PROYECTO FIN DE GRADO

TÍTULO: Sistema de Detección de Anomalías para Proyecciones Multidimensionales en Tiempo Real

AUTOR/A: Rubén Agustín González

TITULACIÓN: Doble grado en Ingeniería electrónica de comunicaciones e Ingeniería telemática

TUTOR/A: Julián Nieto Valhondo

DEPARTAMENTO: DTE

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: María Cristina Guerrero García

TUTOR/A: Julián Nieto Valhondo

SECRETARIO/A: Mariano Ruiz González

Fecha de lectura: Septiembre de 2025

Calificación:

El Secretario/La Secretaria,

Abstract

Abstract

Agradecimientos

Todo list

Contents

List of Figures

List of Tables

Listings

Acronyms

FFT	Fast Fourier Transform
I2A2	*Investigación en Instrumentación y Acústica Aplicada*
IForest	Isolation Forest
ITER	International Thermonuclear Experimental Reactor
LSTM	Long Short-Term Memory
ML	Machine Learning
OC-SVM	One-Class SVM
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Context and Motivation

This project has been developed as a final project for the Electronic Engineering degree at the investigation group *Investigación en Instrumentación y Acústica Aplicada* (I2A2) and belongs to the International Thermonuclear Experimental Reactor (ITER) project. The main goal is to create an outlier detection system that can be used to monitor the performance of the ITER reactor, ensuring its safe and efficient operation.

Chapter 2

State of the Art

2.1 Introduction to the ITER Project

The ITER project is an international collaboration aimed at demonstrating the feasibility of nuclear fusion as a large-scale and carbon-free source of energy. It is being constructed in Cadarache, France, and involves 35 countries, including the European Union, the United States, China, India, Japan, Russia, and South Korea. The project aims to build the world's largest tokamak reactor, which will use magnetic confinement to achieve controlled nuclear fusion.

Inside the tokamak, plasma will be heated to extremely high temperatures, allowing hydrogen isotopes to fuse and release energy. The reactor is designed to produce ten times more energy than it consumes, making it a potential game-changer in the field of energy production.

A *Discharge* is a plasma operation in the tokamak, where the plasma is created and maintained for a certain period. Each discharge is characterized by various parameters, such as plasma current, inductance, density, radiated power or input power. When these parameters deviate from their expected values, it can indicate potential issues or anomalies in the reactor's operation.

A *Disruption* is an event that occurs when the plasma becomes unstable and loses confinement, leading to a rapid cooling of the plasma and a loss of control. For this project, discharges are classified as either disruptive or non-disruptive.

A *Campaign* is a set of discharges that are analyzed together. For this project, there are three campaigns available, each containing a different number of discharges. The available data for the project is summarized in Table 1.

Table 1: Available data for the project

Campaign	**Discharges**	**Disruptive**	**Non–Disruptive**	**Disruptive Rate**
C23	522	32	490	6.13 %
C24	388	26	362	6.70 %
C25	611	41	570	6.71 %

A visual representation of the plasma current on several discharges is shown in Figure 1. This figure illustrates the singular bathtub shape of non-disruptive discharges, whereas disruptive discharges show a more erratic pattern.

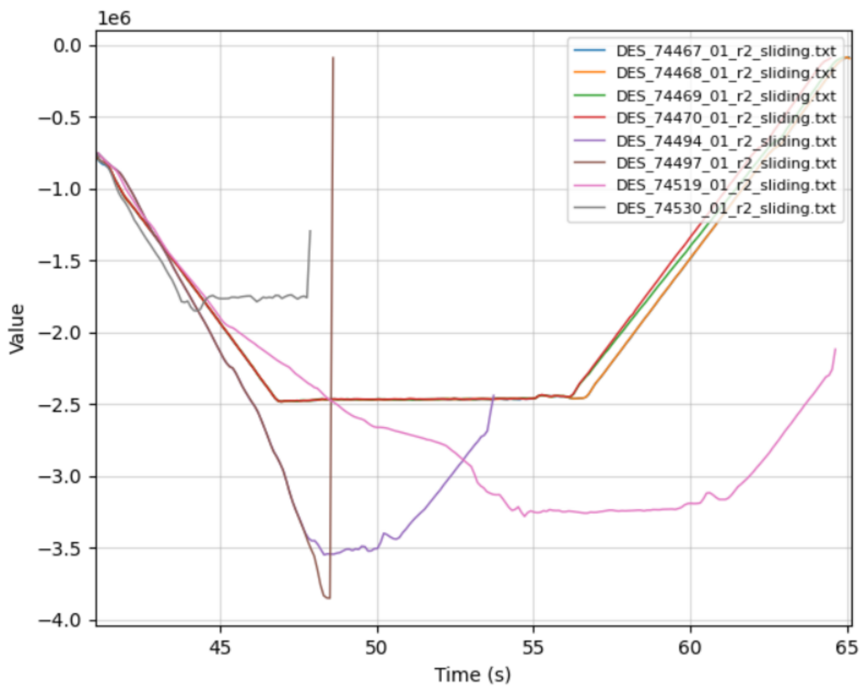


Figure 1: Plasma current on disruptive and non-disruptive discharges

2.2 Anomaly Detection in Nuclear Fusion

Anomaly detection in nuclear fusion is crucial for ensuring the safe and efficient operation of reactors like ITER. The complexity of the plasma behavior and the multitude of parameters involved make it challenging to monitor and control the system effectively. Anomalies can lead to disruptions, which can damage the reactor components and affect the overall performance.

To address this challenge, various machine learning techniques have been applied to analyze the data generated during discharges. These techniques aim to identify patterns and deviations in the data that may indicate potential anomalies. An early detection of these anomalies is essential to prevent disruptions and ensure the stability of the plasma, using gas injection to stop the nuclear fusion reaction before it causes damage to the reactor.

2.2.1 Current implementation

The current implementation is based on a Support Vector Machine (SVM) model trained on historical discharge data. The model is designed to classify discharges as normal or anomalous based on the input parameters. The training process involves using labeled data, where each discharge is categorized as either normal or anomalous.

SVM is a machine learning algorithm whose goal is to find the optimal hyperplane that separates the data into different classes [1]. On the training phase, the model receives entire discharges as input, and the category of the discharge. Then, it divides data in windows of 32 milliseconds, and extracts the mean value and the Fast Fourier Transform (FFT) of each window. On the prediction phase, the model receives a data stream, and fills a buffer to create a window. Then, it extracts the same features as in the training phase, and predicts the category of this window. If the model predicts an anomalous window, the discharge is classified as anomalous. This classification shall be done before the disruption occurs.

There are some limitations to this approach. First, a window-based approach does not take into account the temporal dependencies of the data. This means that the model may not be able to capture the dynamics of the plasma behavior over time. Second, SVM models with non-linear kernels have a complexity between $O(n^2)$ and $O(n^3)$, where n is the number of training samples. This means that the model may not be able to scale to large datasets [2]. Third, even though SVM models do not require a balanced dataset, the model could be biased towards the majority class, leading to a high false negative rate [3]. This is particularly important in this project, as the number of non-disruptive discharges is much higher than the number of disruptive discharges, as shown in Table 1.

2.3 Alternative approaches

This project aims to explore alternative approaches to anomaly detection in nuclear fusion, focusing on the use of other Machine Learning (ML) algorithms. Models used in this project can be grouped into three categories: decision trees based models, machine vector support based models and deep learning models. For each category, there are two subcategories: outlier detection and binary classification. Outlier detection models are trained on a single class of data (non-disruptive discharges), and generate a decision boundary that separates the normal data from the anomalies. Supervised binary classification models are trained on both classes of data (disruptive and non-disruptive discharges), and generate a decision boundary that separates the two classes.

2.3.1 Decision Trees based models

Decision trees are a type of ML algorithm that uses a tree-like structure to make decisions based on the input data. This structure is built at the training phase, where the algorithm recursively splits the data into subsets based on the features that provide the most information gain. The resulting tree can be used to classify new data points by following the branches of the tree based on their feature values [4].

Outlier detection

Decision trees can be used for outlier detection by training the model on a single class of data (non-disruptive discharges) and identifying instances that fall outside the normal patterns. The decision tree learns the characteristics of the normal data and can flag instances that do not conform to these patterns as potential anomalies.

Isolation Forest (IForest) is the canonical example of this approach. It splits the data at random features and thresholds, creating a forest of trees. Anomalies tend to be isolated in fewer splits than normal instances, because they lie in sparse regions of the feature space [5]. Later improvements, like Extended Isolation Forest replace axis-aligned cuts with random hyperplanes to reduce bias [6].

Binary classification

If labelled data of both classes exist, standard decision-tree classifiers or ensembles (Random Forests, Gradient Boosting) can draw an explicit boundary between them. With sufficient minority class samples, these models usually outperform outlier detection models, as the tree learns exactly which feature ranges characterize each class rather than inferring a boundary from a single class. In practice, binary trees are common in fraud detection and network-intrusion datasets, where even a modest number of confirmed attack records allows the model to reach high recall without over-flagging harmless traffic [7, 8].

2.3.2 Machine Vector Support based models

SVM is a well-known algorithm in this category, but there are other algorithms that can be used as well. For example, One-Class SVM (OC-SVM) is a variant of SVM that is trained on a single class of data (non-disruptive discharges), and generates a decision boundary that separates the normal data from the anomalies. This approach is particularly useful when the dataset highly imbalanced.

SVM, OC-SVM

2.3.3 Deep learning models

LSTM, CNN, Autoencoders

Chapter 3

Development

This chapter describes the development of the anomaly detection system, which aims to create a toolkit for anomaly detection, model training, and comparison between different models. The toolkit is designed to be modular, allowing for easy integration of new models and features.

3.1 System Design

The anomaly detection system is composed by independent nodes. There is a central node that orchestrates the system, and several nodes that implement the anomaly detection models. Each model node works as a microservice and is responsible for training and predicting anomalies using its own algorithm. The central node is responsible for managing the data flow between the model nodes and the outlier protocol. The system design is shown in Figure 2.

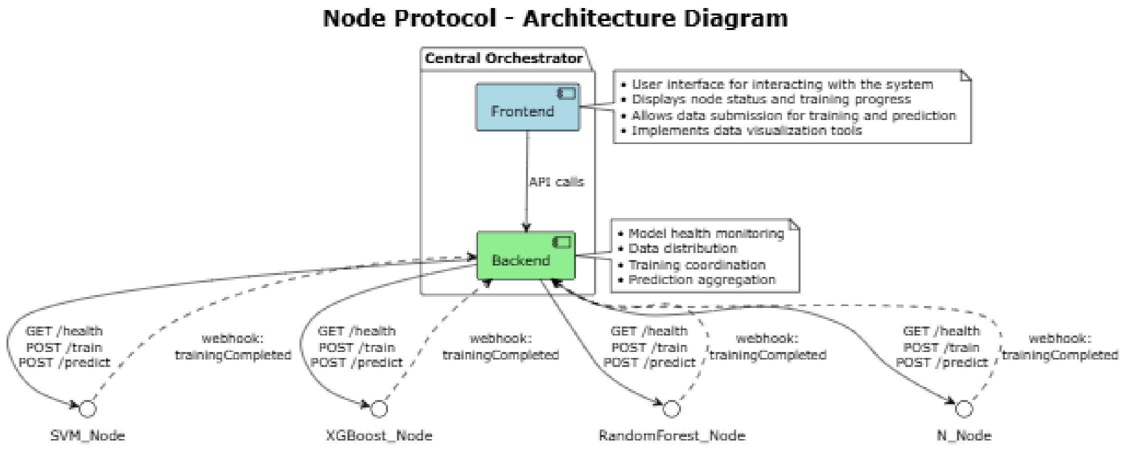


Figure 2: Anomaly detection system design

3.2 Outlier Protocol

The outlier protocol is a custom protocol based on the OpenAPI specification. It is a simple protocol that describes how the orchestrator interacts with the model nodes, and is based on HTTP messages, which can be grouped into three main categories: *health*, *train*, and *predict*.

3.2.1 Health

Models must implement the `/health` endpoint. The orchestrator will send a `get` message at a variable time. If the model does not answer, orchestrator must consider the model as unavailable, and data must not be sent to this model. Figure 3 shows the sequence diagram of the health message and its response. This message also includes the model name, uptime, and last training time.

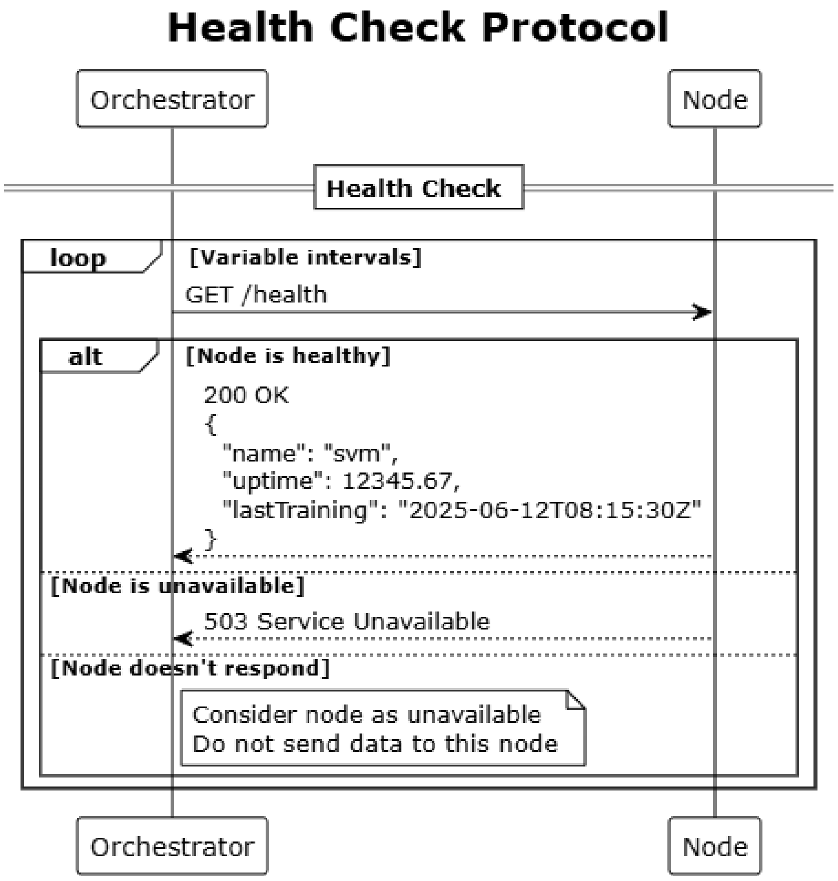


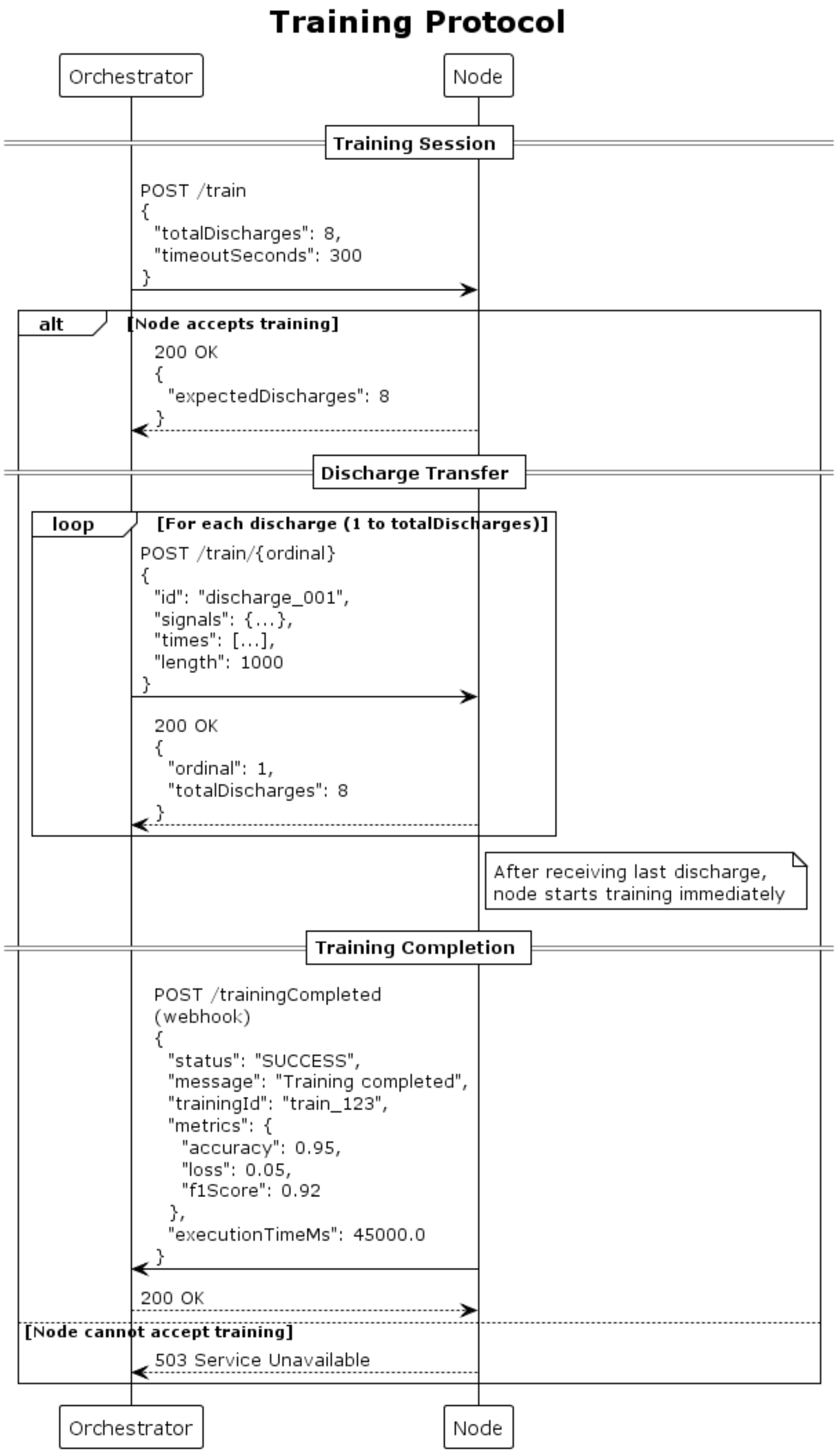
Figure 3: Health message sequence diagram

3.2.2 Train

This category describes how models will be trained in order to acquire data for disruption prediction.

Protocol begins with a `post` message to the `/train` model endpoint that contains the number of discharges that will be sent to the models. Models that want to accept the training, answer with a 200 response. Then the orchestrator sends, one by one, the training discharges. After a discharge is received, models must acknowledge it. When all discharges are sent, models shall start the training.

When the training is done, models must inform the orchestrator, as shown in Figure 4.


Training Protocol
Orchestrator
Node
Training Session
alt
[Node accepts training]
Discharge Transfer
loop
[For each discharge (1 to totalDischarges)]
After receiving last discharge, node starts training immediately
Training Completion
[Node cannot accept training]
Orchestrator
Node

```
POST /train
{
  "totalDischarges": 8,
  "timeoutSeconds": 300
}
```

```
200 OK
{
  "expectedDischarges": 8
}
```

```
POST /train/{ordinal}
{
  "id": "discharge_001",
  "signals": {...},
  "times": [...],
  "length": 1000
}
```

```
200 OK
{
  "ordinal": 1,
  "totalDischarges": 8
}
```

```
POST /trainingCompleted
(webhook)
{
  "status": "SUCCESS",
  "message": "Training completed",
  "trainingId": "train_123",
  "metrics": {
    "accuracy": 0.95,
    "loss": 0.05,
    "f1Score": 0.92
  },
  "executionTimeMs": 45000.0
}
```

```
200 OK
```

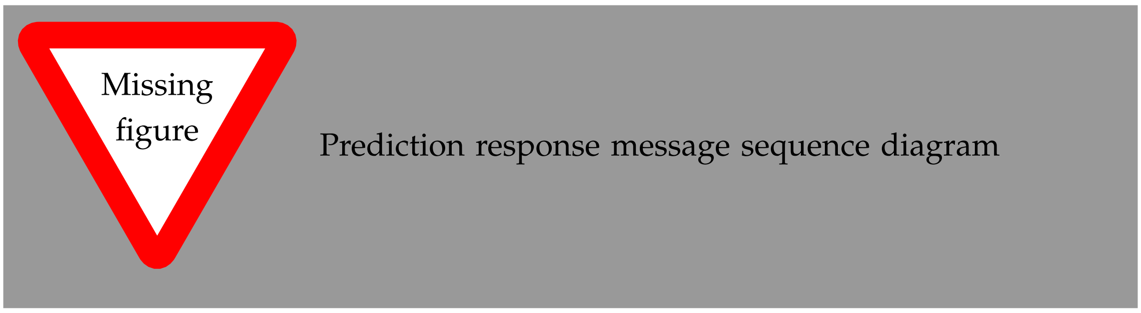
```
503 Service Unavailable
```

Figure 4: Train message sequence diagram

3.2.3 Predict

Actualizar la figura de predict. En el outlier protocol no están puestas las features que se deben enviar. GH issue 3

The predict message is sent by the orchestrator to the alive models. It is a `post` message to the `/predict` model endpoint.



3.3 Outlier Orchestrator

The outlier orchestrator is the central node of the anomaly detection system. It is responsible for managing the data flow between the model nodes, using the outlier protocol to communicate with them.

The orchestrator consists on a frontend and a backend. The frontend is a web application that allows users to interact with the system, while the backend is responsible for managing the data flow and the communication with the model nodes.

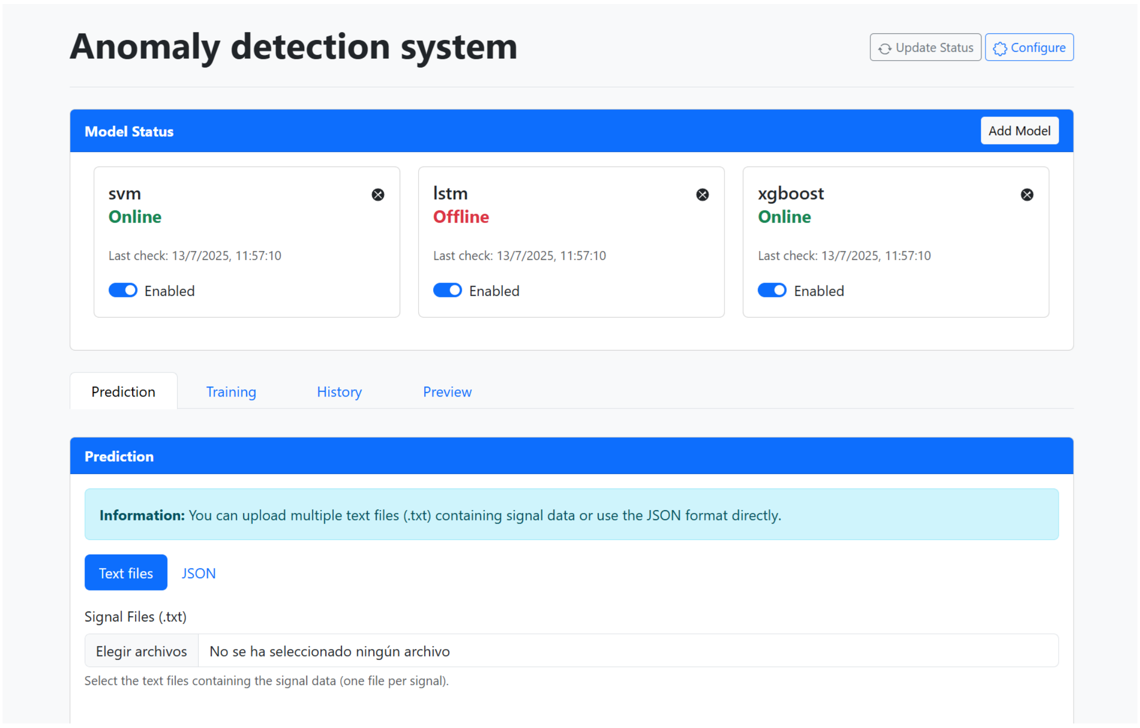
The frontend is built using HTML, CSS, and JavaScript, and provides a user-friendly interface for managing the models and the data. It allows users to start and stop the models, view the health status of the models, and visualize the results of the anomaly detection.

The backend is built using Node.js, which is a JavaScript runtime that allows to run JavaScript code on the server side. It uses the Express framework to handle the HTTP requests and responses and Axios to communicate with the model nodes, implementing the outlier protocol.

Communication between the frontend and the backend is done Socket.IO, which is a library that allows real-time communication between the client and the server. This allows the frontend to receive updates from the backend in real-time, such as the health status of the models or the results of the anomaly detection.

3.3.1 Frontend

The frontend main page is shown in Figure 5. There are two main panels: the top panel shows the available models, and the bottom panel consists on four tabs: *Prediction*, *Training*, *History*, and *Preview*.


Anomaly detection system

Update Status | Configure

Model Status — Add Model

svm
Online
Last check: 13/7/2025, 11:57:10
Enabled

lstm
Offline
Last check: 13/7/2025, 11:57:10
Enabled

xgboost
Online
Last check: 13/7/2025, 11:57:10
Enabled

Prediction | Training | History | Preview

Prediction

Information: You can upload multiple text files (.txt) containing signal data or use the JSON format directly.

Text files | JSON

Signal Files (.txt)

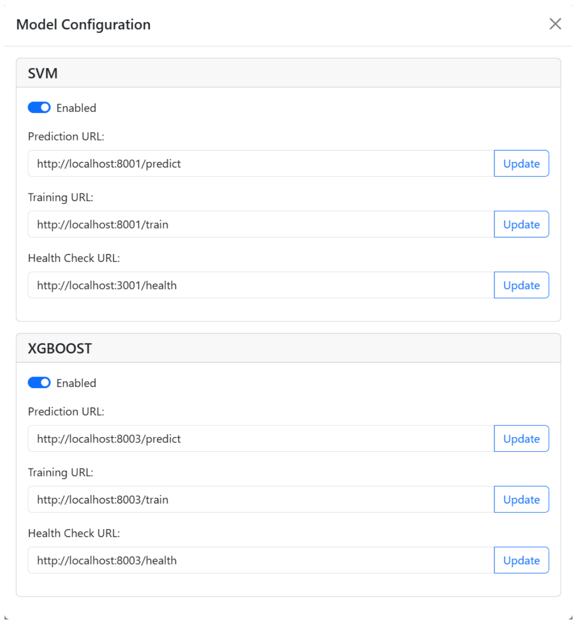
Elegir archivos | No se ha seleccionado ningún archivo

Select the text files containing the signal data (one file per signal).

Figure 5: Frontend main page

The backend is responsible for sending the model status to the frontend, including the health status, the training status and the prediction results, while the frontend is responsible of displaying this information to the user and sending the backend the user actions, such as adding or removing models, enabling or disabling models, changing model endpoints, starting a training, or making a prediction.

Model configuration is done through the *Configure* button, which opens a modal where the user can change the model's endpoints. This modal is shown in Figure 6.


Model Configuration

SVM

Enabled

Prediction URL: http://localhost:8001/predict — Update

Training URL: http://localhost:8001/train — Update

Health Check URL: http://localhost:3001/health — Update

XGBOOST

Enabled

Prediction URL: http://localhost:8003/predict — Update

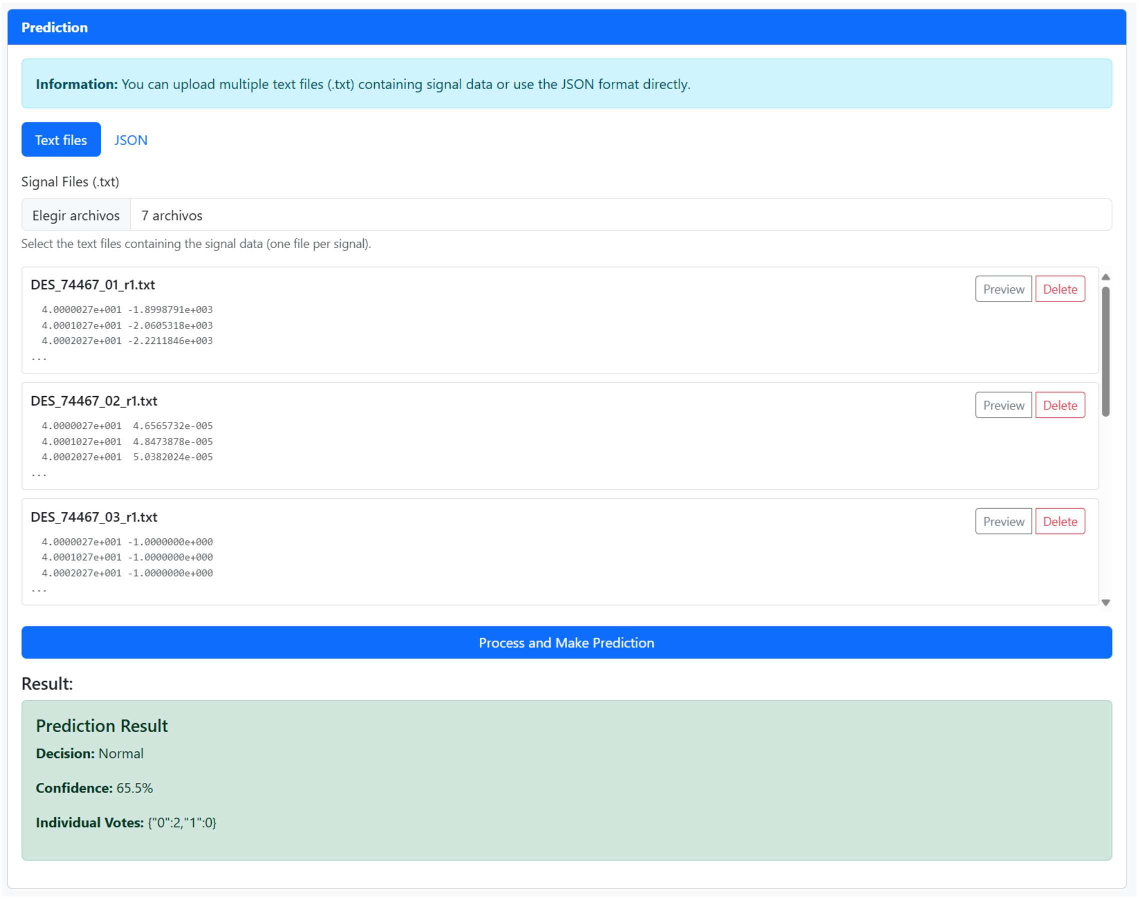
Training URL: http://localhost:8003/train — Update

Health Check URL: http://localhost:8003/health — Update

Figure 6: Model configuration modal

Prediction Tab

The prediction tab allows users to make predictions using the available models. It provides a simple interface where users can choose the discharge to analyze. The Figure 7 shows the prediction tab and the result of a prediction.



Information: You can upload multiple text files (.txt) containing signal data or use the JSON format directly.

Signal Files (.txt)

Elegir archivos 7 archivos

Select the text files containing the signal data (one file per signal).

DES_74467_01_r1.txt

```
  4.0000027e+001 -1.8998791e+003
  4.0001027e+001 -2.0605318e+003
  4.0002027e+001 -2.2211846e+003
...
```

DES_74467_02_r1.txt

```
  4.0000027e+001  4.6565732e-005
  4.0001027e+001  4.8473878e-005
  4.0002027e+001  5.0382024e-005
...
```

DES_74467_03_r1.txt

```
  4.0000027e+001 -1.0000000e+000
  4.0001027e+001 -1.0000000e+000
  4.0002027e+001 -1.0000000e+000
...
```

Result:

Prediction Result

Decision: Normal

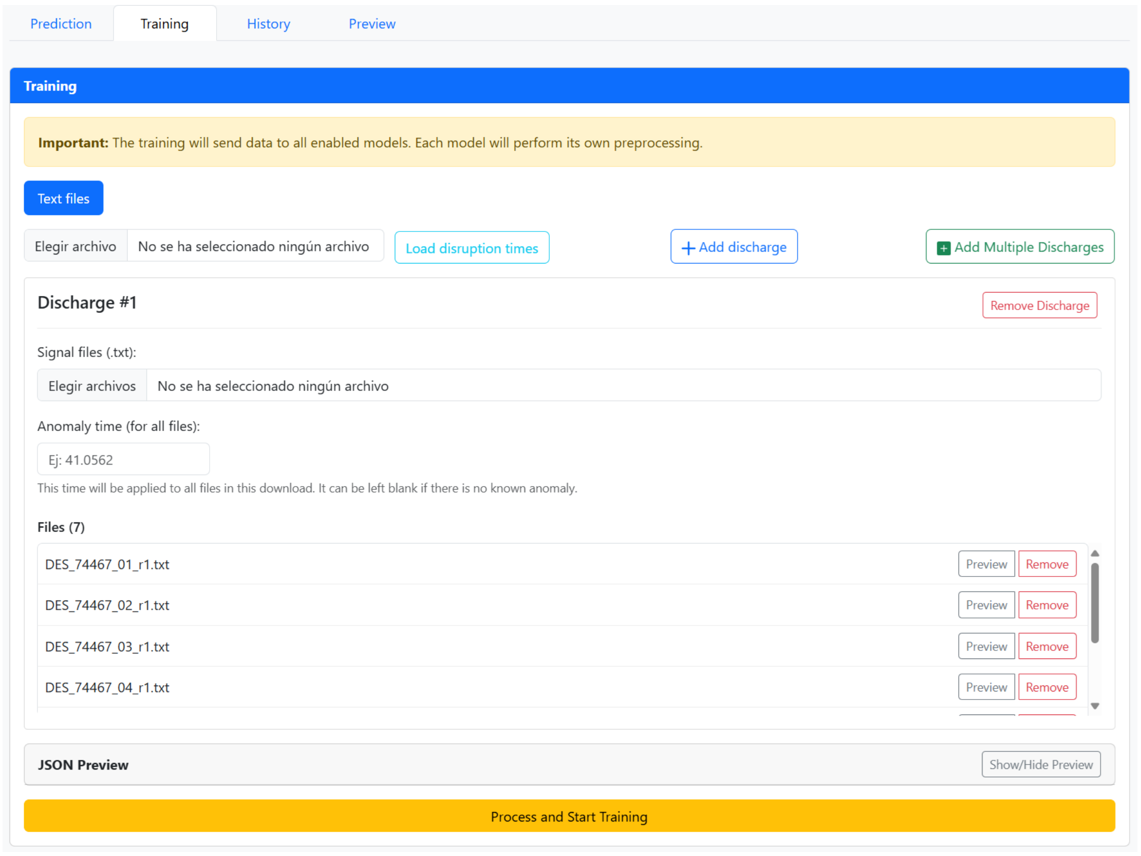
Confidence: 65.5%

Individual Votes: {"0":2,"1":0}

Figure 7: Frontend prediction tab

Training Tab

The training tab (Figure 8) is similar to the prediction tab, but users shall select more than one discharge to train the models. For simplicity, there is a button that allow to add multiple discharges at once, opening a modal that allows to select multiple input files, group them using a custom regular expression, and exclude some files with another regular expression. This modal is shown in Figure 9. Discharges can also be labeled as disruptive or non-disruptive by setting a disruption time.


Prediction | Training | History | Preview

Training

Important: The training will send data to all enabled models. Each model will perform its own preprocessing.

Text files

Elegir archivo | No se ha seleccionado ningún archivo | Load disruption times | Add discharge | Add Multiple Discharges

Discharge #1 | Remove Discharge

Signal files (.txt):

Elegir archivos | No se ha seleccionado ningún archivo

Anomaly time (for all files):

Ej: 41.0562

This time will be applied to all files in this download. It can be left blank if there is no known anomaly.

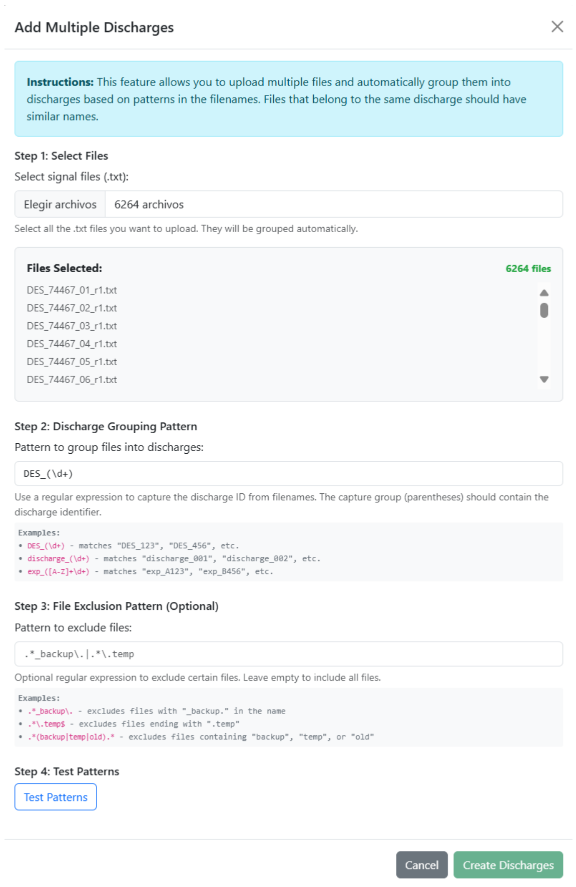
Files (7)

DES_74467_01_r1.txt | Preview | Remove
DES_74467_02_r1.txt | Preview | Remove
DES_74467_03_r1.txt | Preview | Remove
DES_74467_04_r1.txt | Preview | Remove

JSON Preview | Show/Hide Preview

Process and Start Training

Figure 8: Frontend training tab


Add Multiple Discharges

Instructions: This feature allows you to upload multiple files and automatically group them into discharges based on patterns in the filenames. Files that belong to the same discharge should have similar names.

Step 1: Select Files

Select signal files (.txt):

Elegir archivos | 6264 archivos

Select all the .txt files you want to upload. They will be grouped automatically.

Files Selected: 6264 files

DES_74467_01_r1.txt
DES_74467_02_r1.txt
DES_74467_03_r1.txt
DES_74467_04_r1.txt
DES_74467_05_r1.txt
DES_74467_06_r1.txt

Step 2: Discharge Grouping Pattern

Pattern to group files into discharges:

`DES_(\d+)`

Use a regular expression to capture the discharge ID from filenames. The capture group (parentheses) should contain the discharge identifier.

```
Examples:
• DES_(\d+) - matches "DES_123", "DES_456", etc.
• discharge_(\d+) - matches "discharge_001", "discharge_002", etc.
• exp_([A-Z]+\d+) - matches "exp_A123", "exp_B456", etc.
```

Step 3: File Exclusion Pattern (Optional)

Pattern to exclude files:

`.*_backup\.|.*\.temp`

Optional regular expression to exclude certain files. Leave empty to include all files.

```
Examples:
• .*_backup\. - excludes files with "_backup." in the name
• .*\.temp$ - excludes files ending with ".temp"
• .*(backup|temp|old).* - excludes files containing "backup", "temp", or "old"
```

Step 4: Test Patterns

Test Patterns

Cancel | Create Discharges

Figure 9: Add multiple discharges modal

3.3.2 Backend

3.4 Outlier Models

Every model node shall implement the outlier protocol in order to communicate with the orchestrator. This means that every model is a microservice implementing a server that exposes the necessary endpoints to the orchestrator. The models can be implemented in any programming language, as long as they can handle HTTP requests and responses. Models in this project are implemented in Python and Rust.

3.4.1 Supervised Binary Classification

SVM

This project implements a SVM model for anomaly detection, which is trained on both disruptive and non-disruptive discharges. The model is implemented in pure Rust, implementing the outlier protocol using the `actix_server` framework for the comunication with the orchestrator's backend, and the `linfa` crate for the SVM implementation.

The model extracts two features from the discharges: the mean value of the plasma current and the FFT of the plasma current. This model can be used to classify with a very low number of training samples, and it can also be used for real-time predictions.

XGBoost

CNN-FFT

3.4.2 Outlier Detection

OC-SVM

Isolation Forest

LSTM

Chapter 4

Budget

Chapter 5

Impact

Chapter 6

Conclusions and future work

Bibliography

[1] A. Patle **and** D. S. Chouhan, "SVM kernel functions for classification," **in***2013 International Conference on Advances in Technology and Engineering (ICATE)* 2013, **pages** 1–9. DOI: `10.1109/ICAdTE.2013.6524743`.

[2] C. Kekulawala. "Support Vector Machines: A mathematical guide — Part 3," Medium, **urlseen** 12 **july** 2025. **url**: `https://medium.com/@ckekula/support-vector-machines-a-mathematical-guide-part-3-78fd3c8bf44c`.

[3] R. Akbani, S. Kwek **and** N. Japkowicz, "Applying support vector machines to imbalanced datasets," **in***Machine Learning: ECML 2004* J.-F. Boulicaut, F. Esposito, F. Giannotti **and** D. Pedreschi, **editors**, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, **pages** 39–50, ISBN: 978-3-540-30115-8.

[4] L. Rokach **and** O. Maimon, "Top-down induction of decision trees classifiers - a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, **jourvol** 35, **number** 4, **pages** 476–487, 2005. DOI: `10.1109/TSMCC.2004.843247`.

[5] F. T. Liu, K. Ting **and** Z.-H. Zhou, "Isolation forest," **january** 2009, **pages** 413–422. DOI: `10.1109/ICDM.2008.17`.

[6] S. Hariri, M. Kind **and** R. Brunner, "Extended Isolation Forest with Randomly Oriented Hyperplanes," *IEEE Transactions on Knowledge and Data Engineering*, **jourvol** PP, **pages** 1–1, 31 **october** 2019. DOI: `10.1109/TKDE.2019.2947676`.

[7] A. Guezzaz, S. Benkirane, M. Azrour **and** S. Khurram, "A reliable network intrusion detection approach using decision tree with enhanced data quality," *Security and Communication Networks*, **jourvol** 2021, **august** 2021. DOI: `10.1155/2021/1230593`.

[8] J. Leevy, J. Hancock, T. Khoshgoftaar **and** A. Abdollah Zadeh, "Investigating the effectiveness of one-class and binary classification for fraud detection," *Journal of Big Data*, **jourvol** 10, **october** 2023. DOI: `10.1186/s40537-023-00825-1`.