Universitat de Girona
Escola Politècnica Superior

Projecte fi de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Tècniques d'Intel·ligència artificial per calcular la robustesa de xarxes

Document: Memòria

Alumne: Martí Madrenys Masferrer

Tutor: Eusebi Calle Ortega
Departament: ARQUITECTURA I TECNOLOGIA DE COMPUTADORS
Àrea: ARQUITECTURA I TECNOLOGIA DE COMPUTADORS

Convocatòria (mes/any): Juny 2022

Universitat de Girona
Escola Politècnica Superior

PROJECTE FI DE GRAU

Tècniques d'Intel·ligència artificial per calcular la robustesa de xarxes

Autor:
Martí MADRENYS MASFERRER

Juny 2022

Grau en Enginyeria Informàtica

Tutors:
Eusebi CALLE ORTEGA

Resum

La robustesa d'una xarxa de transport (telecomunicacions, aigua, electricitat, etc) ens dona un indicatiu per saber quina resistència tenen front diversos tipus d'atac (atacs dirigits, aleatoris, en cascada, etc.). Per calcular aquesta mètrica es necessita d'uns càlculs computacionalment costosos. En aquest TFG s'estudiaran i aplicaran diverses tècniques d'aprenentatge supervisat amb l'objectiu d'obtenuir un model que pugui pronosticar el valor de la robustesa d'una xarxa.

Agraïments

Per començar vull agrair molt especialment al grup de recerca BCDS per haver-me acollit durant aquest any. Especialment al meu tutor Eusebi Calle per el suport que he rebut durant l'elaboració d'aquest TFG.

Agrair també als companys del grup per haver-me ajudat a resoldre els dubtes que han anat sorgint durant aquest i d'altres projectes.

Finalment, voldria agrair a la meva família pel suport continuat durant aquests anys d'estudi sense el qual no hauria pogut realitzar.

Índex

1 Definicions **1**

2 Introducció **5**
2.1 Motivació 5
2.2 Propòsit 5
2.3 Objectius 5

3 Viabilitat **7**
3.1 Recursos informàtics 7
3.2 Recursos humans 7
3.3 Viabilitat econòmica 8

4 Metodologia **9**
4.1 Scrum 9
4.2 Aplicació al projecte 10

5 Planificació **11**
5.1 Etapes del projecte 11
5.1.1 Etapa 1: Planificació del projecte 11
5.1.2 Etapa 2: Estudi de l'estat de l'art 11
5.1.3 Etapa 3: Obtenció dels datasets i preparació 11
5.1.4 Etapa 4: Implementació i anàlisi dels candidats 12
5.1.5 Etapa 5: Millores als models i als datasets, elecció del model definitiu 12
5.1.6 Etapa 7: Anàlisi de característiques 12
5.1.7 Etapa 8: Exportació i tasques finals 12
5.1.8 Etapa 9: Redacció memòria 12
5.2 Diagrama de Gantt 12

6 Marc de treball i conceptes previs **15**
6.1 NRS 15
6.1.1 Experiments amb l'NRS 16
6.1.2 Mètriques d'una xarxa 17
6.1.3 Robustesa d'una xarxa 23
6.1.4 Xarxes Sintètiques 28
6.2 Jupyter Notebook 28
6.2.1 Magic 29
6.3 SciKit Learn 29

6.3.1 Pipes 30
6.4 Algoritmes d'intel·ligència artificial 30
6.4.1 Algoritmes supervisats 31
6.4.2 Models Lineals 31
6.4.3 Màquines de suport vectorial 32
6.4.4 Veïns propers 32
6.4.5 Arbres de decisió 33
6.4.6 Xarxes neuronals, models supervisats 33

7 Requisits del sistema **37**
7.1 Requisits funcionals 37
7.2 Requisits no funcionals 37

8 Estudi i decisions **39**
8.1 Candidats a model 39
8.2 Llenguatge, llibreries i entorn 39
8.3 Metodologia per avaluar els candidats inicials 40
8.3.1 Validació creuada 40
8.4 Eina d'importació de dades 40
8.4.1 *pandas* i *Dataframe* 41
8.5 Interfície d'execució del model 42

9 Anàlisi i disseny del sistema **43**
9.1 Implementació i avaluació dels candidats 43
9.1.1 Implementació de l'eina d'importació de dades 43
9.1.2 Regressió Lineal 44
9.1.3 K-Veïns Propers 45
9.1.4 Arbre de decisió i bosc 45
9.1.5 Xarxes neuronals de regressió (Multi-Layer Perceptron) . . 45
9.2 Primers resultats 46

10 Implementació i proves **49**
10.1 Millora 1: Hiperparàmetres 49
10.2 Millora 2: Quantitat de mètriques, *Pipes* i avaluació 51
10.2.1 Pipes i nova avaluació 52
10.2.2 Mètriques més rellevants 52
10.3 Millora 3: Unificació de models 57
10.3.1 Nova mètrica 57

11 Implantació i resultats **61**
11.1 Interfície de prediccions 61
11.2 Resultats i comparativa entre els dos mètodes 62

12 Conclusions **65**

13 Treball futur **67**
13.1 Incorporació a l'NRS 67
13.2 Nous *datasets* 67
13.3 Eina recomanadora de xarxes 67

Bibliografia **69**

Índex de figures

4.1 Esquema del funcionament d'*Scrum*. 10

5.1 Diagrama de Gantt 13

6.1 Esquema de l'estructura del simulador NRS. 15
6.2 Visualitzat de la xarxa BSONETEUROPE sobre el mapa real. 17
6.3 Taula amb les mètriques d'una xarxa. 23
6.4 Capçalera de l'NRS, pestanya experiments emmarcada en vermell. 26
6.5 Selecció d'experiments de l'NRS. Opció *METRICS* emmarcada en vermell. 26
6.6 Selecció de xarxes per fer l'experiment, s'han seleccionat algunes xarxes i emmarcat l'opció next en vermell. 26
6.7 Elecció de les característiques d'un experiment amb l'NRS. 27
6.8 Selecció de mètriques d'un experiment amb l'NRS, emmarcada la opció per usar les recomanades. 27
6.9 Elecció de computacions addicionals de l'NRS amb les opcions de robustesa seleccionades. 27
6.10 Captura de la interfície de l'NRS per crear xarxes sintètiques. . . . 28
6.11 Visualització d'una Pipe d'un model complet amb valors numèrics i categòrics. 30
6.12 Esquema d'una xarxa neuronal. [IBM 2020] 34
6.13 Esquema d'un MLP de regressió amb una sola capa interna. [SciKit-Learn 2022] 35

8.1 Visualització d'un *Dataframe* de la llibreria *pandas*. 41

9.1 Gràfic de barres on es poden veure els diferents candidats a l'eix X i els error mitjà a l'eix Y. 46

10.1 Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Betweness centrality*. 54
10.2 Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Closeness centrality*. 54
10.3 Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Eigenvector centrality*. 55

10.4 Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Nodal degree*. . 55
10.5 Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Critical Nodes*. 56
10.6 Esquema de la Pipe a la millora 3. 58
10.7 Gràfic de barres a l'eix X les mètriques del dataset i a l'eix Y la importància relativa sobre la robustesa. Usant un model conjunt pels datasets d'algorismes d'atac dirigit. 59

11.1 Captura de l'execució i temps de còmput de la interfície de prediccions amb P=[0,30], totes les xarxes i tots els tipus d'atac. . . . 62
11.2 Mostra reduïda del contingut del fitxer de sortida de la interfície de prediccions. 64

Índex de taules

9.1 Taula amb els resultats dels candidats inicials implementats. . . . 46

10.1 Taula amb els resultats dels candidats inicials implementat s juntament amb el nou model. 50
10.2 Taula amb els resultats del model a la millora 2. 52
10.3 Taula amb els resultats del model a la millora 3. 57
10.4 Taula amb els resultats del model a la millora 3.2. 58

11.1 Taula amb els temps d'execució dels diferents datasets usant l'NRS. 62

CAPÍTOL 1

Definicions

- Atac (a una xarxa): Fet d'eliminar un o més nodes o arestes d'una xarxa. Un atac genera sempre un subconjunt de la xarxa original, aquesta xarxa resultant tindrà igual o menys connectivitat.
- NRS: (Network Research Simulator) Simulador de xarxes propietat del grup de recerca BCDS, veure apartat 6.1.
- Robustesa: Característica d'una xarxa que reflecteix la capacitat d'aquesta a l'hora continuar funcionant correctament davant de fallades o atacs. La robustesa es calcula usant conceptes de la teoria de grafs, més concretament, de connectivitat de grafs. [Marzo 2019].
- Aprenentatge automàtic o *machine learning*: Tipus d'algorismes d'intel·ligència artificial que permeten que les màquines aprenguin i evolucionin de manera automàtica.
- Aprenentatge supervisat: És una subcategoria del *machine learning* que té per característica el fet que les dades d'entrenament necessiten estar catalogades amb la resposta que s'espera que presentin els models [Saravanan 2018].
- Python: Llenguatge de programació d'alt nivell i de propòsit general enfocat a produir codi comprensible a alt nivell.
- Jupyter Notebook: Entorn de treball que permet treballar amb llibretes que contenen trossos de codi executables de manera independent. Suporta diversos llenguatges però principalment es fa servir per treballar amb Python. Més detalls a la secció 6.2.
- SciKit Learn: Llibreria de Python que encapsula múltiples eines per desenvolupar projectes d'intel·ligència artificial. Incorpora des d'eines per preparar les dades, per fer avaluació, els propis models i algorismes entre d'altres. Per més detalls veure la secció 6.3.
- *dataset*: Conjunt de dades, en aquesta memòria fa referència als conjunts usats per entrenar o validar els models usats.

- *dataset* d'entrenament i validació: Els *datasets* es solen dividir en subconjunts per poder obtenir un més alt nivell d'independència a l'hora d'avaluar els models d'intel·ligència artificial. El *dataset* d'entrenament és el que es fa servir per entrenar el model mentre que el de validació es fa servir per avaluar-lo.
- Sobreentrenament o sobreajustament: Condició d'un model d'intel·ligència artificial que correspon amb massa precisió un subconjunt particular de les dades i que, per tant, pot no ajustar-se bé a les dades noves fent que el model no sigui capaç de produir observacions futures de manera confiable [Oxford 2022].
- Coeficient de determinació o R^2: És una mesura estadística que representa la proporció de la variància per una variable no independent que es pot explicar per una o més variables independents en un model de regressió. Si l'R^2 d'un model és 0.75 llavors el 75% de les variacions observades es poden explicar a partir de les dades d'entrada del model.
- Aprenentatge profund, xarxes neuronals: Algorismes d'intel·ligència artificial que funcionen simulant un sistema per capes semblant al funcionament bàsic del cervell humà. Veure apartat 6.4.6.
- Escalat de dades: Tècnica que s'aplica sobre un conjunt de dades numèriques amb l'objectiu que estiguin dins un rang establert, les transforma. D'aquesta manera es poden evitar desequilibris als models quan es treballa amb característiques que tenen rangs numèrics molt diferents.
- Codificat o *encodding* de dades: Tècnica que transforma un conjunt de dades no numèriques en dades numèriques. Una manera de fer-ho és creant noves característiques, una per cada mostra diferent de dada no numèrica i assignar valors a aquestes en funció de la mostra, això es coneix com a codificació en calent o *hot encodding*.
- Serialització: Mètode que permet transformar un objecte o conjunt de dades en un format fàcilment exportable.
- NP-completesa: En termes de complexitat computacional, són el conjunt de problemes que pertanyen tant a NP com a NP-hard, és a dir són problemes que es poden verificar fàcilment però que trobar-ne la solució és requereix d'un algorisme computacionalment complex.
- Algorisme *greedy* o voraç: Família d'algorismes que per resoldre un problema d'optimització fa diverses iteracions intentant a cada pas arribar a la solució òptima del problema. A cada iteració pren la millor decisió possible i un cop s'ha fet una elecció no la reconsidera en passos futurs. Pot

ser, per tant, que no s'arribi a la solució més òptima del problema.

CAPÍTOL 2

Introducció

2.1 Motivació

La raó principal per escollir aquest projecte és la possibilitat de poder aplicar els coneixements adquirits en els dos àmbits que més m'han influït durant la carrera, la intel·ligència artificial i les xarxes.

Addicionalment, poder aportar millores en el simulador del grup de recerca que m'ha acollit, el BCDS, és un altre dels arguments que em va fer decantar per aquest treball. Poder generar un o varis model de càlcul ràpid de robustesa permetria obrir les portes a la possibilitat d'aconseguir un recomanador que permetria millorar la qualitat de les xarxes al dissenyar-les.

2.2 Propòsit

El propòsit d'aquest projecte consisteix en aconseguir obtenir un o varis models d'intel·ligència artificial que permeti calcular de manera ràpida i precisa la robustesa que té una xarxa. Durant aquest treball s'estudiaran, provaran i compararan diferents algoritmes d'aprenentatge supervisat amb aquest objectiu.

L'abast que s'estableix contempla l'estudi de l'estat de l'art d'aquest camp, la selecció dels models més adequats com a candidats, la comparació i millora d'aquests i la obtenció d'un o varis models finals. Aquest projecte es farà exclusivament amb el llenguatge de programació Python utilitzant implementacions dels algoritmes trobats durant l'estudi inicial.

2.3 Objectius

En aquest Treball de Final de Grau es pretén assolir els següents objectius:

1. Estudiar l'estat de l'art del camp d'intel·ligència artificial, més concretament de l'aprenentatge supervisat.

2. Desenvolupar models d'intel·ligència artificial que permetin calcular la robustesa de xarxes.
3. Estudiar si generant models específics per cada atac millora la precisió dels càlculs.
4. Comparar empíricament el cost benefici d'usar aquests models respecte els algoritmes clàssics.
5. Crear una interfície per poder usar els models i fer prediccions de manera senzilla.

CAPÍTOL 3

Viabilitat

La viabilitat d'aquest projecte depèn de disposar dels següents recursos.

3.1 Recursos informàtics

- Per tal de poder obtenir les dades de robustesa real, és necessari el simulador NRS del grup BCDS. Veure 6.1.
- Per a poder importar dades i entrenar models d'intel·ligència artificial serà necessari un hardware especialitzat. En aquest projecte s'ha fet servir el *Seeder*, un servidor amb Jupyter Notebook [Granger 2021] del grup BCDS específic per aquest tipus de projecte. Aquest servidor té el següent hardware:
  - CPU: AMD Ryzen 5 5600X 6-Core Processor
  - RAM: 32 GB DDR4
  - Disc: 1 Tb SSD NVME4
  - GPU: 2x NVIDIA GeForce GTX 1070

  Aquest servidor està dissenyat per tal que es faci us de la potència de les targetes gràfiques, no obstant, en aquest projecte no han sigut necessàries.
- Software: En tot el projecte s'han fet servir eines open source, self-hosted i sense cost de llicència.

3.2 Recursos humans

En aquest projecte no es preveu la intervenció de cap expert extern ni de personal especialitzat, per aquest motiu, no cal tenir en compte aquest apartat.

3.3 Viabilitat econòmica

Al no requerir de recursos amb cost econòmic podem garantir la viabilitat econòmica del projecte. No obstant, es pot fer una estimació del cost que tindria aquest projecte en cas d'haver de llogar un hardware similar a l'especificat a l'apartat 3.1 i, addicionalment, es pot comptabilitzar el cost que tindria contractar un programador tenint en compte les hores de treball previstes.

- Hores programador informàtic: 200h * 14€/h = 2800€ [1]
- Hores lloguer hardware: 100h * 0.506€/h = 50.60€[2]

Amb aquests dos conceptes es pot estimar un total de 2850.60€ per la realització d'aquest projecte.

En aquesta estimació no s'ha tingut en compte ni les hores dedicades a documentar ni les dedicades a l'estudi de l'estat de l'art. Tampoc s'ha comptabilitzat el cost de l'obtenció dels *datasets* pel fet que l'NRS és un simulador obert sota demanda i com a tal no té cost econòmic generar els experiments.

[1] El preu per hora s'ha obtingut del portal talent, un web especialista en comparar salaris. S'ha tingut en compte el salari per hora mitjà d'un enginyer informàtic a l'estat espanyol.

[2] Per obtenir aquest preu s'ha tingut en compte les característiques del hardware utilitzat i buscat un proveïdor d'un servei similar per hores. El preu surt del proveïdor adquirint un servidor de 192GB de RAM i 48 nuclis.

CAPÍTOL 4

Metodologia

Durant la realització d'aquest treball, sobretot en referència la part pràctica, s'ha utilitzat una metodologia de tipus *Agile* anomenada *Scrum*.

Les metodologies *Agile* són un conjunt de valors i principis que presenten una manera d'encarar el desenvolupament de programari [RedHat 2020].

4.1 Scrum

Scrum és un metodologia que té per objectiu principal ajudar als equips a treballar junts, promou que els equips aprenguin a través d'experiències a autoorganitzar-se al treballar en un problema i a reflexionar sobre els resultats. Aquest sistema permet entregar valor al client continuament, és un entorn, una manera de treballar [Rehkopf 2020].

Aquesta metodologia permet adaptar-se millor als canvis de requeriments, objectius o altres situacions. Amb *Scrum*, el programari es desenvolupa a base d'iteracions anomenades *Sprints*. El projecte es divideix en petites parts cosa que permet treballar de manera més senzilla, en totes les iteracions s'entrega un codi funcional permetent que es pugui rebre comentaris i modificar o adaptar els següents *Sprints*.

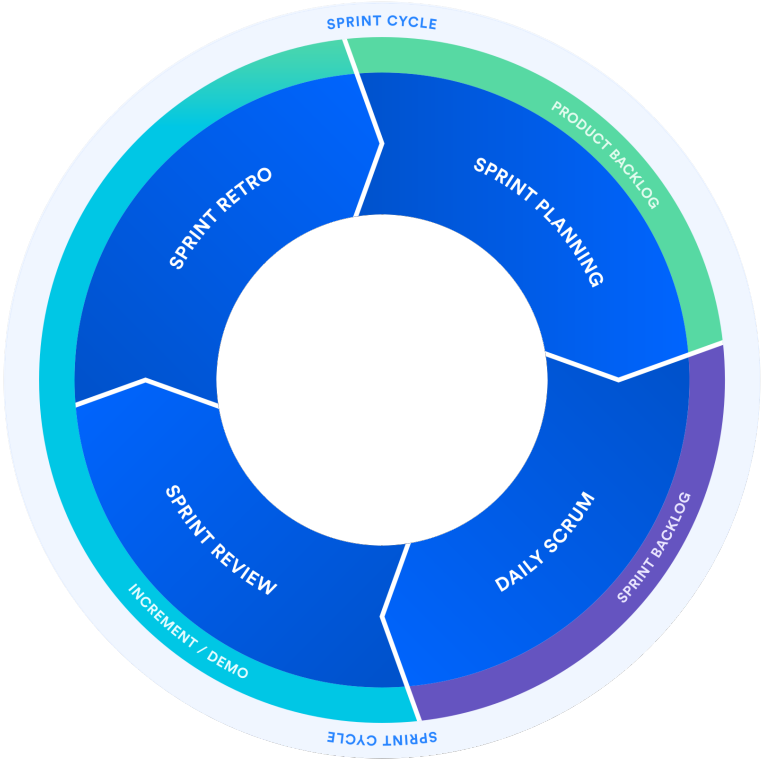

Figura 4.1: Esquema del funcionament d'*Scrum*.

4.2 Aplicació al projecte

En aquest projecte s'utilitza aquesta metodologia com a marc de desenvolupament, no obstant, al ser un projecte individual no s'assignen rols ni es treballen les altres característiques de l'*Scrum*.

L'aplicació al projecte ha sigut la següent:

- Dividir els objectius en tasques.
- Planificació i temporització de tasques de la iteració.
- Implementació de tasques.
- Revisió de resultats, modificació si escau dels objectius i tasques següents.
- Si el projecte no està acabat, tornar al segon pas.

CAPÍTOL 5

Planificació

En aquest capítol es detallarà la planificació seguida a l'hora de desenvolupar aquest projecte.

5.1 Etapes del projecte

5.1.1 Etapa 1: Planificació del projecte

En aquesta primera etapa es planifica la temporització del projecte, en concret, les tasques realitzades han sigut:

1. Crear el primer esbós de les tasques a realitzar.
2. Reunió amb el tutor i aprovació.
3. Modificar l'esbós a partir dels comentaris per obtenir l'esquema de tasques.
4. Temporitzar les tasques i.e. estimar quina durara temporal tindrà cada tasca.

5.1.2 Etapa 2: Estudi de l'estat de l'art

En aquesta segona etapa, l'objectiu va ser investigar quines solucions existien per el problema que es vol solucionar. Es van analitzar diferents candidats potencials i amb l'ajuda del tutor es van definir els candidats definitius del projecte així com quines implementacions es farien servir.

5.1.3 Etapa 3: Obtenció dels datasets i preparació

En aquesta tercera etapa es van obtenir els datasets necessaris per poder entrenar els models triats a l'etapa anterior. Addicionalment, es van fer eines de preparació dels datasets que permetien treure valors fora de rang o convertir i codificar camps entre d'altres específics del dataset existents.

5.1.4 Etapa 4: Implementació i anàlisi dels candidats

Un cop obtinguts els datasets i sabent els candidats inicials a implementar, es procedeix a fer les primeres tasques de programació de models d'intel·ligència artificial.

En aquesta etapa s'obtenen els primers resultats i es pot començar a veure com es desenvolupen els candidats amb els datasets inicials.

5.1.5 Etapa 5: Millores als models i als datasets, elecció del model definitiu

Durant aquesta fase s'implementen millores als models i datasets existents també es fa un estudi de la necessitat d'usar més d'un model o es poden unificar.

Finalment es fa una tria de quins models es faran servir per fer el càlcul de robustesa.

5.1.6 Etapa 7: Anàlisi de característiques

Un cop obtingut el model definitiu, en aquesta etapa, es fa un anàlisi de quines són les característiques més importants d'una xarxa a l'hora de determinar-ne la robustesa amb l'objectiu de saber si coincideixen amb els resultats previstos i poder fer millores.

5.1.7 Etapa 8: Exportació i tasques finals

En la última fase de desenvolupament es creen eines per exportar els candidats definitius i interfícies per poder usar-los en entorns reals.

5.1.8 Etapa 9: Redacció memòria

En l'etapa final s'acaba la documentació del codi i es redacta la present memòria.

5.2 Diagrama de Gantt

A l'hora de planificar aquest treball s'ha utilitzat com a base un diagrama de Gantt. Aquesta eina proporciona una vista general de les tasques programades i de la temporització a nivell de tasca i etapa. El diagrama usat durant aquest projecte es pot visualitzar a la figura 5.1.

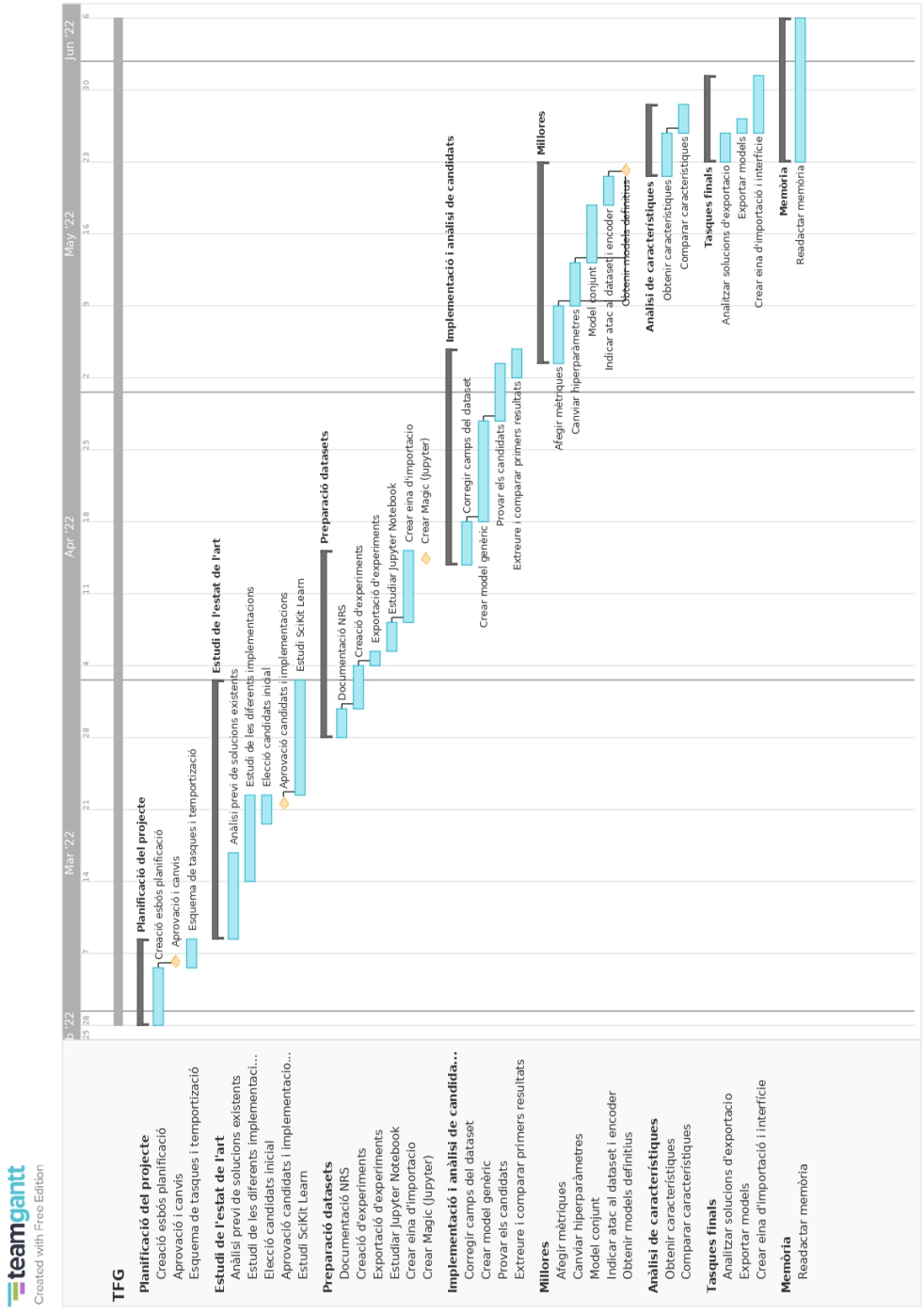

Figura 5.1: Diagrama de Gantt

CAPÍTOL 6

Marc de treball i conceptes previs

6.1 NRS

L'NRS és un simulador de xarxes propietat del grup de recerca BCDS. Amb aquesta eina es poden visualitzar aquestes xarxes, calcular-ne mètriques i calcular la robustesa que presenten davant de certs atacs.

Aquest simulador té una estructura pensada per ser fàcilment escalada, a la figura 6.1 si pot veure un esquema abstracte de la composició.

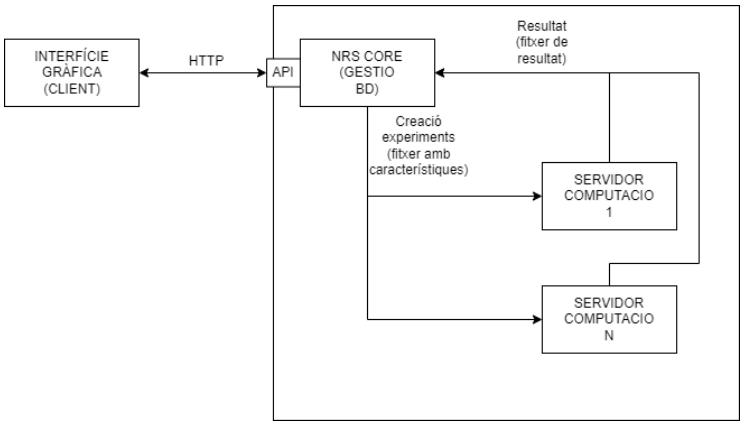

Figura 6.1: Esquema de l'estructura del simulador NRS.

El simulador, com es pot apreciar, es pot escalar fàcilment afegint servidors de computació. El propi nucli ja està programat per fer balanceig de càrrega i els experiments són paral·lelitzables en gran mesura.

Actualment el servidor NRS consta de dos servidors:

- Servidor 1: Conté el nucli i un servidor de computació.
- Servidor 2: Conté un servidor de computació.

Els dos servidors tenen les mateixes característiques de maquinari:

- Processador: Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz (32 nuclis)
- RAM: 256 GB.

- Disc: 2 TB disc dur.

6.1.1 Experiments amb l'NRS

La robustesa és un valor numèric que s'obté al simular atacs sobre una xarxa. Aquests atacs tenen certes característiques que es poden definir a l'hora de fer les simulacions:

- Tipus d'atac: aleatori o amb algoritme.
- Element a atacar: nodes o arestes.
- Càlcul dels elements a atacar: simultani o progressiu.
- Percentatge d'atac: valor numèric fixe o amb rang.
- Nombre d'atacs: Valor enter.

El tipus d'atac aleatori selecciona a cada iteració els objectius de manera aleatòria mentre que els atacs amb algoritme dirigit usen una fórmula concreta per decidir quins elements seran atacats.

D'algoritmes d'atac, actualment, n'hi han 5 d'implementats i varien en funció de quina mètrica dels nodes es prioritza a l'hora de seleccionar els que seran atacats.

1. Betwetness centrality
2. Closeness centrality
3. Eigenvector centrality
4. Nodal degree
5. Critical nodes

Els quatre primers algorismes prioritzen atacar els nodes que tenen un valor més alt de la mètrica de la que tenen nom, per exemple, el *Nodal degree* es centrarà en atacar els nodes amb més grau nodal. Els detalls de cada mètrica es poden consultar a l'apartat 6.1.2.

En el cas del cinquè algorisme, el *Critical Nodes*, a diferència dels anteriors, no funciona prioritzant nodes basat en una sola mètrica sinó que utilitza un algorisme diferent que prioritza atacar els nodes més crítics, és a dir, aquells que de ser atacats minvarien més la connectivitat general de la xarxa [Calle 2021]. Calcular els nodes crítics és un problema típic del camp de teoria de grafs i és computacionalment costós. En el cas de l'NRS s'utilitza un algorisme

greedy que usa diverses mètriques per obtenir una aproximació dels nodes més crítics.

Pel què respecte al càlcul dels elements a atacar, la diferència entre simultani i progressiu és que en aquest últim cas, es recalculen els elements a cada iteració mentre que al simultani es fa un sol cop a l'inici de l'experiment.

El nombre d'atacs ens permet fixar quantes vegades es repetirà l'experiment. Al tenir diversos experiments de les mateixes característiques es pot obtenir un valor de la robustesa més acurat.

Durant la realització d'aquest projecte s'ha usat un conjunt de grafs que ja existeix dins el simulador, correspon al dataset Topology Zoo [Knight 2011] i són un conjunt de xarxes de telecomunicacions que els operadors han fet públiques i accessibles.

Amb aquesta eina podem visualitzar les xarxes, calcular-ne mètriques i simular-hi atacs de diferents tipus per veure quina robustesa presenten. A la figura 6.2 podem un exemple de visualització d'una xarxa real sobre un mapa. Aquest visualitzador permet fer visionats 3D i 2D sense mapa així com filtratge i coloració en temps real de nodes.

Figura 6.2: Visualitzat de la xarxa BSONETEUROPE sobre el mapa real.

6.1.2 Mètriques d'una xarxa

Les mètriques amb les que treballen l'NRS són les que es poden obtenir d'un graf degut a que aquestes són les estructures que accepta el simulador. El recull

d'aquestes mètriques es pot veure a la taula 6.3.

Acronym	*Name*	*Symbol* *Units*	*Description*	*Computation*
ND Nodal degree				
00a-NN	Number of nodes	natural		
00b-NL	Number of links (edges)	natural		
01-ND	Nodal Degree	array	The degree of a node is the number of incident edges	
	01-AND Average Nodal Degree (aka Degree Centrality)	real+	The degree of a node is the number of incident edges. Thus, AND is the average of all node degrees	1) From adjacency matrix A 2) $AND = \frac{2N}{\#L}$ 3) Needs to compute 01-ND: AND = mean(01-ND)
	01m-MND	natural	Maximum ND	Needs to copmute 01-ND: max(01-ND) Special case: complete graph. The maximum number of links is: $L_{max} = \frac{N(N-1)}{2}$ then its 01-AND $= N-1$
	01h-HND Histogram ND	array	distribution of nodal degrees 0, 1, 2, ...	Needs to compute 01-ND
02-HET	Heterogeneity	real	It is the standard deviation of the nodal degree divided by the average nodal degree	Needs to compute 01-ND $HET = \frac{sd(deg(G))}{mean(deg(G))} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - x^{mean})^2}$ where x_i is the ND of node i
Shortest Path Length				
03-ASPL	Average Shortest Path Length	real	Average shortest path length between every node pairs of the network	Needs to compute shortest distance between all pair of nodes $ASPL = \frac{2}{n(n-1)}\sum_{i=1}^{n}\sum_{j=i+1}^{n} d_{ij}$ where d_{ij} is the distance between node i and node j
04-DIA	Diameter	$\emptyset$ natural	Maximum shortest path length between every node pairs of the network	Needs to compute shortest distance between all pair of nodes
05-EFF	Efficiency	real+	This is the averaged sum of the inverse of all the distances (length of the shortest path) between all pairs of nodes (conductance)	Needs to compute shortest distance between all pair of nodes $E = \frac{2}{n(n-1)}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{1}{d_{ij}}$
Laplacian				

Acronym	*Name*	*Symbol* *Units*	*Description*	*Computation*
06-ER	Effective (Graph) Resistance	real+	This considers the graph as an electrical circuit, where an edge is a resistor. The effective resistance between two nodes can be calculated by series and parallel manipulations (Kirchhoff's Current Law) Laplacian [**ELLENS20112491**] [**Samui2016**]. ER is the sum of the effective resistances over all pairs of vertices	Needs to compute 06a-LE $R = \sum_{1 \leq i = j \geq n} Rij = n \sum_{i=2}^{n} \frac{1}{\lambda_i}$
	06a-LE Laplacian Eigenvalues	array (real)		
07-NST	Number of Spanning Trees	ξ natural (large)	A spanning tree is a subgraph containing $N-1$ edges, all N nodes and no cycles ξ is a function of the unweighted Laplacian eigenvalue	Needs to compute 06a-LE $\xi = \frac{1}{n} \prod_{i=2}^{n} \lambda_i$
08-CC	Clustering Coefficient	real	CC measures the probability that the adjacent nodes of a node are connected (among them). CC captures the presence of triangles and compares it to the number of connected triplets	$CC = \sum_{i=1}^{N} \frac{2e_i}{k_i(k_i-1)}$
09-r	Assortativity	r real $[-1, 1]$	r is the preference for a network's node to attach to others which have a similar nodal degree	$r = \frac{M^{-1} \sum_i j_i k_i - [M^1 \sum_i \frac{1}{2}(j_i + k_i)]^2}{M^{-1} \sum_i \frac{1}{2}(j_i^2 + k_i^2) - [M^1 \sum_i \frac{1}{2}(j_i + k_i)]^2}$
Eigenvalues				
10-EV	EigenValues	array (real)	Eigenvalues of the graph (obtained from the adjacency matrix)	$Ax = \lambda x$ the number of eigenvalues is the dimension of the adjacency matrix (number of nodes N)
	10-SR Symmetry Ratio	real	SR is the quotient between the number of distinct eigenvalues of the graph (obtained from the adjacency matrix) and the diameter	Needs to compute shortest distance between all pair of nodes Needs to compute 10-EV
11-LE	Largest Eigenvalue (aka Spectral Radius)	real	Largest Eigenvalue of the adjacency matrix)The higher the more robust. This is also associated in defining the epidemic threshold	Needs to compute 10-EV $LE = \lambda_1$
Connectivity				
12-LCC	Largest Connected Component	natural	LCC is the number of nodes of the largest connected component of the network. This metric is a measure of the global connectivity of the network	Needs to compute the graph's components

Acronym	*Name*	*Symbol Units*	*Description*	*Computation*
	12a-ALCC Analytical LCC	real	Estimation (see Robert docs)	Needs to compute 01-ND
13-FSLC	Fractional Size Largest Component	real	Fraction of nodes of the largest connected component of the network. This metric is a measure of the global connectivity of the network	Needs to compute the graph's components
14-ATTR	Average Two Terminal Reliability	real	ATTR is the probability that a randomly chosen pair of nodes remains connected after a failure. It can be computed as the sum over the number of node pairs in the connected component divided by the total number of node pairs (according to a ratio of removed nodes)	Needs to compute the graph's components
	14lw-ATTR Lower Bound	real		Needs to compute the graph's components
	14ub-ATTR Upper Bound	real		Needs to compute the graph's components
15-DF	Degree of Fragmentation	natural	Number of connected components (islands)	Needs to compute the graph's components
16-EC	Edge (Link) Connectivity	natural	The minimal number of edges which has to be removed to disconnect the graph (aka Resilience Factor)	
17-VC	(Vertex) Nodes Connectivity	natural	The minimal number of nodes which has to be removed to disconnect the graph	
18-AC	Algebraic Connectivity	real real $[0, 2]$	It measures how difficult it is to break the network into disconnected components. Higher values indicates better robustness against both node and link removal. Graphs with identical algebraic connectivity can be compared using natural connectivity	Needs to compute 06a-LE It is defined as the second smallest Laplacian eigenvalue $AC = \mu_{n-1}$
19-NC	Natural Connectivity	real	It is based on the number of closed walks (cw) in a graph, that can be related to the sum of eigenvalues. A walk of length k is a path over the nodes and links of the graph, starting at v0, passing k - 1 nodes and k links to end in vk. If v0 = vk this path is called a closed walk	Needs to compute 10-EV $\lambda = ln(\frac{S}{n}) = ln(\frac{\sum_{i=1}^{n} e^{\lambda_i}}{n})$
			Centrality	
20-DC	Degree Centrality	real	To be described	

Acronym	*Name*	*Symbol Units*	*Description*	*Computation*
21-NBC	Node Betweenness Centrality	real	This measures the fraction of shortest paths that pass through a given node, averaged over all pairs of node in a network	
22-EBC	Edge Betweenness Centrality	real	This measures the fraction of shortest paths that pass through a given link	
	22M-EBC Maximum EBC			
23-CLC	Closeness Centrality	real [0,2]	This measures the degree to which a node is close to other nodes on average considering shortest paths (per node).	$CLC = \sum_i \frac{1}{\sum_j shortestpath(i,j) \forall i \neq j}$
24-EC	Eigenvector Centrality	real-array (JLM)	It is based on the idea that if an important node is connected to important neighbors. The EC of a node is equal to its component of the eigenvector corresponding to the largest eigenvalue of the adjacency matrix	It can be calculated by iterations. $x_i(t+1) = \sum_{j=1} A_{ij} x_j(t)$
End of Table				

Figura 6.3: Taula amb les mètriques d'una xarxa.

A la taula 6.3, podem veure les següents columnes:

1. Acrònim de la mètrica, durant la present memòria es fan servir per fer-hi referència.
2. Nom de la mètrica en anglès.
3. Unitats o estructura esperada.
4. Descripció breu de cada mètrica.
5. Mètode de computació de la mètrica o fórmula per obternir-la.

La taula prové de la documentació del grup de recerca BCDS [BCDS].

6.1.3 Robustesa d'una xarxa

En termes de xarxes, la robustesa es pot definir com la capacitat d'una xarxa de continuar funcionant correctament davant de fallades o atacs. La robustesa es calcula usant conceptes de la teoria de grafs, més concretament, de connectivitat de grafs. [Marzo 2019].

Una mètode de càlcul d'aquesta mètrica podria ser el proposat per Trajanovski et alt. [Trajanovski 2013]:

$R = \sum_{n}^{k=1} s_k t_k$

On s_k representa el pes de la mètrica i t_k el valor de la mètrica k.

Aquest mètode presenta varis problemes:

- Com es seleccionen les mètriques més rellevants d'un graf.
- Quin pes té cada mètrica en el còmput d'R.

Per solucionar-ho el grup de recerca BCDS va fer un anàlisi en detall de quines mètriques són les més importants a l'hora de definir la robustesa d'una xarxa [Marzo 2018].

Aquest anàlisi va tenir en compte inicialment totes les mètriques vistes a la subsecció 6.1.2, aquestes es van anar descartant en fases que van tenir en compte l'escalabilitat, la variança de les mètriques i la relació entre elles. De les 24 mètriques inicials van quedar 10 mètriques que són les que actualment es fan servir per calcular la robustesa amb l'NRS.

Les mètriques resultants van ser:

- 01 - Average Nodal Degree: Grau nodal mitjà.
- 05 - Efficency: Eficiència.
- 11 - Largest Eigenvalue: Valor propi més gran.
- 12 - Largest Connected Component: Component més gran connex.
- 14 - Average Two Terminal Reliability: Mitjana de fiabilitat entre dos nodes.
- 18 - Algebraic Connectivity: Connectivitat algebraica.
- 19 - Natural Connectivity: Connectivitat natural.
- 22 - Edge Betweeness Centrality: Centralitat entre arestes.
- 23 - Closeness Centrality: Centralitat de proximitat.
- 24 - Eigenvector Centrality: Centralitat de vectors propis.

Les descripcions d'aquestes mètriques es poden trobar a la taula 6.3. A aquestes mètriques se les anomena mètriques recomanades en el context del simulador.

Un cop seleccionades les mètriques que es tindran en compte per calcular la robustesa cal saber els pesos que tindrà cada mètrica en el càlcul. El grup BCDS va proposar realitzar un anàlisi de component principal (PCA per les seves sigles en anglès). Amb aquest anàlisi podem obtenir els pesos de manera automatitzada i obtenir la robustesa de cada xarxa.

Internament, el càlcul de robustesa està implementat usant la següent metodologia:

- Per calcular la robustesa d'una xarxa un cert percentatge de nodes s'elimina de manera seqüencial fins a un valor P ($p = [0, P)$). M experiments independents es fan per cada p, M correspon al nombre d'atacs.
- Es considera el graf inicial $p = 0$ com el més robust doncs no ha patit cap atac, els successius grafs $p > 0$ sempre resultaran en una robustesa inferior. Per a propòsits comparatius, el valor de robustesa per p=0 es normalitza a 1, els grafs atacats tindran un valor inferior a aquest.
- Es computen les mètriques del graf M vegades per $p = 0$ que donaran el mateix resultat doncs la xarxa no ha patit cap atac encara ($m = [0, M]$).
- A continuació es determina el conjunt de nodes a ser atacats i eliminats. Aquí cal tenir en compte que en funció de l'algoritme d'atac (veure subsecció 6.1.1) es triaran nodes diferents.

- Es calculen les N mètriques per cada parell (m, p) obtenint la matriu $PxMxN$. A partir de la matriu de correlació de les mètriques es fa l'anàlisi PCA i s'obté el pes de cada mètrica a l'hora de calcular la robstesa.
- Finalment es normalitzen els resultats del PCA i s'obté el valor de la robustesa per cada parell (pxm).

Aquest procediment està implementat de manera que es pot usar paral·lelisme per optimitzar el càlcul tot i que és molt costós quan es treballa amb *datasets* grans, en alguns casos el temps de còmput pot ser de l'ordre de varis dies.

El procés pràctic per realitzar un càlcul automatitzat de robustesa amb l'NRS és el següent:

1. Partim de la pàgina principal del simulador, s'hi arriba just després d'iniciar sessió.
2. Ens dirigim a la capçalera de la pàgina i seleccionem la pestanya *EXPERIMENTS*, veure figura 6.4.
3. Seleccionem *METRICS* com a tipus d'experiment, veure figura 6.5
4. Seleccionem les xarxes amb les que es vol experimentar i premem *NEXT*, veure figura 6.6
5. Escollim les característiques de l'experiment explicades a la secció 6.1. Un cop escollit premem *NEXT* de nou, un exemple de configuració es pot veure a la figura 6.7.
6. Seleccionem a continuació quines mètriques volem que es calculin sobre les xarxes. Aquestes estan explicades a la subsecció 6.1.2 però és recomanable usar les predefinides que es poden seleccionar fàcilment prement el botó *SELECT RECOMENDED METRICS*, veure figura 6.8. A continuació premem *NEXT*.
7. Finalment escollim com a càlcul extra les opcions *Compute PCA* i *Compute* R^* per tal que ens calculi la robustesa, veure figura 6.9, per iniciar l'experiment cal prémer el botó *SPAWN EXPERIMENT* i si ho desitgem podrem posar un nom a l'experiment per poder-lo recuperar fàcilment.

Un cop acabat l'experiment podrem descarregar cada simulació, és a dir, cada xarxa sobre la que hem experimentat, en format *EXCEL* o *JSON*, on podrem veure quin valor de robustesa s'ha obtingut.

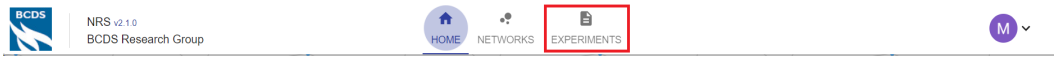

Figura 6.4: Capçalera de l'NRS, pestanya experiments emmarcada en vermell.

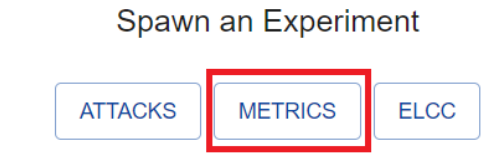

Figura 6.5: Selecció d'experiments de l'NRS. Opció *METRICS* emmarcada en vermell.

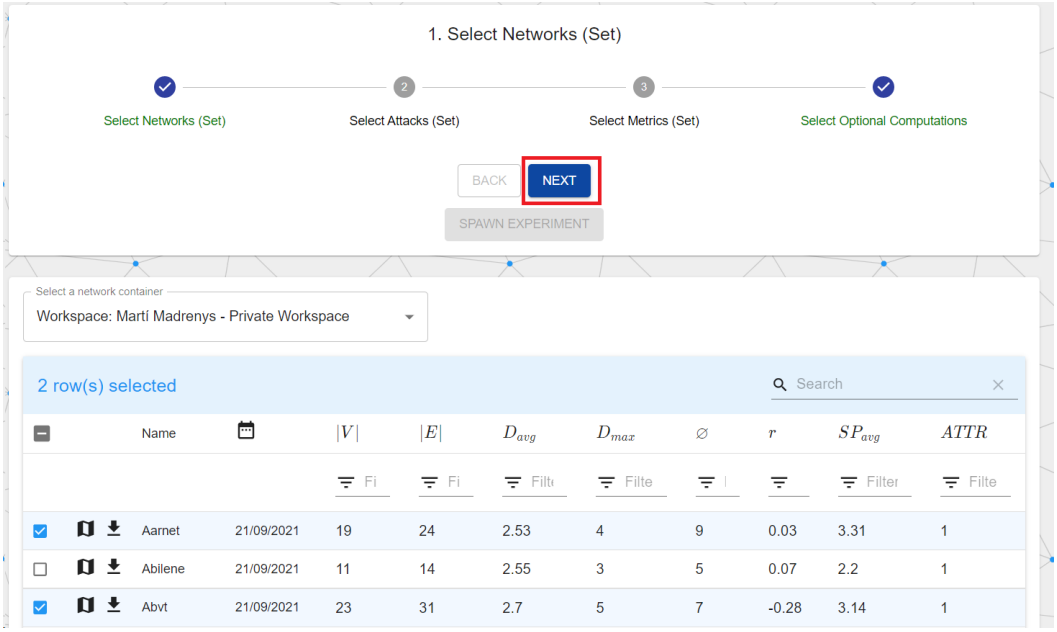

Figura 6.6: Selecció de xarxes per fer l'experiment, s'han seleccionat algunes xarxes i emmarcat l'opció next en vermell.

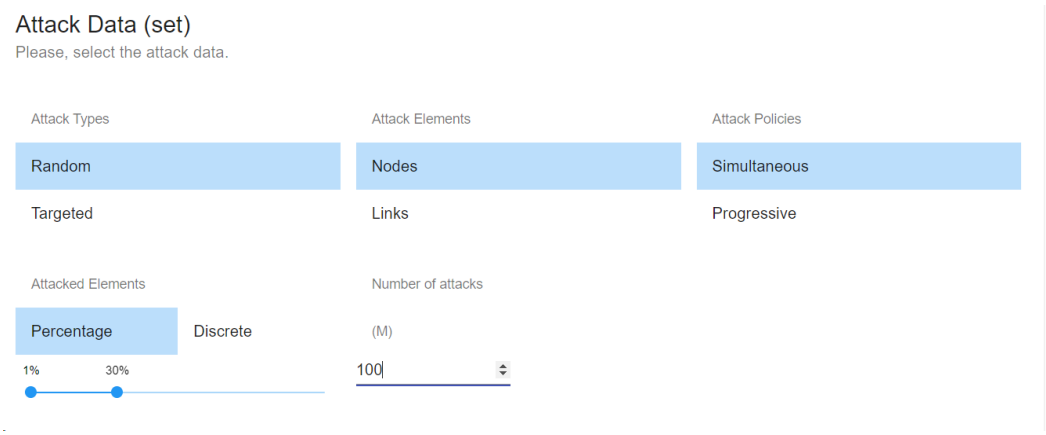

Figura 6.7: Elecció de les característiques d'un experiment amb l'NRS.

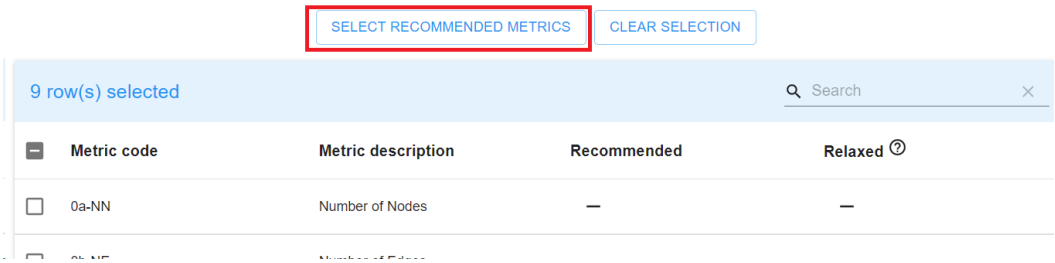

Figura 6.8: Selecció de mètriques d'un experiment amb l'NRS, emmarcada la opció per usar les recomanades.

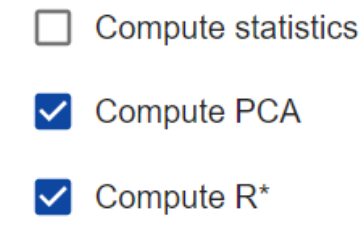

Figura 6.9: Elecció de computacions addicionals de l'NRS amb les opcions de robustesa seleccionades.

6.1.4 Xarxes Sintètiques

Per poder treballar amb l'NRS es necessiten xarxes amb les que experimentar. Actualment les xarxes poden provenir de tres llocs:

1. Xarxes que incorpora per defecte el simulador. És el cas de les xarxes del *Topology Zoo*, un *dataset* de xarxes de telecomunicacions [Knight 2011].
2. Xarxes que l'usuari ha penjat a l'NRS a través del web. El simulador accepta diversos formats d'arxiu.
3. Finalment, la tercera manera d'aconseguir xarxes és que l'usuari en creï, obtenint xarxes sintètiques. El simulador té un apartat específic amb interfície pròpia per a aquest propòsit.

Les xarxes sintètiques són doncs xarxes creades artificialment per l'usuari a partir de certes certes característiques que cal especificar.

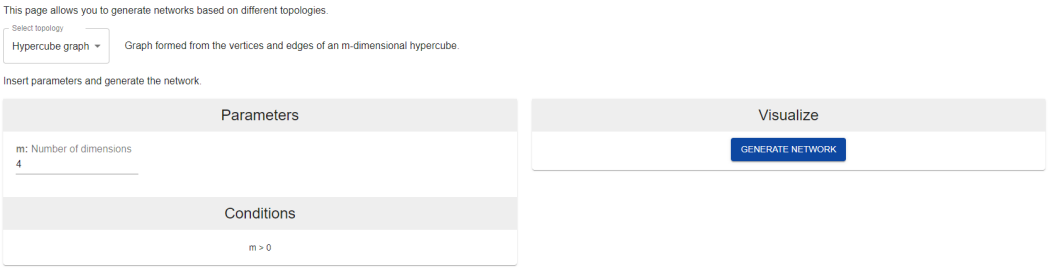

Figura 6.10: Captura de la interfície de l'NRS per crear xarxes sintètiques.

A la figura 6.10 es pot veure la interfície que l'NRS proporciona per a crear xarxes sintètiques. En primer lloc es demana la topologia de les xarxes que es volen crear, a continuació, en funció de la topologia cal entrar els paràmetres desitjats. En aquest cas, per la topologia d'*hipercub*, que és la seleccionada, només cal un sol paràmetre que correspon al nombre de dimensions que tindrà.

Addicionalment, es pot veure que hi ha un apartat de condicions, aquest apartat fa referencia a certes condicions que han de tenir els paràmetres, en aquest cas, és necessari que l'hipercub, com a mínim, tingui una dimensió ($m > 0$).

Finalment, per generar la xarxa cal prémer el botó *Generate Network* obtenint així una nova xarxa artificial amb la que es podrà treballar.

6.2 Jupyter Notebook

Jupyter Notebook és una eina del projecte Jupyter [Granger 2021] basada en web que proporciona un entorn de desenvolupament on es treballa amb llibre-

tes.

Cada llibreta consta de cel·les que es poden crear, editar i eliminar a discreció. Aquestes contenen text o codi i es poden executar en qualsevol ordre produint per cada cel·la una possible sortida. A cada execució se li assigna a cada cel·la un número que indica l'ordre en el qual han sigut executades.

Aquest entorn interactiu és ideal per desenvolupar projectes d'intel·ligència artificial, la practica habitual és instal·lar-ho en un servidor amb certa capacitat computacional i poder executar el codi a través d'aquesta eina web.

Per a complir el seu objectiu, aquesta eina treballa sobre un nucli específic. Existeixen diferents nuclis en funció del llenguatge amb el que es vol treballar.

Tot i que aquesta eina permet treballar amb diferents llenguatges de programació, en aquest treball s'ha programat exclusivament amb el llenguatge Python.

6.2.1 Magic

Les comandes *Magic* són una característica implementada per defecte en les distribucions de Jupyter Notebook que corren sobre el nucli de Python i tenen diverses funcionalitats relacionades amb l'eina.

Les funcionalitats següents són una mostra del que permeten fer:

- *%cd*: Permet executar un canvi de directori en l'entorn d'execució.
- *%colors*: Executa un canvi de color de l'output de la llibreta.
- *%debug*: Activa l'entorn de depuració interactiu.
- *%dhist*: Mostra els directoris visitats.
- *%store*: Permet treballar amb dades entre diferents llibretes. És especialment útil pel fet que podem separar la importació de dades i els models en llibretes diferents.

6.3 SciKit Learn

SciKit Learn [Pedregosa 2011] és una llibreria del llenguatge de programació Python enfocada facilitar l'utilització d'algoritmes d'intel·ligència artificial proporcionant implementacions d'aquests a alt nivell entre d'altres eines.

La llibreria implementa quasi tots els algoritmes d'intel·ligència artificial més utilitzats, ja siguin supervisats o no supervisats, de classificació o regressió inclús

d'aprenentatge profund, tot i que no incorpora suport per càlcul amb GPU de moment.

Addicionalment, proporciona eines de preprocessament de dades, transformadors, codificadors, entre d'altres, que es poden combinar de manera òptima usant *Pipes*.

6.3.1 Pipes

Les *Pipes* de la llibreria SciKit permeten crear models que encapsulen totes les fases d'un model d'intel·ligència artificial de manera òptima [den Bossche 2018].

Un exemple d'us d'una Pipe seria encadenar un escalat de dades per les dades numèriques, un codificador per les dades categòriques i un estimador per obtenir un model complet. Aquestes Pipes es poden visualitzar, un exemple es pot trobar a la figura 6.11.

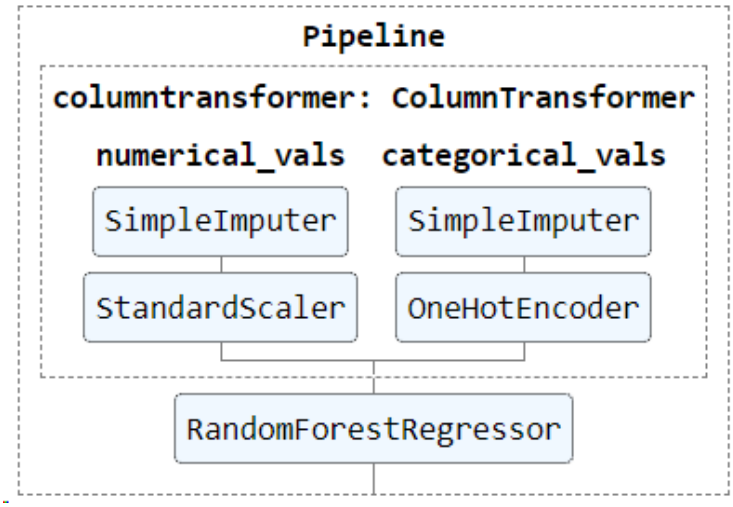

Figura 6.11: Visualització d'una Pipe d'un model complet amb valors numèrics i categòrics.

6.4 Algoritmes d'intel·ligència artificial

En aquesta secció es descriuran els algoritmes d'intel·ligència artificial supervisats que s'han estudiat durant el treball i que són els disponibles a la llibreria utilitzada, veure secció 6.3.

6.4.1 Algoritmes supervisats

Els algoritmes d'intel·ligència artificial supervisats o d'aprenentatge supervisats són una subclasse d'algoritmes que necessiten que les dades amb les que es fa l'entrenament continguin el resultat del que es pretén calcular.

En aquest projecte es treballa amb dades en les que ja sabem el resultat i.e per cada xarxa del dataset en sabem la robustesa, per tant, els algoritmes d'aprenentatge supervisat són els més indicats per aquest problema. [SciKitLearn 2022].

6.4.2 Models Lineals

Els models lineals són mètodes pensats per fer regressió lineal en les que l'objectiu es predir un valor que és la combinació linear de certes característiques (Features).

$$\hat{y}=(w,x)=w_0+w_1x_1+...+w_px_p$$

On $\hat{y}$ és el valor a predir, el vector $w=(w1,\ldots,wp)$ és el vector de coeficients i w_0 l'interceptor i.e. el punt on la funció creua l'eix y.

Algunes de les implementacions més importants disponibles són:

- *Ordinary Least Squares*: Aquest model té per objectiu minimitzar l'R^2 (veure definicions 1) a base d'anar modificant els valors del vector de coeficients $w=(w1,\ldots,wp)$. Aquestes modificacions es fan durant l'entrenament.

  $$\min_w ||Xw-y||\binom{2}{2}$$

- *Ridge Regression*: És una millora de la implementació anterior que busca corregir alguns problemes de l'Ordinary Least Squares penalitzant la mida dels coeficients.

  $$\min_w ||Xw-y||\binom{2}{2}+\alpha||w||\binom{2}{2}$$

  $\alpha \geq 0$ és el paràmetre que controla aquesta penalització.

- *Lasso*: Model lineal que estima els coeficients de dispersió, és útil en contextos en els que es necessiten solucions amb pocs coeficients rellevants, és a dir, pocs coeficients a tenir en compte. Consta d'un model lineal amb un terme de regularització. La funció objectiu és la següent:

  $$\min_w \frac{1}{2n_{samples}}||Xw-y||\binom{2}{2}+\alpha||w||_1$$

6.4.3 Màquines de suport vectorial

Les màquines de suport vectorial son un conjunt d'algoritmes d'aprenentatge supervisat que es poden utilitzar per fer regressió, classificació i detecció de mostres incorrectes.

Les avantatges principals de les màquines de suport vectorial són:

- Efectives sobre datasets amb moltes característiques.
- Efectives amb datasets petits.
- Consumeixen poca memòria.

Com a contrapartida les màquines de suport vectorial tenen problemes de sobreentrenament i no ofereixen un estimador de la certesa quan prenen decisions.

S'utilitzen principalment per fer classificació i detecció de mostres incorrectes però el seu us es pot extendre per fer regressió utilitzant la variant *Regressor de suport vectorial*.

Existeixen 3 implementacions d'aquest model:

- SVR: Model estàndard de regressor de suport vectorial.
- LinearSVR: Model semblant al SVR, internament més senzill però més ràpid d'entrenar.
- NuSVR: Implementació alternativa a l'SVR basada en una altre llibreria.

6.4.4 Veïns propers

Aquest conjunt de models permeten fer tasques de classificació i regressió. El principi bàsic d'aquests models és l'us de dades existents (les d'entrenament) per obtenir les mostres més properes possibles a la que es vol predir i obtenir un resultat a partir d'aquestes.

El nombre de mostres a usar pot ser definit k-veïns o basat en la densitat local de punts. Per mesurar la distància es fa servir l'estàndard de distancia Euclidià.

Les implementacions d'aquest conjunt són:

- KNeighborsRegressor: Versió k-veïns.
- RadiusNeighborsRegressor: Versió basada en la densitat local de punts.

6.4.5 Arbres de decisió

Els arbres de decisió són models d'aprenentatge supervisat que serveixen tant per classificar com per fer regressió. L'objectiu és crear un model que fa prediccions d'un valor objectiu a base d'apendre regles que infereix de les característiques de les dades d'entrenament.

Les avantatges dels arbres de decisió són:

- Fàcils d'interpretar, es poden visualitzar.
- No es necessiten grans quantitats de dades d'entrenament.
- El cost de fer prediccions és logarítmic respecte el nombre de mostres d'entrenament.
- Pot treballar tant amb dades numèriques com categòriques.[1]
- Fàcilment avaluables i permeten extreure les característiques més importants de manera senzilla.

Per contra, aquests models presenten també una sèrie de desavantatges:

- Tendeixen a sobreentrenar-se si no es cuida el mètode d'entrenament. Hi ha maneres de mitigar aquest efecte que les implementacions ja contemplen.
- Petites variacions en les dades poden resultar en arbres completament diferents. En aquest cas es pot mitigar l'efecte usant múltiples arbres com a un únic model creant el que es coneix com a *bosc*.
- Cal tenir cura de les dades d'entrenament, aquestes s'hauran de normalitzar molt bé per poder tenir un bon balanç dels arbres.
- Com que obtenir un arbre òptim és un problema NP-complet, a la pràctica s'han de fer servir algoritmes heurístics per fer els entrenaments.

6.4.6 Xarxes neuronals, models supervisats

Les xarxes neuronals imiten el comportament del cervell humà. Això permet que els computadors puguin reconèixer patrons i solucionar alguns problemes en el camp de la intel·ligència artificial. Aquest conjunt de models són la base del que es coneix com a aprenentatge profund. [IBM 2020]

[1] En la versió actual de SciKit Learn encara és necessari codificar les característiques categòriques.

Una xarxa neuronal està composta de vàries capes de neurones, cada xarxa conté una capa d'entrada, una o més capes internes i una capa de sortida.

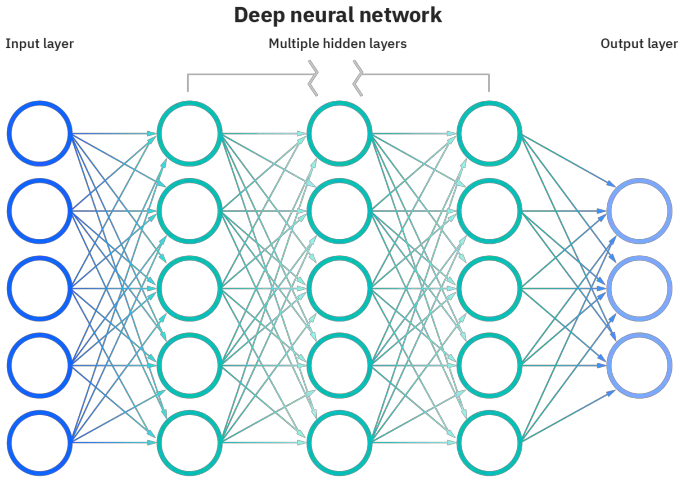

Figura 6.12: Esquema d'una xarxa neuronal. [IBM 2020]

Cada neurona d'una capa es connecta amb cada una de les de la capa següent, aquesta connexió té associat un pes i un llindar. Si la sortida d'una neurona en particular està per sobre del llindar de la connexió, aquesta s'activarà i s'enviaran dades d'una neurona a l'altre.

Hi han diverses implementacions d'aquest model:

- Perceptró multi capa de regressió (MLP): Algoritme d'aprenentatge supervisat que té l'objectiu d'aprendre a estimar una funció no linear de regressió.

  A la primera capa, la d'entrada, hi haurà una neurona per cada característica del dataset. A cada neurona de la capa interna es farà una suma ponderada al pes de la connexió i finalment a la sortida es calcularà la funció d'activació que és hiperbòlica en aquesta implementació. La capa de sortida conté una única neurona (en el cas de regressió) que rep les sortides de la capa interna anterior i obté el resultat de sortida final. Veure figura 6.13

- Implementació equivalent a la anterior però enfocada a solucionar problemes de classificació de dades.

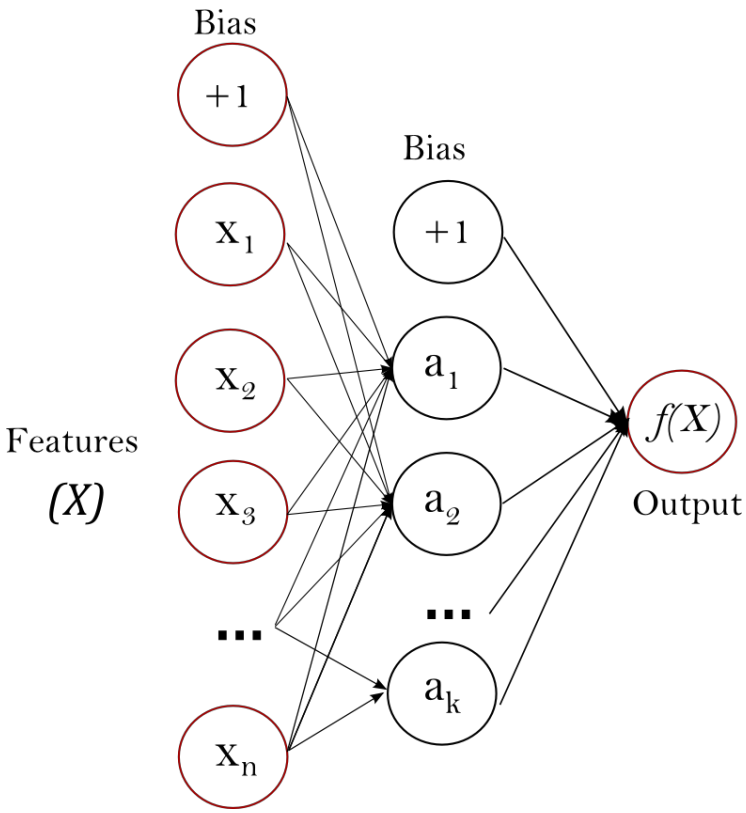

Figura 6.13: Esquema d'un MLP de regressió amb una sola capa interna. [SciKit-Learn 2022]

CAPÍTOL 7

Requisits del sistema

En aquest capítol es descriuran els requisits que s'hauran de complir en el sistema de prediccions.

7.1 Requisits funcionals

Els requisits funcionals d'aquest projecte són:

1. Donades totes les mètriques d'una xarxa (veure subsecció 6.1.2), el percentatge d'atac i el tipus d'atac (veure subsecció 6.1.1), obtenir el valor aproximat de la robstesa de la xarxa.
2. Obtenir les característiques d'una xarxa que més influeixen en el càlcul de la seva robustesa.

7.2 Requisits no funcionals

Els models d'intel·ligència artificials entrenats seran serialitzats per tal de no requerir dependències directes a l'hora de ser usats.

Per fer-ho es farà servir el llenguatge Python i una llibreria de serialització específica del llenguatge. A l'hora d'importar els models per ser usats caldrà només llegir el fitxer on hi ha el model a usar i fer una crida a una funció on caldrà passar les dades necessàries per fer les prediccions.

CAPÍTOL 8

Estudi i decisions

En aquest capítol s'expliquen les principals decisions preses durant el projecte així com els processos que han portat a aquestes decisions.

8.1 Candidats a model

En primer lloc es trien els candidats a implementar per buscar un model que satisfaci els requisits i objectius establerts.

Els candidats triats han sigut:

- *Linear Regression* (Regressió Linear general).
- *K-Nearest Neighbours* (k-veïns propers).
- *Decision Tree / Random forest* (arbres de decisió i boscos).
- *Neural Network* (Xarxa neuronal).

Aquests models estan detallats a la secció 6.4 i són un recull dels principals models usats en aquest tipus de problemes. Tanmateix, en el moment de prendre la decisió no es descarta que si els resultats no són els esperats es puguin incorporar més candidats en un futur.

8.2 Llenguatge, llibreries i entorn

Un cop triats els candidats, es pren la decisió d'utilitzar una llibreria d'alt nivell que ja implementa models per tots els candidats. Aquesta llibreria és la SciKit Learn de Python [Pedregosa 2011], explicada a la secció 6.3.

També es decideix que s'utilitzarà l'entorn Jupyter Notebook explicat a la secció 6.2 pel fet que facilita molt treballar amb funcions pesades i executar petites funcions en models ja entrenats sobre la marxa.

El grup BCDS té un servidor específic per aquest tipus d'experimentació anomenat *Seeder* on s'executaran tots els experiments i els entrenaments dels models.

8.3 Metodologia per avaluar els candidats inicials

Per poder avaluar els candidats inicials cal decidir quina metodologia es farà servir. La llibreria SciKit Learn incorpora varis mètodes d'avaluació dels models, es decideix usar l'error relatiu mitjà com a valor comparatiu entre models.

D'aquesta manera, també podrem obtenir un valor relatiu del sobreentrenament dels models ja que podem comparar el valor resultant del *dataset* de proves amb el valor resultant del *dataset* de validació.

$$ErrorRelatiu = \frac{ValorMesurat - ValorReal}{ValorReal}$$

8.3.1 Validació creuada

Una altra tècnica per avaluar models d'intel·ligència artificial és la validació creuada. Aquest mètode permet assegurar que l'avaluació es fa de manera independent a les dades d'entrenament. D'aquesta manera podem estar més segurs que l'avaluació del model reflecteix si aquest ha après els patrons de les mostres.

El procés de decidir si els resultats numèrics que quantifiquen les relacions entre variables són acceptables com a descripció de les dades es coneix com a validació.

El problema que presenta el mètode clàssic d'avaluació és que estima l'error del model a partir de la diferència entre les respostes pronosticades i les originals o correctes. Per tant, no ens garanteix que el model hagi après tots els patrons de les dades ja que no ens dona un indicació de la capacitat de generalitzar que té sobre dades que no ha vist. Aquesta indicació és precisament la que ens ofereix el mètode de validació creuada.

Hi ha vàries implementacions d'aquest mètode, és d'especial interès la coneguda com a *K-Fold*. Aquesta variant divideix el *dataset* d'entrenament en k subconjunts, a continuació entrena i avalua k vegades el model. A cada iteració s'utilitza un dels subconjunts com a validació sense repetir-lo entre iteracions i els k-1 restants per entrenar. L'error s'estima fent la mitjana dels errors dels k models i s'obté d'aquesta manera l'efectivitat total del model. Es sol recomanar un valor de k entre 5 i 10 [Gupta 2017].

8.4 Eina d'importació de dades

A part dels models candidats a implementar, es decideix implementar una eina per importar automàticament els *datasets* generats amb l'NRS.

Els experiments generats amb el simulador s'exportaran en format *JSON*, aquests seran llegits per l'eina d'importació i formatats en una taula de dades de Python. La taula, que es decideix serà un *Dataframe* de *pandas*, serà guardada en un *Magic* de Jupyter (veure apartat 6.2.1) per ser llegida en els diferents candidats implementats, una llibreta per cada candidat.

Es decideix separar aquesta funcionalitat en aquesta eina específica pel fet que és una necessitat comuna de tots els models i s'evita repetir codi en cada candidat.

8.4.1 *pandas* i *Dataframe*

pandas és una llibreria de manipulació de dades de Python que segons els seus desenvolupadors és ràpida, potent, flexible i fàcil d'usar. [pandas development team 2020], [Wes McKinney 2010].

Ofereix una estructura de dades anomenada *Dataframe* que permet treballar amb grans quantitats de dades de manera molt flexible.

Per fer l'eina d'importació s'ha fet servir aquesta estructura i, utilitzant el visualitzador de la pròpia llibreria podem veure l'estructura i les dades que conté.

	|V|	|E|	D_{avg}	D_{max}	HET	SP_{avg}	DIA	EFF	ER	LE	...	AC	NC	DC	NBC	EBC
0	16	37	4.625000	14	0.982536	2.078125	4	0.540972	156.318144	8.694681	...	0.677279	5.926951	0.625000	0.213961	0.066898
1	16	37	4.625000	14	0.982536	2.078125	4	0.540972	156.318144	8.694681	...	0.677279	5.926951	0.625000	0.213961	0.066898
2	16	37	4.625000	14	0.982536	2.078125	4	0.540972	156.318144	8.694681	...	0.677279	5.926951	0.625000	0.213961	0.066898
3	16	37	4.625000	14	0.982536	2.078125	4	0.540972	156.318144	8.694681	...	0.677279	5.926951	0.625000	0.213961	0.066898
4	16	37	4.625000	14	0.982536	2.078125	4	0.540972	156.318144	8.694681	...	0.677279	5.926951	0.625000	0.213961	0.066898
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
35575	18	24	2.666667	4	0.315063	2.549383	6	0.463617	203.959140	2.966599	...	0.347173	1.115054	0.078431	0.185856	0.073600
35576	18	24	2.666667	4	0.315063	2.549383	6	0.463617	203.959140	2.966599	...	0.347173	1.115054	0.078431	0.185856	0.073600
35577	18	24	2.666667	4	0.315063	2.549383	6	0.463617	203.959140	2.966599	...	0.347173	1.115054	0.078431	0.185856	0.073600
35578	18	24	2.666667	4	0.315063	2.549383	6	0.463617	203.959140	2.966599	...	0.347173	1.115054	0.078431	0.185856	0.073600
35579	18	24	2.666667	4	0.315063	2.549383	6	0.463617	203.959140	2.966599	...	0.347173	1.115054	0.078431	0.185856	0.073600

35580 rows × 33 columns

Figura 8.1: Visualització d'un *Dataframe* de la llibreria *pandas*.

La figura 8.1 és la representació visual del *Dataframe* que crea l'eina d'importació. Es poden veure les mètriques com a columnes i cada fila correspon a una mostra del *dataset*, hi han 35.579 mostres en aquest exemple.

8.5 Interfície d'execució del model

En aquest punt i amb la intenció de poder integrar l'eina en el simulador, es decideix crear una eina que a partir dels models exportats i dades d'una xarxa sigui capaç de fer les prediccions.

En l'apartat de disseny se'n farà una descripció més detallada del que s'espera d'aquesta eina.

CAPÍTOL 9

Anàlisi i disseny del sistema

En aquest capítol s'explicarà en detall els processos de disseny dels models candidats i de les eines que s'implementen per aquest projecte. Finalment es farà un anàlisi dels resultats de la implementació d'aquests models.

9.1 Implementació i avaluació dels candidats

Pel que respecte als candidats triats (veure secció 8.1), es fa una implementació en dues fases:

1. La primera fase és la comuna, on es farà servir la eina d'importació de dades de l'NRS i es farà un primer tractament de dades.
2. En la segona fase s'implementaran les característiques específiques de cada model candidat.

9.1.1 Implementació de l'eina d'importació de dades

Aquesta eina, com s'ha explicat anteriorment, s'utilitzarà per fer un primer tractament de les dades i deixar-les en un format que pugui ser llegit de manera fàcil per cada model.

Amb aquesta eina la metodologia per poder treballar amb els models és la següent:

- Penjar els resultats dels experiments en el servidor *Seeder* (veure secció 8.2) en format *JSON* en un directori concret.
- Obrir la llibreta que conté l'eina d'importació.
- Indicar a la variable corresponent a quin directori hem penjat els resultats per tal que l'eina els pugui llegir.
- Inicialitzar l'eina.
- Aquesta eina anirà mostrant el procés d'importació.

- Finalment l'eina, un cop llegits i tractats tots els fitxers *JSON*, exportarà usant un *Magic* (veure 6.2.1) la taula amb les dades.
- Amb aquest *Magic* es podrà importar la taula en les llibretes de cada candidat.

Aquesta taula que conté el *dataset* té per columnes les mètriques recomanades de la xarxa, el percentatge d'atac i el valor de robustesa. A cada fila hi ha un element o mostra del dataset.

Al tenir M experiments per cada P de cada xarxa, s'ha fet una mitjana dels diferents valors d'M obtenint files on la columna M no és necessària.

9.1.2 Regressió Lineal

El primer model que s'ha implementat ha estat el de regressió lineal, aquest sol ser el primer model que es prova en aquesta tipologia de problemes. La implementació s'ha fet usant el *LinearRegression* de la llibreria SciKit.

El funcionament de la llibreta Jupyter amb el model implementat és la següent:

1. En primer lloc es fa una importació de la taula amb les dades usant el *Magic*.
2. A continuació es fa una separació entre la variable depenent (la robustesa) i les variables independents.
3. Amb les variables separades en dues taules, es fa la divisió entre el dataset que es farà servir com a entrenament i el que es farà servir per validar el model i obtenir-ne una puntuació. Es dedica el 80% del dataset a fer l'entrenament i el 20% a fer la validació. La separació es fa usant el mòdul *train_test_split* de SciKit Learn.
4. Posteriorment, es fa un escalat de dades, una normalització que busca evitar el desequilibri del model en cas de tenir dades en rangs massa diversos. L'escalat es fa tant a les dades d'entrenament com a les dades de validació.
5. En aquest punt es fa l'entrenament del model usant només la part del dataset dedicada a l'entrenament.
6. Per acabar, es fa l'avaluació del model calculant l'error relatiu mitjà que tenen les seves prediccions tant sobre el dataset d'entrenament com el dataset de validació usant les funcions *mean_squared_error* i *mean_absolute_error* de SciKit Learn (veure secció 8.3 on s'explica l'avaluació conjunta).

9.1.3 K-Veïns Propers

El següent model a implementar ha estat el de veïns propers (veure 6.4.4) en la seva variant k-veïns propers o *KNN* per les seves sigles en anglès. Aquest model ja està implementat internament per la llibreria SciKit sota el nom *KNeighborsRegressor*.

El procés és idèntic al de regressió lineal:

1. Lectura del *Magic*.
2. Preparació i separació dels datasets.
3. Entrenament.
4. Avalució fent servir la metodologia establerta.

9.1.4 Arbre de decisió i bosc

Els models basats en arbres de decisió han estat els següents en ser implementats, la llibreria SciKit Learn n'implementa diversos models. En aquest cas s'utilitzen dos models en una sola llibreta Jupyter, en primer lloc, el *DecisionTreeRegressor* l'arbre de decisió dissenyat per fer regressió i, en segon lloc, el *RandomForestRegressor*, que és un bosc d'arbres de decisió (veure subsecció 6.4.5).

En aquesta llibreta s'implementen els dos models alhora però usant la mateixa metodologia que els anteriors candidats. La importació de la taula es fa de manera conjunta però l'entrenament i avaluació es fa de manera independent.

S'ajunten els dos models en una sola llibreta pel fet que estan molt relacionats i d'aquesta manera es poden comparar fàcilment.

9.1.5 Xarxes neuronals de regressió (Multi-Layer Perceptron)

Les xarxes neuronals són complexes estructuralment i intrínsicament difícils d'implementar. SciKit Learn permet treballar amb aquestes estructures a alt nivell a partir dels models que ja estan implementats, en aquest cas s'ha utilitzat el model *MLPRegressor*, un perceptró multi capa de regressió (veure apartat 6.4.6). Aquest tipus d'estructures es solen entrenar usant processadors gràfics pel fet que el nivell de paral·lelisme és més gran i al ser les neurones funcions simples es poden tractar millor amb aquest maquinari. No obstant, SciKit Learn no ofereix, de moment, suport en aquest aspecte.

La importació, entrenament i avaluació s'ha fet seguint els mateixos processos i funcions que els altres candidats.

9.2 Primers resultats

Els candidats implementats inicialment s'avaluen amb el mateix mètode, l'explicat a l'apartat 8.3. Amb aquest sistema podem comparar els primers resultats entre els models que es poden trobar a la taula 9.1. En aquesta primera prova s'ha usat el següent *dataset* (veure secció 6.1):

- P = [0,30). Es considera que un atac on falla més d'un 30% d'una xarxa és massa destructiu com per poder ser efectiva.
- M= 100. Es fan 100 atacs per cada p.
- Els atacs es fan amb l'algorisme *Random*.

Candidat	Error Relatiu Mitja
Linear Regression	5.10%
KNN	2.83%
Decission Tree	2.80%
Random Forest	2.36%
Neural Network	4.62%

Taula 9.1: Taula amb els resultats dels candidats inicials implementats.

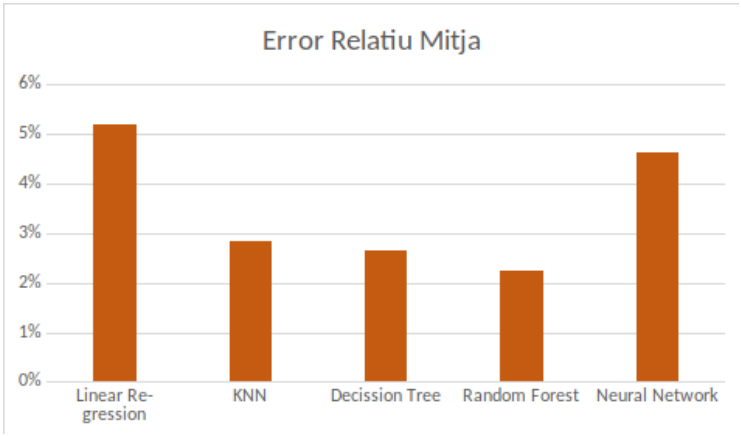

Figura 9.1: Gràfic de barres on es poden veure els diferents candidats a l'eix X i els error mitjà a l'eix Y.

Tots els candidats presenten un error relatiu molt baix, això implica que donades les variables d'entrada i.e. les mètriques de la xarxa i el percentatge d'atac els models poden predir de manera molt precisa el valor de robustesa. Sabent que la robustesa prové únicament de sumar ponderadament les mètriques recomanades, com s'ha vist explicat a la subsecció 6.1.3, i fer una normalització ja es podia esperar un resultat força bo dels candidats inicials.

Com es pot apreciar el millor model ha resultat el *Random Forest*, amb un resultat per sota del 2.5% d'error mitjà. En els casos de tots els candidats no s'aprecia una diferència rellevant comparant els resultats d'error relatiu mitjà dels *datasets* d'entrenament i validació, això ens dona un indicatiu que no s'ha produït sobreentrenament.

CAPÍTOL 10

Implementació i proves

Tot i que els primers models han resultat ser molt precisos, es decideix fer un seguit de millores centrades en el model que millor resultat ha presentat, el *Random Forest*.

10.1 Millora 1: Hiperparàmetres

La primera millora que s'implementa implica tocar les característiques per defecte dels models proporcionats per la llibreria SciKit Learn.

En l'apartat 9.1 s'ha detallat la implementació dels models i s'ha vist que es fan servir els que incorpora SciKit Learn per defecte i sense modificar.

Tots els models d'aquesta llibreria estan basats en els explicats a la secció 8 i tenen un seguit de paràmetres que tenen valors per defecte que els usuaris avançats poden modificar per obtenir encara millors resultats a les prediccions. Aquests paràmetres defineixen l'estructura i funcionament interns del model sobre el que actuen i, per tant, cada model té paràmetres diferents. Per exemple, el model de k-veïns propers (KNN) té un paràmetre que permet modificar l'algorisme que computa els veïns propers, té un valor per defecte però en certs casos pot ser interessant modificar. Aquest paràmetre no està disponible en els arbres de decisió, ja que el seu funcionament és eminentment distint.

En el camp de la intel·ligència artificial és difícil saber com afectarà modificar aquests paràmetres sobretot si se'n modifiquen diversos alhora.

Dins la llibreria SciKit Learn hi ha una funcionalitat que permet afrontar aquest problema i consisteix en fer una exploració limitada d'aquests paràmetres.

Per fer-ho es defineix el model que es farà servir, en aquest cas el *RandomForestRegressor*, els paràmetres que es volen optimitzar, el rang a buscar i quantes iteracions es vol provar.

L'algorisme farà un entrenament de varies versions del model usant combinacions dels valors dels paràmetres amb l'objectiu d'obtenir un model millor a l'inicial. Cal ser curosos al fer aquest tipus d'optimitzacions, doncs solen provocar

sobreentrenament als models i per aquest motiu la següent millora incorporarà el càlcul d'un indicatiu de generalització.

Aquesta funcionalitat sol tenir un cost computacional molt elevat ja que fer combinacions de paràmetres pot generar una quantitat explosiva de models a entrenar. Per sort, la funcionalitat d'aquesta llibreria suporta paral·lelisme per mitigar aquest efecte.

En aquest punt del projecte, els hiperparàmetres que s'han buscat optimitzar han sigut:

- *n_estimators*: Nombre d'arbres del bosc, de 200 a 2000 arbres provant en salts de 10.
- *max_features*: Nombre de mètriques a tenir en compte. S'ha provat el mode automàtic i el mode *sqrt*.
- *max_depth*: Profunditat màxima dels arbres del bosc. De 10 a 110 en salts de 10.
- *min_samples_split*: Mínima quantitat de mostres per crear un node intern nou. S'ha provat amb el conjunt: [2,5,10].
- *min_samples_leaf*:Minima quantita de mostres necesàries al arribar a un node fulla. S'ha provat amb [1,2,4].
- *booststrap*: Booleà, per defecte pren el valor *cert*, s'ha provat també el valor *fals*. Implica si es fa servir tot el *dataset* per crear cada arbre.

La llista completa i la definició completa dels hiperparàmetres d'aquest model concret es pot trobar a la documentació de SciKit Learn.

Amb aquests paràmetres definits i amb l'eina de la llibreria s'ha buscat optimitzar una mica més el model amb els resultats següents:

Candidat	**Error Relaitu Mitja**
Linear Regression	5.10%
KNN	2.83%
Decision Tree	2.80%
Random Forest	2.36%
Neural Network	4.62%
Random Forest v2	2.72%

Taula 10.1: Taula amb els resultats dels candidats inicials implementat s juntament amb el nou model.

A la taula 10.1 es pot veure que el nou model empitjora els resultats de l'original. És conseqüència de que els paràmetres per defecte són millors que els utilitzats per la cerca i, per tant, el nou model és pitjor respecte a l'original. Aquesta millora es descarta.

10.2 Millora 2: Quantitat de mètriques, *Pipes* i avaluació

A l'apartat 6.1.3 es detalla com calcula l'NRS la robustesa d'una xarxa i s'explica que s'utilitzen una sèrie de mètriques, les *mètriques recomanades*.

Degut a que un dels objectius d'aquest treball és obtenir les mètriques més rellevants per calcular la robustesa per tal de poder comparar-les amb les *mètriques recomanades* obtingudes en l'estudi que va fer el grup BCDS, és necessari que els *datasets* tinguin tot el conjunt de mètriques.

Per aquest motiu, aquesta millora ha consistit precisament en modificar el codi per poder treballar amb totes les mètriques.

El primer pas que es va haver de fer va ser adaptar el propi codi de l'NRS, ja que el simulador no escrivia les mètriques no recomanades de la xarxa en els *JSON* de sortida dels experiments.

Un cop el simulador ja va exportar aquestes dades, es va crear una segona eina d'importació per tal de no modificar l'existent i poder importar datasets amb i sense mètriques no recomanades.

En aquesta nova eina es llegeixen els JSON resultants i s'incorporen en una taula que s'exporta usant un *Magic*. En la segona versió de l'eina importadora es llegeix aquesta taula addicional per crear les noves columnes amb les mètriques noves obtenint un dataset amb totes les mètriques.

Durant la implementació d'aquesta segona millora va sorgir el problema de tractar amb mètriques que no són numèriques, poden ser cadenes de text o conjunts de valors. És el cas de la mètrica *10-EV EigenValues*, que no és un valor numèric, sinó un conjunt de valors o *array*.

Per solucionar aquest problema es va decidir descartar aquest tipus de mètriques.

10.2.1 Pipes i nova avaluació

Abans de fer servir el nou importador es decideix fer un seguit de canvis en el model actual, el *RandomForestRegressor*. En primer lloc, s'implementa un nou sistema d'avaluació, la validació encreuada o *cross validation* en anglès, explicat a la subsecció 8.3.1. Concretament s'utilitza una validació creuada de tipus *K-fold* amb 10 iteracions i.e. k=10.

Aquest mètode no es substitueix per l'existent fins ara, sinó que s'utilitza com a complement per obtenir un indicatiu de com de bé generalitzaria el model amb dades no vistes mai.

En aquest punt i amb l'objectiu de preparar les següents millores i optimitzar el model actual, es decideix implementar una Pipe que s'ocuparà de contenir tot el model en una sola estructura que conté un escalador i s'hi afegeix un mòdul que assegura que no hi ha valors fora del rang esperat i que farien fallar el model en l'entrenament.

10.2.2 Mètriques més rellevants

Amb el nou importador que incorpora totes les mètriques i la nova versió del model, es procedeix a generar datasets nous, un per cada algorisme d'atac. Els resultats es poden trobar a la taula 10.2.

Dataset	**Precisio**
TbcNSM100	97,82%
TccNSM100	98,24%
TecNSM100	98,11%
TndNSM100	98.58%
TcnNSM100	98.11%
RandomNSM100	97.58%

Taula 10.2: Taula amb els resultats del model a la millora 2.

A la taula 10.2, podem veure els *datasets* usats i la precisió obtinguda. Les sigles dels datasets tenen la següent nomenclatura:

- T = Algorisme tipus target || Random = Algorisme random.
- Si és Target, la 2a i 3a lletra indica les inicials de l'algorisme usat (veure apartat 6.1.1).
- N = atac sobre nodes, S = atac amb càlcul simultani, M100 vol dir que M = 100 atacs per cada p.

- En tots els casos P = 30 encara que no s'especifica a les sigles.

Al usar el nou mètode d'avaluació, es comprova que el model generalitza adequadament pel fet que en tots els *datasets* s'obté un valor similar al mètode anterior i la desviació estàndard de les iteracions és molt petit, menys d'un 3%.

Amb els nous *datasets* que incorporen totes les mètriques podem comprovar quines són les més rellevants a l'hora de predir el valor de la robustesa, és a dir, quines mètriques expliquen millor el valor buscat.

Per obtenir l'ordre d'importància de les mètriques es fa servir una tècnica que mesura la terminalitat dels nodesla qual mesura si els nodes evoquen a una decisió definitiva. El mètode usat es diu *Mean Decrease in Impurity*(MDI) o disminució mitjana de la impuresa.

Aquest mètode de càlcul presenta problemes al avaluar mètriques no numèriques o mètriques amb cardinalitats molt diferents, en aquest cas, totes les mètriques són numèriques i totes han sigut degudament escalades per minimitzar aquests efectes.

La implementació d'aquest mètode s'ha basat en la documentació oficial de la llibreria SciKit Learn i consta dels següents passos:

1. Obtenció dels nodes de l'arbre (només els noms).
2. Obtenció de les mètriques ordenades de cada arbre del bosc per importància usant l'atribut *feature_importances_*. Aquest atribut ja està normalitzat i és calculat usant MDI.
3. Addicionalment, es calcula la desviació estàndard de la importància de cada mètrica de tots els arbres.
4. Mostrar en un gràfic de barres les importàncies per cada mètrica obtinguda i la seva respectiva desviació estàndard.

Un cop definit i implementat el sistema d'ordenació per importància, s'esperava obtenir les mètriques recomanades com a més rellevants pel fet que són les que usa l'NRS per obtenir el valor de robustesa, no obstant, els resultats van ser totalment diferents.

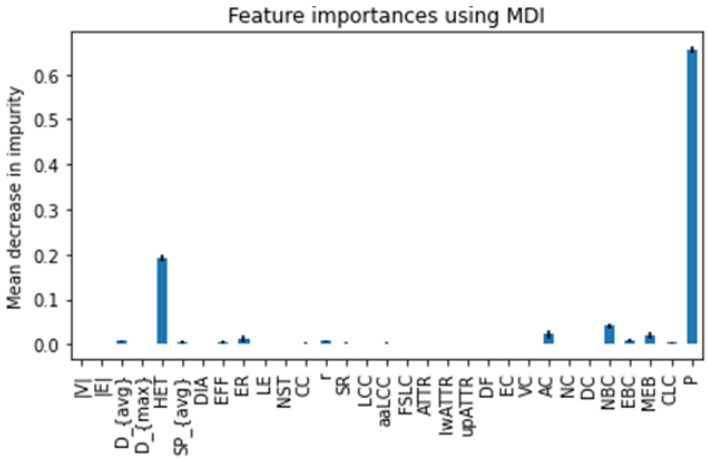

Figura 10.1: Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Betweness centrality*.

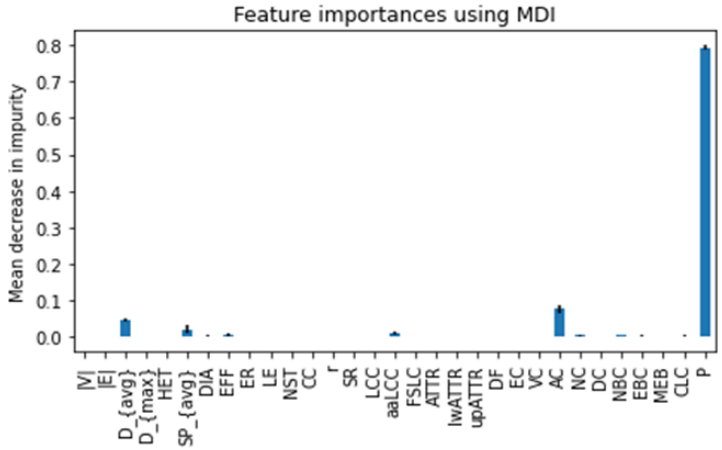

Figura 10.2: Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Closeness centrality*.

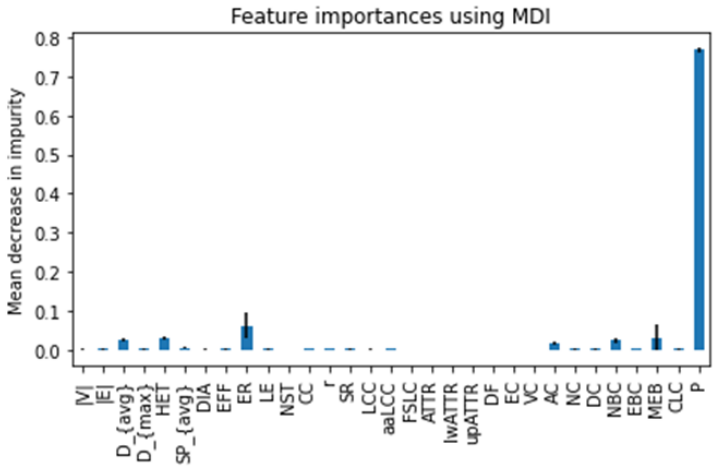

Figura 10.3: Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Eigenvector centrality*.

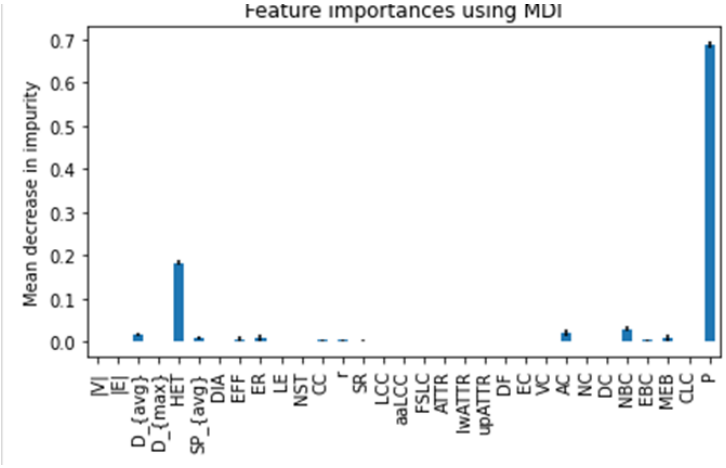

Figura 10.4: Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Nodal degree*.

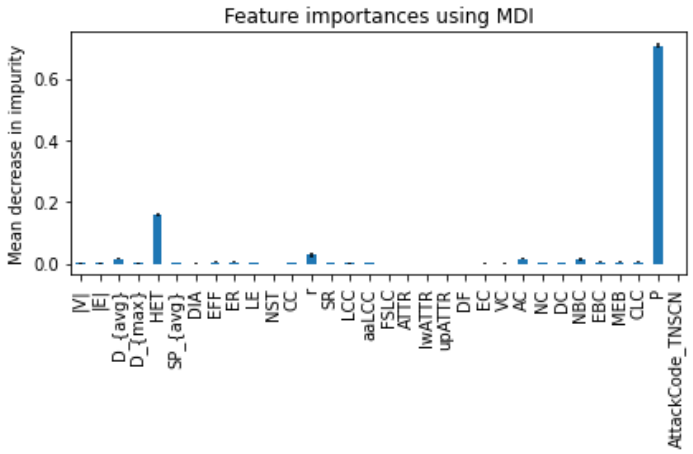

Figura 10.5: Gràfic de barres a l'eix X les mètriques del *dataset* i a l'eix Y la importància relativa sobre la robustesa. Algoritme *Critical Nodes*.

El les figures 10.1 - 10.5 es pot veure gràficament les importàncies de les mètriques per cada algoritme.

Els resultats de les importàncies i.e. de com influeix cada mètrica sobre el valor de robustesa, difereixen molt de l'esperat. A partir d'aquests se'n deriven les següents conclusions:

- El percentatge d'atac P és la mètrica més important. Aquest fet sí que era esperat, doncs és normal que la quantitat percentual d'atac en una xarxa sigui clau en la robustesa. En el cas de l'algorisme *Eigenvector centrality*, arriba a tenir una importància relativa del 80%.
- La mètrica *02-HET Heterogeneity* és la més rellevant en els datasets dels algorismes *Nodal degree*, *Betweness Centrality* i *Critical nodes*.
- Els datasets dels algorismes on la mètrica *02-HET Heterogeneity* no és la més rellevant no tenen una mètrica dominant clara, tot i que en el cas de l'*Eigenvector centrality* sobresurt la mètrica *06-ER Effective Resistance* i en el cas del *Closeness centrality* la mètrica *18-AC Algebraic connectivity*.

Si es comparen les mètriques més rellevants segons el model usat respecte les mètriques que fa servir l'NRS, es pot apreciar que no hi ha relació. Aquests resultats van ser força impactants ja que s'esperava una correlació entre els dos mètodes de càlcul de robustesa.

Després de realitzar un anàlisi dels possibles motius de tal falta de relació es va arribar a la conclusió que la causa d'aquesta diferència és el fet que les mètriques que usa l'NRS com a base pel càlcul estan molt relacionades amb la heterogenitat

i segurament amb aquesta es pot determinar R de manera més directa. En els casos on aquesta mètrica no destaca s'aprecien altres mètriques que també estan relacionades amb les mètriques recomanades en menor mesura. Tot i així, el factor més important sobre la robustesa és el percentatge d'atac que es fa a la xarxa. Aquest permet explicar en gran part el valor de robustesa buscat.

10.3 Millora 3: Unificació de models

La tercera millora que s'implementa és la simplificació dels models passant d'un model per cada tipus d'algorisme d'atac a un dos models, un per atacs amb l'algorisme aleatori i un altre per atacs amb els algorismes dirigits.

Al fer aquesta simplificació es preveu una pèrdua en la precisió pel fet que estem generalitzant al unificar els models.

Els resultats del model unificat es poden trobar a la taula 10.3.

Dataset	**Precisió**
TanyNSM100	61.32%
RandomNSM100	97.58%

Taula 10.3: Taula amb els resultats del model a la millora 3.

Els resultats d'aquesta unificació redueixen la precisió del model molt per sobre del que es podia esperar. En aquest sentit, es va decidir en primera instància descartar la millora.

Després de parlar amb el tutor i analitzar els possibles motius d'aquesta pèrdua de precisió es decideix introduir una nova columna artificialment, una nova mètrica que indiqués el tipus d'atac de cada mostra.

10.3.1 Nova mètrica

Per afegir aquesta nova mètrica es modifica de nou l'eina d'importació per tal que tingui en compte el tipus d'atac al tractar cada mostra.

Un cop afegida la mètrica a la taula sorgeix un problema al fer l'entrenament perquè la columna de la nova mètrica no és numèrica, sinó una cadena de text amb el codi de l'atac. Al haver encapsulat el model en una *Pipe* durant la millora 2, ara es pot solucionar aquest problema afegint un mòdul de codificació a l'estructura del model.

El procés de la nova pipe és el següent:

1. Separació de columnes numèriques i categòriques(resta).
2. A les numèriques s'aplica un filtre per evitar valors fora de rang i un escalat.
3. A les catergòriques es fa una codificat usant la funció *OneHotEncoder* de SciKit Learn. Veure definicions, apartat 1.

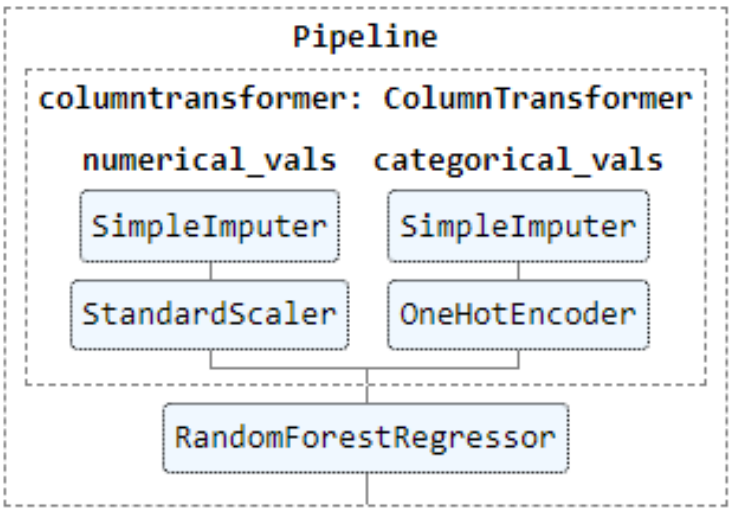

Figura 10.6: Esquema de la Pipe a la millora 3.

Amb aquesta nova mètrica que indica l'atac i usant la Pipe modificada els resultats milloren i tornen als nivells dels models separats tal i com es pot veure a la taula 10.4.

Dataset	**Precisio**
TanyNSM100	98.65%
RandomNSM100	97.58%

Taula 10.4: Taula amb els resultats del model a la millora 3.2.

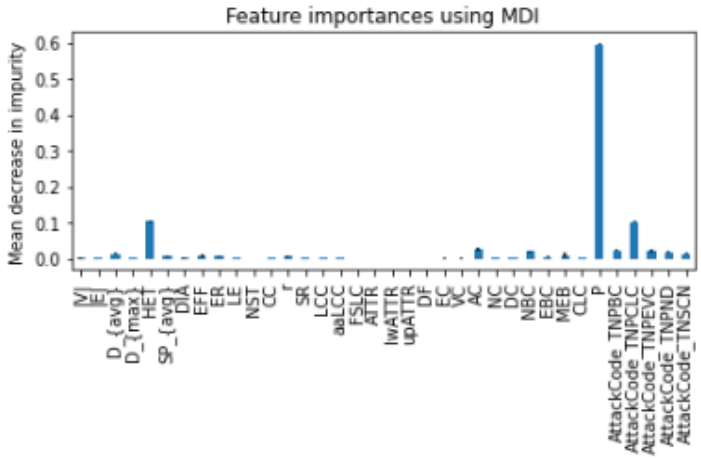

Figura 10.7: Gràfic de barres a l'eix X les mètriques del dataset i a l'eix Y la importància relativa sobre la robustesa. Usant un model conjunt pels datasets d'algorismes d'atac dirigit.

A la figura 10.7 es pot veure que en el nou model es continua tenint molt en compte el valor del percentatge d'atac, però destaca també el fet que les noves columnes generades pel codificador tenen força importància al calcular la robustesa. Aquest fet era d'esperar ja que el tipus d'atac que es fa sobre una xarxa té un important impacte sobre la robustesa.

CAPÍTOL 11

Implantació i resultats

Després d'aplicar les millores s'obtenen els dos models definitius, l'entrenat per atacs amb l'algorisme aleatori i l'entrenat per atacs dirigits.

En aquest punt cal fer l'exportació d'aquests models per tal de poder crear una interfície i ser usats per fer les prediccions.

Les diferents alternatives d'exportació s'obtenen de la pròpia documentació de la llibreria usada per entrenar els models [SciKitLearn 2021] i es basen en serialitzar-los. Per fer-ho, es recomana usar una llibreria especialitzada en aquesta tasca i es decideix usar Joblib [Jobliv 2021] al ser la recomanada. Així doncs, s'implementa la serialització i exportació dels models en el servidor *Seeder*.

11.1 Interfície de prediccions

Un cop exportats els models es crea un programa en Python amb l'objectiu de poder llegir els models exportats i ser una interfície per poder usar-los per fer prediccions de manera senzilla.

En aquest cas no es fa ús de l'entorn Jupyter pel fet que és interessant que sigui el màxim independent possible.

El programa implementat espera exactament dos paràmetres, la ruta on buscarà el fitxer amb les dades d'entrada i la ruta on escriurà el fitxer amb les de sortida.

Les dades d'entrada han d'estar en un fitxer d'extensió *JSON* i seguint una estructura específica que es detalla en la documentació del programa. En aquesta estructura cal indicar les dades de les xarxes (mètriques) i les dades dels atacs (tipus d'atac i percentatges d'atac).

Per cada possible combinació xarxa, atac i percentatge es genera una mostra de sortida que contindrà el valor predit de la robustesa de la corresponent combinació.

Aquests resultats s'escriuen en un fitxer també en extensió *JSON* a la ruta indicada en el segon paràmetre.

11.2 Resultats i comparativa entre els dos mètodes

Calcular la robustesa usant l'NRS suposa un cost computacional elevat, els principals factors a tenir en compte són el nombre d'atacs (M), el rang de percentatge d'atac i l'algorisme que es fa servir per atacar.

A la taula 11.1 es pot veure el temps que s'ha tardat en obtenir els càlculs dels diferents datasets usats en aquest treball.

Dataset	Temps d'execució
TbcNSM100	29h 06m 52s
TccNSM100	44h 56m 52s
TecNSM100	44h 45m 5s
TndNSM100	44h 42m 56s
TcnNSM100	49h 39m 52s
RandomNSM100	10h 12m 38s

Taula 11.1: Taula amb els temps d'execució dels diferents datasets usant l'NRS.

Usant la interfície de prediccions s'ha fet el còmput següent amb l'objectiu de fer comparacions:

- Totes les xarxes del *Topology Zoo* [Knight 2011].
- Tots els tipus d'atac, l'aleatori i els 5 atacs dirigits.
- Percentatge d'atac entre 0 i 30 %.

Com es pot comprovar, no només hi ha igualtat de condicions en el fet que s'utilitza el mateix dataset i els mateixos percentatges, sinó que se li demana a la interfície que calculi tots els tipus d'atac. Els resultats es poden veure a la figura 11.1.

```
linux@ubuntu:~/Desktop/robustnessaicalculator$ time /home/linux/Desktop/robustne
ssaicalculator/venv/bin/python /home/linux/Desktop/robustnessaicalculator/main.p
y /home/linux/Desktop/robustnessaicalculator/examples/fullNetworks.json /home/li
nux/Desktop/robustnessaicalculator/out.json

real    0m2,882s
user    0m2,482s
sys     0m0,455s
```

Figura 11.1: Captura de l'execució i temps de còmput de la interfície de prediccions amb P=[0,30], totes les xarxes i tots els tipus d'atac.

Com es pot apreciar, aquesta interfície té un temps de càlcul mínim, de l'ordre de pocs segons.

En el fitxer de sortida del programa, s'hi pot trobar, per cada combinació de xarxa, tipus d'atac i percentatge, el valor predit de robustesa.

Aquest fitxer té unes 50.000 línies en format JSON, una mostra es pot trobar a la figura 11.2. En aquesta es veu en el primer nivell un objecte que fa referència a la xarxa, en segon nivell, els objectes que corresponen als atacs i en últim nivell els diferents percentatges d'atac i el seu valor predit de robustesa. Es mostra només una xarxa i s'han eliminat els valors de P majors de 5 amb l'objectiu de poder fer una captura més comprensible.

```
{
    "614990f711a9bce86607f404": {
        "TNPBC": {
            "1": 0.825700720000001,
            "2": 0.8231013700000012,
            "3": 0.7828018300000008,
            "4": 0.6731805899999995,
            "5": 0.6429289199999989,
        },
        "TNPCLC": {
            "1": 0.9496587499999992,
            "2": 0.9484476999999994,
            "3": 0.9359888499999994,
            "4": 0.8909367424999995,
            "5": 0.8858731999999994,
        },
        "TNPEVC": {
            "1": 0.9226779699999984,
            "2": 0.9202563899999987,
            "3": 0.9182154899999985,
            "4": 0.8319157260000007,
            "5": 0.828711566000001,
        },
        "TNPND": {
            "1": 0.8993325399999986,
            "2": 0.8974640349999984,
            "3": 0.8878254799999987,
            "4": 0.7886522900000003,
            "5": 0.7828968100000003,
        },
        "TNSCN": {
            "1": 0.9191435099999984,
            "2": 0.9148413849999985,
            "3": 0.8922823999999991,
            "4": 0.81952963,
            "5": 0.8099308199999998,
        },
        "random": {
            "1": 0.9490145033333325,
            "2": 0.9487278299999986,
            "3": 0.9428239749999985,
            "4": 0.91039773,
            "5": 0.8989199333333338,
        }
```


Figura 11.2: Mostra reduïda del contingut del fitxer de sortida de la interfície de prediccions.

CAPÍTOL 12

Conclusions

Aquest treball en un primer moment volia tenir un enfocament molt pràctic amb l'objectiu de desenvolupar una eina que funcioni i resolgui el problema amb nivell acceptable de precisió. No obstant, l'apartat teòric i estudi de l'estat de l'art ha acabat tenint un pes molt important en el desenvolupament del treball i, per tant, s'han hagut d'explorar a fons alguns dels algorismes per tal de comprendre com es podrien millorar els resultats.

S'ha aconseguit crear un sistema capaç de calcular la robustesa d'una xarxa de manera instantània. Aquesta eina permet ser-ne la base d'altres, tal i com es veurà a l'apartat de Treball Futur.

L'eina creada presenta un encert en les prediccions molt alt, per sobre del 97%, fet que podria fer-nos plantejar utilitzar aquest model com a substitut del sistema de càlcul actual.

Alguns dels intents d'optimització dels models s'han hagut de descartar pel fet que empitjoraven els resultats dels models originals.

També s'ha aconseguit detallar les mètriques més rellevants, les que expliquen millor la robustesa d'una xarxa en funció de cada atac. Això permet saber quines mètriques cal reforçar en cas de voler protegir una xarxa davant d'atacs concrets.

Finalment, s'ha creat una interfície que permet treballar amb els models exportats de manera senzilla i que permetrà, ja sigui a través de l'NRS o de manera individual, fer prediccions sense haver de fer adaptacions al codi ja existent.

En resum, s'han assolit tots els objectius marcats a l'inici del projecte.

CAPÍTOL 13

Treball futur

13.1 Incorporació a l'NRS

En primer lloc, caldria incorporar a la web del simulador un apartat per poder fer experiments usant la interfície de prediccions.

Caldria també afegir la interfície als servidors de computació del simulador per tal de poder-ne fer ús.

13.2 Nous *datasets*

El *dataset Topology Zoo* [Knight 2011], l'usat en aquest treball, és un conjunt de xarxes de telecomunicacions. Aquest factor fa que els models creats siguin bons per aquesta tipologia de xarxes, però caldria fer proves amb altres *datasets* per poder afirmar que generalitzen adequadament les prediccions.

Una manera de fer-ho seria aconseguir més xarxes dels operadors però sol ser complicat ja que les topologies es solen considerar un secret empresarial. Alternativament, es podrien usar xarxes sintètiques, és a dir, xarxes creades artificialment a partir de certs paràmetres, el simulador NRS està capacitat per crear-ne.

13.3 Eina recomanadora de xarxes

La continuació natural d'aquest treball és el disseny i desenvolupament d'una eina que donada una xarxa, sigui capaç d'indicar quines són les característiques que podrien millorar-ne la robustesa.

Per fer-ho es podria utilitzar la interfície de prediccions com a base i un algorisme d'intel·ligència artificial recomanador i.e. d'aprenentatge no supervisat per tal de fer aquestes recomanacions.

Bibliografia

[BCDS] BCDS. *Portal Web del BCDS*. https://bcds.udg.edu/. (Accessed on 06/08/2022). (Cited on page 23.)

[Calle 2021] Eusebi Calle, David Martínez, Mariusz Mycek and MichałPióro. *Resilient backup controller placement in distributed SDN under critical targeted attacks*. International Journal of Critical Infrastructure Protection, vol. 33, page 100422, 2021. (Cited on page 16.)

[den Bossche 2018] Joris Van den Bossche. *scikit-learn and Tabular Data: Closing the Gap*, 2018. Presentació interessant sobre com treballar amb sklearn amb pipes. (Cited on page 30.)

[Granger 2021] Brian E. Granger and Fernando Perez. *Jupyter: Thinking and Storytelling With Code and Data*. Computing in Science and Engineering, vol. 23, pages 7–14, 3 2021. (Cited on pages 7 and 28.)

[Gupta 2017] Prashant Gupta. *Cross-Validation in Machine Learning | Towards Data Science*. https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f, 6 2017. (Accessed on 06/10/2022). (Cited on page 40.)

[IBM 2020] IBM. *Neural Networks*, 8 2020. (Cited on pages ix, 33 and 34.)

[Jobliv 2021] Jobliv. *Joblib: running Python functions as pipeline jobs — joblib 1.2.0.dev0 documentation*. https://joblib.readthedocs.io/en/latest/, 2021. (Accessed on 06/08/2022). (Cited on page 61.)

[Knight 2011] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden and Matthew Roughan. *The Internet Topology Zoo*. IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pages 1765–1775, 2011. (Cited on pages 17, 28, 62 and 67.)

[Marzo 2018] Jose L Marzo, Eusebi Calle, Sergio G Cosgaya, Diego Rueda and Andreu Mañosa. *On selecting the relevant metrics of network robustness*. In 2018 10th International Workshop on Resilient Networks Design and Modeling (RNDM), pages 1–7. IEEE, 2018. (Cited on page 23.)

[Marzo 2019] Jose L Marzo, Sergio G Cosgaya, Nina Skorin-Kapov, Caterina Scoglio and Heman Shakeri. *A study of the robustness of optical networks under massive failures*. Optical Switching and Networking, vol. 31, pages 1–7, 2019. (Cited on pages 1 and 23.)

[Oxford 2022] Oxford. *OVERFITTING | Definición de OVERFITTING por Oxford Dictionary en Lexico.com y también el significado de OVERFITTING*. https://www.lexico.com/definition/overfitting, 2022. (Accessed on 06/08/2022). (Cited on page 2.)

[pandas development team 2020] The pandas development team. *pandas-dev/pandas: Pandas*, February 2020. (Cited on page 41.)

[Pedregosa 2011] F Pedregosa, G Varoquaux, A Gramfort, B Michel V., Thirion, O Grisel, M Blondel, R Prettenhofer P., Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot and E Duchesnay. *Scikit-learn: Machine Learning in Python*. volume 12, pages 2825–2830, 2011. (Cited on pages 29 and 39.)

[RedHat 2020] RedHat. *What is agile methodology?* https://www.redhat.com/en/topics/devops/what-is-agile-methodology, 1 2020. (Accessed on 06/10/2022). (Cited on page 9.)

[Rehkopf 2020] Max Rehkopf. *Sprints | Atlassian*. https://www.atlassian.com/agile/scrum/sprints, 2020. (Accessed on 06/10/2022). (Cited on page 9.)

[Saravanan 2018] R. Saravanan and Pothula Sujatha. *A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification*. In 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), pages 945–949, 2018. (Cited on page 1.)

[SciKitLearn 2021] SciKitLearn. *9. Model persistence — scikit-learn 1.1.1 documentation*. https://scikit-learn.org/stable/model_persistence.html, 2021. (Accessed on 06/08/2022). (Cited on page 61.)

[SciKitLearn 2022] SciKitLearn. *Supervised Learning SciKit Learn*, 2022. (Cited on pages ix, 31 and 35.)

[Trajanovski 2013] Stojan Trajanovski, Javier Martín-Hernández, Wynand Winterbach and Piet Van Mieghem. *Robustness envelopes of networks*. Journal of Complex Networks, vol. 1, no. 1, pages 44–62, 2013. (Cited on page 23.)

[Wes McKinney 2010] Wes McKinney. *Data Structures for Statistical Computing in Python*. In Stéfan van der Walt and Jarrod Millman, editeurs, Proceedings of the 9th Python in Science Conference, pages 56 – 61, 2010. (Cited on page 41.)