Auto-Defense ML Testing Document

Contents

[Testing Document 2](#_Toc167715320)

[1. Introduction 2](#_Toc167715321)

[2. Objective 2](#_Toc167715322)

[3. System Overview 2](#_Toc167715323)

[3.1. Modules 2](#_Toc167715324)

[3.2. Main Components 3](#_Toc167715325)

[4. Environment 4](#_Toc167715326)

[4.1. Hardware 4](#_Toc167715327)

[4.2. Software 4](#_Toc167715328)

[5. Scope and Methodology 5](#_Toc167715329)

[5.1 Unit Testing 5](#_Toc167715330)

[5.2 Integration Testing 5](#_Toc167715331)

[5.3 System Testing 6](#_Toc167715332)

[5.4 Usability Testing 6](#_Toc167715333)

[6. Test Suite 7](#_Toc167715334)

[6.1. Core Module Tests 7](#_Toc167715335)

[6.2. UI Module Tests 8](#_Toc167715336)

[6.3. Reports Module Tests 9](#_Toc167715337)

[6.4. Integration Tests 10](#_Toc167715338)

[Conclusion 11](#_Toc167715339)

# 

# Testing Document

## Introduction

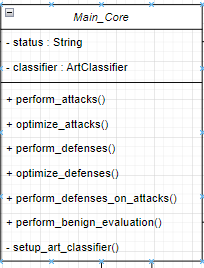
This document outlines the testing strategy and procedures for the automated system developed to run and evaluate attacks and defenses on machine learning models.  
The system currently supports tabular data and XGBoost models, with input paths for the model, x\_test, and y\_test.   
The project is divided into three main modules: Core, UI, and Reports.   
This document will provide detailed test cases for each module and integration tests to ensure the system functions correctly.

## Objective

The objective of this testing document is to ensure the reliability, accuracy, and robustness of the automated system. By performing thorough testing, we aim to identify and fix any issues, ensuring the system can handle various scenarios and provide accurate evaluations of machine learning models under different attack and defense strategies, as well as the generated report.

## System Overview

### 3.1. Modules

- **Core:** Backend functionality including attack execution, defense application, and metrics evaluation.  


- **UI:** User interface for interacting with the system.

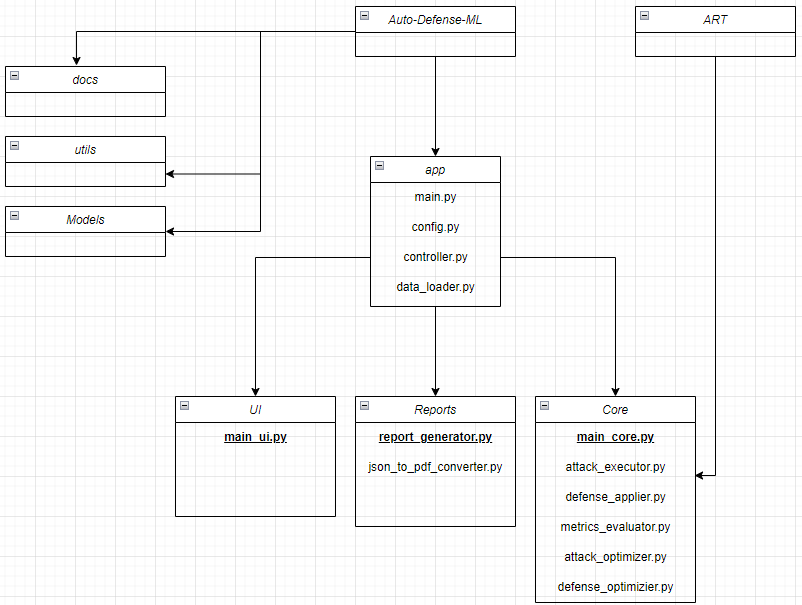
A screen shot of a graph

Description automatically generated  
- **Reports:** Generation of detailed reports based on the evaluation results.  
A screenshot of a computer

Description automatically generated

### 3.2. Main Components

**- `controller.py`:** Manages the integration between Core, UI, and Reports modules.  
**- `config.py`:** Contains configuration parameters for attacks and defenses.  
**- `data\_loader.py`:** Handles loading of model and test data.  
**- `main.py`:** Entry point for the application.



## Environment

### 4.1. Hardware

Standard development machines with at least 8GB RAM, i5 processor.

### 4.2. Software

* Python 3.11.7
* ART library
* XGBoost library
* Dearpygui library
* Pandas, NumPy for data manipulation
* Additional Python libraries as specified in requirements.txt

## Scope and Methodology

### 5.1 Unit Testing

#### 5.1.1 Objective

Ensure individual modules and functions perform as expected.

#### 5.1.2 Scope

* Core module functions
* UI module functions
* Reports module functions
* Data loading functionalities
* Configuration parsing

#### 5.1.3 Tools

* unittest or pytest

### 5.2 Integration Testing

#### 5.2.1 Objective

Verify that the interaction between different modules (Core, UI, Reports) works as intended.

#### 5.2.2 Scope

* Interaction between controller and modules
* Data flow from input to report generation

#### 5.2.3 Tools

* unittest or pytest
* Mocking frameworks

### 5.3 System Testing

#### 5.3.1 Objective

#### Test the entire system's functionality and performance in an environment that mimics production, including monitoring and evaluating CPU and RAM usage to ensure optimal resource utilization and system stability.

#### 5.3.2 Scope

* Full system functionalities
* Realistic data inputs

#### 5.3.3 Tools

* Custom scripts for automated end-to-end tests

### 5.4 Usability Testing

#### 5.4.1 Objective

Ensure the user interface is intuitive and user-friendly.

#### 5.4.2 Scope

* UI/UX of the system

#### 5.4.3 Tools

* User feedback forms
* Usability testing sessions

## Test Suite

### Core Module Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Description | Preconditions | Test Steps | Expected Results | Status |
| TC\_Core\_01 | Test dataloader setter | Dataloader is an instance of DataLoader | 1. Create a DataLoader instance 2. Set the dataloader property of Main\_Core | Dataloader is set without errors |  |
| TC\_Core\_02 | Test optimize\_attacks method | Dataloader and classifier are set | 1. Call optimize\_attacks with a list of attacks | Optimized attacks are returned |  |
| TC\_Core\_03 | Test perform\_attacks method | Dataloader and classifier are set | 1. Call perform\_attacks with a list of attacks | Metrics and adversarial examples are returned |  |
| TC\_Core\_04 | Test perform\_defenses method | Dataloader and classifier are set | 1. Call perform\_defenses with a list of defenses | Metrics and defended examples are returned |  |
| TC\_Core\_05 | Test perform\_defenses\_on\_attacks method | Dataloader and classifier are set, adversarial examples are available | 1. Call perform\_defenses\_on\_attacks with a list of defenses and adversarial examples | Metrics and defended adversarial examples are returned |  |
| TC\_Core\_06 | Test perform\_benign\_evaluation method | Dataloader and classifier are set | 1. Call perform\_benign\_evaluation | Clean evaluation metrics are returned |  |
| TC\_Core\_07 | Test optimize\_defenses method | Dataloader and classifier are set | 1. Call optimize\_defenses with a list of defenses | Optimized defenses are returned |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### UI Module Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Description | Preconditions | Test Steps | Expected Results | Status |
| TC\_UI\_01 | Test UI initialization | N/A | 1. Initialize Main\_UI | UI initializes without errors |  |
| TC\_UI\_02 | Test setting controller | Controller instance is available | 1. Set the controller in UI | Controller is set without errors |  |
| TC\_UI\_03 | Test running UI | Controller is set | 1. Call run method | UI runs and is interactive |  |
| TC\_UI04\_ | Test load a model | 1. UI runs and is interactive.  2. A valid model was uploaded. | 1. Call load\_model method | The right type of model is saved (e.g xGBoost) in data\_loader.py |  |
| TC\_UI\_05 | Test load an x test and y test | 1. UI runs and is interactive  2. A valid path for data was uploaded | 1. Call load\_test method | An array-like object is created containing the test data |  |
| TC\_UI\_06 | Test “load “button | 1. UI runs and is interactive  2. Model and test paths loaded | 1. Call load\_files handler | Attack and defences window is up and interactive |  |
| TC\_UI\_07 | Test configure manually parameters | 1. UI runs and is interactive  2. ‘Configure manually’ radiobutton is chosen | 1. Call “on\_proceed\_with\_selection” handler | 1. Configure parameters window is up  2. With the default configured values |  |
| TC\_UI\_08 | Test default chosen parameters | 1. UI runs and is interactive  2. ‘Run on default parameters’ radiobutton is chosen | 1. Call “on\_proceed\_with\_selection” handler | 1. Progress\_window is shown  2. Controller.handle\_configuration is called |  |
| TC\_UI\_09 | Test progress bar | 1. UI runs and is interactive  2. Controller.handle\_configuration is called | 1. “Update\_progress” method is called after the proper stage | 1. Progress bar is shown correctly if there are no bugs in ‘Core’ module |  |
|  |  | 1. UI runs and is interactive |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### Reports Module Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Description | Preconditions | Test Steps | Expected Results | Status |
| TC\_Reports\_01 | Test report generation | Evaluation metrics are available | 1. Call generate\_report with metrics | Report is generated without errors |  |
| TC\_Reports\_02 | Test PDF report creation | Evaluation metrics are available | 1. Call generate\_pdf with metrics | PDF report is created without errors |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### Integration Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Description | Preconditions | Test Steps | Expected Results | Status |
| TC\_Int\_01 | Test data loading through controller | Model, x\_test, and y\_test paths are available | 1. Call handle\_load with valid paths and library | Dataloader is created and set in Core |  |
| TC\_Int\_02 | Test configuration handling | Dataloader is set in Core | 1. Call handle\_configuration with attacks, defenses, and chosen\_run | Pipeline starts and executes correctly |  |
| TC\_Int\_03 | Test main pipeline execution | Dataloader is set in Core | 1. Call start\_main\_pipeline with attacks and defenses | Clean metrics, attack metrics, defense metrics, and combined metrics are returned |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Conclusion

This document provides a structured approach to testing the automated system for evaluation of attacks and defenses on machine learning models. By following these test cases, we ensure the system's components are thoroughly evaluated, and any issues are identified and resolved promptly. This will contribute to the system's reliability and effectiveness in providing robust evaluations of machine learning models under various scenarios.