
# Applying and Testing Multi-Class and Multi-Output Algorithms in the Mapping of Security Requirements with Technologies and Best Practices

**Pedro Miguel Marques Batista**

Dissertação para obtenção do Grau de Mestre em
**Engenharia Informática**
(2º ciclo de estudos)

Orientador: Doutor Pedro Ricardo Morais Inácio
Coorientador: Doutor Hugo Pedro Martins Carriço Proença

**Covilhã, junho de 2022**

# Acknowledgements

I would like to thank all persons that provided support and help, and allowed me to get through the writing of this dissertation.

First, I would like to thank Professor Pedro Ricardo Morais Inácio for all his support and availability, as well as good humour and guidance, that helped me have a sense of direction through this project, as well as giving a meaningful and interesting topic that can be of use to the digital world. Furthermore, in a professional but also friendly sense, I would like to thank Joana Costa and Bernardo Sequeiros for their input and help, as well as for being available, friendly and supportive throughout the development of this project.

I would also like to thank my family for their love and support, that was essential to reaching this point, and for never doubting my capabilities. Furthermore, I would like to thank my friends Sofia Ascensão, Pedro Cavaleiro, Igor Nunes, Raquel Guerra, Diogo Simões, and Cristiano Santos for always putting me in a good mood and helping when I needed, be it with overall support or technical input, also hoping that their objectives can be fulfilled in the best way.

To my lifelong friends, João Paulo, Bea Marcelino, Inês Peres, Telmo Silva, and Bruno Silva, I would like to thank for all the good moments that were essential to be motivated in my academic life, leading me to get this far. Thank you for always believing in me.

To Alexandre Fonseca, my greatest thanks for all the inputs, technical opinions and being one of the best friends I could have asked for during my entire academic path.

To Daniela Silveira, I would like to thank for always being there and always believing in me, for all the love and support, and for giving me strength when I needed it the most, strength that helped me surpass many challenges that came with the writing and development of this dissertation.

Likewise, I would also like to thank my friends who have not been particularly mentioned here but who helped me and supported me through my academic path.

# Resumo

A Internet das Coisas está em constante crescimento, devido aos benefícios e vantagens que traz aos utilizadores no seu dia a dia. A popularidade de dispositivos pertencentes a este paradigma fez com que fosse cobiçado o seu fabrico e venda, muitas vezes de uma maneira acelerada e pouco cuidada. De maneira a tentar minimizar este problema, o projeto SECURIoTESIGN foi desenvolvido, onde se encontra a plataforma *Security Advisory Modules (SAM)*, que contém vários módulos relevantes para diferentes partes do desenvolvimento de uma aplicação, os quais fazem certas recomendações do que implementar de maneira a tornar estes dispositivos mais seguros.

Os módulos utilizam questionários, cujas respostas geram um conjunto de recomendações, sendo este conjunto alcançado utilizando heurísticas ou regras fixas. De maneira a facilitar o desenvolvimento destes módulos e as recomendações fornecidas, bem como minimizar a necessidade de um especialista em segurança, foi sugerida a implementação de aprendizagem automática para efetuar essa atribuição. Nesta dissertação é relatada a pesquisa efetuada para aplicar aprendizagem automática, os detalhes dessa implementação em dois módulos desta plataforma, inclusive a criação de um conjunto de dados, referente às possibilidades de resposta ao questionário, automaticamente. Adicionalmente foi feita a análise desse conjunto de dados, o aumento artificial do conjunto e o uso dos mesmos para treinar e testar vários modelos multi-classe e *multi-output* com várias variações no tamanho dos dados utilizados, de maneira a testar várias possibilidades de disponibilização de talento e recursos no projeto.

Foi concluído que o uso de algoritmos multi-classe e *multi-output* apresentou resultados positivos com o uso de diferentes tamanhos do conjunto de dados, levando à conclusão que a implementação destes modelos nestes módulos pode ser uma mais-valia e ajudar no desenvolvimento futuro de módulos nesta plataforma.

# Palavras-chave

Aprendizagem Automática, Algoritmos Multi-output, Classificação Multi-classe, Classificação Multi-label, Cibersegurança, Estrutura de segurança, Internet das coisas, Segurança por desenho

# Resumo alargado

## Introdução

O presente capítulo detalha a motivação por detrás da elaboração desta dissertação, apresentando as razões por detrás do projeto original, o qual será expandido e adaptado parcialmente, de maneira a permitir o uso do desenvolvido no âmbito desta dissertação, bem como a necessidade, objetivos e contribuições relacionadas com ela.

## Enquadramento e Objetivos

A Internet das Coisas está em constante crescimento, devido aos benefícios e vantagens que traz aos utilizadores no seu dia a dia. A popularidade de dispositivos pertencentes a este paradigma fez com que fosse cobiçado o seu fabrico e venda, muitas vezes de uma maneira acelerada e pouco cuidada. Sendo dispositivos relevantes ao dia a dia de um utilizador, e muitas vezes lidando com dados pessoais e críticos, a pressa na venda e fabrico, a ignorância sobre o que implementar e a falta de recursos para investigar pode levar a que muitos destes dispositivos sejam vendidos com graves falhas de segurança. De maneira a tentar minimizar este problema, o projeto **SECURIoTESIGN** foi desenvolvido, onde se encontra a plataforma *Security Advisory Modules (SAM)*, que contém vários módulos relevantes a diferentes partes do desenvolvimento de uma aplicação, os quais fazem certas recomendações do que implementar de maneira a tornar estes dispositivos mais seguros. De maneira a evoluir este projeto, foi sugerida a implementação de aprendizagem automática nos seus módulos, utilizando algoritmos multi-classe e *multi-output*, sendo o objetivo principal a implementação no módulo *Security Requirements Elicitation (SRE)*.

Com base no mencionado acima, objetivos foram criados de maneira a facilitar o futuro desenvolvimento da plataforma, acelerar a possível implementação de aprendizagem automática nos seus módulos, e de verificar a sua eficiência e utilidade na plataforma. Foram definidos como objetivos 1) analisar desenvolvimento prévio efetuado para o mesmo efeito, verificando o que poderia ser aproveitado e utilizando a informação recolhida como base para esta dissertação; 2) automatizar a criação do conjunto de dados, seguindo a lógica geral do maior número de módulos possível, sendo assim possível usar o mesmo processo para gerar esse conjunto, precisando de pouca ou nenhuma adaptação; 3) analisar os conjuntos de dados criados em termos de balanço das suas classes de saída, bem como a correlação entre elas; 4) testar, utilizando algoritmos descritos previamente, com uma quantidade alta, baixa e mínima de dados, verificando quais modelos de aprendizagem automática funcionam melhor em cada experiência; e 5) expandir esta adaptação para outros módulos pertencentes à plataforma.

## Principais Contribuições

Tendo em conta os objetivos previamente propostos, as principais contribuições deste projeto são portanto 1) Implementação de aprendizagem automática no módulo SRE da plataforma SAM 2) testar e verificar se essa implementação é eficiente e fidedigna com vários tipos de

conjuntos de dados; e 3) implementar noutros módulos efetuando as mesmas verificações que descritas em 2).

### Estado da Arte e Trabalho Prévio

Este capítulo aborda três tópicos principais, o estado da plataforma SAM, o estado de *Classificação multilabel*, e o trabalho previamente efetuado para implementar aprendizagem automática no módulo SRE. Começando pelo estado da plataforma, é feita uma análise dos seus objetivos, dos seus módulos, como estão estruturados, o seu significado, como são fornecidas as recomendações ao utilizador e finalmente como são armazenados os dados. As principais conclusões retiradas desta análise é que a maioria dos módulos interage com o utilizador através de questionários, de escolha múltipla, cuja resposta irá originar um conjunto de recomendações. Adicionalmente foi verificado que os dados são armazenados numa base de dados, que será futuramente analisada de maneira a ajudar na automatização da criação do conjunto de dados.

Após analisar o estado de *Classificação multilabel*, foi possível verificar que os algoritmos existentes estão divididos em três grupos principais sendo estes Transformação de Problemas, Adaptação de Algoritmos e *Ensembles*. Para a Transformação de Problemas foram explorados diversos algoritmos, sendo os mais importantes *Binary Relevance (BR)*, *Classifier Chains (CC)*, *Label Powerset (LP)* e métodos *Pair-wise*, sendo feita uma análise de cada um. Para a adaptação de algoritmos foram explorados algoritmos adaptados a *Classificação multilabel* baseados em *boosting*, árvores de decisão, redes neuronais, *Support Vector Machines* e adicionalmente o modelo *k-nearest neighbours*. No caso dos *Ensembles*, foram explorados os modelos *AdaBoost.MH*, *Ensembles of LP Classifiers*, *Ensemble of Pruned Sets* e *RAndom k-labELsets*.

Na análise do trabalho prévio foi possível observar que aprendizagem automática já tinha sido implementada e testada com algum sucesso no módulo SRE. Contudo, apresentava algumas necessidades de evolução, sendo estas a criação automática do conjunto de dados baseado com a informação presente na base de dados, de maneira a ser mais fácil adaptar a restantes módulos, testes extra de velocidade com certos modelos e classificadores base, testes extra com poucos dados e uma possibilidade de balancear o conjunto de dados possivelmente através de geração de dados artificiais. Foi possível usar a implementação prévia como base, adotando modelos usados com sucesso e a lógica de transformação do questionário num conjunto de dados apto para o treino desses modelos.

Juntando as três análises feitas, foi possível escolher modelos com níveis de prioridade de implementação e teste.

### Detalhes de Implementação

Neste capítulo foram resumidos detalhes importantes relativos à implementação, que podem ser úteis para desenvolvedores futuros do SECURIoTESIGN , descrições das experiências efetuadas para contextualizar possíveis realidades onde poderia haver muitos, poucos ou praticamente nenhuns dados para treino, bem como detalhes sobre o processo ETL criado para criar o conjunto de dados e informação sobre como foram aumentados os conjuntos de

dados e técnicas utilizadas para dividir o conjunto em treino e teste.

Detalhes explicitados neste âmbito foram: 1) equipamento utilizado, permitindo assim uma contextualização na velocidade obtida nos treinos dos modelos; 2) linguagens de programação utilizadas para o desenvolvimento e bibliotecas de terceiro implementadas para ajudar no desenvolvimento, sendo *python* a linguagem escolhida devido ao seu forte uso no paradigma da aprendizagem automática, bem como por ser a linguagem utilizada para criar módulos para a plataforma SAM, facilitando assim a sua integração.

Foram criadas situações hipotéticas de maneira a melhor enquadrar a necessidade de haver testes com diferentes quantidades de dados de treino e teste, tendo sido assim criadas três experiências: a) esta experiência é focada no tempo de execução e pretende simular uma situação ideal, em que todos os desenvolvedores estão disponíveis, bem como os especialistas no assunto do questionário, assim sendo representativo de um elevado número de dados; b) esta experiência é focada na confiança ao usar poucas instâncias para treinar, pretende simular a falta de um especialista relativo ao tema do questionário, levando assim a que exista um número baixo e aleatório de dados de treino; e c) esta experiência pretende simular haver apenas um especialista a criar os dados de treino, mas serem criados linha a linha baseados em casos reais, utilizando assim um número mínimo de dados, mas baseados em situações realistas.

O processo ETL criado permitiu automatizar a criação do conjunto de dados utilizado para treinar e testar os modelos. Primeiramente foi feita uma análise ao que deveriam ser os nossos dados de entrada e de saída, sendo as respostas ao questionário os dados de entrada, e as recomendações os dados de saída. Para a parte da extração, foi analisada a base de dados da plataforma de maneira a verificar quais os elementos de cada questionário relevantes aquando da criação do conjunto, sendo: a ordem da questão (no questionário); o número de possíveis respostas; as recomendações atribuídas ou retiradas baseadas em cada resposta; dependências em respostas específicas; e se podiam ser escolhidas múltiplas respostas. Adicionalmente, na parte da transformação, foram efetuadas transformações de maneira a limpar e otimizar os dados, tendo sido as principais a síntese, eliminação de dados duplicados e limpeza e adaptação de certos campos. Finalmente na parte do *load* foram criadas tabelas de dados não normalizadas de maneira a mais facilmente aceder a informação relevante sobre os questionários e as recomendações geradas. Este processo permitiu assim a automatização da geração do conjunto de dados, apenas ligeiras alterações à base de dados existente, que não afetariam o seu funcionamento atual.

Para finalizar este capítulo, foi analisada a possibilidade de haver um aumento artificial dos dados, sendo assim decidido usar o algoritmo *Synthetic Minority Over-sampling Technique* em conjunto com uma distribuição teste e treino gerada por estratificação de dados.

### Análise dos Questionários

De maneira a ter uma melhor perceção dos resultados, e do questionário em si, foi feita uma análise para cada um dos módulos em que foi implementada aprendizagem automática, sendo estes o SRE e *Security Best Practice Guidelines (SBPG)*, focando-se primeiramente em

alguns elementos específicos, sendo estes: 1) o número questões existentes; 2) o número de recomendações; 3) o número de questões que são sub-questões (cujo aparecimento depende de respostas específicas); 4) o número de combinações possíveis de resposta; 5) o número de combinações possíveis de recomendações; e 6) o número de combinações possíveis de recomendações presentes nos conjuntos de dados gerados.

Seguidamente, foram analisadas as recomendações que eram geradas, listadas e verificadas que questões poderiam atribuir a sua recomendação, desta maneira foi possível verificar que algumas recomendações estavam inteiramente dependentes de perguntas de sim ou não, e outras apenas de uma única resposta de uma pergunta, podendo assim levar a encravos na aprendizagem se casos específicos não tivessem presentes no conjunto de dados de treino. Foi também possível verificar que, no módulo SRE havia um número mais baixo de combinações de resposta, mas um número maior de combinações de recomendações possíveis e presentes no conjunto de dados comparativamente ao SBPG.

Adicionalmente os conjuntos de dados gerados para cada módulo, inclusive a sua versão aumentada artificialmente, foram analisados em termo de balanceamento. Foi possível verificar que o conjunto de dados original tinha um balanceamento bastante baixo, contudo ao fazer o aumento artificial de dados, ficou próximo de um balanço bastante positivo, apesar de ter aumentado em grande escala o seu tamanho, levando assim a uma velocidade de treino mais baixa. De maneira a complementar a informação anterior, foi também testada a correlação entre *labels*, verificando-se que estas, para ambos os módulos, salvo poucas exceções, apresentavam uma taxa de correlação bastante baixa.

Para finalizar, foi revista a distribuição dos conjuntos de recomendações, ou seja, os dados de saída, pelos conjuntos de dados gerados para ambos os módulos, onde foi possível concluir existir uma grande discrepância, sendo que alguns conjuntos de recomendações são gerados para uma grande variedade de combinações de resposta, enquanto outras apenas em uma. Isto poderia levar a problemas a testar os diversos modelos com um baixo e mínimo número de instâncias de treino.

### Análise de Resultados

De maneira a avaliar os modelos selecionados no estado de arte, foram efetuados testes para as três experiências anteriormente referidas, cada uma com condições e focos diferentes. Para a experiência a) foi usada *accuracy* como uma métrica, mas o principal foco foi o tempo de treino, pois o espaço de entrada é fechado. Tendo o anterior em consideração, foram feitos teste de *accuracy* e velocidade de treino para ambos o conjunto de dados original e o aumentado artificialmente, usando os modelos CC, BR, LP e *Label Space Partitioning (LSP)*, e os classificadores base *Decision Tree Classifier (DTC)*, *Extra Trees Classifier (ETC)*, *Extra Tree Classifier (ETC2)* e *AdaBoost (AB)*. Os testes para esta experiência levaram à conclusão que para o módulo SRE o melhor modelo a ser usado é o BR em conjunto com o DTC, e para o módulo SBPG o modelo LP também em conjunto com o DTC.

Para a experiência b) foram feitos testes, desta vez focando na métrica *Mean Accuracy*, usando os mesmos modelos e classificadores base à exceção do classificador AB devido aos

baixos resultados na experiência anterior. Como um dos objetivos deste teste é ser efetuado com um número baixo de dados de treino, e verificar quão eficiente seria quando usado para avaliar todas as opções, o conjunto de treino foram 2000 instâncias aleatórias para o módulo SRE e 4000 para o módulo SBPG, sendo usadas todas as possíveis combinações de resposta como o conjunto de teste. Foi permitido concluir que, com um número baixo de instâncias de treino, foi possível chegar a resultados positivos, em que no módulo SRE foi atingido 89% e no módulo SBPG acima de 85% de *Mean Accuracy* com apenas 400 instâncias de treino. Foi concluído que o melhor classificador base foi o DTC, e os modelos com melhores resultados em ambos os módulos foram o BR e LSP.

Em último, para a experiência c), foram efetuados testes, mas usando as métricas *Precision*, *Recall* e *F1-score*, e feita uma análise por *label* ao invés da análise geral. Para efetuar este teste foram usadas apenas 10 instâncias de treino, provenientes de um conjunto de dados especializado, e usadas todas as possibilidades de resposta como teste. Adicionalmente, devido à sua eficiência nas situações anteriores, foi apenas usado o DTC como classificador base. Os resultados obtidos foram positivos e bastante similares para todos os modelos, havendo assim 13 de 16 *labels* no módulo SRE e oito de 12 *labels* no módulo SBPG em que foi verificado um *F1-score* superior a 80%. Foi também possível verificar que usar o conjunto de dados especializado traz vantagens quando comparado aos resultados obtidos com o conjunto de dados aleatório.

### Conclusões e Trabalho Futuro

Foi concluído que a implementação de aprendizagem automática, mais especificamente de *Classificação multilabel*, em dois módulos da plataforma SAM foi bem sucedida e positiva. Foi possível atingir maior parte dos objetivos propostos, à exceção do último, pois só foi implementada em dois módulos ao invés de todos os que estavam disponíveis, por ser necessário dedicar mais tempo do que esperado à preparação dos cenários e aparato de teste. Contudo, foi demonstrado que o processo ETL criado é adaptável aos restantes módulos que sigam a mesma lógica e que os modelos são usáveis para dois conjuntos de dados com tamanhos de *input* e *output* bastante diferentes. Foi observada também a diferença de eficiência entre as três situações hipotéticas diferentes, verificando vantagens e desvantagens em cada uma.

Em termos de trabalho futuro, recomenda-se aplicar aprendizagem automática em mais módulos usando os métodos desenvolvidos, uma maior integração do processo ETL na Application Programming Interface (API) da SAM, alterações significativas aos conjuntos de dados gerados tendo em base a informação inferida através do processo criado e testes a mais modelos e classificadores base seguindo a evolução da tecnologia.

# Abstract

Nowadays, companies are increasingly deploying more Internet of Things (IoT) devices into the market without considering the security requirements of these systems. Platforms like the SECURIoTESIGN framework attempt to minimize the number of devices that are released with these vulnerabilities, by informing and guiding interested developers about possible secure implementations without needing to contact a security expert (even though it does not replace his knowledge).

The modules use questionnaires, whose answers generate a set of recommendations, obtained using heuristics or fixed rules. To facilitate the development of this modules and the provided recommendations, as well as minimizing the need of a security expert, embedding of Machine Learning (ML) was proposed and pursued, being applied to two modules of the platform. In this dissertation the research needed to implement ML in this context is explored and explained, along with the implementation details on both modules, including the creation of a dataset containing all possible answer combinations, automatically. Furthermore, an analysis of the generated dataset was made, how to artificially augment it, and its usage examined, using different variations of available data, for training and testing various multi-class and multi-output models, therefore allowing to simulate situations where resources could not be obtained or an expert was not available.

It was possible to conclude that the usage of multi-class and multi-output algorithms presented positive results, when performed with different variations of training data, allowing to conclude that implementing ML in this context may bring advantages to the platform.

# Keywords

Cyber-Security, Internet of Things, Machine Learning, Multi-Class Classification, Multi-Label Classification, Multi-Output Algorithm, Security Framework, Security-by-design

# Contents

# List of Figures

# List of Tables

# Acronyms

**AB** AdaBoost

**ACISM** Assessment of the Correct Integration of Security Mechanisms

**ACM** Association for Computing Machinery

**API** Application Programming Interface

**ARFF** Attribute-Relation File Format

**BR** Binary Relevance

**CC** Classifier Chains

**CCS** Computing Classification System

**CLR** Calibrated Label Ranking

**CPU** Central Processing Unit

**CSV** Comma-separated Values

**CWE** Common Weakness Enumeration

**DTC** Decision Tree Classifier

**EMCL** Ensemble Multi-label Classification

**ETC** Extra Trees Classifier

**ETC2** Extra Tree Classifier

**ETL** Extract, Transform, Load

**GBC** Gradient Boost Classifier

**GPU** Graphics Processing Unit

**HOMER** Hierarchy Of Multi-label classifiERs

**IoT** Internet of Things

**KNC** KNeighbours Classifier

**LP** Label Powerset

**LR** Logistic Regression

**LSP** Label Space Partitioning

**LWCAR** Lightweight Criptographic Algorithms Recommendation

**ML** Machine Learning

**MLC** Multi-label Classification

**PEP** Python Enhancement Proposal

**RaKEL** *RAndom k-labELsets*

**RAM** Random Access Memory

**RFC** Random Forest Classifier

| | |
|---|---|
| **RNC** | Radius Neighbour Classifier |
| **SAM** | Security Advisory Modules |
| **SBPG** | Security Best Practice Guidelines |
| **SGD** | Stochastic Gradient Descent Classifier |
| **SQL** | Structured Query Language |
| **SMOTE** | Synthetic Minority Over-sampling Technique |
| **SRE** | Security Requirements Elicitation |
| **SVM** | Support Vector Machines |
| **TMS** | Threat Modelling Solution |
| **XSS** | Cross-site Scripting |

# Chapter 1

# Introduction

## 1.1 Motivation and Scope

Security is one of the most important aspects in the day to day digital life, worked on for many years and being challenged every day with new types of attacks, devices, and countermeasures. In addition to the daily developments on cyberattacks, quotidian objects are also starting to be integrated with the Internet, leading to the creation of the Internet of Things (IoT) paradigm. The Internet integration, present in these mundane objects, leads to an abundant number of new quality of life features, which can rely on the collection of the user private data, in order to be more efficient or appealing. Keeping data safe, particularly private information, is getting more important by the minute, since the amount of data collected has increased exponentially over the years [Ryd18], and consequently, the necessity to protect that same data has also increased, both from a consumer and developer viewpoint. *"Privacy Not Included buyer's guide"* [Fou22b], a tool developed by Mozilla, helps a user shop by informing how trustworthy the item is, but does not remove the necessity of a safer development in order to protect private data. Due to this necessity, it is important, when developing an IoT application or object, to take security into account, starting on the earlier stages. Data leaks not only bring bad reputation to the company at hand and increasing fears about the IoT paradigm, but can also bring dangers to the user, which might be avoided if security is considered as early as possible, since most flaws come from a poor design phase [JDSTN19].

Security has become one of the most important aspects in the IoT sphere, from both the company and user viewpoint. However, since other costs usually take more importance when developing a system or object (energy costs, size, marketing), the allocation of resources to research and develop secure methods can become a secondary or absent subject. Taking the previous into account, having a system that can reduce those costs, and also reliably tell a developer what should and can be used to improve overall security, can alter the design phase and further phases to take those changes into account, increasing user safety and privacy protection.

Making a system that can reliably answer the doubts about what security related technologies, techniques, measures, tests, and other cautions an IoT application should implement, requires that same system to be scalable and flexible, since IoT is used in many fields, with their own particularities, such as Agriculture, Healthcare, Commerce, and Home appliances and systems [RPMT18, SEC21]. To accomplish those requirements (adaptability along the time and the ability to learn), Machine Learning (ML) can be used, since it can evolve and adapt with additional data, enabling a faster adjustment to when new type of system arrives,

and can give a quick and reliable answer.

Taking the above into account, and under the 2012 version of the Association for Computing Machinery (ACM) Computing Classification System (CCS), the scope of this dissertation can be defined by the categories named:

- **Computing methodologies – ML**;
- Security and privacy – Systems Security;
- Security and privacy – Human and societal aspects of security and privacy.

## 1.2 Objectives

The work behind this dissertation intends to improve the overall quality of the Security Advisory Modules (SAM) platform (see section 2.1), by making it more scalable, adaptable and efficient. The SAMs are the most concrete results of the SECURIoTESIGN project [IFS+21], and can be thought as black boxes that take guided answers as inputs and provide security related recommendations or technical documentation as outputs. The team working with the SECURIoTESIGN framework may not always have direct access to a security expert, but the SAM platform will still need to provide accurate answers in order to be considered a reliable tool. To cope with this requirement, ML, more specifically multi-class and multi-output models, will be implemented in several modules of the platform. This implementation aims to provide a tool to automatically update the output given to the user, with and without the presence of a security expert. To guarantee the reliability of the tool, several models shall be executed and tested, comparing the current trusted output with the one provided by the new implementation. Employing this tool may grant a safer addition of new questions or other, permitting to explore if it can provide a reliable expansion of the amount of information that SAM can present.

As of the writing of this project, ML implementation is partially present in the Security Requirements Elicitation (SRE) module [LCS+21] of the SAM platform (see section 2.4.1). Taking the previous points into account, the objectives and phases of the project described herein be structured as follows:

1. The **first step** of this project will be the analysis of the previously mentioned implementation, and adapt it to the increased size of the dataset;
2. The **second step** of this project will be the automatization of the creation of the dataset, which could also be used for the remaining modules that use the same logic, as well as infer relevant information about each module automatically;
3. The **third step** is to take the new dataset and analyse both the dataset and the questionnaire, retrieving relevant information, and building a basis to expand to the remaining modules;

4. The **fourth step** is to perform tests, using a high, low and minimal amount of data, as well as fine-tuning the models for each quantity. Furthermore, it shall be analysed which algorithms work best on a small scale or large scale *dataset*, granting possible solutions for every module of the SAM platform;

5. The **fifth and final step** consists on the expansion of the ML aspect to the remaining SAM modules, taking into account the nuances of each module in terms of data and output.

The steps described above comprise the overall ambition of this project. However, further tasks could be executed to further improve the SECURIoTESIGN project. Some of those tasks could be, for example, an interface that would allow for simpler questions addition and removal, allowing to also adapt their weight on the final answer, additionally allowing for an automatic update of the *dataset* and retraining of the model. Furthermore, the same interface could display the results of multiple models, including the accuracy and time spent, and choose which model should be used for each module, being intended for a standard developer to be able to easily analyse the results. This tool could allow for an even faster development cycle of the actual and future modules of the SECURIoTESIGN project.

## 1.3 Document Organization

The present document is structured as follows:

- Chapter 1 – **Introduction** – presents the project scope and motivation, including problems found in literature, the project main objectives and envisioned phases, and the organization of the dissertation;

- Chapter 2 – **Background and Related Work** – presents a summary and analysis of the SECURIoTESIGN project, followed by the state of Multi-label Classification (MLC), and previous work on this project scope;

- Chapter 3 – **Implementation Details** – presents a summary of the implementation details, as well as the different hypothetical *Experiments* created, the hardware and technologies used, the Extract, Transform, Load (ETL) process description and how data augmentation and stratification was implemented;

- Chapter 4 – **Analysis of the SRE and SBPG Modules** – presents a summary of all the module questionnaires in which ML was implemented, including information about the generated dataset as well as its balance and further relevant information;

- Chapter 5 – **Analysis of the Results** – details the results of the tests performed for every Environment, including the use of a high, low and minimal amount of training data, with various models and base classifiers combinations;

- Chapter 6 – **Conclusions and Future Work** – details the main conclusions drawn from this dissertation project and discusses future related work.

# Chapter 2

# Background and Related Work

## 2.1 Introduction

This chapter presents an in depth look at the various paths that can be followed in order to achieve the goals detailed in chapter 1. It starts by giving a summary of the SECURIoTESIGN project, including the SAM platform (section 2.2). Next, there is an analysis of multiple multi-class and multi-output algorithms that are suited for the project needs, as well as a comparison between them, focusing on the advantages and disadvantages of each one (section 2.3). Furthermore, a study of the previous work done to solve the issue at hand and its results is provided (section 2.4). The chapter concludes with a summary of the information aggregated in all sections and its importance to the project at hand (section 2.5).

## 2.2 SECURIoTESIGN

SECURIoTESIGN is a project whose objective is to attempt to minimize the problem of IoT security, focusing on its main aspects. The objective is, for a company or individual that intends to develop a product or other for the IoT, to easily discover and know the right security controls and mechanisms to use, and how to correctly and effectively embed them during the design or development phases. Companies and individuals with access to the aforementioned information should be able to have the knowledge on how to mitigate weaknesses and reduce the possibility of threat actors, as well as supplying an adequate and advantageous level of protection against security attacks on their solution. Using the previously discussed tools, SECURIoTESIGN attempts to curtail the gap of security information available for IoT, compared to other areas of Information and Communication Technology.

In order to achieve the objectives described above, the SECURIoTESIGN team divided its work plan into five main activities, as shown in table 2.1.

Table 2.1: Summary of the SECURIoTESIGN activities and their respective leaders as of June 24, 2022.

| Activity | Leader |
|---|---|
| State of the Art on IoT and Security | Mário M. Freire |
| Security Engineering for the IoT | Bernardo Sequeiros |
| Mapping of Security Requirements and Technology | Musa Samaila |
| Testing and Auditing of IoT Systems | Multiple |
| Framework of Tools | Tiago M. C. Simões |
| Project Management, Dissemination and Exploitation | Pedro R. M. Inácio |

According to the official sources [IFS+21], the **first activity "State of the Art on IoT and Security"** covers the technological surveillance aspects and analysis of the specialised literature, laying the groundwork for the **second, third and fourth activities**, which are focused on specific research lines, those being security engineering for IoT, investigating IoT software and systems, and planning the recommendation of security requirements and technology. **The fifth activity** is dedicated in its entirety to prototyping a framework of tools resultant of the aggregation of the investigation work and obtained knowledge. These tools will also be applied on the second, third, and fourth task and this application is where the **current project is included**. Finally, the **sixth and final activity** is dedicated to the management, dissemination and exploitation of the project. Development of **SECURIoTESIGN** ended in january 2022 and received an excellent rating.

### 2.2.1 **SECURIoTESIGN** Security Advisory Modules

SAM is the main framework of the **SECURIoTESIGN** platform, whose purpose is to provide information to a user developing an application, about a myriad of security related topics, with the end goal being to improve the robustness and security of the overall applications in the IoT sphere.

SAM functions as an automatic security expert, where the main intent is for the end user to feel like he is consulting with a human, simulating the interaction through questionnaires. When filled, these questionnaires will provide a detailed report containing the recommendations that better suit the developer, depending on the chosen module. This framework is divided into five main groups, as can be observed in figure 2.1.

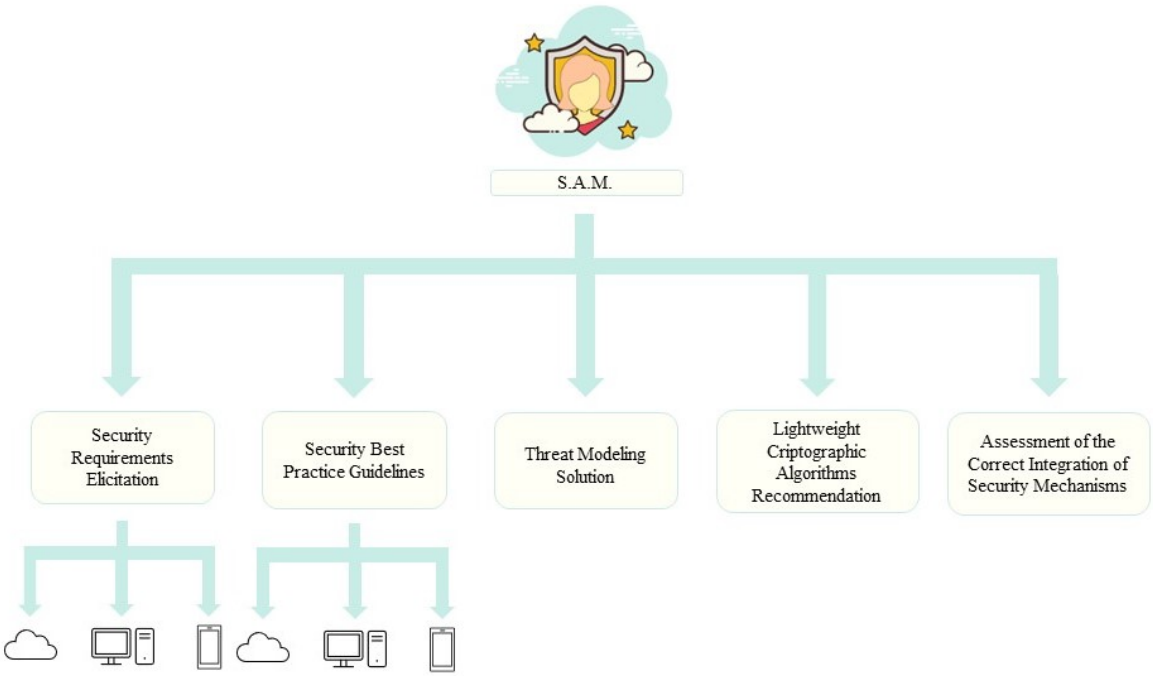


Figure 2.1: Summary of the **SECURIoTESIGN** SAM architecture.

The five main tools, SRE, Security Best Practice Guidelines (SBPG), Threat Modelling Solution (TMS), Lightweight Criptographic Algorithms Recommendation (LWCAR) and Assessment of the Correct Integration of Security Mechanisms (ACISM), can be summarized in the following manner:

- **Security Requirements Elicitation** – This module is focused on providing information regarding the security related requirements, when implementing an application on Cloud, mobile and IoT ecosystems [SCS+20, SLA+20]. As it is the norm in the modules of the SAM platform, it provides a questionnaire [SLA+21], which the user developing an application in the aforementioned conditions will answer. Using those answers the tool will attempt to provide the best and most accurate security requirements. This can include confidentiality, privacy, integrity, among others. This module also has two main parts, one that maps the requirements for the applications in the IoT sphere, and the other one maps the requirements for the cloud and mobile counterparts;

- **Security Best Practice Guidelines** – This module is focused on providing information regarding good practices when implementing an application on the Cloud, mobile and IoT ecosystems [SCS+20, SLA+20]. As it is the norm in the modules of the SAM platform, it provides a questionnaire [SLA+21], which the user developing an application in the aforementioned conditions will answer, while focusing on common potential vulnerabilities that need to be taken into account during the development phase. This module, just like SRE, also has two main parts, one that maps the requirements for the applications in the IoT sphere, and the other one maps the requirements for the cloud and mobile counterparts.

  Using those answers, the tool will attempt to provide the best and most accurate security practices for said application, recommending procedures to avoid threats, e.g., Structured Query Language (SQL) injection, usually also including advantages, disadvantages and areas of caution;

- **Threat Modelling Solution** – Represents the need to modulate the possible threats to the system, what types of attacks can be brought depending on the system specifications. As is the norm in the SAM modules, this is achieved through a questionnaire filled by the user who intends to develop an application on the IoT sphere. This module will present common weaknesses found in software and hardware components, during design and development phases, and provide the respective Common Weakness Enumeration (CWE) identifiers and their descriptions;

- **Lightweight Criptographic Algorithms Recommendation** – This module is focused on providing information on cryptographic algorithms, being divided in three main sections, if it is hardware, existing software, or planned software. This module considers if the application or device is software or hardware based, and according to its processing power, suggests cryptographic algorithms that best suit the security requirements provided by the SRE module;

- **Assessment of the Correct Integration of Security Mechanisms** – This module portrays the correct integration of security mechanisms, by providing tests recommendations depending on malicious intent, for example, if a user wants to test its application and verify if its inputs are well and securely validated, he should test for attacks like Reflected Cross-site Scripting (XSS) and Stored XSS. The objective is to provide the developer with the basic information for how to prevent some types of attack, basic requirements and associated risk levels, so as to better identify what needs to prioritise.

The order of the modules also represents the order of development of a new application, starting with a requirements survey, the generation of best practices guidelines, threat modelling to prepare for potential attacks and, when taking into account the previous assumed threats, a conjunction of algorithms that should be used to protect the application, finishing with an assessment of the implementation and what tests should be done to guarantee the application safety.

The questionnaires developed for each module are the result of an extensive analysis of the actual paradigm of IoT [SNF+18, SSC+17, SSFI18, SJS+19, SSS+20, dMRIMF21, Cos21], and the SAM platform is being continuously improved and analysed [Lop21].

To provide the reader with a concrete idea of how a module work, the answers to a series of questions of the SBPG module are shown below, followed by the generated outputs (recommendations):

- **Questions and Answers**:
  - What is the architecture of the system ? Application Programming Interface (API) Service;
  - Does the system use a database ? No;
  - Which type of authentication will be implemented ? No Authentication;
  - Will there be user registration ? No;
  - Which programming languages will be used ? Java;
  - Will the system allow input forms ? No;
  - Will the system have logging ? Yes.
- **Recommendations**:
  - API;
  - Authentication;
  - Cryptography;
  - Logging.

### 2.2.2 Database Information

In this subsection, it is analysed how the relevant parts of the SAM database, required for the project described in this dissertation, are stored, and therefore, it is focused on the information that can be obtained and transformed into a future dataset to be used in this context, as well as possible needed improvements and adaptations. It is important to note that SAM uses MariaDB [Fou22a].

The first data important for the project is information about the questions, how they are connected to the module and to each other, since there are questions that require others to be answered a certain way in order to appear (see section 2.4.1.1). The second important data is the answers to the previously mentioned questions, if the connection between the answers and the recommendations given by the system are stored in the database, or if it is all given by logic only present in the source code of the backend.

Data about the questions is readily available in the database, the information related to the content of the questions, which module it is associated to, and the order of appearance is all present, as well as information about which questions have certain answers needed to their appearance in the questionnaire. In contrast, the data related to the answers lacks in certain aspects, while it is stored which answers are to be presented to the user for each question, there is no association made between the answer given and the recommendation output, for all but one module, being the module with complete information SBPG, while the remaining modules like SRE require the logic present in the backend to present the user with the recommendations. However, the database is prepared for that information to be stored. In addition, there is no direct relation between each module and its respective recommendations output.

## 2.3 State of Multi-label Classification

This section serves as a summary of what MLC is, what types of MLC classifiers exist, as well as providing reasons why it should be used in the project at hands.

### 2.3.1 Definition

MLC comes as a way to solve the **classification problem**, and is a generalisation of the multi-class classification, while **multi-class** classification allows for categorising each instance to **precisely one** of multiple classes, MLC allows to **categorise to more than one** of those multiple classes.

Given the following parameters:

1. An input space $\mathcal{X} = \mathsf{X}_1 \times ... \times \mathsf{X}_d$, where $d$ is the dimension;
2. An output space $\mathcal{Y} = \{\lambda_1, ..., \lambda_l\}, l > 1$, where $l$ are the labels,

MLC [MGCV18] attempts to create a **predictive model** whose objective would be, for each set of labels, to designate them as **relevant** or **irrelevant**. It is important to note that, if we consider $Y \subseteq \mathcal{Y}$ and $\overline{Y} \subseteq \mathcal{Y}$ as our relevant and irrelevant labels respectively, that $(Y, \overline{Y})$ is a bipartition of $\mathcal{Y}$ .

Three groups [MKGD12] can be formed when describing MLC algorithms, those being **problem transformation**, **algorithm adaptation** and **Ensemble Multi-label Classification (EMCL)**. These groups of algorithms will be explained in more detail in the next subsections. A summary of that split can be seen in figure 2.2.

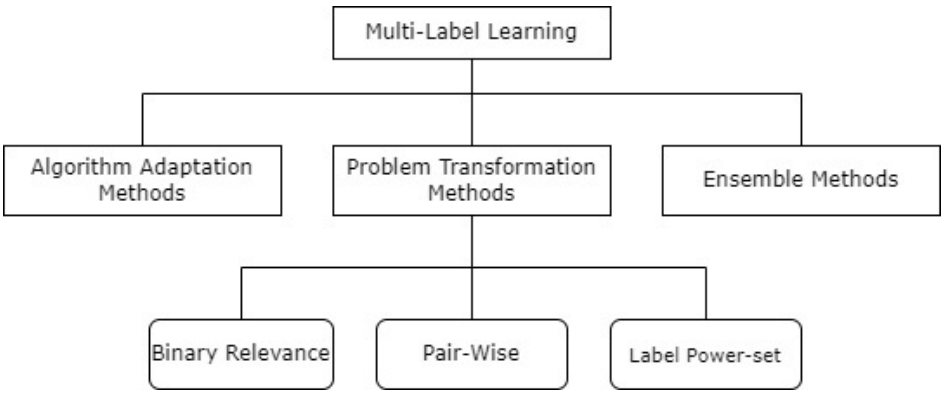


Figure 2.2: Multi-label classifiers divided into groups, adapted from [MKGD12].

MLC comes with high importance to this project, since the main objective is to analyse multiple answers (which are the inputs of the algorithms), and return multiple-outputs, which will be the security recommendations, or in other words, the appropriate labels.

### 2.3.2 Problem Transformation

**Multi-label** learning problems can be transformed in order to take advantage of the already available high number of ML algorithms. This is the basis behind problem transformation methods, and these will perform that transformation by converting them into **regression** or **single-label** classification problems. These types of methods are important for the project described in this dissertation since they were already partially implemented and tested, with success, in previous implementations of MLC in the SAM SRE module [LCS+21]. These methods can be divided into the following concepts:

- **Binary Relevance (BR)** – Focuses on converting multi-label problems into more simple binary classification problems, to accomplish this it uses the one-against-all strategy. We can observe two examples, with BR and Classifier Chains (CC), explained in sections 2.3.2.1 and 2.3.2.2 respectively;
- **Label Powerset (LP)** – This method consists on the transformation of full labelsets into atomic labels, turning the problem into a single-label problem, and allowing to take advantage of the single-label classification methods available, discussed this with more detail in section 2.3.2.3;

- **Pair-wise Methods** – This method consists in covering all pairs of labels, where the first will consist on a positive example and the second on a negative one, followed by the use of a majority voting algorithm. More details on this method in section 2.3.2.4.

#### 2.3.2.1 Binary Relevance

BR adopts a one-against-all approach, starting with the creation of a classifier for each label, where, for each example, if it contains the label it is considered as positive, otherwise, if it does not contain it, it will be considered as negative. Each classifier will then predict the relevance of its associated label for the example, creating a collection of the most relevant. A ranking executed by the corresponding classifier will be used in the end as the way to determine the probability of each label. The following elements of BR [ZLLG17] are noteworthy:

- The complexity increases with the number of available labels;
- It is not restricted to particular learning techniques, so multiple and diverse binary learning algorithms can be used;
- Adaptable to learn from multi-label problems with missing labels.

This model can be suitable for the datasets resulting of processing the information of the module, and can help achieve one of the main objectives of applying ML in this context, since it can generalize beyond available combinations. However, this strategy is not usually suited for large number of labels, and the first module SRE has at least 16 requirements, which are our labels in the context, augmenting the complexity of the problem.

#### 2.3.2.2 Classifier Chains

CC [RPHF09], just like in BR (see section 2.3.2.1), involves a number of binary transformation correspondent to the amount of labels. It provides advantages when compared to BR those being that it considers the results from former iterations and classifications, creating a chain of binary classifiers. Each classifier belonging to the chain is, as in BR, responsible for a label, and will predict and learn its association while also taking into account all the previous BR predictions on the same sequence.

The beginning of the chain marks the start of the classification process, travelling along it, and in the process transmitting label **information between classifiers**. This allows for CC to know correlations in the label space, providing an **advantage**, in relation to BR, which ignores that information, giving CC base classifiers more **predictive power**.

It is important to note that CC is very similar in computational complexity when compared to BR, and also that different chain orders will result in accuracy differences [RPHF09].

In similarity to the previous discussed model, BR, CC can help achieve one of the main objectives of this project, which consists on the generalization of missing labels, while also taking

into account label relations, but can also become more complex due to having numerous labels.

#### 2.3.2.3 Label Powerset

LP objective is to combine the labelsets into single labels, making it a single-label problem. With the transformation of the labelset, output possibilities for all distinct label subsets will exist, from the original problem, which means we can take all label correlations into account. However, it also means that we can have a substantially big number of possible outputs, possibly **sparse and imbalanced**. This can also bring problems in the context of this dissertation, since label correlation may be low in the generated datasets.

To minimize the disadvantage, methods like the pruned problem transformation [Rea08] have been developed, this would allow to only take into account the labels that occur more than $n$ times. Hierarchy Of Multi-label classifiERs (HOMER) [TKV08] method also comes into play, which transforms the label subsets into hierarchies, allowing to build a classifier for each node present in the hierarchy.

The usage of LP in this context may bring problems due to its high dependency on having label combinations present in the dataset, which could be problematic when testing this model with low amounts of data, since most labelsets may not be present.

#### 2.3.2.4 Pair-wise

Pair-wise methods, also known as *round robin* with binary classifiers, consists of using multiple classifiers to cover all label pairs. It divides the labels into two groups of pairs, being the first group the positive items and the second the negative ones.

These two groups are then evaluated by all the chosen classifiers, and the corresponding labels ordered by a sum of their votes, ending with the use of an algorithm that will rank the labels in order to tag the relevant ones for each example

Techniques like the Calibrated Label Ranking (CLR) [PF07] have been developed to extend the pair-wise method approach to MLC. CLR introduces a calibration label $\lambda_0$ which can be used to separate the labels in relevant and irrelevant, where $\lambda_0$ is favoured over all the irrelevant labels and all relevant are approved over it.

Another technique developed to extend the pair-wise method approach, for multi-class classification, was the Quick Weighted voting method [PF07], which in itself is an extension to the CLR technique. This voting presents an upgrade compared to majority voting used in CLR, by taking into consideration that some classes, during voting, can be removed from the top rank classes, since they would never surpass the maximum number of votes even if they were to get all the remaining votes. Pairwise classifiers are chosen based on the number of votes a class did not receive, also known as voting lose value, therefore focusing on the classes with the lowest amount of voting loss. To adapt this technique to MLC, the process

is repeated until all relevant labels are specified, and the leftover labels are irrelevant.

CLR can differentiate relevant from irrelevant labels, nevertheless, it suffers from complexity issues when dealing with large number of relevant labels [FHMB08]. In the context of this project, it should be noted that it is more important to have a false positive than a false negative, since it is more advantageous for the security of an application to implement a secure method rather than ignore it, leading to the possibility of many relevant labels, thence this method may not be applicable on this context.

### 2.3.3 Algorithm Adaptation

Algorithm adaptation methods are those consisting of multi-label methods that widen and adapt an existing ML algorithm to execute the task of multi-label learning. In this section, these methods can be observed on the following ML algorithms:

- **Boosting** — boosting is more an algorithm than a model, since it trains a sequence of weak models, allowing to improve or attempt to improve their prediction power, by having each model compensating the weakness of the former. Examples of this application can be seen in the usage of, for example, *AdaBoost* [FD13, SS00];
- ***k-Nearest neighbours*** — an algorithm that assumes that similar things exist in close proximity, being possible to use for both classification and regression predictive problems, having been adapted to work with MLC [ZZ05];
- **Decision trees** — an algorithm that uses a model of decisions resembling a tree structure, commonly used to fabricate a strategy to reach a particular goal, having been adopted to MLC [CK01, BDRR00];
- **Neural networks** — They are algorithms based on the human brain, attempting to mimic how neurons signal one another, having been adapted to work with MLC [CS03, ZZ06];
- **Support Vector Machines (SVM)** — Given a dimensional space based on the number of features, this algorithm attempts to find a hyperplane in that same space that can be used to plainly classify the data. It has been adapted to work with MLC [Pet06].

In previous implementations of MLC in SRE [LCS+21], *AdaBoost*, Decision Trees and *k-Nearest neighbours* were all used as base classifiers for the chosen models, having been proven effective and a possibility for the current project.

### 2.3.4 Ensemble Multi-label Classification

In MLC, EMCL [MGCV18] are the **ensembles that combine several classifiers that handle multi-label data**, so algorithms like BR, even though they combine multiple classifiers into a single one, they are not considered EMCL, since they only handle single-label

data. The following list presents some examples of these ensembles, followed by a brief explanation of each one.

- ***AdaBoost.MH***(AdaBoost with Multi-class Hamming Loss) [SS00] — An extension of AdaBoost [FS97] (which is a well known and well studied algorithm [Bre98, FS96], used in many ML tasks), and being based on BR (see section 2.3.2.1). This method improves on the original *AdaBoost* by also maintaining weights over the labels, which makes it very diverse, since it uses weights for both instances and labels. Training instances and their corresponding labels get increasing weights in the following classifiers, when they are hard to predict, and lower weights in the opposite scenario.

- **Ensemble of LP classifiers** [MGCV18] — The first ensemble based on LP (see section 2.3.2.3), this ensemble uses bagging to generate a variety of classifiers, followed by a combination of the predictions done by the base classifiers by majority voting. One advantage of this classifier is that it is able to predict labelsets that were not originally present in the training set and also, complexity wise, this ensemble is nor more complex when compared to LP. This makes it a possible candidate to be used in the SAM platform since it has better results and does not increase its complexity.

- **Ensemble of Pruned Sets** [RPH08] — This ensemble is the second based on LP, and briefly mentioned in section 2.3.2.3. Denominated as *Pruned Sets* this ensemble attempts to prune the most uncommon labels, and concentrate on labels with bigger relationships, which, in consequence, will reduce the complexity of the algorithm.

  This ensemble workflow starts with the training of $n$ pruned sets models. This occurs over subsets of the original training set, without replacement. A final prediction is obtained by using the results obtained from each classifier and using a voting mechanism.

  This ensemble attempts to stop the overfitting effects of pruning [RPH08] and, in addition, may allow the prediction of labelsets that are absent in the training data.

- ***RAndom k-labELsets* (RaKEL)** [TKV11] — The third and final ensemble based on LP is RaKEL. This ensemble starts by randomly dividing label sets into smaller ones, followed by selecting a $n$ number of *k-labelsets* and allocating each one to be learned by its own LP classifier. Due to the previous steps, each model will be able to provide binary predictions for its allocated *k-labelset*, and it will output the result based on a multi-label prediction in conjunction with a majority voting procedure for each label.

  Some variants exist for this ensemble, for example, *RaKEL++* [RSI13], RaKELd [TKV11] and ACkEL [WKWJ21]. The first takes into account the confidence values of each classifier to generate the final prediction for each label, instead of using bipartitions. The second one generates subsets of $k$ labels that are disjointed, taking notice of each label only once. The third one borrows the idea of active learning to adapt the label-selection criteria, attempting to achieve a better balance.

  This ensemble provides some advantages, starting with, given that the labelset used is smaller, the tasks for each LP classifier are therefore less intensive and more simple,

furthermore, it can obtain a greater balance and predict labelsets that are not present in the original training set.

These ensembles, albeit powerful, may not be cost-efficient to the project in hand. However, considering the possible scalability of the project, should still be explored if there are no time constraints.

## 2.4 Previous Work

This section describes some of the existing work on the subject at hands. It namely describes the first attempt at embedding ML in the SRE module of the SAM platform. This particular effort was described in the paper entitled "*Machine for Security Requirements Elicitation*" [LCS+21].

### 2.4.1 Machine for Security Requirements Elicitation

*Machine for Security Requirements Elicitation* consists of a first attempt to implement ML on the SRE module of the SAM platform (figure 2.1). The idea to develop this improvement came from the need to quickly update the questionnaires and their possible answers, leaving the questionnaire result dependent on the ML aspect, providing more time to develop informative tutorials and increasing research time. Previously, when a new question was to be added, a developer needed to implement it, by adding the question, what its answer could signify, and all the possibilities when in comparison with the remaining answers given. With the embedding of ML, the objective would be for the developer to only need to add the question, the possible answers and correctly label its results, leaving the most time-costly function, the possibilities depending on the remaining answers, to be evaluated by the machine.

With that objective in mind, SECURIoTESIGN developers started by researching the best possible ML algorithms to be implemented [Joa21, LCS+21] concluding that a multi-class with multi-output, more specifically the binary strand of MLC, since one answer could lead to more than one output. With the research done, the **models**, CC, BR, LP, Multi-label k-Nearest Neighbours, Label Space Partitioning (LSP) [SKK16], k-Nearest Neighbours and Majority Voting were chosen. To complement this models the **base classifiers**, AdaBoost (AB) [FS97], Gradient Boost Classifier (GBC) [Fri01], Extra Trees Classifier (ETC) [GEW06], Random Forest Classifier (RFC) [Bre01], Decision Tree Classifier (DTC) [SH77], Extra Tree Classifier (ETC2) [GEW06], Radius Neighbour Classifier (RNC) [Ben75], KNeighbours Classifier (KNC) [DH73], Stochastic Gradient Descent Classifier (SGD) [Bot12] and Logistic Regression (LR) [M.D44] were chosen to work alongside the previous mentioned models. Both the models and classifiers implementations were provided by the *sklearn* [PVG+11] and *scikit-multilearn* [SK17a] libraries.

It is possible to observe, from the above lists, most of the models/classifiers chosen are based on problem transformation (section 2.3.2) and algorithm adaptation (section 2.3.3), due to

the more approachable implementation and synergy with the *dataset* size.

#### 2.4.1.1 Dataset

The dataset, used for the development of the described project, was constructed using both information contained on the SAM database (the questions and possible answers for the SRE module), and logic applied on its backend (the resulting requirements). To create the dataset, the SECURIoTESIGN development team assumed the possible answers given in the module questionnaire as its input data, and the different possible requirements as its output data, while taking into account that certain questions will never appear if certain criteria is not met.

Answers were transformed into a number, depending on the order of appearance, while being independent of other questions answers, for example, if question one has three possible answers ”A”, ”B” and ”C” presented in that order, the value would be 1, 2, and 3 respectively, and if question two also has three possible answers, and they are ”B”,”C” and ”D” they will be transformed into 1,2 and 3 in the same manner. Since some **questions do not appear unless a previous question has a certain answer** (denominated as a **trigger** answer), when that condition is not met, for all questions whose appearance is dependent on the trigger, the values would be -1, signifying that the question was never shown to the user. A visual representation of how the dataset is organized can be observed in table 2.2, where ***q3*** and ***q4*** appearance are dependent on ***q2*** answer, with 2 being the trigger answer.

Table 2.2: Demonstration of the dataset used in *Machine for SRE*.

| q1 | q2 | q3 | q4 | r1 | r2 | r3 |
|---|---|---|---|---|---|---|
| 1 | 1 | -1 | -1 | 0 | 0 | 0 |
| 1 | 2 | 1 | 1 | 0 | 0 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 0 |

These datasets were generated in both Attribute-Relation File Format (ARFF) and Comma-separated Values (CSV) format.

#### 2.4.1.2 Accuracy Tests

As it can be interpreted in the last subsection, there was a list of ten base classifiers, leading to a myriad of different possibilities. Therefore, the SECURIoTESIGN developers decided to prune the number of possibilities by executing accuracy tests on all base classifiers with some chosen models. These tests would allow to, not only have a notion of which base classifiers work best with which models but also, cut the most inefficient ones for the available *dataset*.

Using the *dataset* containing part of the possible answers and respective outputs from the SRE module at the time, with a train to test distribution of 70 094 to 17 458, the accuracy tests were executed, and the results obtained are summarized in table 2.3.

Table 2.3: Accuracy tests summary based on [LCS+21]

| Models/Classifiers | AB | ETC | RFC | DTC | ETC2 | KNC | RNC |
|---|---|---|---|---|---|---|---|
| Classifier Chains | 96.8 | 95.8 | 95.7 | 95.0 | 89.3 | 94.3 | 80.7 |
| Binary Relevance | 96.8 | 95.7 | 96.5 | 95.2 | 91.7 | 94.2 | 80.7 |
| Label Powerset | 96.8 | 96.8 | 96.8 | 95.0 | 91.0 | 94.0 | 80.7 |
| Label Space Partitioning | 96.8 | 95.7 | 96.7 | 95.3 | 90.8 | 94.2 | 80.7 |
| Majority Voting | 96.8 | 96.2 | 96.5 | 95.5 | 92.2 | 94.2 | 80.7 |

As it can be noted by analysing the aforementioned table, the results for GBC, SGD and LR were not present as they could not be obtained at the time of that study. In addition, the Multi-label k-Nearest Neighbour and BR k-Nearest Neighbour did not need a base classifier, therefore, they were only tested once, giving 90.4 and 93.1 accuracy respectively. As for the remaining results, it can be observed that the **most constant classifier in terms of accuracy** is AB, with ETC and RFC also showing positive results across the board, in contrast, RNC showed the worst results.

These tests revealed that the accuracy, even in the lowest performer classifiers, is still high for the binary based classifiers, with DTC being the best bet to explore.

To further test the implementation of MLC in the SRE module, members of the **SECURIoTESIGN** team performed tests using *Mean Accuracy* [LZC+16], also known as balanced accuracy, as the metric. The results were as presented in table 2.4. Note that the worst suited classifiers from the previous implementation were not tested.

Table 2.4: Base Classifiers Tests, average for each label, using mean based accuracy, based on [LCS+21]

| Models/Classifiers | DTC | ETC2 | SGD | LR |
|---|---|---|---|---|
| Classifier Chains | 100 | 94.3 | 98.0 | 97.8 |
| Binary Relevance | 100 | 97.9 | 97.6 | 97.9 |
| Label Powerset | 100 | 92.0 | 57.0 | 90.5 |
| Label Space Partitioning | 100 | 97.0 | 97.6 | 97.9 |
| Majority Voting | 100 | 96.1 | 97.2 | 97.9 |

As can be observed from the aforementioned table, DTC was the most efficient base classifier and was used as the basis for the rest of the tests performed by the **SECURIoTESIGN** developers.

The final test was done using a low amount of training data, using always the same train dataset, until a maximum of 1000 units. All models present in the previous table using only DTC as the base classifier were tested, with the conclusion that with 800 train units it was possible to achieve 100% mean based accuracy with every model, and needing only 400 units to extrapolate combinations not present in the training dataset.

#### 2.4.1.3 Models Train Speed Test

Speed is an important aspect when training the models, even though a new question addition will not happen frequently. In the current context speed can be the difference between, advising a developer about the most updated piece of information, and giving him outdated content. Taking the previous into account, the SECURIoTESIGN developers decided to, in addition to the base classifiers accuracy and *Mean Accuracy* tests, the models should also be tested in relation to their training speed, using the DTC model. Results can be observed in table 2.5.

Table 2.5: Model speed tests (in seconds) for DTC (according to [Joa21]).

| Models | Time DTC (s) |
|---|---|
| Classifier Chains | 16.445 |
| Binary Relevance | 14.863 |
| Label Powerset | 21.614 |
| Label Space Partitioning | 21.242 |
| Majority Voting | 29.295 |
| Multi-label k-Nearest Neighbours | 847.902 |
| BR k-Nearest Neighbours | 227.357 |

The aforementioned table shows that both Multi-label k-Nearest Neighbour and Binary Relevance Neighbour show a disadvantage speed wise, compared to the remaining models, which show positive results across the board. These tests results might change if the *dataset* increases dramatically in size, justifying testing them with artificial additional data, **being one of the objectives included in this dissertation**.

### 2.4.2 Conclusions and Possible Improvements

As it can be concluded from the analysis of the previous subsections related to the Machine for Security Requirements Elicitation (SRE), there is already work done when it comes to the implementation of the ML aspect in SECURIoTESIGN . However, this work needs to be expanded and improved upon, starting with testing more base classifiers when testing with low amount of data. Furthermore, the remaining base classifiers, other than DTC, should be tested for speed, allowing for further comparisons, furthermore expanding upon on what was described in section 2.4.1.3.

Completing the above tasks would allow for a more detailed analysis of both the base classifiers and models, allowing for a choice to be made. Nevertheless, these choices could depend on the dataset size, which can vary depending on the SAM module where the implementation is occurring, as well as its balance, to contradict this issue, a way to automatically test the models and classifiers with artificial data (augmenting the dataset size) should be provided.

The aforementioned Machine for SRE therefore provides a good and solid base to expand upon and should be taken fully into consideration while developing the current project.

## 2.5 Conclusions

In this chapter, it was possible to do a deep dive on three main factors that are essential for this project: (i) the first being how the platform we plan to improve and expand upon is organized; (ii) the main objectives it attempts to accomplish; and (iii) its architecture. Following this analysis it was possible to do research on the current state of the technology the project will need to implement, more specifically, the current state of MLC. Finishing this chapter, a summary of the work already developed, in the same context as the project at hands, was described allowing for a better view of what is already implemented and what needs to be implemented further.

Table 2.6: Summary of the chosen models/algorithms and the reasoning behind their implementation priority.

| **Models/Algorithms** | **Reason** |
|---|---|
| **First Priority** | |
| Binary Relevance<br>Classifier Chains<br>Label Power-Set<br>Label Space Partitioning | Were used in previous implementations and proven effective, should be retested with the updated dataset to see if they adapted well to the changes |
| **Second Priority** | |
| k-nearest Neighbours<br>*RAndom k-labELsets*<br>Support Vector Machines<br>Neural Networks | These models have implementations in either the scikit-learn and scikit-multilearn libraries and were considered as the backup in case the first priority models prove unnefective |
| **Third Priority** | |
| Adaboost.MH<br>Ensemble of Label Power-Set<br>Ensemble of Pruned Sets | These models are harder to implement and were therefore considered the last possible ones to test |

Taking the previous into account, joining all the information collected and described in this chapter, a conclusion can be reached in what can be and what is already implemented, and how to further improve the SAM platform. To better summarize the possible implementations, grasp if they are suitable for the project at hands, and if they have already been implemented and tested, we can observe table 2.6. As it can be observed from the aforementioned table, the top priority in implementation and testing are, mostly, the ones related to problem transformation, due to the scope of the *dataset* provided by SECURIoTESIGN and since they were already proven effective, as described in section 2.4. Following the algorithms based on problem transformation, comes the ones based in algorithm adaptation, since most of them have a library implementation but are excessive in resource consumption for the *dataset* size. Finally, the algorithms based on ensembles, which not only do not have an easy to implement library, but also are excessive for this project. The tested base classifiers will also take into account what was discussed in section 2.4 where more speed and accuracy tests will be done, taking advantage of what was explored in the previous section. With this research, it was then possible to build a well established base for the project described in this dissertation.

# Chapter 3

# Implementation Details

## 3.1 Introduction

This chapter presents general details of the implementation done to achieve the objectives described in chapter 1. It starts by explaining the hardware and software used, such as details about libraries and the versions used (section 3.2). Secondly, the different possible *Experiments* are detailed, which includes the high data (execution time), low data (low data reliability) and minimal but specialized data (per label analysis) (section 3.3). Additionally, the ETL process and subsequent details are explained, which includes how information contained in the SAM database was extracted into a usable dataset, and details that are relevant to the results, transversal for all SAM modules, that were adapted, in the project described in this dissertation (section 3.4). Finally, data augmentation and stratification, and how it can be used for the dataset balancing, for the project described in this dissertation is examined (section 3.5).

## 3.2 Equipment and Technologies

Achieving the results included in this dissertation required usage of an assortment of suited technologies, libraries and hardware. The disclosure of this equipment and technologies is important to put into perspective test results, for example results related to the time taken training a certain model, as well as limitations regarding, for example, available memory.

Taking the previous into account, details about Random Access Memory (RAM), Central Processing Unit (CPU), Graphics Processing Unit (GPU) and storage components can be observed in table 3.1.

Table 3.1: Technical specification of the computer in which implementation and tests were performed.

| Component | Specification |
|---|---|
| RAM | 32 GB DDR4 2800MHz |
| GPU | Nvidia RTX 2070 Super |
| Storage | SSD Kingston A2000 500GB M.2 NVMe 2280 |
| CPU | AMD Ryzen 5 3600x 6-Core Processor, 3801 MHz |

The programming language used in the scope of the project described in this dissertation was *python*, chosen due to the previous ML implementation (see section 2.4.1), the available libraries that could be used to implement the ML models and classifiers, and due to python

flask [Pal22] framework being used in the SAM backend, allowing for a smoother implementation of the model in the real case scenario. To guarantee further usage, of the code created during this project, in the development or adaptation of future use cases in the **SECURIoTESIGN** platform, it is important to disclose the main libraries used for the ML adaptation and data analysis. A summary of the most relevant libraries can be observed in table 3.2.

Table 3.2: Summary of the relevant python libraries used in the scope of the project described in this dissertation.

| Library | Version | Reason |
|---|---|---|
| *pandas* [pdt22, WM10] | `1.4.1` | Handling of large files and datasets |
| *scikit-learn* [PVG+11] | `1.0.2` | Classification models used as base classifiers |
| *scikit-multilearn* [SK17a] | `0.2.0` | Classification models for MLC |
| *liaac-arff* | `2.5.0` | Handling of ARFF files used in MLC |
| *imbalanced-learn* [LNA17] | `0.9.0` | Oversampling algorithms to balance the generated datasets |

Additionally, all the developed python code was written with the Python Enhancement Proposal (PEP) [TC22] design philosophies in consideration, providing better readability and ease of adaptation to future developers.

Docker [Mer14] was also used due to the employment of the technology on the SAM API and database.

## 3.3 Experiments Summary

This section details the different experiments, including their purpose and hypothetical environments that they insert in the context of this project. These were created as a mean to prepare the analysis of the different models and classifiers to different availabilities of a team, which can lead to different quantities of training data. These experiments also allow analysing if the tested models and classifiers might work in different projects other than the SAM platform models, as well as provide a notion of how well the machine behaves when learning with the lowest amount of data and with the highest amount of data possible, and how much human input it is needed in order to have a trustworthy system, and if that input needs to be from an expert.

### 3.3.1 *Experiment A* - Execution Time

***Experiment A*** represents testing a model with all available data and see its performance, mainly focusing on the execution time, with both the generated and the augmented data, so it can be seen which model works best when all data is available.

The hypothetical scenario for *Experiment A* is represented by a development team, belonging to a company or other, which are fully specialized and purposefully allocated to the creation of the application. This specialized team would be able to, at any time, add questions and

respective answers, as well as determine how those answers would affect the outcome in terms of output, and store that information on a database. The main objective of this team would be to create questionnaires relative to their specialization and use ML to output the end result labels. Therefore, in this experiment, the following are present:

- A questionnaire based on their objective;
- Funds and a large amount of data storage;
- Access to engineers who could automatize the dataset creation;
- A dataset with all the possibilities.

This *Experiment* simulates the current situation in the SAM platform for the SRE module, where knowledge of what answers should be given to create realistic scenarios is present as well as what output should be given based on those answers, while also having access to developers to automatize the process. Training used a large set of all answer combinations, and a proportional range of answer combinations was used as a test set.

### 3.3.2 *Experiment B* - Low Data Reliability

***Experiment B*** represents training a model with a low amount of data and see how reliable it is when tested against the full range of answer possibilities, mainly focusing on its accuracy, and permitting comparing to the previously done work.

The hypothetical scenario for *Experiment B* is represented by a team, subcontracted by a pharmaceutical company, which were given access to a program where they could create a questionnaire and return labels. Users could then answer the questionnaire and would be given an output, using ML, containing those return labels. However, **the model would start untrained**.

The main objective is to have an online questionnaire that would recommend products based on customers complaints. The pharmaceutical company created a standard questionnaire asking patients for symptoms, and gave a manual containing the logic to apply to the answers, in order to obtain the correct output, to the subcontracted team. The subcontracted team job would be **to answer the questionnaire randomly** and **correct the labels given by the program**. Therefore, in this scenario, the following are present:

- A questionnaire based on their objective;
- An evolving dataset with random experimentations;
- No access to engineers who could create the dataset automatically.

This experiment simulates a situation where knowledge of what answers should be given to create realistic scenarios is not present, and there is no access to developers to automatize

the process, but information about what output should be sprung based on the answers is present. Training used a small partial random set of the all answer combinations, and the full range of answer combinations is used as a test set.

### 3.3.3 *Experiment C* - Per Label Analysis

***Experiment C*** represents training a model with a minimal amount of data, but with realistic scenarios, and test how many labels would be correctly recommended to the end user of the platform.

The hypothetical scenario for *Experiment C* is represented by a specialized person, who had access to the same program and contracted by the same teams as described in ***Experiment B***. This person is more expensive to subcontract, however he is specialized in the subject. The main difference between ***Experiment C*** and ***Experiment B*** is that this person will answer the questionnaires based on realistic scenarios. Therefore, the company who contracted this person are expecting to **contract the expert for less time**, due to monetary requirements, but **obtain better data for the ML model to learn**. Hence, in this scenario, the following are present:

- A questionnaire based on their objective;
- An evolving dataset with specific experimentations;
- No access to engineers who could create the dataset automatically.

This experiment simulates a situation where knowledge of what answers should be given to create realistic scenarios is present, as well as the information about what output should be sprung based on the answers. However, there is no access to developers to automatize the process. Training is done using a minimal set of specialized answer combinations, and is tested using the full range of answer combinations.

## 3.4 Extract, Transform, Load (ETL)

This section details the ETL process. This process was created taking into account the different details of the SAM platform, detailed in section 2.2.1. Furthermore, the development of this process was done with the aim of making the process cost and time efficient, as well as adaptable and scalable to all modules following the same patterns of design. It is to be noted that two modules were chosen to implement ML, those being SRE and SBPG, both having similar structure, and representing the majority of the modules of the SAM platform.

### 3.4.1 Extraction

This section summarizes the extraction and adaptation of the data needed for the development and testing of ML in the SAM platform. Starting with a summary of what information

was necessary based on the needs of the project, followed by an analysis of the database and other elements that could contain that information.

#### 3.4.1.1 Dataset Preparation

The SAM platform uses the same questionnaire philosophy (see section 2.1) throughout most of its modules, the logic used for the extraction of the information used to create the dataset was developed in order to be adaptable to all modules.

One of the main objectives of the dataset creation was to make it dependable only on the database, removing the dependency on the source code of the SAM project, this was proposed to allow a better adaptability for future changes, so when a new question or requirement is added, only the database needs to be updated, the process re-run, and the chosen model retrained. This approach was proposed to shorten the development time needed from the SECURIoTESIGN developers to update the questionnaire changes, and require less time from a security expert.

With the previous in mind, firstly it was analysed how the dataset should be built and what would be their input values and output labels.The dataset input and output values were divided in the following manner:

- **Input** — the attributes for our model will be the answers given in the questionnaires of the SAM platform. Since the questions are multiple question, the value of the input depends on the chosen option, taking a numerical value;
- **Output** — the model output will be a set of recommendations based on the input, which will be simulating the recommendations given after a user answers the questionnaire, these will be our labels, and each unique set will be considered a labelset.

This approach mimics the one made by the SECURIoTESIGN developers during the development of *Machine for Security Requirements Elicitation* [LCS+21] (see section 2.4.1). However, the previous implementation obtained the input information using the database and applied the SAM backend logic to create the corresponding labels. The training and test dataset labels were generated this way due to a lack of information in the database, as well as a structure to know which recommendation corresponded to which answer and what answers removed certain recommendations.

### 3.4.2 Database Analysis

The dataset creation required an analysis of the database and what needed to be changed, while not hindering or creating the need to change elements on the current implementation of the SAM platform. An analysis was performed on which elements of a question, and related answers, were crucial for the creation of the dataset, as well as an analysis of its existence in the current database. The following question and answer attributes were deemed as crucial:

- **Order of the question** — One of the most important elements to obtain the correct output, since some questions appearances were dependent of others and other question answers negated the effects done by previous ones. This attribute was already present in the database in a suitable manner;
- **Number of possible answers** — This would dictate the size of the dataset implementation, since the generated dataset would be all the questionnaire answer possibilities and the corresponding output given. This information was already present in the database in a suitable manner;
- **Recommendation that an answer could generate** — This information was possible to obtain, but only partially, for the second module, SBPG. Due to this implementation, the needed structure was employed, so for each module to be adapted, there was a need to analyse of the corresponding logic and to add the data to the corresponding table;
- **Recommendation that an answer could remove** — This information was not present or planned to be in the database, and required further implementation, this was done by adding the previous element data, but applying the logic "if the user answers with $A$, and $A$ recommends $R1$, $R1$ is recommended. If the user answers with another option, and $R1$ is to be recommended, $R1$ is removed from the recommendations". This element therefore required an alteration to the current database structure, but did not affect the remaining functions of the SAM platform;
- **Dependency on another question and answer** — This element was important to know when to use or present a specific answer or when it should be ignored, to know this information, it was necessary to have data about which question was the parent question and what triggers the appearance of the child question. This information was already present in the database;
- **Multiple answers** — Since the input for the model relies on the transformation of possible answers into a numeric value, knowing if a question would allow for a user to choose multiple answers in a question was crucial to know which labels would be added or removed. This information was already present in the database, but since there are no questions that allowed multiple answers in the SRE module, it was never taken into account when making the *Machine for Security Requirements Elicitation* datasets.

With the above analysis concluded, a summary of what information would be extracted, and the needed database adaptations can be seen summarized, using the same order as the above listing, in table 3.3.

Table 3.3: Summary of the database adaptations needed for the dataset creation.

| Attribute | Adaptations |
|---|---|
| *Question Ordering* | Not needed |
| *Question Answers Details* | Not needed |
| *Generated Recommendations* | Some recommendations needed updating |
| *Removed Recommendations* | New attributes to a table needed to be added |
| *Other Question Dependency* | Not needed |
| *Multiple Answers* | Older data needed updating |

### 3.4.3 Transformation

The second step of the ETL process is the transformation of the data, leading to the cleaning and aggregation of the extracted data from the database. The main objective of the ETL process is to create a suitable dataset while also analysing possible inconsistencies and outliers that can help in the adaptation of the SAM platform to use ML. With the previous taken into account, the chosen dataset structure was the one briefly explained in sections 2.4.1.1 and 3.4.1.1. To achieve this data transformation, some steps were taken, the main ones being summarized in the following listing:

- **Cleaning** — information like if a question contains multiple answers (in the database) were only introduced in later revisions of the SAM platform, so the transformation of those NULL elements to a binary field denoted ”Allows Multiple Answer” was performed. This is only one example of some elements that required this transformation;
- **Deduplication** — due to the nature of the questionnaire, when performing the transformation of the questions answers to the dataset input structure, some answers were changed to illustrate not showing due to not meeting prerequisites, this could lead to duplicate data, which was removed in the final dataset;
- **Summarization** — this was performed as a means to more easily spot inconsistencies in the database, as an example, in the second module, SBPG it was detected that the database logic was outdated, when performing summarization followed by the respective loading of the data to a denormalized table. This information can be useful for other not explored modules that may use the developed code.

Performing this transformation allowed for a standardized dataset creation that also allows for a better analysis of the current state of the chosen SAM module.

### 3.4.4 Load

The third and final part of the ETL process is where data will be loaded to important structures and tables to better infer relevant information. When planning how to structure the

information stored, several principles were taken into account, those being:

- **Transparency to future developers** — it was important that the data should be easily understandable and usable by a future developer;
- **Avoid Repetition** — it was important to avoid duplicate data so that the storage space was not misused;
- **Possible to analyse** — the structures created would allow for a future or current developer to analyse the dataset inconsistencies or if a question answer is not recommending the correct feature.

With the above principles taken into consideration, two main structures, outside the created dataset, were formed, with the first one consisting on a summary of the information of every question of a module, and follows the structure present in table 3.4.

Table 3.4: First structure of information stored related to each question of a module.

| Field | Description |
|---|---|
| *q_id* | The question identifier, also present in the database |
| *multiple_answer* | If the question allows for multiple answers to be selected |
| *is_sub* | If the question appearance depends on the answer of another question |
| *parent* | If it is a sub-question, the identifier of the parent question |
| *trigger* | If it is a sub-question, the answer identifier that makes the question appear |
| *possible_answers* | The identifiers of the possible answers for the question |
| *answers_map* | The mapped values of the possible answers for the question (used in the dataset) |

With this structure it is possible to have a notion of the complexity of the generated dataset, by seeing how many questions will have multiple answers, how many questions will appear less due to being dependent on other answers and how many answers each one has. It is the most basic structure, but still can be used to determine the generated dataset complexity and size.

The second structure is more important, since it allows understanding if the information present in the database is updated and correct, as well if the answers given make sense recommending certain recommendations, this structure objective is to have a table where the content for both the question and each answer is present, as well as what each answer generates as a recommendation. The second structure is represented in table 3.5.

This structure allows verifying if the recommendations given, related to each question and answer combination, make sense, as well as see which questions are irrelevant, and which recommendations are more represented and more possible to appear. The final generated element is the dataset, whose structure was summarized in section 3.4.1.1.

Table 3.5: Second structure of information stored related to each question of a module.

| Field | Description |
|---|---|
| *q_id* | The question identifier, also present in the database |
| *q_content* | The question written content |
| *a_id* | The answer identifier, also present in the database |
| *a_mapped* | The mapped answer identifier, present in the dataset |
| *a_content* | The answer written content |
| *q_order* | The order of appearance of the question (if it is not a sub-question) |
| *sq_order* | The order of appearance of the question (if it is a sub-question) |
| *Recommendation* | A column for each possible recommendation and if it is recommended or not for each answer |

With this ETL process, it was therefore possible to automatize the dataset generation as well as preparing structures and tools to verify the balance and consistency of the data present in the database, making it possible to more easily implement MLC in other current or future modules of the SAM platform.

## 3.5 Balancing using Over Sampling

Dataset balancing is an important aspect of ML, an imbalanced dataset can lead to higher classification costs due to an inclined class distribution, some labelsets may not be very present and improperly generalized, and an orthodox questionnaire answering can lead to a bad prediction if the model does not learn properly from the dataset. Some labelset outliers may be considered unrealistic representations of a developer attempting to gather information for its future application or device, however it is still a possible for a user to answer in that manner.

To attempt to make every possibility relevant, and to increase the possibility of the model to properly learn each labelset, data augmentation was used, as well as data stratification to create a better ratio between the labelsets present in the training and test datasets.

To augment the data, the Synthetic Minority Over-sampling Technique (SMOTE) [CBHK02] was used, SMOTE augments the data by taking the minority class and creating new synthetic samples, making sure that the synthetic data is not completely equal but not too different from the minority class. This augmentation, in the context of the project described in this dissertation, was performed to all classes except for the majority class.

To further increase the balancing of the dataset, data stratification was used when splitting the full dataset between train and test, this would allow for a better class balancing between the train and test datasets and have been shown to have advantages in this types of classification [STV11, SK17b].

## 3.6 Conclusions

In this chapter, it was possible to summarize important aspects of the implementation of ML in the SRE module of the SAM platform, as well as other aspects that will be also used in the remaining compatible modules. By giving a detailed look at the equipment and software used we allowed for future SECURIoTESIGN developers or others that wish to use the code to know what type of equipment was used for these tests and implementations, as well as what libraries and their correspondent versions were used to develop it, leading to less possible constraints in future iterations.

Furthermore, the different experiments taken into consideration during the development of the project depicted in this dissertation were explored, giving a more detailed context for some tests, as well as explaining the advantages of each iteration and when they should be used.

Following the experiments, the functioning of the ETL process was detailed, detailing the treatment that was and can possibly still be needed on the SAM database for current and future models, the mindset behind the dataset structure and the analysis on what data should be extracted, including the most important fields and how they affect the dataset creation. In the transformation side, it was possible to explain some different data processing that was done to guarantee a reliable and clean dataset, using cleaning, deduplication and summarization as the main steps of the transformation. In the load section, it summarized the different ways the ETL process will infer its information, the principles behind the inferred information and an explanation of each structure and its purpose in future analysis of a module.

To end, the benefits of oversampling were explored to deal with imbalanced datasets, and the reasoning behind its application in the project described in this dissertation, as well as analysed how data stratification may bring benefits when used to make the train test split of our dataset.

# Chapter 4

# Analysis of the SRE and SBPG Modules

## 4.1 Introduction

In this chapter, the ML implementation details in the SRE and SBPG modules of the SAM platform are summarized. This summary will start with an analysis of the dataset formed using the ETL process described in section 3.4 with the addition of an explanation of the current state of the questionnaire (section 4.2). Succeeding the state of the questionnaire, it will be detailed the dataset balance, how SMOTE affected the balance and the correlation between the dataset labels (section 4.3). To conclude, the distribution of the labelsets throughout the dataset will be examined, allowing to predict impacts on all experiments to be tested (section 4.4).

## 4.2 Questionnaire Analysis

In this section, it will be summarized how the questionnaire is, at is current state, organized, providing further context for the results shown in the next sections and chapters of this dissertation.

### 4.2.1 Security Requirements Elicitation

The questionnaire for the SRE module contains 17 questions and 16 recommendations, these being representative of our input and output respectively. To facilitate showing the results, a code was attributed to each recommendation, being represented in the following listing:

- **RA1** – Accountability;
- **RA2** – Authentication;
- **RA3** – Authorization;
- **RA4** – Availability;
- **RC1** – Confidentiality;
- **RC2** – Confinement;
- **RD1** – Data Freshness;
- **RD2** – Data Origin;
- **RF1** – Forgery Resistance;
- **RI1** – Integrity;
- **RI2** – Interoperability;
- **RN1** – Non-Repudiation;
- **RP1** – Physical Security;
- **RP2** – Privacy;
- **RR1** – Reliability;
- **RT1** – Tamper Detection.

Various aspects of the questionnaire were analysed to guarantee better relevance for future tests, a small summary of some important aspects of the aspects analysed for this module can be observed in table 4.1.

Table 4.1: Summary of relevant elements in the SRE module.

| Element | Quantity |
|---|---|
| Questions | 17 |
| Recommendations | 16 |
| Answer Combinations | 263 424 |
| Questions that are Sub-questions | 6 |
| Possible Recommendations Combinations | 65 536 |
| Existent Recommendations Combinations in the Dataset | 874 |

As it can be observed from the above table, there were a total of 17 questions where six of them are sub-questions, this meaning that they needed a specific answer in another question to appear in the questionnaire. **The higher the number of sub-questions the shorter the dataset will be**, since some options will be removed due to being duplicates, when doing the transformation explained in section 2.4.1.1. It can also be observed that there were 263 424 **unique answer combinations**, which would be in consequence the size of the full dataset. Furthermore, we can observe that the number of possible recommendations combinations, which were our total **possible labelsets**, was 65 536. However, in the generated dataset, only 874 of that total amount of 65 536 were present, representing the realistic scenario.

To further analyse the questionnaire we can see a summary, in table 4.2, of the questions available in the questionnaire, and the possible recommendations their answers could generate. From the data of the aforementioned table, the following conclusions were reached:

- Most questions had only two answers. Further analysing them was noted that they were `yes` or `no` questions, which could lead to a direct understanding from the machine when doing the training;
- **Authentication** and **Non-Repudiation** recommendations were the most likely to have a positive connotation in the output, since both had seven possible questions whose answers could recommend them. The machine could have difficulty learning what causes the recommendation to be given since it is not direct which question answer recommended it;
- **Data Origin** and **Interoperability** recommendations were solely dependent on one question answer, which had two possible answers, possibly making it easier for the machine to learn, but, if the train test split was not correctly done, a situation where the positive or negative scenarios are not present in the train dataset could occur.

Table 4.2: Summary of the possible recommendations given by each question in the SRE module, where *SQ* represents if it is a sub-question, *A* the number of answers, and the following columns include the code for each recommendation. Based on the questionnaires investigated in [SSS+20, SLA+21].

| **Question** | **SQ** | **A** | **RA1** | **RA2** | **RA3** | **RA4** | **RC1** | **RC2** | **RD1** | **RD2** | **RF1** | **RI1** | **RI2** | **RN1** | **RP1** | **RP2** | **RR1** | **RT1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| State the domain type for your IoT system | - | 7 | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | - | - | ✓ | - | ✓ | ✓ | ✓ | ✓ | ✓ |
| Will the system have a user | - | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Will the system have user login | ✓ | 2 | - | ✓ | - | - | - | - | - | - | - | - | - | ✓ | - | - | - | - |
| Will the system hold any user information | ✓ | 2 | - | - | - | - | ✓ | - | - | - | - | - | - | - | - | ✓ | - | - |
| Will the system store any kind of information | ✓ | 2 | - | - | - | - | ✓ | - | - | - | - | - | - | - | - | ✓ | - | - |
| What will be the level of information stored | ✓ | 3 | - | ✓ | ✓ | - | ✓ | - | - | - | ✓ | - | - | ✓ | ✓ | ✓ | - | - |
| Will this information be sent to an entity | ✓ | 2 | - | ✓ | - | - | - | ✓ | - | - | - | - | - | ✓ | - | - | - | - |
| Will the system be connected to the internet | - | 2 | ✓ | - | - | - | - | - | - | - | - | - | - | ✓ | - | - | ✓ | - |
| Will it send its data to a cloud | ✓ | 2 | - | ✓ | - | ✓ | - | - | ✓ | - | ✓ | ✓ | - | ✓ | - | - | - | - |
| Will it store data in a Database | - | 2 | - | ✓ | - | ✓ | - | - | - | - | ✓ | ✓ | - | ✓ | ✓ | - | - | - |
| Will the system receive regular updates | - | 2 | - | - | - | ✓ | - | - | - | - | - | - | - | - | - | - | - | - |
| Will the system work with third-party software | - | 2 | - | - | - | - | - | ✓ | - | - | - | - | ✓ | - | - | - | - | - |
| Is there a possibility of the communications being eavesdropped | - | 2 | - | - | ✓ | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Could the messages sent between the system components be captured and resent | - | 2 | - | - | - | - | - | - | ✓ | ✓ | - | - | - | - | - | - | - | - |
| Can someone try to impersonate a user to gain access to private information | - | 2 | - | ✓ | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Can someone with bad intentions gain physical access to the location where this software will be running and obtain private information | - | 2 | - | - | - | - | - | - | - | - | - | - | - | - | ✓ | - | - | - |
| Can someone gain physical access to the machine where the system operates or some of the system components and preform some type of modification to it | - | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | ✓ |
| Total | | | 2 | 7 | 3 | 4 | 4 | 2 | 2 | 1 | 3 | 3 | 1 | 7 | 4 | 4 | 2 | 2 |

### 4.2.2 Security Best Practice Guidelines

The questionnaire for the SBPG module contains 12 questions and 12 recommendations, having a lower amount of recommendations when compared to the SRE module, which will lead to a lower number of total unique outputs. Applying the same logic as for the SRE module, a code for each recommendation, followed by its name, can be seen in the following listing:

- **RA1** – Access Control;
- **RA2** – API;
- **RA3** – Authentication;
- **RC1** – Cross Site Scripting;
- **RC2** – Cryptography;
- **RF1** – File Upload;
- **RI1** – Input Validation;
- **RI2** – IoT Security;
- **RL1** – Logging;
- **RS1** – Session Management;
- **RS2** – SQL Injection;
- **RW1** – Web Service.

Just like for SRE, various aspects of the questionnaire were analysed to guarantee better relevance for future tests, a small summary of some important aspects of the aspects analysed for this module can be observed in table 4.3.

Table 4.3: Summary of relevant elements in the SBPG module.

| Element | Quantity |
|---|---|
| Questions | 12 |
| Recommendations | 12 |
| Answer Combinations | 1 206 576 |
| Questions that are Sub-questions | 5 |
| Possible Recommendations Combinations | 4 096 |
| Existent Recommendations Combinations in the Dataset | 70 |

As can be observed in the above table, there were a total of 12 questions, where five of them are sub-questions. When compared with the SRE module, SBPG had a much lower number of possible recommendations, having 4 096 compared to the 65 536 present in the SRE module. Having one less sub-question and more answers possible per question overall, increased the number of total answer combinations to 1 206 576, compared to the 263 424 of the previous module. Furthermore, it can be observed that only 70 of 4 096 possible recommendation combinations were present in the full generated dataset. These differences showed that both models had significant differences in size and content.

To further analyse the questionnaire, we can see a summary, in table 4.4, of the questions available in the questionnaire, and the possible recommendations their answers could generate.

Table 4.4: Summary of the possible recommendations given by each question in the SBPG module, where *SQ* represents if it is a sub-question, *A* the number of answers, and the following columns include the code for each recommendation. Based on the questionnaires investigated in [SSS+20, SLA+21].

| **Question** | **SQ** | **A** | **RA1** | **RA2** | **RA3** | **RC1** | **RC2** | **RF1** | **RI1** | **RI2** | **RL1** | **RS1** | **RS2** | **RW1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| What is the architecture of the system? | - | 9 | - | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | - | ✓ | - | ✓ |
| Does the system use a database? | - | 2 | ✓ | - | - | - | - | ✓ | ✓ | - | ✓ | - | ✓ | - |
| What is the type of data storage? | ✓ | 4 | - | - | - | - | - | - | - | - | - | - | ✓ | - |
| Which database is used? | ✓ | 11 | - | ✓ | ✓ | - | - | - | - | - | ✓ | - | ✓ | - |
| What is the type of data stored? | ✓ | 3 | - | - | - | - | - | - | - | - | - | - | - | - |
| Which type of authentication will be implemented? | - | 7 | - | - | ✓ | - | ✓ | - | - | - | - | - | - | - |
| Will there be user registration? | - | 2 | - | - | - | - | ✓ | - | ✓ | - | - | - | - | - |
| Which way of user registration will be used? | ✓ | 2 | ✓ | ✓ | ✓ | - | ✓ | - | ✓ | - | - | - | - | - |
| Which programming languages will be used in the implementation of the system? | - | 8 | - | - | - | - | - | - | - | - | - | - | - | - |
| Will the system allow input forms? | - | 2 | - | - | - | - | - | - | ✓ | - | - | - | - | - |
| Will the system allow file uploads? | ✓ | 2 | - | - | - | - | - | ✓ | ✓ | - | - | - | - | - |
| Will the system have logging? | - | 2 | - | - | - | - | - | - | - | - | ✓ | - | - | - |
| Total | | | 2 | 3 | 4 | 1 | 4 | 2 | 5 | 1 | 3 | 1 | 3 | 1 |

A few details can be observed from the aforementioned table and detailed in the following listing:

- **Cross Site Scripting**, **IoT Security**, **Session Management**, and **Web Service** are dependent only on one question answer, leaving these recommendations unlikely to be recommended if the wrong answer is given, possibly giving a mostly negative connotation and leaving the dataset with few instances of the positive scenario, possibly making it harder for the machine to learn;
- **Access Control**, and **File Upload** recommendations are dependent on two `yes` or `no` questions, where in both scenarios one of the questions is a sub-question, so if the trigger question for the sub-question is not met, they will be only dependent on a single answer;
- Overall, except for the ones mentioned in the two previous points, the recommendations are more disperse and possible to exhibit both positive and negative scenarios when compared with the SRE module.

## 4.3 Dataset Balance, Augmentation, and Label Correlation

Dataset balancing is important when applying ML to a tool or platform, and can many times be the difference between allowing a model to learn properly or miss important use cases. In this section, the dataset balance analysis that was done to further understand the dataset created using the ETL process (described in section 3.4) will be summarized, along with how it could affect the ML implementation in SRE and SBPG modules of the SAM platform.

### 4.3.1 Security Requirements Elicitation

To analyse the created dataset for the SRE module, the number of positive and negative occurrences of each recommendation in the full dataset were also calculated, containing 263 424 answer combinations, leading to the information contained in table 4.5.

As it can be observed from table 4.5, correlating with the information presented in table 4.2, and as was presumed in section 4.2.1, both **Data Origin** and **Interoperability** recommendations showed a 50% chance of being recommended, and **Authentication** and **Non-Repudiation** had an extremely high percentage of recommendation, above 95%. Furthermore, every recommendation which was present in the first question, except **Tamper Detection**, had an elevated positive percentage, since the first question contained seven possible answers, and some recommendations could be given in more than one of those answers, while also possibly recommended when other questions were answered in a specific way.

As it can also be observed from table 4.5, most recommendations had a tendency to have a positive outcome, with only four recommendations being under 70%, which could lead to

Table 4.5: Summary of the dataset balance of the SRE module using the percentage of positive and negative occurrences of each label.

| Code | Requirement | Positive (%) | Negative (%) |
|---|---|---|---|
| RA1 | Accountability | 85.71 | 14.29 |
| RA2 | Authentication | 98.30 | 1.70 |
| RA3 | Authorization | 95.04 | 4.96 |
| RA4 | Availability | 97.62 | 2.38 |
| RC1 | Confidentiality | 97.96 | 2.04 |
| RC2 | Confinement | 74.49 | 25.51 |
| RD1 | Data Freshness | 66.67 | 33.33 |
| RD2 | Data Origin | 50.00 | 50.00 |
| RF1 | Forgery Resistance | 88.44 | 11.56 |
| RI1 | Integrity | 95.24 | 4.76 |
| RI2 | Interoperability | 50.00 | 50.00 |
| RN1 | Non-Repudiation | 96.94 | 3.06 |
| RP1 | Physical Security | 99.85 | 0.15 |
| RP2 | Privacy | 97.96 | 2.04 |
| RR1 | Reliability | 90.48 | 9.52 |
| RT1 | Tamper Detection | 57.14 | 42.86 |

the machine guessing the correct answer based on being mostly likely positive instead of learning.

Following the dataset, generated by the ETL process, data augmentation was applied, using the SMOTE algorithm. To see how data augmentation balanced the dataset, the results, presented in table 4.6, were extracted.

As it can be observed from the aforementioned table, the dataset generated using the SMOTE algorithm was more balanced, with only four recommendations presenting over 70% in one of their possibilities. The SMOTE algorithm used the previous 263 424 answer combinations to generate 22 222 848 extra answer combinations, leading to a **total of 22 486 272 combinations**. This high number of combinations increases the dataset complexity, leading to some models not being viable to be used in this context, due to hardware restrictions, curtailing possible models in case data augmentation is chosen as a solution to balance the dataset.

To conclude the dataset balance and correlation analysis for the SRE module, the correlation between the recommendations output was studied, to provide further context to the ML implementation. A heatmap of the correlation (calculated using the *Kendall Tau correlation coefficient* implementation on the *pandas* python library [pdt22, WM10]) between recommendations can be observed in figure 4.1.

Several conclusions could be taken from analysing the label correlations, for example, most labels are not correlated, with a few exceptions, those being: 1) Privacy and Confidentiality; 2) Reliability and Accountability; 3) Availability and Integrity; 4) Data Freshness and Data Origin, and 5) Confinement and Interoperability.

Table 4.6: Summary of the dataset balance, augmented using the SMOTE algorithm, of the SRE module using the percentage of positive and negative occurrences of each label.

| Code | Requirement | Positive (%) | Negative (%) |
|---|---|---|---|
| RA1 | Accountability | 47.83 | 52.17 |
| RA2 | Authentication | 67.96 | 32.04 |
| RA3 | Authorization | 61.33 | 38.67 |
| RA4 | Availability | 70.71 | 29.29 |
| RC1 | Confidentiality | 51.49 | 48.51 |
| RC2 | Confinement | 54.92 | 45.08 |
| RD1 | Data Freshness | 52.06 | 47.94 |
| RD2 | Data Origin | 48.86 | 51.14 |
| RF1 | Forgery Resistance | 25.17 | 74.83 |
| RI1 | Integrity | 41.42 | 58.58 |
| RI2 | Interoperability | 45.08 | 54.92 |
| RN1 | Non-Repudiation | 61.10 | 38.90 |
| RP1 | Physical Security | 79.86 | 20.14 |
| RP2 | Privacy | 51.49 | 48.51 |
| RR1 | Reliability | 56.06 | 43.94 |
| RT1 | Tamper Detection | 55.61 | 44.39 |

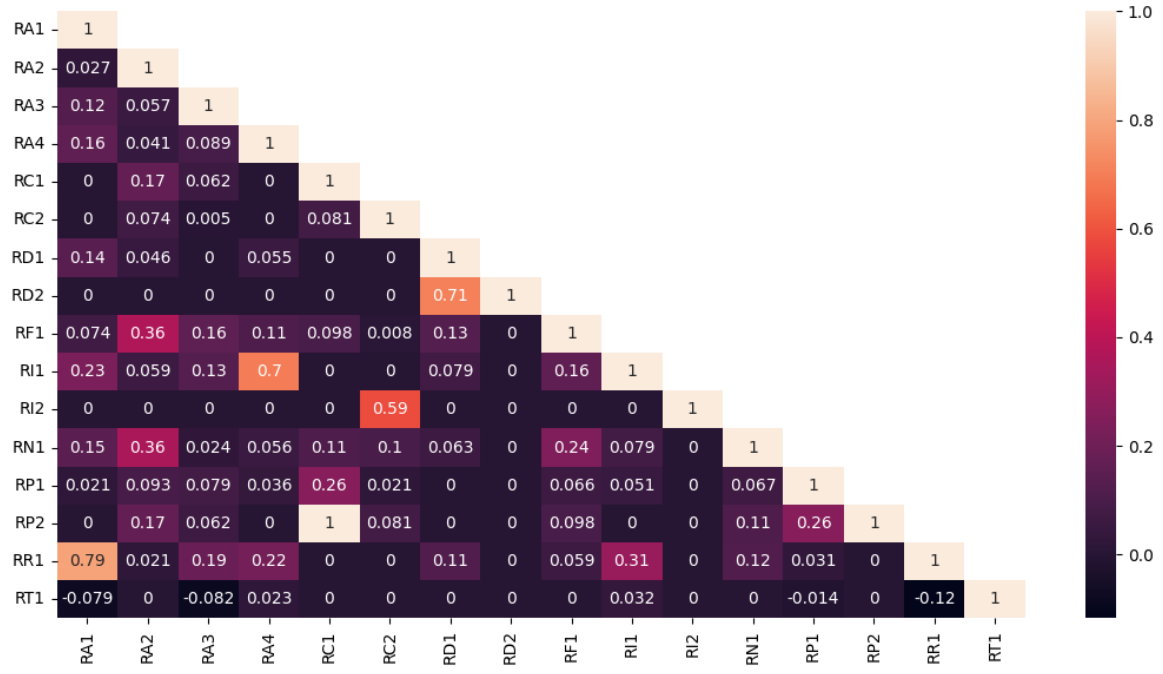


Figure 4.1: Label correlation in the SRE module, with values rounded to three decimal cases.

Most groups presented a low or non-existing correlation, this could bring problems for *Experiment C* since it requires minimal amount of data.

With this analysis it was possible to see that, taking into account the original dataset balance and the label correlations, tests for both the *Experiment B* *Experiment C* could prove difficult. However, since there is a stronger tendency towards positive results (outputting a

certain recommendation), the machine could still correctly predict difficult situations.

### 4.3.2 Security Best Practice Guidelines

To analyse the created dataset for the SBPG module, the same procedure that was described in section 4.3.1 was performed, but for the SBPG module, which contained answer 1 206 576 combinations in the full dataset, leading to the information contained in table 4.7.

Table 4.7: Summary of the dataset balance of the SBPG module using the percentage of positive and negative occurrences of each label.

| Code | Requirement | Positive (%) | Negative (%) |
|---|---|---|---|
| RA1 | Access Control | 99.50 | 0.50 |
| RA2 | API | 62.13 | 37.87 |
| RA3 | Authentication | 100.00 | 0.00 |
| RC1 | Cross Site Scripting | 22.22 | 77.78 |
| RC2 | Cryptography | 100.00 | 0.00 |
| RF1 | File Upload | 99.50 | 0.50 |
| RI1 | Input Validation | 99.92 | 0.08 |
| RI2 | IoT Security | 11.11 | 88.89 |
| RL1 | Logging | 99.62 | 0.38 |
| RS1 | Session Management | 22.22 | 77.78 |
| RS2 | SQL Injection | 99.25 | 0.75 |
| RW1 | Web Service | 11.11 | 88.89 |

As can be observed in the aforementioned table, some recommendations had 100% positive probability, more specifically **Authentication** and **Cryptography**. This was due to being recommended in each answer of the first question, and since the first question was not a sub-question, they would be **recommended in all answer combinations**. Furthermore, just like in the SRE module, most recommendations had a mostly positive probability, with, excluding the previously mentioned, five having **over 99% positive probability**, namely **Access Control**, **File Upload**, **Input Validation**, **Logging**, and **SQL Injection**. In contrast, **IoT Security**, **Web Service**, and **Cross Site Scripting** had all a mostly negative connotation, since, as shown in table 4.2.2, they were fully dependent on only one question, with Web Service and IoT security being only recommended on one of the nine answers, and Cross Side Scripting being present in two. SBPG showed a stronger imbalance when compared to SRE module, since even though it had less possible recommendations and more answer combinations, it had a larger number of answers per question and a lower ratio of recommendation per question.

Following the same procedure as in the SRE module, the SMOTE algorithm was applied, leading to the results shown in table 4.8.

As it can be seen in the aforementioned table, the **Authentication** and **Cryptography** continue to have a 100% positive probability, since there were no negative examples to augment. Overall, the dataset is more balanced between positive and negative cases, with some

exceptions, those being **Input Validation**, **Session Management** and **Web Service**, that, albeit being more balanced than the original dataset, it still had a discrepancy between cases. The augmented dataset contained a total of 36 126 720 answer combinations, 34 920 144 more than the original dataset.

Table 4.8: Summary of the dataset balance, augmented using the SMOTE algorithm, of the SBPG module using the percentage of positive and negative occurrences of each label.

| Code | Requirement | Positive (%) | Negative (%) |
|---|---|---|---|
| RA1 | Access Control | 40.00 | 60.00 |
| RA2 | API | 37.14 | 62.86 |
| RA3 | Authentication | 100.00 | 0.00 |
| RC1 | Cross Site Scripting | 45.71 | 54.29 |
| RC2 | Cryptography | 100.00 | 0.00 |
| RF1 | File Upload | 48.57 | 51.43 |
| RI1 | Input Validation | 85.71 | 14.29 |
| RI2 | IoT Security | 22.86 | 77.14 |
| RL1 | Logging | 55.71 | 44.29 |
| RS1 | Session Management | 11.43 | 88.57 |
| RS2 | SQL Injection | 45.71 | 54.29 |
| RW1 | Web Service | 22.86 | 77.14 |

To conclude the dataset analysis, the label correlation was studied, leading to the graphical representation in figure 4.2.

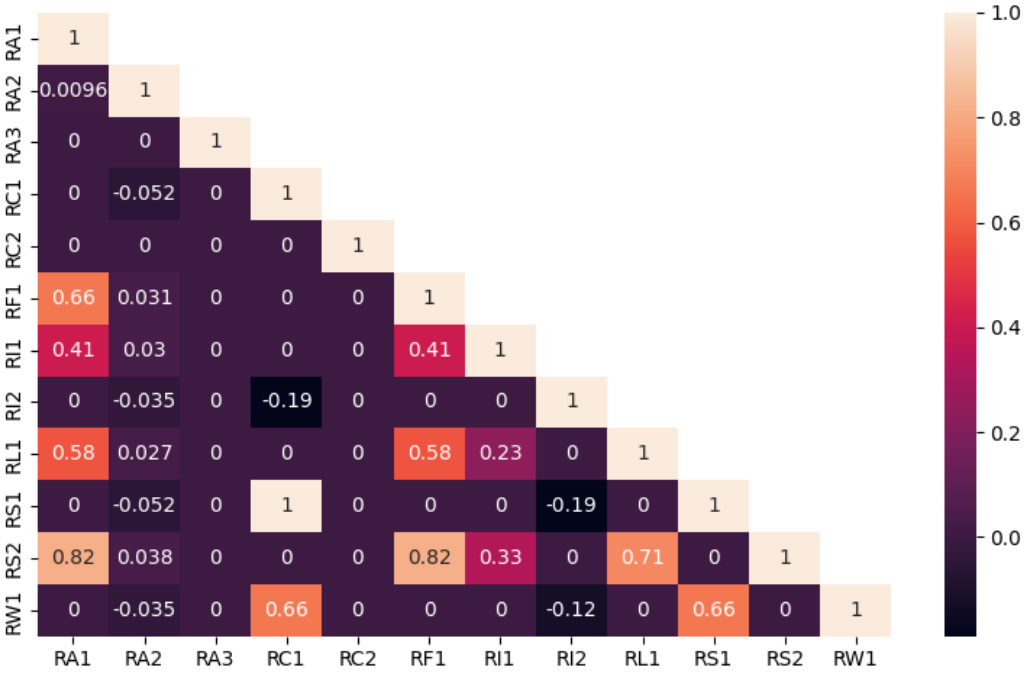


Figure 4.2: Label correlation in the SBPG module, with values rounded to four decimal cases.

In the previously mentioned figure, it can be noted that in similarity to the SRE module, most labels had a low correlation, besides a few exceptions. The strongest correlation came from

Cross Site Scripting and Session Management, with a perfect positive correlation. Additionally, there were some groups with above 70% positive correlation, those being: 1) File Upload and SQL Injection; 2) Access Control and SQL Injection, and 3) SQL Injection and Logging. Just like in the SRE module, this low correlation between modules could bring issues when testing for *Experiment B* and *Experiment C*.

## 4.4 Recommendation Combinations Distribution

To finalize the analysis of the questionnaire for each module, the distribution of the recommendations combinations (labelsets) for both the SRE and SBPG dataset was examined, leading to the data included in table 4.9.

Table 4.9: Summary of the recommendations combinations distribution in the SRE and SBPG modules.

| **Attribute** | **SRE** | **SBPG** |
|---|---|---|
| Minimum Occurrence of a combination | 1 | 56 |
| Maximum Occurrence of a combination | 25 728 | 516 096 |
| Average Occurrence of a combination | 301 | 17237 |
| Combinations That only occurs once | 256 | 0 |
| Combinations That occur more than the average | 68 | 8 |
| Combinations That occur less than the average | 806 | 62 |

### 4.4.1 Security Requirements Elicitation

As can be observed in table 4.9, and reminding that the SRE module had a total of 874 unique sets of recommendations and a total of 263 424 instances, this module contains a large number of labelsets that had only one occurrence in the entire dataset, and a low quantity of labelsets that occurred more than the average. Following this, for each *Experiment*, the following conclusions were reached:

- ***Experiment A*** — There are no notorious impacts expected for this experiment, since, due to the use of data stratification, the labels which only happen once will be present in the training split, leading the model to learn that combination but not test it;
- ***Experiment B*** — There were predicted impacts for this experiment due to the low and random amount of training data used. Using only 800 instances, it may be possible for the machine to only train with the less prevalent labelsets, but still be able to predict the most predominant;
- ***Experiment C*** — The use of minimal but specialized dataset can bring advantages to this dataset, since it appears to have a low number of labelsets that are more prevalent. If the specialized dataset contained answers which correctly represented those

labelsets, the model would be able to correctly predict most of the answer combinations. However, it is to be noted that it also means that there are a high number of answer combinations that lead to the same labelset, which can lead to difficulties for the machine to interpret what causes that specific label to be recommended.

It is important to note that the low average occurrence of a recommendation is influenced by the number of combinations that only occur once.

### 4.4.2 Security Best Practice Guidelines

As can be observed in table 4.9, and reminding that the SBPG module had a total of 70 unique sets of recommendations and a total of 1 206 576 instances, the dataset generated for this module did not have an output combination that occurred only once, but still had a high quantity of labelsets without a significant presence in the dataset. Following this, for each *Experiment*, the following conclusions were reached:

- ***Experiment A*** — Just like for SRE, there were no noteworthy impacts expected for this experiment, due to the use of data stratification;
- ***Experiment B*** — There were predicted impacts for this experiment, due to the low and random amount of training data to be used, since there is a high quantity of instances and a low amount of different combinations, meaning that the random training dataset may take more instances to have an example of one of the most prevalent labelsets to train;
- ***Experiment C*** — The use of minimal but specialized dataset can bring advantages to this dataset, since it only has 70 labelsets. If the specialized dataset contained those labelsets, starting with the most prevalent ones, it could start to correctly predict early. However, it also means that a numerous amount of answer combinations existed for each dataset, possibly leading the machine to not properly learn what causes each label to be recommended.

## 4.5 Conclusions

In this chapter, it was possible to do a deep analysis of the current state of the SRE and SBPG module questionnaires, as well as an analysis of the datasets generated using the ETL process and the application of the SMOTE algorithm. It was possible to conclude that, for the SRE module, even though it had a lower amount of possible answer combinations when compared to the SBPG module, the amount of unique outputs present is much larger, due to the higher number of both possible recommendations and questions. It was also possible to see that some recommendations had a low representation in the questionnaire, with some being only dependent on one question output, so a correct split, of the different labelsets, between the training and testing dataset, must be done to prevent some outputs from not being trained.

Furthermore, when analysing the generated dataset balances, it was possible to verify that both the generated datasets for each model are imbalanced, but had a mostly positive connotation for a lot of recommendations, whereas the biggest missing factor are examples of negative outputs. It was also possible to see that applying the SMOTE algorithm improves the aforementioned problems, but greatly increases the dataset size, leading to a probable increase in computational complexity when applying ML as well as possible storage issues. Additionally, for each module, it was possible to analyse the label correlation, concluding that for both datasets, most labels had low correlation between each other, albeit with some exceptions. This could lead to problems when learning with certain models that use label correlation as an advantage.

Finally, it was possible to examine how the different recommendations combinations are distributed, concluding that it can bring impacts for both *Experiment B* and *Experiment C* in both modules.

# Chapter 5

# Analysis of the Results

## 5.1 Introduction

In this chapter, the MLC implementation results for the SRE and SBPG modules of the SAM platform are summarized. Each experiment will be explored and examined for both modules (sections 5.2,5.3, and 5.4), ending with a conclusion of the best model and base classifier for each situation (section 5.5).

## 5.2 Results Experiment A

*Experiment A*, as detailed in section 3.3, was the most realistic scenario for the SRE and SBPG module in the SAM platform as of the writing of this dissertation. This experiment considers that all the information regarding the questionnaire is present in the database, and the ETL process previously detailed in section 3.4 had been used to create a dataset with all possibilities.

With the above taken into consideration, and using the knowledge obtained in chapter 2, and the dataset creation detailed in sections 4.2 and 4.3, the models CC, BR, LSP and LP were chosen to be tested, complemented with the base classifiers DTC, ETC, ETC2 and AB.

The metric used for these tests was accuracy, which is calculated using formula 5.1. Notice that $\boldsymbol{TP}$ denotes true positives, $\boldsymbol{TN}$ denotes true negatives, $\boldsymbol{FP}$ denotes false positives and $\boldsymbol{FN}$ denotes false negatives. This notation is also used later when explaining the metrics in sections 5.3 and 5.4.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.1)$$

It is also important to note that good accuracy results are to be expected, due to being a closed input space, where all answer combinations are present in the dataset, making the most relevant part of these results the training time. Furthermore, in the computer specified in table 3.1, the creation of the dataset took an average of five seconds for each module, and the augmentation of the data using SMOTE took around 20 seconds for each module.

### 5.2.1 Security Requirements Elicitation

As it was detailed in section 4.2.1, SRE contained a total of 263 424 answer combinations, leading to a split of 237 081 combinations for the training dataset and 26 342 combinations for the testing dataset, this split was done using data stratification, as explained in section 3.5. The results in terms of accuracy and training speed are the ones presented in table 5.1, where N/A values represent memory errors (in this particular case, the library used was loading all data structures to RAM memory, which was not sufficient for some cases and classifiers).

Table 5.1: Results of the model and base classifiers tests in terms of accuracy and training speed for *Experiment A* in the SRE module.

| **Model** | **DTC** | **ETC** | **ETC2** | **AB** | **Avg.** |
|---|---|---|---|---|---|
| **Accuracy (%)** | | | | | |
| Classifier Chains | 100 | 100 | 93 | 100 | 98.25 |
| Binary Relevance | 100 | 100 | N/A | 100 | 100 |
| Label Powerset | 100 | N/A | 80 | 23 | 67.23 |
| Label Space Partitioning | 100 | 99 | 88 | 100 | 96.75 |
| **Avg.** | 100 | 99.67 | 87 | 80.75 | |
| **Train Time (s)** | | | | | |
| Classifier Chains | 160 | 2005 | 192 | 150 | 914 |
| Binary Relevance | 5 | 1839 | N/A | 106 | 650 |
| Label Powerset | 5 | N/A | 10 | 262 | 92.3 |
| Label Space Partitioning | 9 | 1839 | 31 | 108 | 496.75 |
| **Avg.** | 44.79 | 682 | 77.67 | 156.2 | |

The analysis of the results can be summarized as follows, starting with the base classifiers:

- **DTC** — In terms of accuracy, DTC shows the best results of all the base classifiers, obtaining 100% accuracy in all tests. In terms of time, it also shows the best results overall, taking only five seconds when used in combination with LP and LSP;
- **ETC** — ETC failed when used in combination with LP, but showed very good accuracy results when used with CC and BR. In terms of time, ETC showed the worst results when compared to the remaining base classifiers;
- **ETC2** — In terms of accuracy, ETC2 showed the worst results and failed when used in combination with BR. In terms of speed, it presented better results than ETC and AB, but fell short when compared with DTC;
- **AB** — In terms of accuracy, AB had the worst average, but only due to its use when combined with LP that only achieved 23%, which contrasts its results with the remaining models, where it achieved 100%. In terms of speed, AB had the second-slowest time.

Following the base classifiers, the models results can be summarized as follows:

- **CC** — In terms of accuracy, CC obtained very positive results, showing the second-best results overall. In terms of speed, its training is slow compared to LP and BR;

- **LP** — In terms of accuracy, LP showed the worst results overall, specially when paired with AB, not being able to surpass the 70% accuracy. In terms of speed, it was the fastest training model, but it did not compensate the low accuracy results;
- **BR** — In terms of accuracy, BR showed the best results, although it did fail, due to memory constraints, when paired with ETC2. In terms of speed, it showed good results, albeit still far from the LP model training speed;
- **LSP** — In terms of accuracy, LSP showed good results, having above 95% accuracy on average, and 100% accuracy when paired with DTC and AB. In terms of speed, it was the second-fastest training model, but still far from LP, albeit being superior in accuracy results.

For *Experiment A*, accuracy and speed tests were also performed using the dataset generated with the SMOTE algorithm, using all models in combination with the DTC classifier, since it was the best performing base classifier in terms of speed and accuracy. A training dataset of 17 989 017 and a testing dataset of 4 497 254 answer combinations were used, generated using data stratification, as explained in section 3.5. The results achieved are shown in table 5.2.

Table 5.2: Results of each model using the DTC classifier tests in terms of accuracy and training speed for *Experiment A* using the augmented dataset in the SRE module.

| **Model** | **Accuracy (%)** | **Train Speed (s)** |
|---|---|---|
| Classifier Chains | 100 | 691 |
| Label Powerset | 100 | 731 |
| Binary Relevance | 100 | 441 |
| Label Space Partitioning | 100 | 641 |

As it can be observed from the results, all models achieved 100% accuracy, with BR having the best results in terms of speed under seven minutes of training, followed by LSP with around 11 minutes of training, and finally CC and LP with 12 minutes. The efficiency of these results would depend on if it could be acceptable that the model would require this amount of time to train and how many times it would need to train.

In conclusion, if data augmentation was to be used in order to balance the dataset, the best model would be BR with the DTC classifier. If data augmentation was not to be used, both LSP and LP would be good fits, when used in conjunction with DTC, since both achieved the lowest amount of training time and had 100% accuracy.

### 5.2.2 Security Best Practice Guidelines

As it was detailed in section 4.2.2, SBPG contained a total of 1 206 576 answer combinations, leading to a split of 1 085 918 combinations for the training dataset and 1 206 57 combinations for the test dataset, this split was done using data stratification, as explained in section 3.5. The results in terms of accuracy and training speed are the ones presented in table 5.3.

The analysis of the results can be summarized as follows, starting with the base classifiers:

- **DTC** — In terms of accuracy, DTC shows the best results of all the base classifiers, obtaining 100% accuracy in all tests. In terms of time, it also shows the best results overall, taking between 10 and 20 seconds when used alongside a model;
- **ETC** — Using ETC as a classifier showed very good accuracy results. In terms of time, ETC showed the worst results when compared to the remaining base classifiers, by a wide margin;
- **ETC2** — In terms of accuracy, ETC2 showed the worst results, if LP used in combination with AB is to be ignored, but still showed good results overall. In terms of speed, it presented the second-best results compared to the remaining base classifiers;
- **AB** — In terms of accuracy, AB had the worst average, but only due to its use when combined with LP, which contrast its results with the remaining models, where it achieved above 90% accuracy. In terms of speed, AB had the second-slowest training time.

Table 5.3: Results of the model and base classifiers tests in terms of accuracy and training speed for *Experiment A* in the SBPG module.

| **Model** | **DTC** | **ETC** | **ETC2** | **AB** | **Avg.** |
|---|---|---|---|---|---|
| **Accuracy (%)** | | | | | |
| Classifier Chains | 100 | 100 | 95 | 100 | 98.75 |
| Binary Relevance | 100 | 100 | 92 | 100 | 98 |
| Label Powerset | 100 | 100 | 96 | 52 | 87 |
| Label Space Partitioning | 100 | 100 | 100 | 93 | 98.25 |
| **Avg.** | 100 | 100 | 95.75 | 86.25 | |
| **Train Time (s)** | | | | | |
| Classifier Chains | 16 | 11 720 | 71 | 304 | 3 027.75 |
| Binary Relevance | 10 | 13 798 | 98 | 255 | 3 540.25 |
| Label Powerset | 11 | 9 432 | 38 | 127 | 9 608 |
| Label Space Partitioning | 19 | 13 715 | 89 | 252 | 3 518.75 |
| **Avg.** | 14 | 12 166.25 | 74 | 234.5 | |

Following the base classifiers, the models results can be summarized as follows:

- **CC** — In terms of accuracy, CC obtained the best average. In terms of training speed, on average, it is slow compared to LP, but close to the remaining modules;
- **LP** — In terms of accuracy, LP showed the worst average. However, ignoring its use with the AB classifier, its accuracy showed very good results, having above 90% accuracy. In terms of speed it was the fastest training model;
- **BR** — In terms of accuracy, BR showed the third-best results on average, but with a low margin to the top scored models. In terms of speed, it showed good results, albeit still far from the LP model training speed;

- **LSP** — In terms of accuracy, LSP showed good results, obtaining the second-best results on average. In terms of speed, it was the third-fastest training model, but still distant from the low training speeds of the LP model.

For *Experiment A*, accuracy and speed tests were also performed using the dataset generated with the SMOTE algorithm, using all models in combination with the DTC classifier. To perform this tests, a training dataset of 28 901 376 and a testing dataset of 7 225 344 answer combinations were used, generated using data stratification, as explained in section 3.5. The results achieved are shown in table 5.4.

Table 5.4: Results of each model using the DTC classifier tests in terms of accuracy and train speed for *Experiment A* using the augmented dataset in the SRE module.

| Model | Accuracy (%) | Train Speed (s) |
|---|---|---|
| Classifier Chains | 100 | 600 |
| Label Powerset | 100 | 270 |
| Binary Relevance | 100 | 438 |
| Label Space Partitioning | 100 | 713 |

Similar to the tests performed for the SRE module, all models achieved 100% accuracy, with the training speed being the defining factor, having the LP model achieving the best training time in comparison to the remaining models.

In conclusion, if data augmentation was to be used in order to balance the dataset, the best model would be LP with the DTC classifier. If data augmentation was not to be used, BR or LP in conjunction with the DTC base classifier would be the best choices, since both achieved the lowest amount of training time and had 100% accuracy.

## 5.3 Results Experiment B

*Experiment B*, as detailed in section 3.3, is a possible scenario for the SRE and SBPG modules in the SAM platform in the future, if a security expert is not available but left a manual for future questions, or if he can correct possible given answers. This occurs when there is a questionnaire based on the objective, but there is no security expert available to correctly update the database, and simultaneously it is not possible to create a dataset with all possibilities.

With the above taken into consideration, and using the knowledge obtained in chapter 2, and the dataset creation detailed in sections 4.2 and 4.3, the models CC, BR, LSP and LP were chosen to be tested, complemented with the base classifiers DTC, ETC and ETC2. Furthermore, the primary objective when testing this experiment is to analyse how well the models learn with low amounts of random data. Therefore, a maximum number of 2000 instances for the SRE module, and 4000 instances for the SBPG were used in the training set, and all possible answer combinations as the testing set, simulating the real case scenario. Different training sizes are used to show how the classifiers adapt and improve according to the

number of cases.

To better analyse the results present in *Experiment B*, mean based accuracy [LZC+16], also known as balanced accuracy, was used, being calculated using the equation 5.2.

$$MeanAccuracy = \frac{1}{2N} \sum_{i=1}^{L} (TP_i/P_i + TN_i/N_i) \quad (5.2)$$

For this equation, $\boldsymbol{L}$ represents the number of attributes, $\boldsymbol{N}$ the number of examples, $\boldsymbol{Pi}$ the quantity of positive example, $\boldsymbol{TPi}$ the well predicted positive examples, $\boldsymbol{Ni}$ the same as $\boldsymbol{Pi}$ but for negative examples, $\boldsymbol{TNi}$ the same as $\boldsymbol{TPi}$ but for negative examples. The use of this metric will allow comparing the results with the ones achieved in the previous implementation [LCS+21] (recall that the mean accuracy was used for this experiment as per explanation in 2.4).

### 5.3.1 Security Requirements Elicitation

As it was detailed in section 4.2.1, SRE supports a total of 237 081 answer combinations, this being our testing dataset in this context, where 2000 random instances were chosen as the training dataset, since one of the objectives is for the model to learn with the lowest amount of data. The results in terms of mean accuracy were the ones presented in table 5.5 and the training speed in table 5.6.

Table 5.5: Results of the model and base classifiers tests in terms of mean accuracy for *Experiment B* in the SRE module.

| **Model** | **Training Size** | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 10 | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 | 1500 | 2000 |
| **DTC** | | | | | | | | | | | | | | |
| CC | 0.5 | 0.67 | 0.84 | 0.87 | 0.87 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 |
| LP | 0.5 | 0.56 | 0.71 | 0.7 | 0.71 | 0.7 | 0.72 | 0.74 | 0.74 | 0.75 | 0.77 | 0.76 | 0.78 | 0.79 |
| BR | 0.5 | 0.68 | 0.84 | 0.87 | 0.87 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 |
| LSP | 0.5 | 0.68 | 0.85 | 0.87 | 0.87 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 |
| **ETC** | | | | | | | | | | | | | | |
| CC | 0.5 | 0.64 | 0.78 | 0.79 | 0.8 | 0.8 | 0.81 | 0.81 | 0.83 | 0.84 | 0.85 | 0.85 | 0.85 | 0.86 |
| LP | 0.5 | 0.6 | 0.73 | 0.74 | 0.75 | 0.75 | 0.74 | 0.75 | 0.75 | 0.76 | 0.76 | 0.77 | 0.78 | 0.79 |
| BR | 0.5 | 0.65 | 0.79 | 0.79 | 0.81 | 0.81 | 0.81 | 0.82 | 0.83 | 0.84 | 0.83 | 0.85 | 0.85 | 0.87 |
| LSP | 0.5 | 0.64 | 0.79 | 0.8 | 0.81 | 0.81 | 0.82 | 0.81 | 0.84 | 0.84 | 0.85 | 0.85 | 0.86 | 0.86 |
| **ETC2** | | | | | | | | | | | | | | |
| CC | 0.5 | 0.62 | 0.74 | 0.75 | 0.76 | 0.74 | 0.72 | 0.77 | 0.76 | 0.81 | 0.77 | 0.80 | 0.77 | 0.83 |
| LP | 0.5 | 0.57 | 0.7 | 0.73 | 0.71 | 0.7 | 0.72 | 0.74 | 0.74 | 0.74 | 0.74 | 0.75 | 0.77 | 0.79 |
| BR | 0.5 | 0.68 | 0.84 | 0.87 | 0.87 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 |
| LSP | 0.5 | 0.68 | 0.84 | 0.86 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 |

The analysis of the results can be summarized as follows, starting with the base classifiers:

- **DTC** — In terms of mean accuracy, DTC shows good results with all models except LP, being able to achieve its max mean accuracy values with 400 instances, which corresponds to the best results when compared to the remaining base classifiers. In terms

of speed it presents very good results, with every training instance running under one second;

- **ETC** — In terms of mean accuracy, ETC shows the worst results overall, not achieving the best results of the remaining base classifiers. In terms of speed, it also presents the worst results;
- **ETC2** — In terms of mean accuracy, ETC2 performs better with BR and LSP, being able to achieve its max mean accuracy values with 400 instances, just like DTC, also corresponding to second-best result compared to the remaining base classifiers. In terms of speed, it presents similar results to DTC, with every training instance running under one second;

Table 5.6: Results of the model and base classifiers tests in terms of training time, in seconds, for *Experiment B* and SRE module.

| **Model** | Training Size | | | |
|---|---|---|---|---|
| | 500 | 1000 | 1500 | 2000 |
| **DTC** | | | | |
| CC | 0.02 | 0.02 | 0.05 | 0.03 |
| LP | 0.01 | 0.01 | 0.02 | 0.02 |
| BR | 0.01 | 0.03 | 0.03 | 0.05 |
| LSP | 0.03 | 0.05 | 0.06 | 0.08 |
| **ETC** | | | | |
| CC | 2.55 | 2.86 | 3.56 | 5.08 |
| LP | 0.33 | 0.77 | 1.70 | 2.78 |
| BR | 2.11 | 2.86 | 3.96 | 4.45 |
| LSP | 2.15 | 3.03 | 3.08 | 5.12 |
| **ETC2** | | | | |
| CC | 0.02 | 0.03 | 0.05 | 0.05 |
| LP | 0.01 | 0.01 | 0.01 | 0.02 |
| BR | 0.01 | 0.01 | 0.02 | 0.05 |
| LSP | 0.03 | 0.05 | 0.06 | 0.07 |

Following the analysis of the base classifiers, the models results can be summarized as follows:

- **CC** — In terms of mean accuracy, CC is able to increasingly have better mean accuracy results, showing the second-best results overall. In terms of speed, its training is slow compared to LP and BR;
- **LP** — In terms of mean accuracy, LP shows the worst results overall, not being able to surpass the 80% accuracy with only 2000 instances. In terms of speed it is the fastest training model;
- **BR** — In terms of mean accuracy, BR shows good results, being able to achieve 89% with specific base classifiers. In terms of speed, it presents optimal results, being second only to the LP model;
- **LSP** — In terms of mean accuracy, LSP shows very good results, surpassing CC when paired with ETC2. In terms of speed, it falls amongst the two slowest models. However, it takes little over one second to reach the top results with DTC as a base classifier.

The results were more positive than expected given the label correlation showed in figure 4.1, where it could be observed that most labels did not have a good correlation with each other. The positive results can be attributed to certain labelsets having a bigger presence in the dataset compared to the remaining, leading to the machine learning those specific labelsets, while the remaining cases in which the correct output was not achieved represents specific and possibly unrealistic scenarios. This reason could also explain the low performance of the LP method when compared to the remaining models, since it would encounter problems when attempting to predict unseen unique labelsets, while BR, even with its limitation of ignoring label relations, still achieves good results. Furthermore, it can also explain how using only 400 instances an 89% mean accuracy result was achieved, since in those random instances most common labelsets may be present. There is no confirmation that the machine is properly learning each label, since most labels have a majority positive or negative connotation, possibly leading to guessing some outputs depending on the majority of cases, although, from a user standpoint, if they were to use the platform whose output was given by the machine, it would mostly get the correct output recommended to them.

Comparing the results with the ones obtained in the previous ML implementation in the SRE module [LCS+21], the peak *Mean Accuracy* results were not as elevated in this implementation compared to the previous implementation, possibly due to the questionnaire having grown larger since the last implementation, having extra answers to some questions as well as using the full dataset as the test dataset in this implementation.

To conclude the analysis, all models paired with DTC as a base classifier, except for LP would be a good choice to use in the SRE module for *Experiment B*, taking only 400 instances to reach very good results, with CC and BR being the best suited models to be used due to having high mean accuracy and also low training time.

### 5.3.2 Security Best Practice Guidelines

As it was detailed in section 4.2.2, SBPG contained a distribution of 1 206 576 answer combinations, this being our test dataset in this context, and, due to the higher amount of answer combinations, the max number of random instances used for the training dataset were increased to 4000 compared to the 2000 used for the SRE module. The results in terms of mean accuracy were the ones presented in table 5.7 and the training speed in table 5.8.

The analysis of the results can be summarized as follows, starting with the base classifiers:

- **DTC** — In terms of mean accuracy, DTC shows great results with all models, being its usage with LSP the only one not achieving 91% with 4000 instances. It is to be noted that using between 300 and 2000 training instances lowered the results, probably as it did not learn correctly due to a lack of specific cases, but did start to better adapt when it reached 2000 instances. In terms of speed it presented very good results, with every training instance running under one second;
- **ETC** — In terms of mean accuracy, ETC achieved the second best results both in the

earlier stages and with 4000 training instances, and also presented a steadier growth in accuracy compared to DTC but never achieving as good results as it. In terms of speed, it presented the worst results;

- **ETC2** — In terms of mean accuracy, ETC2 showed the worst results when compared to the remaining base classifiers, having the lowest max mean accuracy with the 4000 instances and in the earlier stages. In terms of speed, it presented similar results to DTC, with every training instance running under one second.

Table 5.7: Results of the model and base classifiers tests in terms of mean accuracy for *Experiment B* in the SBPG module.

| **Model** | **Training Size** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 10 | 100 | 200 | 300 | 400 | 500 | 700 | 1000 | 1500 | 2000 | 3000 | 4000 |
| **DTC** | | | | | | | | | | | | | |
| CC | 0.5 | 0.87 | 0.9 | 0.9 | 0.88 | 0.87 | 0.87 | 0.87 | 0.86 | 0.84 | 0.87 | 0.89 | 0.91 |
| LP | 0.5 | 0.87 | 0.9 | 0.9 | 0.88 | 0.87 | 0.87 | 0.87 | 0.86 | 0.84 | 0.87 | 0.89 | 0.91 |
| BR | 0.5 | 0.88 | 0.9 | 0.9 | 0.88 | 0.87 | 0.87 | 0.87 | 0.82 | 0.82 | 0.89 | 0.89 | 0.91 |
| LSP | 0.5 | 0.88 | 0.9 | 0.9 | 0.87 | 0.87 | 0.87 | 0.87 | 0.83 | 0.85 | 0.89 | 0.89 | 0.9 |
| **ETC** | | | | | | | | | | | | | |
| CC | 0.5 | 0.73 | 0.78 | 0.79 | 0.78 | 0.8 | 0.79 | 0.8 | 0.79 | 0.8 | 0.87 | 0.88 | 0.9 |
| LP | 0.5 | 0.75 | 0.78 | 0.81 | 0.79 | 0.8 | 0.8 | 0.82 | 0.8 | 0.83 | 0.87 | 0.87 | 0.91 |
| BR | 0.5 | 0.71 | 0.76 | 0.79 | 0.77 | 0.76 | 0.79 | 0.8 | 0.81 | 0.82 | 0.88 | 0.89 | 0.91 |
| LSP | 0.5 | 0.72 | 0.79 | 0.78 | 0.78 | 0.79 | 0.8 | 0.8 | 0.8 | 0.82 | 0.88 | 0.89 | 0.9 |
| **ETC2** | | | | | | | | | | | | | |
| CC | 0.5 | 0.67 | 0.73 | 0.71 | 0.74 | 0.72 | 0.79 | 0.77 | 0.79 | 0.73 | 0.76 | 0.75 | 0.89 |
| LP | 0.5 | 0.72 | 0.72 | 0.68 | 0.59 | 0.75 | 0.78 | 0.72 | 0.74 | 0.73 | 0.79 | 0.75 | 0.82 |
| BR | 0.5 | 0.7 | 0.72 | 0.75 | 0.74 | 0.81 | 0.72 | 0.73 | 0.76 | 0.74 | 0.82 | 0.82 | 0.82 |
| LSP | 0.5 | 0.6 | 0.65 | 0.67 | 0.67 | 0.81 | 0.74 | 0.76 | 0.78 | 0.73 | 0.84 | 0.82 | 0.81 |

Table 5.8: Results of the model and base classifiers tests in terms of training time, in seconds, for *Experiment B* and SBPG module.

| **Model** | Training Size | | | |
|---|---|---|---|---|
| | 500 | 1000 | 2000 | 4000 |
| **DTC** | | | | |
| CC | 0.02 | 0.02 | 0.03 | 0.05 |
| LP | 0.01 | 0.01 | 0.02 | 0.02 |
| BR | 0.01 | 0.02 | 0.03 | 0.03 |
| LSP | 0.02 | 0.02 | 0.05 | 0.06 |
| **ETC** | | | | |
| CC | 1.25 | 2.08 | 3.19 | 5.52 |
| LP | 0.17 | 0.53 | 1.53 | 2.30 |
| BR | 2.47 | 2.72 | 3.95 | 6.86 |
| LSP | 2.44 | 2.91 | 4.55 | 7.44 |
| **ETC2** | | | | |
| CC | 0.02 | 0.02 | 0.03 | 0.05 |
| LP | 0.01 | 0.02 | 0.02 | 0.02 |
| BR | 0.02 | 0.03 | 0.03 | 0.05 |
| LSP | 0.03 | 0.03 | 0.05 | 0.09 |

Following the analysis of the base classifiers, the models results can be summarized as follows:

- **CC** — In terms of mean accuracy, CC presents the top results compared with other models, and obtains its peak accuracy with 200 instances and 4000 instances. CC reaches mean accuracy above 80% throughout most of the test when paired with DTC. In terms of speed, its training is slow compared to LP, but has similar results to BR;

- **LP** — In terms of mean accuracy, LP shows good results, showing worse results when paired with ETC2 and in comparison to the remaining models. In terms of speed it is the fastest training model;
- **BR** — In terms of mean accuracy, BR shows similar results to LP, but achieving the second accuracy peak earlier with 2000 instances when paired with ETC. In terms of speed, it is the second-fastest training model except when paired with ETC;
- **LSP** — In terms of mean accuracy, LSP shows similar results to CC, while, analogously to BR, achieving the second accuracy peak earlier with 2000 instances. In terms of speed, it is the second-slowest model.

Similar to the tests performed for this experiment in the SRE module, the results were more positive than expected given the label correlation showed in figure 4.2, which had even worse correlation compared to the SRE module. Related to the reasons given for the SRE module, the positive results can be attributed to certain labelsets having a larger presence in the dataset compared to the remaining, as it was explored in section 4.4.2. Furthermore, there were recommendations which had a 100% positive representation, leading to the machine to better adapt to providing those answers. The maximum mean accuracy also took longer to be reached, which can be explained by the bigger number of possible answer combinations in conjunction with a lower amount of unique recommendation combinations.

To conclude the analysis, if MLC was to be developed for SBPG with a low amount of random data, LSP or BR paired with DTC would be the most fitting choice. Compared to the results obtained for the SRE module, the models take more time to reach peak accuracy and start to fall off between 400 and 1500 training instances. This is most probably due to a much larger amount of total answer combinations while having a lower amount of labels and labelsets, meaning that if in the random training dataset those labelsets are not present, it is harder for the model to predict specific scenarios.

## 5.4 Results Experiment C

*Experiment C*, as detailed in section 3.3, is a possible scenario for the SRE and SBPG modules in the SAM platform in the future, if only a security expert is available, but having a restricted schedule and only being able to give and correct 10 answer combinations, which would correctly relate to realistic scenarios. The main purpose of these results is to evaluate, if the above were to be confirmed, how many of the labels would be guessed correctly.

The metrics **Precision**, **Recall** and **F1-score** were chosen, and calculated in a per label basis allowing to have a better understanding how well each label is being predicted instead of a general evaluation, since a full labelset can be wrongly predicted, but most of the labels be correctly predicted. The metrics can be summarized as follows:

- **Precision** : The percentage of true positives amid all positive predictions, calculated

using equation 5.3;

$$Precision = \frac{TP}{TP + FP} \quad (5.3)$$

- **Recall** : The percentage of true positives amid all positive examples, calculated using equation 5.4;

$$Recall = \frac{TP}{TP + FN} \quad (5.4)$$

- **F1-score** : The harmonic mean average between precision and recall, calculated using equation 5.5;

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (5.5)$$

Furthermore, the dataset was created manually, using information from the dataset created automatically. However, the chosen answer combinations were based on realistic IoT scenarios proposed by **SECURIoTESIGN** developers [SEC21], which contained scenarios for: smart cities; smart transportation and logistics; healthcare; smart manufacturing; smart wearables; agriculture; smart grids, and smart environments. Using the aforementioned scenarios it was then possible to create a specialized dataset for both the SRE and SBPG module, which was used as a training dataset when performing the tests. Akin to the *Experiment B* tests, the full dataset of possible answer combinations was used as a test dataset.

Due to the conclusions taken for *Experiment B*, the chosen base classifier to use in conjunction with the models was DTC.

### 5.4.1 Security Requirements Elicitation and Security Best Practice Guidelines

The results of Experiment C on both modules can be observed in tables 5.9 and 5.10 for the SRE and SBPG modules, respectively. As it can be observed from the results, all models had similar results, and due to the possibility of escalation, the models chosen for the *Experiment B* should be the ones also chosen in the future if a larger specialized dataset was to be created. The results were very positive, having only 10 instances for training, 13 of the 16 recommendations in the SRE module, and eight of 12 recommendations in the SBPG module had above 80% *f1-score* in at least two models. The recommendations with the biggest prediction difficulties were the ones with a more balanced distribution between positive and negative scenarios (for example, RD1 and RD2 in the SRE module), or highly specific scenarios not present in the specialized dataset (for example, RW1 in the SBPG). Related to the previously mentioned, the positive results on the SBPG module are also probably related to a high number of recommendations having a close to 100% probability of being recommended, and for some recommendations having only a case where it contradicts the majority sample, leading to a scenario that if the specialized dataset contains the specific case, the model adapts with more ease.

Table 5.9: Results of the tests in terms of precision, recall and f1-score for *Experiment C* using all models with DTC base classifier in the SRE module.

| Model | Metric | RA1 | RA2 | RA3 | RA4 | RC1 | RC2 | RD1 | RD2 | RF1 | RI1 | RI2 | RN1 | RP1 | RP2 | RR1 | RT1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Classifier Chains | Precision | 0.86 | 0.98 | 0.95 | 0.98 | 1.0 | 0.74 | 0.67 | 0.5 | 0.91 | 0.95 | 1.0 | 1.0 | 1.0 | 0.98 | 0.9 | 1.0 |
| | Recall | 1.0 | 0.5 | 1.0 | 1.0 | 1.0 | 0.5 | 0.5 | 0.5 | 0.69 | 1.0 | 1.0 | 0.69 | 1.0 | 0.67 | 1.0 | 0.88 |
| | F1-score | 0.92 | 0.66 | 0.97 | 0.99 | 1.0 | 0.6 | 0.57 | 0.5 | 0.79 | 0.98 | 1.0 | 0.81 | 1.0 | 0.79 | 0.95 | 0.93 |
| Label Powerset | Precision | 0.86 | 0.99 | 0.95 | 0.98 | 0.99 | 0.75 | 0.67 | 0.5 | 0.89 | 0.95 | 0.5 | 0.97 | 1.0 | 0.99 | 0.9 | 0.63 |
| | Recall | 1.0 | 0.99 | 1.0 | 1.0 | 1.0 | 0.99 | 0.99 | 0.99 | 0.99 | 1.0 | 0.88 | 0.99 | 1.0 | 1.0 | 1.0 | 0.94 |
| | F1-score | 0.92 | 0.99 | 0.97 | 0.99 | 1.0 | 0.85 | 0.8 | 0.67 | 0.94 | 0.98 | 0.64 | 0.98 | 1.0 | 1.0 | 0.95 | 0.75 |
| Binary Relevance | Precision | 0.86 | 0.99 | 0.95 | 0.98 | 0.98 | 0.74 | 0.67 | 0.5 | 0.89 | 0.95 | 1.0 | 0.97 | 1.0 | 0.98 | 0.9 | 1.0 |
| | Recall | 1.0 | 0.67 | 1.0 | 1.0 | 0.5 | 0.5 | 0.5 | 0.67 | 0.98 | 1.0 | 1.0 | 0.98 | 1.0 | 0.67 | 1.0 | 0.88 |
| | F1-score | 0.92 | 0.8 | 0.97 | 0.99 | 0.66 | 0.6 | 0.57 | 0.57 | 0.93 | 0.98 | 1.0 | 0.98 | 1.0 | 0.79 | 0.95 | 0.93 |
| Label Space Partitioning | Precision | 0.86 | 0.99 | 0.95 | 0.98 | 0.98 | 0.74 | 1.0 | 0.5 | 0.88 | 0.95 | 1.0 | 1.0 | 1.0 | 0.98 | 0.9 | 1.0 |
| | Recall | 1.0 | 0.98 | 1.0 | 1.0 | 0.67 | 0.5 | 0.75 | 0.5 | 0.5 | 1.0 | 1.0 | 0.52 | 1.0 | 0.5 | 1.0 | 0.88 |
| | F1-score | 0.92 | 0.98 | 0.97 | 0.99 | 0.79 | 0.6 | 0.86 | 0.5 | 0.64 | 0.98 | 1.0 | 0.68 | 1.0 | 0.66 | 0.95 | 0.93 |

Table 5.10: Results of the tests in terms of precision, recall and f1-score for *Experiment C* using all models with DTC base classifier in the SBPG module.

| Model | Metric | RA1 | RA2 | RA3 | RC1 | RC2 | RF1 | RI1 | RI2 | RL1 | RS1 | RS2 | RW1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Classifier Chains | Precision | 1.0 | 0.64 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.5 | 1.0 | 1.0 | 1.0 | 0.5 |
| | Recall | 1.0 | 0.26 | 1.0 | 1.0 | 1.0 | 1.0 | 0.99 | 1.0 | 0.5 | 1.0 | 1.0 | 0.01 |
| | F1-score | 1.0 | 0.37 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.67 | 0.67 | 1.0 | 1.0 | 0.01 |
| Label Powerset | Precision | 1.0 | 0.67 | 1.0 | 0.81 | 1.0 | 0.99 | 1.0 | 0.22 | 1.0 | 0.81 | 0.99 | 0.25 |
| | Recall | 0.75 | 0.34 | 1.0 | 0.54 | 1.0 | 0.75 | 0.75 | 0.33 | 0.88 | 0.54 | 0.75 | 0.12 |
| | F1-score | 0.86 | 0.45 | 1.0 | 0.65 | 1.0 | 0.86 | 0.86 | 0.27 | 0.93 | 0.65 | 0.85 | 0.17 |
| Binary Relevance | Precision | 1.0 | 0.64 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.5 | 1.0 | 1.0 | 1.0 | 0.5 |
| | Recall | 1.0 | 0.26 | 1.0 | 1.0 | 1.0 | 1.0 | 0.99 | 1.0 | 0.5 | 1.0 | 1.0 | 0.01 |
| | F1-score | 1.0 | 0.37 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.67 | 0.67 | 1.0 | 1.0 | 0.01 |
| Label Space Partitioning | Precision | 1.0 | 0.64 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.5 | 1.0 | 1.0 | 1.0 | 0.5 |
| | Recall | 1.0 | 0.26 | 1.0 | 1.0 | 1.0 | 1.0 | 0.99 | 1.0 | 0.5 | 1.0 | 1.0 | 0.01 |
| | F1-score | 1.0 | 0.37 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.67 | 0.67 | 1.0 | 1.0 | 0.01 |

It is to be noted that these positive results may result from the machine guessing a high number of elements, and since most labels have a higher tendency to either be recommended or not recommended, it will tend to predict that outcome. However, if a security expert attempts to contradict labels which always show positive results by creating scenarios where it should not, the machine would learn the contradictory situation and learn better.

The advantages of using a specialized dataset can be observed, for example, when analysing the results for the RT1 recommendation in the SRE module, and comparing it with the same results when using the random dataset, the differences being visible in table 5.11.

Table 5.11: Comparison of the results obtained, using BR, for the *RT1* recommendation in *Experiment B* and *C* for the SRE module.

| Metric | Specialized Dataset | Random Dataset |
|---|---|---|
| Precision | 1.00 | 0.64 |
| Recall | 0.88 | 0.94 |
| F1-score | 0.93 | 0.76 |

As it can be observed from the results, and taking into account that, in the random dataset, not all use cases for this recommendation output were present, we can see that, by having the specific case scenarios, the precision greatly increased when using the specialized dataset, meaning that the predictions of this recommendation were correctly identified 100% of the cases compared to the 64% percent when using the random dataset while having similar values in recall.

It can be concluded that a specialized dataset brings advantages to this implementation, and with a minimal amount of data can bring good results. To avoid making the machine guess some labels, the expert should also focus on creating scenarios that contradict the recommendations of high performing labels, which may lead to have both realistic scenarios and unrealistic scenarios. The end user answering a questionnaire whose output uses this machine, with a specialized dataset, will have most labels be well recommended.

## 5.5 Conclusions

In this chapter the results for the tests regarding *Experiments A, B and C* were analysed. *Experiments A* and *B* are not really realistic, and they serve the purpose of providing a basis to compare techniques to test in *Experiment C*. *Experiment A* contains the entire possible set, and corresponds to making a model that knows all the outcomes. A summary of the best models and base classifiers for every experiment and module can be seen in table 5.12.

For ***Experiment A***, positive accuracy results were thus to be expected due to having a limited and fully accessible input and output space, being the main focus how data augmentation would affect this situation and how fast the models would learn, leading to the top models learning in five seconds or less for the original dataset and under 10 minutes for the augmented versions. For ***Experiment B***, it was tested how well the MLC implemen-

tation would work with a low amount of random data, also having positive mean accuracy results, with the SRE module only needing 400 training instances to achieve the very good mean accuracy results presented in the test for the best model and base classifier combination, concluding that it is possible to apply ML to both modules using low amount of training data. For ***Experiment C***, it was tested how well would the MLC implementation work with a minimal amount of specialized data, testing how well each label would be predicted for all possible answer combinations, concluding that more than half the labels, for every model, had a high probability of being correctly output. It was also possible to note that if a dataset was to be progressively build using the specialized knowledge (which represents the realistic scenario and main purpose of this work), it would probably bring better results than the ones presented in ***Experiment B***, possibly covering most realistic answer combinations.

Table 5.12: Best models and base classifier combinations for all experiments and modules.

| **Experiment** | **Module** | **Model** | **Base Classifier** |
|---|---|---|---|
| A | SRE | BR or LSP | DTC |
| A (Augmented Dataset) | SRE | BR | DTC |
| B | SRE | CC, BR or LSP | DTC |
| A | SBPG | BR orLP | DTC |
| A (Augmented Dataset) | SBPG | LP | DTC |
| B | SBPG | BR or LSP | DTC |
| C | SRE and SBPG | CC, BR, LP or LSP | DTC |

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

Currently, the IoT market is increasingly more demanding and ever-evolving, being progressively more present in a person day to day life, as well as in many companies. Due to this popularity, and the rush in which companies are partaking to develop new products that fit the IoT sphere, developing a secure product is sometimes extraneous, especially when there is not a security specialist available. Up to this date, we did not find any framework, besides SECURIoTESIGN , that can reliably and safely propose a myriad of suggestions, techniques and tests, that would help a company that cares about the cybersecurity of a product on the specific sphere of IoT, allowing to make IoT systems and software more secure and reliable, without the economic and human cost of contracting or contacting a security expert.

The proposed approach to enhance the SECURIoTESIGN project can be seen as a way to simplify future development, by implementing ML aspects, more specifically MLC, in the various modules of the SAM platform. It was possible to analyse how the platform is distributed and some caveats of each module, allowing for further knowledge about what needs to be done in order to implement the ML aspect on it. It was also executed a deep analysis of the current state of MLC, allowing to discuss and examine some of the most appropriate algorithms that could be used in the current context. In addition to the current state of MLC, the work done for the project scope was also possible to analyse, allowing to have a better understanding on what can be implemented and how the information can be gathered and used to create suitable datasets to train the models, as well as which models can be better suited for that same data. Gathering this information allowed for an early diagnostic, and provided a base to pursue this implementation.

In addition, it was possible to do a summary of the equipment and libraries used, as well as split the tests and situations into suitable *Experiments*. Having these *Experiments* allowed to see how well MLC reacts to different types of data availability, which could be important information when developing a new module with ML implementation planned upfront. Furthermore, it was explained the logic behind the ETL process developed for the creation of a dataset based on the different modules as well as how relevant data was inferred, as well as information of current elements that needed to be adapted in order to automatize the process. It was also possible to evaluate how data augmentation can be performed in the dataset in order to provide a better balance in case the questionnaire generates an imbalanced dataset, as well as techniques to create a more equilibrated train test split.

Dataset balancing and questionnaire analysis was also looked at for both modules in which

ML was decided to be implemented, these being SRE and SBPG. Each of the modules were examined in terms of different number of questions (features), recommendations (labels), answer combinations (input space), and recommendation combinations (labelsets). Furthermore, a study on how the questionnaire is built, and which questions allow for each recommendation to be generated, allowed to better understand the presence of each label. Moreover, each dataset was dissected in terms of balance and studied in terms of label correlation. By analysing the datasets balance, it was possible to conclude that the original datasets were very imbalanced, in contrast to their augmented versions. It was also possible to analyse the label correlation for each original dataset, which allowed to conclude that there is a low label correlation, that could hinder models dependent on it. In addition, it was possible to conclude that data augmentation, even though it provides advantages on dataset balancing, also makes the dataset much larger, which can be hindering in terms of training time and hardware requirements.

Finally, the results of the ML learning implementation for each module and for each *Experiment* were analysed. For *Experiment A*, which represented having all available data, the main focus was the training time, both for the original and augmented dataset, in which we could observe that the top model and base classifier combinations took around five seconds when training using the original dataset and around seven minutes when using the augmented dataset. For *Experiment B*, very positive results were achieved, specially when considering the overall dataset balance, size and amount of training data. The results for the SRE module present an 89% mean accuracy with only 400 training instances when using the best rated models, while for SBPG showed above 85% mean accuracy with also 400 instances for the best rated models. Lastly, *Experiment C* showed positive results for more than half of all labels, for each module. With these tests it was possible to conclude that using a minimal amount of data, of a specialized dataset created by an expert, can bring advantages compared to the random dataset used for *Experiment B*, showing it can be further scalable with specific answer combinations that cover most realistic premises. The best model and base classifier combinations concluded for each module can be seen in table 5.12.

To conclude, it was possible to implement ML in both the SRE and SBPG modules, using a high, low and minimal amount of data, all with positive results. This implementation can be adapted to the remaining modules of the SAM platform that use the same philosophy, and cover different talent availability within the **SECURIoTESIGN** team. Even so, there is room for improvement, which will be discussed in the following section.

## 6.2 Future Work

Although the implementation of ML presented positive results, there are still improvements that could be achieved, these being, for example:

- More modules belonging to the SAM platform should be tested and analysed, verifying if the ML implementation would also be a good fit for them;

- A better implementation of the ETL process on the SAM API should also be considered;
- Even though all answer combinations should be considered, further pruning and transformations could be performed in the datasets and evaluated to see if the results would further increase, for example, in the SBPG module, the questions which have no impact in the recommendations could be removed, and the recommendations with 100% probability of either being recommended or not could be removed from the output space;
- Additional testing and implementation of more models and base classifiers, following the evolution of the technology and the used libraries for development.

The suggestions above could further improve the work developed for this dissertation, as well as expand its effects.

# Bibliography


[BDRR00] Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-Down Induction of Clustering Trees. *Proc. 15th Intl. Conf. on Machine Learning*, 12 2000. 13

[Ben75] Jon L Bentley. A Survey of Techniques for Fixed Radius near Neighbor Searching. Technical report, Stanford University, Stanford, CA, USA, 1975. 15

[Bot12] Leon Bottou. *Stochastic Gradient Descent Tricks*, volume 7700 of *Lecture Notes in Computer Science (LNCS)*, pages 430–445. Springer, neural networks, tricks of the trade, reloaded edition, January 2012. Available from: `https://www.microsoft.com/en-us/research/publication/stochastic-gradient-tricks/`. 15

[Bre98] Leo Breiman. Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, 26(3):801 – 849, 1998. Available from: `https://doi.org/10.1214/aos/1024691079`. 14

[Bre01] Leo Breiman. Random Forests. *Machine learning*, 45(1):5–32, 2001. 15

[CBHK02] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of artificial intelligence research*, 16:321–357, 2002. 29

[CK01] Amanda Clare and Ross D. King. Knowledge discovery in multi-label phenotype data. In Luc De Raedt and Arno Siebes, editors, *In Proceedings of Principles of Data Mining and Knowledge Discovery*, pages 42–53, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. 13

[Cos21] Joana Cabral Amaral Nunes da Costa. Threat Modeling Solution for Internet of Things in a Web Based Security Framework. Master's thesis, Universidade Da Beira Interior, Oct 2021. Available from: `http://hdl.handle.net/10400.6/11849`. 8

[CS03] Koby Crammer and Yoram Singer. A Family of Additive Online Algorithms for Category Ranking. *The Journal of Machine Learning Research*, 3:1025–1058, 03 2003. 13

[DH73] Richard O. Duda and Peter E. Hart. Pattern classification and scene analysis. In *A Wiley-Interscience publication*, 1973. 15

[dMRIMF21] Vinícius de Miranda Rios, Pedro R.M. Inácio, Damien Magoni, and Mário M. Freire. Detection of Reduction-of-quality DDoS Attacks Using Fuzzy Logic and Machine Learning Algorithms. *Computer Networks*, 186:107792, 2021. Available from: `https://www.sciencedirect.com/science/article/pii/S1389128620313633`. 8

[FD13] Hasan Fleyeh and Erfan Davami. Multiclass Adaboost Based on an Ensemble of Binary AdaBoosts. *American journal of intellgent systems*, 3:57–70, 09 2013. 13

[FHMB08] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Mencia, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73:133–153, 11 2008. 13

[Fou22a] MariaDB Foundation. MariaDB Foundation, Nov 2022. Available from: `https://mariadb.org/` [cited 18-06-2022]. 9

[Fou22b] Mozilla Foundation. Privacy Not Included: A Buyer's Guide for Connected Products, 2022. Available from: `https://foundation.mozilla.org/en/privacynotincluded/` [cited 18 June 2022]. 1

[Fri01] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 15

[FS96] Yoav Freund and Robert E. Schapire. Experiments with a New Boosting Algorithm. In *In Proceedings of Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, page 148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. 14

[FS97] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. Available from: `https://www.sciencedirect.com/science/article/pii/S002200009791504X`. 14, 15

[GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely Randomized Trees. *Mach. Learn.*, 63(1):3–42, apr 2006. Available from: `https://doi.org/10.1007/s10994-006-6226-1`. 15

[IFS+21] Pedro R. M. Inácio, Mário M. Freire, Tiago M. C. Simões, Bernardo Sequeiros, Musa Samaila, Francisco Chimuco, Joana C. A. N. da Costa, and Carolina G. Lopes. Securiotesign, 2021. Available from: `http://lx.it.pt/securIoTesign/` [cited 18-06-2022]. 2, 6

[JDSTN19] Dan Bergh Johnsson, Daniel Deogun, Daniel Sawano, and Daniel Terhorst-North. *Secure by Design*. Manning, 2019. 1

[Joa21] Joana Cabral Costa. Máquina para Security Requirements Elicitation (SRE) [online]. 2021. Available from: `https://github.com/Joana-Cabral/SRE_Tests` [cited 20 Junho 2011]. xxi, 15, 18

[LCS+21] Carolina Lopes, Joana C. Costa, João B. F. Sequeiros, Tiago M. C. Simões, Mário M. Freire, and Pedro R. M. Inácio. Machine Learning Applied to Security Requirements Elicitation: Learning From Experience. *Atas do 12º Sim-*

*pósio de Informática (INForum 2021), Lisboa, Portugal*, pages 0 – 12, 2021. xxi, 2, 10, 13, 15, 17, 25, 50, 52

[LNA17] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. Available from: http://jmlr.org/papers/v18/16-365. 22

[Lop21] Carolina Galvão Lopes. SemiAutomatic Generation of Tests for Assessing Correct Integration of Security Mechanisms in the Internet of Things. Master's thesis, Universidade Da Beira Interior, Oct 2021. Available from: http://hdl.handle.net/10400.6/11844. 8

[LZC+16] Dangwei Li, Zhang Zhang, Xiaotang Chen, Haibin Ling, and Kaiqi Huang. A richly annotated dataset for pedestrian attribute recognition. *arXiv preprint arXiv:1603.07054*, 2016. 17, 50

[M.D44] Joseph Berkson M.D. Application of the Logistic Function to Bio-Assay. *Journal of the American Statistical Association*, 39(227):357–365, 1944. Available from: https://doi.org/10.1080/01621459.1944.10500699. 15

[Mer14] Dirk Merkel. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux journal*, 2014(239):2, 2014. 22

[MGCV18] Jose M. Moyano, Eva L. Gibaja, Krzysztof J. Cios, and Sebastián Ventura. Review of Ensembles of Multi-label Classifiers: Models, Experimental Study and Prospects. *Information Fusion*, 44:33–45, 2018. Available from: https://www.sciencedirect.com/science/article/pii/S1566253517307169. 10, 13, 14

[MKGD12] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. An Extensive Experimental Comparison of Methods for Multi-label Learning. *Pattern Recognition*, 45(9):3084–3104, 2012. Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011). Available from: https://www.sciencedirect.com/science/article/pii/S0031320312001203. xix, 10

[Pal22] Pallets, 2022. Available from: https://palletsprojects.com/p/flask/ [cited 18-06-2022]. 22

[pdt22] The pandas development team. pandas-dev/pandas: Pandas, 2022. Available from: https://doi.org/10.5281/zenodo.3509134. 22, 37

[Pet06] M. Petrovskiy. Paired Comparisons Method for Solving Multi-Label Learning Problem. In *In Proceedings of 2006 Sixth International Conference on Hybrid Intelligent Systems*, page 42, Los Alamitos, CA, USA, dec 2006. IEEE Computer Society. Available from: https://doi.ieeecomputersociety.org/10.1109/HIS.2006.54. 13

[PF07] Sang-Hyeun Park and Johannes Fürnkranz. Efficient Pairwise Classification. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *In Proceedings of Machine Learning: ECML 2007*, pages 658–665, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 12

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 15, 22

[Rea08] Jesse Read. A Pruned Problem Transformation Method for Multi-label Classification. In *In Proceedings of Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008)*, volume 143150, page 41, 2008. 12

[RPH08] Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label Classification Using Ensembles of Pruned Sets. In *In Proceedings of 2008 Eighth IEEE International Conference on Data Mining*, pages 995–1000, 2008. 14

[RPHF09] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier Chains for Multi-label Classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer, 2009. 11

[RPMT18] Syed Rizvi, Ryan Pipetti, Nicholas McIntyre, and Jonathan Todd. An Attack Vector for IoT Networks. In *In Proceedings of 2018 International Conference on Software Security and Assurance (ICSSA)*, pages 39–44, 2018. 1

[RSI13] Lior Rokach, Alon Schclar, and Ehud Itach. Ensemble Methods for Multi-label Classification, 2013. 14

[Ryd18] David Reinsel-John Gantz-John Rydning. The Digitization of the World from Edge to Core. *Framingham: International Data Corporation*, page 16, 2018. 1

[SCS+20] João B. F. Sequeiros, Francisco T. Chimuco, Musa G. Samaila, Mário M. Freire, and Pedro R. M. Inácio. Attack and System Modeling Applied to IoT, Cloud, and Mobile Ecosystems: Embedding Security by Design. *ACM Comput. Surv.*, 53(2), mar 2020. Available from: `https://doi.org/10.1145/3376123`. 7

[SEC21] SECURIoTESIGN Team. Internet of things scenarios for testing proposes, 2021. `https://securiotesign.di.ubi.pt/docs/IoT_Scenarios_Presentation.pdf`, Last accessed on 2022-05-30. 1, 55

[SH77] Philip H. Swain and Hans Hauska. The Decision Tree classifier: Design and Potential. *IEEE Transactions on Geoscience Electronics*, 15(3):142–147, 1977. 15

[SJS+19] Musa G. Samaila, Moser Z. V. José, João B. F. Sequeiros, Mário M. Freire, and Pedro R. M. Inácio. IoT-HarPSecA: A Framework for Facilitating the Design and Development of Secure IoT Devices. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, New York, NY, USA, 2019. Association for Computing Machinery. Available from: https://doi.org/10.1145/3339252.3340514. 8

[SK17a] P. Szymański and T. Kajdanowicz. A scikit-based Python Environment for Performing Multi-label Classification. *ArXiv e-prints*, February 2017. 15, 22

[SK17b] Piotr Szymański and Tomasz Kajdanowicz. A network perspective on stratification of multi-label data. In *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pages 22–35. PMLR, 2017. 29

[SKK16] Piotr Szymański, Tomasz Kajdanowicz, and Kristian Kersting. How Is a Data-Driven Approach Better than Random Choice in Label Space Division for Multi-Label Classification? *Entropy*, 18(8), 2016. Available from: http://www.mdpi.com/1099-4300/18/8/282. 15

[SLA+20] Musa G. Samaila, Carolina Lopes, Édi Aires, João B. F. Sequeiros, Tiago Simões, Mário M. Freire, and Pedro R. M. Inácio. A Preliminary Evaluation of the SRE and SBPG Components of the IoT-HarPSecA Framework. In *2020 Global Internet of Things Summit (GIoTS)*, pages 1–7, 2020. 7

[SLA+21] Musa Samaila, Carolina Lopes, Édi Aires, Bernardo Sequeiros, Tiago Simões, Mário Freire, and Pedro Inácio. Performance Evaluation of the SRE and SBPG Components of the IoT Hardware Platform Security Advisor Framework. *Computer Networks*, 199:108496, 09 2021. xxi, 7, 33, 35

[SNF+18] Musa G. Samaila, Miguel Neto, Diogo A. B. Fernandes, Mário M. Freire, and Pedro R. M. Inácio. Challenges of Securing Internet of Things devices: A Survey. *Security and Privacy*, 1(2):e20, 2018. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/spy2.20. 8

[SS00] Robert Schapire and Yoram Singer. BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning - ML*, 39:135–168, 05 2000. 13, 14

[SSC+17] Musa G Samaila, João BF Sequeiros, Acácio FPP Correia, Mário M Freire, and Pedro RM Inácio. A Quick Perspective on the Current State of IoT Security: A Survey. In *Networks of the Future*, pages 431–464. Chapman and Hall/CRC, 2017. 8

[SSFI18] Musa G. Samaila, João B. F. Sequeiros, Mário M. Freire, and Pedro R. M. Inácio. Security Threats and Possible Countermeasures in IoT Applications Covering Different Industry Domains. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ARES 2018, New York, NY, USA, 2018. Association for Computing Machinery. Available from: https://doi.org/10.1145/3230833.3232800. 8

[SSS+20] Musa G. Samaila, João B. F. Sequeiros, Tiago Simões, Mário M. Freire, and Pedro R. M. Inácio. IoT-HarPSecA: A Framework and Roadmap for Secure Design and Development of Devices and Applications in the IoT Space. *IEEE Access*, 8:16462–16494, 2020. xxi, 8, 33, 35

[STV11] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 145–158. Springer, 2011. 29

[TC22] The Python Core Team and Community. Python Enhancement Proposals, 2022. Available from: https://peps.python.org/pep-0000/ [cited 18-06-2022]. 22

[TKV08] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *In Proceedings of Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, volume 21, pages 53–59, 2008. 12

[TKV11] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-Labelsets for Multilabel Classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2011. 14

[WKWJ21] Ran Wang, Sam Kwong, Xu Wang, and Yuheng Jia. Active k-labelsets Ensemble for Multi-label Classification. *Pattern Recognition*, 109:107583, 2021. Available from: https://www.sciencedirect.com/science/article/pii/S0031320320303861. 14

[WM10] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. 22, 37

[ZLLG17] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary Relevance for Multi-label learning: an Overview. *Frontiers of Computer Science*, 12, 11 2017. 11

[ZZ05] Min-Ling Zhang and Zhi-Hua Zhou. A k-nearest Neighbor Based Algorithm for Multi-label Classification. In *In Proceedings of 2005 IEEE international conference on granular computing*, volume 2, pages 718–721. IEEE, 2005. 13

[ZZ06] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 18:1338– 1351, 11 2006. 13