# Project Report: Electricity Fraud Detection

Luís Sá Couto no 79078
Luís Pedro Borges no 78349
Cláudio Correia no 81959

Instituto Superior Técnico, Universidade de Lisboa
`luis.sa.couto@tecnico.ulisboa.pt`
`luis.borges@tecnico.ulisboa.pt`
`claudio.correia@tecnico.ulisboa.pt`

## 1 Introduction

Electricy frauds, in its many forms[1], are very common in developing countries, where electricity infrastructures are not as modern as in other developed countries. While electricy thefts cause financial loss, they are also a potentially life threatening hazard for those responsible for such frauds. One solution is the use of smart meters[2]. However, for this report, we are interested in the application of intelligent methods in the context of the Intelligent Decision and Control course at IST, for the task of detecting whether an electricy consumer is comitting a fraud or not, and if the consumer is, what kind of fraud is he comitting. Specifically, we will be experimenting with a *multi-layer perceptron*, a Takagi-Sugeno Fuzzy model, and an *ANFIS*, described in Sections 4,5, and 6, respectively. In order to estimate the optimal hyperparameters of the aforementioned models, we will define such an estimation as a search problem and we will solve it through genetic optimization. An explanation of genetic algorithms is provided in Section 3. The task and the dataset are explained in Section 2. Section 7 addresses the problem of the dataset class imbalance, with the creation of additional models which will aggregate all the previously proposed approaches to create a final prediction. A brief discussion of the results is shown in Section 8, and Section 9 contains the closing remarks of our work.

[1] https://lowvelder.co.za/354097/6-forms-of-electricity-theft/
[2] https://news.nationalgeographic.com/news/energy/2011/09/110913-smart-meters-for-electricity-theft/

# Table of Contents

## 2 Defining the Task and the Dataset

The objective of our work is to be able to use an intelligent system to classify whether an electricity consumer is committing a fraud or not. If the consumer is indeed commiting electricity theft, we still classify such fraud as a *simple fraud* or a *complex fraud*. In order to do so, we need a labeled dataset to train our models. Professor Joaquim Viegas[3] provided us with a dataset containing 60,360 data points. These data points are composed of six features, based on the intuition that if the theft starts on day $t$, the change of pattern should be reflected on a change of consumption behavior in comparison with the past, and if the theft has started before it should be reflected on a change of consumption behavior in comparison to similar consumers. There are three possible labels: 0 (i.e., consumer committed no fraud), 1 (i.e., consumer committed a simple fraud, detected by an abrupt consumption change), or 2 (i.e., consumer committed a complex fraud, detected by subtle pattern changes). Specifically, the six features as as follows:

1. $\boldsymbol{I_1}$: Indicator of consumption variation. Ratio between the consumption of the last day and the last 5 days.
2. $\boldsymbol{I_2^d}$: Indicator of hourly consumption pattern change. Relates the hourly pattern of a day with the mean hourly pattern of the 5 days before. Uses the euclidean distance, and changes in absolute consumption will be the most relevant for the indicator.
3. $\boldsymbol{I_2^c}$: Another indicator of hourly consumption pattern change. Uses the Pearson correlation, and changes of dynamic can be detected.
4. $\boldsymbol{I_3}$: Indicator of consumption difference in comparison to the set of 10 consumers with the greatest similarity when comparing characteristics. Compares the mean consumption of the last 5 days to the mean consumption for the same days for the consumers with the most similar characteristics.
5. $\boldsymbol{I_4^d}$: Indicator of hourly consumption pattern difference in comparison to the 10 consumers with the greatest similarity when comparing characteristics. Relates the mean hourly consumption of the last 5 days between consumers. Uses the euclidean distance, and changes in absolute consumption will be the most relevant for the indicator.
6. $\boldsymbol{I_4^c}$: Another indicator of the same type as $I_4^d$. Uses the Pearson correlation, and changes of dynamic can be detected.

Since there are no consumer repetitions between the data points, no additional care is necessary when splitting the data onto training, testing, and validation sets. However, there is one major problem: the dataset is highly imbalanced, with roughly 83% of the data points belonging to class 0, and the remaining 17% distributed relatively evenly between labels 1 and 2. See Figure 1 for a graphical representation.

In order to tackle this imbalance, we used 10-fold cross validation for model training and testing, and we further considered three different training sets:

[3] https://scholar.google.pt/citations?user=BDe3dzEAAAAJ&hl=pt-PT

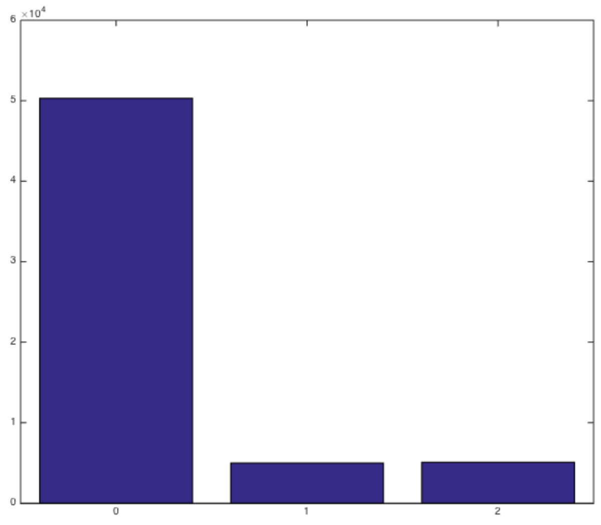


**Fig. 1.** The class distribution over the dataset

- **Regular dataset**: The whole and imbalanced original dataset
- **Balanced dataset**: A balanced subset (i.e., the three classes represented equally) of the original dataset
- **Fraud dataset**: A fraud subset (i.e., only label 1 and label 2 are represented) of the original subset

The main use of the aforementioned datasets will be detailed later in this report. The cross-validation process works over 10 iterations. Firstly, we split the original dataset in 10 parts. In each iteration, we choose one part to serve as a testing set and the other nine parts serve as the training set, which takes the form of one of the three previously explained training sets. After training and testing, we choose a different test set, and we repeat. Measurements such as the accuracy or the True Positive Rate are then averaged over the 10 iterations.

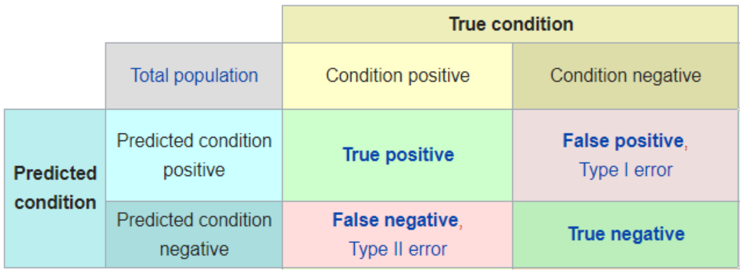
Given the previously mentioned imbalance, the accuracy measure is not the best to evaluate a model – a baseline model always outputting the label 0 would achieve an 83 % accuracy. Therefore, we considered not only the accuracy, but also the True Positive Rate (TPR) and the True Negative Rate (TNR). Accuracy measures the number of correct outputs of a system divided by the total number of outputs. TPR measures the proportion of positives that are correctly identified as such, while TNR measure the proportion of negatives that are correctly identified as such. The equations are as follows:

$$\textbf{Accuracy} = \frac{TruePositives + TrueNegatives}{TruePositives + FalseNegatives + FalsePositives + TrueNegatives} \tag{1}$$

$$\textbf{True Positive Rate} = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2}$$

$$\textbf{True Negative Rate} = \frac{TrueNegatives}{TrueNegative + FalsePositive} \tag{3}$$

In the previous set of equations, the variables *TruePositives*, *TrueNegatives*, *FalsePositives* and *FalseNegatives* are extracted from a *confusion matrix*. Figure 2 illustrates such concept for a binary classification problem – a $k$ class classification task is analogous.



| | | True condition | |
|---|---|---|---|
| | Total population | Condition positive | Condition negative |
| Predicted condition | Predicted condition positive | **True positive** | **False positive**, Type I error |
| | Predicted condition negative | **False negative**, Type II error | **True negative** |

**Fig. 2.** A confusion matrix for a binary classification problem

## 3 Genetic Algorithms

The genetic algorithm employed in our project was already included in our MATLAB distribution. Genetic algorithms are commonly used for optimization and search problems [5]. In a general way, in a genetic algorithm, a population of *individuals* (i.e. a *generation*) is repeatedly evolving towards a better solution. Each individual is represented by a vector of properties (e.g. 0's and 1's), which will be crossed with the properties of other individuals, in order for them to reproduce and create a new generation, with better individuals.

Firstly, the initial population is generated, often randomly. Then, the evolution process begins. A portion of the individuals is selected to reproduce, based on a *fitness function* which evaluates each individual. The individuals with a better fitness value will more likely be chosen. In our work, every individual is composed of six bits (i.e., a 1 on the $i$th position means that the $i$th feature is to be considered for model training and testing) plus a value for an hyperparameter of a model (if it's the case), and the fitness function trains the model with such features and hyperparameters, and then returns a measure on the testing set derived from the previously described electricity theft dataset. After selecting the fitter individuals, they will reproduce, thus creating a new and better generation. The children are created using *crossover* and *mutation* operators, which will attempt to provide the children a balance of the best characteristics of both parents. Mutation operators are specially useful to prevent the algorithm from getting stuck in local minima.

This process of selection and reproduction will continue until some stopping criteria is met, e.g. a fixed number of generations, when an individual is sufficiently fit so it is no longer necessary to continue, or computational limitations.

For our experiments, we considered 20 generations as stopping criteria, together with 20 individuals in each generation.

## 4 Neural Approach

In general, neural networks can be seen as computational artifacts that channel information through a series of mathematical operations, with the general purpose of accurately classifying inputs. Mathematically, neural networks can be seen as nested composite functions, whose parameters can be trained directly to minimize a given loss function computed over the outputs and the expected results. This is achieved through a training procedure known as back-propagation, in combination with gradient descent optimization of the parameters. A Multi-Layer Perceptron (MLP) consists of a set of nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the network layer-by-layer, until it reaches the output node(s). Figure 3 illustrates this concept, making use of three input signal, one hidden layer with three neurons, and two outputs. See [2] for a further description and analysis on Multi-Layer Perceptrons.

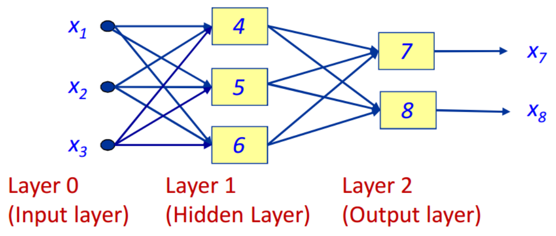
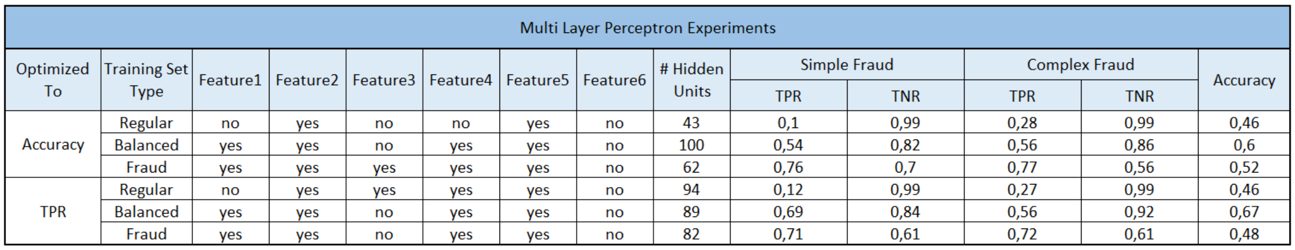


**Fig. 3.** A Multi-Layer Perceptron

### 4.1 Experimental Results

We ran the already explained genetic algorithm in order to determine what combination of features, together with the number of neurons in the hidden layer, was optimal. Figure 4 illustrates the experiments performed on MLP classifiers.



Multi Layer Perceptron Experiments

| Optimized To | Training Set Type | Feature1 | Feature2 | Feature3 | Feature4 | Feature5 | Feature6 | # Hidden Units | Simple Fraud | | Complex Fraud | | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | TPR | TNR | TPR | TNR | |
| Accuracy | Regular | no | yes | no | no | yes | no | 43 | 0,1 | 0,99 | 0,28 | 0,99 | 0,46 |
| | Balanced | yes | yes | no | yes | yes | no | 100 | 0,54 | 0,82 | 0,56 | 0,86 | 0,6 |
| | Fraud | yes | yes | yes | yes | yes | no | 62 | 0,76 | 0,7 | 0,77 | 0,56 | 0,52 |
| TPR | Regular | no | yes | yes | yes | yes | no | 94 | 0,12 | 0,99 | 0,27 | 0,99 | 0,46 |
| | Balanced | yes | yes | no | yes | yes | no | 89 | 0,69 | 0,84 | 0,56 | 0,92 | 0,67 |
| | Fraud | yes | yes | no | yes | yes | no | 82 | 0,71 | 0,61 | 0,72 | 0,61 | 0,48 |

**Fig. 4.** Experiments performed with a Multi-Layer Perceptron

In a more specific manner, the genetic algorithm worked with individuals represented as a vector of 7 dimensions – 6 bits corresponding to the use (or not) of the six features (i.e., the bit at position $i$ indicated whether we should use feature $i$), plus an integer corresponding to the number of neurons to consider in the hidden layer of the MLP. As fitness functions, we considered the accuracy and the sum of the True Positive Rates for each label, i.e., the fitness of an individual was measured as the accuracy or TPR achieved by that specific combination of features together with that specific number of hidden neurons, on the three datasets described in Section 2.

## 5 Fuzzy Clustering Approach

Besides neural approaches there are other ways to approximate nonlinear mapping between input features and outputs. A very popular mathematical grey-box model is a fuzzy inference system. In this report, more specifically, Takagi-Sugeno systems [6].

Such systems are based on rules with fuzzy antecedents and crisp consequents. To derive these rules from data, one can use a clustering algorithm like fuzzy c-means. The first step is to use such algorithm to cluster data points in input-output product space. After that, since each cluster corresponds to a rule, one can map the points in each cluster to input space and fit parametric membership functions to get the fuzzy sets for each rules antecedents. Finally, Sugenos consequent parameters can be estimated by minimizing mean squared error with least means squares.

After fitting a fuzzy inference system to the training data, one can use it for classification by: first computing the degree of fulfillment to the antecedents of the rules; second, for each rule that fired retrieve the consequent value; third, compute the output of the system by taking the mean value of all consequents. The described process can be seen on figure 5 that was taken from *MATLABs* documentation.

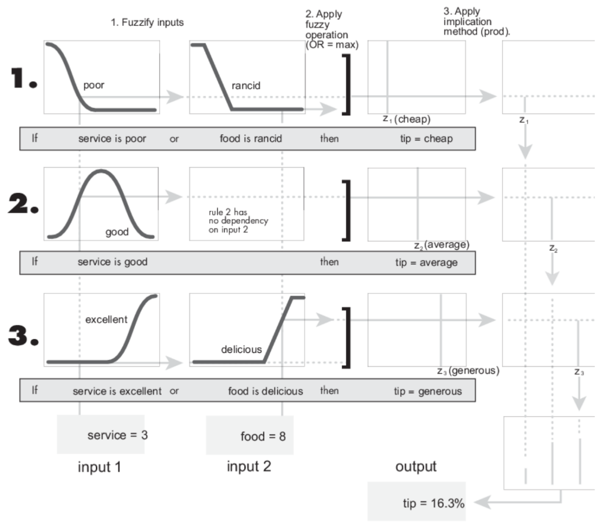

1. Fuzzify inputs
2. Apply fuzzy operation (OR = max)
3. Apply implication method (prod).
1.
poor
rancid
z₁ (cheap)
If service is poor or food is rancid then tip = cheap
2.
good
rule 2 has no dependency on input 2
z₂ (average)
If service is good then tip = average
3.
excellent
delicious
z₃ (generous)
If service is excellent or food is delicious then tip = generous
service = 3
food = 8
input 1
input 2
output
tip = 16.3%

**Fig. 5.** The inference process of a Takagi-Sugeno inference system.

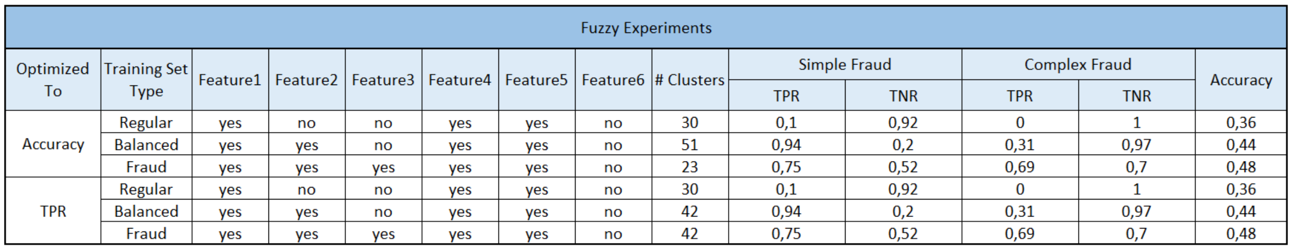
## 5.1 Experimental Results

The success of a system as the one described highly depends on both the quality of the clustering and on the input features that are chosen. Just like in the neural approach, in this report, we rely on derivative-free genetic optimization to choose the clustering parameterization and to perform feature selection.

So, we can formulate such an optimization as search for a parameterization vector that achieves the greater accuracy, or TPR, on a predefined test set. Such a vector stores knowledge about used features and, since we are using fuzzy c-means, about the number of clusters.

Since the provided dataset is composed by six different features, the parameterization vector has seven dimensions, one per feature, and another for the number of clusters. Furthermore, each of the feature assigned dimension $i$ can alternate between 0 which means that feature $i$ is not used and 1 which means that feature $i$ is used. Whereas, the last dimension belong to the positive natural integers and represents the number of clusters.

We ran the already explained genetic algorithm in order to determine the optimal parameterization vectors. Figure 6 illustrates the experiments. Once again, as fitness functions, we considered the accuracy and the True Positive Rate, i.e., the fitness of an individual was measured as the accuracy or the TPR



| Fuzzy Experiments | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Optimized To | Training Set Type | Feature1 | Feature2 | Feature3 | Feature4 | Feature5 | Feature6 | # Clusters | Simple Fraud | | Complex Fraud | | Accuracy |
| | | | | | | | | | TPR | TNR | TPR | TNR | |
| Accuracy | Regular | yes | no | no | yes | yes | no | 30 | 0,1 | 0,92 | 0 | 1 | 0,36 |
| | Balanced | yes | yes | no | yes | yes | no | 51 | 0,94 | 0,2 | 0,31 | 0,97 | 0,44 |
| | Fraud | yes | yes | yes | yes | yes | no | 23 | 0,75 | 0,52 | 0,69 | 0,7 | 0,48 |
| TPR | Regular | yes | no | no | yes | yes | no | 30 | 0,1 | 0,92 | 0 | 1 | 0,36 |
| | Balanced | yes | yes | no | yes | yes | no | 42 | 0,94 | 0,2 | 0,31 | 0,97 | 0,44 |
| | Fraud | yes | yes | yes | yes | yes | no | 42 | 0,75 | 0,52 | 0,69 | 0,7 | 0,48 |

**Fig. 6.** Experiments performed with a Takagi-Sugeno Fuzzy model

(i.e. the sum for the TPR of each label) achieved by that specific combination of features together with that specific number of hidden neurons, on the three datasets described in Section 2.

## 6 Neuro-Fuzzy Approach

As a way to combine neural approaches with fuzzy ones, neuro-fuzzy models appeared [4]. These models combine the self-organizing perks of neural networks with possible grey-box interpretability from fuzzy systems. In fact, such a model is mathematically equivalent to a Takagi-Sugeno system in certain conditions.

Although ultimately equivalent to fuzzy systems, neuro-fuzzy models have some architectural differences. First of all, such models can automatically compute the membership functions of the antecedents of the rules by using an input-output space grid partitioning which is somewhat related to clustering. Second of all, a neural network kind of learning is in place with error based adjustment of parameters. Figure 7 sums up the described architecture.

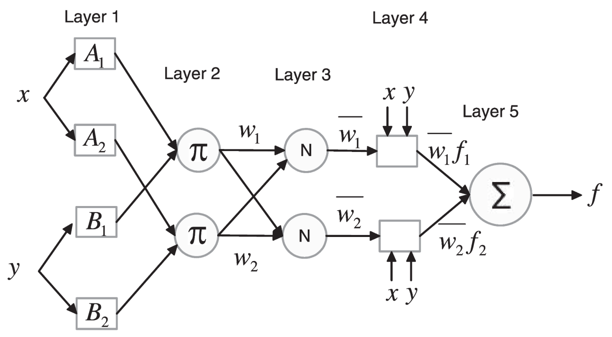


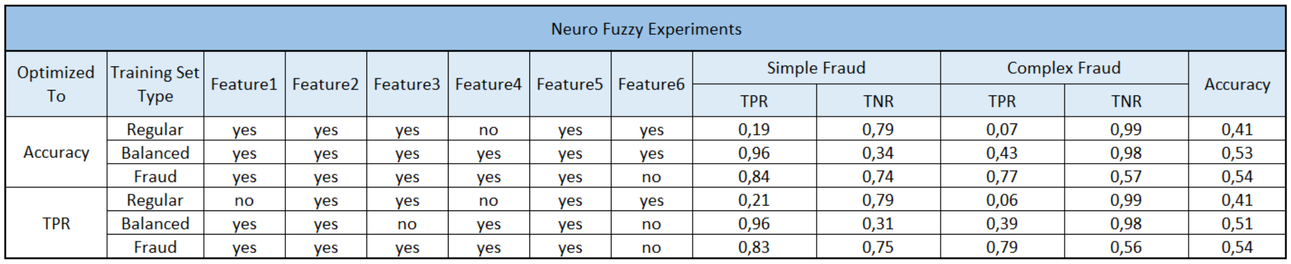
**Fig. 7.** The architecture of an artificial neuro-fuzzy inference system.

## 6.1 Experimental Results

The success of a system as the one described highly depends on the quality of the input features that are chosen. Just like in previous approaches, in this report, we rely on derivative-free genetic optimization to perform feature selection.

So, we can formulate such an optimization as search for a parameterization vector that achieves the greater accuracy on a predefined test set. Since the provided dataset is composed by six different features, such a vector has six dimensions, one per feature. Furthermore, each dimension $i$ can alternate between 0 which means that feature $i$ is not used and 1 which means that feature $i$ is used.

We once again ran the already explained genetic algorithm in order to determine the optimal parameterization vectors. Figure 8 illustrates the experiments. Again, as fitness functions, we considered the accuracy and the True Positive



Neuro Fuzzy Experiments

| Optimized To | Training Set Type | Feature1 | Feature2 | Feature3 | Feature4 | Feature5 | Feature6 | Simple Fraud | | Complex Fraud | | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | TPR | TNR | TPR | TNR | |
| Accuracy | Regular | yes | yes | yes | no | yes | yes | 0,19 | 0,79 | 0,07 | 0,99 | 0,41 |
| | Balanced | yes | yes | yes | yes | yes | yes | 0,96 | 0,34 | 0,43 | 0,98 | 0,53 |
| | Fraud | yes | yes | yes | yes | yes | no | 0,84 | 0,74 | 0,77 | 0,57 | 0,54 |
| TPR | Regular | no | yes | yes | no | yes | yes | 0,21 | 0,79 | 0,06 | 0,99 | 0,41 |
| | Balanced | yes | yes | no | yes | yes | no | 0,96 | 0,31 | 0,39 | 0,98 | 0,51 |
| | Fraud | yes | yes | yes | yes | yes | no | 0,83 | 0,75 | 0,79 | 0,56 | 0,54 |

**Fig. 8.** Experiments performed with an ANFIS approach

Rate, i.e., the fitness of an individual was measured as the accuracy or the TPR (i.e. the sum for the TPR of each label) achieved by that specific combination of features together with that specific number of hidden neurons, on the three datasets described in Section 2.

## 7 Ensemble Models

The results from the experiments from Sections 4,5, and 6 are 9 models, either optimized for accuracy or TPR. Specifically:

1. MLP trained on the Regular Dataset
2. MLP trained on the Balanced Dataset
3. MLP trained on the Fraud Dataset
4. Fuzzy Model trained on the Regular Dataset
5. Fuzzy Model trained on the Balanced Dataset
6. Fuzzy Model trained on the Fraud Dataset
7. Neuro-Fuzzy Model trained on the Regular Dataset
8. Neuro-Fuzzy Model trained on the Balanced Dataset

9. Neuro-Fuzzy Model trained on the Fraud Dataset

These different approaches, in theory, should be better at specific sub-tasks. For example, a model trained on the Regular Dataset should be better at detecting legitimate consumers, a model trained on the Balanced Dataset should be a better classifier in general, while models trained on the Fraud Dataset are specialized on distinguishing the fraud that's been committed. In order to leverage these characteristics, and given the improvement potential associated with ensemble methods [3], we created three ensembles of these 9 approaches, which will be discussed in the following sub-sections.

## 7.1 Regular Weight Voting

The first ensemble consisted of a simple weighted voting between the models to determine the class to which the input data belonged to – i.e., for a given data point, each of the 9 models outputted a prediction. Then, based on their voting weights, we calculated a score for each of the 3 labels, and the label with the highest score was outputted as the final prediction. See Figure 9 for a visual representation.

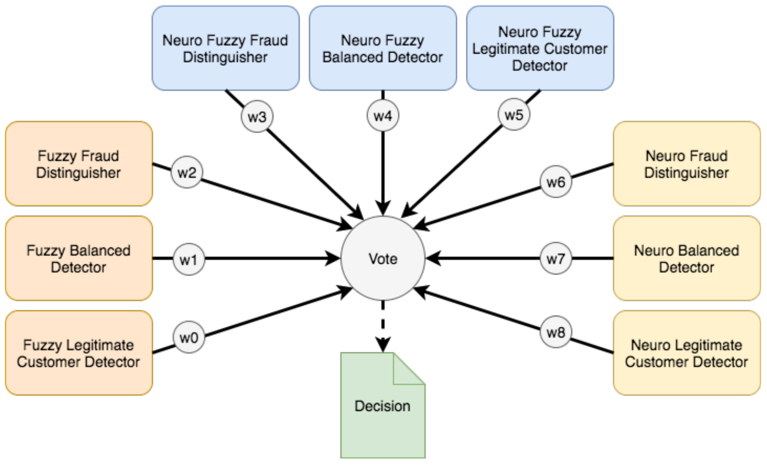
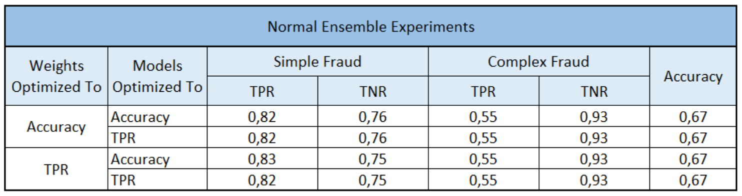

Neuro Fuzzy Fraud Distinguisher
Neuro Fuzzy Balanced Detector
Neuro Fuzzy Legitimate Customer Detector
w3
w4
w5
Fuzzy Fraud Distinguisher
w2
w6
Neuro Fraud Distinguisher
Fuzzy Balanced Detector
w1
Vote
w7
Neuro Balanced Detector
Fuzzy Legitimate Customer Detector
w0
w8
Neuro Legitimate Customer Detector
Decision

**Fig. 9.** The Regular Voting Ensemble Model

The weights were optimized with the genetic algorithm. Again, the ensemble was optimized either for accuracy or TPR. The models that compose the ensemble are the optimal models from the previous Sections, and were also optimized either for accuracy or TPR. Figure 10 presents the experiments on the ensemble.



| Normal Ensemble Experiments | | | | | | |
|---|---|---|---|---|---|---|
| Weights Optimized To | Models Optimized To | Simple Fraud | | Complex Fraud | | Accuracy |
| | | TPR | TNR | TPR | TNR | |
| Accuracy | Accuracy | 0,82 | 0,76 | 0,55 | 0,93 | 0,67 |
| | TPR | 0,82 | 0,76 | 0,55 | 0,93 | 0,67 |
| TPR | Accuracy | 0,83 | 0,75 | 0,55 | 0,93 | 0,67 |
| | TPR | 0,82 | 0,75 | 0,55 | 0,93 | 0,67 |

**Fig. 10.** The Regular Voting Ensemble Model Experiments

## 7.2 Hierarchic Voting of Two Committees with Shared Voting Weights

The next ensemble is composed of two committees – one for deciding whether the user is committing a fraud or not, and the other to decide, in case the user is indeed committing a fraud, the type of theft being dealt with. Concretely, 3 models trained on the Balanced Dataset will vote to determine whether the user is committing fraud or not. If the user is considered to be committing fraud, the same 3 Balanced Dataset models, together with other 3 models trained on the Fraud Dataset, will decide amongst them the kind of fraud being committed, outputting the result. If the user is considered to be legitimate, the 3 Balanced Dataset models will vote together with 3 Regular Dataset models (i.e., Legitimate Detectors), in order to really determine whether someone is actually legitimate, or is in fact a thief. If the consumer is a thief, the fraud type will be outputted. Figure 11 provides a graphical illustration.

The weights, denoted in the Figure as *wi*, were optimized via genetic algorithm. Once again, the models were either optimized for accuracy or for TPR. The ensemble was also optimized for one of those measures. Figure 12 presents the results obtained from the conducted experiments.

## 7.3 Hierarchic Voting of Two Committees without Shared Voting Weights

The last tested ensemble is identical to the previous one – with the exception that the voting weights are not shared between committees. All weights were optimized via genetic algorithm. Both the 9 models and the ensemble were optimized either for accuracy or for TPR. Figures 13 and 14 illustrate the ensemble scheme and the results from the experiments, respectively.

## 8 Results Discussion and State-of-the-Art Comparison

With all the experiments performed, pointing to a model and calling it "the best" is not straightforward. If we consider a quality measure as $TPR_1 + TNR_1 + TPR_2 + TNR_2$, where 1 and 2 are the simple frauds and complex frauds, respectively, then our top 3 is composed by:

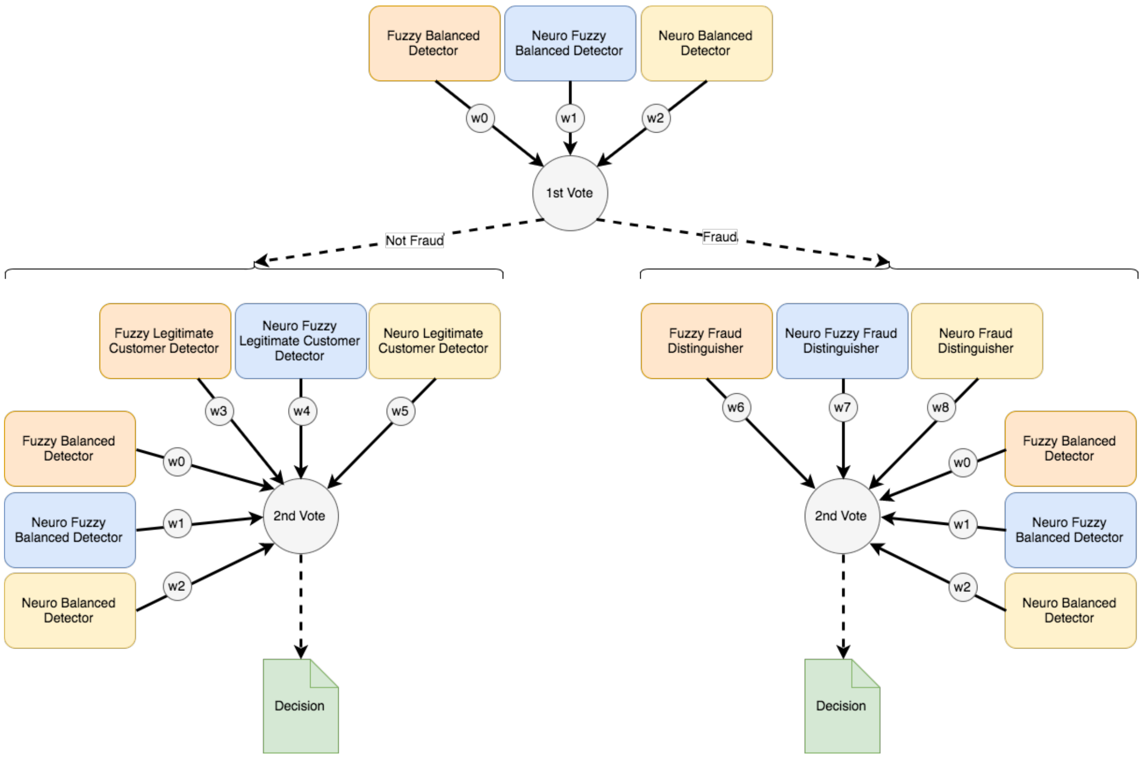
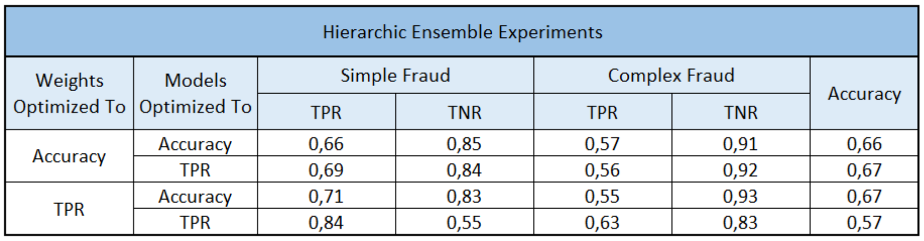

Fuzzy Balanced Detector
Neuro Fuzzy Balanced Detector
Neuro Balanced Detector
w0
w1
w2
1st Vote
Not Fraud
Fraud
Fuzzy Legitimate Customer Detector
Neuro Fuzzy Legitimate Customer Detector
Neuro Legitimate Customer Detector
w3
w4
w5
Fuzzy Balanced Detector
Neuro Fuzzy Balanced Detector
Neuro Balanced Detector
w0
w1
w2
2nd Vote
Decision
Fuzzy Fraud Distinguisher
Neuro Fuzzy Fraud Distinguisher
Neuro Fraud Distinguisher
w6
w7
w8
Fuzzy Balanced Detector
Neuro Fuzzy Balanced Detector
Neuro Balanced Detector
w0
w1
w2
2nd Vote
Decision

**Fig. 11.** The Shared Hierarchic Voting Ensemble Model



| Hierarchic Ensemble Experiments | | | | | | |
|---|---|---|---|---|---|---|
| Weights Optimized To | Models Optimized To | Simple Fraud | | Complex Fraud | | Accuracy |
| | | TPR | TNR | TPR | TNR | |
| Accuracy | Accuracy | 0,66 | 0,85 | 0,57 | 0,91 | 0,66 |
| | TPR | 0,69 | 0,84 | 0,56 | 0,92 | 0,67 |
| TPR | Accuracy | 0,71 | 0,83 | 0,55 | 0,93 | 0,67 |
| | TPR | 0,84 | 0,55 | 0,63 | 0,83 | 0,57 |

**Fig. 12.** The Shared Hierarchic Voting Ensemble Model Experiments

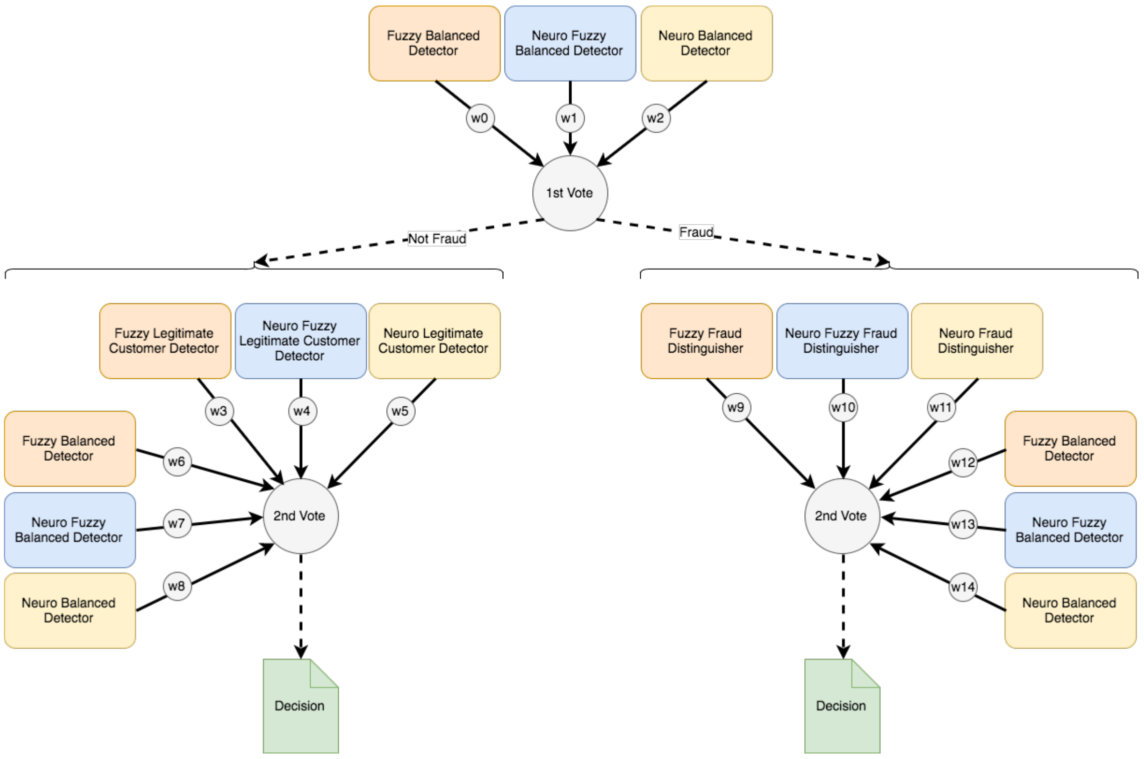
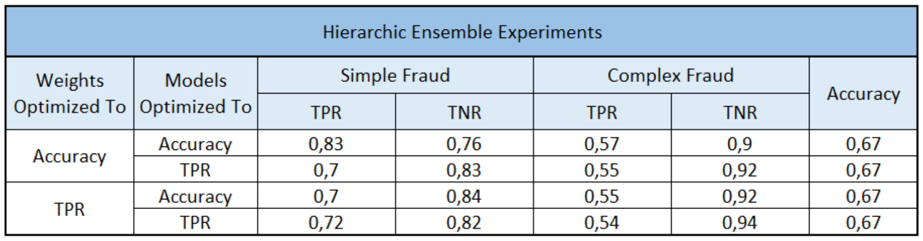

Fuzzy Balanced Detector
Neuro Fuzzy Balanced Detector
Neuro Balanced Detector
w0
w1
w2
1st Vote
Not Fraud
Fraud
Fuzzy Legitimate Customer Detector
Neuro Fuzzy Legitimate Customer Detector
Neuro Legitimate Customer Detector
w3
w4
w5
Fuzzy Balanced Detector
w6
Neuro Fuzzy Balanced Detector
w7
Neuro Balanced Detector
w8
2nd Vote
Decision
Fuzzy Fraud Distinguisher
Neuro Fuzzy Fraud Distinguisher
Neuro Fraud Distinguisher
w9
w10
w11
Fuzzy Balanced Detector
w12
Neuro Fuzzy Balanced Detector
w13
Neuro Balanced Detector
w14
2nd Vote
Decision

**Fig. 13.** The Hierarchic Voting Ensemble Model, without shared weights

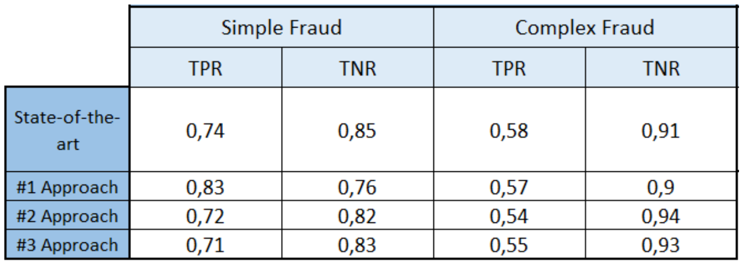


| Hierarchic Ensemble Experiments | | | | | | |
|---|---|---|---|---|---|---|
| Weights Optimized To | Models Optimized To | Simple Fraud | | Complex Fraud | | Accuracy |
| | | TPR | TNR | TPR | TNR | |
| Accuracy | Accuracy | 0,83 | 0,76 | 0,57 | 0,9 | 0,67 |
| | TPR | 0,7 | 0,83 | 0,55 | 0,92 | 0,67 |
| TPR | Accuracy | 0,7 | 0,84 | 0,55 | 0,92 | 0,67 |
| | TPR | 0,72 | 0,82 | 0,54 | 0,94 | 0,67 |

**Fig. 14.** The Shared Hierarchic Voting Ensemble Model Experiments, without shared voting weights

1. **The Hierarchic Voting of Two Committees without Shared Voting Weights Ensemble Model**, optimized for accuracy, composed of models also optimized for accuracy
2. **The Hierarchic Voting of Two Committees without Shared Voting Weights Ensemble Model**, optimized for TPR, composed of models also optimized for TPR
3. **The Hierarchic Voting of Two Committees with Shared Voting Weights Ensemble Model**, optimized for TPR, composed of models optimized for accuracy

As for previous work regarding this dataset, Professor Joaquim Viegas provided us with previous work, to which we compared our results. Figure 15 reports the results the Professor gave us.



| | Simple Fraud | | Complex Fraud | |
|---|---|---|---|---|
| | TPR | TNR | TPR | TNR |
| State-of-the-art | 0,74 | 0,85 | 0,58 | 0,91 |
| #1 Approach | 0,83 | 0,76 | 0,57 | 0,9 |
| #2 Approach | 0,72 | 0,82 | 0,54 | 0,94 |
| #3 Approach | 0,71 | 0,83 | 0,55 | 0,93 |

**Fig. 15.** The State-of-the-Art results

We can therefore conclude that we achieved results pretty close to the state-of-the-art, and in some models, we actually surpassed some metrics. In others, we achieved a bit lower results, but in general, we achieved competitive results.

## 9 Conclusions

Our work was based on the Intelligent Decision and Control course at IST, and we conducted a study on a Electricity Fraud Dataset, in order to detect fraudulent users. The approaches we considered – Neuro approaches, Fuzzy approaches and Neuro-Fuzzy approaches – were all taught in the aforementioned course, and when combining these models to create a voting ensemble, we were able to achieve state-of-the-art results.

For future work, we suggest two research directions: the first one, is to consider *data augmentation*. Since the dataset we were given is so highly imbalanced, augmenting the data in order to even the class distribution (e.g. using data augmentation algorithms such as SMOTE [1]) may be a good idea for a better general classifier. Our second suggestion is an adaptation of the learning algorithm, where the error function would penalize a lot more a fraud/no fraud mistake than a specific fraud classification, hence improving fraud detection.

## References

1. Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 2002.
2. Walter H Delashmit and Michael T Manry. Recent developments in multilayer perceptron neural networks. In *Proceedings of the seventh Annual Memphis Area Engineering and Science Conference, MAESC*, 2005.
3. Thomas G Dietterich et al. Ensemble methods in machine learning. *Multiple classifier systems*, 1857, 2000.
4. J-SR Jang. ANFIS: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3), 1993.
5. Kim-Fung Man, Kit-Sang Tang, and Sam Kwong. Genetic algorithms: concepts and applications [in engineering design]. *IEEE transactions on Industrial Electronics*, 43(5), 1996.
6. Michio Sugeno. *Industrial applications of fuzzy control.* Elsevier Science Inc., 1985.