. This result is not surprising, given the high accuracy rates of the solution in the initial population of chromosomes, obtained when we applied the gradient-descent training algorithm to train the neural-network model. Also, our findings validate what was already reported by other authors (e.g., Schaffer 1994): training the ANN using GAs is not as efficient as the classical ANN training (the training that uses gradient-descent algorithms).

- We are convinced that, with a proper setting of its parameters, our GA algorithm can considerably improve a moderately-high accuracy rate solution.

6. Experiment – applying GA-based ANNs

Table 5. The comparison of different classification techniques for assessing NFIs' financial performance

Technique	Ease-of-implementation	Result explanation capability	Pattern allocation (in terms of training accuracy)	Pattern allocation (in terms of testing accuracy)
Multinomial Logistic Regression	Third best	Second best	100.00	77.27 or 81.81
Decision Trees (C4.5 algorithm)	Second best	Best	87.90	81.80 or 88.30
Artificial Neural Networks	Best	Third best	93.75	91.67
Artificial Neural Networks trained with a Genetic Algorithm	Worst	Worst	The GA could not improve the ANN training accuracy	The GA could not improve the ANN testing accuracy

7. Conclusions

- We have applied Data Mining to formalize the process of assessing comparatively the performance of non-banking financial institutions in Romania.
- Methodology: clustering (FCM) for description and classification (ANNs) for prediction.
- FCM: based on the average caracteristics of the input variables we characterized each individual cluster (4 clusters as in our previous work).
- ANNs: we used Scaled Conjugate Gradient (a gradient-descent like algorithm) to initially train the network and determine the proper ANN arhitecture. Then we used a genetic algorithm to try improve the already trained network (the classification model).
- We obtained high training and testing accuracy rates. Small differences between these rates. The GA did not improve the performance of the ANN classifier. We found no correlation between the type of the crossover operator and the GA performance.
- Future research: check how retraining mechanism could improve the classification model performance and apply reinforcement learning specific algorithms to tackle the research problem.

Thank you!