UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Cyber vulnerability exploitability classifier

Théo Fuhrmann

February 2022

Thesis director: René Serral Gracià
GEP tutor: Joan Sardà
Computer Science Degree
Specialisation: Computation

Contents

1 Context **6**

1.1 Introduction 6

1.2 Cyberattacks 6

1.3 Problem 6

1.4 Stakeholders 7

2 Justification **8**

2.1 Related Work 9

3 Scope and Obstacles **10**

3.1 Objectives 10

3.2 Requirements 10

3.3 Potential Constraints, Obstacles, and Risks 10

4 Methodology and Follow-up **12**

4.1 Methodology 12

4.2 Monitoring 12

5 Description of Tasks **13**

5.1 Task Identification 13

5.1.1 Project's Documentation (D) 13

5.1.2 Classifier's Development (C) 14

5.1.3 Project's Monitoring (M) 15

5.2 Task Resources 15

5.3 Task Summary 16

6 Gantt Chart **17**

7 Risk Management: Alternative Plans and Obstacles **18**

8 Budget **19**

8.1 Identification of Costs 19

8.1.1 Staff Costs 19

8.1.2 Development Costs 20

8.1.3 Contingencies and Incidents 21

8.2 Cost Estimates 22

8.3 Management Control 22

9 Sustainability Report **23**

9.1 Environmental Impact 23

9.2 Economic Impact 24

9.3 Social Impact 24

10 Data Selection **25**

11 Preprocessing **27**

11.1 Label Feature Extraction 27

11.1.1 Analysing the Data 28

11.1.2 Dropping Outliers 28

11.2 Feature Extraction 30

11.2.1 Extracting Numerical Features 30

11.3 Natural Language Processing Techniques 30

11.3.1 Bag of Words + TF-IDF 30

11.3.2 Doc2Vec 32
11.4 Feature Selection 34
11.4.1 Mutual Information 35
11.4.2 ANOVA F-Value 35
11.4.3 Conclusion 36

12 Model Selection **37**

12.1 Linear Support Vector Machines 37
12.2 Random Forest 38
12.3 Multinomial Naive Bayes 39
12.4 AdaBoost 39
12.5 Rejected Classifiers 40

13 Model Implementation **41**

13.1 Data Splitting 41
13.2 Hyperparameter Tuning 41
13.3 Resampling 42
13.4 Scaling 42
13.5 Training Analysis 43
13.6 Choosing the Number of Dimensions for the Models & Learning curves 44
13.7 Overcoming the Overfitting of Random Forest 47
13.7.1 Oversampling and Undersampling 47
13.7.2 Hyperparameter Tuning 48
13.7.3 Conclusion 50
13.8 Choosing the Best Model 50
13.8.1 Interpretability Analysis 53

13.8.2 Comparison with CVSS Scoring 55

14 Conclusions **56**

A Charts **60**

B Code **64**

B.1 MITRE Crawler 64

B.2 JSON payload of a CVE obtained with the NVD API 64

B.3 Python code notebook to obtain the data through the NVD API 67

B.4 Python code notebook to train the Doc2Vec model 67

1 Context

1.1 Introduction

This thesis project concludes my Computer Science degree at the Universitat Politècnica de Catalunya (UPC). Focused on the computation specialisation, it forms part of a larger project conducted by my thesis director, René Serral, an associate professor in the Department of Computer Architecture at the UPC.

The larger project is a software tool designed to help companies analyse their cybersecurity by performing attacks on their own infrastructures. The end goal is to reinforce cybersecurity for companies by increasing awareness of and decreasing exposure to cyberthreats.

As part of the larger project, this one focuses on a Machine Learning predictor capable of detecting whether or not a cybervulnerability may be exploited.

1.2 Cyberattacks

For contextualisation, this project begins by formally defining cyberattacks.

A cyberattack is a malicious attempt at targeting computers or networks to disable them, steal data, or access unauthorised system areas. Through a plethora of methods such as malware, denial of service, and ransomware, a cyberattack can be performed by any individual or group. The aforementioned methods **exploit** cybervulnerabilities as gateways to achieve the objectives of a cyberattack.

The damage that a cyberattack can generate is broad, ranging from stealing credit card credentials to compromising the data of a whole nation. Therefore, over the years [1], as cybersecurity has become more sophisticated, a variety of tools combating cybercrime have been introduced within the field. Ultimately, this project contributes to the protection against the ever-growing threat of cybercrime.

1.3 Problem

As we descend into the digital era, technological progress expands at an exponential rate. The unprecedented discoveries of new technologies are continually integrated into society, increasing dependencies on cybersecurity softwares [2]. With the advent of electronic transactions, social media, and internet marketplaces, it is customary to divulge valuable and personal information. Reframing

global interactions ranging from politics and economics to the sciences and education, all fields have become inevitably dependent on cybersecurity.

Alternatively, having easily exploited the digital revolution, cybercriminals have capitalised upon other new technologies that abet all sorts of cyberattacks, threatening the cybersecurity of such spaces [3]. Consequently, the sudden rise of cyberattacks has affected individuals, organisations, and countries alike. Therefore, this project concentrates on the companies struggling with cyberrisk management [4].

1.4 Stakeholders

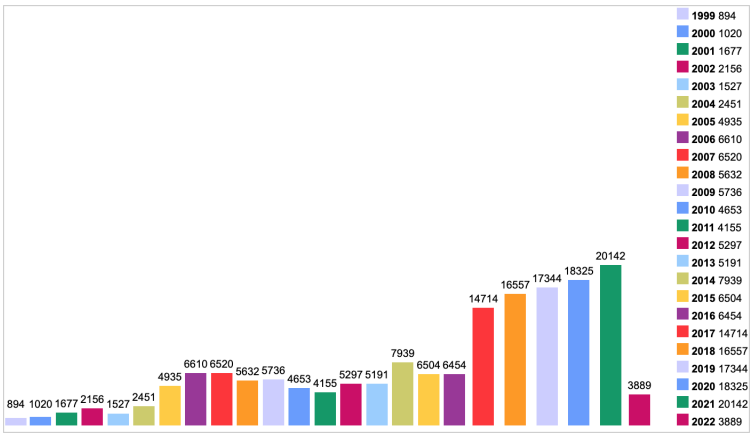
As previously stated, this project is part of a larger whole, with René Serral as the main stakeholder.

Though each individual student project remains independent, all contribute to the realisation of Serral's larger vision. Because the five students involved work to assemble a collective project, each one is also a stakeholder.

The target users for this project are companies who want to improve and streamline cybersecurity on their computer systems. Thus, if Serral's project reaches a stage of completion wherein users can benefit from improved cybersecurity, target users can be considered indirect stakeholders.

2 Justification

The number of detected cybervulnerabilites increases each year (Figure 1). Therefore, to ensure that companies have safe digital environments, cybersecurity protocols must constantly be updated to avoid emerging cyberthreats.



Source: https://www.cvedetails.com/browse-by-date.php
Figure 1: Classified vulnerabilities by year on the MITRE database

Designed to assist companies in running penetration tests (pentests), this software tool simulates cyberattack automation to detect security flaws within a system. Originally, the main task of this project was to identify and provide relevant CVEs[1] to facilitate subsequent treatment of CVE data and effectively analyse cybersecurity systems. However, after investigating the state of the art, the focus of this project shifted from the collection of CVE data to cybervulnerability prediction.

Both cybervulnerability prediction and cybersecurity management depends greatly on time. When companies detect vulnerabilities in their softwares, they race to release patch updates before such vulnerabilities are targeted and exploited in cyberattacks. Furthermore, because software vendors may face several simultaneous vulnerabilities, addressing them all may consume excess resources. Consequently, prioritizing vulnerabilities based on the likelihood of their exploitation can minimize the damages of potential cyberattacks. Since the most popular system of cybervulnerability severity does not have consistent results (see Section 13.8.2), a Machine Learning (ML) model capable of predicting the likelihood of cybervulnerability exploitation was developed to help companies improve such predictions.

[1]The Common Vulnerabilities and Exposures system is a list of publicly known cybersecurity flaws. Every cybersecurity flaw on the list has a CVE ID number assigned. CVEs can help cybersecurity professionals address these vulnerabilities and make computer systems safer.

2.1 Related Work

A large section of cybersecurity research has focused on the detection of new cyberthreats. In recent years many papers have been published with differing approaches.

In 2010, M. Bozorgi et al. [5] developed a predictor using a linear Support Vector Machine classifier based on publicly disclosed vulnerabilities from OSVDB and the MITRE database.

Five years later, C. Sabottkle et al. [6] introduced an exploitability predictor based on vulnerabilities disclosed from Twitter. They used data from ExploitDB, OSVDB, Microsoft security advisories and Symantec as ground truth.

In 2018, using word embedding techniques, N. Tavabi et al. [7] tackled the problem of cyberthreats by gathering data from messages posted on darkweb and deepweb sites and experimented with SVMs and Random Forest models. They used data from Symantec and Metasploit as ground truth.

Another approach was developed in 2021 by O. Suciu et al. [8] where they proposed a method to predict exploitability based on the lifespan of the vulnerability, claiming that the precision of their model substantially improves over time.

3 Scope and Obstacles

3.1 Objectives

The main goal of this project is to design, implement, and analyse a ML model that classifies CVEs as exploitable or non-exploitable. These classifications extract new information that automate cyberattacks for penetration testing. To construct this ML model, the following objectives must be completed.

1. Research different machine learning techniques that can be applied to the classifier model.
2. Analyse, implement, and compare the different machine learning techniques (Support Vector Machines, Random Forest, Doc2Vec, Naive Bayes, and others).
3. Find the classifier with the highest performance metrics used (precision, recall, ROC-AUC, etc.).

This classifier should be able to outperform the most used vulnerability severity score system currently [9].

3.2 Requirements

The following requirements are meant as a guide to accomplish the aforementioned objectives:

- The classifier must minimise misclassifications.
- All code sections must be well integrated among themselves.
- All code must be comprehensive and maintainable.
- Workflow must be consistent throughout the project.

3.3 Potential Constraints, Obstacles, and Risks

Awareness of potential constraints, obstacles, and risks is critical for workflow management. In relation to this project, the following have been identified.

- **Deadlines:** Time constraints caused by deadlines may lead to unfinished project tasks. Therefore, the final product may be affected by the time involved in its development.
- **Inexperience:** The field of cybersecurity is novel. Furthermore, since new cybersecurity methods are always in development, constant research is crucial for the progression of this project. However, because of the time commitment that research requires, repeatedly doing so may cause some delays.
- **Unknown roadblocks:** As in any development timeline, unforeseen obstacles may arise. Thus, improvised solutions may affect the end goals of this project.
- **Unrealistic results:** This project does not contain the capacity to test its final product on a large user base. Thus, the true effectiveness of the project deliverable cannot be gauged, potentially misleading its conclusions.

4 Methodology and Follow-up

4.1 Methodology

This project utilizes the Kanban methodology, which streamlines workflow management through a series of cards and columns that represent task and task status labels, respectively. Because many larger tasks must be subdivided and grouped by completion status, the Kanban methodology was chosen for its practicality and scope.

Given that multiple collaborators are contributing individual projects to culminate in the larger software tool, a program must be used to facilitate team collaboration and discussion. The program Trello [10] was chosen due to its layout and consolidation of both personal and collective tasks.

Furthermore, all code will be uploaded to a team GitLab [11] repository. This platform will be used to share all code and build the project simultaneously. GitLab will act as a version control tool to avoid incompatibilities between respective works, comfortably integrating project code.

4.2 Monitoring

To monitor the progress of this project and fix contingencies, weekly team meetings, either in-person or online through Google Meet, will be headed by the project director. By consistently attending meetings, awareness of team progress will increase. Furthermore, breakthroughs and solutions to roadblocks will be better enabled through scheduled group discussions.

To ensure consistent team communication, this project utilizes Discord [12], an instant messaging and digital distribution platform that facilitates communication through text, voice calls, and video calls in either private chats or communities called servers. By using a Discord server on this project, team communication will not be restricted to weekly meetings.

5 Description of Tasks

The development of this project began on February 1st. Accounting for the defense of this thesis project, which is scheduled between June 27th and July 1st, the final report and presentation of this thesis should be ready by June 19th. This week of margin accounts for any unexpected obstacles or tasks. Therefore, according to the timeline specified above, this project should be completed in 139 days, or 20 weeks. Working 20 hour per week, except during the Holy Week, this project will take about 410 hours in total.

5.1 Task Identification

This section will define the two major tasks to be completed during the development of the project simultaneously: the documentation of the whole project (D) and the development of the classifier (C). Besides these two major tasks, simultaneous monitoring through weekly team meetings will be considered an additional task (M).

5.1.1 Project's Documentation (D)

The task of project documentation is subdivided into three main blocks:

- **(DG) GEP documentation:** GEP documentation encompasses all content for the GEP subject. It involves all documentation during the first stage of the project, which includes project planning and structuring to maintain direction throughout its completion. This block can be further subdivided into the four GEP deliveries:
  - **(DG1) Contextualisation and scope:** Write the first GEP delivery, which contains the introduction, context, justification, scope, and methodology of the project.
  - **(DG2) Time planning:** Write the second GEP delivery, which contains the description of tasks, the Gantt chart, and the risk management of the project.
  - **(DG3) Budget and sustainability:** Write the third GEP delivery, which contains cost identification and estimates of the project, as well as its sustainability report.
  - **(DG4) Document's final integration:** Merge the previous three deliveries and make the appropriate modifications to make the entire deliverable cohesive and coherent.

- **(DF) Final document:** Final documentation encompasses project results and observations. Additionally, because certain project objectives may be modified, this task also includes the refitting of DG.
- **(DK) Keynote:** The keynote encompasses the preparation of the presentation for the defence of this thesis. The keynote will summarise the whole project and emphasise its most important parts while also making the keynote itself dynamic and enjoyable.

5.1.2 Classifier's Development (C)

Classifier development is subdivided into different tasks related to the stages of classifier development:

- **(CF) Classifier familiarisation:** By discussing both the classifier and approaches to the whole project with the director and the team, a level of consistency can be maintained throughout the entirety of the project. This task includes research on the remaining topics involved, such as the cybervulnerability community and the state of the art.
- **(CD) Classifer's data:** It includes all the tasks related to the data used to feed the machine learning models:
  - **(CD1) Researching databases:** Databases available and suitable for the problem must be found.
  - **(CD2) Obtaining the data:** A method to obtain the data from the chosen databases must be developed (crawlers[2], API calls, downloads, etc.).
  - **(CD3) Preprocessing:** Coding types will be fixed, outliers treated, and features extracted (Natural Language Processing techniques).
  - **(CD4) Analysis:** The data will be visualised to understand it better.
- **CML) Classifier's Machine learning:** It encompasses all the tasks related to the development of the machine learning models. These can be broken down into:
  - **(CML1) Research different models:** Since there are so many machine learning techniques, the techniques must be analysed and the

[2] A crawler is an internet bot that browses the world wide web, and it can search multiple types of information, such as websites, RSS feeds, or email addresses. It looks for information on the Web that is then categorised, indexed, and catalogued so that this crawled information is retrievable and can be evaluated. The objective of a crawler is to create indexes so that search engines can use them to return matching information on search queries, but it can also be used for other goals, such as data mining. [13]

ones most compatible with the exploitability classification must be chosen. Both supervised learning and unsupervised learning methods will be attempted.

- **(CML2) Models implementation:** Parameters of the chosen ML models will be programmed and tweaked to maximise their performance.
- **(CML3) Filter and choose a model:** Models will be tested and the model that outperforms the rest based on the chosen metrics will be kept.
- **(CML4) Interpretability analysis:** Models' behaviour will be studied to gain more understanding.

5.1.3 Project's Monitoring (M)

This task consists of all weekly team meetings attended throughout the entirety of the project, as well as supplementary project communication (i.e. Discord, additional meetings, etc.).

5.2 Task Resources

To complete this project, the following human and material resources are required:

- **Human resources:**
  - Project director (D)
  - Team members (T): Adrià Pagés, Alexis Peralta, Oriol Ortiz, and Valentina Dumitrasc.
- **Material resources:**
  - (P) Planning: Atenea, Trello, Notion, Racó, and GitLab
  - (M) Monitoring: University room, Google Meet, and Discord
  - (L) Learning: Google, Google Scholar, and YouTube
  - (C) Coding: Jupyter Notebooks, Visual Studio Code, Google Colab, and Vim
  - (DO) Documentation: Overleaf, Ganttproject, and Keynote

  Additionally, throughout the entirety of this project, my personal computer, a 2014 Macbook Pro with 8GB of RAM and a 2.6GHz Dual-Core Intel Core i5 processor, will be used.

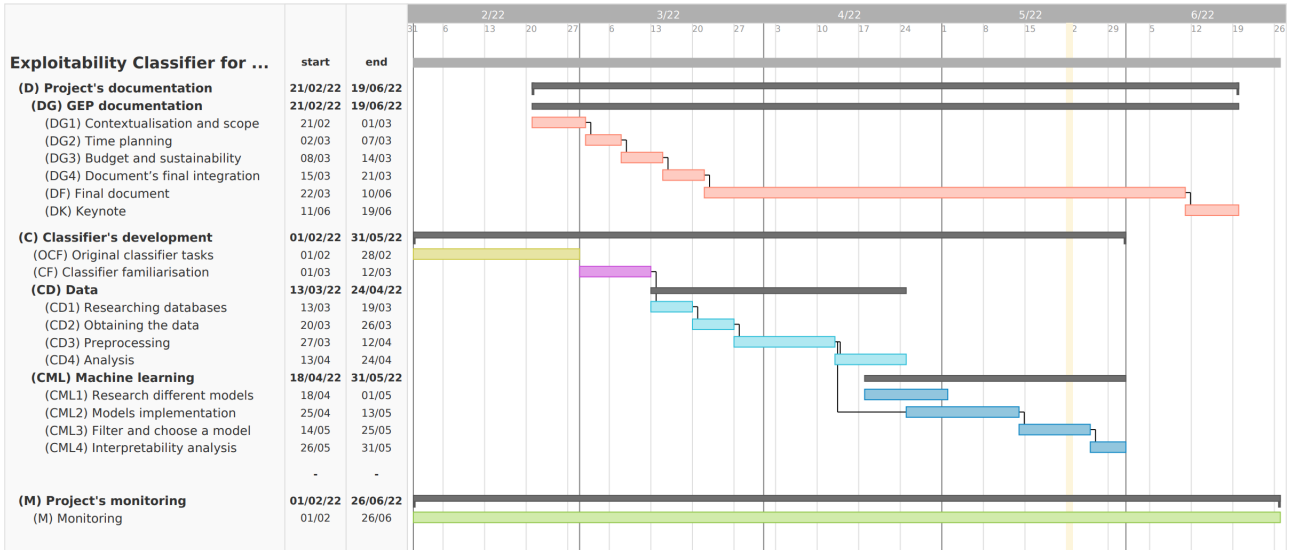
5.3 Task Summary

ID	Task	Time	Dependencies	Resources
DG1	Contextualisation and scope	15h	-	P, M, L, DO
DG2	Time planning	10h	DG1	P, M, L, DO
DG3	Budget and sustainability	10h	DG2	P, M, L, DO
DG4	Document's final integration	15h	DG3	P, M, L, DO
DF	Final document	80h	DG4	P, M, L, DO, D
DK	Keynote	30h	DF	P, M, L, DO
CF	Classifier familiarisation	20h	-	P, C, M, L
CD1	Researching databases	20h	CF	P, C, M, L
CD2	Obtaining the data	20h	CD1	P, C, M, L
CD3	Preprocessing	60h	CD2	P, C, M, L
CD4	Analysis	20h	CD3	P, C, M, L
CML1	Research different models	20h	CF	P, C, M, L
CML2	Models implementation	60h	CD3	P, C, M, L
CML3	Filter and choose a model	20h	CML2	P, C, M, L
CML4	Interpretability analysis	10h	CML3	P, C, M, L
M	Monitoring	30h	-	M, D, T
Total nº of hours: 410				

Table 1: Task summary

6 Gantt Chart

The following chart visualizes the distribution of tasks throughout the timeline set of the project. Moreover, this chart outlines task dependencies described in Section 5.3.



Exploitability Classifier for ...	start	end
(D) Project's documentation	**21/02/22**	**19/06/22**
(DG) GEP documentation	**21/02/22**	**19/06/22**
(DG1) Contextualisation and scope	21/02	01/03
(DG2) Time planning	02/03	07/03
(DG3) Budget and sustainability	08/03	14/03
(DG4) Document's final integration	15/03	21/03
(DF) Final document	22/03	10/06
(DK) Keynote	11/06	19/06
(C) Classifier's development	**01/02/22**	**31/05/22**
(OCF) Original classifier tasks	01/02	28/02
(CF) Classifier familiarisation	01/03	12/03
(CD) Data	**13/03/22**	**24/04/22**
(CD1) Researching databases	13/03	19/03
(CD2) Obtaining the data	20/03	26/03
(CD3) Preprocessing	27/03	12/04
(CD4) Analysis	13/04	24/04
(CML) Machine learning	**18/04/22**	**31/05/22**
(CML1) Research different models	18/04	01/05
(CML2) Models implementation	25/04	13/05
(CML3) Filter and choose a model	14/05	25/05
(CML4) Interpretability analysis	26/05	31/05
	-	-
(M) Project's monitoring	**01/02/22**	**26/06/22**
(M) Monitoring	01/02	26/06

Source: Own compilation, made with Teamgantt
Figure 2: Gantt chart

The old Gantt chart can be found in Figure 26 on page 62

7 Risk Management: Alternative Plans and Obstacles

To minimise the impact of risks and obstacles as mentioned in section 3.3, the following items are alternative plans for potential exposures.

- **Deadline (Impact: medium):** If time becomes a considerable constraint, tasks will be rescheduled. Furthermore, the number of weekly working hours will be increased from 20 to 25 or 30. The number of hours increased would consider a margin of error to avoid the similar future risks.
- **Inexperience (Impact: low):** Unfamiliarity with certain concepts or code will be mitigated through increased research time and frequency (tasks CF, CML1 and CD1). Though a set number of hours has already been reserved for project research, more research time may be required than previously anticipated. Thus, as additional hours incur, other tasks will be rescheduled to maintain project management.
- **Unknown roadblocks (Impact: high):** Unexpected obstacles may acutely hinder task completion (mainly tasks CML2 and CD2). To mitigate negative effects, obstacles will be communicated to both the project director and team members. Appropriate decisions will be made as a collective. Such decisions may involve shifting the direction of the project, rescheduling affected tasks, and documenting obstacles. All decisions are subject to the severeness of an obstacle.
- **Unrealistic results (Impact: medium):** Due to small sample size, in the case of unrealistic results, a new method of testing will be designed to obtain more realistic and meaningful results. New developed testing methods will consist of problem documentation, additional task creation, design and implementation of the new test, and rescheduling of affected dependent tasks.

Because working hours and electricity usage may increase, alternative plans may dip into staff and development contingency budgets. Additionally, rescheduling old tasks may incur use of the contingency week.

All of the alternative plans would imply using some of the staff and development contingency budgets since the working hours and the electricity usage would increase. On the other hand, the rescheduling of the tasks on every plan could lead to the use of the remaining week before the deadline if needed.

8 Budget

8.1 Identification of Costs

The following section discusses the economic costs of this project. Costs are divided into staff costs and development costs. Moreover, contingencies and incidents will also be considered and reviewed.

8.1.1 Staff Costs

This project involves the following personnel:

- **Author:** The author will play the roles of Project Manager (PM), Junior Data Scientist (DS), Junior Software Developer (SD), and Analyst (A).
- **Project Director:** The project director will be responsible for the direction of this project and the tool's development (PD).
- **Team:** Other team members will play the role of Junior Software Developers (TSD) during weekly meetings.

The following table shows annual salaries for every role. The costs per hour of each respective role are calculated by dividing the annual salary (net annual salary plus 1.30% of social charges) by the number of hours worked annually in Spain [14]

Role	Net Annual salary (€)	Annual salary (€)	Cost (€)/h
Project Director	60717 [15]	61506.32	36.44
Project Manager	41000 [16]	41533.00	24.60
Junior Data Scientist	27569 [17]	27927.40	16.54
Junior Software Developer	21029 [18]	21302.38	12.62
Analyst	26753 [19]	27100.79	16.05

Table 2: Role salaries and costs per hour

The following table calculates the cost per task considering annual salaries.

Task	Role	Hours	Cost (€)
DG1	PM	15h	369.00
DG2	PM	10h	246.00
DG3	PM	10h	246.00
DG4	PM	15h	369.00
DF	PM	60h	1476.00
DF	A	20h	321.00
DK	PM	30h	738.00
CF	DS	20h	330.80
CD1	DS	20h	330.80
CD2	DS	20h	330.80
CD3	DS	60h	992.40
CD4	DS	20h	330.80
CML1	DS	20h	330.80
CML2	DS	60h	992.40
CML3	DS	20h	330.80
CML4	DS	20h	330.80
M	TSD (x4)	30h	1514.40
M	PM	30h	738.00
M	PD	30h	1093.20
Total cost: 11411.00 €			

Table 3: Costs per task

8.1.2 Development Costs

Because all software used for this project is open source, the only additional cost of this thesis project is the computer used to develop and document it.

To calculate the hardware amortization for this project, the following formula will be used.

$$price \times \frac{1}{lifeExpectancy} \times \frac{1}{hoursWorked} \times hoursUsed$$

A MacBook Pro will be used during the whole project's development. Used for three hours per day on average, with an original price of 1250€, and a life expectancy of 10 years, the hardware amortization will be 113.91€.

Given the global pandemic, work will be at home, so there are no additional costs for the rent of a working space. However, to increment the precision of the budget report, internet and electricity costs will be taken into account.

- **Internet:** The cost of internet at home is 45€ per month. Because it will be used for 5 months, the final cost of internet is 225€.
- **Electricity:** The average cost of electricity this year is 0,211 €/kWh [20]. MacBooks have an average power consumption of 0.061 kWh. Because this computer will be used for 410 hours throughout the duration of the project, the final cost of electricity is 5.27€.

The overall development costs are the following.

Resource	Cost (€)
Hardware	113.91
Internet	225
Electricity	5.27
Total: 344.18 €	

Table 4: Development costs

8.1.3 Contingencies and Incidents

A 15% contingency margin will be kept on the overall staff costs since they are prone to change overtime. New tasks may appear in the future or some set tasks may take longer than expected, as mentioned in *Risk management: alternative plans and obstacles*.

The development costs will take a 7.5% contingency margin since they partially depend on the project's execution time but aren't as volatile as the staff costs. The only cost correlated to the staff is the electricity.

Finally, the risks and obstacles mentioned in *Risk Management* have been considered in the earlier contingency margins. Thus we will not include additional cost percentages.

8.2 Cost Estimates

The total predicted cost for the project is calculated in the following table.

Category	Cost (€)	Contingency %	Total (€)
Staff	11411.00	15	13122.65
Development	344.18	7.5	369.99
Total: 13492.64 €			

Table 5: Project's predicted cost

8.3 Management Control

To monitor the possible budget deviations, the task execution times will be logged in real time so that they can be compared to the original predicted task execution times and the difference between both times can be calculated. This difference

$$\sum_{i=1}^{n}(TaskPredictedTime_i - TaskRealTime_i) \times TaskCost_i$$

, where n symbolises the current number of completed tasks, will be used to calculate the budget deviation in real time: If the difference is positive it will indicate an underestimation of the time and budget, while a negative difference would imply the opposite, indicating that the proportional contingency costs should be assigned to the error.

The difference between the predicted costs and the real costs will be used for the development costs, following the same logic used for the tasks.

9 Sustainability Report

With the advent of the Industrial Revolution, the technological industry brought extraordinary economic growth to the globe. Blinded by technological advancements and their benefits to the market, industry leaders disregarded the compromised living conditions and communities they left in their wake. It took about three generations for leaders to make the first concerted effort at bringing pollution under control, and society has lagged behind the social, economical, and environmental consequences of this ever-growing industry since then.

The problem is that Earth is headed towards an inflection point that could leave it incurable, with most of its inhabitants wiped out, humanity included.

Over the course of my Computer Science degree at the UPC there have been several attempts at giving students a bigger picture of the sustainability scene in the technological field. Talks about the environmental, social, and economical impacts of the technological industry increased students' awareness of both current and potential future issues.

The dedicated sustainability section in this Thesis serves as a reminder that every action has a consequence and that one of the goals of this project is to minimise its environmental, economical, and social impact by making the most sustainable decisions during its planning and execution. The report will be a great way to put me in a sustainable mindset that will not only affect the way of carrying out this Thesis, but the future projects that I will carry throughout my professional career and personal life.

9.1 Environmental Impact

No materials have been bought for the purpose of this project, so already-owned hardware will be employed for its development. Additionally, the electricity used throughout the project will not be of a significant quantity, as it will only involve a laptop.

The cybersecurity tool that this project takes part of will require the user to have a computer, internet, and electricity. The cloud will be used to store all necessary information for the platform's functionality. Using the cloud, rather than traditional servers, will result in a much smaller environmental footprint.

The software developed will have a reduced ecological impact, considering:

- The small amount of low-impact resources used for its development.
- It is emission-free and does not require the extraction of natural resources.

- Users will not need to employ any environmentally-harmful item to use it.

9.2 Economic Impact

The costs of the project have been evaluated in the *Budget* section, and the economic impact they entail is relatively low. There are no irrelevant costs in the evaluation, as every resource taken into account is essential for the project.

The cybersecurity tool only requires that the user have a device to access it, avoiding the need to purchase additional hardware, as most people already own such a device. Since it will not require economic input from the user, this tool has a significantly low economic impact.

Finally, maintaining this project will not be an obstacle economically since open source code will be used for much of its development, and it will use cloud computing for cost-saving scalability.

9.3 Social Impact

This tool aims to help companies improve their cybersecurity without requiring them to invest any money. It is a great way for companies to combat cybercrime, a widely known problem that currently takes advantage of unprotected individuals and collectives.

This tool could have a very positive social impact, since it will be running on open source code, paving the way for a new cybersecurity community. On the other hand, making the tool free for users could encourage them to use it without any compromise.

The parties damaged by this tool are its competitors and cybercriminals. The tool's social impact will be determined by its success, so there is no way to accurately predict how much it could affect society currently.

10 Data Selection

A main data source used in previous studies was the Open Source Vulnerability Database (OSVDB). However, after this popular database was shut down in 2016 [21], information on the exploit status of cybervulnerabilities can no longer be accessed through it.

Other sources used in past studies are distributed by private, commercial companies. These sources include Secunia's vulnerability manager, which was bought by Flexera [22]; Symantec's DeepSight [23]; IBM X-Force Exchange [24]; and Accenture's BugTraq [25].

As previously stated (see page 8), the original approach to this project planned on collecting and using data from MITRE's CVE Database [26], which contained all the registered CVEs. Initially, CVE data for this project was scraped from this database with a python crawler (see section B.1 on page 64) developed using Scrapy [27].

However, MITRE's CVE Database did not provide enough information to feed a ML algorithm. Thus, data collection switched to the National Vulnerability Database (NVD) [28], a U.S. government repository of cybervulnerabilities. This repository builds upon MITRE's CVE Database, providing other relevant information, such as respective CWEs and affected software versions (to see all fields of a retrieved CVE, see section B.2 on page 64). All data for this project were then fetched through NVD's public API [29] (see Figure 28 for a full example on page 67). The following data were kept as JSON.

Feature name	**Description**
`assigner`	Entity that assigned the CVE
`cwes`	CWEs associated to the CVE
`references`	Relevant links to the CVE, they can provide additional information about the vulnerability. Every entry has at least one reference.
`description`	Short description of what the vulnerability is, it usually contains the impact, the component, the attack vector and if there's any current fix.
`cpes`	CPE products associated with the vulnerability.
`publishedDate`	Published date of the entry
`lastModifiedDate`	Last modifed date of the entry

Table 6: Selected features from the dataset

The JSON data also contain their respective CVSS[3] with all subscores, including exploitability. Though previous studies have included the CVSS base score to train their models [5] [7], this feature was not included in this project for the following reasons.

- **Timeliness:** CVSS base scores are not immediately available in the NVD. Rather, these scores are manually calculated by analysts and may take many days until published online. Without this information, affected companies may underestimate the exploitability of a software vulnerability. Thus, the wait time for the calculation of this score may enable potential hackers. Consequently, since new entries cannot be classified until the CVSS base score is available, creating a dependency in the cybervulnerability predictor of this project.
- **Bias:** Premeditated scores, such as CVSS base scores, should not be used in calculating the exploitation of a vulnerability. Doing so may bias the predictor because CVSS base scores are correlated with the exploitability subscore of an entry.
- **Conjecture:** Though metrics used to calculate CVSS base scores have been widely studied and tested, manual calculation of these scores may incur human error, potentially misleading our model during training.

[3]The Common Vulnerability Scoring System (CVSS) allows generating a score for a vulnerability based on a selection of the vulnerability's characteristics to reflect its severity.

11 Preprocessing

Due to its malleability and compatibility with other ML libraries, the popular Dataframe data structure from the Python library Pandas [30] was chosen for data preprocessing, analysis, and manipulation. Furthermore, the two-dimensional tabular format of the Dataframe structure also improves data readability and organisation.

All JSON data obtained using the NVD API were transformed into a Dataframe. However, the unprocessed data were a multidimensional nested structure containing arrays within certain fields. To fit this data into a Dataframe, the JSON had to be normalised. This procedure flattened the data to the first dimension, keeping nested JSON data as text in the Dataframe.

Most features needed to have their information extracted from the remaining JSON encoded strings to have usable data. To do so, only values from the JSON structures that represented each feature were kept.

11.1 Label Feature Extraction

To categorise the type of information each reference contains (see listing 1), the NVD assigns tags to reference links. The current tags are: 'Mailing List', 'Third Party Advisory', 'Patch', 'Exploit', 'Product', 'Vendor Advisory', 'Issue Tracking' and 'VDB Entry'.

```
{
    "url": "http://packetstormsecurity.com/files/166559/IdeaRE-
        RefTree-Shell-Upload.html",
    "name": "http://packetstormsecurity.com/files/166559/IdeaRE-
        RefTree-Shell-Upload.html",
    "refsource": "MISC",
    "tags": ["Exploit", "Third Party Advisory", "VDB Entry"]
}
```

Listing 1: Example of an NVD reference

The 'Exploit' tag was assigned to a link when it referenced an exploit of a vulnerability, indicating that the respective vulnerability had an exploit. Thus, to create the label feature that classifies a vulnerability into exploitable or non-exploitable, every entry was checked for this tag in one of its references. If this tag was present, the entry was considered to have an exploit and the label feature was set to 1; else, if the tag was not present, the value was set to 0. The label feature was named `hasExploit`.

11.1.1 Analysing the Data

Before continuing with feature extraction, the label feature was analysed to better understand the collected data. First, the ratio of exploited vulnerabilities to not exploited ones was plotted with a pie chart (see Figure 3). A clear data imbalance was noticed between the two classes. Because training ML classifiers with an imbalanced dataset can lead to very bad performances, this caused problems with the accuracy of predicting the minority class.

The following example explains the issue outlined above. Given a dataset with 100 entries having 95 negative labels and 5 positive labels, if a classifier predicts that 100% of the dataset is negative, it will have achieved a 95% percent accuracy score. However, it will have failed to predict 100% of the minority class. Thus, this imbalance leads to biassed models.

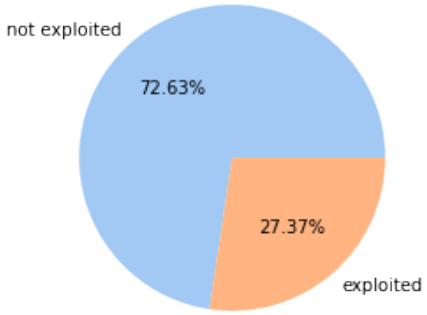


Figure 3

To address the issue of imbalance, multiple techniques were used for the model selection phase (see page 13.7).

11.1.2 Dropping Outliers

After plotting the number of NVD classified vulnerabilities since 1988 (see Figure 4), most vulnerabilities were observed to have been classified after the year 1999. Moreover, vulnerabilities classified before 2000 represent 0.39% of the dataset.

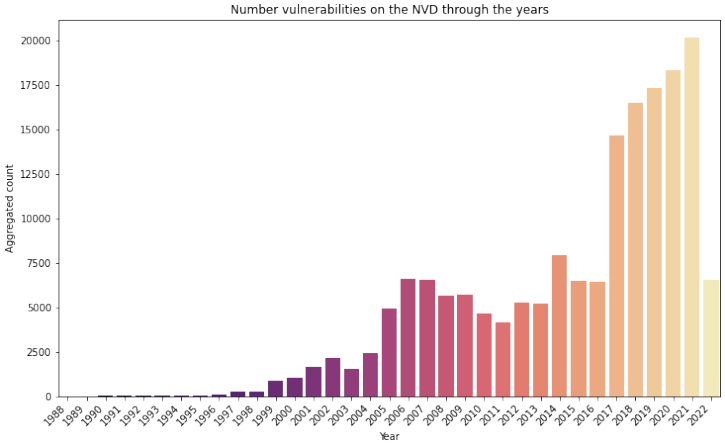


Figure 4

The number of known exploits in the NVD per year (see Figure 5) were also plotted. Only 9.40% of vulnerabilities were observed to contain known exploits before the year 2000. These data were removed to prevent a potential imbalance.

The vulnerabilities classified during the same month of data retrieval (March of 2022) were also analysed. Only 2.83% of these vulnerabilities were observed to contain an exploit, likely because these data were entered too recently for exploits to have been detected. Therefore, these entries were also removed (which represented 0.30% of the dataset).

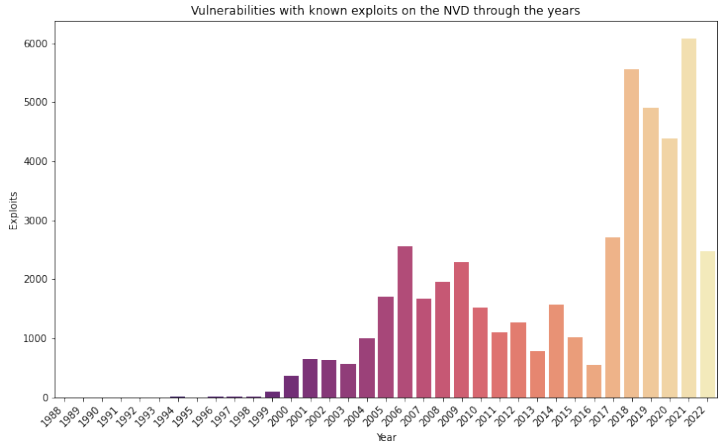


Figure 5

After dropping the previously mentioned outliers, as expected, we observed that

the minority class percentage barely incremented from 27.37% to 27.61%. The minimal percentage increase occurred because only 1.21% of the dataset entries (2104 in total) was deleted.

11.2 Feature Extraction

11.2.1 Extracting Numerical Features

All selected variables, except for `publishedDate` and `lastModifiedDate`, are text data. Therefore, to examine new numerical variables and better explain the data, we derived four new features from the previously collected information:

- `numReferences:` Size of the reference list in the `references` feature per entry.
- `numCPEs:` Number of CPEs in the `cpes` feature tree per entry.
- `descriptionLength:` Length of the description in the *description* feature per entry.
- `dayDifference:` Number of days between `publishedDate` and `lastModifiedDate` per entry.

After numerical feature extraction, `cpes` was deleted because its information was too specific to each entry (since it contained information about the versions of every product affected) and would add too much noise to the data.

11.3 Natural Language Processing Techniques

The remaining unprocessed features (`assigner`, `cwes`, `references`, and `description`) were textual data. Because ML models only understand numerical data, transforming these data into numerical data was required to feed them into the classifiers. Furthermore, because all of the textual data was in English, it was possible to use all of the data.

11.3.1 Bag of Words + TF-IDF

Initially, all textual data were processed with a combination of different NLP techniques outlined below.

The *bag of words* technique represents a text as a multiset of words, keeping count of the appearance of each word in the text.

The *TF-IDF* (Term-Frequency - Inverse-Document-Frequency) term weighting technique indicates the importance/relevance of a word to a document within a corpus. Instead of using the raw frequencies of word occurrence, this technique reweights a text corpus to give more value to rare terms. Doing so increases their relevance, thereby preventing more frequent terms from shadowing the frequencies of rare terms. The formula used to compute the TF-IDF for a term t in a document d in a document set is:

$$TFIDF(t, d) = tf(t, d) * idf(t)$$

And $idf(t)$ is computed as:

$$idf(t) = log[n/df(t)] + 1$$

Where n corresponds to the total number of documents in the corpus and df corresponds to the number of documents in the document set that contain the term t.

To process features, the function `TfidfVectorizer`[31] from the sklearn library was used, combining the above two techniques. The following steps were performed for each feature.

1. A bag of words was created, generating a new column for each word that appeared in all combined entries. For every entry of a represented word, its corresponding column contained its count. To determine the nature of words suitable for the bag, we used a specific regex pattern per feature (see Table 7). (For `description`, the bag was created without considering words in the English stop word list [32].)
2. The bag of words matrix is transformed to a TF-IDF representation.
3. The Dataframe was concatenated with the matrix, increasing the dimension of the dataset by the number of words contained in the bag of words.

Feature name	Regex	Explanation
`assigner`	`[^\d \W ]+\w +`	Only accepting words with length higher or equal than 2
`cwes`	`^\S +@\S +\.\S +$`	Accepting the email address format
`references`	`[\w +\-]+\w +`	Accepting the CWE labels (with format: CWE-X)
`description`	`[\w ]+[.\w ]+`	Accepting preprocessed URLs only without subdirectories

Table 7: Regex implementation per feature

After performing the previous steps and concatenating the four generated matrices, the number of dimensions added to the dataset were analysed. `assigner` and `cwes` had 193 and 315 dimensions, respectively. In comparison, `references` had 12638 and `description`, 146591.

The dimensionality added by the latter two is a problem. Named the *Curse of Dimensionality* by Richard Bellman [33], this concept explains that as dimensionality of data increases, the number of data points required for good performance on ML models increases exponentially. This is because the volume of the space increases so fast that data becomes sparse, preventing models from forming groups and finding patterns. Additionally, this extra dimensionality drastically increases the computational power required to train models. To lower the impact of the *Curse of Dimensionality*, the number of features generated by `references` and `description` were reduced.

Only the 100 most frequent features in `references` were kept. However, adding such a threshold for the features in `description` would cause tremendous data loss because more than 90% of the words would have to be discarded. Thus, to represent the descriptions in our dataset, we used a different NLP technique, *Doc2Vec*.

11.3.2 Doc2Vec

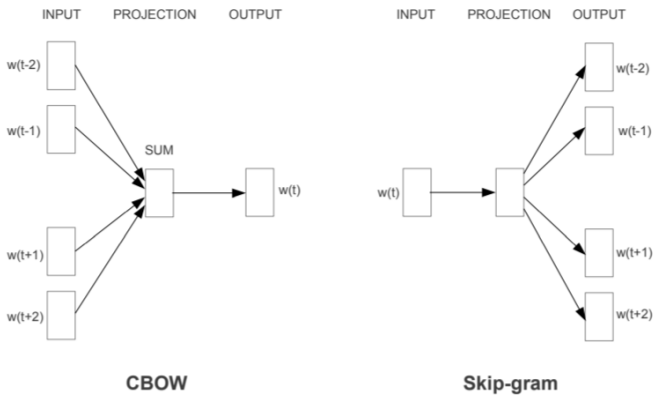
Doc2Vec is a document embedding technique used to create a vectorised representation of each document. *Doc2Vec* is best understood in context with its parent, *Word2Vec*.

Word2Vec provides a numeric representation for every word in a text corpus. This representation encapsulates different relations between words (synonyms, antonyms, analogies, etc.) using two potential algorithms:

- **Continuous bag of words (CBOW):** This algorithm combines sur-

rounding words (context) to predict each word.

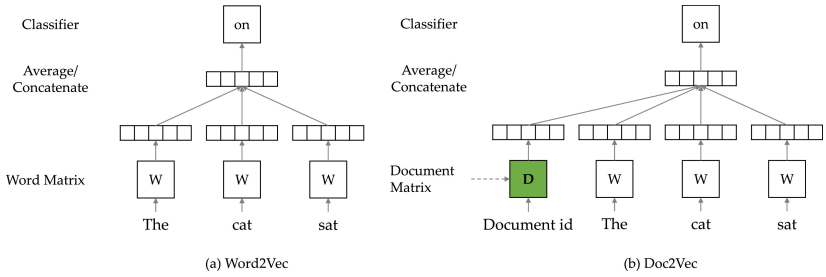
- **Skip gram:** The opposite of CBOW, this algorithm uses the distributed representation of an input word to predict the context.



Source: Exploiting Similarities among Languages for Machine Translation [34]
Figure 6: diagram of CBOW and Skip gram

Doc2Vec can be seen as an extension of *Word2Vec*. This technique can also use two algorithms, which are adaptations of the ones previously explained.

- The **Distributed Memory Model of Paragraph Vector (PV-DM)** is an extension of the CBOW. When predicting a word, this algorithm adds a document-unique feature vector (see Figure 7) that acts as a memory for information missing from the current context.
- The **Distributed Bag of Words of Paragraph Vector (PV-DBOW)** can be classified as an extension of the Skip gram since the vector representation of a document is used to predict its context.



Source: Difference between Word2Vec and Do2Vec [35]
Figure 7: diagram of CBOW and Skip gram

The Python implementation of *Doc2Vec* by Gensim [36] was used (see the code in figure 29 on page 67), and the parameters chosen were the following.

- **PV-DM mode:** Although this model takes more time and memory to train, it was used instead of PV-DBOW because it usually performs better.
- **100 dimensions to represent the descriptions:** 100 dimensions were added to our dataset instead of 146591, reducing the feature's dimensionality by 99.93% while still meaningfully representing the descriptions in the dataset.
- **Window size of 100:** This measurement is the maximum distance between the current and predicted word within a sentence.
- **Minimum count of 1:** Words with a length lower than two will not be considered.

11.4 Feature Selection

After preprocessing, the final dataset contained 171342 samples and 715 features. However, such large dimensions consume too much computational power to train ML models (as explained in Section 13.3). Thus, considering the *Curse of Dimensionality*, the number of features was reduced.

Because information loss is inevitable, selecting features is a delicate task. To minimise this loss, two different selection methods were used to study feature importance: *mutual information* and *ANOVA F-Value*. However, the use of principal components analysis (PCA) was not viable due to the computational power required to run this method with the size of the dataset.

11.4.1 Mutual Information

Mutual information measures the mutual dependence between two variables. To calculate mutual information, a function provided by Sklearn [37] was used. This function is based on entropy estimation from the distances of k-nearest neighbours and quantifies the "amount of information" obtained from our features by observing the label feature. Additionally, this function can capture any kind of statistical dependency.

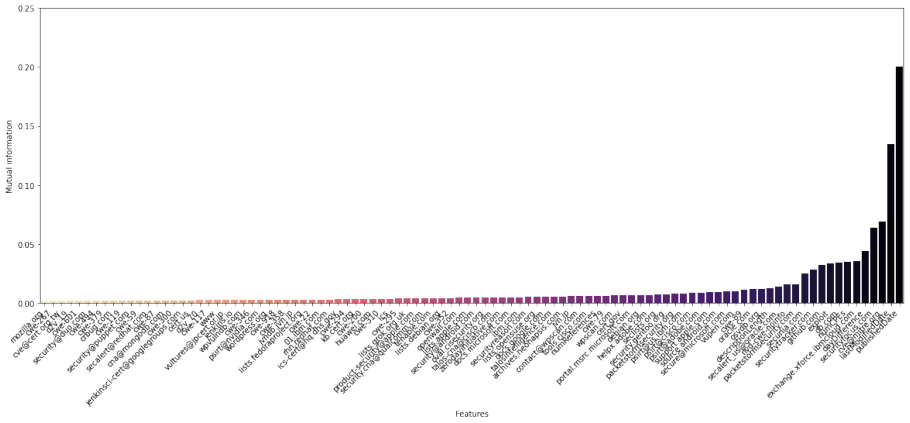


Figure 8: 100 highest mutual information features

After the execution of this function, the top 100 features with the highest mutual information with respect to `hasExploit` (see Figure 8) were displayed. Overall scores were observed to be very low (scores can range from 0 to 1). Only two features, (`publishedDate` and `modifiedDate`), had a score over 0.1. Furthermore, only two of the Doc2Vec features appeared on the plot.

11.4.2 ANOVA F-Value

Based on the F-test, this method estimates the degree of linear dependency between the label feature and the other features. This estimation is calculated by dividing two mean squares, which determines the ratio of explained variance to unexplained variance. The Sklearn estimator [38] was used.

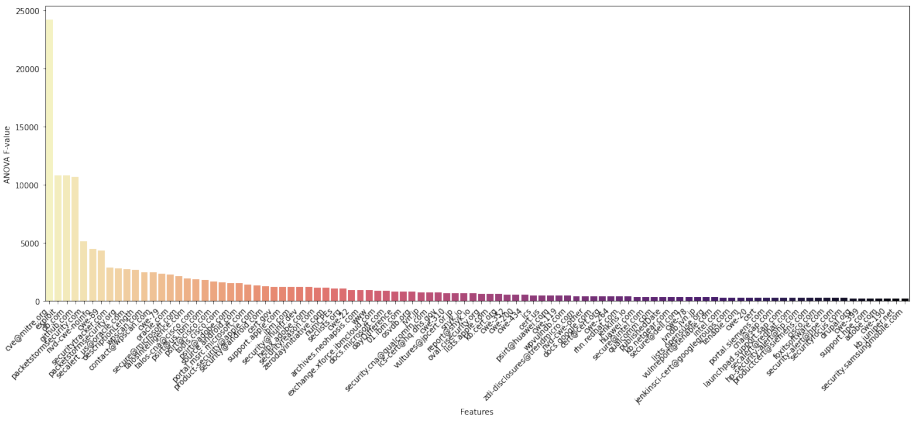


Figure 9: top 100 features with highest f-value

Again, the 100 features with the highest scores were plotted. A similar distribution to the mutual information plot was observed. However, the highest ranking feature was the assigner `cve@mitre.org`, differing from the previous method. No dimension from the Doc2Vec vectors survived.

11.4.3 Conclusion

Both plots described a theoretical lack of independent correlation between the features and the predicted label. Although, considering different combinations of features from the `descriptions` Doc2Vec vectors results, some may have provided relevant information to the ML classifiers.

The distributions obtained from the *mutual information* and *anova f-value* methods were very similar. However, because the *mutual information* method is capable of capturing all kinds of dependency and robustness, only the highest 100 features from the *mutual information* method were kept. Nevertheless, considering the lack of independent relevance of the Doc2Vec vector dimensions with the label, another dataset that combined the highest 100 features from the *mutual information* method with the 100 Doc2Vec dimensions was created for experimentation.

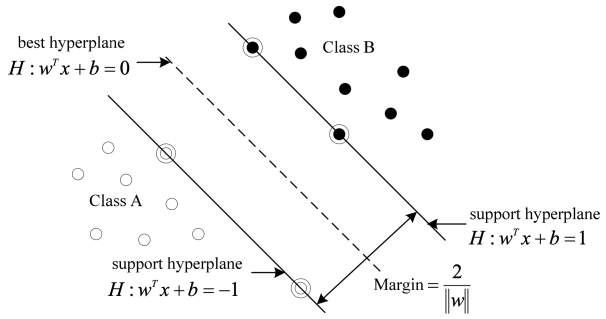
12 Model Selection

Various models were tested to ensure breadth of results and achieve the best performance. After doing so, the models selected were *Linear Support Vector Machines*, *Random Forest*, *Multinomial Naive Bayes*, and *AdaBoost*.

12.1 Linear Support Vector Machines

A Support Vector Machine (SVM) maps training samples to points in space, designating each into one of two previously mentioned categories: exploitable or non-exploitable. Doing so maximises the length between the classes with a hyperplane (see Figure 10). To predict the category of new samples, each one is mapped into that same space. The class of a sample is selected based on its location in the hyperplane

An SVM can also perform nonlinear classifications with a technique called the kernel trick, which maps the input into high-dimensional feature spaces.



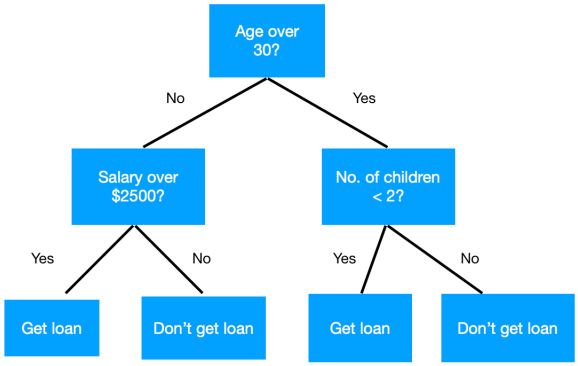
Source: Support vector machine for vibration fault classification of steam turbine-generator sets[39]

Figure 10: Hyperplane separating two classes in 2 dimensions

Because of the dataset size, an SVM is appropriate for the classification problem as it accounts for large numbers of features and entries.

12.2 Random Forest

To understand the Random Forest (RF) classifier, decision trees must first be described. A decision tree contains branches, nodes, and leaves. Its nodes are labelled with a condition based on a dataset feature. Branches coming out of a node represent all possible values of the feature. Therefore, nodes split the dataset based on the value of the feature they represent. This is done recursively until the entries fall into a leaf. Leaves are labelled with either a class or a probability distribution over the classes.



Source: https://eloquentarduino.github.io/2020/10/decision-tree-random-forest-and-xgboost-on-arduino/
Figure 11: Example of a decision tree classifier

A RF is an ensemble learning method since it uses a set of decision trees. Classification in a RF is based on the class selected by the majority of trees.

By using a set of decision trees, RF usually performs better than single decision tree methods because it corrects the overfitting sensitivity of decision trees.

As a nonlinear classifier, RF is convenient because it effectively processes large datasets and usually achieves high accuracy.

12.3 Multinomial Naive Bayes

The naive Bayes method classifies data based on its "likelihood" of belonging to a specific class by using Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

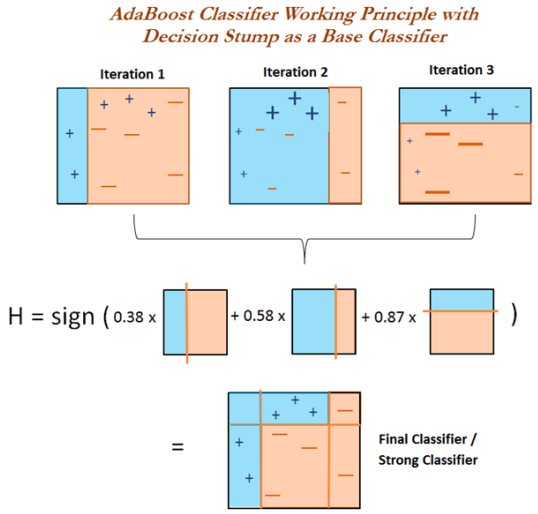
, where:

- $P(A|B)$ is the conditional probability of A occurring given that B is true
- $P(B|A)$ is also a conditional probability that can be understood as the likelihood of A given a fixed B.
- $P(A)$ and $P(B)$ are the probabilities of A and B occurring respectively.

Using the same naive Bayes principle, the multinomial model has the ability to classify data that cannot be represented numerically. Since most of our features originated from text, this simple method is very relevant because of its wide use in document classification.

12.4 AdaBoost

The Adaboost model is a meta-estimator that creates a sequence of learning algorithms (in this case, only decision trees). Using the outputs of each algorithm, Adaboost learns from the misclassifications of previous iterations to improve the accuracy of future ones.



Source: Numerical Computing with Python [40]
Figure 12: AdaBoost classifying example

Similar to RF, Adaboost is tree-based. However, though more sensitive to overfitting, Adaboost usually performs better than its former counterpart.

12.5 Rejected Classifiers

In addition to these classifiers, the Radial Basis Function kernel SVM was also considered. Unlike the linear kernel, this kernel can handle nonlinear relationships between class labels and attributes. However, during the initial stages of the training phase, this classifier was observed to be exponentially more time consuming with respect to the other classifiers.

The time consuming nature of kernel-based SVMs is caused by their computational drain. Kernel-based SVMs require n^2 computation for training and $n * d$ computation for classification, where n is the number of training entries and d the dimension of the entries. However, 2-class linear SVMs have a computational cost of nd (times the number of training iterations, which remains small even for large n) for training and d for classification. Therefore, when the training set is very large (such as in this case), RBF kernel SVMs become very expensive for training and classification.

13 Model Implementation

13.1 Data Splitting

Data used to train and test a model must differ. If the same data were fed, during testing, the model would mimic the labels of samples seen during training and achieve a perfect score. However, if the model were fed with new data, it would fail to predict them, a phenomena known as overfitting.

To avoid overfitting the models, the dataset was split into a training set and a testing set. The training set contained 2/3 of the dataset, and the testing set contained the other 1/3.

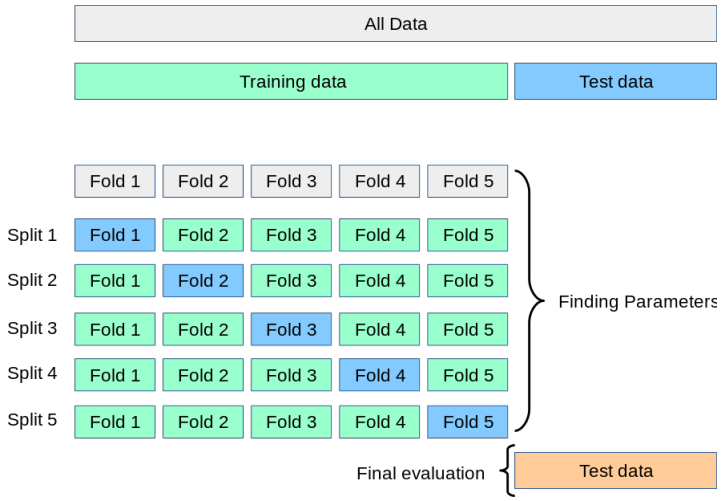
The split of data was stratified to maintain the ratio of the label feature `hasExploit`. Additionally, doing so prevented the training set from having an unrealistic number of exploited samples with respect to the overall dataset.

13.2 Hyperparameter Tuning

To select the best performing model, all selected models were trained with a variety of different hyperparameter combinations. Those with the best performing hyper-parameters were kept. The grid-search strategy, an exhaustive search process over all specified hyper-parameters for every model, was utilised to manage hyperparameter tuning for each classifier.

The risk of overfitting still exists when evaluating the performance of every combination. This probability is due to hyper-parameter tweaking, wherein hyper-parameters are adjusted until they perfectly integrate with the testing set, allowing for knowledge to leak. To this problem, k-fold cross-validation was performed.

To perform k-fold cross-validation, the training set was split into k sets, and the model was trained with $k - 1$ sets for every fold. The resulting model was validated with the remaining set (see Figure 13). Obtaining the evaluation in the training set was done by scoring values computed at each iteration were averaged after iterating over all k folds. In this case, k was set to 5.



Source: https://scikit-learn.org/stable/modules/cross_validation.html
Figure 13: Grid search with 5-fold cross-validation

13.3 Resampling

Grid-search, combined with cross-validation, can be computationally very expensive on large datasets, such as the one that was used. After feature selection, two datasets (one with 100 features and the other with 198) were kept. Time taken to train a model with the larger of the two was discovered to be exponentially higher. Thus, executing the methods on all of the models was not feasible, as experimenting with different settings was very difficult to practically accomplish.

Therefore, to select the best performing model in a reasonable amount of time, the smaller dataset was used with all of its samples, and the bigger dataset was used with a stratified random sample representing the 25% of the training set. However, once the best hyper-parameters for every model were discovered, the whole training set was trained to investigate the consistency of scores.

13.4 Scaling

Before outlining the training of the models, each classifier will be summarised below.

Linear SVM is a distance-based model. If there are certain features with higher scales than others (which happens with the temporal features), linear SVM can develop a bias towards the higher scaled features, overshadowing the rest. To

avoid this problem, linear SVM was trained and tested with standardised data, wherein features were rescaled to have a mean of 0 and a variance of 1. Doing so reduced all of the features to a common scale without distorting the differences in the range of their values. The `StandardScaler` from Sklrean was used for standardisation. To avoid introducing new information into the testing set, only the training set was fitted. Subsequently, the computed mean and variance was used to transform the whole dataset (training and testing sets).

Multinomial Naive Bayes assumes amongst its features a multinomial distribution of solely non-negative values. Therefore, to set all data in the range of [0,1] and avoid negative values, the `MinMaxScaler` from Sklearn was used.

As for the other tree-based classifiers (Random Forest and AdaBoost), rescaling was not required, as they are both insensitive to feature scaling.

13.5 Training Analysis

To measure the performance of the classifiers, appropriate metrics were required to avoid misleading conclusions. If, through the predictions of these models, a company discovers a vulnerability with a high chance of exploitation, a false positive prediction is significantly less harmful than a false negative one, since letting a cybervulnerability with high exploit potential slip through could be fatal for the company. Thus, the main aim of this classification is to maximise the number of correct predictions of vulnerabilities that contain known exploits. Though all vulnerabilities prone to exploits cannot be predicted, an effective balance must be discovered.

To analyse and determine the performance of the models, confusion matrices were used to display *True Positives (TP)*, *True Negatives (TN)*, *False Positives (FP)*, and *False Negatives (FN)* in a 2x2 matrix. A *positive* sample indicated that `hasExploit = 1`, and a *negative* one indicated that `hasExploit = 0`. *true* and *false* indicated that the sample was classified either correctly or incorrectly, respectively. This information was subsequently used to calculate the following four metrics (with values ranging from 0 to 1).

- **Accuracy:** This metric represents the number of correctly classified samples over the total number of instances. It is defined as $\frac{TP+TN}{TP+FP+TN+FN}$. As mentioned in Section 11.1.1, imbalanced datasets can mislead accuracy. Thus, this metric was not considered to be most important.
- **Precision:** This metric represents the number of correctly classified samples with an exploitable cybervulnerability over the total number of classified samples with an exploitable cybervulnerability. It is defined as $\frac{TP}{TP+FP}$. In relation to the data used, this metric displayed the number of samples with an actual exploit out of all exploit-labelled samples.

- **Recall:** This metric represents the number of correctly classified samples with an exploitable cybervulnerability over all the samples with an exploitable cybervulnerability. It is defined as $\frac{TP}{TP+FN}$. Recall was important in identifying the number of samples with an actual exploit that were labelled.
- **F1-Score:** This metric represents the weighted average of the precision and the recall. It is defined as $2 * \frac{Precision*Recall}{Precision+Recall}$.

Another good metric considered was the Receiver Operating Characteristics - Area Under The Curve (ROC-AUC). ROC is a probability curve, and the AUC is the degree or measure of separability. In the context of this project, this metric indicated the capability for a model to distinguish the value of `hasExploit` as either 0 or 1.

13.6 Choosing the Number of Dimensions for the Models & Learning curves

As described in 11.1.1, the `hasExploit` label class contained an imbalance that could have led to potential overfitting. To detect biases in the models learning curves were used, displaying the cross-validated training and validation scores for increasing training set sizes. The following plots demonstrate these learning curves and their standard deviation:

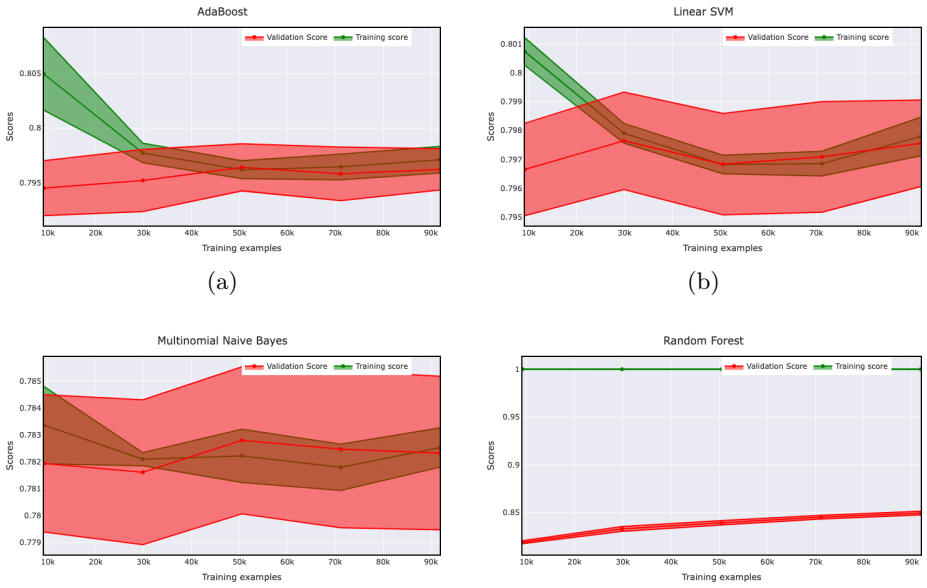


Figure 14: Learning curves with 100 features

The learning curves of the models corresponding to subfigures 14a, 14b, and 14c indicate that these models were robust enough to avoid overfitting. However, the RF plot describes a clear pattern of overfitting in relation to the high performance of training in comparison to testing.

These learning curves encouraged caution in regard to the performance of RF. If results directly averaging the 5-fold cross validation metrics of all the models (as demonstrated in Figure 15) were compared, then outperformance of the RF model in all metrics could have been argued (with an accuracy of 0.8492, a precision of 0.7672, a recall of 0.6515 and an F1-score of 0.7046). However, the likelihood of this model struggling to generalise with unseen data was high. Therefore, the overfitting of RF will be minimised while preserving similar metrics without such high bias.

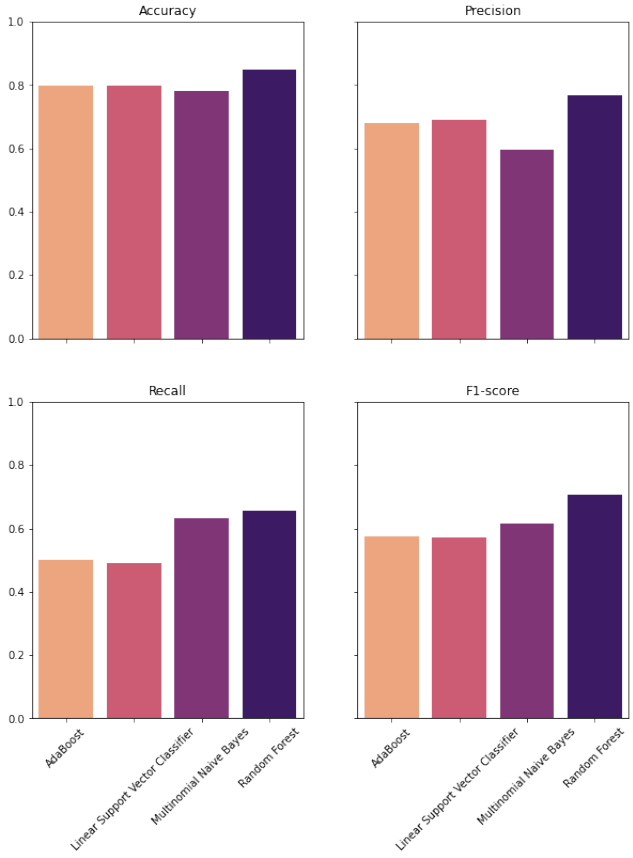
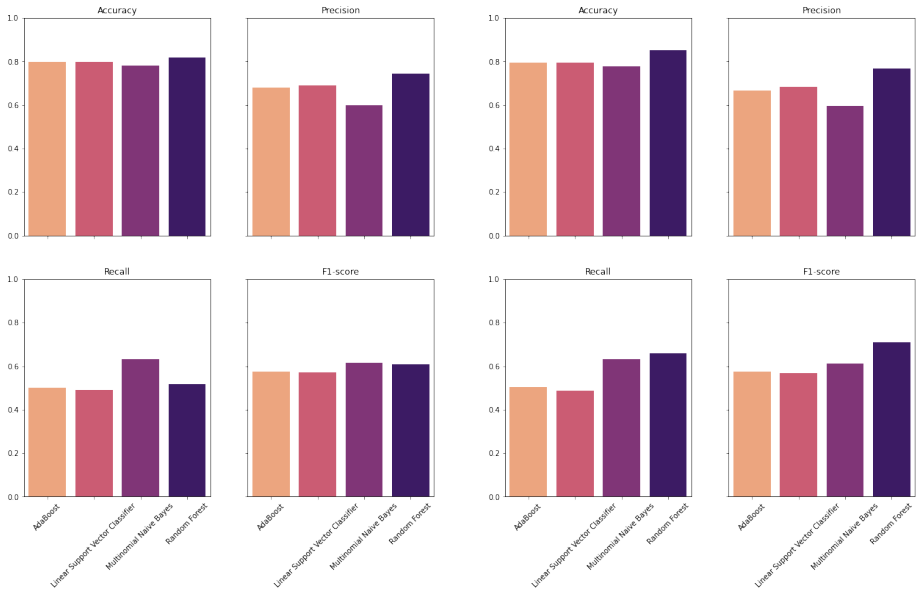


Figure 15: Metrics without hyperparameter tuning with top 100 features

Before reducing the bias of RF, the same process was repeated with the dataset that included the additional 100 dimensions from Doc2Vec (a total of 198 dimensions). Similar learning curves were obtained, and overfitting was still present on RF. Once more, metrics were plotted from every model (see Figure 16a). Though overall scores didn't change much, the scores of RF were significantly worse, indicating that the features of Doc2Vec added noise to the training. Therefore, to identify the effect of condensing dimensions and observe if doing so would increase relevance, the same steps were repeated with a new Doc2Vec model consisting of only 20 dimensions. The training metrics of this 120 feature dataset were plotted (see Figure 16b). However, results indicated slightly worse performance than the 100 feature one. This helped further reinforce the idea of the counterproductivity of Doc2Vec during the training phase.



(a) Top 198 features

(b) Top 120 features

Figure 16: Metrics without hyperparameter tuning

Finally, feature importance of RF with the 100 Doc2Vec features was plotted, as shown in Figures 24 and 25 on pages 60 and 61. These plots show similar importance across the Doc2Vec extracted features. Therefore, considering the previous analysis, they were discarded from use during the remaining phases.

13.7 Overcoming the Overfitting of Random Forest

There are multiple solutions to handling overfitting in a ML model. Some involve performing more preprocessing to the data before training. Others can be implemented directly on the model. For this project, the `RandomForestClassifier` function from Sklearn was used [41], which contains some hyperparameters that aim at compensating the imbalance. The following subsections outline the implementation of both kinds of techniques.

13.7.1 Oversampling and Undersampling

These two techniques are widely used for balancing unbalanced datasets. The *Oversampling* approach creates new synthetic data points that imitate the behaviour of the minority class. Doing so increases the size of the minority class

until it reaches the size of the majority class. The *Undersampling* approach will delete samples from the majority class until its size matches that of the minority class.

To apply both of these techniques, python implementations from the Imbalanced Learn [42] library were used. Oversampling was handled through the Synthetic Minority Oversampling Technique (SMOTE), which creates new synthetic data points that imitate the behaviour of the minority class. Undersampling was handled through Random Undersampling, which removes entries from the majority class randomly.

To ensure no data leakage, three different strategies were tested by using a pipeline containing three different flows. The strategies are outlined below.

1. `RandomUnderSampler` followed by the classifier.
2. `SMOTE` oversampler followed by the classifier.
3. `SMOTE` oversampler followed by `RandomUnderSampler` and then the classifier. In this case, a different sampling strategy for both resamplings was used. A 0.5 ratio with the oversampling was created. Then, the data was undersampled to a ratio of 0.8. The combination of both techniques was experimented with because doing so may result in better performances than their use in isolation.

As seen in Table 8, the oversampling-only approach was still affected by severe overfitting, reflected similarly in 14d. Nonetheless, the other two techniques, although symptomatic of slight overfitting, appear to be less biased.

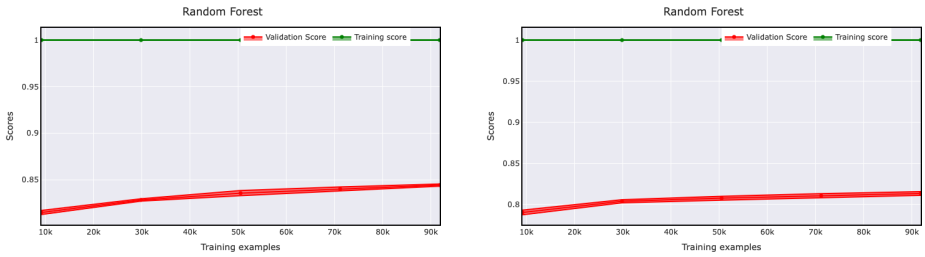
	Accuracy		Precision		Recall		F1	
	train	test	train	test	train	test	train	test
U	0.8886	0.7754	0.7126	0.5628	1	0.8368	0.8322	0.6729
O	1	0.8149	1	0.6844	1	0.6119	1	0.6461
C	0.9569	0.8090	0.8651	0.6345	1	0.7268	0.9277	0.6775

Table 8: 5-fold scores of undersampling (U), oversampling (O) and combination of both(C)

13.7.2 Hyperparameter Tuning

Overfitting on RF can also be resolved by tweaking some hyperparameters. In this case, to do so, the `RandomForestClassifier` was used. First, the `class_weight = 'balanced'` parameter was used, automatically increasing the weight of the minority class. However, as seen in Figure 17a, the overfitting was still

present. Next, the class weight ratio was manually set to `class_weight = 0:1, 1:3`, wherein keys are the values of `hasExploit` and numbers are the weight of the classes. As seen in Figure 17b, this method of tuning still indicated strong overfitting.



(a) Balanced parameter (b) Manual class weights

Figure 17: Learning curves with modifications to Random Forest

Two other parameters that may reduce the overfitting are the number of trees in RF and the depth of these trees. Usually, a high number of estimators decreases the potential bias of a model. Furthermore, if the depth of the trees is too large, then the trees may memorise the dataset and become easily biased. Therefore, controlling depth is important in preventing the potential bias of RF.

A grid-search cross-validation was performed with the `n_estimators` and the `max_depth` hyperparameters in the Sklearn model. Models with`n_estimators = 100` behaved very similarly to others with `n_estimators = 500`. Even more interestingly, as depth increased in the `max_depth` tests, better scores were generally obtained, however, with more overfitting. These results are shown in Table 9.

Depth	Accuracy		Precision		Recall		F1	
	train	test	train	test	train	test	train	test
5	0.7508	0.7498	0.8604	0.8466	0.1166	0.1145	0.2052	0.2016
11	0.8164	0.8018	0.8253	0.7748	0.4249	0.3980	0.5610	0.5259
17	0.8680	0.8089	0.9258	0.7667	0.5673	0.4424	0.7035	0.5611
25	0.9608	0.8146	0.9985	0.7535	0.8594	0.4882	0.9237	0.5925

Table 9: 3-fold scores of the Random Forest model with different depths

13.7.3 Conclusion

Overfitting was slightly reduced at the cost of also slighting reducing the validation scores. However, the data obtained through a combination of oversampling and undersampling was kept. As seen in Figure 18, although the model using this data slightly underperformed in most metrics, it maintained similar scores to the overfitted model and outperformed on recall. This choice was made considering the generalisation of the model for unseen data.

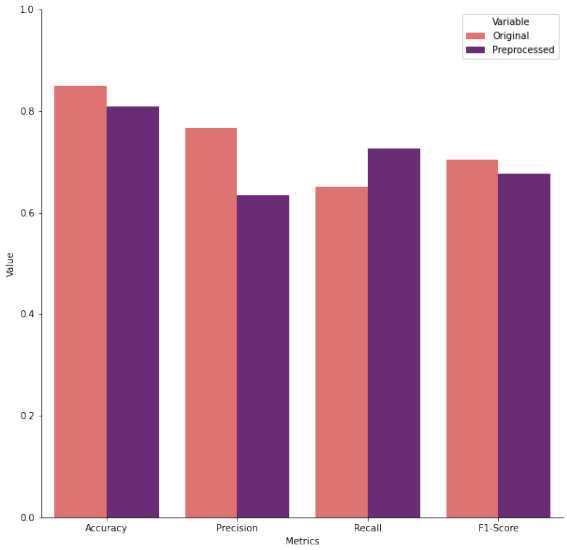


Figure 18: Metrics of unprocessed vs. preprocessed Random Forest

13.8 Choosing the Best Model

Although some parameters were previously tuned to reduce the overfitting of RF, the models were initially trained without hyperparameters. During this phase, using the same pipeline as RF on the previous section, oversampling combined with undersampling was performed to preprocess the data (except for the scaling) and compare the resulting metrics with the best estimators of grid-search.

All grid-searches were executed with a stratified sample of 25% of the training data. Additionally, the estimator was refitted with the F1-score to consider the best F1-scores in selecting the best estimator. Thus, the best balance was maintained between precision and recall. The resulting best estimators from the search were cross-validated with the entire training set to ensure consistent scores.

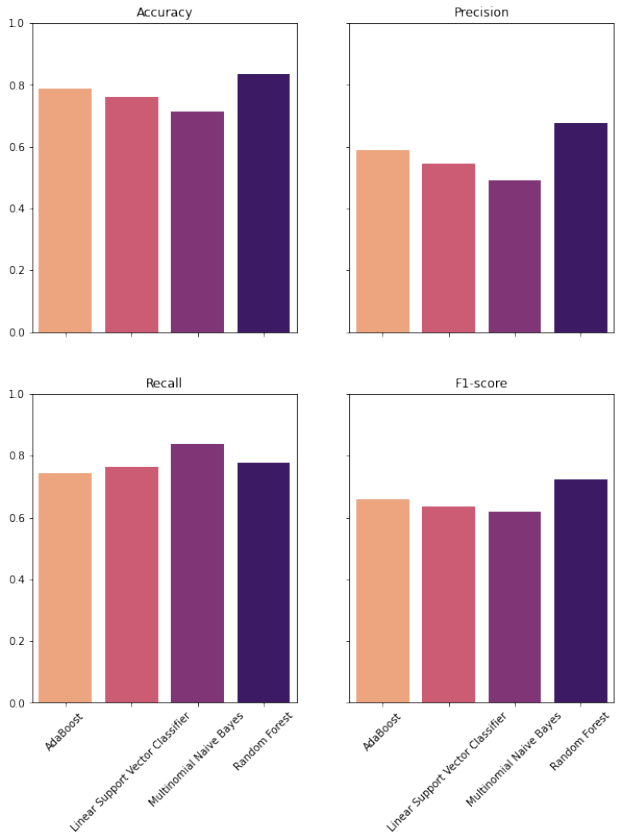
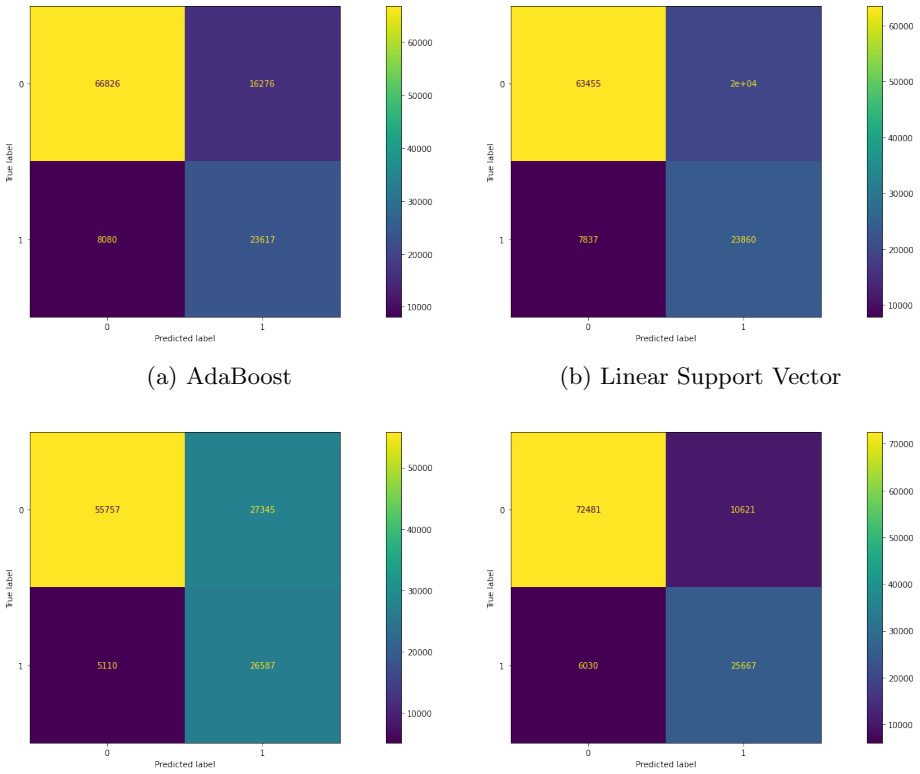


Figure 19: Metrics with grid-search cross-validation

The best scores of the estimators (see Figure 19) were plotted, demonstrating that RF kept performing slightly better than the remaining models in most metrics. More specifically, RF achieved 0.84 in accuracy, 0.68 in precision, 0.78 in recall, and 0.72 in the F1-score. Multinomial Naive Bayes performed very well on the recall metric with a 0.84 score, yet its precision was poor (0.49). This low precision occurred because the model predicted too many samples with `hasExploit = 1` (see Figure 20c) whether or not that prediction was correct. AdaBoost was also observed to perform similarly to RF (as we can also see in Figures 20a and 20d respectively) but still had 46% more misclassifications. Lastly, the Linear Support Vector Machine's confusion matrix (Figure 20b) had slightly less misclassifications of `hasExploit = 1` than AdaBoost, but displayed many more with the majority class.



(a) AdaBoost

(b) Linear Support Vector

(c) Multinomial Naive Bayes

(d) Random Forest

Figure 20: Confusion matrices from all the models

Finally the ROC-AUCs (see Figure 21) were plotted to further contrast the metrics. Observing that RF had a notably bigger area under the curve, RF was kept as the final model.

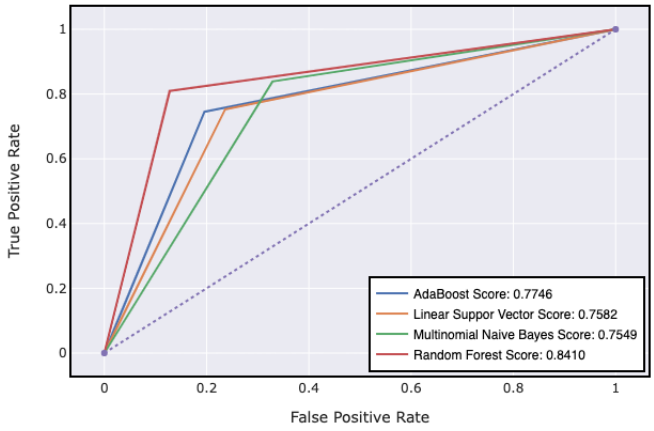


Figure 21: ROC curves of all models

13.8.1 Interpretability Analysis

After discovering the best model, its performance was tested with the test set of 56543 samples, as shown in the classification report on Table 10. The precision of the Exploit class was significantly lower than that precision of No Exploit. Since the minority class is rarer, this precision imbalance was expected, as demonstrated in the data analysis. The macro averages[4] were 2-6% below the weighted averages[5], suggesting that the model was solid against data imbalance.

	precision	recall	f1-score	support
No Exploit	0.90	0.85	0.87	40931
Exploit	0.65	0.74	0.69	15612
accuracy	-	-	0.82	56543
macro avg	. 0.77	0.80	0.78	56543
weighted avg.	0.83	0.82	0.82	56543

Table 10: Classification report by Sklearn

To visualise the results, the confusion matrix (see Figure 22) was plotted. Its distribution was very similar to that obtained with the training set, implying once again that the classifier was able to generalise.

[4]Unweighted mean of the metrics for each label. It does not take label imbalance into account.

[5]Mean of the metrics for each label weighted by support

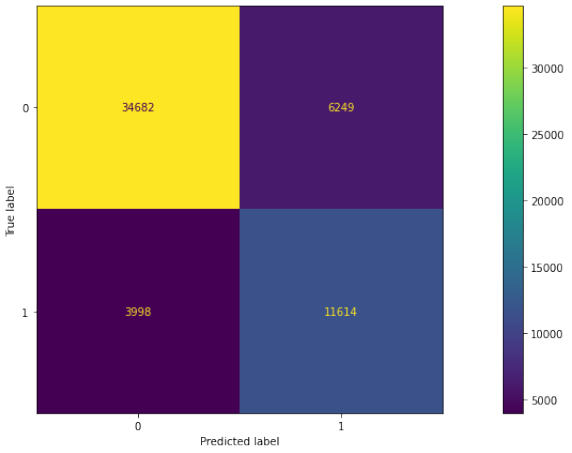


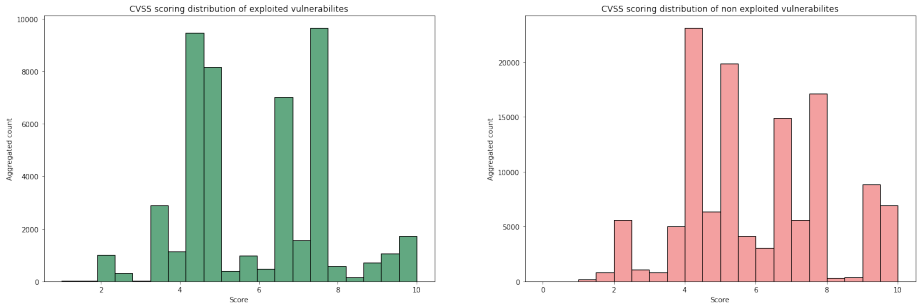
Figure 22: Test confusion matrix

The 25 most important features (see Figure 27 on page 63) for the prediction of the exploits contained all of the original numerical features extracted in Section 11.2.1. The most important feature, (`publishedDate`), the fourth most important feature, (`lastModifiedDate`), and the fifth most important feature, (`dayDifference`) are time based features, indicating that the model relies strongly on time to classify. It is also interesting how the 70th dimension of the vector generated by Doc2Vec was the sixth most important feature. The remaining features came from the NLP processing of the `assigner`, `references`, and `cwes`.

Another observation made was that they also contained 15 of the top 25 features on the RF feature importance with the Doc2Vec features.

13.8.2 Comparison with CVSS Scoring

The CVSS from the NVD dataset was plotted (see Figure 23) to show inconsistencies between the scoring and their exploit status. About the same quantity of exploited cybervulnerabilities were scored under 5, and mostly between 4 and 5 (Figure 23a). A similar behaviour can be seen on the non exploited cybervulnerabilities (Figure 23b). No pattern was detected in any of the histograms.



(a) (b)

Figure 23: Distribution of CVSS Scoring

Moreover, considering scores greater or equal than 5 being equivalent to `hasExploit = 1`, and the scores lower than 5 being equivalent to `hasExploit = 0`, the metrics of the CVSS base score the are the following.

	precision	recall	f1-score
No Exploit	0.25	0.35	0.15
Exploit	0.18	0.66	0.14
accuracy	-	-	0.43

Table 11: CVSS scoring metrics

Although these metrics are representative, they still were significantly lower than the ones obtained with the ML classifier. Therefore, these results support the idea that the ML classifier outperformed the most popular cybervulnerability severity scoring system.

14 Conclusions

Prioritising the treatment of cybervulnerabilities prone to exploits is essential for software companies. Due to increased frequency and limited resources, the ranking of cybervulnerabilites has become progressively difficult. Current methodologies of detecting cybervulnerability exploits rely heavily on human calculation, which takes time and is prone to error. However, the approach taken in this project aims to bridge the gap between the detection and volume of cybervulnerabilities. By using an ML classifier to automate exploit detection, analysis time was significantly reduced and overall performance was considerably increased.

Compared to current severity scoring systems, this predictor utilises a very different set of information from each cybervulnerability in the ranking process. Other NLP techniques enable the use of known information that is disregarded in current methodologies. By exploring comparisons between a variety of ML models, the highest performing one was selected, boosting performance metrics. Since the classification process can take many more factors into account, the use of ML is a very flexible approach for new predictions.

However, certain limitations were revealed. The computational power required for this project exceeded that of the computer that was used for its development. The handling of large datasets was made impossible by the lack of RAM and CPU. Servers from Google Colab, which have 12 GB of RAM and 2 CPU cores, were used to gain more RAM. Still, this hardware was not optimal, so the size of data was restrained to allow the machine to train the models. Furthermore, cybervulnerability descriptions were reduced to custom trained vectors of 100 dimensions, though later, these reductions proved not to be helpful. Perhaps, due to this general loss of information, scores could have improved significantly if more information was taken into account when training the models.

Obtained model metrics were overall positive. Though overfitting caused by class imbalance was corrected, retrospectively, scores may have room for improvement. To do this, the same process should be recreated with more computational power, avoiding as much loss of information as possible during the training of the ML models. The Doc2Vec vector could also improve by tuning its hyperparameters. Gathering data from a wider variety of databases, thus gaining more information and features for the ML, could be another potential improvement.

The ML methods, including this one, have room for improvement. However, the potential for ML to change the sphere of cybervulnerability detection is great and should be further explored.

References

[1] Christopher Kuner et al. 'The rise of cybersecurity and its impact on data protection'. In: *International Data Privacy Law* (June 2017). URL: https://doi.org/10.1093/idpl/ipx009 (visited on 26/02/2022).

[2] Frans Berkhout. 'Technological regimes, path dependency and the environment'. In: *Global environmental change* 12.1 (2002), pp. 1–4. URL: https://www.academia.edu/34502146/Technological_regimes_path_dependency_and_the_environment?from=cover_page (visited on 26/02/2022).

[3] Symantec. *ISTR Security Report*. 2019. URL: https://docs.broadcom.com/doc/istr-24-2019-en (visited on 26/02/2022).

[4] David Chinn, James Kaplan and Allen Weinberg. 'Risk and responsibility in a hyperconnected world: Implications for enterprises'. In: (2014). URL: http://dln.jaipuria.ac.in:8080/jspui/bitstream/123456789/2210/1/Risk%20and%20responsibility%20in%20a%20hyperconnected%20world%20%20Implications%20for%20enterprises.pdf (visited on 26/02/2022).

[5] Mehran Bozorgi et al. 'Beyond heuristics: learning to classify vulnerabilities and predict exploits'. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 105–114.

[6] Carl Sabottke, Octavian Suciu and Tudor Dumitraș. 'Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting {Real-World} Exploits'. In: *24th USENIX Security Symposium (USENIX Security 15)*. 2015, pp. 1041–1056.

[7] Nazgol Tavabi et al. 'Darkembed: Exploit prediction with neural language models'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[8] Octavian Suciu et al. 'Expected exploitability: Predicting the development of functional vulnerability exploits'. In: *arXiv preprint arXiv:2102.07869* (2021).

[9] *Common Vulnerability Scoring System SIG*. URL: https://www.first.org/cvss/ (visited on 27/02/2022).

[10] *Trello*. URL: https://trello.com/ (visited on 27/02/2022).

[11] *GitLab*. URL: https://gitlab.com/ (visited on 27/02/2022).

[12] *Discord*. URL: https://discord.com/ (visited on 27/02/2022).

[13] Ryte. *What is a web crawler and how does it work?* URL: https://en.ryte.com/wiki/Crawler#:~:text=A%20crawler%20is%20a%20computer,internet%20and%20build%20an%20index. (visited on 27/02/2022).

[14] Ministerio de trabajo y economía social. *Disposición 18607 del BOE núm. 272 de 2021*. URL: https://www.boe.es/boe/dias/2021/11/13/pdfs/BOE-A-2021-18607.pdf (visited on 07/03/2022).

[15] Glassdoor. *Average Project Director salary*. URL: https://www.glassdoor.es/Sueldos/project-director-sueldo-SRCH_KO0,16.htm?clickSource=searchBtn (visited on 07/03/2022).

[16] Glassdoor. *Average Project Manager salary*. URL: https://www.glassdoor.es/Sueldos/project-manager-sueldo-SRCH_KO0,15.htm?clickSource=searchBtn (visited on 07/03/2022).

[17] Glassdoor. *Average Junior Data Scientist salary*. URL: https://www.glassdoor.es/Sueldos/junior-data-scientist-sueldo-SRCH_KO0,21.htm?clickSource=searchBtn (visited on 07/03/2022).

[18] Glassdoor. *Average Junior Software Developer salary*. URL: https://www.glassdoor.es/Sueldos/junior-software-developer-sueldo-SRCH_KO0,25.htm?clickSource=searchBtn (visited on 07/03/2022).

[19] Glassdoor. *Average Junior Analyst salary*. URL: https://www.glassdoor.es/Sueldos/junior-analyst-sueldo-SRCH_KO0,14.htm?clickSource=searchBtn (visited on 07/03/2022).

[20] Statista. *Precio medio final anual de la electricidad en España de 2010 a 2022*. URL: https://es.statista.com/estadisticas/993787/precio-medio-final-de-la-electricidad-en-espana/ (visited on 08/03/2022).

[21] Wikipedia. *Open Source Vulnerability Database*. URL: https://en.wikipedia.org/wiki/Open_Source_Vulnerability_Database (visited on 21/03/2022).

[22] Flexera. *Flexera*. URL: https://www.flexera.com/products/software-vulnerability-manager (visited on 21/03/2022).

[23] BoradCom. *Symantec DeepSight*. URL: https://docs.broadcom.com/doc/deepsight-intelligence-overview-en (visited on 21/03/2022).

[24] IBM. *IBM X-Force Exchange*. URL: https://exchange.xforce.ibmcloud.com/ (visited on 21/03/2022).

[25] Accenture. *BugTraq*. URL: https://bugtraq.securityfocus.com/archive (visited on 21/03/2022).

[26] MITRE. *MITRE's CVE Database*. URL: https://cve.mitre.org/ (visited on 21/03/2022).

[27] Scrapy. *Scrapy*. URL: https://scrapy.org/ (visited on 12/02/2022).

[28] NIST. *National Vulnerability Database*. URL: https://nvd.nist.gov/ (visited on 21/03/2022).

[29] NIST. *National Vulnerability Database API*. URL: https://nvd.nist.gov/developers/vulnerabilities (visited on 21/03/2022).

[30] Pandas. *Pandas*. URL: https://pandas.pydata.org/ (visited on 20/03/2022).

[31] Scikit-learn. *TfidfVectorizer*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer (visited on 27/03/2022).

[32] Joel Nothman, Hanmin Qin and Roman Yurchak. 'Stop Word Lists in Free Open-source Software Packages'. In: *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 7–12. DOI: 10.18653/v1/W18-2502. URL: https://aclanthology.org/W18-2502.

[33] Richard Bellman. 'Dynamic programming'. In: *Science* 153.3731 (1966), pp. 34–37.

[34] Tomas Mikolov, Quoc V Le and Ilya Sutskever. 'Exploiting similarities among languages for machine translation'. In: *arXiv preprint arXiv:1309.4168* (2013).

[35] Donghwa Kim et al. 'Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec'. In: *Information Sciences* 477 (2019), pp. 15–29. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2018.10.006. URL: https://www.sciencedirect.com/science/article/pii/S0020025518308028.

[36] Gensim. *Gensim's Doc2Vec documentation*. URL: https://radimrehurek.com/gensim/models/doc2vec.html (visited on 05/04/2022).

[37] Sklearn. *Mutual Information*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html#r50b872b699c4-1 (visited on 10/05/2022).

[38] Sklearn. *ANOVA F-Value (f_classif)*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html (visited on 10/05/2022).

[39] Huo-Ching Sun and Yann-Chang Huang. 'Support vector machine for vibration fault classification of steam turbine-generator sets'. In: *Procedia Engineering* 24 (2011), pp. 38–42.

[40] Dangeti et al. *Numerical Computing with Python: Harness the Power of Python to Analyze and Find Hidden Patterns in the Data*. 2018.

[41] Sklearn. *RandomForestClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%5C%20forest#sklearn.ensemble.RandomForestClassifier (visited on 13/05/2022).

[42] Imbalanced Learn. *Imbalanced Learn*. URL: https://imbalanced-learn.org/stable/index.html (visited on 12/05/2022).

A Charts

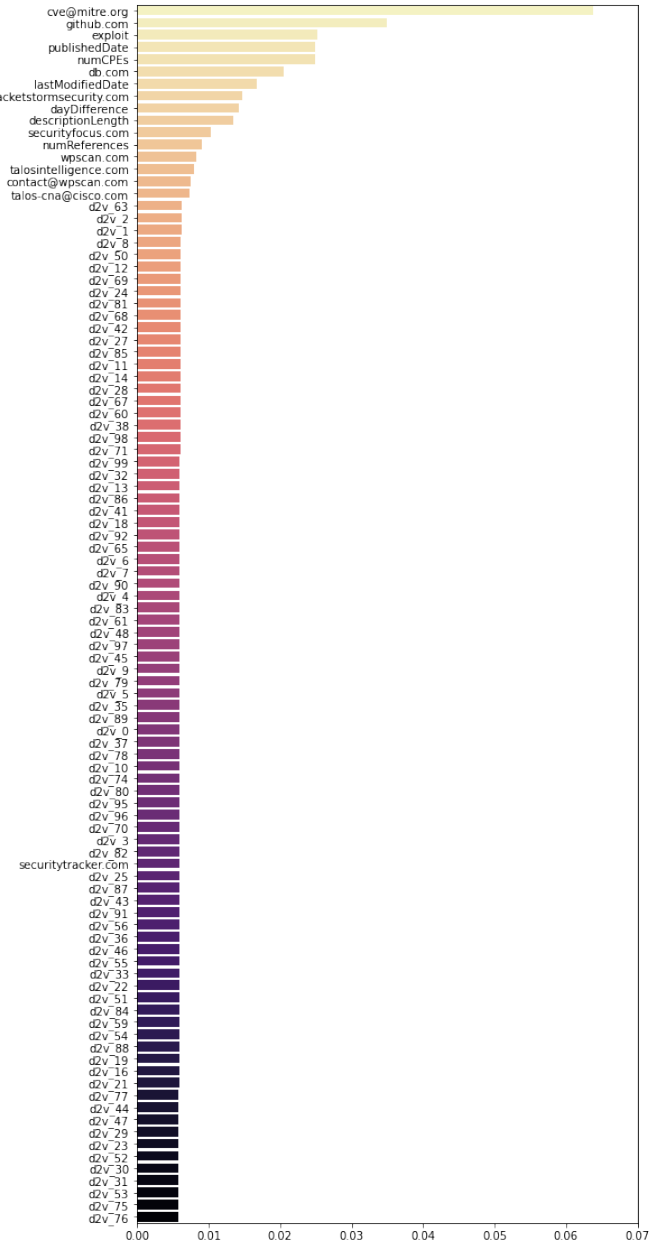


Figure 24: Feature importance with top 100 features combined with the 100 Doc2Vec dimensions (First 100)

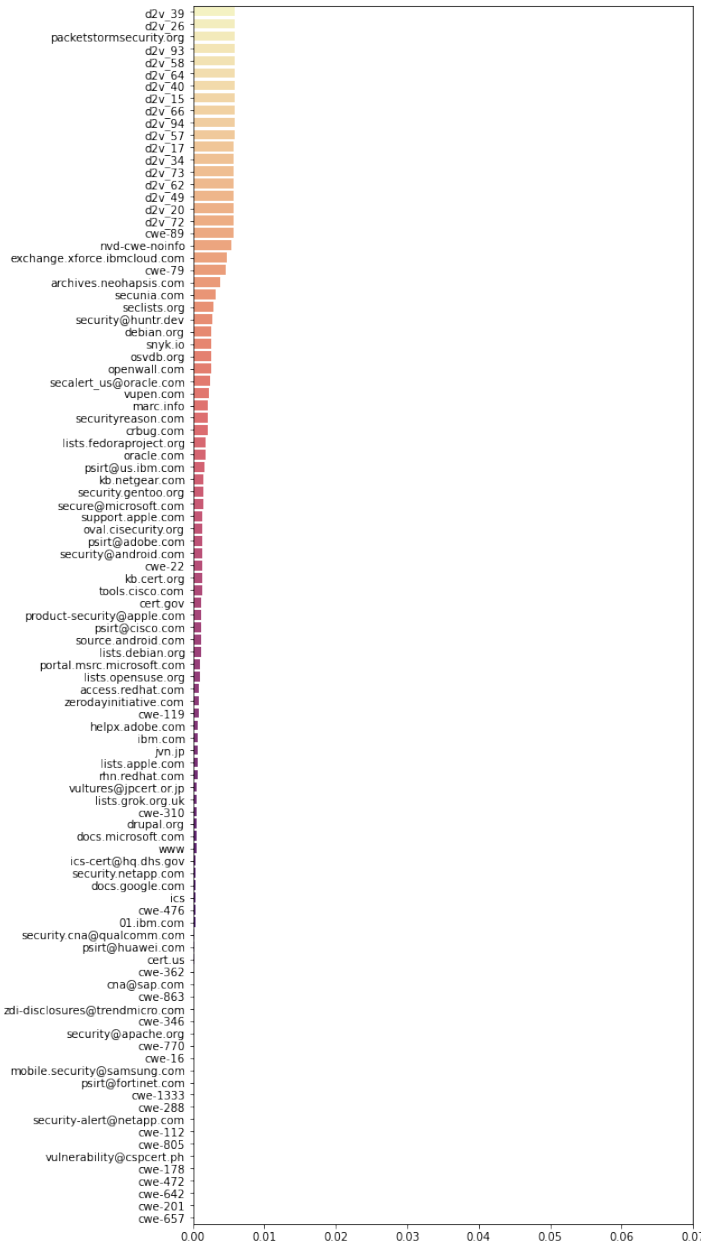
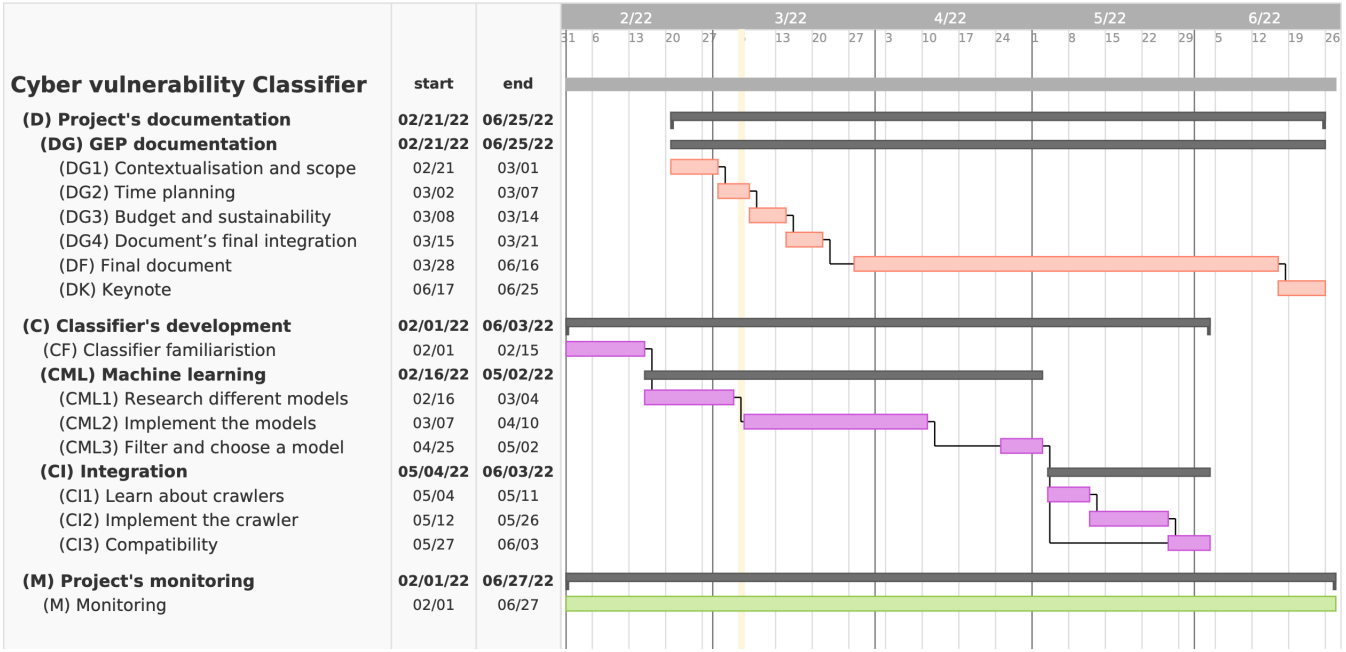


Figure 25: Feature importance with top 100 features combined with the 100 Doc2Vec dimensions (Remaining features)



Cyber vulnerability Classifier	start	end
(D) Project's documentation	**02/21/22**	**06/25/22**
(DG) GEP documentation	**02/21/22**	**06/25/22**
(DG1) Contextualisation and scope	02/21	03/01
(DG2) Time planning	03/02	03/07
(DG3) Budget and sustainability	03/08	03/14
(DG4) Document's final integration	03/15	03/21
(DF) Final document	03/28	06/16
(DK) Keynote	06/17	06/25
(C) Classifier's development	**02/01/22**	**06/03/22**
(CF) Classifier familiaristion	02/01	02/15
(CML) Machine learning	**02/16/22**	**05/02/22**
(CML1) Research different models	02/16	03/04
(CML2) Implement the models	03/07	04/10
(CML3) Filter and choose a model	04/25	05/02
(CI) Integration	**05/04/22**	**06/03/22**
(CI1) Learn about crawlers	05/04	05/11
(CI2) Implement the crawler	05/12	05/26
(CI3) Compatibility	05/27	06/03
(M) Project's monitoring	**02/01/22**	**06/27/22**
(M) Monitoring	02/01	06/27

Source: Own compilation, made with Teamgantt

Figure 26: Gantt chart

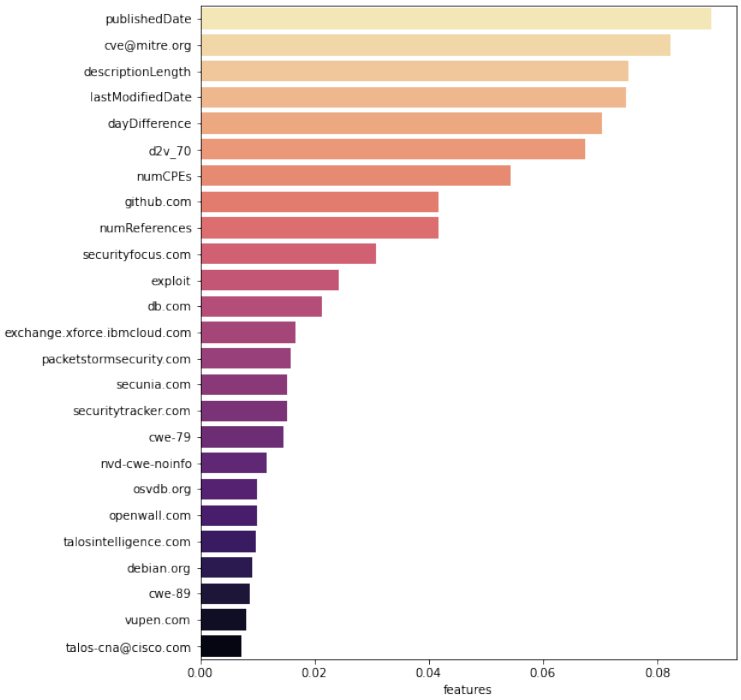


Figure 27: 25 most important features

B Code

B.1 MITRE Crawler

```
import scrapy

class MITREDETAILSpider(scrapy.Spider):
    name = "mitre_cveshref"

    def start_requests(self):
        url = 'https://cve.mitre.org/cgi-bin/cvekey.cgi?
            keyword='
        keyword = getattr(self, 'keyword', None)
        if keyword is not None:
            url += keyword
        yield scrapy.Request(url=url, callback=self.parse_cve)

    def parse_cve(self, response):
        cve_links = response.css('div#TableWithRules tr td a::
            attr(href)')
        yield from response.follow_all(cve_links, self.
            parse_cve_ref)

    def parse_cve_ref(self, response):
        print(response)
        yield {
            'cve_id': response.css('div#GeneratedTable tr:nth-
                child(2) td h2::text').get(),
            'link': response.url,
            'description': response.css('div#GeneratedTable tr
                :nth-child(4) td::text').get(),
            'refs': response.css('div#GeneratedTable tr:nth-
                child(7) td ul li a::attr(href)').getall()
        }
```

B.2 JSON payload of a CVE obtained with the NVD API

```
{
    "cve": {
      "data_type": "CVE",
      "data_format": "MITRE",
      "data_version": "4.0",
      "CVE_data_meta": { "ID": "CVE-2022-27249", "ASSIGNER": "
          cve@mitre.org" },
      "problemtype": {
        "problemtype_data": [
          { "description": [{ "lang": "en", "value": "CWE-434"
              }] }
```

```
      ]
    },
    "references": {
      "reference_data": [
        {
          "url": "http://packetstormsecurity.com/files
              /166559/IdeaRE-RefTree-Shell-Upload.html",
          "name": "http://packetstormsecurity.com/files
              /166559/IdeaRE-RefTree-Shell-Upload.html",
          "refsource": "MISC",
          "tags": ["Exploit", "Third Party Advisory", "VDB
              Entry"]
        },
        {
          "url": "https://www.idearespa.eu",
          "name": "https://www.idearespa.eu",
          "refsource": "MISC",
          "tags": ["Product"]
        }
      ]
    },
    "description": {
      "description_data": [
        {
          "lang": "en",
          "value": "An unrestricted file upload
              vulnerability in IdeaRE RefTree before
              2021.09.17 allows remote authenticated users
              to execute arbitrary code by using UploadDwg
              to upload a crafted aspx file to the web root,
               and then visiting the URL for this aspx
              resource."
        }
      ]
    }
  },
  "configurations": {
    "CVE_data_version": "4.0",
    "nodes": [
      {
        "operator": "OR",
        "children": [],
        "cpe_match": [
          {
            "vulnerable": true,
            "cpe23Uri": "cpe:2.3:a:idearespa:reftree
                :*:*:*:*:*:*:*:*",
            "versionEndExcluding": "2021.09.17",
            "cpe_name": []
          }
```

```
            ]
          }
        ]
      },
      "impact": {
        "baseMetricV3": {
          "cvssV3": {
            "version": "3.1",
            "vectorString": "CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:
                H/I:H/A:H",
            "attackVector": "NETWORK",
            "attackComplexity": "LOW",
            "privilegesRequired": "LOW",
            "userInteraction": "NONE",
            "scope": "UNCHANGED",
            "confidentialityImpact": "HIGH",
            "integrityImpact": "HIGH",
            "availabilityImpact": "HIGH",
            "baseScore": 8.8,
            "baseSeverity": "HIGH"
          },
          "exploitabilityScore": 2.8,
          "impactScore": 5.9
        },
        "baseMetricV2": {
          "cvssV2": {
            "version": "2.0",
            "vectorString": "AV:N/AC:L/Au:S/C:C/I:C/A:C",
            "accessVector": "NETWORK",
            "accessComplexity": "LOW",
            "authentication": "SINGLE",
            "confidentialityImpact": "COMPLETE",
            "integrityImpact": "COMPLETE",
            "availabilityImpact": "COMPLETE",
            "baseScore": 9.0
          },
          "severity": "HIGH",
          "exploitabilityScore": 8.0,
          "impactScore": 10.0,
          "acInsufInfo": false,
          "obtainAllPrivilege": false,
          "obtainUserPrivilege": false,
          "obtainOtherPrivilege": false,
          "userInteractionRequired": false
        }
      },
      "publishedDate": "2022-04-03T23:15Z",
      "lastModifiedDate": "2022-04-09T15:07Z"
}
```

B.3 Python code notebook to obtain the data through the NVD API

```
[ ]: import requests
     import json
     import time
```

```
[ ]: index = 0
     data = []
     while (True):
         response = requests.get(f'https://services.nvd.nist.gov/rest/json/cves/1.0/?
      ↪resultsPerPage=2000&startIndex={index}')
         json_data = json.loads(response.text)
         if json_data['totalResults'] == 0:
             break
         for cve in json_data['result']['CVE_Items']:
             data.append(cve)
         index += 2000
         time.sleep(8)
```

```
[ ]: with open('all_.json', 'w') as outfile:
         json.dump(data, outfile)
```

Figure 28: NVD Crawler code

B.4 Python code notebook to train the Doc2Vec model

```
[1]: # import nltk
     # nltk.download('punkt')
     from nltk.tokenize import word_tokenize
     import pandas as pd
     from gensim.models.doc2vec import Doc2Vec, TaggedDocument
```

```
[2]: data = pd.read_csv('data/all_cves_pre.csv')
```

```
[3]: tokenized_desc = []
     for d in data['description']:
         tokenized_desc.append(word_tokenize(d.lower()))
```

```
[4]: tagged_data = [TaggedDocument(d, [i]) for i, d in enumerate(tokenized_desc)]
```

```
[5]: model = Doc2Vec(tagged_data, vector_size = 100, window = 100, min_count = 1,␣
      ↪epochs = 100, workers=-1)
```

```
[6]: model.save('descriptions_100d_100w.d2v')
```

Figure 29: Doc2Vec code