Energy Fraud Detection System based on Deep Learning

Davide Ermellino @ Università degli Studi di Cagliari

August 9, 2025

Abstract

This document describes a comprehensive system for energy fraud detection using deep learning techniques. The system combines recurrent autoencoders for feature extraction with advanced classifiers (DNN, XGBoost, and Random Tree) to identify anomalies in energy consumption patterns. The proposed approach utilizes multiple time series for each meter (Supply ID) and implements both threshold-based binary classification and supervised multi-class classification strategies.

1 Introduction

Energy fraud detection represents a critical challenge for distribution companies. This system develops a deep learning-based solution that analyzes multiple time series to identify anomalous patterns in energy consumption.

The dataset consists of five types of time series for each Supply ID:

- **ANAGRAFICA**: Customer details
- **CONSUMI**: Quarter-hourly consumption (in kWh)
- **INTERRUZIONI**: Interruption that occurred on a Supply ID
- **LAVORI**: Works carried out on the supply points
- **PAROLE_DI_STATO**: Alarms that are triggered on the meter

2 System Architecture

2.1 Data Preprocessing Pipeline

Before applying machine learning techniques, extensive preprocessing is performed on the raw datasets to ensure data quality and consistency. The preprocessing pipeline handles each dataset type differently:

2.1.1 CONSUMI Dataset

The consumption dataset undergoes the following transformations:

- **Supply ID normalization**: Extract numeric part from Supply_ID strings
- **Missing value imputation**: Replace empty strings with mean values on "val" attribute
- **Comma to dot**: the decimal values are written with the comma (e.g 6,5), changed to dot (6.5)

- **Temporal feature engineering**: Parse measurement timestamps into year, month, day, and time components
- **Cyclic encoding**: Apply sinusoidal transformations to temporal features:

$$f_{sin} = \sin\left(\frac{2\pi \cdot t}{T}\right), \quad f_{cos} = \cos\left(\frac{2\pi \cdot t}{T}\right) \tag{1}$$

  where t is the time component and T is the period (12 for months, 31 for days, 96 for daily quarters)
- **Categorical encoding**: Convert "magnitude" attribute to integer codes
- **Standardization**: Apply StandardScaler to continuous variables ("consumi" values and "years")

2.1.2 ANAGRAFICA Dataset

Customer details data preprocessing includes:

- **Date parsing**: Extract begin and end reference dates with cyclic encoding
- **Missing data handling**: Forward and backward fill for "available power" attribute
- **Categorical encoding**: Convert "supply status" to numeric codes
- **Standardization**: StandardScaler the continous variable "available power" and temporal features ("begin_date_year", "end_date_year")

2.1.3 LAVORI Dataset

Work events are processed with:

- **Date normalization**: Convert dates to cyclic features
- **Categorical handling**: Encode "woa_activity_type" and "woa_activity_subtype" to natural numbers
- **Missing value treatment**: Remove records with Nan on "woa_activity_type" attribute. Replace Nan values of "woa_activity_subtype" with the mode (most frequent)
- **Duration scaling**: Apply StandardScaler on "execution year" attribute

2.1.4 INTERRUZIONI Dataset

Interruption events are processed with:

- **Date normalization**: Convert dates to cyclic features
- **Categorical handling**: Encode "tipologia_interruzione" attribute to natural numbers
- **Duration scaling**: Apply StandardScaler on "start_date_year", "end_date_year", "durata_netta" attributes

2.1.5 PAROLE_DI_STATO Dataset

Triggered Alarms preprocessing involves:

- **Temporal encoding**: Apply cyclic transformations to measurement timestamps
- **Multi-label binarization**: Convert comma-separated "ListaParole" attribute to binary vectors using MultiLabelBinarizer (one hot encoding)
- **Feature scaling**: Apply StandardScaler on "meas_ts_year" attribute

2.1.6 LABELS Dataset

Triggered Alarms preprocessing involves:

- **Categorical handling**: Encode "CLUSTERS" to natural numbers
  - "Regolare" : 2
  - "Anomalia" : 0
  - "Frode" : 1

2.2 Main Pipeline

The system implements a modular pipeline composed of five main phases:

1. **Data Preprocessing**: Raw data cleaning and feature engineering
2. **Data Loading and Preprocessing(data preparing)**: Handle time series with variable lengths
3. **Feature Extraction**: Use recurrent autoencoders to generate embeddings
4. **Feature Combination**: Concatenate multi-modal embeddings
5. **Classification**: Apply classifiers for anomaly detection

2.3 Recurrent Autoencoders

Sequence Lenght Managed the different sequence length in the dataset by computing the mean length. Truncated/added padding to each series to the mean sequence length. Set max sequence length to 1000 (for my computer performance). In the future, we can remove the upper bound limit.

For each time series type, a recurrent autoencoder is trained:

$$z = \text{Encoder}(X) \in \mathbb{R}^d \tag{2}$$

$$\hat{X} = \text{Decoder}(z) \tag{3}$$

where $X \in \mathbb{R}^{T \times F}$ is the input time series with T timesteps and F features, z is the embedding of dimension $d = 16$, and $\hat{X}$ is the reconstruction.

The reconstruction error (MAE) is computed as:

$$\mathcal{L}(\hat{\text{X}}) = \frac{1}{T \cdot F} \sum_{t=1}^{T} \sum_{f=1}^{F} |x_{t,f} - \hat{x}_{t,f}| \tag{4}$$

2.4 Embedding Combination

To handle all sources, we apply an intermediate-level fusion technique [1]: We first create embeddings from preprocessed data (input) and then fuse them through concatenation. note: each embedding has the same length

$$E_{combined} = [e_{anagrafica}, e_{consumi}, e_{interruzioni}, e_{lavori}, e_{parole}, r_{errors}] \tag{5}$$

where r_{errors} represents normalized reconstruction errors (mean).

3 Classification Methodologies

3.1 Threshold-based Binary Classification

For anomaly detection, an adaptive threshold is used:

$$\tau = \mu(R) + \alpha \cdot \sigma(R) \tag{6}$$

where R are the reconstruction errors, μ and σ are the mean and standard deviation, and $\alpha = 0.5$ is a tuning parameter.

Classification occurs according to:

$$y_{pred} = \begin{cases} 0 & \text{if } r < \tau \text{ (normal)} \\ 1 & \text{if } r \geq \tau \text{ (anomalous)} \end{cases} \tag{7}$$

the division of the classes is:

- Class 0: Regolare
- Class 1: Anomalia, Frode

3.2 Supervised Classification

3.2.1 Deep Neural Network (DNN)

The neural network implements a feed-forward architecture with:

- Input layer: combined embeddings dimension
- Hidden layers: [128, 64, 32] neurons with ReLU activation
- Output layer: 3 value indicating the 3 classes probabilities (the softmax is applied in the loss function in Pytorch)
- Dropout: [0.3, 0.3, 0.2] for regularization

The loss function uses weighted cross-entropy to handle class imbalance:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{N} w_{y_i} \log(p_{y_i}) \tag{8}$$

hyperparameter tuning For this model, hyperparameter tuning was performed on learning rate, batch size, and weight decay (for Adam optimizer). A possible extension is to tune more hyperparameters (num layer, layer dim, dropout)

3.2.2 XGBoost

The XGBoost classifier is configured with (after hyperparameter tuning):

- Max depth: 4
- Learning rate: 0.0005
- Regularization: L1=0.1, L2=0.1
- Early stopping: 20 rounds
- Sample weights to handle imbalance

hyperparameter tuning For this model, hyperparameter tuning was performed on learning rate and max depth.

4 Training Strategy

4.1 Autoencoders

For the autoencoder training i used 2 approaches:

4.1.1 "All Clusters" Mode

Autoencoders are trained every kind of data (CLUSTER=2): the bias of the anomalous data reduce generalization capabilities of the models

4.1.2 "Only Regular" Mode

To improve detection capability, autoencoders are trained exclusively on regular samples (CLUSTER=2):

- Better learning of normal patterns
- Higher sensitivity to anomalies during inference
- Reduction of bias due to anomalous samples

4.2 Classification

Because of the unbalanced and limited amount of data, I went for cross-validation + ensemble.

4.2.1 Cross-Validation

- 4-fold Stratified Cross-Validation
- returns 4 different models trained on different split of training data

4.2.2 Ensamble

Given the 4 models I tested them and ensembled the result with hard and soft voting.

5 Implementation

The system is implemented in Python using:

- **PyTorch**: For autoencoders and DNN
- **XGBoost**: For gradient boosting
- **Scikit-learn**: For preprocessing and metrics
- **Pandas/NumPy**: For data manipulation and initial preprocessing
- **Polars**: For efficient large dataset processing
- **shap**: For XAI

5.1 Modular Structure

The code is organized in specialized modules:

- `build_features.py`: Raw data preprocessing and feature engineering
- `data_processing.py`: Loading and sequence preprocessing
- `training_orchestrator.py`: Training coordination
- `embedding_generator.py`: Embedding generation
- `binary_classifiers.py`: Threshold-based classification
- `evaluation.py`: Evaluation and supervised training
- `visualization/`: Result visualization modules

6 Evaluation

6.1 Metrics

The system is evaluated using:

- **Weighted Recall**: Prioritizes minority classes
- **Accuracy**: Overall precision
- **Precision/Recall**: Per class
- **Confusion Matrix**: Detailed error analysis

6.2 Weighted Recall for Minority Classes

Defined as:

$$R_{weighted} = \frac{\sum_{c=1}^{C} w_c \cdot R_c}{\sum_{c=1}^{C} w_c} \tag{9}$$

where $w_c = \frac{N}{C \cdot N_c}$ with N total samples, C classes, N_c samples per class c.

7 Results and Conclusions

7.1 Experimental Results

The system was evaluated on a dataset containing 100 supply IDs with their associated time series data.

7.1.1 Autoencoder Training Results

The recurrent autoencoders achieved convergence across all dataset types being trained with and without irregular samples:

- **CONSUMI**: Final reconstruction MAE 0.35 and 0.13
- **ANAGRAFICA**: Final reconstruction MAE of 0.38 and 0.37
- **INTERRUZIONI**: Final reconstruction MAE of 0.50 and 0.36
- **LAVORI**: Final reconstruction MAE of 0.9 and 0.8
- **PAROLE_DI_STATO**: Final reconstruction MAE of 0.16 and 0.14

Figure 2 shows the training curves for all autoencoders, trained on REGULAR data only. While the 1 shows the training curves for all autoencoders but trained on all the clusters

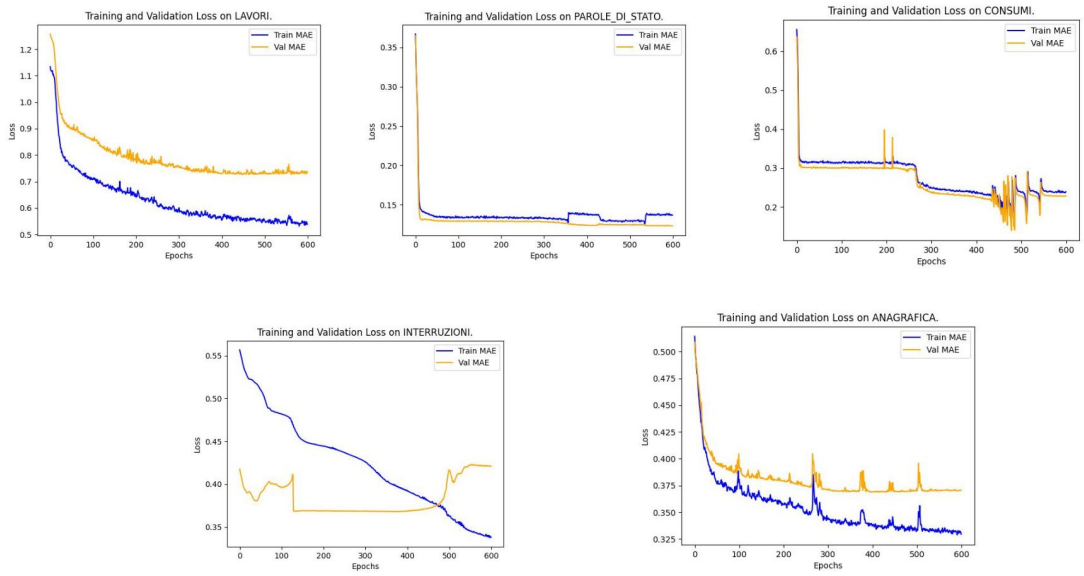

Figure 1: Training and validation loss curves for all autoencoder models

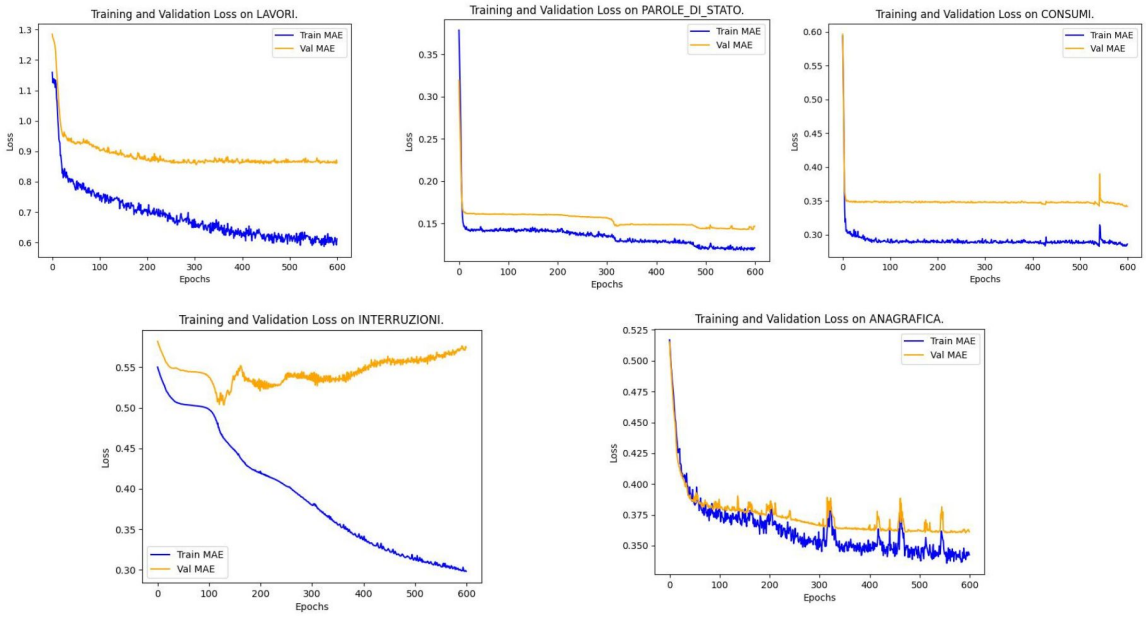

Figure 2: Training and validation loss curves for all autoencoder models.

When comparing the two sets of training plots, we notice a consistent improvement in validation loss when the autoencoders are trained exclusively on regular samples. This highlights the importance of excluding anomalies during training to avoid biasing the reconstruction model.

Specifically, for the CONSUMI feature, the validation MAE drops from approximately 0.36 (trained on all clusters) to 0.12 (trained only on regular data). Similarly, for INTERRUZIONI, the validation loss decreases from around 0.50 to 0.35. These substantial reductions indicate that anomalies in these datasets are structurally different from normal samples and negatively impact training if included.

Overall, this shows that training autoencoders on clean, regular data allows them to better learn the true underlying structure, leading to improved anomaly detection — particularly in features like INTERRUZIONI and CONSUMI, which seem to contain highly distinguishable irregular patterns.

7.1.2 Classification Performance

Binary Classifier on single autoencoder For this classifier i used all the data. For this reason the classification of the regular data is not reliable. We focus on the classification of the positive class.

Because of the difference on the training with the CONSUMI dataset i tried to use the reconstruction error of that dataset only. But looking at the AUC val the model is not able to make a better decision than a random one

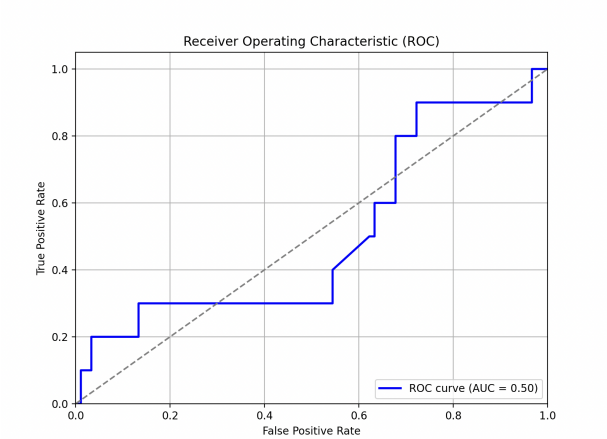

Figure 3: ROC curve and AUC val of CONSUMI

Binary Classifier on combined embeddings After concatenating the embedding the new classifier has better AUC results (0.69)

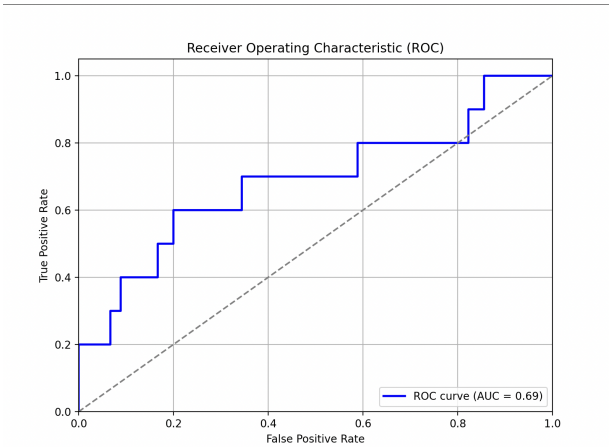

Figure 4: ROC curve and AUC val of combined embedding

So i implemented a binary classifier with a treshold (mean reconstruction error above all embeddings). It was able to detect 4/10 anomaly-fraud data

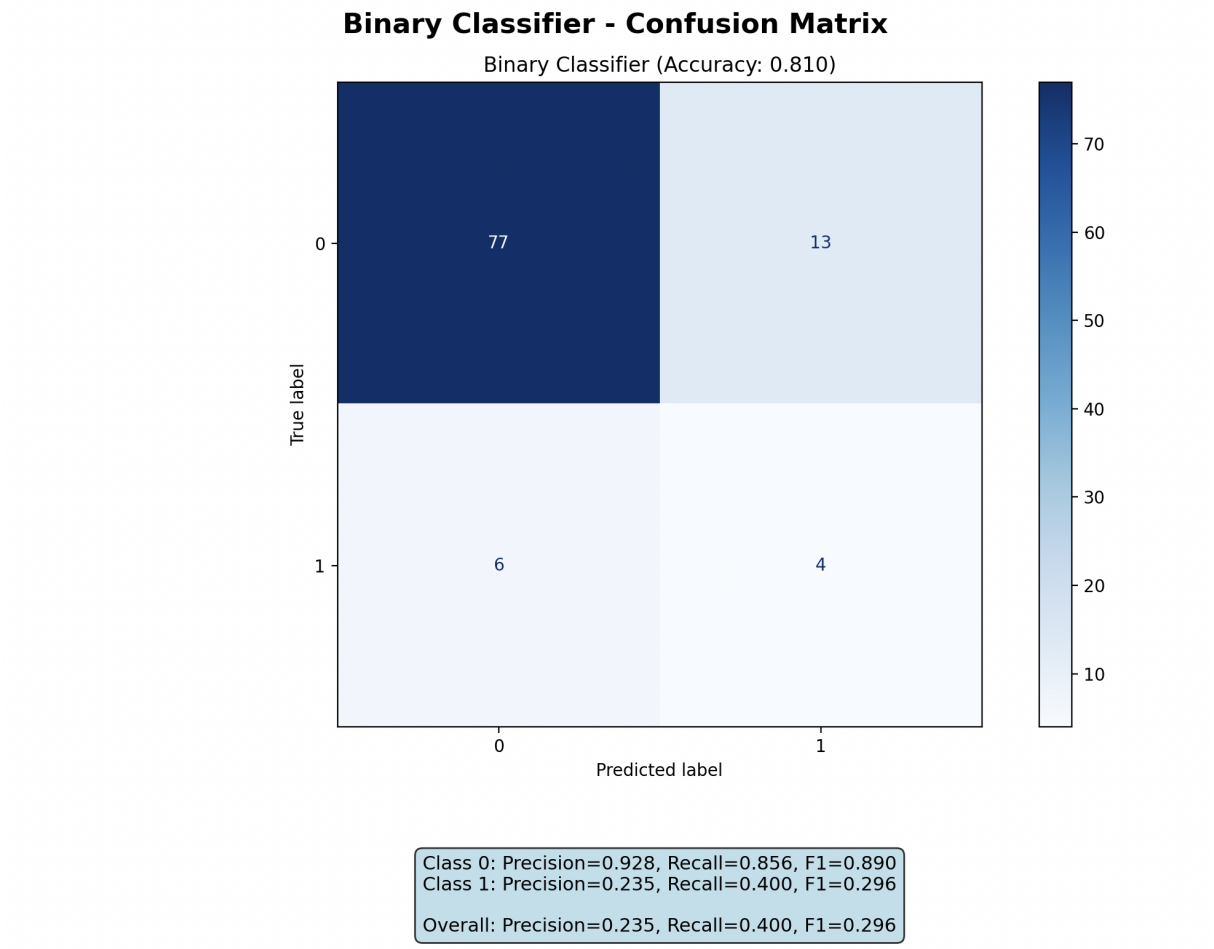

Figure 5: Confusion matrix of binary classifier

8 Multiclass Classifiers

8.1 ONLY REGULAR

8.1.1 Data Split

Split data in Train, Test and Val, with stratified technique, I train the folds on the Train data only, the Test is used only on the ensamble test.

- Test size: 40% of dataset
- Validation size: 14% of training set

The supervised classifiers were evaluated using 4-fold cross-validation and then the ensamble between the 4 models:

Ensemble Methods Both DNN, XGBoost and RandomForest classifiers utilize ensemble approaches to improve robustness and performance: The 4-fold cross-validation creates 4 independent models, each trained on different data splits (train). During inference, predictions are combined using hard and soft voting.

Table 5 provides a comprehensive comparison of all approaches across different metrics (f1-score is for the class="Frode").

Table 1: Performance comparison of different classification approaches

Method	Weighted Recall	Accuracy	F1-Score
DNN Hard Voting	0.560	0.70	0.27
DNN Soft Voting	0.568	0.73	0.29
XGBoost Hard Voting	0.0255	0.85	0
XGBoost Soft Voting	0.0255	0.85	0
RF Soft Voting	0.027	0.9	0
RF Soft Voting	0.027	0.9	0
Threshold Binary	0.4456	0.81	0.2963

8.1.2 Confusion Matrix Analysis

The Figures below shows the confusion matrices for the ensambles models of DNN, XGBOOST and RF

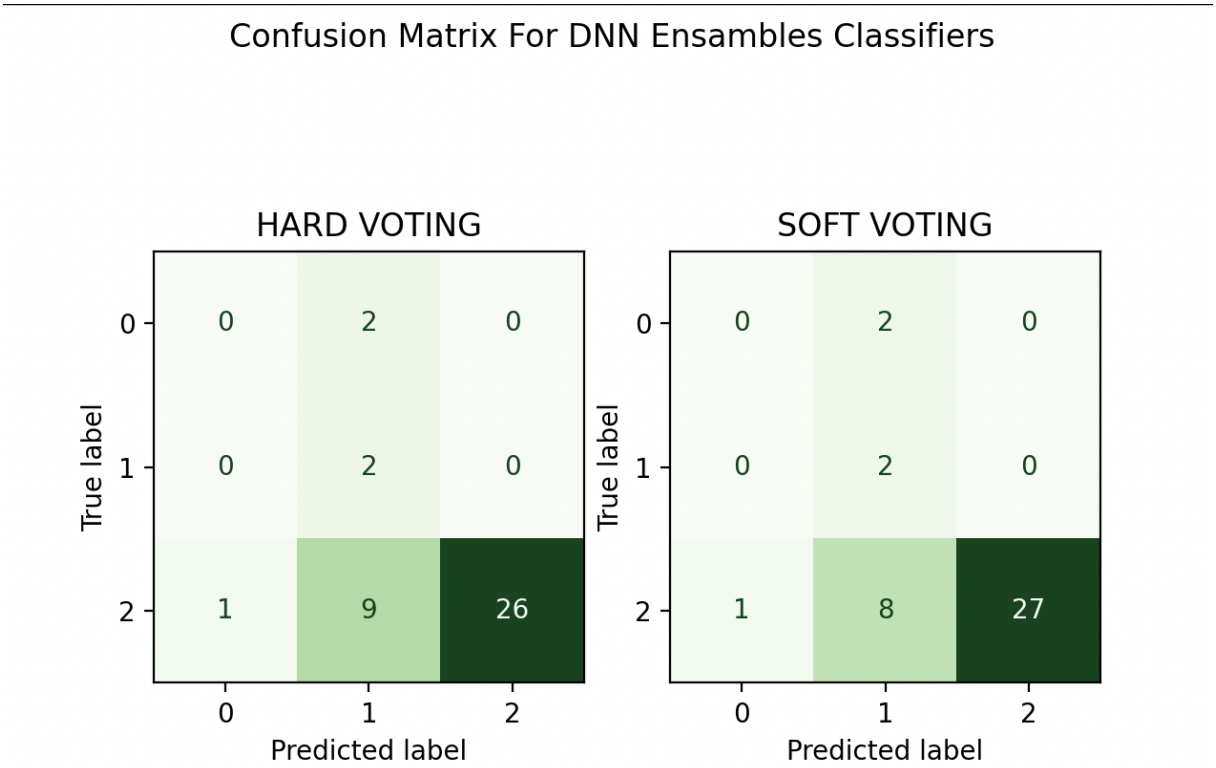

Figure 6: DNN CONF MATRIX

The DNN model was able to detect all the fraud (2) but 0/2 Anomalies detected

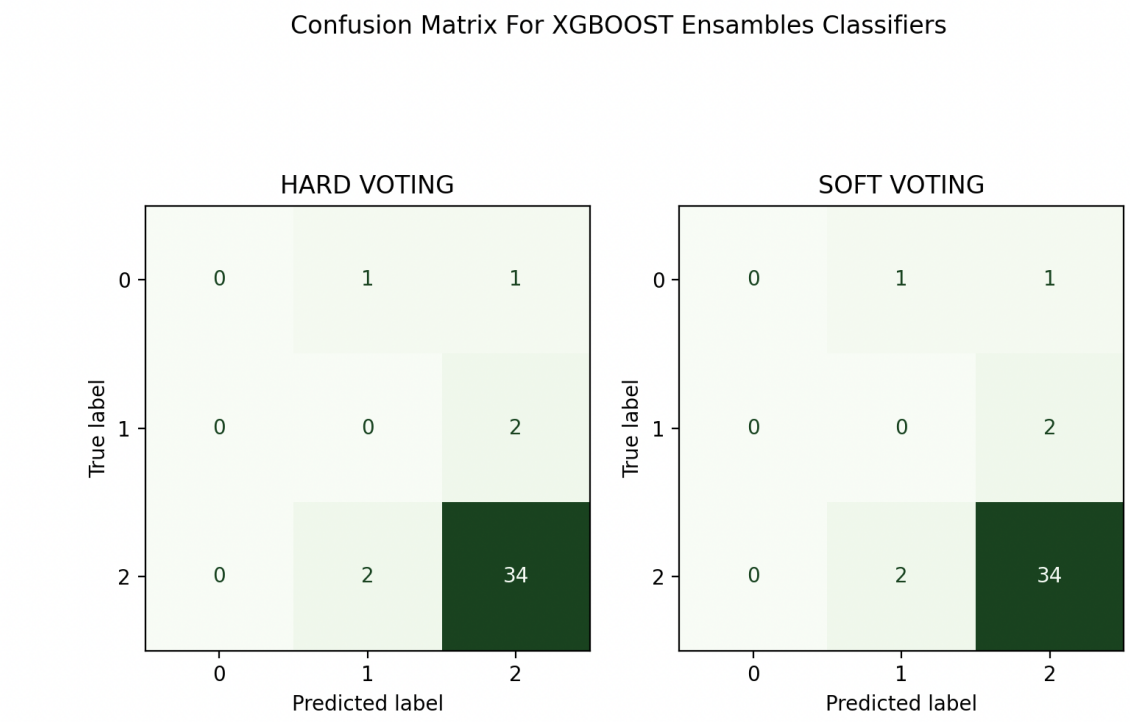

Figure 7: XGBOOST CONF MATRIX

XGBOOST failed to detect both Fraud and anomalies

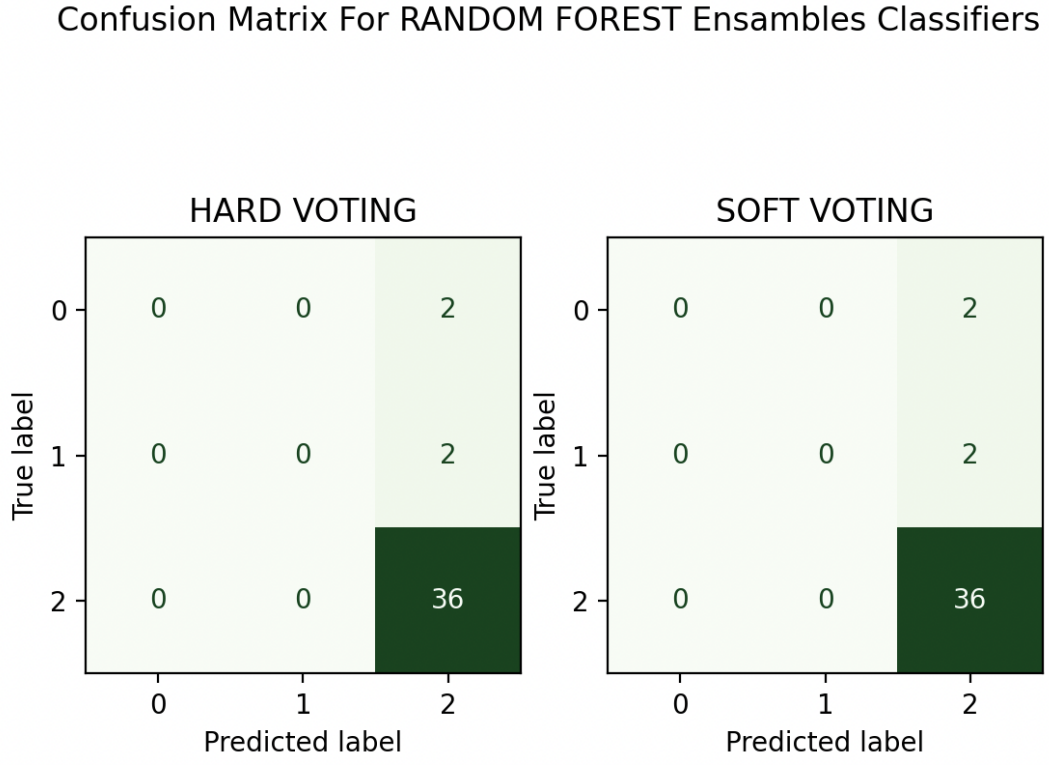

Figure 8: RF CONF MATRIX

RF predicted only the class 2

8.1.3 Feature Importance Analysis

The analysis of embedding contributions using SHAP (SHapley Additive exPlanations) values revealed different patterns between the two classifiers:

Table 2: Dataset contributions analysis using SHAP for XGBoost classifier

Dataset	**Contribution (%)**
ANAGRAFICA embeddings	9.6
INTERRUZIONI embeddings	21.1
CONSUMI embeddings	59.3
LAVORI embeddings	2.1
PAROLE_DI_STATO embeddings	7.9

Table 3: Dataset contributions analysis using SHAP for DNN classifier

Dataset	**Contribution (%)**
ANAGRAFICA embeddings	25.2
INTERRUZIONI embeddings	29.9
CONSUMI embeddings	20.7
PAROLE_DI_STATO embeddings	10.4
LAVORI embeddings	13.8

Table 4: Dataset contributions analysis using SHAP for RF classifier

Dataset	**Contribution (%)**
ANAGRAFICA embeddings	45.2
INTERRUZIONI embeddings	17.4
CONSUMI embeddings	18.3
PAROLE_DI_STATO embeddings	2.7
LAVORI embeddings	16.4

Figure 9: Feature importance multi bar chart

8.2 ALL CLUSTERS

8.2.1 Data split

the split technique is the same as the ONLY REGULAR method

Table 5 provides a comprehensive comparison of all approaches across different metrics (f1-score is for the class="Frode").

Table 5: Performance comparison of different classification approaches

Method	Weighted Recall	Accuracy	F1-Score
DNN Hard Voting	0.2598	0.57	0.12
DNN Soft Voting	0.2628	0.68	0.15
XGBoost Hard Voting	0.0248	0.82	0
XGBoost Soft Voting	0.0255	0.85	0
RF Soft Voting	0.027	0.9	0
RF Soft Voting	0.027	0.9	0
Threshold Binary	0.4456	0.81	0.2963

8.2.2 Confusion Matrix Analysis

The Figures below shows the confusion matrices for the ensambles models of DNN, XGBOOST and RF with the embeddings trained with all clusters type.

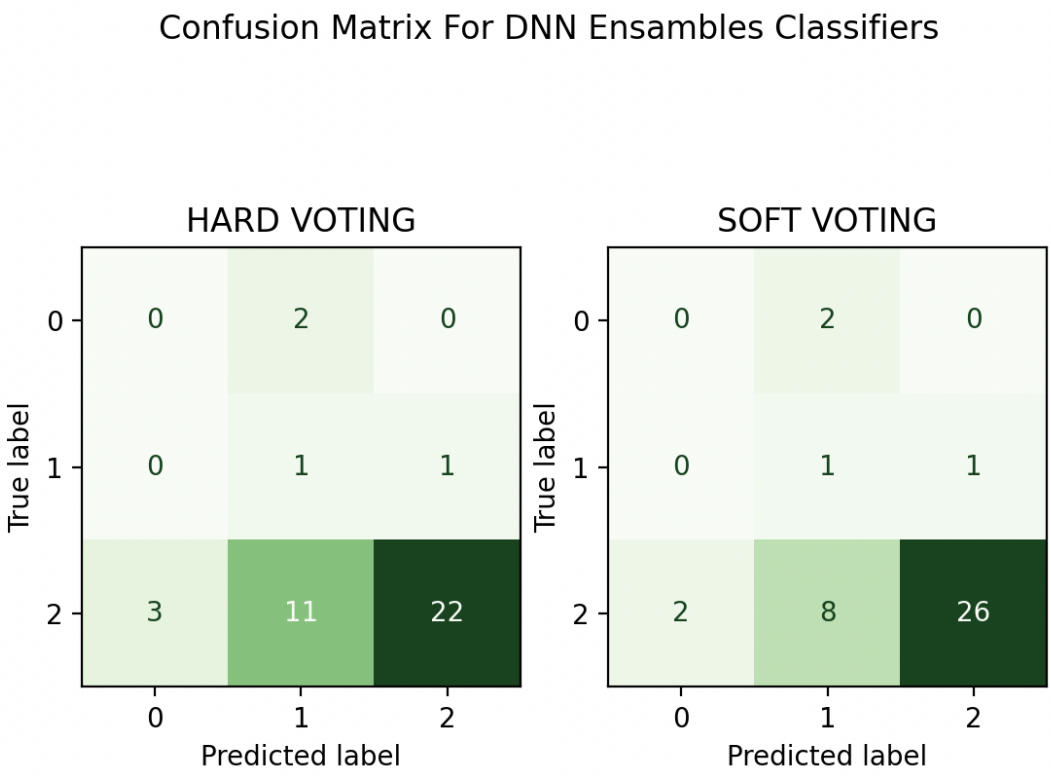

Figure 10: DNN CONF MATRIX

The performance of DNN are slightly worse, only 1/2 right prediction on fraud and still 0 right prediction on anomalyes

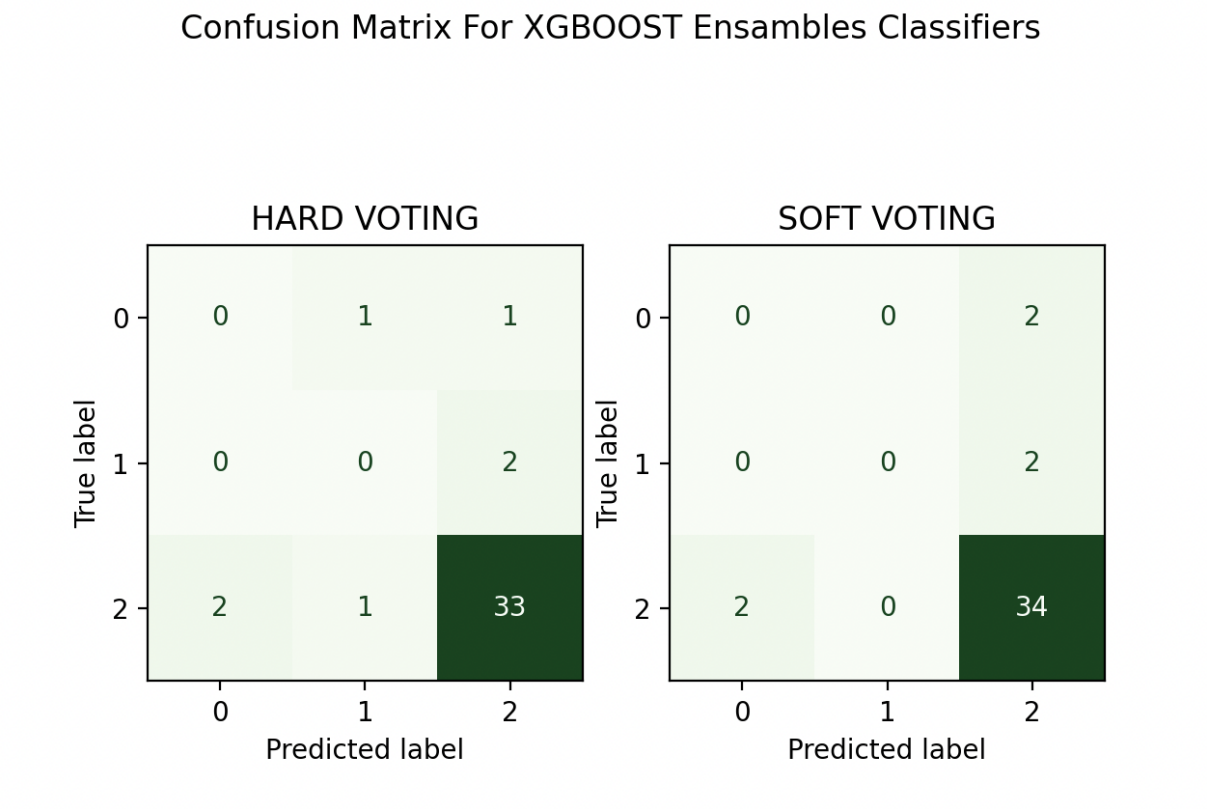

Figure 11: XGBOOST CONF MATRIX

Basically the same as before

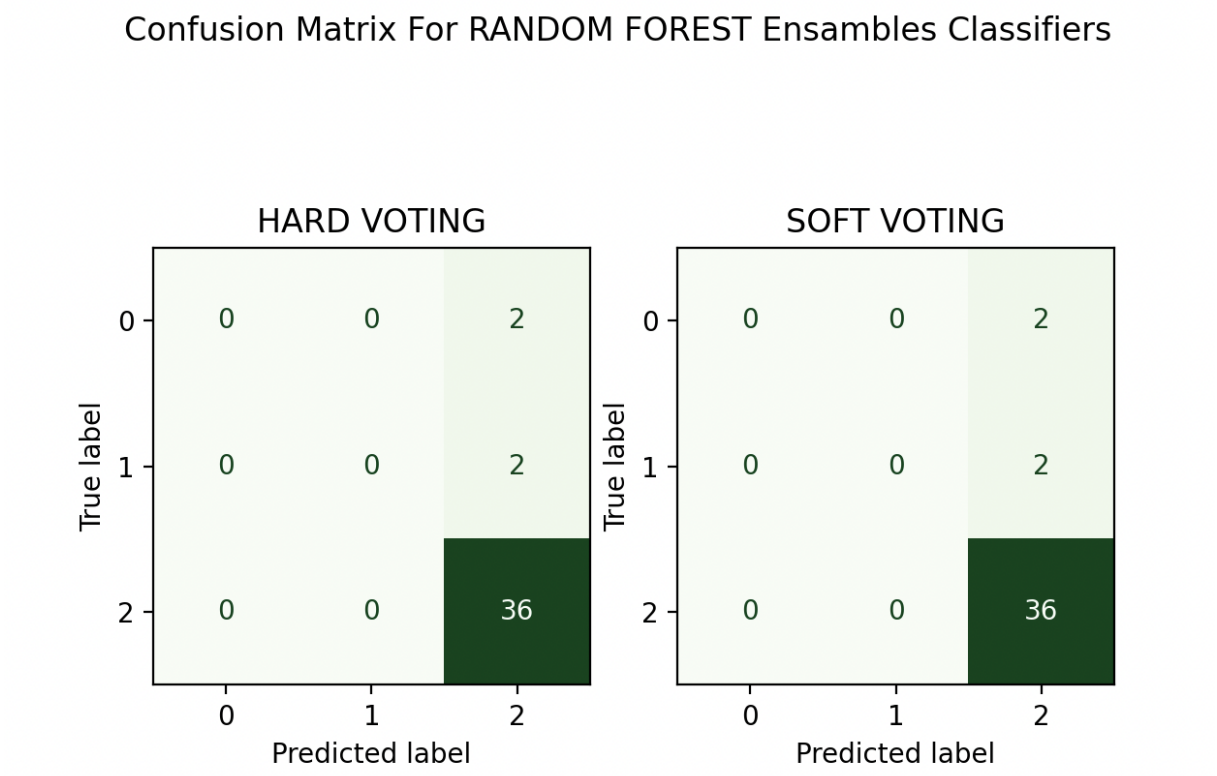

Figure 12: RF CONF MATRIX

Same stats for Random Forest

8.2.3 Feature Importance Analysis

The analysis of embedding contributions using SHAP (SHapley Additive exPlanations) values revealed different patterns between the two classifiers:

Table 6: Dataset contributions analysis using SHAP for XGBoost classifier

Dataset	**Contribution (%)**
ANAGRAFICA embeddings	55.7
INTERRUZIONI embeddings	12.9
CONSUMI embeddings	27.1
LAVORI embeddings	4.2
PAROLE_DI_STATO embeddings	0.0

Table 7: Dataset contributions analysis using SHAP for DNN classifier

Dataset	**Contribution (%)**
ANAGRAFICA embeddings	44.7
INTERRUZIONI embeddings	23.9
CONSUMI embeddings	3
PAROLE_DI_STATO embeddings	18.6
LAVORI embeddings	9.8

Table 8: Dataset contributions analysis using SHAP for RF classifier

Dataset	Contribution (%)
ANAGRAFICA embeddings	34.1
INTERRUZIONI embeddings	26.8
CONSUMI embeddings	17.4
PAROLE_DI_STATO embeddings	14.8
LAVORI embeddings	6.9

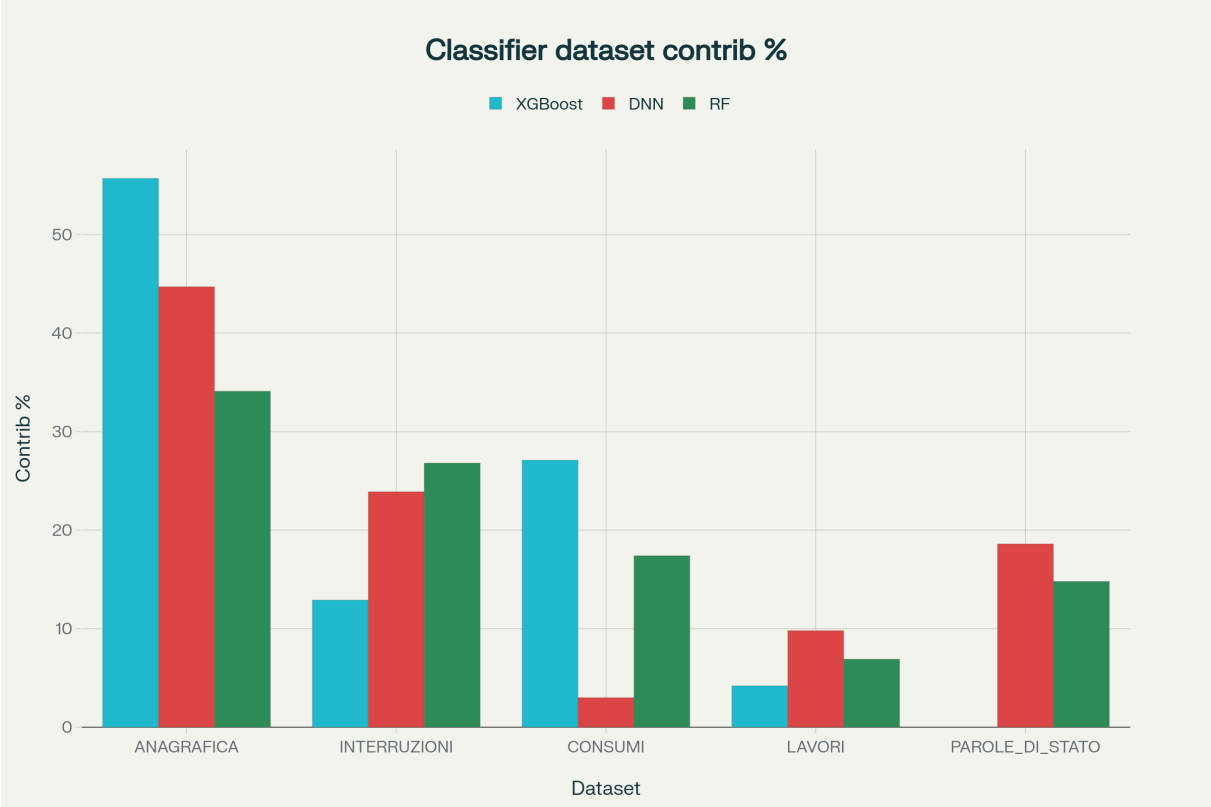

Figure 13: Feature Importance Analysis

9 Discussion

The experimental results demonstrate several key findings across both evaluation modes (**Only Regular** vs. **All Clusters**):

1. **Impact of Training Data (Only Regular vs. All Clusters)**: Training autoencoders exclusively on regular samples (*Only Regular*) consistently improves reconstruction error performance, particularly for datasets such as CONSUMI and INTERRUZIONI, where anomalies differ significantly from normal patterns. This improvement carries over to classification: the DNN in *Only Regular* mode achieves a weighted recall of 0.568 compared to 0.2628 in *All Clusters*.

2. **Classification Performance**: In both modes, the DNN classifier outperforms XGBoost and Random Forest in terms of weighted recall, which is crucial for fraud detection scenarios where identifying minority classes is paramount. However, the performance drop in *All Clusters* mode suggests that including anomalous samples during autoencoder training reduces the model's ability to learn discriminative features for rare events.

3. **Accuracy vs. Recall Trade-off**: Random Forest achieves the highest overall accuracy in both modes (0.9), but completely fails to detect minority classes (predict only class 2). The DNN, while having slightly lower accuracy, achieves significantly better recall, especially in *Only Regular* mode, indicating a better trade-off for fraud detection.

4. **Feature Utilization Differences**: SHAP analysis shows that XGBoost relies heavily on a single modality (ANAGRAFICA) in both modes, with even greater dependency in *All Clusters* (55.7% vs. 9.6% in *Only Regular*), indicating a bias toward non-temporal attributes when noisy embeddings are present. The DNN maintains a more balanced use of features, especially in *Only Regular*, where INTERRUZIONI and CONSUMI embeddings play a more prominent role. This diversification is linked to its superior minority class detection.

5. **Ensemble Voting**: Soft voting consistently outperforms hard voting across classifiers in both modes, with accuracy improvements of up to 3%. This suggests that probability aggregation mitigates the instability of individual models, especially in imbalanced contexts.

6. **SHAP Interpretability**: The use of SHAP values provides insights into how data cleaning strategies impact model behavior. In *Only Regular* mode, feature contributions are more evenly distributed, whereas in *All Clusters* mode, certain modalities dominate, indicating possible overfitting or noise-driven bias.

Summary of Mode Comparison:

- **Only Regular**: Better reconstruction quality, higher weighted recall, more balanced feature usage, improved anomaly detection.
- **All Clusters**: Slightly higher accuracy in some models due to easier prediction of majority class, but much lower minority class detection and more biased feature importance.

Limitations The performance metrics presented should be interpreted with caution, as the low number of available samples, especially in the positive classes, makes the results not fully representative of real-world performance.Future work should validate the proposed methodology on larger datasets to confirm its effectiveness.

10 Future Developments

Possible system extensions include:

- Implementation of attention mechanisms in autoencoders
- Use of transformers for long time series
- Apply models to a larger number of data samples

The multi-modal approach adopted in this project is supported by recent literature, which highlights the effectiveness of integrating multiple types of time series to improve performance in complex analysis tasks.

References

[1] Yushan Jiang, Kanghui Ning, Zijie Pan, Xuyang Shen, Jingchao Ni, Wenchao Yu, Anderson Schneider, Haifeng Chen, Yuriy Nevmyvaka, and Dongjin Song. Multi-modal time series analysis: A tutorial and survey, 2025.