

Facilitating the analysis and management of data for cancer care

Agastya Silvina

A thesis submitted for the degree of EngD
at the
University of St Andrews



2021

Full metadata for this thesis is available in
St Andrews Research Repository
at:

<https://research-repository.st-andrews.ac.uk/>

Identifier to use to cite or link to this thesis:

DOI: <https://doi.org/10.17630/sta/788>

This item is protected by original copyright

To my absent friend . . .

Candidate's declaration

I, Agastya Silvina, do hereby certify that this thesis, submitted for the degree of EngD, which is approximately 36,279 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for any degree. I confirm that any appendices included in my thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy. I was admitted as a research student at the University of St Andrews in November 2016. I received funding from an organisation or institution and have acknowledged the funder(s) in the full text of my thesis.

26 October 2021

Supervisor's declaration

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of EngD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree. I confirm that any appendices included in the thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

26 October 2021

Permission for publication

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand, unless exempt by an award of an embargo as requested below, that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that this thesis will be electronically accessible for personal or research use and that the library has the right to migrate this thesis into new electronic forms as required to ensure continued access to the thesis. I, Agastya Silvina, confirm that my thesis does not contain any third-party material that requires copyright clearance. The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

Printed copy, No embargo on print copy. Electronic copy, No embargo on electronic copy.

signature of candidate,

signature of supervisor

26 October 2021

Underpinning Research Data or Digital Outputs

Candidate's declaration

I, Agastya Silvina, understand that by declaring that I have original research data or digital outputs, I should make every effort in meeting the University's and research funders' requirements on the deposit and sharing of research data or research digital outputs.

Date

26 October 2021

Signature of candidate

Permission for publication of underpinning research data or digital outputs

We understand that for any original research data or digital outputs which are deposited, we are giving permission for them to be made available for use in accordance with the requirements of the University and research funders, for the time being in force. We also understand that the title and the description will be published, and that the underpinning research data or digital outputs will be electronically accessible for use in accordance with the license specified at the point of deposit, unless exempt by award of an embargo as requested below. The following is an agreed request by candidate and supervisor regarding the publication of underpinning research data or digital outputs: No embargo on underpinning research data or digital outputs.

Date

26 October 2021

Signature of candidate

Date

26 October 2021

Signature of supervisor

Acknowledgements

I must begin by thanking my supervisor, Dr. Juliana Küster Filipe Bowles, for her endless patience throughout the realisation of this thesis. I thank her for all the time, help, and guidance during the project.

Dr. Peter Hall, Christina Lilley, and the team in Edinburgh Cancer Centre (ECC) for the support during the course of four years.

My family, for the encouragement to keep on going.

I also want to thank Jin Anne Elizabeth Lee for the constant reminder of how precious life is, Ylenia "Dr. Gelaxon" Giarratano for the rations, gossips, jokes and tons of giggles during tough times of my EngD, Holly Tibble for her chocolate and snuggles, Romain Enjalbert for the time we spend exchanging interesting ideas on our time commuting back from work, Sofia de la Fuente Garcia for simply sitting in front of my desk, Pierre Albert for being the beloved big brother on the corner, Kevin C.H. Tsang for his problem solving ability, Minhong Wang for the unoccupied space on the left side of my desk, Guilherme Redeker for his "invaluable" advice and time to solve the problems I dumped at him, David Hardman for his juvenile wit and unbounded knowledge for the things that are not considered as matters of consequence, and Ana Vilaplana Velasco for being the cheerful lovely little young lady in the office.

To all my friends in the Edinburgh BQ9, thank you for giving me welcome distractions and entertainments.

Lastly, to my absent friend, thank you for showing me the "beautiful-sadness" of being alive.

Funding

This work was funded by the Data Lab and NHS Lothian.

Abstract

The Edinburgh Cancer Centre (ECC) is an institution containing the National Health Service (NHS) Lothian cancer patient data from multiple resources. These resources are scattered across different systems and platforms, making it difficult to use the information collected in a useful way. There is a lack of proxy between the different (sub)systems, and this thesis presents a series of applications/projects to promote data usage and interoperability. We develop both front-end and back-end applications to bring together several databases, such as *ChemoCare*, Trak, and Oncology database. We create the South East Scotland Oncology (SESO) Gateway to improve the quality and capability of reporting outcomes within South East Scotland Oncology databases in real-time using routinely captured and integrated electronic healthcare data. With SESO Gateway, we focus on cancer pathway data visualisation for both the personal timeline and the cohort summary for various treatments. We also carry out a database migration and evaluate several reporting services for the newly migrated database to accelerate data access. We then perform data analysis for the patient's treatment waiting time. By analysing the waiting time and comparing it to the intended pathway, we can simplify the auditing process of the first stage of patients' cancer care journey. Further, we use the patients' treatment data, recorded toxicity level, and various observations concerning breast cancer patients to create models to analyse the outcome of the treatments, mainly chemotherapy. We compare several different techniques applied to the same data set to predict the toxicity outcome of the treatment. Through analysis and evaluation of the performance of these techniques, we can determine which method is more suitable in different situations to assist the oncologists in real-time clinical practice. After training the models, we create a dashboard as a placeholder for the models. Lastly, we explore how to define rules for cancer data and use a constraint based approach to fabricate a large cancer dataset, which will allow us to explore more techniques and further improve our system capability in the future. With our proposed systems, healthcare professionals can directly access and analyse patient data to gain further insights regarding the treatment that is best suited for an individual patient.

Publications

Juliana Bowles, Agastya Silvina, Eyal Bin, and Michael Vinov. On Defining Rules for Cancer Data Fabrication. In *4th International Joint Conference on Rules and Reasoning*. RuleML+RR, 2020.

Vladimir Janjic, JKF Bowles, Andreas Francois Vermeulen, Agastya Silvina, Marios Belk, Christos Fidas, Andreas Pitsillides, Mohit Kumar, M Rossbory, Michael Vinov, et al. The SERUMS tool-chain: Ensuring Security and Privacy of Medical Data in Smart Patient-Centric Healthcare Systems. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2726–2735. IEEE, 2019.

A. Silvina, G. Redeker, T. Webber, and J. Bowles. A Simulation-based Approach for the Behavioural Analysis of Cancer Pathways. In *9th International Symposium from Data to Model and Back*. Datamod, 2020.

Agastya Silvina, Juliana Bowles, and Peter Hall. On Predicting the Outcomes of Chemotherapy Treatments in Breast Cancer. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 180–190. Springer, 2019.

Agastya Silvina, Juliana Kuster Filipe Bowles, and Peter Hall. Combining patient pathway visualisation with prediction outcomes for chemotherapy treatments. In *12th International Conference on eHealth, Telemedicine, and Social Medicine (eTELEMED 2020), 21-25 November 2020, Valencia, Spain*. International Academy, Research, and Industry Association, 2020.

Table of contents

List of figures	i
List of tables	i
1 Introduction	7
1.1 Cancer Overview	7
1.2 Objectives	9
1.3 Thesis Outline	10
2 Context	13
2.1 Introduction	13
2.2 Concurrent Programming	14
2.3 Simulation Models	14
2.3.1 Discrete Event Simulation	15
2.4 Constraint Programming	16
2.5 Machine Learning	18
2.5.1 Association rules	18
2.5.2 Classification	18
2.5.3 Regression	19
2.5.4 Unsupervised Learning	19
2.5.5 Reinforcement Learning	20
2.6 Methods	20
2.6.1 Cox Regression	20
2.6.2 Markov Model	21
2.6.3 Hidden Markov Model	22
2.6.4 Random Forest	23
2.6.5 Artificial Neural Network	25
2.6.6 Recurrent Neural Network	26

2.7	Summary	29
3	Patient Timeline Visualisation	31
3.1	Introduction	31
3.2	Related Work	32
3.3	Requirements	34
3.4	Data Analysis	35
3.4.1	Data Source	35
3.4.2	Data Cleansing	36
3.5	Implementation	36
3.5.1	System Design	37
3.5.2	Software Architecture	43
3.6	Analysis	52
3.6.1	Software Testing	52
3.6.2	Case Study: Feedback and commentary	57
3.7	Summary	58
3.7.1	Delivery of the key requirements	58
3.7.2	Scalability	60
3.7.3	Contribution	61
4	Migration	63
4.1	Introduction	63
4.2	Legacy Database	63
4.3	Motivation	64
4.4	Database Migration	64
4.4.1	Extraction of Data	64
4.4.2	Data Transformation	65
4.4.3	Data Loading	68
4.4.4	Execution	68
4.4.5	Data Validation	69
4.5	Issues	69
4.5.1	Driver installation	70
4.5.2	Invalid UHPI and CHI	70
4.5.3	Invalid date	70
4.5.4	Invalid drop-down value	70
4.6	Reporting Service	71
4.6.1	In-House Visualisation Tool Integration	71

4.6.2	SQL Server Reporting Service (SSRS)	72
4.6.3	PowerBI	73
4.6.4	Stimulsoft	75
4.6.5	Tableau	76
4.6.6	Shiny	77
4.7	Summary and Contribution	78
5	Cancer Waiting Time (CWT)	79
5.1	Introduction	79
5.2	Related Work	81
5.3	Data Characteristic	81
5.4	Data Analysis and Query Development	83
5.5	Simulation	85
5.6	Validation	90
5.7	Summary	91
5.7.1	Contribution	93
6	Toxicity Predictor	95
6.1	Introduction	95
6.2	Related Work	96
6.3	Methodology	96
6.4	Data Analysis	98
6.4.1	Data Characteristics	98
6.4.2	Feature Analysis	99
6.5	Predictor Models	101
6.6	Statistical Models	101
6.6.1	Survival Analysis	101
6.6.2	Markov Model (MM)	103
6.7	Machine Learning and Deep Learning Models	104
6.7.1	Hidden Markov Model (HMM)	104
6.7.2	Random Forest (RF)	107
6.7.3	Recurrent Neural Network (RNN)	108
6.8	Model Evaluation	108
6.9	Summary	111
7	Patient Timeline Visualisation and Toxicity Predictor Integration	113
7.1	Introduction	113

7.2	Frameworks	113
7.3	Use Case	114
7.4	Implementation	115
7.4.1	Front-End	116
7.4.2	Back-End	117
7.5	Evaluation	118
7.6	Summary	120
8	Generating Synthetic Cancer Data	125
8.1	Introduction	125
8.2	Related Work	127
8.3	Data Structure	128
8.4	IBM's Data Fabrication Platform	130
8.5	Methodology	132
8.6	Rule Design	135
8.7	Validation	142
8.7.1	Baseline Training Set	142
8.7.2	Validation Result	143
8.8	Summary	152
8.8.1	Coronavirus Remark	152
9	Conclusion and Future Work	153
9.1	Contributions	155
	References	157
	Appendix A Architecture and Design Pattern	169
A.1	Representational state transfer (REST) API	169
A.2	Model-View-Controller	169
	Appendix B Statistical Tools	171
B.1	Weibull Distribution	171
B.2	Generalized Extreme Value Distribution	173
B.3	Kaplan-Meier Estimator	174
B.4	Nelson-Allen Estimator	175
B.5	Shapiro-Wilk Test	176
B.6	Kolmogorov–Smirnov Test	176

Appendix C Databases	179
C.1 Microsoft SQL Server	179
C.1.1 Transact-SQL (T-SQL)	179
C.2 Microsoft Access	179
C.3 SQLite	180
C.4 Redis	180
Appendix D Framework	181
D.1 AngularJS	181
D.2 Bootstrap	183
D.3 Django	183
D.4 Gatling	184
Appendix E Libraries and Toolkit	185
E.1 NumPy	185
E.2 pandas	185
E.3 SciPy	186
E.4 scikit-learn	186
E.5 Matplotlib	186
E.6 Lifelines	187
E.7 Keras	187
E.8 Tensorflow	187
E.9 Django REST Framework	187
E.10 Celery	188
E.11 pickle	188
E.12 Salabim	188
E.13 pyodbc	189
E.14 JDBC	189
E.15 vis.js	189
E.16 Data-Driven-Documents (D3)	189
E.17 Gradle	190

List of figures

1.1	ECC cancer patient data sources	8
2.1	Project Scope	13
2.2	Discrete event simulation flowchart	16
2.3	An interaction between an agent and its environment	20
2.4	An example of an HMM model. Source: [80]	22
2.5	Decision tree classification. Source: [90]	23
2.6	Bagging classification. Source: [90]	24
2.7	Schematic of a single hidden layer, feed-forward neural network.	25
2.8	An example of LSTM cell. Source: [97]	28
2.9	LSTM cell control gate. Source: [97]	29
3.1	An example of <i>ChemoCare</i> data observation	33
3.2	The <i>SESO Gateway</i> features	34
3.3	<i>SESO Gateway</i> homepage wireframe	38
3.4	<i>SESO Gateway</i> individual timeline wireframe	39
3.5	<i>SESO Gateway</i> surgical wireframe	40
3.6	<i>SESO Gateway</i> radiotherapy wireframe	40
3.7	<i>SESO Gateway</i> chemotherapy wireframe	41
3.8	<i>SESO Gateway</i> hormone therapy wireframe	42
3.9	<i>SESO Gateway</i> architecture	43
3.10	<i>SESO Gateway</i> individual timeline with dummy data	44
3.11	<i>SESO Gateway</i> individual timeline use case	45
3.12	<i>SESO Gateway</i> individual timeline sequence diagram	46
3.13	<i>SESO Gateway</i> individual timeline sequence diagram	47
3.14	<i>NVD3</i> building blocks	48
3.15	<i>SESO Gateway</i> patient's stage proportion	48
3.16	<i>SESO Gateway</i> Sugical page	49

3.17	<i>SESO Gateway</i> unit and integration tests coverage	56
3.18	<i>SESO Gateway</i> event timeline	57
3.19	<i>SESO Gateway</i> updated system architecture	60
4.1	Patients' diagnosis with Tableau	77
5.1	Lung cancer NHS Lothian pathway guideline. Source: [120]	80
5.2	Simplified lung cancer pathway	86
5.3	Observed lung cancer pathway	87
5.4	Lung cancer pathway DES	88
5.5	Number of services for the CWT events	89
5.6	Patients' waiting time distribution	90
6.1	Patients' proportion against low toxicity	100
6.2	Adjuvant therapies fields' correlation map	100
6.3	<i>Kaplan-Meier</i> survival analysis	102
6.4	Cumulative hazard function	103
6.5	The diagram representing the Markov chain for patients' toxicity outcome .	104
6.6	The viterbi calculation for the toxicity predictor	106
6.7	The feature importance in <i>Neo-adjuvant</i> treatments	107
6.8	The Distributions for Chemotherapy treatments	109
7.1	Distributed Task Queue Architecture. Source: [161]	114
7.2	Rendered model charts	121
7.3	Sequence diagram for treatment analyses	122
7.4	<i>SESO Gateway</i> system architecture	122
7.5	Predictors performance test result (with <i>celery</i>)	123
7.6	Predictors performance test result (without <i>celery</i>)	123
8.1	Relations between stakeholders in a safe-haven context	126
8.2	The simplified pathway for cancer patients	129
8.3	Chemotherapy treatment protocol	130
8.4	Data fabrication flow. Source:[77]	131
8.5	Rules generation flowchart	134
8.6	CHI components	135
8.7	BMI, Age-Intention distribution	137
8.8	Example of validation nodes	144
8.9	Demographics accuracy	148

8.10 Smr01s accuracy	148
A.1 A Web API interaction. Source: [102]	169
A.2 An example of an MVC application. Source: [117]	170
B.1 Weibull densities with differing value of the location parameter	172
B.2 Weibull densities with differing value of the scale parameter	172
B.3 Weibull densities with differing value of the shape parameter	173
B.4 An example of GEV random sample [105]	173
B.5 An example of <i>Kaplan-Meier</i> estimated survival curve. Source: [45]	174
B.6 Cumulative hazard function of global regimes. Source: [94]	175
B.7 Kolmogorov-Smirnov statistic. Source: [101]	177
B.8 Kolmogorov-Smirnov critical values. Source: [103]	178
D.1 Client-server request-response cycle. Source: [76]	182
D.2 Two ways data binding [12]	182

List of tables

1.1	ECC Datasources	8
2.1	Different type of RNN	26
3.1	<i>SESO Gateway</i> project scope	36
5.1	CWT dataset	81
5.2	Cancer Waiting Time States	88
5.3	Patients' waiting time descriptive statistic	91
5.4	CWT simulated scenarios	91
6.1	The treatments' intentions	98
6.2	Preprocessing steps of chemotherapy data	98
6.3	Models Comparison	101
6.4	Intentions hazard ratio	103
6.5	The transition probability for all adjuvant treatment regimes	103
6.6	The HMM components for predicting the toxicity outcome	104
6.7	The transition probability for all adjuvant treatment regimes	106
6.8	Model test result (mean-std)	109
8.1	Source table brief overview	128
8.2	Chemotherapy treatment data representation	130
8.3	IBM Solver syntaxes	133
8.4	Helper tables' foreign key	140
8.5	Performance Status correlation	142
8.6	Baseline training accuracy result for Demographics table	143
8.7	Individual table validation result	143
8.8	First iteration validation	145
8.9	A mock example of general practitioners from the real data	145
8.10	A mock example of general practitioners from the fabricated data	146

8.11	Second iteration validation	147
8.12	First iteration re-validation	149
8.13	Second iteration re-validation	149
8.14	Third iteration validation	150
8.15	List of validated fields	150

List of Abbreviations and Acronyms

1NF	First Normal Form
AJCC	American Joint Committee on Cancer
API	Application Programming Interface
ANN	Artificial Neural Network
BMI	Body Mass Index
CHI	Community Health Index
CNN	Convolutional Neural Network
COBOL	Common Business Oriented Language
COD	Caused of Death
CRN	Case Reference Number
CSS	Cascading Style Sheets
CT	Computed Tomography
CWT	Cancer Waiting Time
D3	Data Driven Documents
DB	Database
DCO DB	Legacy Oncology Database
DES	Discrete Event Simulation
ECC	Edinburgh Cancer Centre
ETL	Extraction Transformation Loading
GP	General Practitioner
HTML	Hypertext Markup Language

IBM	International Business Machines
ICD	International Classification of Diseases
JDK	Java Development Kit
KM	Kaplan-Meier
KS	Kolmogorov-Smirnov
MDM	Multidisciplinary Meeting
ML	Machine Learning
MM	Markov Model
MRI	Magnetic Resonance Imaging
MVC	Model View Controller
MS Access	Microsoft Access
MSSQL	Microsoft Structured Query Language
NHS	National Health Service
NLP	Natural Language Processing
OP	Outpatient
PET	Positron Emission Tomography
PS	Performance Status
RDBMS	Relational Database Management System
REST	Representational state transfer
RF	Random Forest
RNN	Recurrent Neural Network
SAS	Statistical Analysis System
SES	South East Scotland
SESO	South East Scotland Oncology
SPA	Single Page Application
SVG	Scalable Vector Graphics
SQL	Structured Query Language

T-SQL	Transact-Sequel
UAT	User Acceptance Test
UHPI	Unique Hospital Patient Identification
WGH	Western General Hospital

List of Medical Terminology

ablation	the surgical removal of body tissue.
adjuvant	the treatment after the primary treatment.
aramidex	aromatase inhibitor approved by the U.S. Food and Drug Administration (FDA) to treat postmenopausal women diagnosed with hormone-receptor-positive.
cancer	a disease caused by an uncontrolled division of abnormal cells in a part of the body.
chemotherapy	the treatment of disease by the use of chemical substances, especially the treatment of cancer by cytotoxic and other drugs.
comorbidity	the simultaneous presence of two or more diseases or medical conditions in a patient.
constipation	a condition in which there is difficulty in emptying the bowels, usually associated with hardened faeces.
curative	able to cure disease.
diarrhoea	a condition in which faeces are discharged from the bowels frequently and in a liquid form.
esophagitis	inflammation that may damage tissues of the esophagus.
Exemestane	a hormonal therapy drug used to treat breast cancer.
hazard ratio	the relative risk of the complication based on comparison of event rates.
histology	the study of the microscopic structure of tissues.
hormone	a regulatory substance produced in an organism and transported in tissue fluids such as blood or sap to stimulate specific cells or tissues into action.

implant	any device or material, especially of an inert substance, used for repairing or replacing part of the body.
intention	The aim of the initial treatment for cancer for the particular patient.
intracavitary	Within a cavity or space, such as the abdomen, pelvis, or chest.
letrozole	one of the medicines used for treating breast cancer.
lumpectomy	surgery to remove cancer or other abnormal tissue from the breast.
mastectomy	a surgical operation to remove a breast.
metastasis	the development of secondary malignant growths at a distance from a primary site of cancer.
nausea	an uneasiness of the stomach that often comes before vomiting.
neo-adjuvant	treatment given as a first step to shrink a tumor before the main treatment.
treatment neurotoxicity	damage to the brain or peripheral nervous system caused by exposure to natural or man-made toxic.
neutron	A neutron destroys a tumour cell through a nuclear reaction.
therapy oncology	the study and treatment of tumours.
oral mucositis	inflammation in the mouth.
palliative	one of the cancer intention treatment which purpose is to extend the patients life.
prognosis	the likely course of a medical condition.
prostate	a gland that produces some of the fluid that carries sperm during ejaculation.
radiotherapy	the treatment of disease, especially cancer, using X-rays or similar forms of radiation.
regime	a coordinated programme for the promotion or restoration of health; a regimen.
tamoxifen	type of drugs that block the effects of estrogen in the breast tissue.
toxicity	the quality of being toxic or poisonous.
tumour	a swelling of a part of the body, generally without inflammation, caused by an abnormal growth of tissue, whether benign or malignant.

Chapter 1

Introduction

1.1 Cancer Overview

Cancer is a chronic disease where cells grow and multiply uncontrollably. If left untreated, cancer cells spread and invade surrounding tissues and eventually metastases in other parts or organs of the body. Cancer has become a huge medical problem, one of the primary causes of mortality in the UK. It is diagnosed each year in more than a hundred people, men or women alike [166]. Several main methods for treating cancer include surgery, radiotherapy (for primary tumour), and chemotherapy. For some tumours, such as breast cancer, the ultimate aim of the treatment has been directed towards defining methods to eradicate the primary tumour: through surgery. Unfortunately, this method alone does not improve the prognosis of cancer patients because one of the most important causes of mortality is metastatic spread. Even though breast cancer surgery can diminish the possibility of recurrences, metastasis often develops before diagnosis or at the beginning of the treatment. Hence, the prognosis (i.e. the outcome, chance, or likelihood of recovery from cancer) is unlikely altered by the primary treatment [166]. It is possible to improve the patient's prognosis by taking immediate action based on their condition. Therefore, capturing and monitoring the patient's condition have become more significant over the years. One way of monitoring the patient condition is observing the patient dataset. The Edinburgh Cancer Centre (ECC) is an environment containing National Health Service Lothian (NHS Lothian) cancer patient data from multiple resources [95].

Within NHS Lothian, the databases are divided into different categories based on their function: Direct Clinical Care Databases (DCCD) for recording, observing, and analysing the direct patient care; and Secondary/Derived Databases (SDD) for auditing and reporting purposes.

Table 1.1 ECC Datasources

<i>System/Team</i>	<i>Function</i>
PACS	Radiology Investigation System (DCCD)
ARIA	Verify and record system for XRT linear Accelerators (DCCD)
Trak	Patient Administration System (PAS) (DCCD)
ChemoCare	Chemotherapy ePrescribing System (DCCD)
Oncology Database	General cancer patients data (DCCD)
Trak Questionnaire	Ensure timely diagnosis and treatment (SDD)
Trak Module	Collect the quality performance indicator dataset (SDD)
Oncology Coding	Collect the whole patient pathway (SDD)
Hospital Coding	Collect the submission to national cancer registry (SDD)

Table 1.1 shows a brief overview of several DCCD and SDD databases. The system features the patient pathway at specific points. Hence, there is some transfer of information within each system (e.g. radiology, chemotherapy). The Edinburgh Cancer Centre (ECC) is data-rich; however, because of the lack of proxy, as shown in Figure 1.1, the system has no cohesive view of the patient's journey through cancer care. To facilitate this functionality, hospitals usually rely on several teams or manual input.

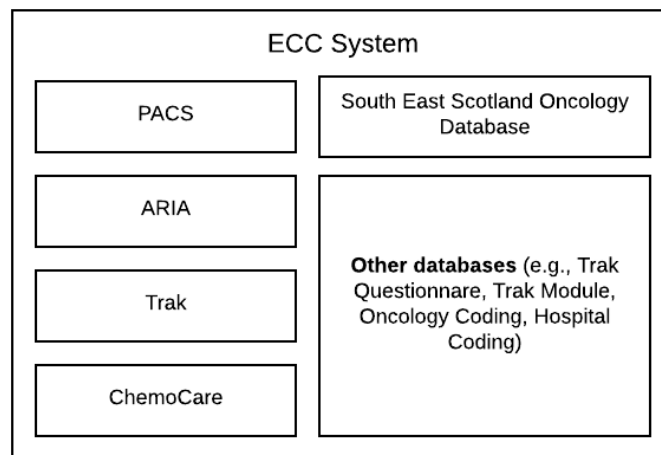


Fig. 1.1 ECC cancer patient data sources

At the moment, it requires significant time and effort to carry out a manual retrospective analysis of data held across multiple systems [95]. Motivated by the issues previously mentioned and the current NHS Lothian's objectives, such as improving patient care [95], we aim to improve the process of analysing the data. Jointly employed as staff at the NHS

Lothian and as an EngD student at the University of St. Andrews, we worked on several projects to tackle some issues mentioned above.

1.2 Objectives

Our projects are concerned with investigating, designing, developing, and implementing suitable frameworks, applications, or systems as a proof-of-concept to improve the use of data within the Edinburgh Cancer Centre (ECC). For four years, we worked on several different but integrated projects which supported the vision of contributing to better provision of cancer care in the future. Each project focuses on different objectives, as explained below:

- Patient Data Visualisation
 - The main objective of this project is designing a method to accurately visualise the patient pathway data from the South East Scotland Oncology Database. Initially, we observe the pathway progression at an individual level (e.g. in the clinic). Then, we expand the visualisation on a cohort basis to analyse against a set of metrics, including the time between treatments and the outcome against disease management protocols.
 - The secondary objective of this project is to help us familiarise ourselves with the different characteristics of cohort data across several different databases within the ECC.
- Data Migration
 - The main objective of this project is to migrate and update one of the primary databases for the general patient information (i.e. the South East Scotland (SES) Oncology Database) and use it as a data warehouse which integrates the overview/summarised of the patients' records from the other databases, such as, *ChemoCare* and *Trak*.
- Cancer Waiting Time
 - Cancer Waiting time (CWT) is the interval between the first referral by the General Practitioner (GP) to the patient's first treatment. During CWT, the optimal delivery of services during this time is crucial as it can provide further psychological benefits to patients [40]. This project aims to create a representation of the real pathway experience in accordance with the data compared with the guidelines initially suggested by the NHS Board. The outcome of this analysis

can highlight any possible bottlenecks, time, and inconsistent actions suggested by the CWT guideline, which will also help design the optimal CWT pathway in the future.

- Toxicity Predictor
 - The main objective of this project is to create a dashboard in the form of a reporting tool. It enables the clinicians to analyse the outcome of a chemotherapy treatment based on the cancer regimens, the patient’s condition, and characteristics. Ultimately, healthcare professionals can use the interpreted result as a further suggestion for determining and possibly revising the upcoming treatment of their patients.
- Generating Synthetic Cancer Care Data
 - One of the main issues we face when training several systems/models to predict the outcome of cancer treatments is the acquisition of real data when trying to use data to train and improve a model/system. This project aims to generate more data to train models better by using synthetic data that is close to the real data. With synthetic data, we can continue the process of improving our models as a proof-of-concept before enough patient data can be obtained in practice. We explore how to validate the quality of fabricated data with machine learning. We show how to formulate the rules that capture the essence of the data used in the toxicity predictor project to fabricate synthetic data.

1.3 Thesis Outline

The remainder of this thesis have the following chapters. Chapter 2 describes the knowledge/general information related to the projects carried out in this thesis.

Chapter 3 to Chapter 8 describe the details of the research projects carried out in the EngD. All projects are essentially self-contained but given in an order per their role in the cancer care trajectory, as shown below.

- Visualising the treatment care (Chapter 3).
- Bringing together data from multiple databases together (Chapter 4).
- Identifying potential bottlenecks causing potential delays in the start of cancer treatment (Chapter 5).

-
- Using data to improve treatment in real-time by creating toxicity predictors for chemotherapy treatments (Chapter 6).
 - Combining the visualisation and toxicity predictors (Chapter 7).
 - Exploring mechanism to facilitate the acquisition of data to enhance models such as those for predicting toxicity by fabricating synthetic data (Chapter 8).

Each chapter has its introduction, related work, methodology, implementation, and summary, depending on what is relevant for the project at hand. Lastly, Chapter 9 contains a discussion on the result obtained and a reflection on possible future work directions.

Chapter 2

Context

2.1 Introduction

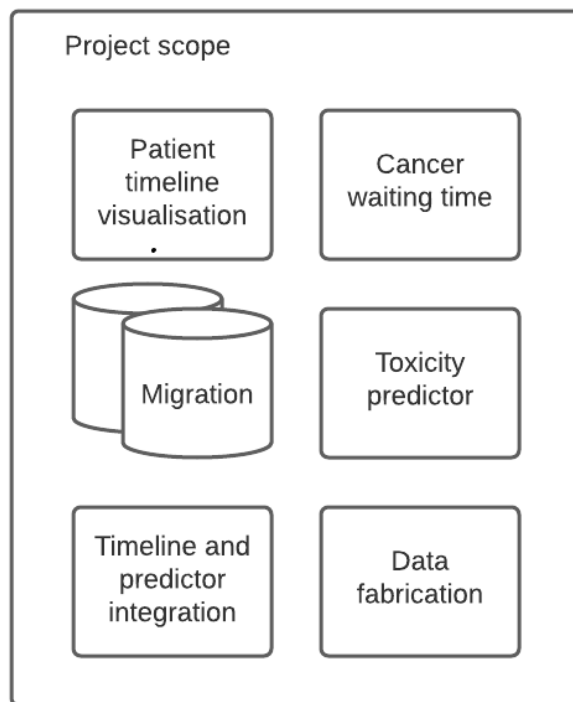


Fig. 2.1 Project Scope

The scope of our project which aims to facilitate the analysis and management of cancer care is shown in Figure 2.1. Before we analysed the data using state of the art methods, we familiarised ourselves with the characteristic of some medical data used in the Edinburgh Cancer Centre (ECC). Hence, our projects started with a proxy server and patient timeline visualisation implementation.

We then migrated one of the Oncology Database, previously known as the Designated Clinical Officer (DCO) Database [84]. Once we familiarised ourselves with the cancer data, we observed the journey of the cancer patients. We began analysing the cancer waiting time (CWT), a period before the patients get their first treatment.

We developed a toxicity predictor for the chemotherapy treatment's outcome. Lastly, to further facilitate cancer research, we fabricated the cancer dataset used in our projects.

This chapter provides a brief overview of the concepts, frameworks, tools, and libraries used for all our projects. Any specific notions relevant to an individual project are described in detail within the chapter for that project.

2.2 Concurrent Programming

All of our projects involve managing, observing, and/or analysing cancer care data. Our datasets vary in size. Some of the dataset, used for the migration project, has millions of data. It is necessary to perform several operations concurrently to allow efficient use of our resources and time (e.g. for the patient timeline visualisation (Chapter 3), migration (Chapter 4)).

We created our loading application as a concurrent application to accelerate the extract, transform, load (ETL) process for our migration project.

A concurrent program is a program which does more than one thing at the same time instead of processing it sequentially. Concurrency and parallelism are related but not the same [81]. Concurrency is the capability to handle many tasks during the same time span, while parallelism is the capability of processing many tasks literally at the same time. The task in a concurrent application would be sent and processed to separate processors for true parallelism. There are many types of mechanisms for concurrency. Two of the classic mechanism are multiprocessing and multithreading. Multiprocessing is the use of more than one central processing units (CPUs) in a single machine. Multithreading is the ability of the CPUs to create multiple threads to execute tasks simultaneously. Today, many applications require concurrency (e.g. for managing GUI while performing the underlying logic, queues up requests) to deliver their services on time [81].

2.3 Simulation Models

Simulation models are abstractions of a real-world system. When developing a model, we remove some characteristics from the system we want to simulate to allow essential

characteristic observations. Though the model does not contain every component of the real-world system we try to imitate, it can solve some real-world problems safely and efficiently. Using models to simulate specific cancer care events, for instance, the events from the first general practitioner (GP) referral to the first treatment, we can comprehend the Cancer Waiting Time (CWT) process and its possible bottleneck.

In one of our projects (i.e. Chapter 5), we developed a Discrete Event Simulation (DES) model to compare the guided pathway to the pathway generated from the dataset for CWT within NHS Lothian.

2.3.1 Discrete Event Simulation

Discrete event simulation (DES) is a method of simulating the behaviour and performance of a real-life process, facility or system [7]. It represents a real-world event as a series of instantaneous occurrences. The discrete event system satisfies the following properties: the state space is a discrete set while the state transition is event-driven [28]. In between these events, the system is approximated as fixed and unchanging. In other words, we assume that nothing happens between two consecutive events [9]. The DES can model simple real-world events such as machine states to a complex system like the human biological ecosystem. Here are several examples of discrete event systems:

- Dungeon states in a video game as EXPLORED or UNEXPLORED
- A simplified version of human appetite as FULL, HUNGRY, RAVENOUS

With the increasing speed and memory of computers to process high complexity problems, we can simulate most discrete-event systems with the components shown in Figure 2.2. It highlights each component interaction. At the start of the simulation, the main program will invoke the initialisation routine, timing routine, and event scheduler. The event scheduler will update the system current state. When the system state reaches its final state, the simulation will be terminated. Otherwise, the simulation will keep on iterating accordingly.

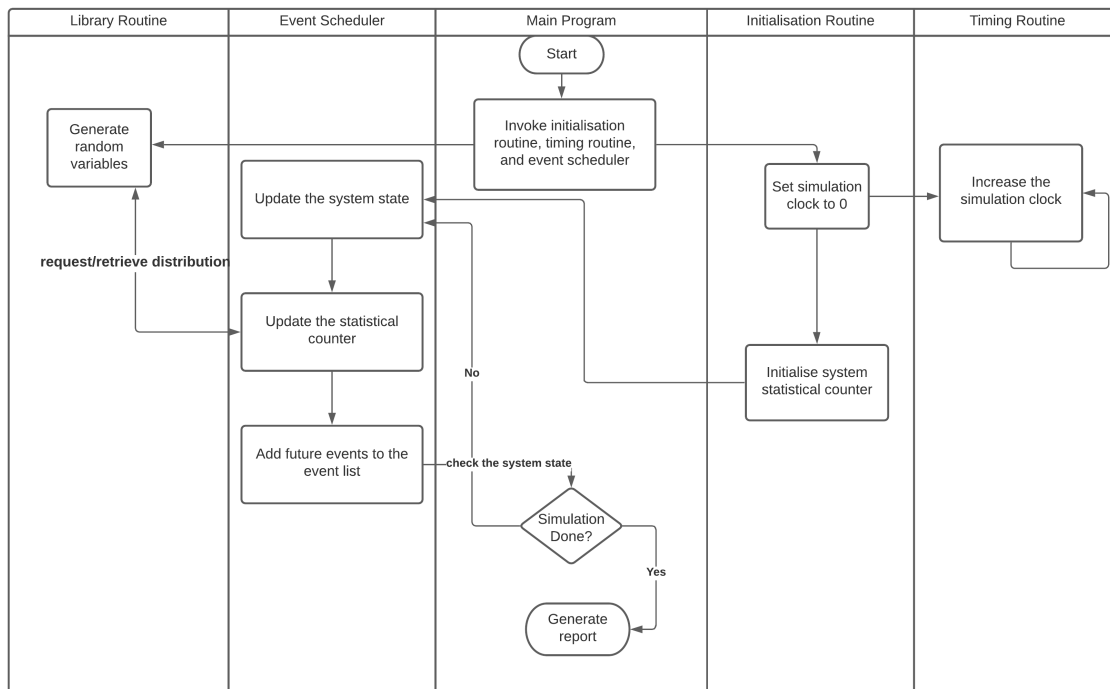


Fig. 2.2 Discrete event simulation flowchart

2.4 Constraint Programming

One of the main issues we encounter during the machine learning model development is the lack of relevant data for creating a good model. Real cancer data is difficult to get. Obtaining more data requires time, as well as resources. In our case, even with access to cancer data, the machine for accessing the data might not have the appropriate driver, application, or software to do advanced data analytics.

To tackle these issues, we fabricated the data so we can further explore and develop our applications. There are several ways to perform data fabrication, including using machine learning or constraint solvers. However, machine learning models require access to the real-data to infer some data characteristic for fabricating the data. Access to private information, such as cancer data is limited. It is also impossible to transfer the data outside the environment. For this reason, we use other alternatives to generate the dataset (i.e. constraint solver). Together with IBM Research in Haifa [74], as part of the serums project [77], we developed a set of rules to create synthetic data. We were able to develop rules to generate fabricated data without requiring direct access to the data sources by using a constraint solver [4].

Constraint Programming (CP) is a paradigm/technique for solving constraint satisfaction problems (CSP). A CSP is a problem that requires its solution to be within some conditions, known as constraints. A CSP requires a selected value from a finite domain to be assigned to each variable in the problem [23]. The aim is that all constraints relating variables are satisfied. Hence, the CP is based on finding feasible solutions rather than finding the optimal solution (i.e. optimisation) [60]. It focuses on the constraint and variable.

Formally, a CSP is a tuple (X, D, C) [141]:

$$X = \{x_1, x_2, \dots, x_n\} \quad (2.1)$$

$$D = \{d_1, d_2, \dots, d_n\} \quad (2.2)$$

$$C = \{c_1, c_2, \dots, c_m\} \quad (2.3)$$

Where:

- X is the set of variables.
- D is the set of domains (i.e. d_i is a finite set of potential values for x_i).
- C is a set of constraints.

A constraint c is a pair of (S, R) :

$$S = (x_{i1}, \dots, x_{ik}) \quad (2.4)$$

$$R \subseteq d_{i1} \times \dots \times d_{ik} \quad (2.5)$$

Where:

- S are the variables of C (i.e. scope)
- R are the tuples satisfying C (i.e. relation)

Solving a CSP involves finding a solution for the variables, for example:

$x, y, z \in \{0, 1\}, x + y = z$ is a CSP, where the variables are x, y, z and the domains are $d_x = d_y = d_z = \{0, 1\}$. We have a single constraint: $x + y = z$. The solution for this CSP is $((x, y, z), \{(0, 0, 0), (1, 0, 1), (0, 1, 1)\})$.

2.5 Machine Learning

One of our projects aims to consider data analysis and use such analysis information to help the health professional observe the upcoming treatments.

Machine learning (ML) is now a commonly used technique for data observation. The use of machine learning allows a model to learn from data and infer its properties.

In our context, we used machine learning to create various models to predict the chemotherapy outcome. This section presents and compares various methods, which we later explore to create suitable predictors for our chemotherapy dataset.

Machine Learning [8] is a study of algorithms and statistical models to optimise a performance criterion by learning from example data or past experience. Machine learning helps users find solutions to many problems in various domains, such as speech and image recognition, robotics, or even healthcare [44].

It involves training a model to infer data properties or pattern from the training sample and then use the model to predict outcomes. We train the model with extracted data. Here, an efficient algorithm is necessary to solve the optimisation problem and store and process its data. In many applications, the learning/inference algorithm measured by space and time complexity may be as crucial as its predictive accuracy.

Below are examples of machine learning applications and their uses.

2.5.1 Association rules

Association rule is a rule-based machine learning method that shows the probability of relationships between the dataset variables. Here, machine learning modules are used to determine a conditional probability of $P(Y|X)$. Where:

- Y is the events/products/information we want to observe, based on X .
- X is the known events/information.

For example, we want to know the likelihood of a bag of potato crisps bought together with beer, sweets, and biscuit.

2.5.2 Classification

In machine learning, classification is used to assign a class label to the testing instances from the predictors' value. A classifier can be either supervised or unsupervised [146]. One example of classifier machines is a pattern recognition machine such as optical character

recognition. Here, the machine learning model is used to recognise character codes from their image.

2.5.3 Regression

Regression analysis focuses on performing a statistical process of mapping the input to output in a dataset. It is a supervised learning problem when there is an input X and output Y . For regression, we determine the mapping function from the input to the output where a model is assumed to have a set of parameters:

$$Y = g(X|\theta) \quad (2.6)$$

Where:

- $g(\cdot)$ is the model.
- θ are the model parameters.

$g(\cdot)$ is a regression function. In a classifier, $g(\cdot)$ is the discriminant function. Recommendation system (e.g. movies recommendation system) is an example of regression. Here, we want to generate a list of ordered items based on the rate which is calculated from the regression function.

2.5.4 Unsupervised Learning

The goal of the machine here is to learn to produce the correct output given the input. However, unlike supervised learning, we do not have the desired outputs in the input data [58].

Here, we want to find the regularities in the input and then group the input accordingly based on its characteristics. This process is known as clustering. We can use clustering analysis to get an insight by observing which groups the input fall into when we apply a clustering algorithm (e.g. k-means, Expectation-Maximisation (EM) clustering) [2].

2.5.5 Reinforcement Learning

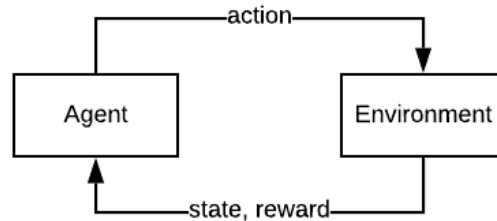


Fig. 2.3 An interaction between an agent and its environment

Reinforcement learning is a branch of machine learning concerned with developing agents to react with their environment to maximise rewards. It involves learning what to do, how to map the state into action. Here, the learning system action influences the input. Reinforcement Learning has four main components as follow [163].

The constant interaction between agents and the environment is an essential component of the model. As shown in Figure 2.3, the learning system action influences the input. Reinforcement Learning has four main components as follow.

- Policies determine learning agent behaviour or actions.
- A reward signal defines the goal of the reinforcement learning problem.
- A value-function specifies what is suitable in the long run. The value of a state is the total amount of expected reward accumulated by an agent, starting from that state.
- A model of an environment, the world in which an agent performs actions.

2.6 Methods

2.6.1 Cox Regression

A Cox proportional hazard model is a statistical technique for survival analysis. In survival analysis, we are concerned with the time between entry to a study and a subsequent event (e.g. death) [174]. However, sometimes we may have additional information aside from duration. With the Cox model, we observe the hazard rate $h(t|x)$ as a function of t and some covariates x by regressing the covariates (e.g. age, weight) against the duration/time as shown

in equation 2.7. It is the product of the baseline hazard and the exponential function of the linear combination of the predictors (i.e. the partial-hazard) [93].

$$\underbrace{h(t|x)}_{\text{hazard}} = \underbrace{h_0(t)}_{\text{baseline hazard}} \underbrace{\exp\left(\sum_{i=1}^n b_i(x_i - \bar{x}_i)\right)}_{\text{partial hazard}} \quad (2.7)$$

Where:

- t represents the survival time
- $h(t|x)$ is the hazard function.
- x_1, \dots, x_n are a set of n covariates.
- The coefficients b_1, \dots, b_n measure the impact of the covariates.

As the name implies, the Cox proportional hazard assumes independence of survival times between distinct individuals in the sample, a multiplicative relationship between the predictors and the hazard, and constant hazard ratio over time [89].

2.6.2 Markov Model

Markov model is a stochastic model to observe a decision problem which involves a risk that is continuous over time [85]. The simplest form of Markov Model is a Markov chain. The Markov chain can give us information related to the probabilities of a sequence of random variables. It makes the prediction solely based on the current state. The Markov model follows a Markov assumption (i.e. the states before the current state have no impact for predicting the next state probability).

$$\text{Markov Assumption} : P(q_i = a | q_1, \dots, q_{i-1}) = P(q_i = a | q_{i-1}) \quad (2.8)$$

Where:

- q_1, \dots, q_{i-1} are a sequence of state variables.
- a is the transition probability

Formally, a Markov chain consists of the following components [80]:

- a set of N states, $Q = q_1 q_2 \dots q_N$

- a transition probability matrix A , $A = a_{11}a_{12}...a_{n1}...a_{nn}$. Each a_{ij} represents the probability of moving from state i to state j , $\sum_{j=1}^n a_{ij} = 1 \forall i$.
- an initial probability distribution over states, $\pi = \pi_1, \pi_2, \dots, \pi_N$. Some state may have zero probability, meaning that they cannot be initial states. Also, $\sum_{i=1}^n \pi_i = 1$

2.6.3 Hidden Markov Model

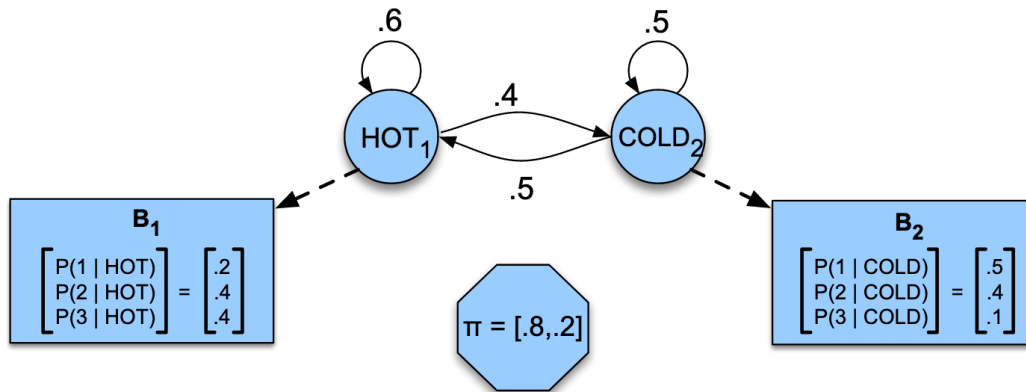


Fig. 2.4 An example of an HMM model. Source: [80]

Hidden Markov Model (HMM) is a statistical Markov Model which has additional hidden states. HMM allows us to observe the observed and hidden states/events. HMM is widely used in the Natural Language Processing (NLP) field (e.g. part-of-speech tagging [88]). In part-of-speech tagging, we want to know the category of words. Here, the input words are the observed states while the word categories (e.g. subject, adverb) are the hidden states. The HMM contains the Markov chain with additional components [80]:

- a sequence of T observations, $O = o_1 o_2 \dots o_T$.
- a sequence of observation likelihoods (i.e. emission probability), $B = b_i(o_t)$. The emission probability expresses the probability of an observation o_t being generated from a state i .

A first-order Hidden Markov Model has two simplified assumptions as follows:

$$\text{Markov Assumption} : P(q_i = a | q_1, \dots, q_{i-1}) = P(q_i = a | q_{i-1}) \quad (2.9)$$

$$\text{Output Independence} : P(o_i | q_1 \dots q_i, \dots, q_T, o_1, \dots, o_i, \dots, o_T) = P(o_i | q_i) \quad (2.10)$$

As shown in the equation 2.10, the probability of an output observation o_i depends only on the state that produces the observation q_i .

Figure 2.4 shows an example of Hidden Markov Model for the "ice cream task" introduced by Jason Eisner [47]. The two hidden states (i.e. *HOT*, *COLD*) correspond to the hot and cold weather. The observations (i.e. $O = 1, 2, 3$) correspond to the number of ice cream eaten on a given day.

2.6.4 Random Forest

Ensemble learning is a type of learning which combines the result of many models (e.g. classifier) to predict an outcome. Two of the most common ensemble learning techniques are bagging and boosting of the classification tree [92]. Bagging introduces a new model that is created by aggregating multiple models and averaging their prediction result to yield a new outcome (i.e. mean for regression or voting for classifier). In classifier, the model typically collect votes. However, if the classifier returns class probability, we can use the mean average similar to regression to improve the model predicting capability.

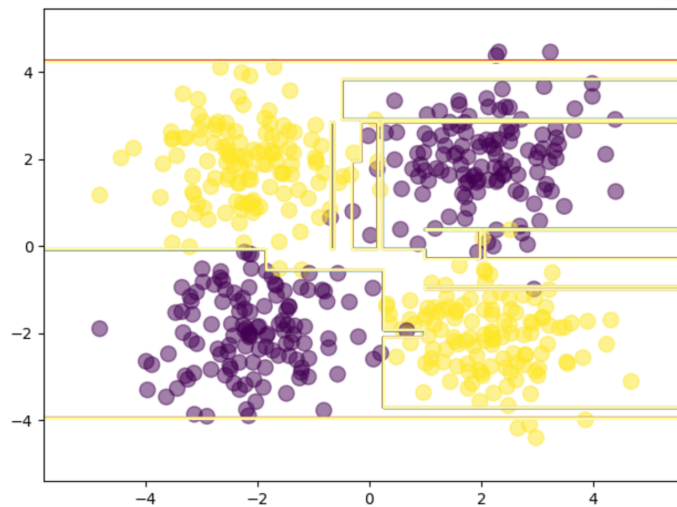


Fig. 2.5 Decision tree classification. Source: [90]

For example, a decision tree can go exceptionally deep to get 100% prediction accuracy. Figure 2.5 shows an example of a decision tree classifier boundary with 100% prediction accuracy. By using the boundary inferred from the input data, it can predict the training dataset category at a cost. The model has a problem with over fitting (i.e. low-bias and high-variance).

Bagging can reduce the high-variance problem, as shown in Figure 2.6. Although by using bagging, the model will have better accuracy when predicting the test dataset, performing

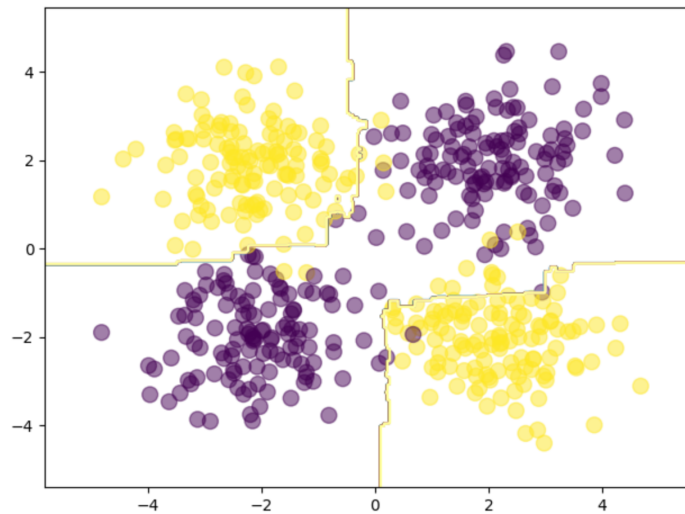


Fig. 2.6 Bagging classification. Source: [90]

bagging alone might not be enough as the trees that compose the new model may have a high correlation with each other.

Breiman introduced the Random Forest (RF) algorithm by adding a layer of randomness to the bagging by features selection in 2001 [24]. Random Forest can de-correlate the trees by choosing the features for each tree's training dataset. The number of features, m , depends on the number of predictors, as shown in equation 2.11 [54]. Because of the feature selection, Random Forest is also called feature bagging.

$$m = \begin{cases} \text{floor}(\sqrt{D}), & \text{for classification} \\ \text{floor}(\frac{D}{3}), & \text{for regression} \end{cases} \quad (2.11)$$

Where D is the number of predictors.

2.6.5 Artificial Neural Network

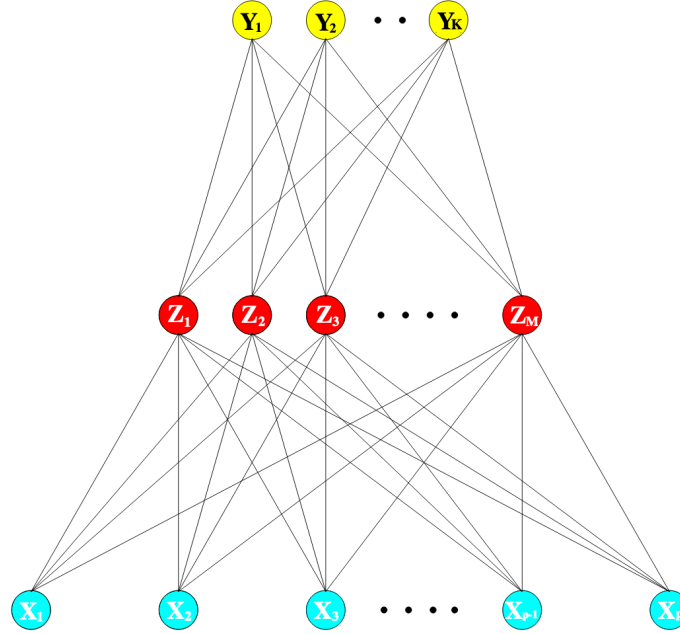


Fig. 2.7 Schematic of a single hidden layer, feed-forward neural network.

Neural Network is one of the machine learning techniques inspired by the human brain. A typical neural network consists of several layers, input layer, hidden layer, and output layer, combined as a network diagram. Figure 2.7 shows an example of a feed-forward neural network with one hidden layer. For K -class classification, there are K target measurements $Y_k, k = 1, \dots, K$. Equation 2.12 shows the relation between layers [55].

$$\begin{aligned}
 Z_m &= \sigma(\alpha_{0m} + \alpha_m^T), m = 1, \dots, M, \\
 T_k &= \beta_{0k} + \beta_k^T Z, k = 1, \dots, K, \\
 f_k(X) &= g_k(T), k = 1, \dots, K
 \end{aligned} \tag{2.12}$$

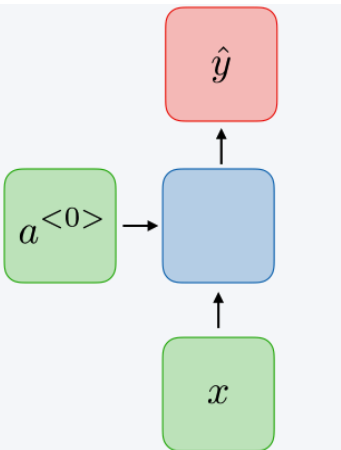
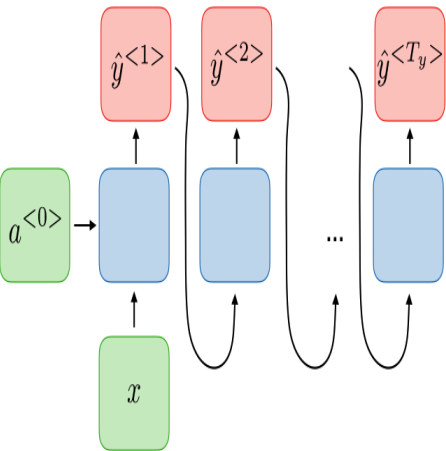
Where α and β are the weights.

There are several different types of activation/transfer function we can choose (e.g. *sigmoid*, *linear*, *ReLU*[5]). Depending on the activation function, it maps the input values in between 0 to 1 or -1 to 1, or any other values. It is also common to add a bias unit to the hidden and output layer. Lastly, it is common for classification Neural Network to use *softmax* function, g_k , to transform the vector output T into the desired output type. The *softmax* function normalises the output of the hidden networks to a probability distribution over predicted output classes.

2.6.6 Recurrent Neural Network

Recurrent Neural Network (RNN) is a class of Neural Network that allows the sequence processing of its input and output. Compare to the feed-forward Neural Network, RNN offers much flexibility in term of its input and output size [82]. Table 2.1 shows different applications of RNN based on its input and output size [138]. Even if we do not have the notion of sequence in the dataset, it is possible to perform sequence processing of fixed input [67].

Table 2.1 Different type of RNN

Nodes	Info
	<ul style="list-style-type: none"> • One-to-one • $T_x = T_y = 1$ • e.g. Traditional Neural Network • T is timestep of the input/output
	<ul style="list-style-type: none"> • One-to-many • $T_x = 1, T_y > 1$ • e.g. Music generation

	<ul style="list-style-type: none"> • Many-to-one • $T_x > 1, T_y = 1$ • e.g. Sentiment classification
	<ul style="list-style-type: none"> • Many-to-many • $T_x = T_y$ • e.g. Name entity recognition
	<ul style="list-style-type: none"> • Many-to-many • $T_x \neq T_y$ • e.g. Machine translation

At every timestep, the RNN updates its state by applying the recurrent formula to a sequence of vector x , as shown in equation 6.2 [138].

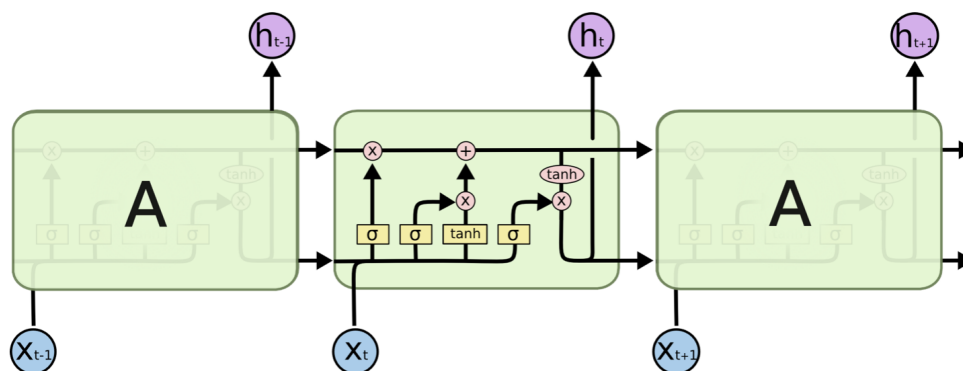
$$\begin{aligned} a^{<t>} &= g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a), \\ y^{<t>} &= g_2(W_{ya}a^{<t>} + b_y) \end{aligned} \quad (2.13)$$

Where:

- $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ coefficients that are shared temporally
- g_1, g_2 are the activation function.

Similar to the feed-forward neural network, there are many kind activation functions (e.g. *sigmoid*, *Tanh*, *RELU*).

Long Short Term Memory (LSTM)



Where:

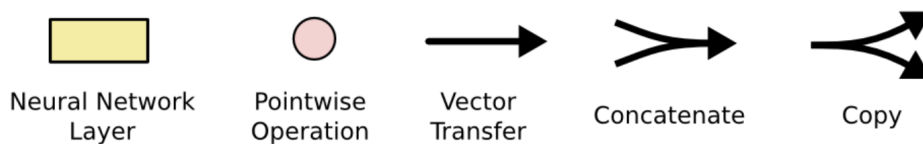


Fig. 2.8 An example of LSTM cell. Source: [97]

The LSTM is a type of recurrent neural network with the capability of learning long-term dependency. It is specifically designed to tackle the vanishing gradient problem with RNN [152] (i.e. the problem relates to the recurring weight). Over time, the weight of the RNN network may explode or vanish. When we update the RNN state, we multiply the weight multiple times. The multiplication may cause an issue. For example, when we multiply

something by a small number, the value decreases very quickly and vanishes over time. The LSTM solves this problem by introducing several other variables and gates as shown in Figure 2.8.

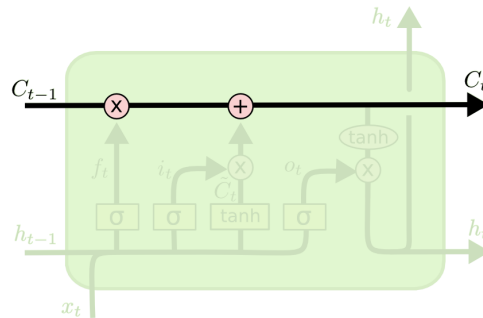


Fig. 2.9 LSTM cell control gate. Source: [97]

The LSTM cell is more complex than the vanilla RNN. With the additional gate (e.g. control gate (C)), LSTM can remove or add cell information to the state. The overall improvement comes from their additive interaction as it updates the gradient flow of a network as shown in Figure 2.9. Currently, there are many types of research regarding the recurring function of the LSTM [66]. Along with its variants (e.g. GRU, LSTM cell performs much better and more widely used compared to the vanilla RNN [82].

2.7 Summary

This chapter introduces some key concepts and model used for some of the different projects that make up this thesis. Further details on specifics libraries are given in the appendix.

Chapter 3

Patient Timeline Visualisation

3.1 Introduction

Data is a vital asset in many organisations but is often too fragmented and crude to be useful. We gain more insight from the information it contains by integrating data from a variety of sources. Understanding, analysing and building models from complex data has the potential to improve decision making. This is the case for many domains and is particularly so in our healthcare domain when aiming to improve cancer care.

Data integration is the process of combining/aggregating data from different sources to provide meaningful and valuable information to the end-users. There are several ways to perform data integration [70], such as building an enterprise warehouse and creating a proxy server. In the case of the former, it corresponds to creating a centralised database that holds all the business information of an organisation and makes it accessible across the company. By contrast, in the case of the latter, it corresponds to creating an application which acts as a proxy to provide the data to the end-users directly from various sources/servers. Both approaches can be used for heterogeneous data as well. When data is provided in different formats, it is the responsibility of the database or proxy to make sense of the heterogeneous nature of the data.

In this chapter, we describe the work carried out to provide a proxy between the different (sub)systems within the ECC. The developed *South East Scotland Oncology (SESO) Gateway* can be seen as one of the analytical solutions for improving the quality and capability of real-time outcomes reporting within South East Scotland using routinely captured and integrated electronic healthcare data. The users, which for this system are going to be mainly the clinical oncologists, can view the pathway progress at an *individual level*: that is, during the clinic and when deciding whether the treatment should be continued for a particular patient. Alternatively, users can also focus on a *cohort view* to analyse a treatment against

a set of metrics, such as time between treatment, cancer screening procedures to follow a particular pathway, the outcome against disease management protocols, and so on. Hence, clinicians can use cohort information to better target the care of a new patient with similar characteristics.

The ultimate goal of this project is to investigate ways to engage more personnel within the NHS Lothian to use their data more thoroughly. In this project, we have primary and secondary objectives as the main driver of developing the *SESO Gateway*. The objectives are as follows.

- **Primary:** Design a method to accurately visualise patient pathway data from the South East Scotland Oncology Database (Oncology DB). A user can view the pathway progress at an individual level, for example, in-clinic, as well as a cohort basis with respect to a set of metrics (e.g. the time between treatment, treatment's pathway, the outcome against disease management protocols).
- **Secondary:** Assess transferability and extensibility of the approach by applying the same method to a dataset that includes *ChemoCare* data to evaluate the effect of the use of drugs, time on treatment, and relative dose intensity within the time between lines of therapy (duration of disease control).

3.2 Related Work

SESO Gateway extends earlier work from the University of Edinburgh to discover predictors of the chemotherapy toxicity in breast cancer patients from the *ChemoCare* data extraction [179]. The data was extracted, transformed, and processed using multivariate logistic regression, as shown in Figure 3.1.

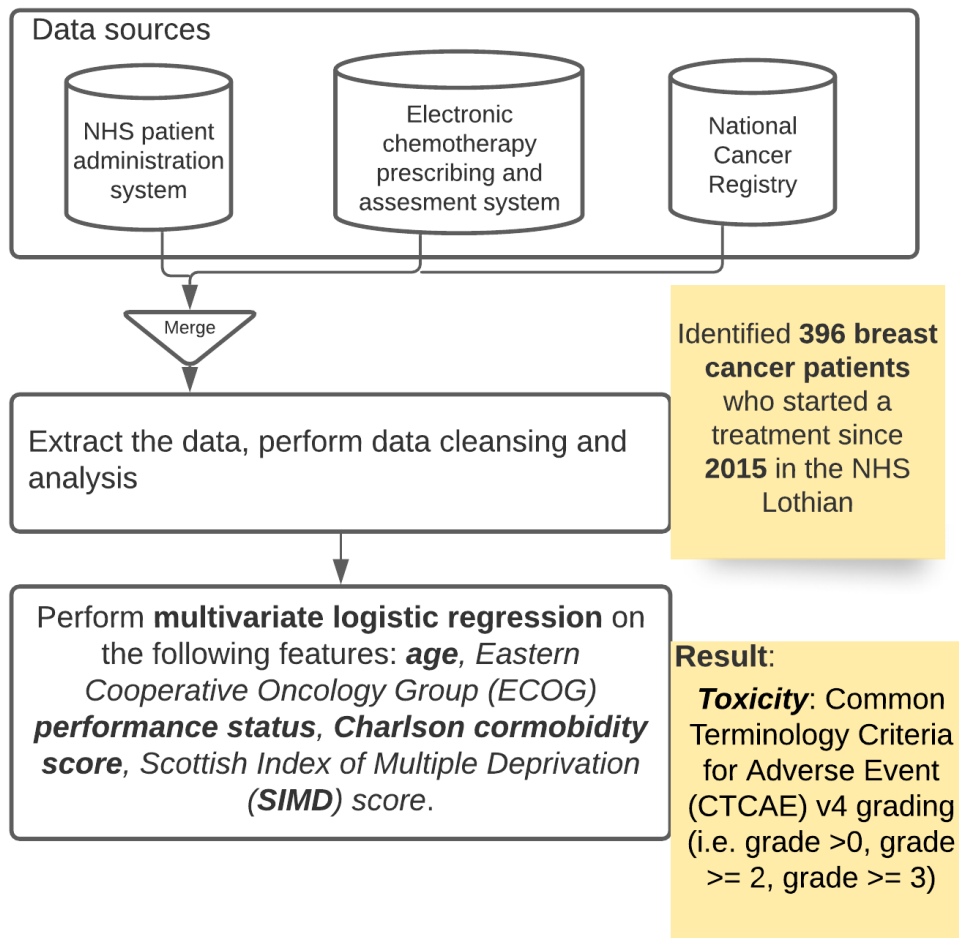


Fig. 3.1 An example of *ChemoCare* data observation

The study aimed to identify the toxicity rate from commonly used anti-cancer regimens for the NHS Lothian patient populations and finding the baseline patient characteristics for predicting the excess toxicity. Figure 3.1 shows that it is possible to profile the pattern of cancer patients, which helps in predicting the side effects of the chemotherapy regimen, such as toxicity by using regression technique against several patient's characteristics.

The use of data visualisation helps the users to understand the concept and underlying patterns of some data sources. The data visualisation amplifies the benefits of health informatics databases and networks by dramatically expanding the capacity of patients, clinicians, and public health policy-makers to make better decisions [154]. As a result, the use of the multivariate application is becoming more widely adopted in many fields, such as healthcare.

It is widely recognised that integration with existing healthcare systems may help healthcare professionals make decisions and improve the treatment of their patients.

Several existing visualisation tools have inspired the making of *SESO Gateway* (e.g. LifeLine [128]). *LifeLine* is a visualisation tool to enhance navigation and analysis of patient records. It provides a general visualisation environment for personal histories. The *LifeLine* prototype was developed in 1997 as a research project between IBM Research and the University of Maryland. The basis for modelling the record was a newly operational clinical information system at Kaiser Permanente Colorado.

3.3 Requirements

The *SESO Gateway* has both functional and non-functional requirements to guide us in developing the application.

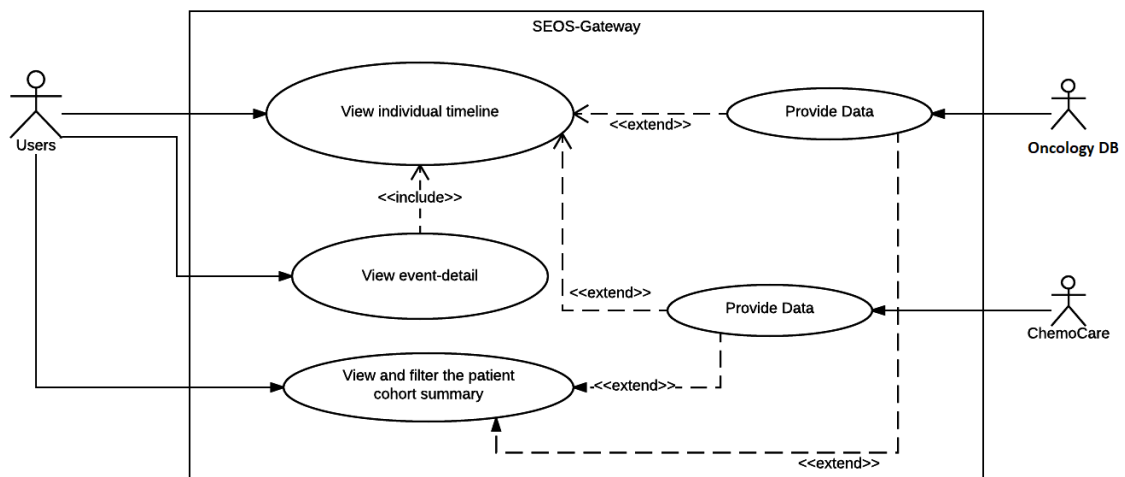


Fig. 3.2 The *SESO Gateway* features

Figure 3.2 shows a use case illustrating the functional requirements for the *SESO Gateway*. The list of requirements is detailed below. Some of the functionalities are related as shown through the *include* and *extend* relationships between use cases.

- **Functional requirements**

- Users should be able to view the individual’s event-timeline.
- Users should be able to get detail information about each event in the timeline.

- Users should be able to view the cohort summary for various kind of treatment.
- Users should be able to search/filter for the summary of patient’s cohort.

- **Non-functional requirements**

- The system is designed for the NHS Lothian. In the future, the system should be compatible to be deployed in the NHS Lothian Network.
- The primary users of the system are the Western General Hospital (WGH) Edinburgh Cancer Centre oncologists and other users working in the field of Cancer treatment within NHS Lothian (e.g. managers, audit team) which consists of no more than 100 personnel within the NHS Lothian.

3.4 Data Analysis

3.4.1 Data Source

The Oncology DB was established in 1974. The database was written in Common Business-Oriented Language (COBOL) [33]. In 1979, a significant rewrite of the software took place to remove all system bugs. The database was written using Statistical Analysis System (SAS)[145] in 1992 . However, the restricted SAS licence eventually rendered this impossible. As there were still several aspects of the system which were rendered unsatisfactory, another consultant was employed in 1996 to refine the structure of the database [95].

With the increasing number of staff relying on using the system (users) at the same time, it became a necessity to work with multiple resources. Attempts to set up a multi-user input process using SAS failed, consequently NHS Lothian decided to rewrite the system using Microsoft Access [111] instead.

Although Microsoft Access facilitates access from multiple systems, several issues emerged as the user base grew. In particular, since all users should have write-read access to the database, the Oncology DB becomes less secure. Also, the Microsoft Access database does not scale well to many simultaneous users at once. When a user accesses the database in the shared folder, a lock file is created, which prevents other users from accessing the database. Hence, in 2018 the NHS Lothian started another migration project to migrate the Oncology DB into a Microsoft Structured Query Language (MSSQL) Server.

For our project, we created another database during the development of *SESO Gateway* to keep our development independent from the ongoing migration work. We performed a simple migration from the Microsoft Access *.mdb* file into *SQLite* [158] database which

has the same core properties as the MSSQL server, as both *SQLite* and MSSQL server are relational database management systems (RDBMSs) [68].

One of the benefits of using *SQLite* is that it allows us to focus on the design of the data visualisation rather than the connection between the application to the Database. As an embedded SQL database engine, *SQLite* does not require a separate server process.

The process of migrating into a *SQLite* format is straightforward. We created a python script to access the Microsoft Access *.mdb* file and convert it into a *SQLite* format.

3.4.2 Data Cleansing

During development, we created several batches of patient data. Each batch contains 10,000 patient's data extracted from the Oncology DB. Each patient has several events (i.e. around 10 - 15 events, including diagnosis, hormone therapy, surgeries). First, we perform data cleansing. In this process, we fix some incorrectly formatted data (e.g. Community Health Index (CHI) as *varchar* instead of *integer*), impute some missing fields, and remove the duplication of data. We exclude data containing null values on several fields (e.g. the Cause of Death (COD), type of cancer, cancer's site). After filtering the null values, we remove duplicated data by using aggregation in our SQL's query. We obtain a distinct value for those duplicated data by grouping the patient based on specific fields (i.e. *first seen date*, *episode*, *site*, *method*, *duration*).

3.5 Implementation

The *SESO Gateway* project is a multiple-domain project as shown in Table 3.1.

Table 3.1 *SESO Gateway* project scope

Domain	Computer Science	Data Analysis	Graphic Design
Process	Acquire, Parsing	Filter, Analysis	Represent, Refine

The scope of the project includes:

- Designing a solution for delivering patient data visualisation from various databases in the NHS Lothian.
- Creating sandbox databases for the project.
- Creating the *SESO Gateway* modules (i.e. Front-end, Back-end).
- Integrating various sandbox databases.

The scope of the project does not include:

- Using the real databases for the application. During the project development, the developer does not have any access to the migrated Oncology DB as well as the *ChemoCare* server.
- Deploying the application within NHS Lothian network.
- Integrating security into the application.

3.5.1 System Design

The *SESO Gateway* has various components that allows user to enter, query, and search for a specific patient's timeline or a cohort summary visualisation. The software architecture is designed to combine all data and its modifications from integrated databases which consist of various treatments data. The system includes several sub-components: the front-end, the back-end, and mock databases. This architecture enhances flexibility since it provides more opportunities for replacing entities (at either end) depending on the NHS Lothian needs.

Front-End

The system front-end/presentation layer is a web-client developed using the *AngularJS* 1.x framework [10]. The *AngularJS* framework increases the modularity of the *SESO Gateway*.

We only considered frameworks written in JavaScript because JavaScript is one the most common languages used to develop client side applications. Hence, we compared two of the most popular frameworks during the development, *AngularJS* and *React* [176]. *AngularJS* is a full framework with libraries, architectural design, and various web-client tools. *React*, on the other hand, is a small view library that is needed while making a web client [176]. We choosed *AngularJS* because it provides more compact tools than *React*. This allows us to focus more on the data analysis part. By combining the *AngularJS Two-Way* data binding feature, and the Model View Controller (MVC) architectural pattern, *AngularJS* provides a separation of concerns between the User Interface (UI) and the business logic. The *SESO Gateway* front-end serves several pages, as in the following. In this section, we show the list of website page schematic/wireframes in the *SESO Gateway*.

Homepage wireframe The homepage consists of three main components (i.e. navigation bar, header, and the body containing the general information about *SESO Gateway*). The header includes the name of the application and a login button. Ultimately, we want to integrate this application with the credentials used within the NHS Lothian. The body

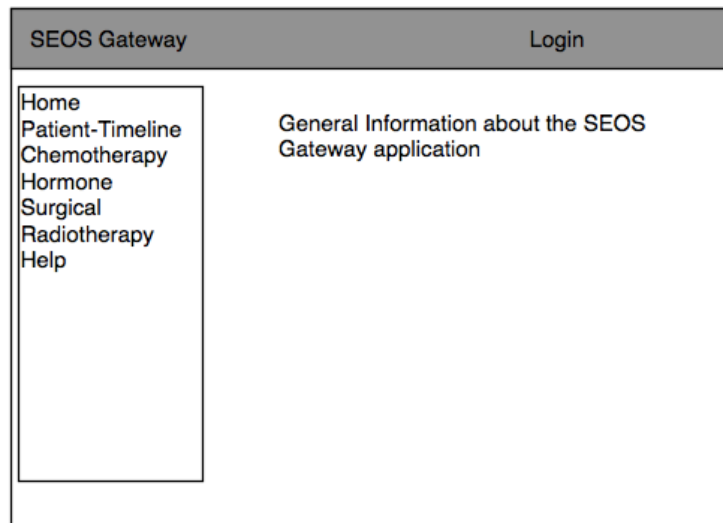


Fig. 3.3 *SESO Gateway* homepage wireframe

contains the general information about the *SESO Gateway* and a navigation bar containing the links to the other application pages as shown below.

- *Home* navigates the users back to the homepage.
- *Patient Timeline* navigates the users to the event-timeline page. It allows the users to observe the events for a specific patient based on the patient identification number and Community Health Index (CHI) (i.e. for integrating the timeline with the information obtained from the *ChemoCare* data extractions).
- *Chemotherapy* navigates the users into a page showing the cohort summary for chemotherapy treatments in the Western General Hospital (WGH).
- *Hormone* navigates the users into a page showing the cohort summary for hormone therapy treatments in the WGH.
- *Surgical* navigates the users into the summary page about the surgical procedures extracted from the Oncology DB.
- *Help* navigates the users to the help page.

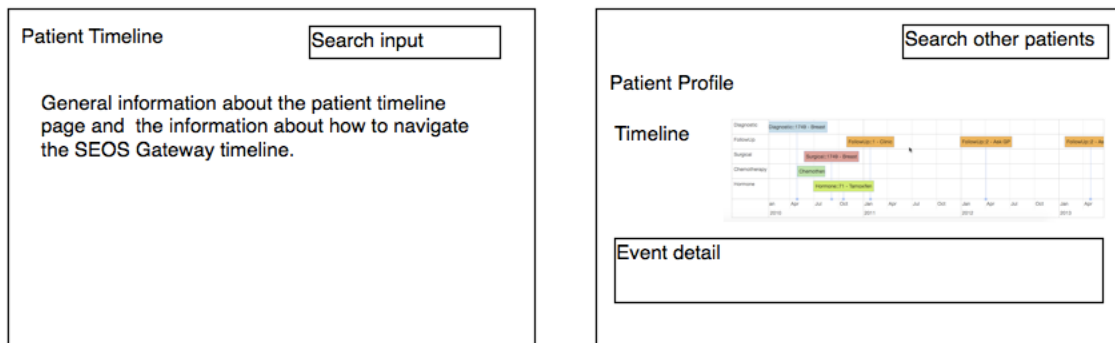


Fig. 3.4 *SESO Gateway* individual timeline wireframe

Individual timeline wireframe With the *AngularJS* framework, we can implement an Single Page Application (SPA) for the *SESO Gateway*. An SPA is a web application which allows a one-time load of an HTML page and dynamically updates the page as the user interacts with the application [76]. The *AngularJS* framework allows us only to modify the body of our application if there is no significant change to the other components (i.e. the header and the navigation bar), the wireframes only show the body of the webpage unless specified otherwise. Figure 3.4 shows the *SESO Gateway* individual timeline wireframe. It has two main pages. The first page is a welcome page for the event-timeline. It consists of two main components, the search input, and the general information. If a user enters the patient's identification number and clicks the search button, it navigates the user to the second page. This page consists of three components (i.e. the patient profile, the event timeline, and the event detail). Once the application loads the patient data, both patient's profile and event-timeline sections will be updated. The users can look at the detail of each event by clicking on the span object located in the event timeline component. The timeline has a button to integrate the data taken from the *ChemoCare* data extractions.

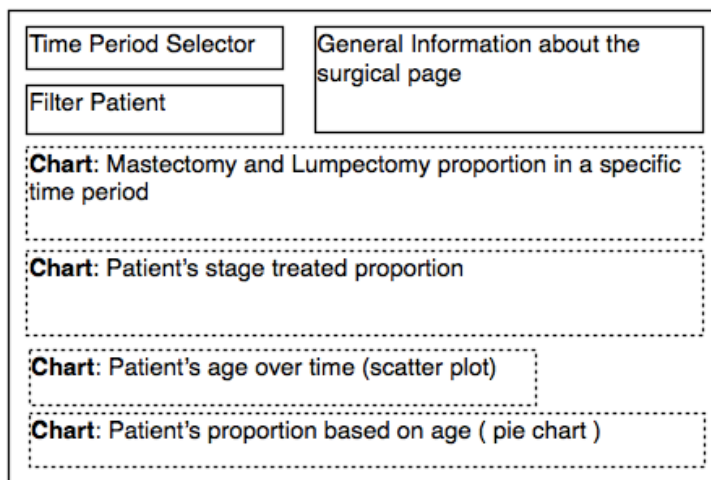


Fig. 3.5 *SESO Gateway* surgical wireframe

Surgical wireframe The surgical page has two components: the filter selector and the charts section. A user needs to specify the time period before observing the cohort summary. This action filters the patient's summary based on the selected time. The application uses a default time (i.e. previous ten years from the current year) if no period is selected. Once loaded, a few charts appear below the filter selector. Figure 3.5 shows various information about the surgeries performed based on the data from Oncology DB. In the later section, we further discuss the method to obtain the information for populating the charts.

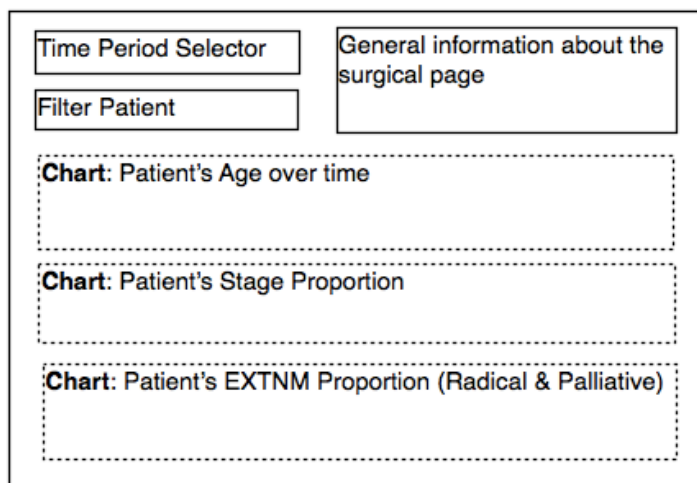


Fig. 3.6 *SESO Gateway* radiotherapy wireframe

Radiotherapy wireframe Similar to the surgical page, the radiotherapy page has two components (i.e. the time selector filter and the charts section) as shown in Figure 3.6. Once the filter is selected, the user can obtain the cohort summary extracted from the Oncology DB. The charts section then visualises the data for the radiotherapy treatments performed in Western General Hospital (WGH).

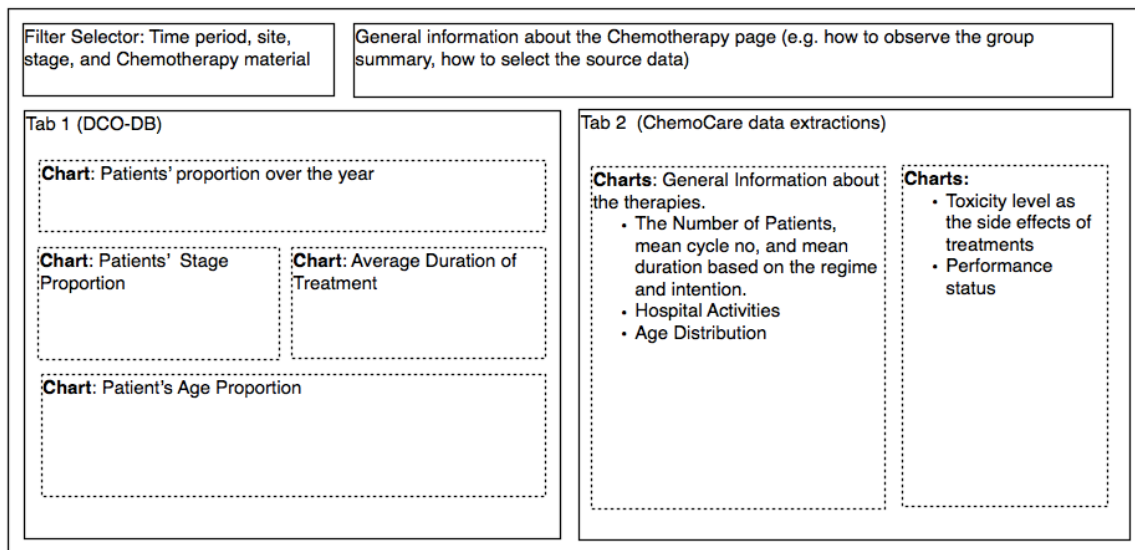


Fig. 3.7 *SESO Gateway* chemotherapy wireframe

Chemotherapy wireframe We include both Oncology DB and *ChemoCare* data extraction as our sources for the chemotherapy cohort summary. In comparison to the Radiotherapy page, the Chemotherapy page provides more comprehensive filters such as, time, site, stage, and chemotherapy materials. Each of the filters has a default value but the chemotherapy material. The users are required to select the material from the list of materials extracted from Oncology DB Chemotherapy material lookup table (i.e. *ChemoMaterialLUT*).

The Chemotherapy page consists of two tab components. The first tab shows the summary obtained from the Oncology DB while the second tab shows the cohort summary from the *ChemoCare* data extractions. The list of charts containing the overview of the chemotherapy, as shown in Figure 3.7.

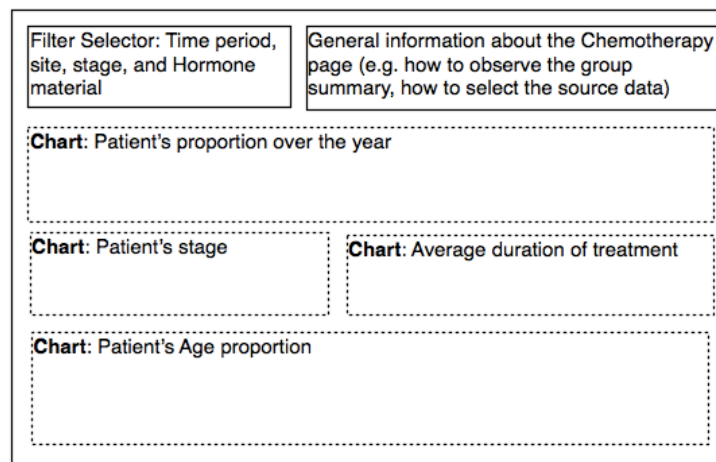


Fig. 3.8 *SESO Gateway* hormone therapy wireframe

Hormone therapy wireframe The hormone therapy page is similar to the Chemotherapy section page (i.e. it has two main components). It has several filters, such as the time-period, site, stage, and hormone materials. Other than hormone materials, every other filter has a default value. Once the data is fully loaded from the Oncology DB, the chart section will be updated accordingly.

Back-End

This section focuses on the design of the *SESO Gateway* backend. We developed *SESO Gateway* as a proxy application. It is a Representational state transfer (REST) API [102] service that has several endpoints. We use the REST API as the proxy between the user interface and the data sources to mimic the application developed within the NHS Lothian used for accessing the data from the Oncology DB. We develop the REST API services as a *Django* application.

We use *Django* for our web application because *Django* provides a toolkit of components that our web application needs. In this project, we want to focus more on designing a method to visualise the patients' pathway. Because *Django* encourages rapid development and pragmatic design, it accelerates the development of an application from its concept to completion. Another main advantage of using *Django* is its portability. *Python* runs on most platforms (e.g. Windows, MAC, Linux) and *Django* can run wherever *python* works. Therefore, all development and production environment can be easily supported [41].

As mentioned in the previous section about the lack of real databases access, we need to have a mock database for our data sources. We use *SQLite* as the database engine for our mock database. *SQLite* is a compact RDBMS used in programming and designed for

managing data. It uses the SQL (i.e. with some extension) language for the retrieval of data which allows us to update the *SESO Gateway* application in the future once the SQL Server Oncology DB is up and running.

3.5.2 Software Architecture

We created a *Django* application which architecture is shown in Figure 3.9 to solve the primary objective mentioned in the previous section.

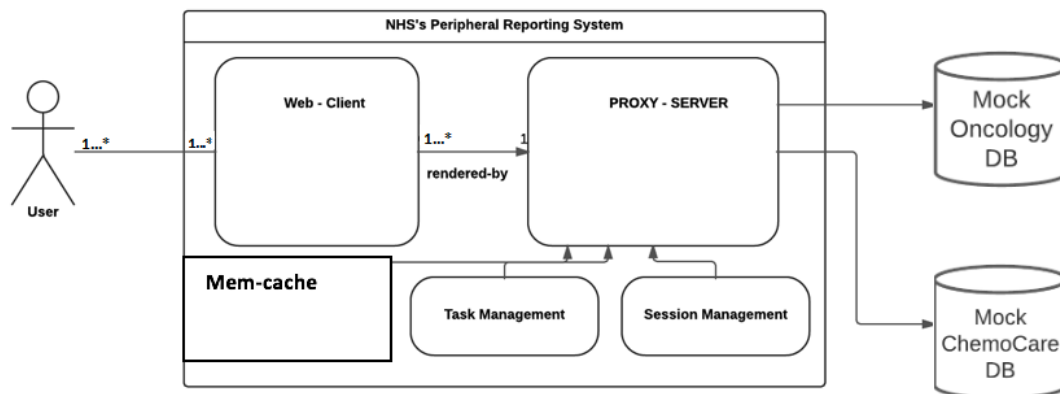


Fig. 3.9 *SESO Gateway* architecture

The *SESO Gateway* consists of three main components, the user interface (i.e. web-client), the software integration (i.e. the REST API services), and the databases (e.g. the Oncology DB and *ChemoCare* data extraction), as shown in Figure 3.9. This separation contributes to the modularity of the system, which allows us to change each module for the future changes that may happen in the NHS infrastructure.

Front-end

The web client has two main functionalities. First, it shows the event timeline for a certain patient during their treatments in the NHS Lothian. Second, the web client allows the user to observe patterns for various treatments by using cohort summaries. We implement the Model View Controller (MVC) pattern for the *SESO Gateway* web client. It allows the *SESO Gateway* separates the development of the Graphical User Interface (GUI) with the business logic or the data model. We separate each functionality in different modules. These modules are being integrated into a root module called *Apps*. In this module, we registered the built-in *AngularJS* (e.g. *ngRoute*, *ngResources*) modules as well as the custom modules

that we created (e.g. *event-timeline*, *group-detail*). It contains the main controller for the *SESO Gateway* client. With the main controller, we generate a SPA by only updating specific divisions of the client. We register several *urls* that can be accessed by the users in the Apps.

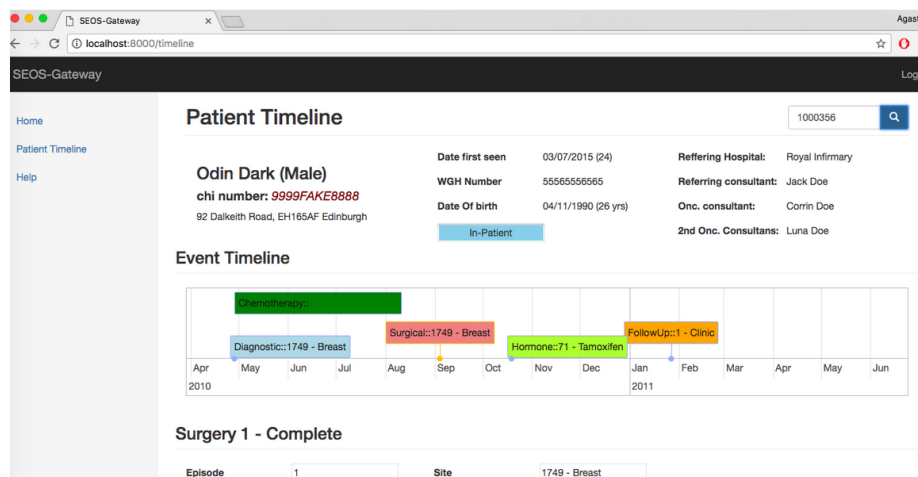


Fig. 3.10 *SESO Gateway* individual timeline with dummy data

Individual Timeline In this section, we discuss the architecture for our individual timeline components to create the page as shown in Figure 3.10. We separate the modules for the individual timeline and cohort/group summaries. With this separation, we have loosely coupled components. Therefore, when we need to change the behaviour of the individual timeline, we do not need to update any components in the group summaries. Figure 3.11 shows the whole functionality of the individual timeline.

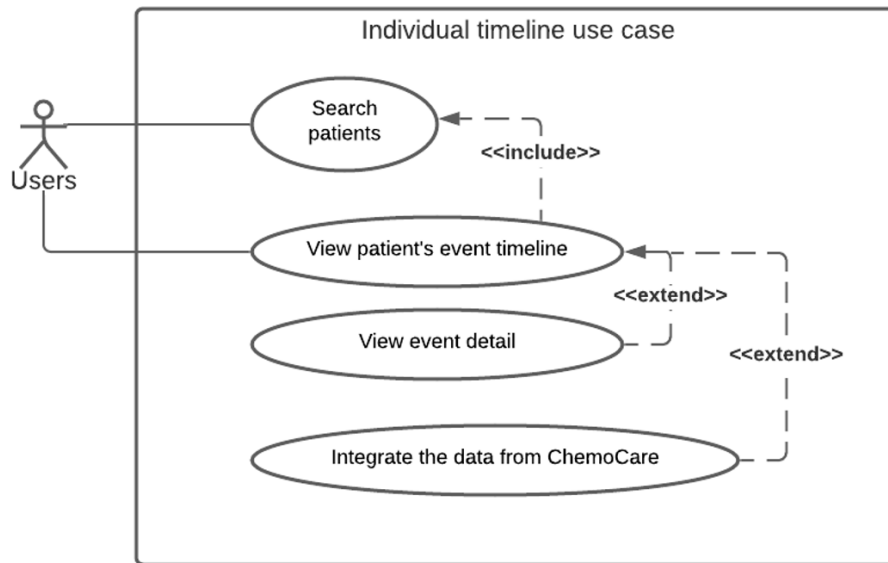


Fig. 3.11 *SESO Gateway* individual timeline use case

The individual timeline has several components (i.e. angularJS templates, angularJS modules, and several angularJS custom directives) for implementing the features shown in Figure 3.11.

Below is the complete list of components we have for the individual timeline:

- *AngularJS-templates*
 - *event-timeline.html*
 - *event-detail.html*
- *AngularJS* custom module: *event-timeline* module. This module has a scope object which control the items in the timeline and two main functions:
 - *getTimeline()* allows us to retrieve the information from REST API services about the events for a specific patient. It modifies the response to be compatible with the *vis-timeline* directive by creating a *vis.js* dataset and uses it for the timeline input.
 - *integrateChemocare()* integrates the patient information from the ChemoCare data extractions to the existing timeline.
- *AngularJS-directives*
 - *vis-timeline* directive creates the DOM for the timeline by using *vis.js* library. The *vis.js* is a dynamic, browser based visualisation library. The library is designed to

be easy to use, handle large amounts of dynamic data manipulation. This library consists of the several components, such as *DataSet* and *Timeline* [172]. The input object for this directive is a *vis.js* dataset object which is updated in the *event-timeline* module's controller.

Figure 3.12 shows the detail of interactions between *individual-timeline* components mentioned before.

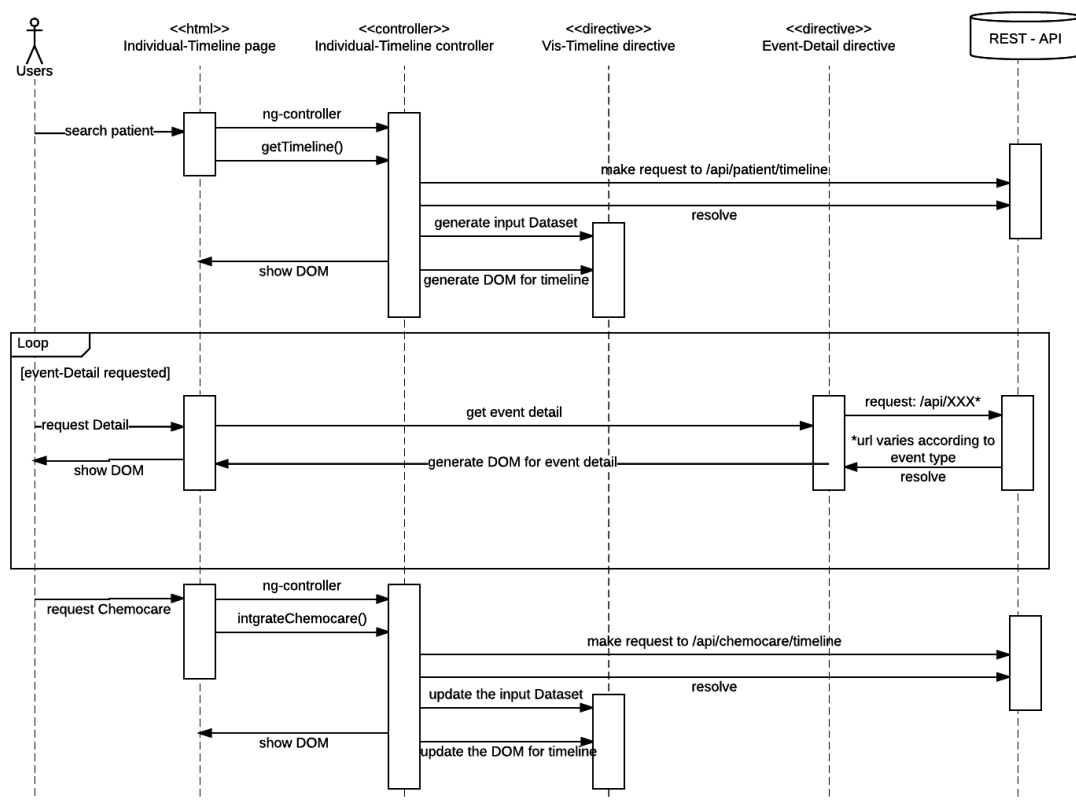


Fig. 3.12 *SESO Gateway* individual timeline sequence diagram

Cohort/Group Summary In this section, we discuss the architecture for the group/cohort summary. There are several modules based on the type of treatments (e.g. radiotherapy, hormone therapy, chemotherapy) registered in the *SESO Gateway* for the cohort summary. Those modules have a similar structure. For each treatment, we have a module which contains methods to allow the users to apply several filters and then fetch the data from REST API. Once the data is fetched, we use a third party library, *Angular-Nvd3*, for the data visualisation. Figure 3.13 shows the sequence diagram for general group summary module.

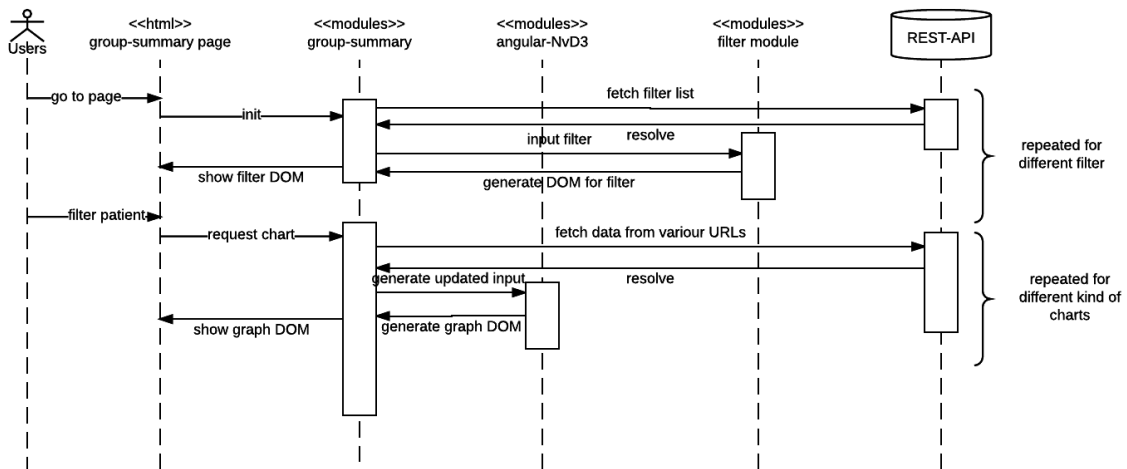
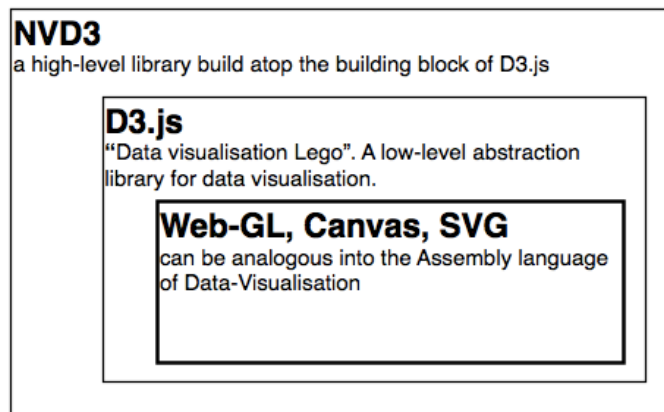


Fig. 3.13 *SESO Gateway* individual timeline sequence diagram

In general, each of the cohort module has several components as listed:

- *AngularJS-templates*
 - *module-detail.html*
- *AngularJS* custom module. The module contains the method (i.e. *filterPatient*) to fetch the data from the *SESO Gateway*'s RESI API. Here, we perform several HTTP requests to our API for each chart registered in the module. We perform separate asynchronous requests to accelerate the time we need for updating the charts. With the asynchronous requests, we make the browser more responsive by not blocking the client. Therefore, all the requests can be performed simultaneously. This module has several objects saved in the angular-scope as the filter (e.g. time-period, stage, site). The list of items mentioned in those filters is downloaded from the Oncology DB lookup tables.
- *AngularJS directives*: *Angular-nvd3* is an *AngularJS* directive for *NVD3*, a re-usable dynamic charting library (based on *D3.js* library). The *D3.js* built on the open web-standards. It is written in *JavaScript* and allows seamless binding of the data to *HTML*, *SVG*, *CSS*, etc. The building blocks of the *NVD3* library is shown in Figure 3.14.

Fig. 3.14 *NVD3* building blocks

The *SESO Gateway* contains several charts. We specify the type of charts in the *Angular-scope* object and update the responses acquired from *SESO Gateway*'s REST-API into appropriate *Angular-nvd3* [11] input formats.

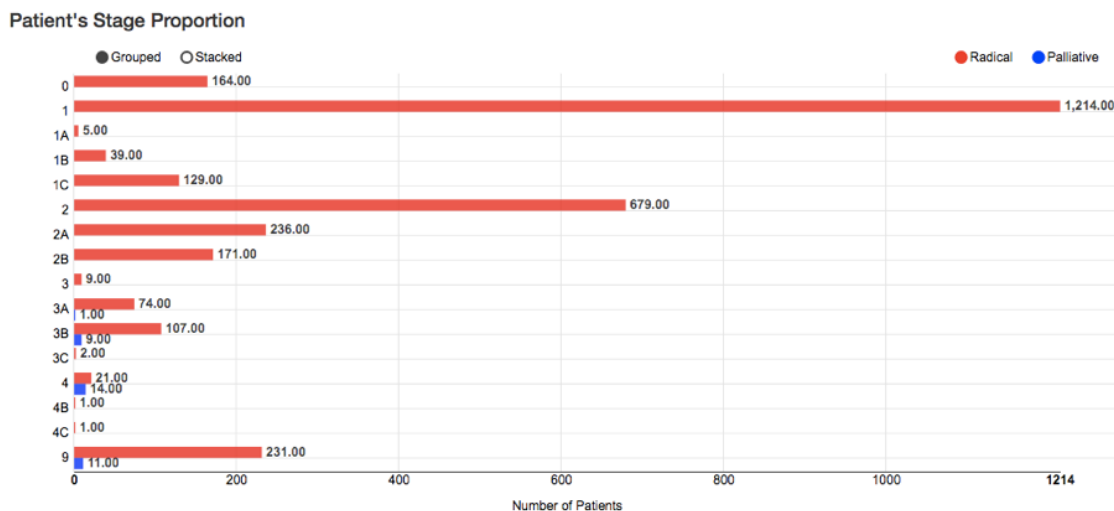
Fig. 3.15 *SESO Gateway* patient's stage proportion

Figure 3.15 shows one example of the charts generated by the library. The library also supports the other charts such as bar chart and pie chart as shown in Figure 3.16.

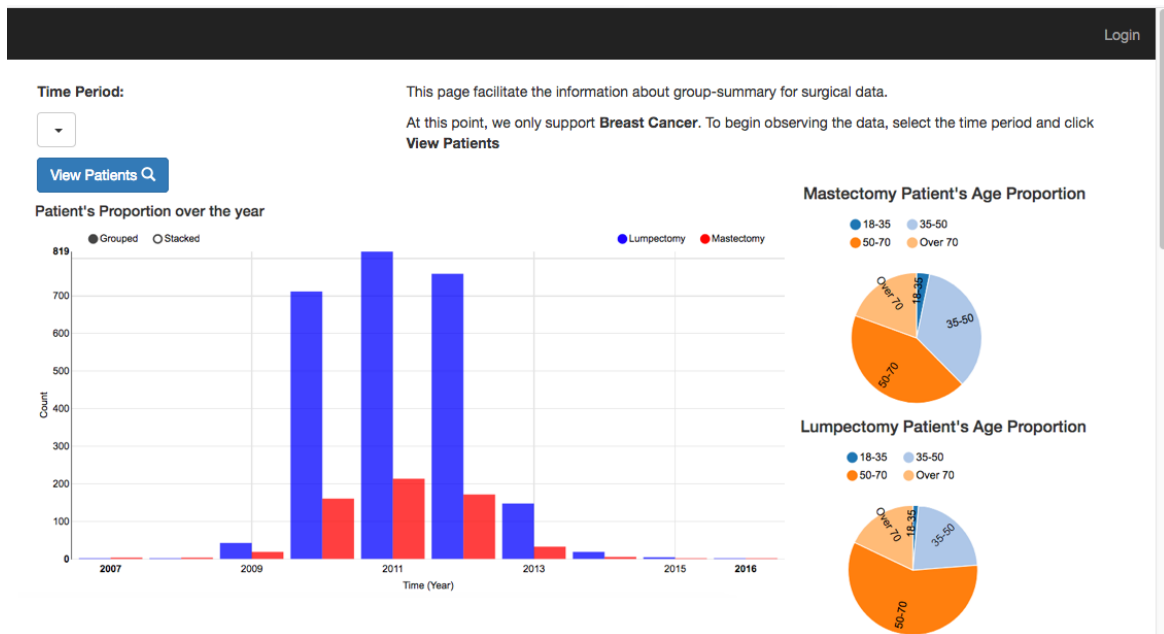


Fig. 3.16 *SESO Gateway* Surgical page

Back-End

The *SESO Gateway* backend is a *Django* application. It has several modules which were isolated based on the data sources (i.e. the Oncology DB and the *ChemoCare* data extraction). These two applications are *patient* and *chemocare*. The *patient* application handles the REST API request for the Oncology DB, while the *chemocare*, as the name suggested, handles the request for the *ChemoCare* data extraction. We use a third party library, the *Django REST framework* [43], to help us rapidly create a browsable Web API for the *SESO Gateway*.

A typical *Django* application consists of a script for various environment settings (i.e. *setting.py*) and several modules for its subapplications. The subapplications are registered in the *setting.py* together with the other third party libraries.

Each subapplication has a router, a model, and an *api* package. In the router, we have several functions to connect this application to appropriate databases. Before we set up the application, we define the databases in the *setting.py*. An example is shown below:

```
DATABASES = {
    "chemocare": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": os.path.join(BASE_DIR, "chemoCare.sqlite3"),
    },
    "patient": {
```

```

    "ENGINE": "django.db.backends.sqlite3",
    "NAME": os.path.join(BASE_DIR, "sampleData.sqlite3"),
},
}

```

The *Django* framework supports several database engines. If we change the database in the future, we need to update the *setting.py* accordingly.

The classes in the *model* module contain the essential fields and behaviours of the data in our mock databases. With the help of Django database access API, we generate the model for the *SESO Gateway*. Each class in the model represents a table in the database. In our model's class meta we set the *managed* property into *False* because we only allow read functionality for our DB.

The *api* package is the core of our Django RESTful API application. it includes several modules, such as, *query*, *view*, *urls*, and *utilities*.

We list the queries we used to get the correct result for both the individual-timeline and cohort result in the *query* module. In the next section, we explain one of the queries we use for the *SESO Gateway*.

The *view* contains the API-methods for responding the request from the client. Each method has different parameters associated with a specific SQL statement. One example is *chemocareAppointment* method. This method returns the patient's list of appointments/events based on the *ChemoCare* extraction. It takes the CHI number as a parameter.

In this method, we combine the parameter and query to get the result. Each query will have the bracket (i.e. { }) sign. The bracket will be replaced with an appropriate value taken from the parameters, as shown below:

- request: *http://localhost:8000/api/chemocare/timeline/appointment/?chi=xxxx*
- query:

```

select [index] as id,
       [Type.of.Therapy] as tot,
       [Intention] as intention,
       [Regime.Course.1] as regime_course,
       [Appointment.Date] as start
from ChemoData
where CHI = {}

```

- combined query:


```

select [index] as id,
       [Type.of.Therapy] as tot,
       [Intention] as intention,
       [Regime.Course.1] as regime_course,
       [Appointment.Date] as start
from ChemoData
where CHI = 0001

```

The other methods for our RESTful API service follow the same principal.

Oncology DB Query As previously mentioned, we have a dedicated module to keep the SQL-statement to fetch the data from the server. All the SQL statements/queries we use in the *SESO Gateway* only consist of the *select* statement to avoid any changes in the databases.

Some queries contain grouping, case, and more SQL-commands other than the simple *select-from*. An example of the queries is shown below.

```

surgicalQueries["count"] = """
Select
  Surgical.'index',
  case
    when suoper in ('381') then 'Lumpectomy'
    when suoper in ('382', '383', '384', '385') then 'Mastectomy'
  end as method,
  strftime('%Y', SUDATE) yr,
  count(*) count
from Surgical
where
  yr in {}
  and
  susite = '1749'
group by method, yr
"""

```

With the query shown above, we tally the proportion of breast cancer surgical procedures (i.e. mastectomy or lumpectomy [160]) based on the method for a particular time. Here we do not differentiate the stage of the treatment. The other query handles the separation for us.

3.6 Analysis

In this section, we analyse the quality of the *SESO Gateway*. We assess the quality of our software based on the successful delivery of the requirement through various methods.

3.6.1 Software Testing

One way to assess the application is working as expected is to perform a series of software testing. The tests are group by the level and purpose of the tests themselves. For *SESO Gateway*, we perform three groups of tests (i.e. the unit, integration, and performance testing)

Unit Testing

Our unit tests are performed on the developed module to clarify if the method performs as expected in a set of condition. The *SESO Gateway* unit tests are grouped in a module called *test*. Here, we want to check the correctness of the model generated by *Django* and the query we use to get the data. In addition to that, we perform the unit test for the utility methods.

Utilities method test The *utilities* module has several functions used in the *patient* and *ChemoCare* module. We use these functions to get the parameters from the request, define the HTTP responses, and convert the query result into a Python dictionary. To test this module, we create a *DummyRequest* class. We mock the *dummyRequest* object and verify the method against that mock object, as shown below:

```
def test_get_valid_year(self):
    request = DummyRequest()
    request.query_params["years"] = '2016,2017'
    years = getValidYear(request)
    self.assertEqual(years, '("2016","2017")')
```

Because the method only consumes a particular attribute of the object, we do not need to mock the other attributes of the request object itself. Shown in the example, we only need the query parameters for the *getValidYear* method. Python's *Duck typing*, "If it walks like a duck and it quacks like a duck, then it must be a duck." [132], allows us to create a simple mock request object.

For the converter method, we test this method with the queries we use for getting the data from the databases. In the next section, we discuss the query tests.

Model tests Before we perform the query tests, we run the model tests. These tests are used to check whether Django has correctly generated the database model for us. In this test, we first create a mock database, populate the mock database with mock data, retrieve the data using Django model and assert it. We repeat this process for all tables in the *SESO Gateway*.

Query tests The query tests require a mock database similar to the model tests. In the query tests, we use the *executeQueryReturnDict* method. This method is the converter we mentioned in the previous section. For every query, we perform several sequence tasks.

The sequence is as follow:

- We create a mock database with mock data in the *setUpClass* method.

```
...
@classmethod
def setUpClass(cls):
    datetime(1998, 4, 15, 0, 0, 0)
    # Create the Object for Chemodata
    # REGIME A
    Chemodata.objects.create(
        index=1,
        chi="001",
        intention="Palliative",
        regime_course_1="A",
        type_of_therapy="CHEMO",
        appointment_date=datetime(2017, 1, 1, 0, 0, 0)
    )
...
```

- We get the query from the *query* module and format it accordingly. Next, we create a connection to the mock database.

```
...
def test_mean_cycle_no(self):
    query = chemoInfoQueries["mean_cycle_no"]
        .format("'Palliative'", "('2017', '2016')", "'CHEMO'")
    cursor = connections['chemocare'].cursor()
...
```

- Lastly, we use the converter method to change the result into dictionary object, and then we perform an assertion.

```

...
queryResult = executeQueryReturnDict(cursor, query)
self.assertIsNotNone(queryResult)
self.assertEqual(queryResult, [
    {'no_of_patients': 1, 'regime_course': 'B', 'mean_avg_cycle': 2.0},
    {'no_of_patients': 1, 'regime_course': 'A', 'mean_avg_cycle': 3.0}
])
...

```

We repeat the same procedure for the queries we use in the *SESO Gateway*. Within this tests, we perform the test for the Model generated.

Integration Testing

The *SESO Gateway* integration testing combines several modules within the application and tests those modules as a group. The integration testing takes the modules that have previously tested in the unit testing. The integration testing module is part of the *tests* module. Our integration testing focus on testing the pathway from the first time the client makes a request, retrieve the correct data from the databases, and return the correct response to that request.

To perform the integration test, we perform several tasks:

- Prepare the mock database for the test. This step is similar to the action we perform in the unit testing.
- Create a request object. Unlike the request we use for our unit testing, the request here is a full request object with all the request attributes like headers or body.
- Test the request object to the method corresponding to the request URL and then assert the response returned.

The tasks are repeated for the other kind of requests.

Performance Testing

In the *SESO Gateway* requirements, we have around 100 users. Hence, we assume that they will be a time when all of the users access the application simultaneously. Based on the assumption, we need to perform a performance test to observe the robustness of our system.

One of the issue of *go-live* is the performance (e.g. latency, bottleneck). The performance test can simulate the situation where many users try to access our application simultaneously. We use *Gatling-tool* [56] to perform the performance test.

Gatling is a highly capable load testing tool. It has the support for HTTP-Protocol. The *Gatling-tool* is asynchronous as long as its underlying protocol can be implemented in a non-blocking way. The *Gatling-tool* allows us to do the load-testing of the *SESO Gateway*'s HTTP Server by creating hundred of virtual users.

To perform the performance test, we have several steps as shown below. All of the performance tests are written in *Scala* [147].

- Build and run the server. We create a separate virtual environment to perform the performance tests without polluting the local environment.
- Create the requests for the test and then set the number of users. We create 100 virtual users for the individual event tests as shown below.

```
trait SESOSimulation extends Simulation {
  val SESO_ENDPOINT = System.getProperty("seosEndpoint")
  def defaultHttpConfig = http
    .baseUrl(SESO_ENDPOINT)
    .acceptHeader("text/html,application/json")
    .acceptLanguageHeader("en-US,en;q=0.5")
    .acceptEncodingHeader("gzip, deflate")
    .warmUp(SESO_ENDPOINT)
}

class IndividualTimeline extends SESOSimulation {
  val users = scenario("Request timeline for a certain patient")
    .exec(http("retrieve the Individual Events")
      .get("api/patient/individual-event/?dco=XXXXXX")
      .check(status.is(200)))
    .pause(8)
    .exec(http("retrieve the Individual Events from ChemoCare")
      .get("api/chemocare/timeline/appointment/?chi=XXXXXX")
      .check(status.is(200)))
    .pause(8)
    .exec(http("retrieve the CHI number from DCO-NO")
      .get("api/patient/get-chi/?dco=XXXXXX"))
}
```

```

        .check(status.is(200)))

setUp(
    users.inject(
        atOnceUsers(100)
    )
).protocols(defaultHttpConfig)
}

```

- Run the performance test and observe the result.

Test Result

At the beginning, we run both unit test and integration test before we run the server for performing the load-testing (i.e. performance-test). Figure 3.17 shows the coverage of our unit and integration testing. We manage to get a high-coverage (around 90% coverage) to test the correctness of our modules.

Patient Coverage:

Element	Statistics, %
__init__.py	
chemo.py	89% lines covered
filterSelector.py	100% lines covered
hormone.py	89% lines covered
query.py	100% lines covered
radio.py	89% lines covered
surgical.py	93% lines covered

Chemocare Coverage:

Element	Statistics, %
__init__.py	
chemoAdministrationInfo.py	85% lines covered
chemoInfo.py	84% lines covered
toxicityInfo.py	85% lines covered

Fig. 3.17 *SESO Gateway* unit and integration tests coverage

The *SESO Gateway* can handle 150 simultaneous requests from the user. However, this is only applicable to the mock server. Our mock server consists of 10,000 user data. We do not have the resource for testing the *SESO Gateway* against millions of patient data. The result will be highly dependable from the server we access and the place where we run our application.

3.6.2 Case Study: Feedback and commentary

In this section, we present a case study which we conducted during the development phase of our software. Most of the staff involved in the demo were oncologists and manager from the WGH. During the development process, we conducted several presentations and demos for the *SESO Gateway* and several volunteers from the ECC tested the application. This process allowed us to evolve the *SESO Gateway* application to be more usable for the users (i.e. the oncologists and ECC staffs).

Based on their feedback, we expanded the *SESO Gateway* accordingly. During the early development of the *SESO Gateway*, this application only focused on the individual timeline. At its first stage, the *SESO Gateway's* timeline was not as user-friendly. We updated the timeline by adding group functionality, better colour coded after a demo with an oncologist. The user asked the developer to change the colour of the individual timeline, and suggest adding groups for different kind of events (e.g. Diagnosis, Surgical) to improve the clarity of the information as displayed in Figure 3.18.

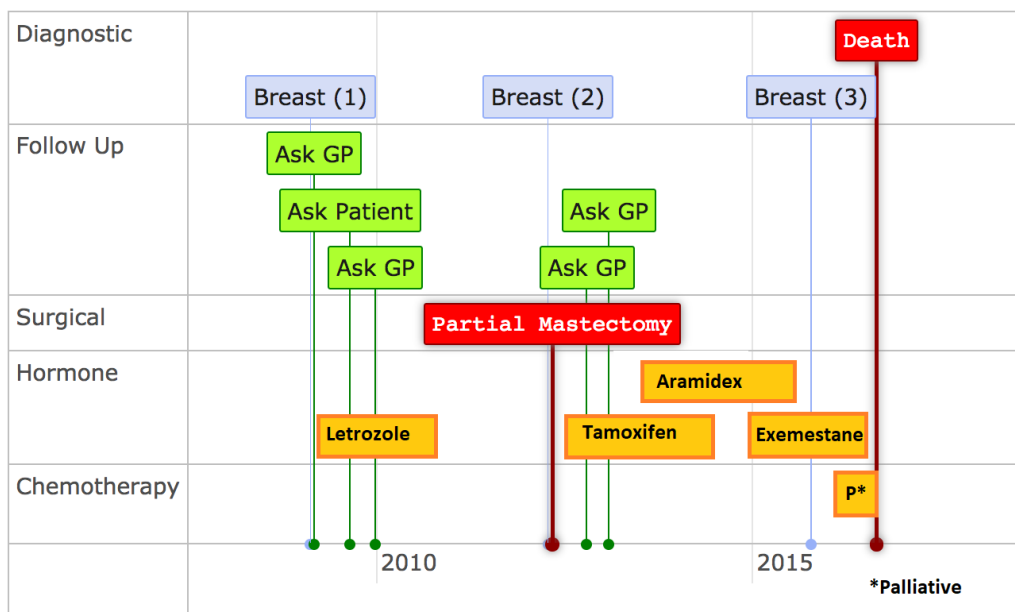


Fig. 3.18 *SESO Gateway* event timeline

After developing the timeline, we expanded our application to provide information on patient cohorts. Overall, the feedback was very positive. We also got further feedback about the features that can be implemented in future work.

For instance, having more filters for the cohort functions based on the patient's oncologist will be very beneficial for the oncologists. In addition, the ability to implement the

functionality for generating the list of patients used in the cohort graph can ease the auditing process in ECC.

From our user feedback, we are aware of more functionality for *SESO Gateway* in the future. Hence, it is a good idea to conduct a closed beta trial of the software before we developed the full application. It allows us to analyse the requirements during the development progress, which minimise the need for changing the architecture for the *SESO Gateway*.

3.7 Summary

In this section, we discuss the solutions we provide for the requirements as well as problems arisen during the development of the *SESO Gateway*. NHS Lothian has huge amount of patients data stored in several different formats. Our developed proxy application allows easy access without the need to migrate all the data or change the data format.

3.7.1 Delivery of the key requirements

In the previous section, we present both functional and non-functional requirements. These requirements contain various staff's needs within the ECC, NHS Lothian. In this section, we discuss the solution and issues related to the requirements.

Functional Requirements

The functional requirements for our project derive from our primary objectives: providing the tool for patient data visualisation both for the individual event as well as the cohort summary.

- **The user should be able to view the individual timeline.** To satisfy this requirement, the *SESO Gateway* serves a page dedicated to individual patient's event-timeline for the user. The option *Timeline* in the dashboard's navigation bar redirect the user to the individual-timeline page. This requirement is the first feature implemented in the *SESO Gateway* project. During the development of the *SESO Gateway*, we perform several updates to the presentation of our timeline.
- **A user should be able to get detail information about each event in the timeline.** Some issues in implementing this requirement are related to the relevant data shown in the event-detail component and the need to modify the library we use for creating the timeline (i.e. *vis.js*). After the timeline is attached to the html-body section, we need

to re-compile each event so that it will trigger a *GET* request once it is clicked by the user. The event detail is shown under the event-timeline component.

- **A user should be able to view the cohort summary for various kinds of treatments.** The *SESO Gateway* has several pages dedicated to cohort-summary visualisation. This requirement is the hardest requirement to implement. We update the cohort summary component every time we get a new data extraction from the NHS Lothian. The problems of providing the cohort summary arise for several aspects:
 - What cohort information is relevant.
 - Which filters need to be implemented.

During development, we followed an iterative approach to determine the chart we want to show in the *SESO Gateway*. We first developed various charts based on the data given to us. We then conducted a demo to expected end-users. Based on their feedback, we updated both the charts and filters. Our solution considered the cohort summaries for surgical, hormone-therapy, radiotherapy, chemotherapy treatments in separate pages.

- **A user should be able to search for certain groups of patients.** This requirement is highly related to the previous requirement about cohort summary visualisation. We implement the search functionality by adding a filter in the cohort-summary page.

Non-functional requirements

In this section, we discuss the problems and solutions related to the non-functional requirements previously mentioned.

- **The system should be compatible to be deployed in the NHS Lothian Network.** This requirement is quite difficult to be fulfilled. At the start of the project, we are not fully aware of the structure of the NHS Network. However, we need to develop an application that can be independent of such consideration. One of our solutions is to create an application with different sub-applications. With the structure of our project, we can take any component and change it accordingly without the need to modify the other components. For example, we can change the REST API service modules without changing the web-client. With this approach, we can integrate the *SESO Gateway* easily and deployed it in the NHS Lothian's network in the future.
- **The system should be able to be used by 100 users simultaneously.** To test whether our application can handle many requests at the same time, we performed performance

testing. This testing allows us to see how many connections died and how long it takes to serve the requested page. Based on performance testing, 90% of the requests are successfully serviced. The only concern with this is the number of data used for testing our application. We only verified the application for 10,000 patients records in our database. The result may be significantly different if we are required to connect to various data sources. Once we connect *SESO Gateway* to real databases with million of patient's records we need to update the system architecture for scalability in the future. However, this is not part of our current project scope.

3.7.2 Scalability

Although the scope of the project does not include scalability, in this section we discuss one of the possible ways to scale the *SESO Gateway*. Scaling can be done in two different ways: vertical scaling (i.e. scale-up) and horizontal scaling (i.e. scale-out) [21]. Vertical scaling includes the process of adding more power to the *SESO Gateway* server. However, vertical scaling comes with some limitations. Vertical scaling has a limit because it is impractical to add unlimited CPU and memory to a single server. Furthermore, vertical scaling does not have a failover: if the server goes down, the system goes down with it completely. If we need to scale the *SESO Gateway*, horizontal scaling is more desirable (e.g. by adding load balancer, server duplications) because of the limit of vertical scaling.

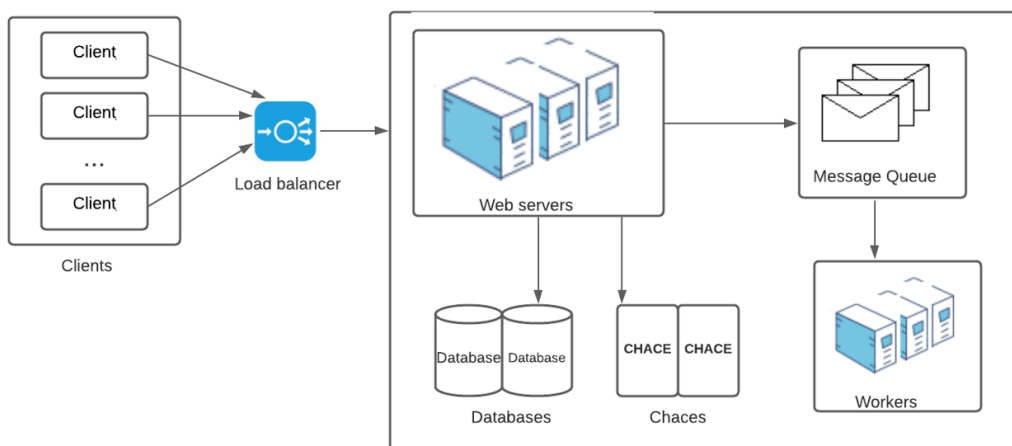


Fig. 3.19 *SESO Gateway* updated system architecture

Figure 3.19 shows one example of *SESO Gateway*'s architecture with horizontal scaling. Several notable components of the system are listed below.

- Load balancer, the load balancer will evenly distribute the traffic from the clients.

- Cache, a cache is a temporary storage that allows access of frequently as data more efficient.
- Message queues and the designated workers allow asynchronous communication between clients and the producer to solve the bottleneck issue of performing several tasks from many users at the same time.

3.7.3 Contribution

At the moment this thesis is written, the NHS Lothian is still in the process of appraising the reporting service for the new MSSQL Oncology DB. *SESO Gateway* is considered as a potential in-house reporting service application for the migrated oncology DB.

Chapter 4

Migration

4.1 Introduction

In our first project, presented in chapter 3, we familiarised ourselves with the cancer data in the Edinburgh Cancer Centre (ECC) by developing a data visualisation tool referred to as *SESO Gateway*. After we developed the *SESO Gateway*, we were tasked to migrate the database that we used on the first project to a new more scalable database. In this section, we describes the extraction, transformation, loading (ETL) process of data migration.

4.2 Legacy Database

As mentioned previously, The Oncology database was established in 1974. Bill Shearer wrote it with design guidance from Bill Duncan and Gill Kerr (members of NHS Lothian team at that time). The database was written in Common Business Oriented Language (COBOL). In 1979, a significant rewrite of the software took place to remove severe bugs. The process resulted in the loss of a small amount of data. In 1992, NHS employed a consultant to rewrite the database using Statistical Analysis System (SAS) [39]. However, the restricted SAS licence eventually rendered this impossible. As there were still several unsatisfactory aspects of the system combined with the increasing workload, it became a necessity to work with multiple resources [84]. Attempts to set up a multi-user input process using the legacy system failed, therefore NHS Lothian rewrote the system using Microsoft Access (MS Access) [114].

4.3 Motivation

Although MS Access facilitates the access from multiple systems, further issues emerged as the user base grew. As more and more users have write and read access to the database, the Access database (DB) becomes less secure. Furthermore, the database did not scale well to the use of many simultaneous users. When a user access the database in a shared folder, a lock file is created which prevent other users from accessing the database at the same time. Hence, a new robust and more scalable database is needed for recording the cancer patient data within National Health Service (NHS) Lothian.

4.4 Database Migration

Data migration is the process of transferring data from one system to another. A typical data migration includes the step of extraction, transformation, and loading (*ETL*) [13]. The process requires converting data into a new format that can be output from the old database and input into the new database. It is a multi phase process, which usually includes assessment, database schema conversion, data migration, functional, and validation testing. Unlike the traditional sequence, we perform the extraction, loading, and then the transformation because of the decision by the ECC to copy the whole legacy database without changing any field structure into the new data warehouse. The result of the migration includes both the new tables containing the patient information and the old legacy oncology database tables.

4.4.1 Extraction of Data

The extraction process involves the extraction of the data from the sources [13]. The source can be in the format of flat files, other source systems like the Relational Database Management System (RDBMS) or NoSQL. During the extraction, the desired data is identified and extracted from the source. For the Oncology DB, the source is the latest *.mdb* file (i.e. *SESOncData.mdb*). The *.mdb* file is a database file used by Microsoft Access. With *pyodbc* and *pandas* libraries, we use a Python script to perform the data extraction. Here, the script reads the *.mdb* file and converts it into both SQLite file and *.sql* format. The extraction process is straightforward as we copy all the data without making any field selection. The list of tables we need to extract are:

- *Maindata*; contains the diagnosis and demographics of the patients.
- *Ablation*; contains the ablation treatment of the patients.

- *Chemotherapy*; contains the chemotherapy treatments of the patients.
- *Follow-Up*; contains the follow up treatments or appointment of the patients.
- *Hormone*; contains the hormone therapy treatments given to the patients.
- *Implants*; contains the implants therapies given to the patients.
- *Neutrons*; contains the neutrons therapies given to the patients.
- *Intracav*; contains the data for *intracavity* radiation therapy.
- *Radiotherapy*; contains the data regarding patients' radiotherapy.
- *Surgical*; contains the data regarding patients surgeries.
- *Unsealed*; contains the data regarding the unsealed therapy.
- *Morbidity*; contains the outcome of treatments regarding the patients' morbidity.
- *NewMets*; contains the outcome of treatments regarding the new metastases.
- *NewNode*; contains the outcome of treatments regarding the cancer nodes.
- *NewPrimary*; contains the outcome of treatments regarding the patients' primary cancer.

The data extraction step generates the flat files for the loading step.

4.4.2 Data Transformation

Data transformation is the process of converting data from one format or structure into the target format. We use Transact-SQL (TSQL) [168] scripts to transform the legacy data into the appropriate table.

All the new table transformations and new columns for each table in the new database are determined by the NHS party. Here, we only perform the first normal form (1NF) transformation (i.e. each attribute of a table must only have a single value) for specific tables requested by the NHS. Normalisation is performed to avoid duplication for the *Demographics* table.

We get the data for the *Demographics* table from the *Maindata* table in the legacy database. This *Maindata* contains information regarding the patient's demographics and diagnosis. As requested by the NHS Lothian, each patient can only have one data in the *Demographics*

tables. By aggregating the *Maindata* table based on several attributes such as CHI and taking the other information to put into the *Demographics* table from the latest diagnosis data in the *Maindata* table.

The data transformation aims to improve the quality of the data. As previously mentioned in the legacy database, the *Maindata* table is the source of both patients' demographics and diagnoses. Most of the patient's demographics (e.g. *name*, *date of birth*, *ethnicity*) remain the same throughout the cancer care. We separate the *Maindata* table into *Diagnoses* and *Demographics* tables to avoid repeated information regarding the patients' demographics. We also perform other transformation and describe a list of notable transformation in the following.

***DemographicId* as patient identifiers**

We generated *DemographicId* to internally identify the patients instead of using the Community Health Index (CHI) fields like in the legacy database. The CHI is a unique 10-character numeric identifier, allocated to each patient on first time the patients uses an NHS services and are registered with the system. Even though CHI is the standard practice for patient identification in Scotland, the current formula for CHI is not reliable enough to avoid patients sharing the same CHI. The current CHI number consists of the six-digit Date of Birth (DDMMYY) followed by a three-digit sequence number and a check digit. The ninth digit is always even for females and odd for males [156]. Hence, CHI only relies on three-digits to make the it unique for every patient as the patients can share the same gender and date of birth [19].

Date of death discrepancies

In the *Maindata* table, we have two fields which contain the information about the date of death (i.e. *DATDEATH* and *DOD*) for each patient. However, there are patients with different *DATDEATH* and *DOD* in their diagnosis data. For these patients, NHS board agreed to migrate the *DATDEATH* instead of *DOD*. We use *DOD* if the *DATDEATH* value is *null*.

The latest information of the patient demographics selection

When we separate the *Maindata* table into both *Demographics* and *Diagnosis* tables, we only migrate the latest demographics information of the patients. In the *Demographics* table, one patient can only have one row of data. To identify the latest demographics, we use the *PRIMARY* field instead of the *DiagnosisDate* or first seen date (*FSDate*) because it is possible to have a null value in those fields. Previously, each time a patient undergoes treatment, the

record is saved under a different *PRIMARY*. The *PRIMARY* is then incremented by one for the next diagnosis data. Hence, the *PRIMARY* field can be used to identify the latest event of the patients in the *Maindata* table.

Fields Conversion

Several fields in various tables (e.g. *Maindata*, *Surgical*, *Radiotherapy*) such as *Site*, *Histology*, *Grade*, and *PerfStat* need to be converted. During the migration, we were provided the mapping table and the list of fields that need to be converted.

Data Separation

The *Maindata* table contains various information such as patients' diagnosis, follow-up summaries, and previous primaries (i.e. primary cancer before the patients are diagnosed). For the new Oncology database, we separate this information into several tables (i.e. *Diagnoses*, *FollowUpSummaries*, *PreviousPrimaries*). Once separated, we use the *DemographicId* as the foreign key in the *Diagnoses* table to integrate it with the *Demographics* table. Many of the tables in the legacy database (e.g. *Surgical*, *Radiotherapy*) or the new tables (e.g. *PreviousPrimaries*) ties to diagnosis data instead of only to the patients' demographic. For these tables, we use both *DemographicId* and *DiagnosisId* as the foreign key. We use *DiagnosisId* as the index in the *Diagnoses* table.

Previous Primaries Duplication

The *previous primary*, if applicable, gives the information regarding the *primary* cancer found before the diagnosis of the patients. Each patient may have up to four previous primaries and hence also have several diagnoses. During the User Acceptance Test (UAT), we found duplication for the previous primaries. To address this, we grouped the data based on several fields in the *Maindata* table such as the *PreviousPrimaryDate*, *Site*, *TNM_T*, *TNM_N*, *TNM_M*, and *Treatment* to avoid duplication and only migrate data which contains unique combination of the fields mentioned. We note that cancer staging, indicated by *tumour node and metastasis* (TNM), is a method used to specify the stage of cancer. TNM refers to the extent of a patient's cancer, such as the size of cancer cells, the spreading of cancer and the development of secondary malignant cancer at a distance from a primary site of cancer. The above fields in the database give us some information regarding the state of cancer, for example: *PreviousPrimaryDate* indicates the date when the previous primary cancer was diagnosed; *Site* indicates the location of the primary cancer; *TNM_T* indicates the tumour;

TNM_N indicates the node; *TNM_M* indicates metastasis, and *Treatment* indicates the name of chosen treatment for the cancer (e.g., hormone therapy, chemotherapy, etc).

Episode duplication

Some tables in the legacy databases, such as *Surgical* and *Radiotherapy*, use the *Episode* field as the event identification for the patients. However, in the legacy database, this field is not populated correctly, and many of the patients have duplicate episodes. During the transformation, we update all episodes based on the date (e.g. Surgical Date, *SUDATE*). We order the data based on the date and assign a new value to the *Episode* field by using the following syntax:

```
row_number() over(  
  partition by Diagnoses.DiagnosisID  
  order by surgicalLegacy.SUDATE  
) Episode
```

The data is partitioned by the *DiagnosisId* because the data in the *Surgical* table is tied to the diagnosis. We perform this conversion for the other tables as well to guarantee consistency across all tables.

4.4.3 Data Loading

Data loading is the process of copying the dataset from source files into a database. We created a Java application for the loading step. We use 50 threads for loading all the legacy data into the target DB. Here, we use the *ExecutorService* framework provided by the Java Development Kit (JDK). The *ExecutorService* simplifies the execution of tasks in the asynchronous mode because it automatically provides a pool of threads and API our loading tasks. Each thread loads around 75000 rows of data. The input of our Java application is the flat files extracted from the data extraction step. As previously mentioned, the flat files are *.sql* files. The *.sql* files contain the table creation and insertion queries.

4.4.4 Execution

We create *Gradle* tasks to perform both transformation and loading. A *Gradle* task is a procedure which a build performs. Tasks range from compiling some classes, generating *Javadoc*, publishing some achieves to repositories, or any custom procedures such as transformation and loading [64]. Below is the list of the tasks for the migration.

- *runTableCreation*. The *runTableCreation* task creates the table in the target databases. It will drop the existing table if tables with the same name exist in the database.
- *runLoading*. The *runLoading* task loads all the legacy data into the legacy tables in the target database.
- *runTransformation*. The *runTransformation* task performs the transformation of the legacy tables into the target tables.

By using *Gradle*, we can streamline the migration process.

4.4.5 Data Validation

After finishing the transformation phase of the migration, we run the validation scripts (i.e. written in T-SQL) to validate the migration process. Each validation scripts has a set of tests. We test several aspects/properties of the newly transformed data against the legacy tables in the MSSQL DB as shown below:

- Diagnosis count is correct
- Unique demographic count is correct
- Diagnosis links to the correct demographic data
- Sub-tables (e.g. Surgeries, SACTs, FollowUps, HormoneAblation, HormoneAdministered) link to the correct diagnosis data
- All sub-tables have valid value for the new dropdowns
- All expected data fields are populated in the new system

Once we complete the validation step, we manually check for a random selection of patients (i.e. around 10 to 20 patients) to see if those patients data has been correctly migrated.

4.5 Issues

Several issues here emerged during the development process associated with the migration. We describe the issues we encountered below. Even though most issues were resolved, it highlighted the problem in the legacy system, such as indexing and data type.

4.5.1 Driver installation

Only a 32-bit Microsoft Access driver was installed in the machine used for the migration process while all other applications are 64-bit programs. One significant difference between 32-bit and 64-bit architecture is the width of a pointer, or how much memory a process can address. It is impossible to pass an address that is 64 bits wide when the driver assumes it can only be 32 bits wide. Hence, the driver and the applications were not compatible. Due to the lack of administrative privilege, the developer could not install the correct driver (i.e. 64-bit Microsoft Access driver). To solve the issue, we re-installed Python 32-bit instead of 64-bit to access the legacy database and then created a Python script and Java application instead of using the SQL Server Migration Assistant (SSMA) for Access directly [112].

4.5.2 Invalid UHPI and CHI

During the User Acceptance Test (UAT) migration, we found several patients (i.e. 27 patients) with invalid Unique Hospital Patient Number (UHPI) and CHI. For these patients, their UHPI or CHI digit exceeds the maximum length (i.e. ten digits) of the CHI/UHPI format. For these patients, the NHS team decided to manually update their patients CHI and UHPI. We note the impracticality of the solution.

4.5.3 Invalid date

There are several data in the legacy DB with invalid dates on several date fields. Typos are the most common causes for the invalid date (e.g. a patient was born in the year of 105 AD). After locating all instances with invalid date, the NHS team manually fixed that by checking the paper records accordingly.

4.5.4 Invalid drop-down value

After migration, we also found several fields with unknown values. In other words, The values of the fields are not recorded in the look-up table. The unknown value causes an error in the database front-end. Some of the fields are listed as follow:

- **Site.** There are several fields across different tables with the issue: *MainData.SITE*, *MainData.SITEMET1*, *MainData.SITEMET2*, *MainData.SITEMET3*, *NewMets.METSITE*, *NewPrimary.NPSITE*, *Radiotherapy.EXTSITE*, *Surgical.SUSITE*, *Ablation.ORGAN*, *Surgical.SUSITE*. The incomplete mapping table for the legacy site to the International Classification of Diseases for Oncology Topography (ICD03T) and leading '0' or '00'

before each site in the legacy DB (e.g. 0040) cause the *Site* field to be populated with an unknown value (UNK). We remove the leading 00 for the site and update the mapping table to transform this field correctly.

- **Grade.** During the migration user acceptance test (UAT), we observe that the *grade* field is incorrectly transformed because of using the wrong mapping table. There is an unknown value (i.e. 9) found in the *Diagnosis* table. We resolved this issue by updating the mapping table.
- **Cause of death.** There are some unknown values (i.e. which are not found in the mapping table from the legacy look-up table into the International Statistical Classification of Diseases and Related Health Problems 10 (ICD-10) format for the cause of death of the patients.

4.6 Reporting Service

After the database was successfully migrated, we developed a system for generating a comprehensive report for the dataset known as reporting service. The system aims to help the healthcare professional or auditing team to make an effective business decision using the data from the new Oncology database. There are multiple reporting service systems, such as *PowerBI* [129], *Tableau* [164], *R Shiny* [133]) available in the market which we evaluated first. With the evaluation, we informed the NHS board about all the available options to help them decide a suitable system for the new database. We evaluated the system based on the cost, setup, accessibility, maintenance, usability, and the lifetime of the product. The evaluation result is as follow.

4.6.1 In-House Visualisation Tool Integration

Overview

It is possible to develop our own custom in-house reporting pages by embedding an application to the Oncology database client that was developed by NHS team and was written in *ASP.NET* [14]. The application would be developed using JavaScript libraries based on the visualisation tool developed in the previous project, *SESO Gateway* (i.e. described in Chapter 3). This option would require the involvement of future developers to continue to build and support these reports as web pages that are integrated into the Oncology database.

Cost

Free, except for the developer time. The hosting cost is part of hosting the oncology database.

Setup

The report can be set up as part of the oncology database web pages and require no further hardware or software setup.

Accessibility

The application will be available on the internet within the NHS Lothian network through a reporting area on the Oncology Database using a web browser such as Internet Explorer or Firefox.

Maintenance

A developer would be required to make changes or build new reports as required in the future.

Usability

The reports should have a straightforward interface. The usability can be enhanced by choosing to go with the technology which is familiar to the NHS staffs.

The lifetime of product

It depends on the lifetime of the database web page technology for the client and any JavaScript libraries for generating the reports. This would have to be considered as part of general system maintenance.

4.6.2 SQL Server Reporting Service (SSRS)

Overview

SSRS is a Microsoft reporting tool that comes with SQL Server [159]. As we use SQL Server for the Oncology database, we can have this functionality for free with the database. Furthermore, the NHS *eHealth* [46] widely uses SSRS for their reporting and it would be straightforward for the eHealth team to set this up and running with it. We can get all the necessary reporting that we need by using SSRS.

Cost

- Software: free.
- Hardware and eHealth setup costs: there may be some minor costs involved in getting eHealth to set up SSRS as it is not installed by default on the SQL Server.

Setup

eHealth needs to set up a report server so that reports can be uploaded and viewed.

Accessibility

Available on the internet within NHS Network through a reporting website that is maintained separately from the oncology database.

Maintenance

A developer or someone with SSRS knowledge can maintain the reporting service.

Usability

The reports can have a straightforward interface. There is a range of graphs and reporting tools available, but they are not as modern/complete as the *PowerBI* reporting tools [129, 134] described next.

The lifetime of product

It depends on the lifetime support of the SSRS for the SQL Server.

4.6.3 PowerBI**Overview**

PowerBI [129, 134] is a Microsoft reporting tool and their latest reporting service. Reports are hosted either for free on the Microsoft Cloud or viewed entirely through the desktop tool [129]. Alternatively, the reports can be hosted using a local server at a cost. A desktop tool is required to build the reports and upload them to the report server.

Cost

At the time we perform the analysis for the reporting service, the *PowerBI* Desktop Edition was free if the reports were uploaded to the Microsoft Cloud or managed through the desktop software. *PowerBI* Premium is available to host reports on a local server by purchasing its licence.

Setup

The NHS has a clear policy that disallows the use of cloud storage services in order to prevent identifiable data leaving the trusted environment of the NHS Lothian network. For this reason, the appraisal of the reporting service could not make use of available solutions such as *PowerBI* cloud services, and all the reports had to be stored locally in the NHS network. For a solution based on cloud services to be practical, the *eHealth* needs to acquire a local *PowerBI* server within the NHS Lothian network.

Accessibility

If the reports are uploaded to the Microsoft cloud, the reports can be accessed anywhere using the internet. Otherwise, the access will only be available from within the NHS Network.

Maintenance

A developer or someone with *PowerBI* knowledge can maintain the reporting service.

Usability

The reports will have an interactive dashboard for showing the data visualisation. *PowerBI* provides a wide range of reporting options. It has a complete/extensive reporting functionality similar to the Tableau reporting tool [27] which we describe next.

The lifetime of product

It depends on the support provided by Microsoft. As one of Microsoft's newest reporting tool [129], it may have more lifetime advantage than the SSRS.

4.6.4 Stimulsoft

Overview

Stimulsoft [162] is a reporting tool for ASP.NET that *ChemoCare* used for their earlier reports. *ChemoCare* is a chemotherapy electronic prescribing system used by NHS Lothian. However, *ChemoCare*'s new reports are all now custom-built in-house and no longer use *Stimulsoft*.

Cost

Stimulsoft is available as a reporting service option and can be hosted on a local server by purchasing its licence (e.g. single or team license).

Setup

A developer can install the reporting service as part of the database. It can be set and maintained as an integral part of the database web pages, which was developed with *ASP.NET*.

Accessibility

The reporting service is accessible within the NHS Network.

Maintenance

A developer will be required to maintain the reporting service, and it would be maintained as part of the database web pages.

Usability

Due to the lack of an available license, we were not able to try building a report with this framework. Hence, we cannot determine the exact usability for *Stimulsoft*.

The lifetime of product

It depends on the lifetime of the *Stimulsoft* service and its *ASP.NET* support.

4.6.5 Tableau

Overview

Tableau is third-party software tool for interactive data visualisation and reporting [116]. NHS Lothian has a license for a *Tableau Server* to host reports on the NHS Lothian network. A license is required for *Tableau Desktop* so developers can build the reports and upload them to the report server. Regular users that view reports do not pay a fee. They can either view reports through a browser over the network or have *Tableau Reader* installed for free on their computers.

Cost

A license is required for developers to build reports.

Setup

The NHS Lothian Analytical Services team have offered to order *Tableau Desktop* on our behalf if the board would like to use *Tableau*. Any setup will be done in association with Analytical services.

Accessibility

The reporting service is accessible across the NHS Lothian Network, and computers with *Tableau Reader* installed. *Tableau Reader* is a desktop application that used to open and interact with data visualisations built in *Tableau Desktop* [164].

Maintenance

A developer with knowledge of how to build *Tableau* reports will be required to maintain the reports. Furthermore, the *Tableau Desktop* software is needed to maintain reports.

Usability

Through our initial tests, both *Tableau* and *PowerBI* report generation learning curve are not as steep as for the other options. Both options have complete/extensive reporting capability. Figure 4.1 shows an example of data visualisation developed with *Tableau desktop*.

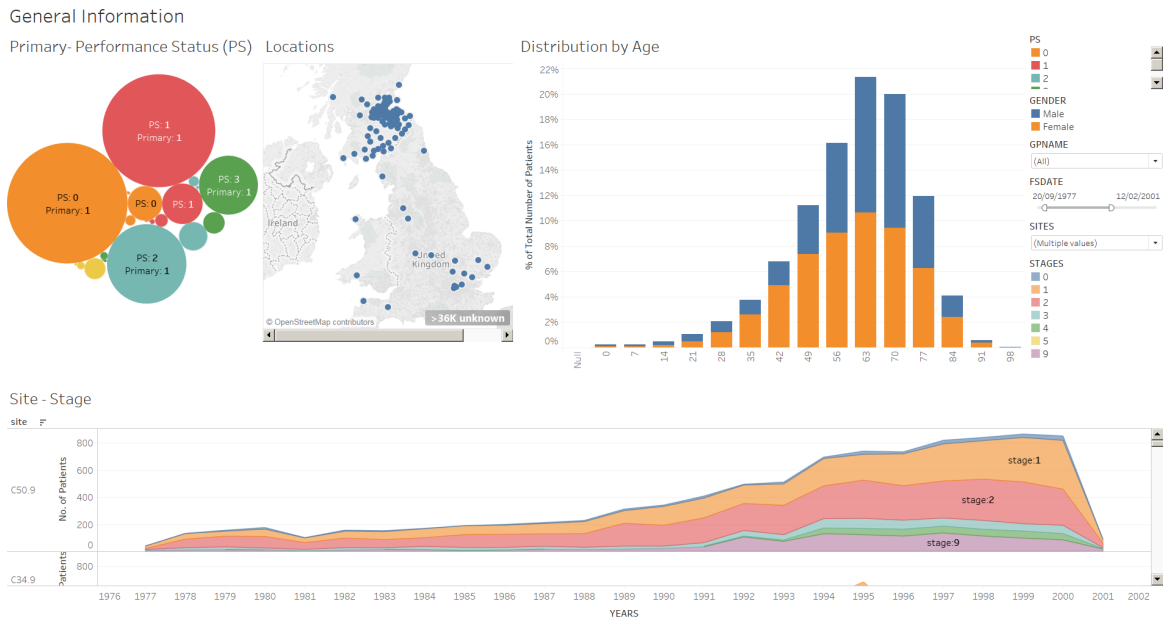


Fig. 4.1 Patients' diagnosis with Tableau

Figure 4.1 shows an example of the data that can be observed and visualise from the Oncology DB with the Tableau reporting service. Tableau eases the development of data visualisation with its built in chart and filter. Tableau supports different charts, such as line, bubble, bar chart as well as the geographical location.

The lifetime of product

It depends on the lifetime of the *Tableau* software and the *Tableau* server license.

4.6.6 Shiny

Overview

Shiny/R Shiny is a third-party reporting software that uses R to create reports for deployment on an R Shiny server [133]. To develop a reporting service with *Shiny*, knowledge of R is required along with knowledge of the Shiny app or library to build a user interface for the report. The reports would be deployed on a local R Shiny server or to the cloud.

Cost

Shiny Server open source is free while the pro edition requires a licence and fee.

Setup

We need eHealth assistance to set up, support and maintain either the free or pro edition of shiny-server to host reports locally. *Shiny Server*, *Shiny Server Pro* and *RStudio Connect* require a local Linux server to run. For the free open source version, we would need eHealth's permission.

Accessibility

The reporting service will be accessible across the NHS Lothian Network if a local server is used.

Maintenance

A developer with knowledge of R and the *Shiny* API can develop and maintain the reports.

Usability

We have not tested developing reports with R shiny. However, with R integration, it can use R statistical libraries. Hence, it is possible to generate interactive and comprehensive reports with R Shiny [100].

The lifetime of product

It depends on the lifetime of the R Shiny software.

4.7 Summary and Contribution

We managed to migrate the Oncology legacy database to the production SQL Server database. The NHS Lothian manager responsible for overseeing the migration task, has confirmed via email in March 2020 that the new database has gone live in the production server from March 2020. Edinburgh Cancer Centre uses the SQL database as a data warehouse. The SQL Server provides more robust, secure, and scalable database than the Microsoft Access databases. Further, the database has an integration to both *ChemoCare* and *Trak* using stored procedures developed by the NHS Lothian team.

Chapter 5

Cancer Waiting Time (CWT)

5.1 Introduction

For some variation of cancer, such as lung cancer, early diagnosis and treatment can be decisive in saving a patient's life. This project looks into the time period between diagnosis and the start of treatment. From what the current recommendation at the ECC and what the data tell us, patients with chronic cancer have to undergo a series of treatments to attempt the eradication of the disease [178].

Before starting a treatment plan that has been agreed by board members consultants, there is a period of waiting, which according to recommendation usually ranges from 14 to 62 days [121]. This period of waiting before the decision to treat the patients is known as *cancer waiting time (CWT)*.

Figure 5.1 shows the cancer waiting time pathway for lung cancer. Within the CWT period, patients are referred by their general practitioner (GP) and then undergo several tests. During the next 14 to 28 days, the patient will have further tests. Once the board agrees on the first treatment, the decision is made within 28 to 62 days. Usually, the waiting time should not exceed 62 days. However, when looking at data, there are many cases when it does happen. This is particularly problematic for lung cancer for it has a poor prognosis: over half of people diagnosed with lung cancer die within one year of diagnosis and around 17.8% within the 5-year survival [178].

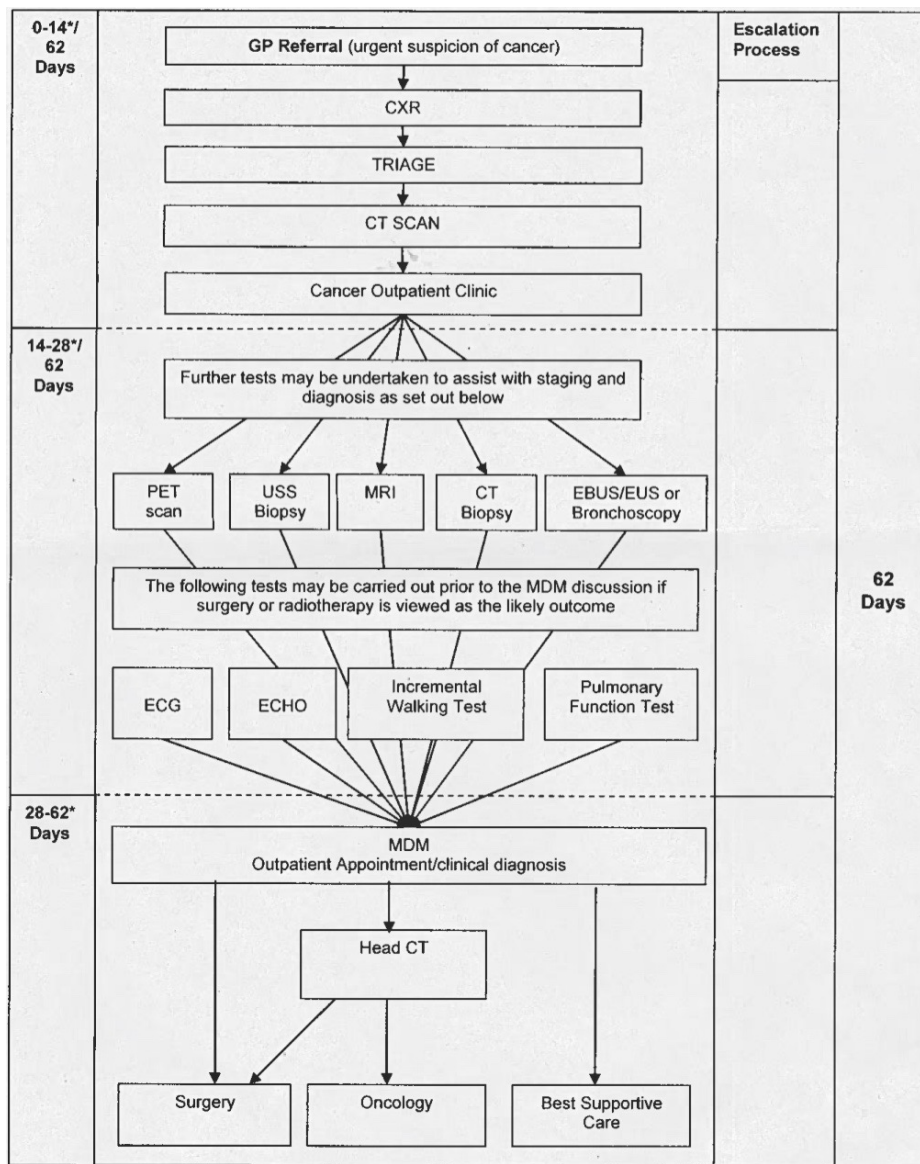


Fig. 5.1 Lung cancer NHS Lothian pathway guideline. Source: [120]

With the aim to achieve the best patient outcome, the intention is to decrease the waiting time from the suspicion of cancer and GP Referral to the first treatment. In this project, we analyse a given CWT dataset through simulation and compare the ‘actual’ pathway derived from data with the NHS Boards, Our analysis can detect and highlight bottlenecks and transition times that exhibit high variability across patients. These insights provide quantitative evidence that might be used to design and revise follow-up interventions. For example, insight gained from simulation can lead to new policies to reduce the variability of specific identified transition times and hence may lead to overall improvements in CWT

and reduction in the cumulative delays. In addition, this will assist in the redesign of a more optimal pathway with improved patient outcomes.

5.2 Related Work

Discrete Event Simulation (DES) has been widely use in many sectors [18] including healthcare [7]. DES is particularly useful to understand the complex process and identify tasks/steps that are resource-intensive, take more time, and so on. Through simulation, we can measure how simple changes can affect the overall performance of a process. DES allows health professionals to assess the efficiency of the existing healthcare delivery system. In healthcare, it has many applications ranging from forecasting the impact of changes in patient flow [79], examining the resource needed and available within a hospital [20], to observing the improvement of patient experience in an emergency department [3]. DES is particularly beneficial for our CWT project because it make it possible to investigate the complex relationships between various model variables (e.g. rate of arrivals, rate of service, etc). This information allows us to consider the management/system alternatives that can be used to reconfigure existing systems, to improve system performance, and to plan new systems, without altering the present system [79]. In a comparable context to our own project, a DES model has also been developed in the field of radiation therapy to reduce patient waiting time and improve the waiting time within the treatment process [17] by treating all lung cancer care steps from the first GP referral to the first treatment as a series of events, we can simulate the pathway with DES. This enables us to improve the understanding of the treatment process, complexities, the patient flow and any existing bottlenecks for the CWT pathway process.

5.3 Data Characteristic

To analyse the correctness of the CWT pathway guideline from Figure 5.1, we use a dataset for lung cancer from three different hospitals within the Lothian. The dataset contains 642 lung cancer patients from 2016 to 2019.

The dataset contains information regarding the outpatient and inpatient treatments, and patients' orders. It contains the information related to the type of the event (e.g. services for the outpatients, order_item for orders), the event start date, event execution date, event finish date. The dataset also contains other information as shown in Table 5.1:

Table 5.1 CWT dataset

Column	Info
SOURCE	The type of event (Inpatient, Outpatient, Orders)
CRN	Case Reference Number
CHI	Patient Identifier Number
DATE_OF_RECEIPT	GP Referral date
DATE_OF_FIRST_TREATMENT	The date of the first treatment
CANCER_TYPE_SPECIALITY	Speciality associated to the type of cancer, such as lung cancer and urology
TRAK_ADMISSION_DATE	The admission date (for inpatient cases)
TRAK_DISCHARGE_DATE	The discharge date (for inpatient cases)
START_DATE	The event start date (inpatient)
END_DATE	The event end date (inpatient)
SPECIALITY	The department in charge of the event
APPOINTMENT_DATE	The appointment date (for Outpatient cases)
UNIQUE_APPOINTMENT_ID	The Unique ID for the Appointment
APPOINTMENT_TYPE	The type of the appointment (New patient, Return patient)
SERVICE	The event type (for Outpatients)
ORDER_STATUS	The status of the order
EXECUTION_DATE	The execution date (for ORDERS)
FINISH_DATE_OF_THE_ORDER	The end date/finish date of orders' event
ORDER_START_DATE	The start date of the order
ORDER_DATE	The start date of the order being processed
UPDATE_DATE	The date of order being updated
ORDER_CATEGORY	The category of the orders
ORDER_ITEM	The type of order
MODE	The type of the first treatment given to the patients
QUESTIONNAIRE_HOSPITAL	The hospital location

5.4 Data Analysis and Query Development

On this project, we aimed to help the health professionals analysing lung cancer waiting time data. After several discussion with the health professionals, we used the dataset to observe the events shown below. Below is the list of analysis we performed for creating the NHS Lothian DES.

- Calculate the waiting time and the number of patients between the first referral to the definitive multidisciplinary meeting (MDM). MDM is one of the first events before the patient's treatment. In this meeting, the oncologists, consultants, and the other healthcare professionals determine whether a patient needs further care. The definitive MDM is the last MDM before the patients' treatments (e.g. radiotherapy, surgery).
- Calculate the number of patients in each group of tests requested (i.e. *CT Scan*, *CT Biopsy*, *PET*, *Bronch/EBUS*, *Surgical staging/Biopsy*).
- Calculate the time from the first outpatient (OP) appointment to the definitive multidisciplinary meeting (MDM).
- Calculate the number of patients who have more than one outpatient (OP) appointment before the definitive MDM.
- Calculate the waiting time between Triage to the first OP appointment.
- Calculate the waiting time for these tests: CT Scan, PET Scan, CT Biopsy, grouped by the type of first treatment given to the patients.

To calculate the requested observations, we designed the SQL queries to get the information directly from the temporary table we can access in the *Trak* Oracle database. The design of these queries is similar to each other. First, we categorise the event based on the *service* for the outpatient and *order_item* for orders. There are more than 1000 different *services* and *order_items*. However, there is no list to categorise these events. Together with the health professionals, we created a new categorisation list. For some *services/order_items*, we use keywords (e.g. *CT Scan*, *PET Scan*). Once we get the list of the *services/order_items* (event), we perform a manual check. For the other events without a reliable keyword, we choose the event that was performed to a significant number of patients (i.e. more than 50).

Once we categorise the service, we calculate the elapsed days between the categorised events and the *DATE_OF_RECEIPT*, i.e. if the value is null, we use the first treatment date instead. With the elapsed days, we determine the first or last events (i.e. the definitive MDM

is the MDM which has the highest elapsed days). Hence, we can calculate the waiting time and the number of patients between various events, as shown in the following query.

```

WITH TEMP AS (
  SELECT
    source,
    CHI,
    SERVICE,
    APPOINTMENT_DATE,
    FLOOR(APPOINTMENT_DATE - DATE_OF_RECEIPT) AS ELAPSED_DAYS,
    "MODE" AS "MODE",
  FROM CWT_PATHWAYS
  WHERE DATE_OF_FIRST_TREATMENT IS NOT NULL
    AND DATE_OF_RECEIPT IS NOT NULL
    AND SERVICE IN (
      'Discussion - MDM - Lung',
      'FGEB - MDM - SJH',
      ...
      'TWM - MDM - Lung'
    )
  ORDER BY CRN, ELAPSED_DAYS
)
SELECT
  QUESTIONNAIRE_HOSPITAL,
  "MODE",
  FLOOR(AVG(WAITING_TIME)) AS AVERAGE_WAITING_TIME,
  COUNT(*) AS TOTAL_PATIENTS
FROM (
  SELECT
    CHI,
    -- THE LATEST MDM (definitive MDM)
    MAX(ELAPSED_DAYS) WAITING_TIME,
    QUESTIONNAIRE_HOSPITAL,
    "MODE"
  FROM TEMP
  GROUP BY CHI, QUESTIONNAIRE_HOSPITAL, "MODE"
)

```

```
GROUP BY QUESTIONNAIRE_HOSPITAL, "MODE";
```

5.5 Simulation

After analysing the dataset, we simulate the cancer waiting time (CWT) by creating a discrete event simulation (DES). DES has been an effective tool to simulate a wide variety of healthcare issues [180]. Our goals are to compare the pathway in the guideline and the real data. By knowing the difference between the two, we can then investigate and perform optimisation for the process of cancer waiting time within NHS Lothian.

To generate our model, we follow these following steps:

- We define the problem. Here, we want to simulate the pathway from the first referral of the patient by their general practitioner (GP) (i.e. the first time the patient were seen in the hospital) until the moment the patient receives the treatment. The time elapsed between these two events is defined as cancer waiting time. We are interested in finding the event/occasion when the elapsed days exceeds 62 days (the maximum days for cancer waiting time according to the guideline).
- From the guideline, we model the process flow of the system.
- We determine the composition of the system (e.g. patients, MDM team) and assign them to the appropriate part of the model (e.g. state, resources).
- Once all the components have been defined, we choose the performance metric for our simulation. During this process, we identify several assumptions for our model.
- After the requirement, process flow, and component have been define, we indicate the model's characteristics and probability distribution (e.g. the rate of incoming patients).
- Once the variable is set, we build the conceptual model and do the iterative process to improve the model.

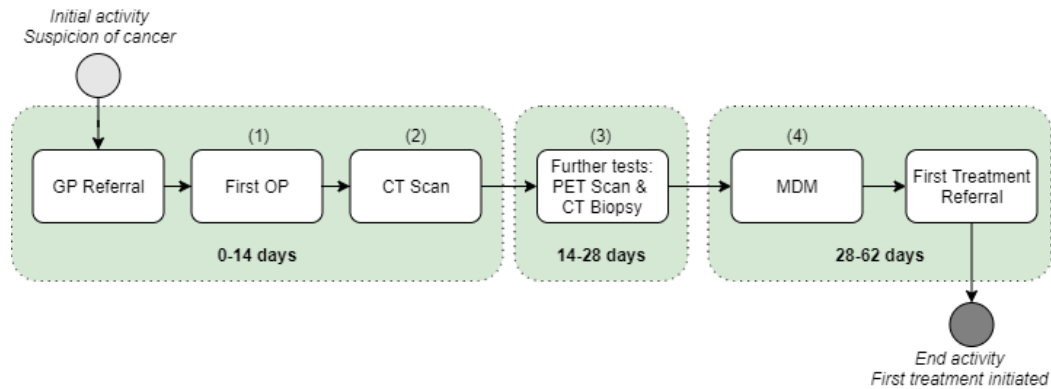


Fig. 5.2 Simplified lung cancer pathway

Before we simulate the CWT, we simplify the pathway shown in Figure 5.1. For our model, we choose several events during the CWT based on the number of event occurrences and categorised them.

- First appointment (*First OP*), such as Triage, Cancer Outpatient appointment.
- *CT Scan* (i.e. primary test before further tests)
- Most common further tests (i.e. *CT Biopsy* and *PET Scan*)
- *MDM*

After choosing and categorising the events, we update the current pathway. Figure 5.2 shows the pathway with the chosen events from the list above. Next, we compare our simplified pathway with the pathway captured from the dataset. From the dataset, we find that cancer waiting time does not strictly follow the pathway from the guideline. The pathway is an iterative process, as shown in Figure 5.3. From our observation, any event can happen in any particular orders, and any tests, appointments, and discussion can happen several times. Because we want to simulate the pathway-based the dataset, we create a DES model based on the pathway in Figure 5.3.

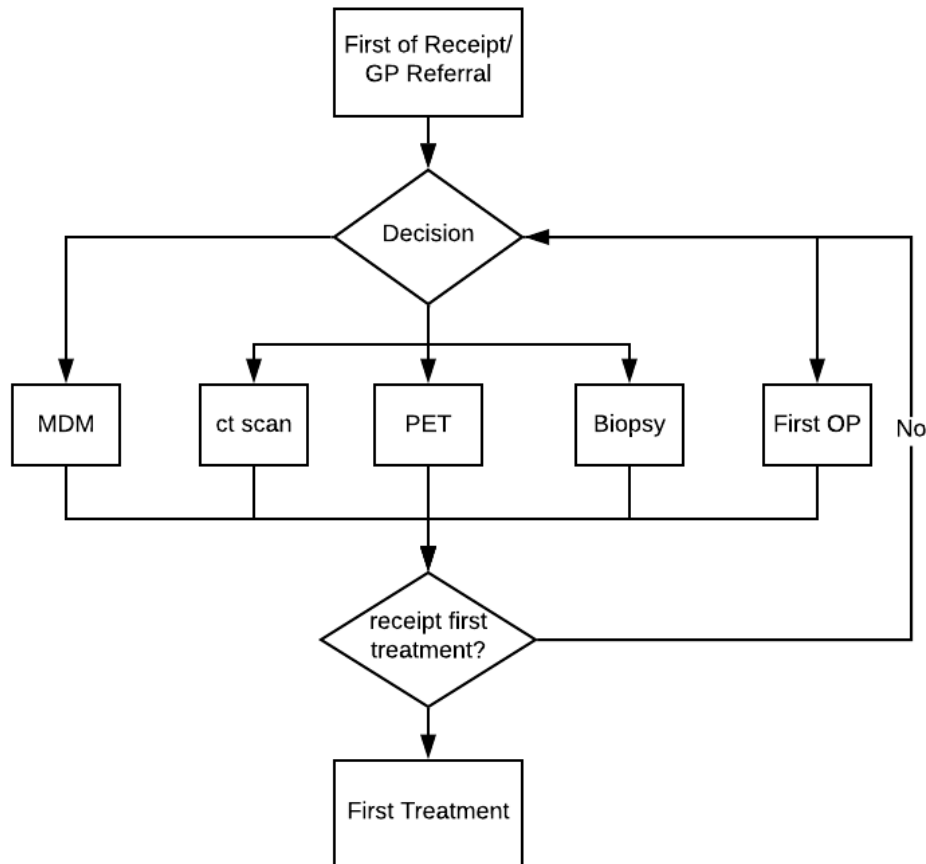


Fig. 5.3 Observed lung cancer pathway

To model the CWT, we focus on determining the system states and the process flow as the other components are correctly handled by the *Salabim* framework we use to model our system. The process flow is shown in Figure 5.3. Table 5.2 shows the state and the distribution for each state/activity. The states/activities are *First OP*, *MDM*, *CT Scan*, *CT Biopsy*, *PET Scan*. In addition to the events shown in Figure 5.4, we add queues to simulate the transitions process where the patients do not undergo any tests, appointments, or discussion as shown in Figure 5.4.

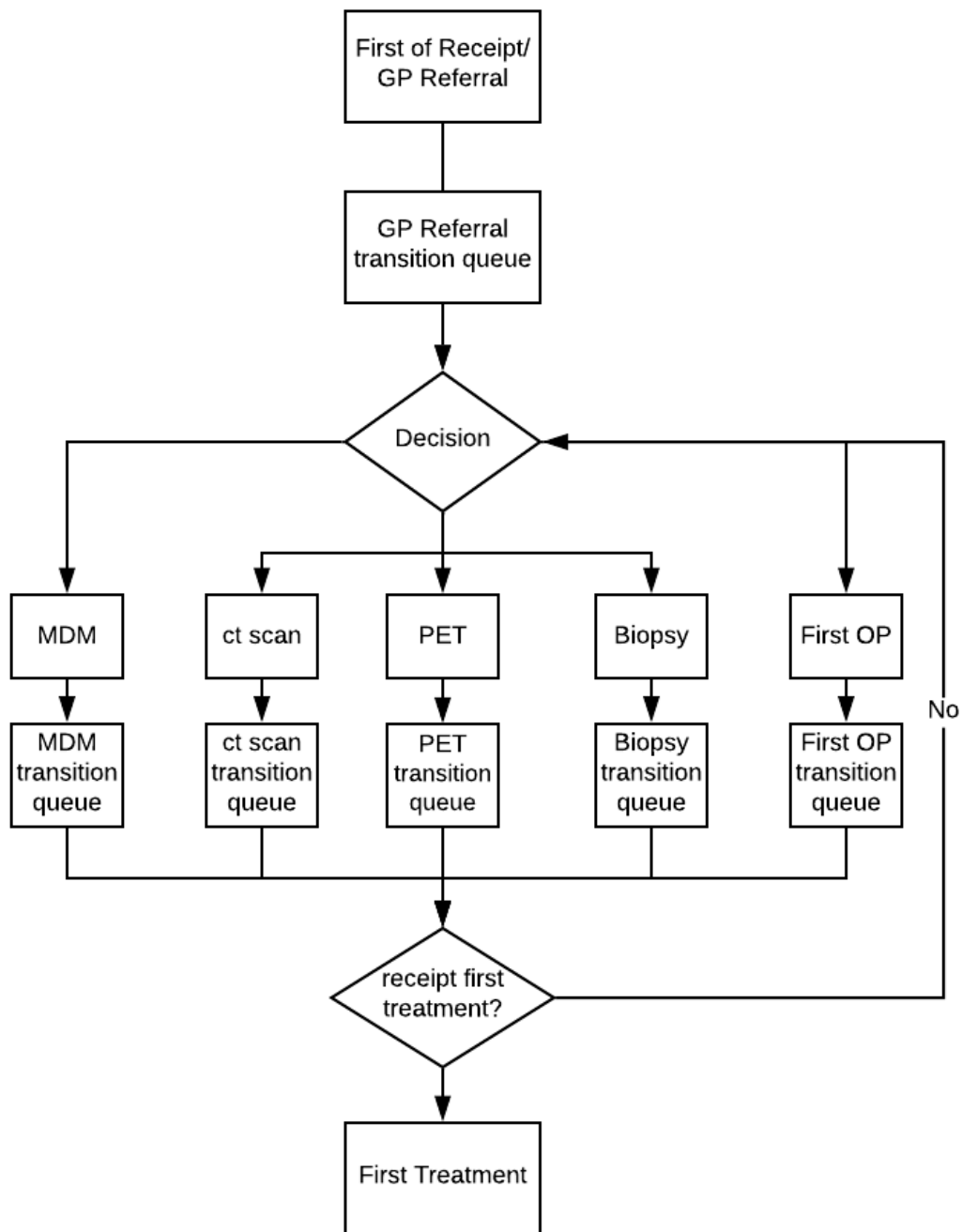


Fig. 5.4 Lung cancer pathway DES

Table 5.2 Cancer Waiting Time States

States	Timing/parameters
Patient Arrival	Interval time: Generalised Extreme value distribution
<i>First OP</i>	Service time: constant (1 day).

<i>CT Scan</i>	Service time: Generalised Extreme value distribution
<i>CT Biopsy</i>	Service time: Weibull distribution
<i>PET Scan</i>	Service time: Generalised extreme value distribution
<i>MDM</i>	Service time: constant (1 day).

As described in the previous section, our dataset contains no information regarding the number of resources (i.e. staff availability or the number of machines). However, we still want to incorporate the resources to our activity, so we simulate the alternatives of having more health care professional and machine to the overall patient waiting time. We calculate the number of resources by observing the maximum number of services and orders (i.e. for the tests) that can happen in any days from our dataset. Figure 5.5 shows the number of services for the selected events. Though these numbers may not be an accurate representation of the number of resources, we can use the figure as a starting point before we obtain the knowledge (e.g. from an interview, survey or another dataset) regarding the resources number and their availability and capacity.

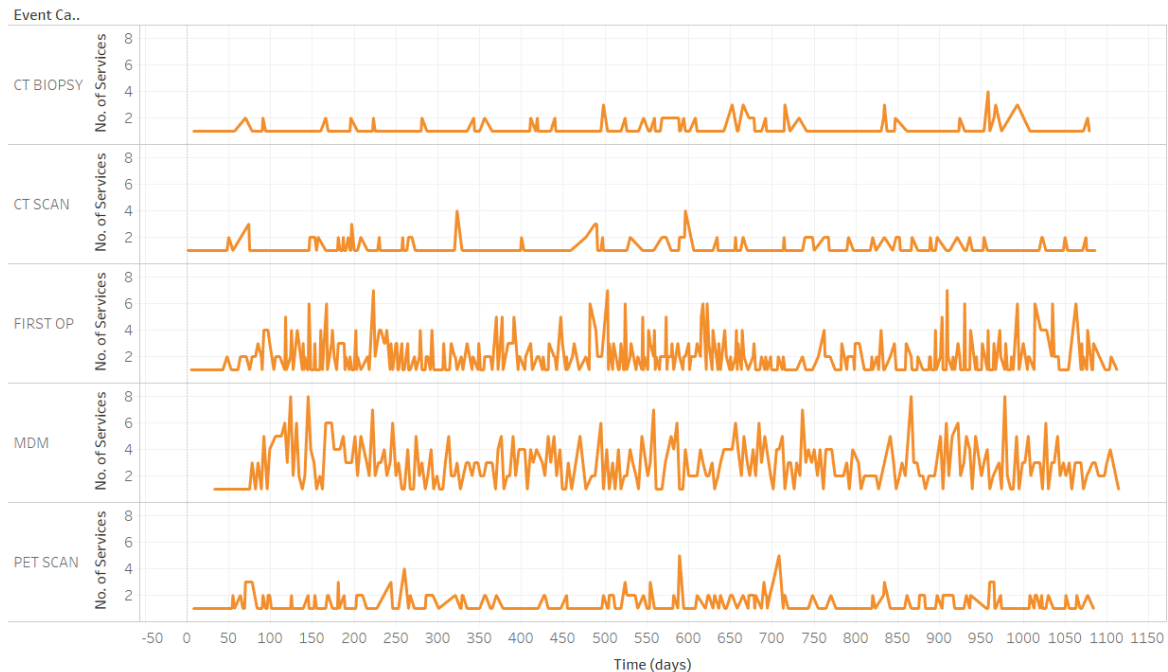


Fig. 5.5 Number of services for the CWT events

To determine the distributions for the states and queues' waiting time, we use the *Kolmogorov-Smirnov test (kstest)*. The *kstest* compares the two sample statistical distributions. It is a non-parametric test which does not require the data to follow the normal

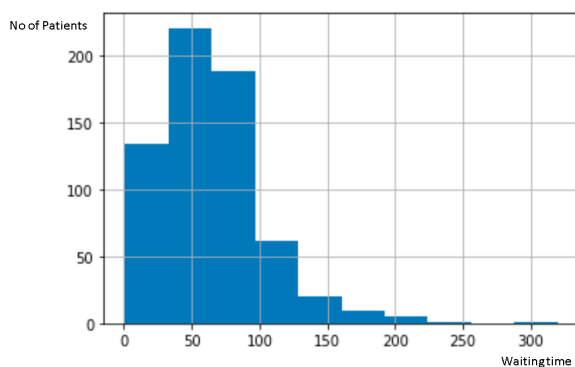
distribution. When we compare the possible distributions (e.g. *normal*, *exponential*) to the dataset, we get the *D statistic* and *p* value. The *D statistic* is the maximum absolute difference between two distribution functions. By comparing the *D statistic* and *p* value for each distribution (i.e. minimum value), we can determine the best possible distribution for the data. The procedure is shown below.

Algorithm 1: get_best_Distribution

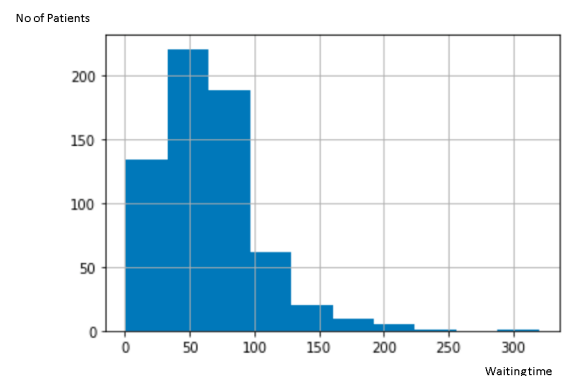
Input: data
Result: *best_distribution*, *parameters*
distributions \leftarrow list of distributions
dist_results \leftarrow []
for *dist* in *distributions* **do**
 | *parameters* \leftarrow *dist.fit*(data)
 | *D*, *p* \leftarrow *kstest*(data, *dist*, *parameters*)
 | *dist_results.append*((*dist*, *D*, *p*, *parameters*))
end
best_distribution, *parameters* \leftarrow *best*(*dist_results*)

As for decisions, we calculate the number of transitions from each state. For each state, we set different probability/weight for determining the next state (e.g. more than 50% of the patients will undergo *First OP* after *GP Referral*, 49% of patients get their first treatment after *MDM*)

5.6 Validation



(a) Real patients' waiting time distribution



(b) Simulated patients' waiting time distribution

Fig. 5.6 Patients' waiting time distribution

Figure 5.6 shows the waiting time distribution for both real data and simulation. Because the waiting time is not normally distributed, we use Kolmogorov–Smirnov (KS) test to compare both dataset distributions [177]. The result is as follow:

- *KS-statistic* = 0.06984
- *p-value* = 0.152

We know that If the KS statistic is small or the *p-value* is high, then we cannot reject the hypothesis that the distributions of the two samples are the same. From the result, the *p-value* is high (more than 5% level of significance). Hence, we cannot reject that the distributions of the two samples are the same. Furthermore, the result of the descriptive statistic for both datasets shown in Table 5.3 supports the close correspondence between the simulated and observed outcome. Therefore, our model can provide reasonable approximation and observation of the expected system behaviour.

Table 5.3 Patients' waiting time descriptive statistic

Info	Real	Simulated
mean	64.417	65.871
std	37.672	39.035
25% percentile	38	37
50% percentile	61	56
75% percentile	82	84

5.7 Summary

Table 5.4 CWT simulated scenarios

Scenarios	mean waiting time (95% CI)	<62%
50% increase service capacity for <i>First OP</i>	67.057 (59.78, 74.32)	56
limit waiting transition time after <i>First OP</i> (5 days)	63.985 (56.718, 71.253)	60
50% increase service capacity for MDM	59.85 (53.89, 65.82)	62
limit waiting transition time after <i>MDM</i> (5 days)	49.45 (44.143, 54.76)	74
50% increase service capacity for <i>CT Scan</i>	63.911 (56.97, 70.85)	59
decrease <i>CT Scan</i> service time (max 5 days)	64.44 (55.75, 73.133)	59

limit waiting transition time after <i>CT Scan</i> (5 days)	64.300 (57.61, 70.98)	58
50% increase service capacity for <i>CT Biopsy</i>	65.97 (59.12, 72.81)	53
decrease <i>CT Biopsy</i> service time (max 5 days)	69.56 (58.72, 80.4)	51
limit waiting transition time after <i>CT Biopsy</i> (5 days)	65.811 (58.11, 73.512)	54
50% increase service capacity for <i>PET Scan</i>	65.80 (54.37, 77.38)	53
decrease <i>PET Scan</i> service time (max 5 days)	65.03 (53.027, 77.052)	53
limit waiting transition time after <i>PET Scan</i> (5 days)	65.80 (54.37, 77.28)	53

We create the DES to simulate different scenarios/alternative for the lung CWT process. Table 5.4 shows several different scenarios and its simulation's result.

From the scenarios, we know that increasing the service capacity for *MDM* and decreasing the transition time after *MDM* can significantly improve the percentage of patients getting the first treatment before 62 days. The result is as expected because almost all patients will undergo *MDM* before getting their first treatment. We also conclude that most delays are caused by the transition time instead of the service time/availability of resources (i.e. from the *First OP* and *MDM*). Adding more resources (by increasing the service capacity) for *CT Scan*, *PET Scan* or *CT Biopsy* has less impact on the overall patients waiting time.

As shown in Table 5.4, the probability of *MDM* or *First OP* as the next state is higher compare to the likelihood for the tests (i.e. *CT Scan*, *PET Scan*, *CT Biopsy*). Even though we can simulate the lung CWT events, our model has many limitations. First, we do not consider the other services (e.g. MRI, Head CT, Pulmonary function tests) by merging all of the other services into queue transition after the states. During the simulation run, the result of simulation may contain several outliers compare to the dataset as shown in Figure 5.6 because the distribution function that we use (e.g. exponential, gen-extreme) to get the service distribution. Also, the number of resources for our model is based on estimations. In the future, we can improve the simulation by incorporating more services and update the resource number and service time with the information given by the hospital (e.g. by interview or another dataset extraction). We can improve the simulation by adding more characteristics, such as adding the type of the first treatment of the patients. This may improve the simulation accuracy because the average waiting time is vary based on the patient's first treatment, as shown in Table 5.4.

With the use of the DES model, we can evaluate every aspect of the CWT procedure without allocating real resources. Hence, the evaluation will not disturb the current healthcare services given in the NHS Lothian. Furthermore, new policies and different CWT procedures or new methods can be simulated, analyzed, observed in a short period of time.

All in all, even with the model limitation, we can highlight the bottleneck process and give a close observation regarding the alternatives scenarios for the lung CWT. The bottlenecks of the lung CWT are the *MDM* state and the *MDM* transition. Once we add the others services, fine-tune the service time and resources number, the DES can help to provide insight and second opinion to improve the overall cancer care for the lung CWT.

5.7.1 Contribution

The NHS Lothian requested the SQL scripts used for analysing the data for cancer waiting time. At the moment of writing this thesis, the SQL scripts for analysing the CWT data extraction are used to help with the audit process in Edinburgh Cancer Centre (ECC).

Chapter 6

Toxicity Predictor

6.1 Introduction

Chemotherapy is one of the primary treatments used for treating cancer patients and new anti-cancer drugs because the chemotherapy treatments are constantly being developed and researched [123]. However, chemotherapy treatments can cause unpleasant and very serious side effects [30]. The side effects of chemotherapy depend on the type of drugs, dosage, length of treatment as well as the patients characteristics. The ECC at NHS Lothian has collected information on side effects that patients have experienced from the chemotherapy treatments over the year. However, because of the lack of tools for analysing the data, clinicians do not use the full potential of the collected observations. By having a reporting tool for existing data, clinicians can further analyse the treatments given to the patients, and adopt them to suit individual patients leading to better outcomes.

The main objective of this project is to develop and train a machine/deep learning model for enabling clinicians to predict the toxicity outcome for patients based on their characteristics. For this project, we train, evaluate, and compare several different techniques, including a Markov model (MM), a Hidden Markov model (HMM), a Random Forest (RF) and a Recurrent Neural Network (RNN), to predict the outcome (the toxicity level) of chemotherapy treatments for breast cancer patients. We believe that finding the correlations and patterns between patients and their treatments is essential to improve the overall healthcare system offered by the NHS Lothian.

6.2 Related Work

In the past, some healthcare providers have used multivariate programs to help diagnose and characterise cancer, such as prostate cancer. In addition to helping predicting patient diagnosis, some programs can be used to forecast the prognosis of the patients as well [72]. Similarly, there has been a lot of ongoing research to develop multivariate systems aiming at designing personalised cancer treatments for patients by corporate institutions (e.g. Google [61], IBM Watson[98], Microsoft Research[130], NHS[175]) and academic research. Such studies use several different techniques, such as random forest [119], Bayesian logistic regression [99], and support vector machine classifier [31] to predict cancer diagnosis and/or prognostic. In addition to using machine learning, we also adopt and explore other techniques commonly used in Natural Language Processing (NLP) such as Hidden Markov Models (HMM) [80] and Recurrent Neural Networks (RNN) [65] because our data extraction consists of sequence data. We believe that exploring different techniques more widely used in NLP can be beneficial in our context since our data contains sequences as well (for all different events associated with the treatments).

HMM is a sequence model for part-of-speech tagging. A sequence model is one whose job is to assign a label or class to each unit in a sequence, thus mapping a sequence of observations to a sequence of labels. Given a sequence of units (words, letters, morphemes, sentences, and so on), a HMM computes the probability distribution over possible sequences of labels and chooses the best label sequence [80]. We believe that exploring different techniques more widely used in NLP can be beneficial in our context since our data contains sequences as well (for all different events associated with treatment).

RNN is one of an enhancement to a neural network algorithm because RNN add another time dimension to the input dataset. RNN consists of several layers of ANNs, which allows us to process sequence data for which the input can be longer than one sequence [65].

6.3 Methodology

In this project, we followed the iterative steps as shown below to create the model [16].

- **Decide the target outcomes.** Before training the models, we define the problem/outcome (i.e. label/target answer) of our models. Here, we want to predict the patient's toxicity after each chemotherapy treatment.
- **Collect the labelled data.** Labelled data is data for which the target answer is already discovered. To train the model the dataset must contain two main elements:

- The target answer. Here, we use toxicity scale based on the Chemotherapy Risk Assessment Scale for High-Age Patients (CRASH) score [48]. For our dataset, we have 0 toxicity categorised as low toxicity and 4 as high toxicity.
 - Variables/features. The features/variables are used to identify patterns for predicting the target answer.
- **Analyse the data.** We analyse the dataset to determine the data characteristic and correlation between the variables/features and target before feeding the labelled data into the model. High correlation implies that there is a relationship between the variable and the target class. We want to include variables with high correlation because they are the ones with higher predictive power (signal), and leave out variables with low correlation because they are likely irrelevant.
 - **Features Processing.** The feature processing step includes transforming the variables further to make them more relevant for the model. Including relevant features helps to improve prediction. It is common to include all features in the model training and then gradually exclude the irrelevant features because it is not always possible to know the features with predictive influence in advance. Below are examples of feature processing:
 - Replace missing/invalid data with more meaningful values.
 - Perform non-linear transformation, such as binding the numeric variables to categories.
 - Define the domain-specific features and the variable-specific features
 - **Split the data into training and evaluation data.** To train the model, we split the data into a training and an evaluation subset. Our splitting ratio is **70-80% for training** and **20-30% for evaluation**. As the name suggests, we then use the training data to train and the evaluation data to evaluate the predictive quality of the trained model.
 - **Model Training.** The model will learn from the training data and update its characteristics. The model can be used to get the predictions on new data for which we do not know the target answer.
 - **Evaluate Model Accuracy.** For our predictors, we use accuracy, precision, recall and f1-score as the metrics to measure the predictive accuracy of the model.
 - **Improve the model accuracy.** We perform some iterative steps to improve the model accuracy as shown below.

- Add/decrease the variables/features used to train the model.
- Tune the model parameter that is, we consider some alternative values for the training parameters
- Increase the percentage of data used for training.

6.4 Data Analysis

6.4.1 Data Characteristics

To develop our machine learning model, we use a data extraction from Edinburgh Cancer Centre with information on treatment cycles, recorded side effects (here, toxicity level), and various observations concerning breast cancer patients for three years (from 2014 to 2016).

The extraction has data for 51661 treatments of which 13030 are of breast cancer treatments. There are 933 unique patients, and some patients have two or three different treatments/regimes during the time period. Each regime has several cycles ranging from one to more than 50 cycles (e.g. 85).

Table 6.1 The treatments' intentions

Intention	Total patients
Adjuvant	620
Neo-Adjuvant	427
Palliative	483
Curative	17

Table 6.1 shows the number of patients for different intentions. We exclude the *Curative* treatment because we did not have enough data for training our model.

Along with the general patients' characteristics extraction, we receive the toxicity level and measurement of the patients in separate flat files.

We combine the data by connecting the treatment appointment date with the date when the toxicity and other measurements (i.e. weight, height, surface area) were obtained. We ignore patient data with no toxicity information. After we perform data cleansing, we have 2752 instances (i.e. 213 patients) for the *Palliative* treatment, 1855 instances (i.e. 382 patients) for the *Adjuvant* treatment, and 1209 instances (i.e. 205 patients) for the *Neo-adjuvant* treatment.

Table 6.2 shows the pre-processing steps for the chemotherapy data.

Table 6.2 Preprocessing steps of chemotherapy data

Task	Detail
Merge the dataset	We merge the <i>ChemoCare</i> treatment and toxicity data using the <i>Last Toxicity Date</i> as the foreign key.
Select the relevant variables	CHI, weight, height, intention, regime, cycle, performance status, nausea, vomiting, diarrhoea, constipation, oral mucositis, oesophagitis, neurotoxicity, hand/foot, skin, hypersensitivity, fatigue
Filter the dataset	We only use breast cancer data
Derive several variables	We derive the patients' age from CHI and BMI from weight and height. We calculate the patients' toxicity from these following fields: nausea, vomiting, diarrhoea, constipation, oral mucositis, oesophagitis, neurotoxicity, hand/foot, skin, hypersensitivity, fatigue.
Clean	We perform the data-cleansing (e.g. remove duplicates, delete the dataset without toxicity, perform regression imputation for the missing performance status)
Separate the dataset	We divide the dataset into three different sets based on the intention.

6.4.2 Feature Analysis

Before we input the data into the models, we order the data by the cycles to make sequences. The cycle field determines the number of treatments the patients undergo treatment for a particular regimen. We determine the target answer (i.e. toxicity) and predictors. Once we categorise the fields, we check the relation between each predictor in the dataset to the toxicity outcome. Fig. 6.1 shows that at the beginning of the treatment, most of the patients have low toxicity which is to be expected.

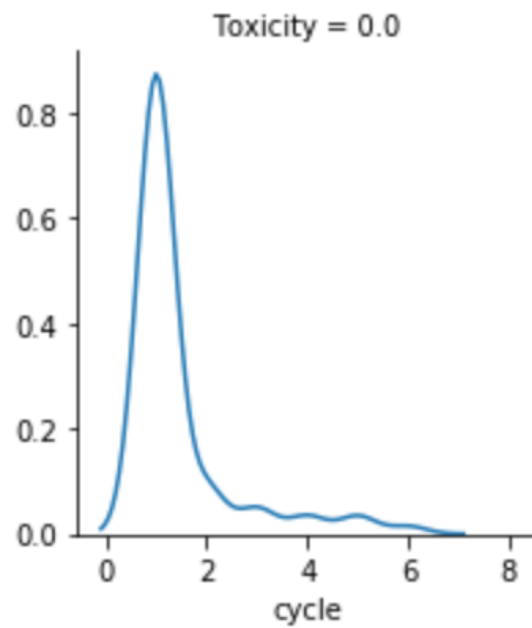


Fig. 6.1 Patients' proportion against low toxicity

For our models, we first include all the variables in the dataset and gradually decrease it until we were left with age, BMI, regime, cycle, toxicity and performance status. Next, we calculate the Pearson correlation [69] between the selected predictors to the target answers (i.e. for the adjuvant intention), as shown in Figure 6.2. We want to include variables with high correlation because these features are assumed to be the ones with higher predictive power [16].

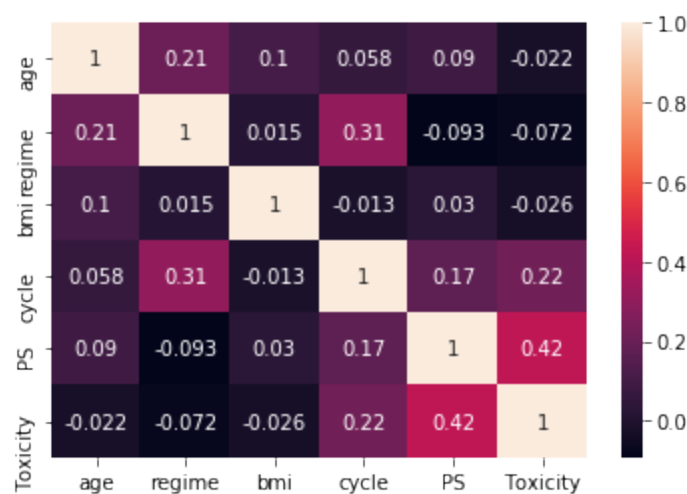


Fig. 6.2 Adjuvant therapies fields' correlation map

Finally, we perform data cleaning. We replace some of the missing/invalid data in our predictors. We use the mean average for fields like age or body mass index (BMI) while we use regression imputation for the performance status (PS). To avoid the class imbalance problem, when some regime has more data than the others, we create a new dataset by duplicating some of the data. We perform this only for the RF model training because, unlike for the other models used (HMM and RNN), our RF model is not dependent on the previous observation. For example, we have 141 patients treated with *FEC (D)* while only 80 patients treated with *PACLITAX*. Here, we then copy some of the data from the *PACLITAX* to match the number of patients treated with *FEC (D)*.

6.5 Predictor Models

Based on the Markov assumption, our first assumption for creating the toxicity predictor is that the toxicity of the patients depends on the previous toxicities and treatments. Hence, we compare different techniques that incorporate prior toxicity to the calculation. Table 6.3 shows several brief comparisons of the models that we develop for this project. Finally, we also analyse the technique/ statistical model (i.e. survival analysis) that is widely used in the medical domain.

Table 6.3 Models Comparison

Model	Brief Overview	State-memories
Markov Model (MM)	A statistical model	no state memory
Hidden Markov Model (HMM)	A statistical markov model	one state memory
Random Forest (RF)	An ensemble learning	one state memory
Recurrent Neural Network (RNN)	A sequential Artificial Neural Network (ANN)	all states

6.6 Statistical Models

6.6.1 Survival Analysis

We perform survival analysis on the chemotherapy dataset. Survival analysis is used to analyse the expected duration of time until one or more events happen (e.g. failure time, survival time, event time). By performing survival analysis, we want to investigate the effect

of several variables at the time a specified event takes to happen (i.e. Cox regression for survival analysis).

Before performing the Cox regression and survival analysis, we determine the failure time event and the time unit for our observation. We use day as our time unit and *high toxicity* occurrence as the failure event because we want to know the time until the patient reaches high toxicity during their cancer care. We censor the patient that has never reached high toxicity outcome during their treatment. High toxicity outcome is the condition when the toxicity outcome of the patient reaches level 2. Here, we set the high toxicity when the patients' toxicity level reach level 2 because we do not have enough data where the patients' toxicity reach level 3. For our observation, we want to see the hazard ratio and the survival comparison for the intention fields.

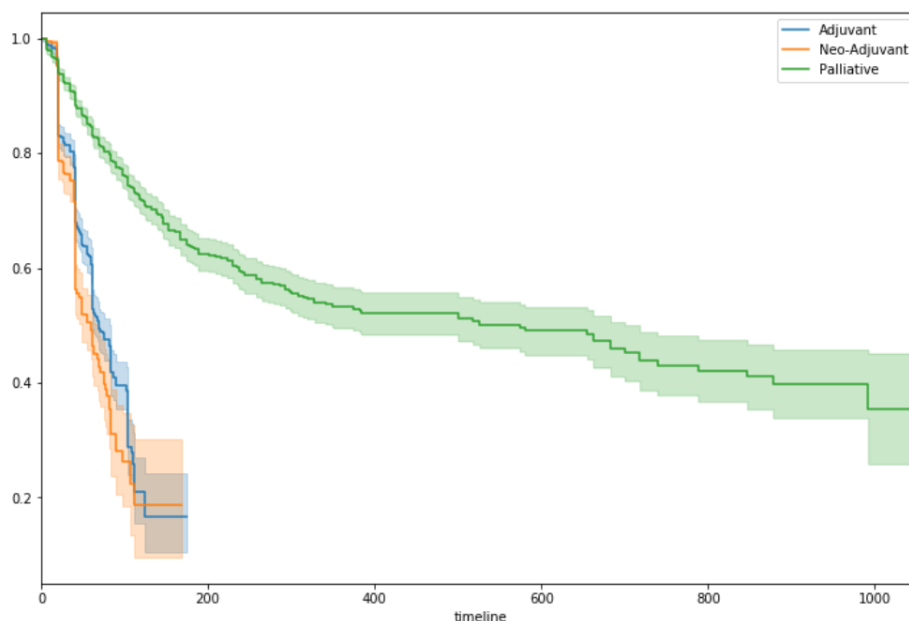


Fig. 6.3 *Kaplan-Meier* survival analysis

Once we determine the failure condition, we perform the *Kaplan-Meier* estimator for several intentions. The *Kaplan-Meier* estimator can be used without knowing whether the hazard is proportional. Figure 6.3 shows the result of the survival analysis for all intentions with the *Kaplan-Meier* estimator.

After plotting the patient survival rate with the *KM* fitter, we calculate and compare the cumulative hazard using the *Nelson-Allen* fitter as shown in Figure 6.4. We know from Figure 6.4 the hazard rate between the *Neo-Adjuvant* and *Adjuvant* intention is not proportional. However, if we compare each intention to the *Palliative* intention, the intentions' hazard rate is proportional. Hence, we can perform the *COX* regression and use the *Palliative* intention

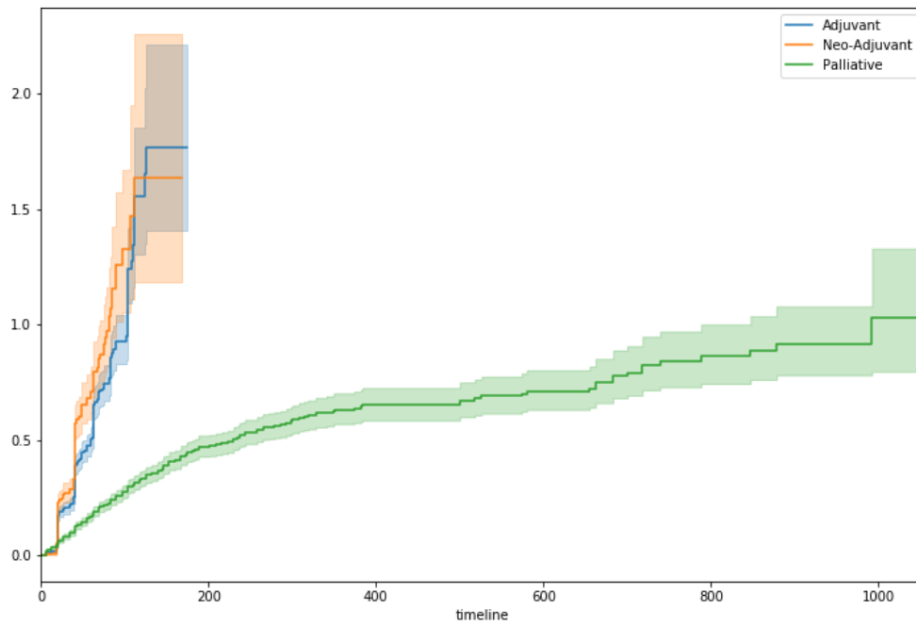


Fig. 6.4 Cumulative hazard function

as the control variables. The result of our *COX* regression fitter is shown in Table 6.4. In the *COX* proportional hazard model, a higher hazard means more at risk of the event occurring. Based on the result from the Table 6.4, we know that both *Adjuvant* and *Neo-Adjuvant* treatments have much higher toxicity outcome than the *Palliative* treatment.

Table 6.4 Intentions hazard ratio

Intention	Hazard ratio	Lower 95% CI	Upper 95% CI	p
<i>Adjuvant</i>	3.51	3.08	3.99	<0.005
<i>Neo-Adjuvant</i>	4.13	3.52	4.84	<0.005

6.6.2 Markov Model (MM)

A Markov model is a stochastic model with the assumption that a future state only depends on the current state [80].

Based on the toxicity in the data extractions, we create a discrete time Markov chain shown in Fig. 6.5 where the states represent the different levels of toxicity (e.g. T_0 corresponds to no toxicity, and T_3 is very high toxicity) and transitions reflect the treatment effects over time. We omit the transition probability values in the figure but show them separately in Table 6.5.

Table 6.5 The transition probability for all adjuvant treatment regimes

	T0	T1	T2	T3
T0	0.062	0.50	0.41	0.03
T1	0.04	0.63	0.31	0.02
T2	0.01	0.44	0.52	0.03
T3	0	0.32	0.64	0.04

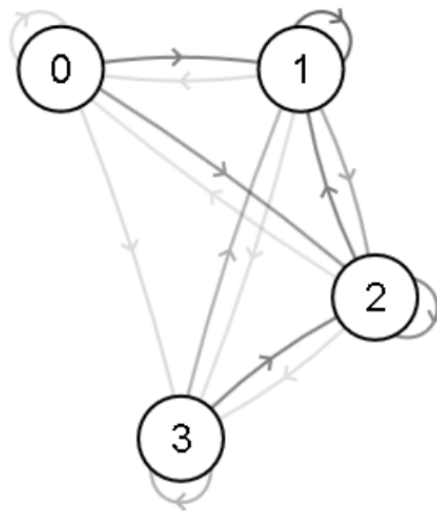


Fig. 6.5 The diagram representing the Markov chain for patients' toxicity outcome

6.7 Machine Learning and Deep Learning Models

From our survival and hazard ratio analysis we perform and describe in the previous section, we know that the *Palliative* treatment has the least hazard in accordance with the patients' toxicity. Hence, We decide to create several different models based on their intention so we can compare the other predictors variables, such as *age*, *regimen*, *cycles*.

6.7.1 Hidden Markov Model (HMM)

In other domain, such as Natural Language Processing (NLP), Hidden Markov Model (HMM) is used as a sequence classifier. A sequence classifier or labeller is a model whose job is to assign label class to each unit in the sequence. Our HMM has an observed Markov model. An observed Markov model is an extension of a finite automata. It is a weighted automaton in which the input sequence uniquely determines which states the automaton will go through. On top of the observed Markov chain, HMM has hidden events (i.e. the patients' toxicities). We use the previous-current, current-next treatments' state and toxicity result to deduct and assign the label to the sequence. For this project, we want to infer/predict the toxicity level based on the patient's characteristics. Table 6.6 specifies the components of our HMM.

Table 6.6 The HMM components for predicting the toxicity outcome

Component	Description
-----------	-------------

States	The toxicity level of the patients (i.e. T0, T1, T2, T3)
Transition probabilities	The transition from one toxicity level to another toxicity level (e.g. from T0 to T1, from T1 to T3, etc).
Observations	The observed events obtained from the data extraction (i.e. cycle, age, BMI, regime). We categorise the value of each observation to simplify the process of training our HMM. For example, 1-2-3-1 denotes the observation for an overweight patient who gets the <i>FEC-D</i> (<i>D</i>) in their first cycle and is aged less than 50 years old.
Emission Probabilities	Each member represents the probability of the observations generated from the toxicity state.

To predict the toxicity from the sequence of the patients' events, and as is usual for HMM, we use the *Viterbi* algorithm. The *Viterbi* algorithm is a dynamic programming algorithm used for finding the most likely sequence of hidden states (aka path). We analyse the observation from left to right (i.e. first cycle to the last cycle) to fill out the trellis as shown in Figure 6.6. Each cell of the trellis $v_t(j)$ represents the probability that HMM in state j after seeing the first t observation and passing through the most probable (maximum probability) state sequence q_1, q_2, \dots, q_{t-1} , given the automaton λ [80]. Formally, each cell probability is calculated, as shown below:

$$v_t(j) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1, q_2, \dots, q_{t-1}, o_1, o_2, \dots, q_{t-1}, q_t = j | \lambda) \quad (6.1)$$

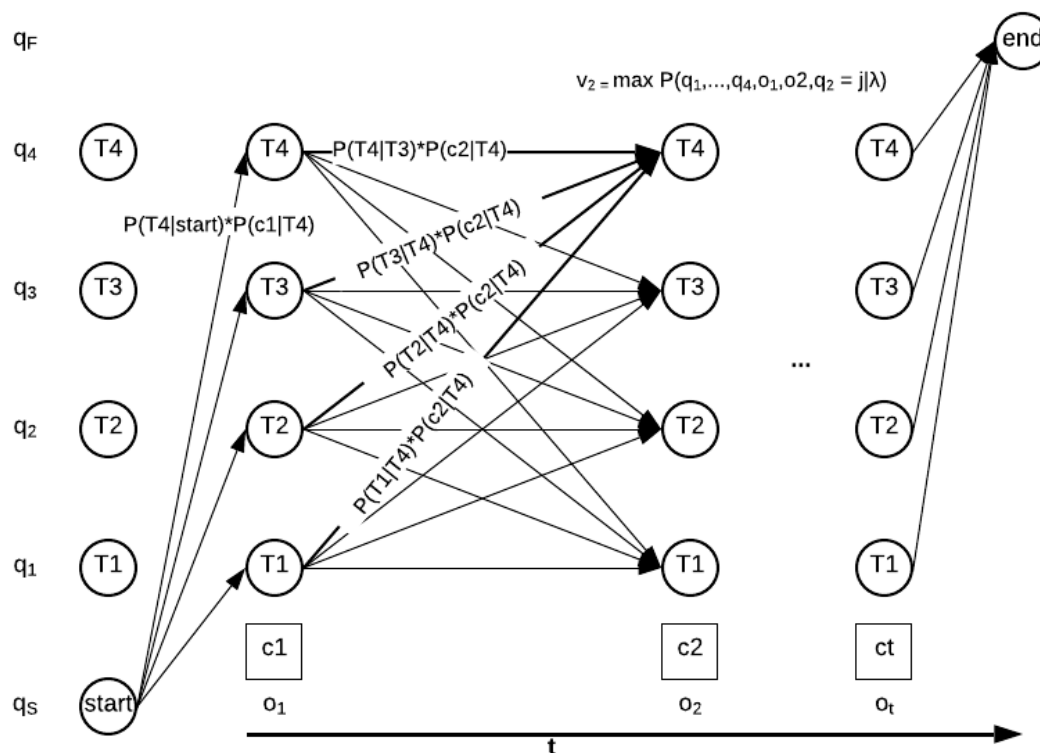


Fig. 6.6 The viterbi calculation for the toxicity predictor

Table 6.7 The transition probability for all adjuvant treatment regimes

Observed events	Toxicity Outcome
1-2-3-2/1-2-3-2/2-2-3-2	T0/T1/T1
1-2-3-4/1-2-3-4/2-2-3-4	T3/T3/T2
1-2-3-4/1-2-3-4/2-2-3-4/2-2-3-4	T3/T3/T2/T2

Figure 6.6 shows the calculation for computing the best part through the hidden state for our toxicity predictor. Here, q_t denotes the hidden state (i.e. patients' toxicity) and o_t indicates the observed state (i.e. the chemotherapy treatment). After calculating the most probable state, we can determine the toxicity of the patients. Table 6.7 shows an example of using HMM to predict the toxicity outcome for patients.

6.7.2 Random Forest (RF)

Random forest (RF) is an ensemble of decision trees for solving classification problems. The random forest classifier uses several features to predict the outcome [15]. For our RF model, we use the following features: *age*, *BMI*, *cycle*, *Regime*, *previous performance status*, *previous toxicity level* to predict the toxicity outcome of the treatment. To simulate one state of memory, we retain the information regarding the patient previous performance status and the previous toxicity. We do not include the information regarding the previous regime because there are no regime changes for every patient in our prepared dataset. For this project, we have not considered the protocol treatments (i.e. where the patients are given several different regimes that belong to one protocol simultaneously as a follow-up treatment). We created three RF models for each treatment intention (i.e. adjuvant, neo-adjuvant, palliative), and categorised most of the features (except age) for training our model. After we created our first RF model, we manipulate the hyperparameters to get a better prediction result. By using one of the functions that are provided by the machine learning library, *GridSearchCV*, we streamline the process of performing an exhaustive search over specified parameters for our models [15]. These hyperparameters are *number of estimators*, *minimum sample leafs*, *minimum sample splits*, and *the maximum depth of each tree*.

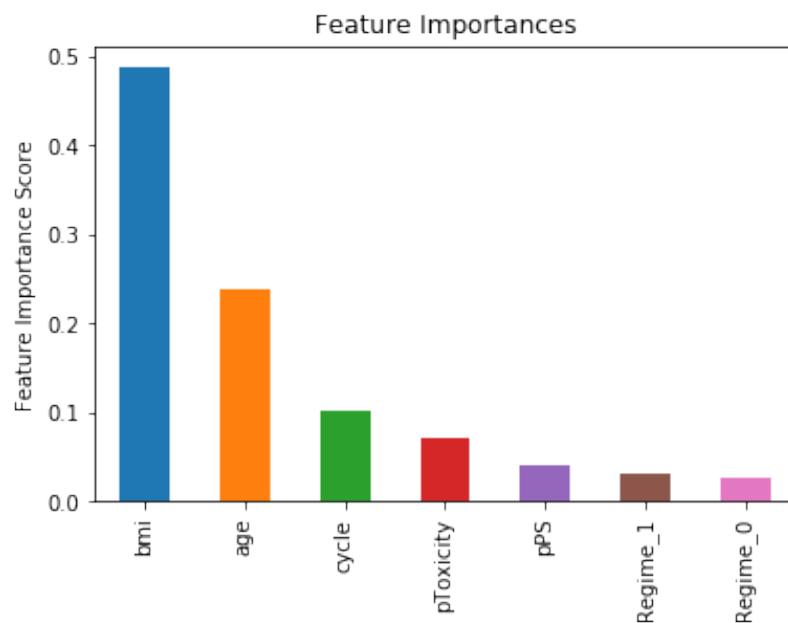


Fig. 6.7 The feature importance in *Neo-adjuvant* treatments

Lastly, we observe the feature importance of each field. We get an estimate of the importance of a feature by computing the average depth at which it appears across all trees

in the forest [15]. The RF libraries we used for this work allowed us to compute the feature importance automatically for every feature after training. Figure 6.7 shows the graph of the feature importance for the fields used to predict the toxicity outcome.

6.7.3 Recurrent Neural Network (RNN)

One of the main functionality of the RNN is to perform forecasting. Forecasting means to predict the next values of time series. Here, we consider toxicity predictor as one of the forecasting problems. We want to predict the next toxicity based on previous treatments and toxicity. Ideally, we want to predict several next treatment outcomes. However, for the first milestone, we only predict the next toxicity. Similarly, we have three models for each intentions.

The RNN models take several inputs and produce one output for each input based on the treatment cycle. During the training, we use similar features from the RF models. Compare to the other machine learning algorithm, the RNN models preserve states across time steps (in other words, has memory cells) [15]. For all models, we use the long short-term memory (LSTM) [65] units.

Here, instead of using a simple RNN cells, we use the LSTM cells to avoid the vanishing gradient problem. For example, Equation 6.2 shows the output calculation of a predictor. The output prediction is a composite function depending on the value of x_1, x_2, \dots, x_T . Here, we can see that x_1 is nested deeply in the equation. By using only the simple RNN cells, the RNN model is vulnerable because the longer the sequence T , the more its gradient vanishes.

$$y(T) = \sigma(W_0^T \sigma(W_{xh}^T x_T + W_{hh}^T \sigma(W_{hh}^T x_{T-1} + \sigma(\dots x_1)))) \quad (6.2)$$

where T, x, y, h are the time sequences, input, output, and hidden layers. parameters.

6.8 Model Evaluation

For the MM, we observe the general pattern of the treatment outcome for each cycle and then compare it with the outcome distribution obtained from the data extraction. Fig. 6.8 shows both datasets plotted together. The dashed line represents the value obtained from the Markov chain. From that we get the steady-state probability after 5/6 cycles. The distribution obtained from the MM resembles the distribution obtained from the real data.

Once we train the model, we calculate the accuracy of the dataset. However, we also calculate the other metric to measure the performance of our models. We measure the performance of our classifier models by using several metrics (i.e. precision, recall, accuracy,

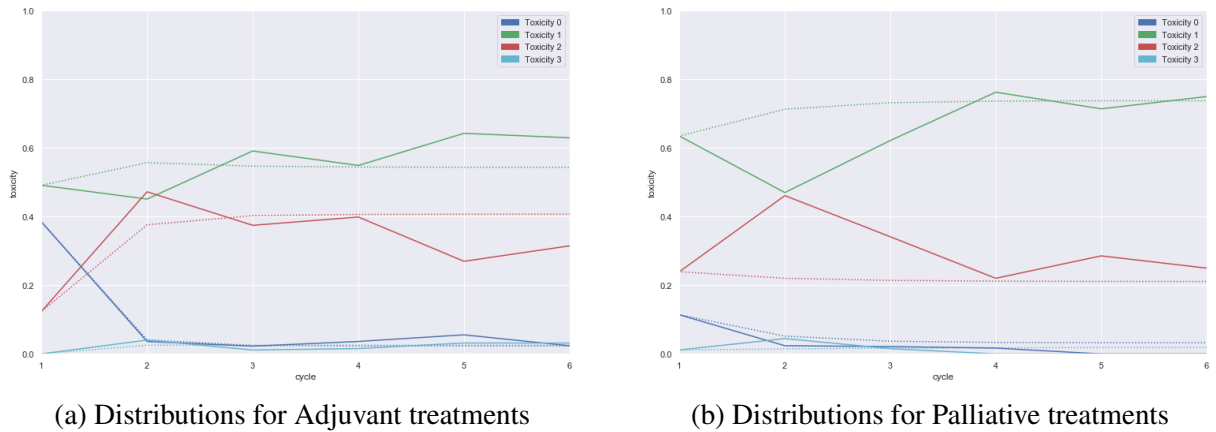


Fig. 6.8 The Distributions for Chemotherapy treatments

and f1-score) after performing the cross-validation test [15]. This allows us to observe the model with its false positives and false negative result. The precision measures the correctly identified positive cases from all the predicted positive cases. In our cases, correctly categorised the result of the treatment as high/low toxicity. The recall measures the correctly identified positive cases from all actual positive cases. Then, we used F1-score as a metric that combines both precision and recall. This set of metrics allows a better measurement than using accuracy alone.

Table 6.8 Model test result (mean-std)

Model	Regime	Accuracy	Precision	Recall	F1-score
RF	Adjuvant	0.81(+/-0.11)	T0:0.55(+/-0.50)	0.92(+/-0.32)	0.65(+/-0.46)
			T1:0.85(+/-0.15)	0.83(+/-0.09)	0.84(+/- 0.09)
			T2:0.82(+/-0.09)	0.78(+/-0.20)	0.80(+/-0.13)
			T3:0.57(+/-0.52)	0.83(+/-0.67)	0.67(+/-0.55)
	Neo-Adj	0.72(+/-0.09)	T0:0.53(+/-0.80)	0.60(+/-0.88)	0.52(+/-0.74)
			T1:0.77(+/-0.11)	0.80(+/-0.08)	0.79(+/-0.07)
			T2:0.63(+/-0.16)	0.61(+/-0.22)	0.62(+/-0.17)
			T3:0.33(+/-0.77)	0.23(+/-0.61)	0.23(+/-0.49)
	Palliative	0.78(+/-0.08)	T0:0.43(+/-0.23)	1.00(+/-0.00)	0.60(+/-0.24)
			T1:0.96(+/-0.03)	0.73(+/-0.09)	0.83(+/-0.06)
			T2:0.56(+/-0.17)	0.88(+/-0.10)	0.68(+/-0.15)
			T3:0.55(+/-0.81)	0.70(+/-0.92)	0.61(+/-0.83)
RNN	Adjuvant	0.85(+/-0.09)	T0:0.70(+/-0.22)	0.96(+/-0.08)	0.80(+/-0.15)
			T1:0.87(+/-0.11)	0.86(+/-0.14)	0.86(+/-0.10)

			T2:0.94(+/-0.10)	0.79(+/-0.16)	0.85(+/-0.11)
			T3:0.85(+/-0.64)	0.72(+/-0.63)	0.77(+/-0.61)
	Neo-Adj	0.81(+/-0.09)	T0:0.58(+/-0.35)	0.82(+/-0.25)	0.67(+/-0.31)
			T1:0.84(+/-0.10)	0.84(+/-0.11)	0.84(+/-0.09)
			T2:0.85(+/-0.17)	0.77(+/-0.12)	0.81(+/-0.13)
			T3:0.95(+/-0.30)	0.78(+/-0.47)	0.82(+/-0.34)
	Palliative	0.85(+/-0.09)	T0:0.67(+/-0.94)	0.24(+/-0.44)	0.33(+/-0.57)
			T1:0.85(+/-0.12)	0.94(+/-0.05)	0.89(+/-0.07)
			T2:0.83(+/-0.17)	0.75(+/-0.20)	0.79(+/-0.15)
			T3:0.53(+/-0.96)	0.56(+/-0.99)	0.54(+/-0.97)
HMM(c*)	Adjuvant	0.53(+/-0.00)	NA	NA	NA
HMM(m*)		0.70(+/-0.00)	NA	NA	NA
HMM(c*)	Neo-Adj	0.62(+/-0.00)	NA	NA	NA
HMM(m*)		0.70(+/-0.00)	NA	NA	NA
HMM(c*)	Palliative	0.4(+/-0.00)	NA	NA	NA
HMM(m*)		0.72(+/-0.00)	NA	NA	NA
		*c=corner			
		*m=middle			

We do not perform the cross-validation for the HMM because the performance measured with the splitting train-test method is much lower compared to the RF or RNN models. Table 6.8 shows the result of the evaluation for all classifier models.

We need more data to train the corner cases (i.e. initial and end of the treatments) for the HMM models. The accuracy for the corner cases is significantly lower than the middle/transition case because the dataset has more transition cases than the initial or end cases. Similarly, we can see the same characteristic for the F1-score for each treatment outcome. The T1 and T2 have higher F1-score compared to the extreme case, T0 and T3 because our datasets have more data with T1/T2 as its outcome. The RNN models outperform the RF models because, unlike RF, the RNN has LSTM units which allow the model to consider all the observations since the first treatment. Since our datasets are given as a time series, the previous treatments may affect the result of the current treatment. Hence, the RNN has an advantage compared to the RF that only considers the current state and limited information about the previous treatment result.

6.9 Summary

With our classifiers, we can predict the toxicity outcome of the chemotherapy treatments with around 0.8 - 0.85 accuracy. We observe that the RNN model performed better than all other models because it considers all the patients' treatments. Both RF and HMM have limited observation (one previous state). However, RF has an advantage because it does not differentiate between corner cases (the first and end of treatment) and the middle cases. Furthermore, the datasets for our RF models have a less class-imbalance problem than HMM. In comparison to the MM, the classifiers are more tailored for an individual patient. The MM shows the general pattern of the treatment while the classifiers can help predict the toxicity outcome of the patient. Both the MM and the classifiers are a complement to each other. Although our models have high performance, we can still improve their accuracy further if we have more data regarding the cancer characteristics or patients' comorbidity. In our datasets, the information regarding the state of cancer is limited. We lack the crucial information, such as tumour site, node and metastasis (TNM), *ER status*, *HER2 status* [175], about breast cancer in our data. Without this information, we cannot reliably recommend which regimes are better for different patients because we need both combinations of the toxicity outcome and cancer TNM to evaluate the treatment performance. For instance, some treatments might inhibit the growth of cancer better even though it may give higher toxicity to the patients in the short term. We believe that once we get and re-train the model with the new data extractions that contain more cancer characteristic and patients' comorbidities, our classifiers and MM distribution can be helpful to the physician as a second opinion to decide which regime is suitable for an individual patient.

Chapter 7

Patient Timeline Visualisation and Toxicity Predictor Integration

7.1 Introduction

As previously mentioned, our overall objective is to improve the delivery of healthcare within ECC at the NHS Lothian by mean of cancer data utilisation. After we train our machine learning models (Chapter 6), we integrate the models into our previously developed patient timeline visualisation (recall our *SESO Gateway* project on chapter 3) . Integrating these projects can be seen as adding new features to the original *SESO Gateway*. The reporting tool allows for more personalised treatment for the patients. This will enable the clinical team to act more quickly when complications arise. It can reduce the overall complications and likelihood of hospitalisation as a consequence. It may increase patients' well-being and the patients will likely feel more involved in their treatment plans. In this section, we describe the use case, implementation, evaluation, and summary of the integrated system.

7.2 Frameworks

Deployed AI platforms ship with bulky system architectures which present bottlenecks and a high risk of failure. A serverless deployment can mitigate these factors and provide a cost-effective, automatically scalable (up or down) and elastic real-time on-demand AI solution [32]. Serverless deployment allows us to focus more on business logic without thinking about how we are serving our application. This is a good alternative to serve our predictors. However, due to the policies within ECC at NHS Lothian regarding the limitation of using the cloud, we refrain from such solutions.

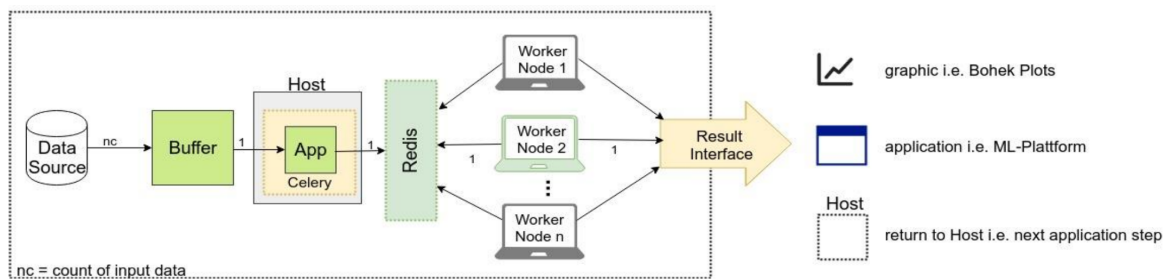


Fig. 7.1 Distributed Task Queue Architecture. Source: [161]

One of the solutions is to process our predictor's workload on the server asynchronously. Hence, we use a distributed task queue because it allows us to offload the work to another process asynchronously.

We use a third party library, *Celery* [29]) to implement a distributed task queue in our application (i.e. *SESO Gateway*). *Celery* streamlines the implement of tasks queue for many workers. As *celery* takes care of the application for us, we avoid reinventing the wheel as well as allowing us to focus further on developing the predictors and data visualisation module.

The architecture of a distributed task queue with *Celery* is shown in Fig. 7.1. The input data are messages sent from the client containing a special tag. The input data will be collected up to a predefined count of input data in the buffer. The collected data package will be sent to the host system, which only contains distribution components of *Celery* and the pre-processing application. However, the host will not transform the data. Instead, it will be passed into *Redis*. *Redis*, which acts as both message broker and a database, manages the *Celery*'s workers on several nodes. These worker nodes transform/process the data accordingly based on the functions/procedures which are defined in the application. If all sub-tasks are done, the result can be passed to the result-interface or stored in a database [161].

7.3 Use Case

We integrate the first project and the toxicity predictor by developing a dashboard. The application can help oncologists observe, monitor, and analyse the condition of their patients over time. Together with the oncologists from the Western General Hospital (WGH), we describe the scenario of the users of the system as follows.

Emma is a 38 years old patient in the WGH who has been diagnosed with breast cancer. To prevent the spreading of the tumour, she undergoes breast surgery. After her surgery, chemotherapy treatment is given as a follow-up to her surgery. She is now dismissed and only

visits the hospital for her chemotherapy appointments. To ensure her recovery (i.e. by being able to determine the correct given dose for her treatment), a treatment plan and regimen have been established (this will be over several months with treatment in the hospital every three weeks). Emma also has comorbidity. As any cancer patient on chemotherapy, she might have higher toxicity levels as a result, but it is crucial to guarantee that the scale does not go above level two. Based on the Chemotherapy Risk Assessment Scale for High-Age Patients (CRASH) score [48], toxicity levels range from 0 (no toxicity) to 5 (very high toxicity).

Via a user-friendly web application, Emma can provide information on symptoms daily throughout her treatment. Severe reported symptoms can be picked up by the clinical team and acted upon asap.

The conditions that are being monitored and provided by the patients are *nausea, vomiting, diarrhoea, constipation, oral mucositis, oesophagitis, neurotoxicity, hand/foot, skin, hypersensitivity, and fatigue*. With these conditions, the oncologist can determine the level of toxicity a patient may have.

The information Emma provided, data about patient characteristics, cancer information hospitalisation data, and information about comorbidities are combined. This combined data will help clinicians adapt treatments better to Emma as an individual patient which results in controlled toxicity levels and improved health outcomes. It uses data from several patients treated over the years with comparable characteristics.

If during a treatment there are signs that toxicity levels are high or that the condition of Emma is deteriorating, one of the members of the clinical team (e.g. oncologist, specialist consultant, nurse, GP) notes in the irregularities in Emma's data and contact Emma to intervene.

During a phone call, a decision is made for the GP/nurse to visit Emma at home and provide some additional medication to alleviate symptoms. Admission to hospital is not necessary. As scheduled, Emma comes to WGH for the next chemotherapy treatment. This procedure is iterative until the end of the chemotherapy treatment.

Overall, Emma can have more personalised treatment. If a complication arises, the clinical team can act more quickly. Furthermore, Emma well-being increases as she gets more involved in her treatment plans.

7.4 Implementation

In Chapter 6, we described several models developed to predict the toxicity outcome of chemotherapy treatments. We initially choose the *Kaplan-Meier* (KM) survival analysis, Markov Model (MM), and Random Forest (RF) models. We add the Markov model for

showing both the general condition and transition of a cohort of patients. We choose the cohort based on the intention and treatment regimen. As mentioned before, the RF performs better than the Hidden Markov Model (HMM), especially in predicting the toxicity outcome during the patients' treatments [155]. Although the Recurrent Neural Network (RNN) model performed better than the Random Forest model - mainly because of its capability to integrate all the previous treatments for predicting the treatment result - it is prone to over-fitting. This is due to the size of our dataset, as we presently do not have enough data to train the RNN models optimally [153].

For showing the capabilities of integrating machine learning modules with a patient timeline, the accuracy of our models and/or the quality of the data used is of no consequence. Our prototypical solutions reinforce the capabilities offered for improved cancer care, and if the data in the future improves, so does the benefits of our integrated dashboard solutions.

7.4.1 Front-End

We add several modules to both the front-end and back-end for integrating the toxicity predictor to the *SESO Gateway*. We explain the process as follow.

We create a new component to integrate the toxicity predictor the *SESO Gateway*. The component handles the feature to show the analytic result. In the component, there are several graphs to show the result of the toxicity prediction of the patient. Specifically, we use a line chart for the KM and MM models. For the RF, we use a bar chart. Figure 7.2 shows the charts in the component once rendered.

We add several services (i.e. *toxicity*, *treatment*) to manage the treatment data. Both services have the URLs to request the patient/toxicity data. Similar to the patient timeline visualisation, we then save the current patient's treatments in the services. The other service we add is the patient's analyser (i.e. *patientAnalyser*) service. This service handles the communication between the client and server when a user requests a treatment analysis.

Figure 7.3 shows the overall process for requesting and performing the toxicity prediction in the Front-End.

When a user requests a treatment analysis, the client performs a *POST* request containing the treatment detail in the body.

Once the server receives the request, it registers the request into a queue and response the request with a *response id*. The *response id* is used to retrieve the result of the treatment analysis once the server has finished analysing/processing the request. Later, the client performs a *GET* request to retrieve the treatment analysis result with the *response id* as its query parameter. To avoid server timeout due to possible heavy processing of the model to predict the result of the treatment, the client keeps on requesting periodically until it receives

a response from the server. Once the client has obtained the treatment analysis result, it re-renders the component to display the analytic result.

7.4.2 Back-End

When integrating the predictors into the *SESO Gateway*, we create a new endpoint to facilitate the request that will be made by the user. For this, we add two *django applications* for the analytical function, *ml_predictor* for our machine learning predictors and *survival_analysis* for the *Kaplan-Meier* estimator and Markov Model (MM).

In both applications, we have a *Predictor* class. The *Predictor* class consists of functions to load the machine learning/regression models, run the predictor, and convert the predictor output into an appropriate format that will be consumed by the client. Here, we serialised machine learning models. The serialised model is integrated into the newly added applications. During the start-up of the server, we de-serialise the machine learning model so that we can call the *fit* function when analysing the treatment.

Because the model analysis might take a longer time than a simple data retrieval from the database, we update the components of the *SESO Gateway* back-end. The integrated system has an asynchronous and periodic task handler as well as an internal database, which consists of the message broker and asynchronous requests' result storage, as shown in Figure 7.4. We then tag several of the Predictor's functions as celery tasks. We use a third party python library (i.e. *Celery* and *Redis*) to implement the asynchronous request handler.

When a client requests to analyse a treatment, the request is processed asynchronously by the asynchronous task handler as it might require a longer time to be processed (i.e. in comparison with requesting the patient's medical history). The handler puts the request in the tasks queue and tags it with a unique *task id*. The *task id* is later given to the user/ client as the response of the POST request. Once the *SESO Gateway* finishes predicting the result of the upcoming treatment, the outcome of the prediction is saved in the result storage which can be retrieved by the client via a GET request with the *task id* as the query parameter if the task is complete.

Below are the examples for both the *POST* and *GET* request:

- *POST* request url: `http://.../api/upcoming-treatment/`
- *POST* request body:

```
{  
  "age": 77,  
  "height": 1.47,
```

```

    "weight": 53.0,
    "sa":1.45,
    "intention":"Adjuvant",
    "regime":"PACLITAX WKLY",
    "totalCycles": 5,
    "totalDurations": 100,
    "previousToxicity": 2,
    "startDate": "14-03-2020"
  }

```

- *GET* request url: <http://.../api/upcoming-treatment/?id=72bb707c-60d6-43f6-b1ac>

7.5 Evaluation

To evaluate the *SESO Gateway*, we conducted different kinds of testing, organised by the level and purpose of the tests themselves. For the *SESO Gateway*, this included unit-testing, integration-testing and performance-testing.

Our unit tests are performed on the developed module to clarify if the method performs as expected in a set of conditions. Here, we focus on the unit testing for the *Predictor* class. To test the *Predictor* class, we create several mock objects to simulate the machine learning model.

For the *SESO Gateway* integration testing, we combine several modules within the application and test those modules as a group. The integration testing takes the modules that have been tested previously in the unit testing. We use the integration testing to simulate the process of the user/client requesting to analyse the upcoming treatment. Our integration testing focuses on examining the pathway from the first time the client makes a request for analysing the chemotherapy treatments and returning the right response for that request (i.e. *response id* and treatment outcome). Below are the coverage result of the unit and integration testing.

Name	Stmts	Miss	Cover

...			
ml_predictor/__init__.py	0	0	100%
ml_predictor/api.py	57	18	68%
ml_predictor/apps.py	3	3	0%
ml_predictor/migrations/0001_initial.py	6	0	100%

ml_predictor/migrations/__init__.py	0	0	100%
ml_predictor/mock_models.py	23	0	100%
ml_predictor/models.py	6	0	100%
ml_predictor/rf_models.py	160	38	76%
ml_predictor/tests.py	126	13	90%
ml_predictor/urls.py	3	0	100%
...			
...			
...			
survival_analysis/__init__.py	0	0	100%
survival_analysis/api.py	23	2	91%
survival_analysis/apps.py	3	3	0%
survival_analysis/migrations/__init__.py	0	0	100%
survival_analysis/models.py	1	0	100%
survival_analysis/tests.py	15	0	100%
survival_analysis/urls.py	3	0	100%
...			
...			

...			

For the testing, we estimate that between 0 to 150 users will access the application at the same time. Similar to the tests we previously conduct for the patients' timeline visualisation, we assume that there could be a time when all users try to access the application simultaneously. Based on this assumption, we need to perform a performance test to observe the robustness of our system.

One issue of *go-live* is performance, such as latency from highly sensitive and slow responses. The performance test allows us to simulate a situation where many users try to access our application at the same time. We use a third party library (i.e. *Gatling-tool* [56]) to perform such a performance test.

There is no standard default regarding connection timeout. The timeout is set depending on the expected complexity of the query/process. Unfortunately, we failed to observe or analyse the time for our machine to perform prediction. In the future, we would like to evaluate this and added a timeout for our request. When a timeout is triggered, we can report an error, set a retry policy for our application. Hence, for our performance testing, we use the default/assumed timeout provided by our performance testing framework.

Here, we perform the load testing for the our new API for the toxicity predictors. We deploy five *celery* workers for the load testing and perform simultaneous request for 300 users. Figure 7.5 shows the performance tests result.

The *SESO Gateway* is capable of handling 300 simultaneous requests from the user, as requested by the customer with less than 5% failure requests. We also test the *api* for the survival analysis. Here, we do not make the function to perform the survival analysis as a *celery* task. Because analysing the data to get the patient survival rate for the requested cohort is heavy, most of the requests took a longer time to be responded as shown in Figure 7.6.

However, there is no reason why it cannot process more requests. Similar to the patient timeline visualisation, the result of performance testing will be highly dependant on the server we access and the environment where we deploy our application.

7.6 Summary

Our primary goal is designing a method to accurately visualise the patient pathway data from the South-East Scotland Oncology Database. The users can view the pathway progress at an individual level (for example in clinic) as well as a cohort basis to analyse a treatment pathway against a set of metrics (e.g. time between treatment, following a particular pathway, outcome against disease management protocols). With *SESO Gateway*, we have a system which enhances observations for cancer patients in South East Scotland by incorporating a data visualisation tool with additional capabilities to analyse the upcoming treatments. The integration to the new database enables users to effectively see connections and patterns between patient treatments and their condition, in real-time. Finding these correlations among the data is necessary for improving the healthcare systems for *ECC*.

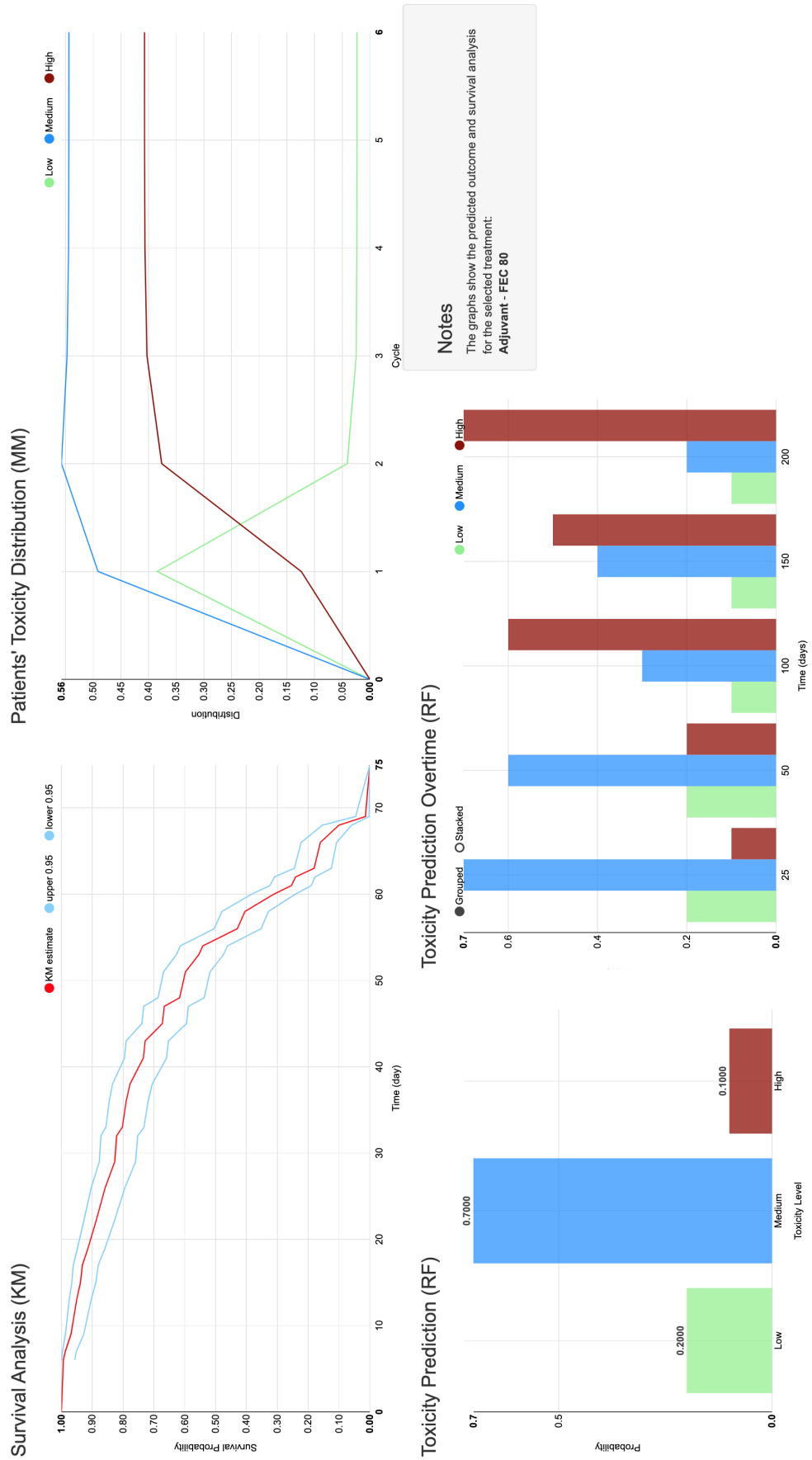


Fig. 7.2 Rendered model charts

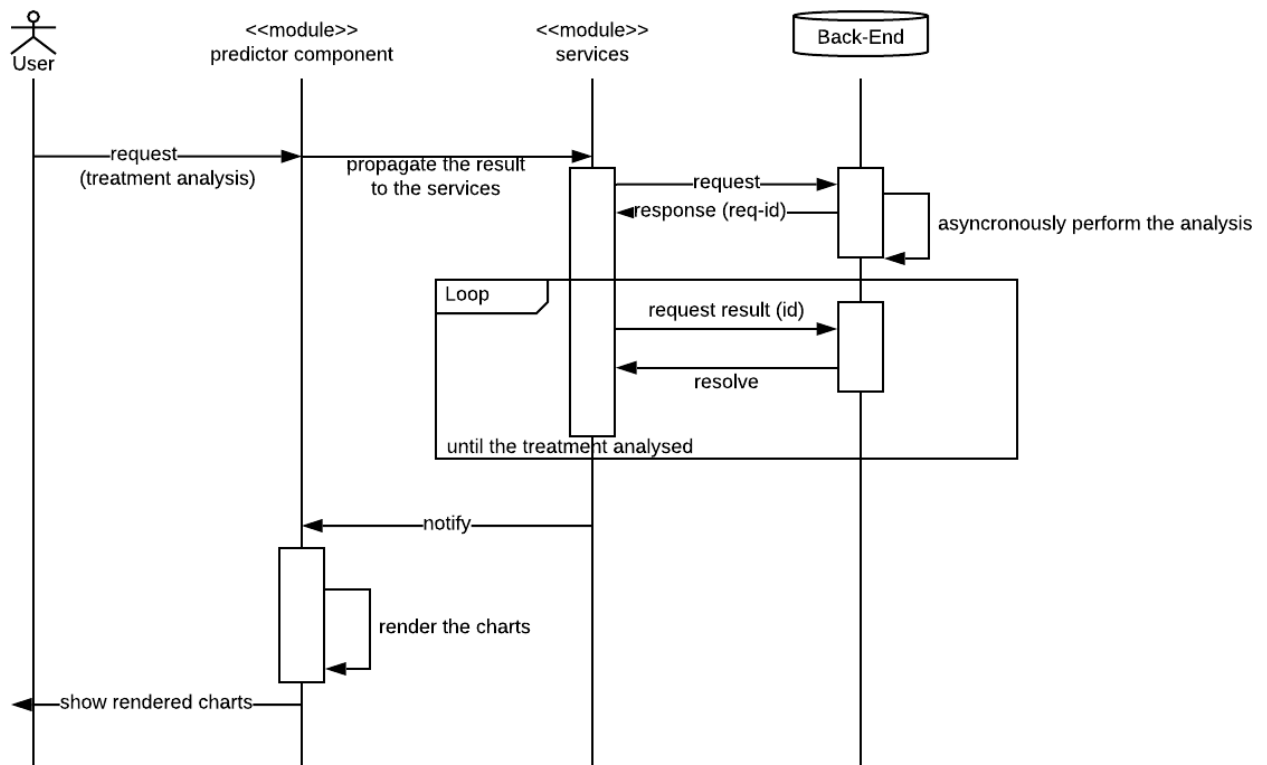


Fig. 7.3 Sequence diagram for treatment analyses

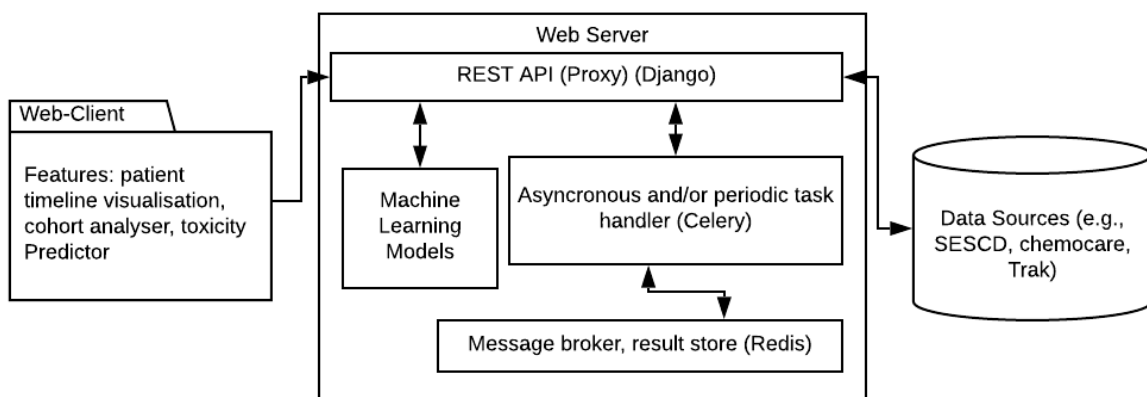


Fig. 7.4 SESO Gateway system architecture

Fig. 7.5 Predictors performance test result (with *celery*)Fig. 7.6 Predictors performance test result (without *celery*)

Chapter 8

Generating Synthetic Cancer Data

8.1 Introduction

Data is essential for machine learning, and to improve the accuracy of most machine learning models we need considerable amounts of data. Even though we have access to extractions from the National Health Service (NHS) Lothian database [95] for our projects, it is not easy to use them for developing accurate and reliable prediction models such as required for the toxicity prediction project of Chapter 6 and shown in [155]. In this chapter, we explore a different approach and use *synthetic data* to generate a large data set to train our model further and develop a proof-of-concept (POC) solution for our problem of predicting the toxicity of patients undergoing chemotherapy treatments. Furthermore, we discuss overall benefits to the use of synthetic data.

In general, one possible benefit of synthetic or *fabricated* data¹, is that its use may minimise the costs associated to data acquisition. To obtain more data in medical research, it is often necessary to conduct expensive and time consuming clinical trials. In addition, when looking at creating models for specific contexts, datasets may be too small to enable the creation of meaningful machine learning models. By using fabricated data, we can quickly generate more data with similar characteristics without additional cost. The use of fabricated data allows developers to continue improving their model without the need for a real/sensitive dataset, and could potentially already provide considerable insights. Once we collect enough real data, we can then retrain the model using the actual dataset. Nonetheless the use of synthetic data for these purposes require trust in how such data is generated. We elaborate on this later in the Chapter.

¹Here, we use both terms interchangeably to mean the same thing.

Most of our projects, as described in earlier chapters, we used small datasets with less than 1000 patients. Models trained on small datasets are more likely going to exhibit very specific patterns for that particular dataset which results in high variance. When using these particular dataset, our predictions are too dependent on the particular patients characteristics in our dataset, which would cause overfit to the data.

In addition, this often results in overfitting problems. One of the solutions for this problem is to extract more routinely collected clinical data. However, this solution has several challenges of its own, such as privacy concerns. Some patient data can be easily identifiable, is therefore sensitive, and might require further anonymisation to secure the process. Furthermore, data governance constrains the use of unconsented data and it is hence essential to check whether this applies, in other words, if some of the data might not have been consented by the data owner (the patient).

A different way to access healthcare data within NHS Lothian is through their data *safe-haven* [144]. A data *safe-haven* is a proxy between researchers/industrial partners with the data providers and data owners. It allows data to be shared in a secure way while maintaining the rules enforced by the governing board as shown in Figure 8.1. In other words, data owners and providers interact with the safe-havens directly, a process which is overseen by the governing body (this interaction is depicted as arrows on the right hand side). On the other side, researchers and industrial partners use the safe-haven, but do not need to know the details on data provenance and ownership. This approach is widely used in healthcare studies across the UK to ensure a process is followed before NHS data is used for research, and restrict what can be done, where and how.

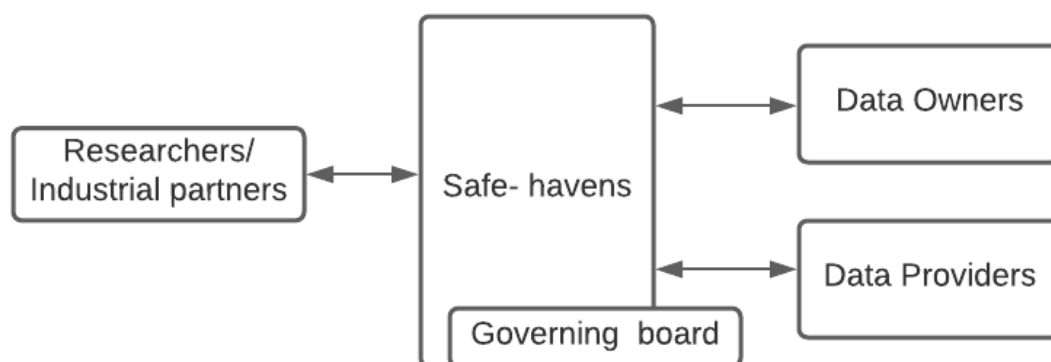


Fig. 8.1 Relations between stakeholders in a safe-haven context

In the context of the Edinburgh Cancer Centre, a *safe-haven* provides a secure network which allow access to a daily updated anonymous medical dataset (i.e. including national datasets through specialised local datasets) and is supported by trained NHS staff. Even though the general benefits of safe-havens are well understood, there are regional differences in how they operate and what can be done and used within a safe-haven for analysing data. In particular, the NHS Lothian safe-haven currently has limited support for the capability to analyse a dataset using state-of-the-art machine learning and/or deep learning techniques effectively, and are only equipped with limited statistical tools. One possible justification for this is that most medical researchers will use R and statistical packages, whereas data scientists with a computer science background would favour python and other environments.

Once we can generate the dataset quickly by fabricating the dataset, it allows the rapid development of extensive scale application for testing purpose. By having a large scale dataset, we can improve the model to resist over-fitting (e.g. by having more well-balanced dataset). Furthermore, the use of synthetic data allows us to simulate outlier events, such as rare diseases condition.

The challenges lie in the generation of fabricated data itself. We use IBM's Data Fabrication Platform (DFP) for doing so. The DFP works by providing a model, given as a set of rules, which describes the characteristics of the real data. Every solution to the model, in a constraint satisfactory sense, corresponds to a valid data instance. Running the DFP in search of valid solutions that satisfy all rules, we can generate as many data instances as required. Further detail will be given in section 8.4.

8.2 Related Work

There are several methods for generating fabricated data to solve the issues of using a real dataset and data availability. One of the techniques involves the use of existing dataset and imputing (i.e. assign (a value) to something by statistic inference) the values for the desired field. Rubin [142] proposed the idea of using multiple imputations for all the data-point in the datasets to generate a partial or full synthetic dataset. In statistics, imputation is the process of replacing missing data with substituted values. Given enough data and iterations, it is possible to generate a set of a synthetic dataset for specific purposes.

With the rise of machine learning in data mining fields, Rieter et al. [25, 136] extended the idea of using multiple imputers by using several machine learning models to generate the data synthetically. At the moment, there are many data synthesisers available with machine learning models, such as linear regression, random forest, decision tree, neural networks [37] as their backbone to generate either full or partial synthetic dataset.

However, there are caveats when using machine learning models to generate the dataset. First, we need to have a sufficient amount of data to have the machine infer the pattern in the dataset. Secondly, several machine learning frameworks for synthesising datasets cannot capture the notion of sequences in the dataset accurately. This dataset has several rows of data that tightly correlated. The machine learning techniques mentioned above (e.g. Random Forest, Decision tree) have no memory of the previous event. Therefore, to fabricate our dataset, we must use a solver because the chemotherapy treatment that we observe has a time-varying element.

8.3 Data Structure

Our source for performing the data fabrication is the dataset of the cancer patients within Lothian from 2014 to 2016. The data was extracted from the Scottish cancer registry (i.e. *SMR06*, *SMR01*, *Chalson*), National Records of Scotland (i.e. *Data on Deaths*), Oncology DCO database (i.e. *Demographics*, *Diagnosis*, *Surgery*), and *ChemoCare* (i.e. *chemocare_general* and *chemocare_toxicity*) as shown in Table 8.1.

Table 8.1 Source table brief overview

Table	Description
Standardised Mortality Ratio(SMR) 01[26]	General/Acute inpatient and daycase.
<i>SMR06</i> [26]	Information on Scottish residents when they are diagnosed with malignant or benign tumour.
<i>Charlson</i>	Information on patient's comorbidity [169] (i.e. the existence of one or more additional conditions coexisting with the primary disease) and their Charlson index [140].
<i>Deaths Data</i> [118]	Patient's death data
<i>Demographics</i>	Patient's demographics (e.g. date of birth, gender, ethnicity)
<i>Diagnosis</i>	General information regarding the patient's main diagnosis (e.g. stage, site)
<i>Surgery</i>	General information regarding cancer patient's surgery within ECC, NHS Lothian
<i>General</i>	General information regarding chemotherapy treatments

<i>Toxicity</i>	General information regarding the patient's conditions (or side effects) prior/after the chemotherapy treatments.
-----------------	---

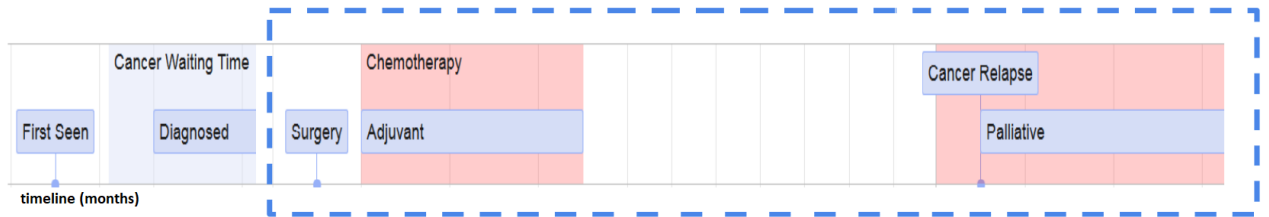


Fig. 8.2 The simplified pathway for cancer patients

Each patient may be given a series of different treatments, also known as treatment pathway, as shown in Figure 8.2. New patients, before every treatment, undergo several different sets of tests (e.g. MRI, CT SCAN) to determine the type of cancer and the first treatment to be given. The period before the treatment is given/chosen is called Cancer Waiting Times (CWT) [121]. There are several types of primary and follow-up treatments. However, for our first milestone, we only consider surgeries and chemotherapy treatments. The chemotherapy given within the NHS Lothian has general patterns that need to be taking into account when determining the rules. The patterns are as follow:

- A patient can only be treated with one *intention* (i.e. the purpose of the treatment, such as prolong the patients' live, cure cancer) at a time (e.g. adjuvant).
- After a specific time has passed, in the case of cancer relapsed, the patient might be given other treatment with different intentions (e.g. Palliative or Curative).
- Each intention has several different regimes
- Each regimen has several different drugs.
- The treatment may last for several weeks or months that is given in cycles. Hence, each regimen may have more than one cycles.
- A patient may be given several regimes at time.
- Some regimes may belong to one protocol, as shown in figure 8.3.

Table 8.2 shows how chemotherapy treatment is represented in the dataset.

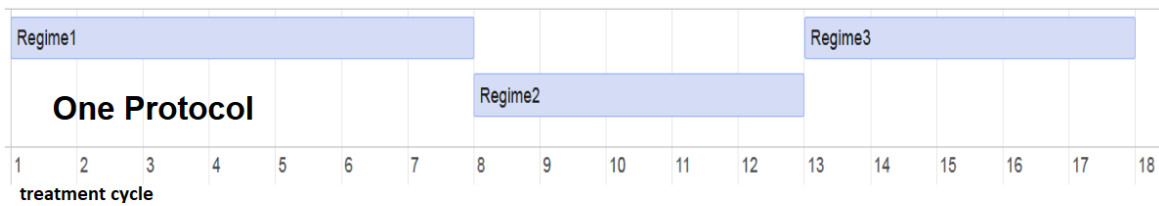


Fig. 8.3 Chemotherapy treatment protocol

Table 8.2 Chemotherapy treatment data representation

Chi	Appointment Date	Intention	Regime	Drug	Cycle
patient1	1/12/2019	Adjuvant	FED	drug1	1
patient1	1/12/2019	Adjuvant	FED	drug2	1
patient1	7/12/2019	Adjuvant	FED	drug1	2
patient1	7/12/2019	Adjuvant	FED	drug1	2
patient1	13/12/2019	Adjuvant	FED	drug1	3
patient1	13/12/2019	Adjuvant	FED	drug1	3

8.4 IBM's Data Fabrication Platform

We use IBM's Data Fabrication Platform (DFP) to generate the fabricated data. We chose to work with the IBM tool for fabricating our database, because the tool allows us to combine both information coming from domain knowledge (not necessarily apparent in a given data extraction) and the result of statistical inferences from the dataset. This is a powerful feature that enables us to create potentially more realistic data. IBM's Data Fabrication Platform (DFP) is a web-based central platform that provides a consistent and organisational-wide methodology (rule-guided fabrication) for generating high-quality data for testing, development, and training. Fabrication of synthetic data consists of two stages - data modelling and data generation [77].

In rule guided fabrication, the database logic can be extracted automatically and is augmented by application logic and testing logic modelled by the user. The application logic and the testing logic can be modelled using rules that the platform provides, but the users can also add new rules. The DFP has a fabrication engine embedded with CSP solver. The fabrication engine generates data by following user-defined rules. As the user, we define the rules, type

of data, the volume of data, the relationships among different columns in databases. The rule types include:

- Constraint rules: domains, mathematical functions, arithmetical relations, string relations, regulator expressions.
- Knowledge-based rules: chosen from existing data sources.
- Analytic rules: value and pattern distributions, smart classifications.
- Transformation rules: constraints describing relations between targets and sources, can be bundled to transform tuples.
- Programmatic rules: user-defined code/script functions that generate target values.

Once the user has defined the data sources and rules, the solution builds the fabrication task, maintaining the referential integrity of data based on database constraints or applied constraints. Here, the constraint is solved by the solver and the solution is used to construct the fabricated data [4]. The output can have multiple different formats/extensions. Figure 8.4 shows the flow of data fabrication with the IBM's Data Fabrication Platform.

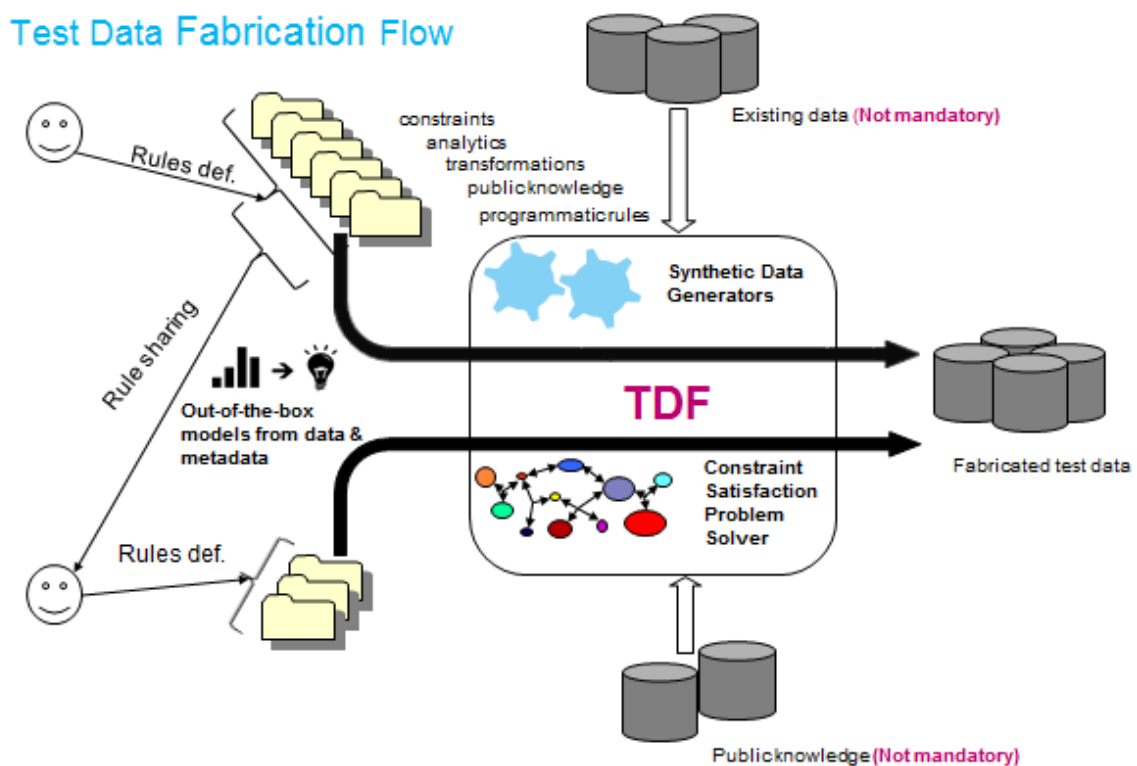


Fig. 8.4 Data fabrication flow. Source:[77]

Our task here is to define the chemocare rules used for the data fabrication flow. We first define the tables and columns we want to populate. We then specify the rules for each column. Some rules can be applied to several different columns or tables. With those rules, the IBM solver populates the tables while making sure that the Constraint Satisfactory Problem (CSP) defined by the rule is solved and satisfied.

8.5 Methodology

Motivated by those above, we use an IBM's DFP to tackle both issues. We specify the problem (i.e. fabricate data generation) by providing the constraints of the variables with domains, here the data fields and values. After we construct the constraints, the solver finds solutions by constraint propagation and search. By having the rules which capture the pattern of the real dataset, we can fabricate data despite the limitations like the availability of the data and the sequence nature of the chemotherapy treatment.

Here, each field associates with a constraint that follows several syntaxes specific to the solver. The syntaxes can express some conditions, such as unique, bounded, constant, and random values. We can also represent the dependencies between each field, including mathematical equations (e.g. $a = b + c$), order relationships (e.g. such as equal, less than, greater than), and other Boolean conditions. Lastly, we can express weighted/probability, normal, and random distribution to determine the value of our fields. Table 8.3 shows some of the common syntaxes and their descriptions.

Table 8.3 IBM Solver syntaxes

Syntax	Description
<i>randomBool</i>	returns <i>true</i> N times out of 100. N is the parameter.
<i>randomNumber</i>	returns a uniform random value in the range [0,N-1]. N is the first parameter.
<i>randomCover</i>	satisfies all the coverage model (second parameter) to the first parameter. If using within database fabrication, the coverage model may point to column of another table.
<i>randomWeightedNumber</i>	randomly returns the value using weights specified in the parameters.
<i>normalDistributionValue</i>	returns the value according to the mean and variance. The value is rounded to the closest represented value.
<i>allDiff</i>	satisfies that all the arithmetic expressions (specified by the last parameter) get different value.
<i>table</i>	returns a boolean value if the table has no return value. If the table returns a value, the returned format is according to the return value.
<i>monotonic</i>	satisfies that all arithmetic expressions have sequential monotonic values. The last parameter defines the different between two elements.

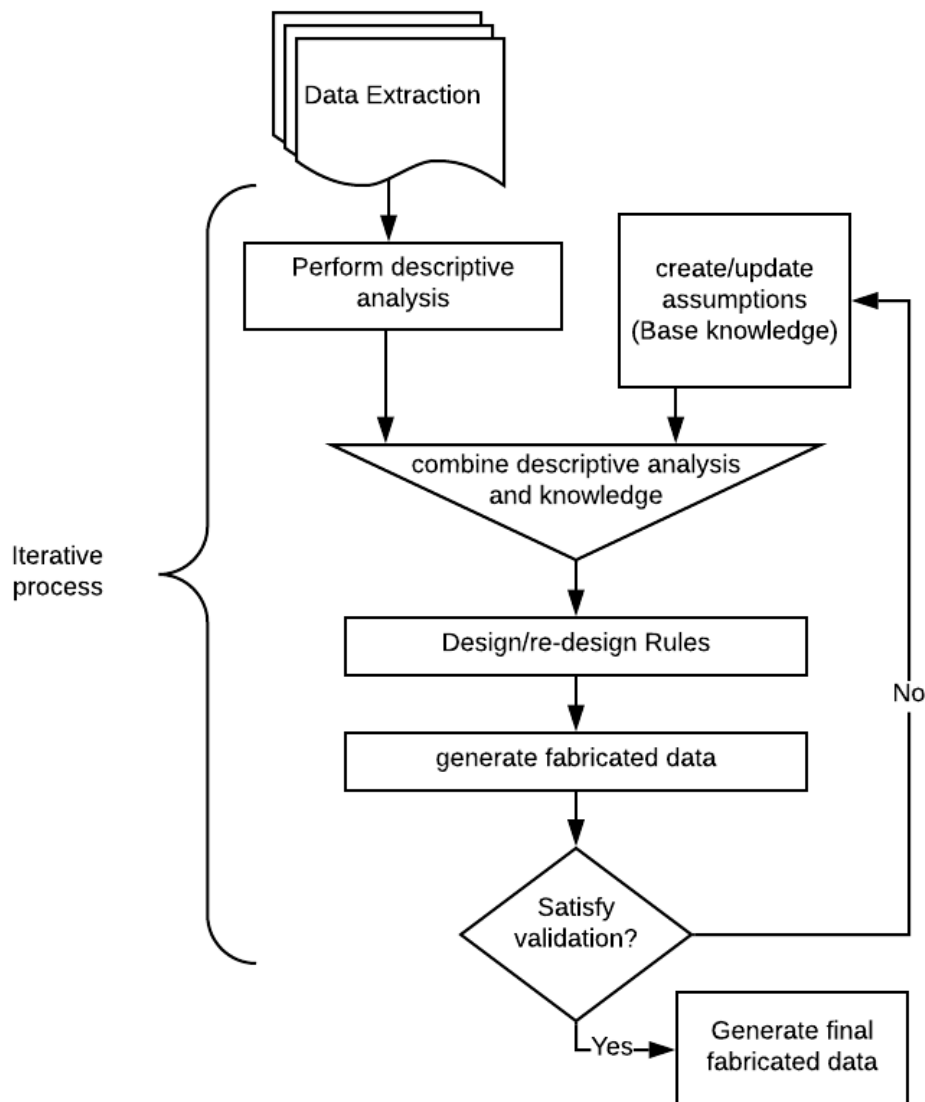


Fig. 8.5 Rules generation flowchart

Figure 8.5 shows the flowchart for defining the rule. We use both descriptive statistics and the knowledge from the oncologist to determine the rules for our dataset. Once we define the rules, we then generate the fabricated data using the solver and perform data validation. It is an iterative process, repeating until the result of the validation is satisfying. The syntax below shows the combination of both. The first part is the knowledge part (i.e. if cancer has metastasised into site *C34.9*, we set the *pulmonary_flag* as 1). If there is no value in the *metastasis*, we use weight distribution to set the value of *pulmonary_flag*. We infer the weight distribution from the data extraction.

```
general.pulmonary_flag = (
```

```
// Knowledge:
(general.metastasis1 == 'C34.9' || general.metastasis2 == 'C34.9'
|| general.metastasis3 == 'C34.9') ? 1 :
// From extraction:
randomWeightedValue(general.pulmonary_flag,
    1200? 0,
    120 ? 1
)
)
```

8.6 Rule Design

This section explains the method of formulating the rule for several different columns.² These methods can be applied to the other columns that are not specified here (i.e. shown in appendix)

When designing the rules, we start from the patients' general information (e.g. patients' demographics, diagnosis) rather than the patients' chemotherapy treatments. In Scotland, each patient has a unique identifier (i.e. Community Health Index (CHI)) [156].

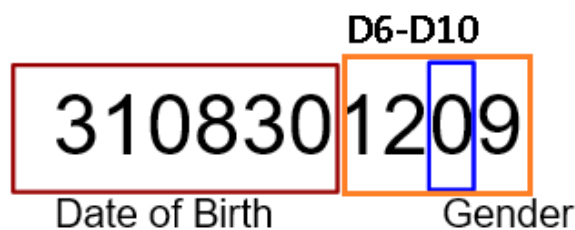


Fig. 8.6 CHI components

Figure 8.6 shows the component of the CHI. The current CHI number consists of the six-digit Date of Birth (*DDMMYY*) followed by a three-digit sequence number and a check digit. The ninth digit is always even for females and odd for males. To generate the proper CHI for each patient, we create several rules. The first rule specifies the uniqueness of the field:

```
allDiff(from(general), general.chi)
```

²All the rules we used for creating the synthetic data is available in the following link: <https://github.com/agasvina/DataFabrication/blob/main/serums.xml>

As previously mentioned, *allDiff* specifies that the CHI needs to be unique. Next, We create new fields/objects (D7, D8,D9,D10) to construct the CHI as shown below.

```
general.chi = concat(
  dateToString(general.DOB, DMy),
  intToString(general.D7),
  intToString(general.D8),
  intToString(general.D9),
  intToString(general.D10)
)
```

D7-8-10 follows specific constraint (e.g. $0 \leq \text{general.D7} \leq 9$). *D9* is use to indicate gender (i.e. *randomBool(99)* ? $\text{general.D9} = 0,2,4,6,8$: $\text{general.D9} = 1,3,5,7,9$)

The tool allows the use of equality-inequality relation for various data types, such as *date*. The rule below shows how we can manipulate the important events' date for the patients, for example, patients' incident date.

We specify the first incident date (i.e. the date when the patient is diagnosed) to occur approximately between 2014 and 2016 by using the *equality-inequality* relation:

```
currentDate-(6*365)<general.incidence_date<= currentDate-(4*365)
```

To generate a string with some characteristics, such as postcode, the solver allows the use of *regex*.

Some rules are consists of simple mathematical equation. Here, we use summation to populate the *Charlson Comorbidity* index.

```
general.charlson_score =
  general.cong_heart_fail_flag +
  general.dementia_flag +
  general.pulmonary_flag +
  general.con_tiss_disease_rheum_flag +
  general.diabetes_flag +
  general.para_hemiplegia_flag +
  general.renal_flag +
  general.liver_flag +
  general.AIDS_HIV_flag +
  general.cancer_flag
```

Some other features include assigning value to a column based on a constant field or other column. We assign a constant into field like site (i.e. *general.site = s'C50.9'*). The *C50.9* indicates breast cancer.

There are fields for which values influence the values of other fields (e.g. *metastasis1-2-3*). For these fields, we use implications to capture this condition.

```
(general.metastasis1 is Null -->> general.metastasis2 is Null)
->> general.metastasis3 is Null
```

To populate some fields, such as age or weight, we determine if we can use the normal distribution and add another correlation between several fields. First, we manually observe the pattern by using charts as shown in Figure 8.7.

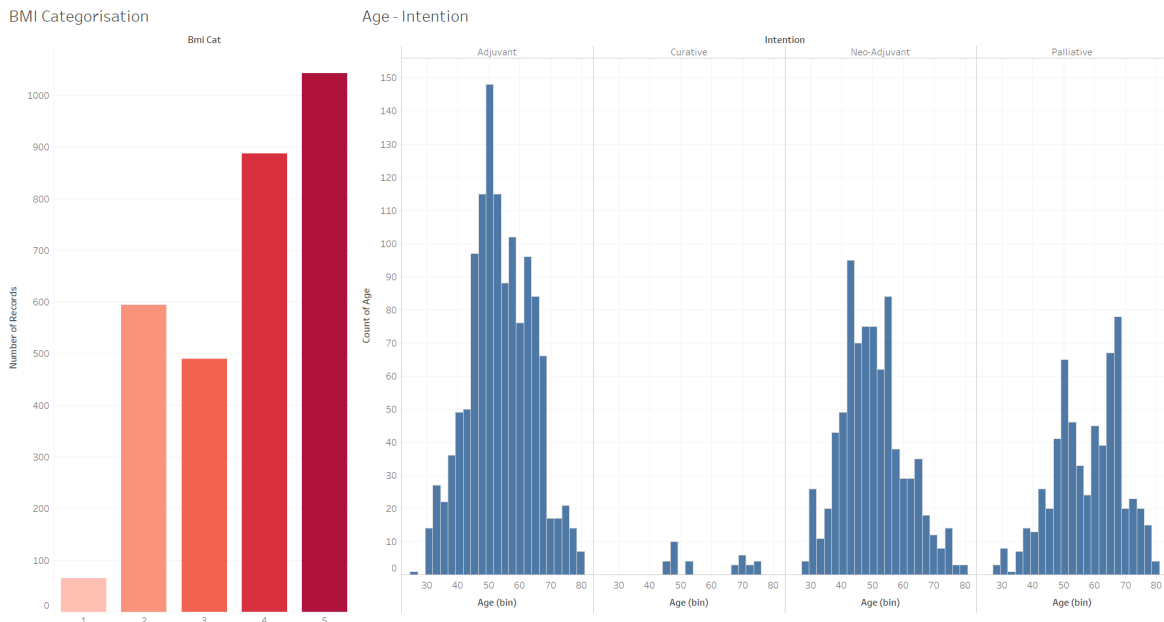


Fig. 8.7 BMI, Age-Intention distribution

For the BMI, we use the probability distribution to determine the BMI category (e.g. underweight, normal, overweight) and then we use the normal distribution to populate the exact BMI value for the patients for each category. As for *age*, we perform Shapiro-Wilk [151] test to test the normality of the patients' age (i.e. $W = 0.976$, and $p_value < 0.01$). The result indicates that the distribution slightly differs from the normal distribution. For our first milestone we use the normal distribution for *age* because the p_value is less than 0.05 and W is close to 1. The syntaxes for both fields are shown below. In the next iteration, we plan to add relation between age, bmi, intention and other fields as shown in Figure 8.7.

Some patients may have more than one admissions during their cancer care. Here, we create a new table for the patient admission and then we use the CHI as a reference/ foreign key to the general table. To fabricate this, we use the following steps. First, we specify the admission rate to fabricate the admission data. The rule is as follow:

```
numOf(from(smr01s), smr01s.chi = general.chi) =
  randomWeightedNumber(
    500 ? 1,
    300 ? 2,
    200 ? 0,
  )
```

With this rule, 50% of patients have one admission, 30% have two admissions and 20% patients have no admissions. Because the admission date is time based (sequential), we create another helper field, *elapsed_days*. The admission date depends on both incidence date and *elapsed_days*.

```
smr01s.admission_date = (smr01s.incidence_date + smr01s.elapsed_days)
```

The *elapsed_date* has a monotonic increasing value based on the patients' CHI as follow:

```
monotonic(from(smr01s), per(smr01s.chi), smr01s.elapsed_days,
  {normalDistributionNumber(110.4, 17.2)}, randomNumber(14,100))
```

The first value for patients' *elapsed_days* is populated using a normal distribution with 110.4 as the *mean* and 17.2 as the *variance*. The next instance of *elapsed_days* increases by a random number between 14 to 100 days. Because the admission date is calculated by adding the *elapsed_days* into *incidence_date*, its value is sequentially increasing as well. With this, we can fabricate the patient admission event.

The next dataset we fabricate is the patients' chemotherapy treatments. As described in the previous section, the main challenge of fabricating this dataset is capturing the relation between data that belong to the same patient. Briefly, a patient may have more than one intention, and each intention may have more than one regime. Each regime has more than one cycle and so on.

To capture this relation, we create five helper tables (i.e. *patients*, *intentions*, *regimes*, *cycles*, and *drugs*). There are similarities between these helper tables. We create *patients* as the reference point. We create *intentions* to model the condition where each patient may have one or more intentions. Similarly, *regimes* is created to model the condition where each

intention may have one or more regimes (i.e. *cycles* and *drugs* have the same purpose). Each helper table have foreign keys to each other (e.g. *patient_id*, *intention_id*).

The foreign keys are as shown in Table 8.4.

Table 8.4 Helper tables' foreign key

Table 1	Table 2	Foreign key
<i>general</i>	<i>patients</i>	<i>CHI</i>
<i>patients</i>	<i>intentions</i>	<i>patient_id</i>
<i>intentions</i>	<i>regimes</i>	<i>intention_id</i>
<i>regimes</i>	<i>cycles</i>	<i>regime_id</i>
<i>cycles</i>	<i>drugs</i>	<i>cycle_id</i>

The *patients* table has the patient demographic information during the treatment, with values assumed to be relatively constant, such as *CHI*, *height*, *hospital*, *tumour_group*. The *patients* table acts as the proxy to the *general* (i.e. *CHI* is used as the foreign key). The ratio between the data in the *patients* and *general* table is set to one. We also have the *first_intention* field in this table, used as the reference for populating the intention value. We use *randomWeightedValue* to populate this field. By counting the number of each intention occurring in the first cycle, we can get the weight value. The rule for the *first_intention* is shown below:

```
patients.first_intention = randomWeightedNumber(
  350? s'Adjuvant',
  200? s'Palliative',
  180? s'Neo-Adjuvant',
  15? s'Durable Remission',
  5? s'Curative'
)
```

The ratio between the *patients* and *intentions* tables are determined by the *first_intention* because some intentions may or may not have follow up treatments. We specify the *intentions.ratio* rule as follow:

```
numOf(from(intentions),
  intentions.patient_id = patients.patient_id)= (
  intentions.first_intention == 'Adjuvant' ?
    randomWeightedNumber(
      15? 2: 1
    ),
  intentions.first_intention == 'Durable Remission' ? 1,
  intentions.first_intention == 'Neo-adjuvant' ?
```

```

    randomWeightedNumber(
      60? 2: 1
    ),
    intentions.first_intention == 'Palliative' ? 1,
    intentions.first_intention == 'Curative' ? 1,
  )

```

The *first_intention* field determines the value of the next instance of *intention*. Similar to the *patients*, we have a field *first_regime*. The value of this field depends on *intentions.intention* and has the same function like the field *first_intention* (i.e. this method is repeated to capture the sequence behaviour for *cycles* and *drugs*).

To populate the treatment appointment date we use a similar rule (i.e. for the patients' admission) as mentioned before. We have the *appointment_date* field in *intentions* to populate the first *appointment_date* for each intention. In the *intentions* table we then set the elapsed days of based on the regime ratio, cycle ratio and regime interval days to prevent the overlap between appointment date for each regime.

In the regimes, we have another *elapsed_day* field to determine the date of the first regime. The starting date for the *regime.elapsed_day* is taken from the *regime.init_appointment_date*. The *regime.init_appointment_date* equals to *intention.appointment_date*. The rule for the *regime.elapsed_day* is as shown below:

```

monotonic (from (regimes), per(regime.intention_id),
  regime.elapsed_days,
  (cycle_ratio * regime_interval_days),
  {regime.init_appointment_date},
)

```

Unlike *intentions* and *regimes*, we have the value of *cycles' ratio* in the *regimes* because we need to know the number of cycles for determining the correct elapsed days between regime.

Lastly, to populate several fields like the toxicities outcome, *regimes*, and *performance status*, we integrate a simple Markov model into the rules (i.e. the value of the current fields depends only on its previous value). We use the previous value because we see a relatively high correlation (i.e. based on the Pearson standard of correlation [148]) between the previous value and the current value compare to the other fields. Table 8.5 shows an example of the calculated correlation between *performance_status* (*ps*) and other fields (i.e. *previous_performance_status* (*pps*), cycle, age, weight, intentions).

Table 8.5 Performance Status correlation

	intention	cycle	pps	ps	age	weight
intention	1	0.38	0.11	0.04	0.18	-0.16
cycle	0.38	1	0.05	-0.01	0.01	-0.08
pps	0.11	0.05	1	0.45	0.16	0.02
ps	0.04	-0.01	0.45	1	0.14	0.01
age	0.18	0.16	0.16	0.14	1	-0.02
weight	-0.16	-0.08	0.02	0.01	-0.02	1

8.7 Validation

Before we use the fabricated data to train the models and developed our systems, we need to determine the quality of the data as well as making sure that the generated data is a good representation of the real data. We use machine learning to evaluate the quality of the generated dataset. Machine learning provides a useful approach to verify the quality of the data. It enables us to find issues in the fabrication that is not obvious and easily spotted during the manual validation process. Furthermore, the results of this validation are used as feedback to improve and update the rule we use to generate the synthetic data.

8.7.1 Baseline Training Set

We take the initial real medical dataset to describe in the previous section and randomly label half of the dataset as *real* and half as *fabricated*. This dataset is used for calculating baselines measurements for determining the quality of the fabrication data. Because all the data for the baseline training is real, the tagging procedure produces a dataset that should have no exact distinguishing features between real and fabricated data. Hence, the machine learning should not be able to distinguish between real and fabricated data (i.e. The accuracy result should be 50% which is close to random guess). We also run the baseline training several times. For each run, we add more data than the previous one. We train the machine with a different number of data to observe the impact of the amount of data our machine learning performance. Table 8.6 shows the result of our baseline training for the diagnosis table.

Table 8.6 Baseline training accuracy result for Demographics table

Total data	Gini	entropy	AdaBoost
500	0.64	0.62	0.61
1000	0.56	0.55	0.54
2500	0.56	0.50	0.51
5000	0.52	0.51	0.53
10000	0.52	0.51	0.51
15000	0.51	0.52	0.51

Table 8.7 Individual table validation result

Table	Gini	Entropy	AdaBoost	Random Forest
Demographics	0.53	0.52	0.52	0.57
Diagnosis	0.53	0.52	0.52	0.58
smr01s	0.48	0.48	0.49	0.45
smr06s	0.42	0.42	0.46	0.39

8.7.2 Validation Result

To improve the quality of our data fabrication (i.e. 50% accuracy), we go through several iteration processes. We have several different models to validate the generated fabricated data (i.e. decision tree and AdaBoost classifier). When we use the decision tree classifier model, the model generates a tree from the input fields. From this tree, we can determine which field has the highest feature importance. By observing the nodes, we can determine the significant features and update the rule to improve the quality of our fabricated data. Figure 8.8 shows an example of the nodes.

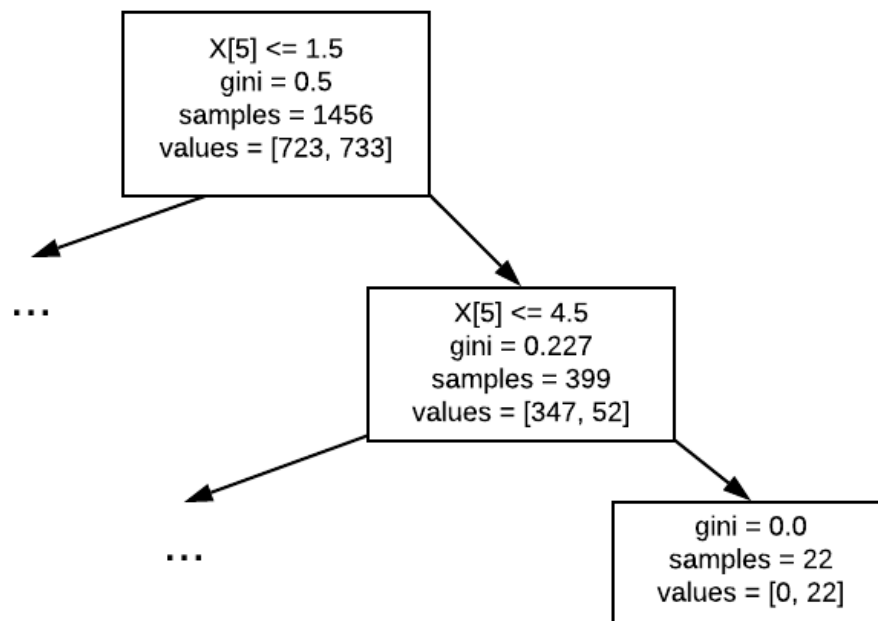


Fig. 8.8 Example of validation nodes

Here, feature 5 (i.e. feature 5 represents the *metastasisI*) is very significant. It is the first node in the trees, and a value greater than 4.5 indicates the data is fabricated (or real). So this value alone is enough to isolate 22 samples as completely fabricated (or real). Hence, by tracing through the graph, we can see which combinations of features lead to isolating groups of samples.

Due to the limitations of the validation tool when the thesis is written, we only perform the validation for the general dataset. The *ChemoCare* dataset has a time-sequential relation between each patient due to the cancer regimens.

We perform two different validations for the first iteration based on the number of data in both the real dataset and fabricated dataset. First, we use all data from both data sources. Because the number of data produced by the solver differs from the number of the real data, we select random data from the real data to match the total number of the fabricated data to avoid data-skewness. Here, we only use the data from 2014 to 2016. We then use all dataset from both data sources to perform the validation as we specify the number of data generated from the solver.

Table 8.8 shows the result of the validation for the first iteration.

Table 8.8 First iteration validation

Table	Gini	Entropy	AdaBoost
Demographics			
All	0.80	0.80	0.83
Selected	0.77	0.77	0.83
Diagnosis			
All	0.90	0.90	0.91
Selected	0.89	0.89	0.92
smr01s			
All	0.98	0.98	0.97
Selected	0.98	0.98	0.98
smr06s			
All	0.98	0.98	0.97
Selected	0.98	0.98	0.98
charlson			
All	0.99	0.99	1
Selected	0.98	0.98	0.99

Our validators perform well classifying the Charlson dataset. This is due to the knowledge we added to the rule (i.e. mainly the Cancer flag). All the patients in the dataset has *Cancer flag* equal to one. With this field alone, the machine can easily differentiate the dataset. Because the validation tool cannot integrate additional knowledge, for the second iteration onward, we do not include Charlson table.

We notice another issue of the validators. Several sensitive fields like general practitioners (GPs) and patients name can be easily used to differentiate between real and fabricated data. At the same time, we do not want to use real names as our knowledge-based to populate the fabricated data. An example is as follow.

Table 8.9 A mock example of general practitioners from the real data

GP name	No. of patients
Sobeck	60
Nightshade	40
Miller	40
Skywalker	40

Table 8.10 A mock example of general practitioners from the fabricated data

GP name	No. of patients
Valjean	60
Baggins	40
Kelmeckis	40
Potter	40

When we input the data shown in Table 8.9 and 8.10, the validator shows 100% accuracy and disregarded the value from the other fields due to the characteristic of a decision tree. The most significant field alone can be used to separate the dataset, and the validator will disregard the other fields. During the writing of this thesis, we ignore these fields and use another knowledge base dictionary for the name generation.

The result of the first iteration shows that the ML algorithm does not take into account if the input data is unbalanced/skewed. In the first generation, we have more real data than the fabricated data. When the input data is skewed, the machine has a tendency to have better accuracy, as shown in the *all* condition. For example, suppose we have 100 fabricated data and ten real data. If we only select ten fabricated data and ten real data as an input, all data pattern/weight ratio from the specified rules are not included. Here, the machine can easily distinguish fabricated data from real data. Similarly, if we use all 100 fabricated data and ten real data, even without learning, the machine can easily say all data is fabricated while still having high accuracy.

After the first iteration, we decide to re-select the real data sources and purposely generate fabricated data as much as the real data for the next iteration. By doing this, the amount of data fabricated by the solver is equal to the data source. Hence, we do not need to select random data for the validation. We want to make sure that the proportion of value for several fields is correctly defined in the updated rule.

Based on the result of the validation from the first iteration, we update several rules. We re-evaluate the correction between each field and update the rule accordingly. Here, we use the Pearson correlation. For several fields like the stage, we simplify the value by using a helper fields (i.e. 1,2,3,4 instead of 1A,1B,...1C) to easily calculate the correlation to the other fields.

Table 8.11 Second iteration validation

Table	Gini	Entropy	AdaBoost
Demographics	0.59	0.60	0.58
Diagnosis	0.90	0.90	0.92
smr01s	0.80	0.82	0.72
smr06s	0.87	0.88	0.71

The main purpose of the first iteration is to test and familiarise ourselves with the solver. We observe that all the generated data has the correct data types. We test all the data types that we can generate on the first iteration. Because there no issue with the data types, from the second iteration onward, we then gradually add more fields to be fabricated, as shown in Table 8.15.

There are several issues we have on the second iteration. Because of the different ratio between *demographics* and *smr06s* datasets, we separate the *smr06s* table and set the ratio with the following rule:

```
numOf(from(smr06s),smr06s.chi = general.chi) = randomWeightedNumber(
  3811 ? 0,
  1034 ? 1
)
```

From the rule above, more than half of the patients' will not have data in the *smr06s* table. Unfortunately, the solver does not support 0 value inside the *numOf* syntax. Hence, on the second iteration, the *smr06s* table has incorrect *CHI*. It becomes a separate entity without any connection to the general table.

We also have several issues with the character generation for several fields (i.e. *marital_status*, *ethnic_group*). Typos cause this. When we receive the result from IBM, those fields have characters we do not specify, like *#NAME*. Hence, we exclude those fields from the validation.

From the result of the second validation, we observe significant accuracy differences between the decision tree classifier and AdaBoost classifier models for both *smr01s* and *smr06s* tables. This might be caused by overfitting in the decision tree classifiers. Since there are no limits on the tree size during the process of data validation, the runs generate very large trees.

Before performing the third iteration, we test the validator by altering the percentage of training and testing dataset. From the plots shown both in Figure 8.9 and 8.10, the accuracy of the decision tree models fluctuates faster than the AdaBoost model. Unlike the decision

tree, the AdaBoost classifiers' accuracy converge to some value and do not have an issue with overfitting.

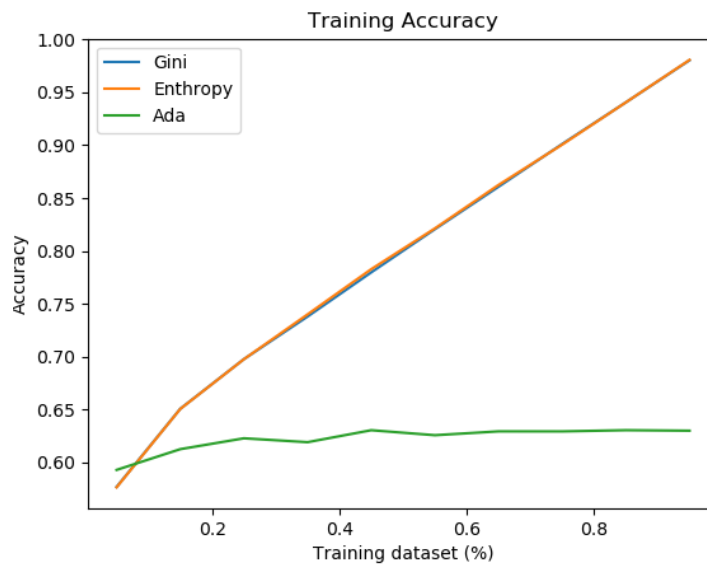


Fig. 8.9 Demographics accuracy

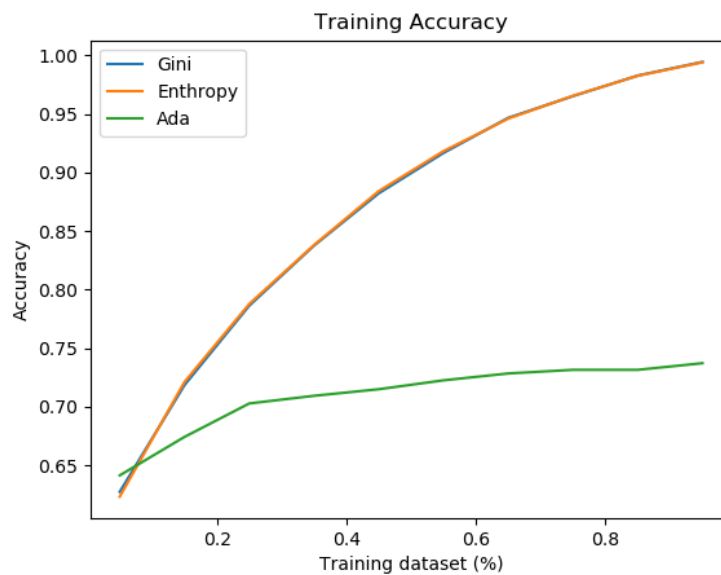


Fig. 8.10 Smr01s accuracy

We update the validators by limiting the amount of the tree generated by the decision tree classifiers and then re-perform the validation for both the first and second iteration, as shown in both Table 8.12 and 8.13. The validators still capture the issues with the fabricated dataset,

and there are less significant differences between the decision tree classifier models and the AdaBoost classifier models.

Table 8.12 First iteration re-validation

Table	Gini	Entropy	AdaBoost	Random Forest
Demographics				
All	0.82	0.82	0.83	0.82
Selected	0.81	0.81	0.83	0.83
Diagnosis				
All	0.91	0.91	0.92	0.93
Selected	0.90	0.90	0.90	0.90
smr01s				
All	1	0.99	0.99	1
Selected	0.98	0.97	0.99	0.99
smr06s				
All	0.94	0.94	0.95	0.95
Selected	0.93	0.93	0.95	0.94
charlson				
All	0.99	0.99	0.99	1
Selected	0.98	0.99	0.99	0.99

Table 8.13 Second iteration re-validation

Table	Gini	Entropy	AdaBoost	Random Forest
Demographics	0.62	0.63	0.59	0.63
Diagnosis	0.93	0.93	0.93	0.92
smr01s	0.78	0.78	0.73	0.78
smr06s	0.77	0.76	0.71	0.80

Overall, we see improvement by updating the rule based on the validation result. On the third iteration, we perform the same procedure: updating some rules and adding more fields, as shown in Table 8.15.

After we update the rule and fix several issues from the second data fabrication iteration, we generate another synthetic data and perform another validation, as shown in Table 8.14. Here, we observe a slight improvement in the Demographics table after we update the Histology weight distribution. As for the Demographics, we do not see much improvement

because we keep on adding more fields for each iteration. The last field we add to the Demographics is the patient date of death.

Table 8.14 Third iteration validation

Table	Gini	Entropy	AdaBoost	Random Forest
Demographics	0.63	0.63	0.60	0.63
Diagnosis	0.60	0.59	0.58	0.68
smr01s	0.72	0.72	0.72	0.74
smr06s	0.85	0.83	0.80	0.86

We see further improvement in the quality of our data fabrication on the third iteration (i.e. the accuracy of the validation result keep decreasing). We can improve the data fabrication by adding more complexity and relation between each field. Here, we also ask the feedback of healthcare professional from Edinburgh Cancer Centre. They immediately notice the invalid postcode from our dataset because we use regex to generate UK postcode. We can use the knowledge table to avoid the issue with postcodes. However, we refrain from doing to avoid using real sensitive data. In the future, we might generate real postcode into our fabricated dataset if we want to observe how location affects the patients' condition. As for the clinical validity, according to one of the oncologists, generally, the data looks ok. However, In smr06s table, we have a lot of high T and high N and more metastasis than they would usually expect. There is an implausible discrepancy between size and T stage. To fix this, we need to add more knowledge to the rules in the smr06s table. However, by integrating more knowledge (e.g. from American Joint Committee on Cancer (AJCC) for cancer staging), it may decrease the quality of fabricated data according to the validators due to the limitation of the validators and the invalid data in our data sources.

Table 8.15 List of validated fields

Table	Column	1st	2nd	3rd
Demographics	chi	✓	✓	✓
	Date of birth	✓	✓	✓
	name			✓
	postcode	✓	✓	✓
	age	✓	✓	✓
	gender	✓	✓	✓
	civil st.		✓	✓
	religion		✓	✓

	hospital date of death		✓	✓
Diagnosis	chi primary first seen date age site side histology stage tnm (t) tnm (n) tnm (m) performance status metastasis site	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
SMR01s	chi marital status ethnic group waiting list type admission date discharge date main condition other condition main operation	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
SMR06s	chi incidence date er status her2 status stage clinical t stage clinical n stage clinical m number positive tumour size	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
Charlson	chi postcode simd	✓ ✓ ✓		

incidence date	✓		
heart fail flag	✓		
dementia flag	✓		
pulmonary flag	✓		
con tis flag	✓		
diabetes flag	✓		
para hemiplegia flag	✓		
renal flag	✓		
liver flag	✓		
aids hiv flag	✓		
cancer flag	✓		
Charlson score	✓		

8.8 Summary

We managed to validate the data for the patient's diagnosis as shown in the previous section. Unfortunately, due to *COVID-19*, we had no access to the *ChemoCare* dataset. For the chemotherapy dataset, we only generated the data without doing the comparison between the real and fabricated dataset.

Though the use of synthetic data can solve several issues that emerge during the development of the system, it is not a perfect replacement as it comes with its caveats. For one, we know the model might have the same biases with the dataset, such as a mismatch between the way the machine learns and the way doctors works and treat the patients. Fabricated data can excel our machine/system development (e.g. scaling, exploration of various techniques, and over-fitting test). We still need to perform several iterations and adjustment to both rules and validators. With the generated data, we can continue to develop our systems and then re-train and re-adjust the system once we obtain enough real data.

8.8.1 Coronavirus Remark

Due to data access issues during *COVID-19* [34], we were not able to conduct further comparisons between the real and fabricated data. Initially, we would like to extract more *ChemoCare* data extraction and update the rule further. Next, we would like to retrain the model developed in chapter 4 with both fabricated and real data. We would like to both real dataset and fabricated to validate the models. With this, we can have more observations regarding the quality of our fabricated dataset.

Chapter 9

Conclusion and Future Work

We developed several different projects with the shared goal to contribute to facilitating better analysis and management of cancer data within NHS Lothian. We used various techniques from within computer science, software engineering and artificial intelligence to show how they can contribute in a novel way to improve cancer care. This EngD tackled projects for patient timeline visualisation, cohort based visualisation, database migration, analysing cancer waiting time, toxicity predictor and investigated the generation of fabricated data. Throughout this project, we used software engineering techniques, as well as simulation, machine learning and constraint based programming.

First, we created the patient timeline visualisation. We then performed database migration and then working on cancer waiting time. The cancer waiting time project covers the analysis of the data during the first phase of cancer treatment. We managed to identify the bottleneck of the process. By using the DES model, we can test various scenarios.

Next, we integrated our projects. We first integrated our timeline visualisation with the toxicity predictor. In the future, we would like to add more features to the integrated dashboard as well as using the newly migrated database. We observe some features that can be improved and implemented. The features are derived from various user feedback as well as the project scope itself. The details are as follows.

- **Add more cohort summaries and filters** The SESO Gateway will keep expanding as we get new data extractions. According to the user feedback, we should implement additional cohort summaries that we have not considered before (e.g. BMI proportion, adding a filter based on the oncologists in charge of the treatment, adding the functionality to get the list of patients). In doing so, we need to make sure that it is also integrated well with the previous features that have been implemented.

- **Integrating security to the application** It is necessary to adequately protect our application because of the type of data used as an input. We need to implement proper data protection (i.e., this should be similar to the security procedures implemented within NHS) before deploying the SESO Gateway to the network. These added security protections can prevent cyberattacks by ensuring patient data is protected to prevent fraud.

Several security aspects that we can consider are listed below [77].

- use JWT tokens or similar technology to allow signature when we securely transmit information between parties.
 - use blockchain as a robust digital ledger to log the activity of the users.
 - use data encryption and decryption.
- **Use a real database and deploy the application to the network** Once the DCO migration finished, we can update our application by changing its data sources. We need to update the SQL-Statement used to retrieve the data and obtain permission from the NHS to access the migrated database. If the NHS also provides the API, we need to update the web-client and update the client requests accordingly.

One of our objectives was to be able to give a second opinion regarding the upcoming treatments. Here, we can achieve this by improving our toxicity predictor. By using synthetic data, we can further develop our predictor and explore more advanced techniques as a proof of concept.

Lastly, more access to data is essential for improving our systems further. Our last project specified rules to capture cancer data and hence be able to generate a large amount of synthetic data, and be able to strengthen machine learnings models for further insights. We believe that synthetic data can help us explore more varied techniques for better analysis and hence contribute to improving our integrated system.

In the future, we can expand our range of techniques for different kinds of cancer treatments (e.g. radiotherapy) as well as different types of cancer. By integrating several projects, it is possible to have a digital solution to observe and analyse the whole patient cancer care journey. If implemented and adapted, these solutions can improve the overall cancer care within the ECC at NHS Lothian.

9.1 Contributions

During our EngD programme, we have contributed expertise and capacity to the Cancer Information Team specifically on the development of the South East Scotland Cancer Database upgrade and migration of legacy data onto the new platform (Chapter 4). Further, the Edinburgh Cancer Information Team has developed an understanding of how an industrial placement of a software engineering student can promote innovative design in our data infrastructure solutions (Chapter 3 and Chapter 7). Lastly, we have developed prototype software solutions that will provide a strong basis for further development of the innovative use of data for improved patient care within NHS Lothian (Chapter 5, Chapter 6, and Chapter 8).

References

- [1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems.
- [2] Abbas, O. A. (2008). Comparisons between data clustering algorithms. *International Arab Journal of Information Technology (IAJIT)*, 5(3).
- [3] Abo-Hamad, W. and Arisha, A. (2013). Simulation-based framework to improve patient experience in an emergency department. *European Journal of Operational Research*, 224(1):154 – 166.
- [4] Adir, A., Levy, R., and Salman, T. (2011). Dynamic test data generation for data intensive applications. In *Haifa Verification Conference*, pages 219–233. Springer.
- [5] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.
- [6] Alchin, M., Kaplan-Moss, J., and Vilches, G. (2013). *Pro Django*, volume 2. Springer.
- [7] Allen M., Spencer A., G. A. (2015). *What is discrete event simulation, and why use it?* NIHR Journals Library, Southampton, UK.
- [8] Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.
- [9] Andras, V. (2005). Discrete event simulation system. *Omnet++*.
- [10] Angular (2019). Angular: Introduction to the Angular Docs. URL: <https://angular.io/> [accessed: 2019-23-09].
- [11] Angular-nvD3 (2017). An AngularJS directive for NVD3. URL: <http://krispo.github.io/angular-nvd3/> [accessed: 2017-17-08].
- [12] ANGULARJS (2019). AngularJS: Developer Guide: Data binding. URL: <https://docs.angularjs.org/guide/databinding> [accessed: 2020-16-03].
- [13] Arora, N., Batham, N., Madan, S., and Hans, R. (2011). Analysis on Extract, Transform and Load (ETL). In *Proceedings of the 5th National Conference; INDIACom-2011*.

- [14] ASP.NET (2020). Open-source web framework for .NET. URL: <https://dotnet.microsoft.com/apps/aspnet> [accessed: 2020-16-03].
- [15] Aurelien, G. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. OReilly Media.
- [16] awsML (2019). Machine learning concepts. URL: <http://docs.aws.amazon.com/machine-learning/latest/dg/machine-learning-concepts.html> [accessed: 2019-12-10].
- [17] Babashov, V., Aivas, I., Begen, M., Cao, J., Rodrigues, G., D'Souza, D., and Zaric, G. (2017). Reducing patient wait times for radiation therapy and improving treatment planning: A discrete-event simulation model. *Clinical Oncology*, 29.
- [18] Bangsow, S. (2016). *Use Cases of Discrete Event Simulation: Appliance and Research*. Springer Publishing Company, Incorporated.
- [19] Banks, S. M. and Pandiani, J. A. (2001). Probabilistic population estimation of the size and overlap of data sets based on date of birth. *Statistics in Medicine*, 20(9-10):1421–1430.
- [20] Baril, C., Gascon, V., Miller, J., and Bounhol, C. (2017). The importance of considering resource's tasks when modeling healthcare services with discrete-event simulation: an approach using work sampling method. *Journal of Simulation*, 11(2):103–114.
- [21] Barzu, A.-P., Barbulescu, M., and Carabas, M. (2017). Horizontal scalability towards server performance improvement. In *2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, pages 1–6. IEEE.
- [22] Bootstrap (2020). The most popular HTML, CSS, and JS library in the world. URL: <https://getbootstrap.com/> [accessed: 2020-16-03].
- [23] Brailsford, S. C., Potts, C. N., and Smith, B. M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119(3):557–581.
- [24] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.
- [25] Caiola, G. and Reiter, J. P. (2010). Random forests for generating partially synthetic, categorical data. *Transactions on Data Privacy*, 3:27–42.
- [26] Cancer Registry (2019). Scottish cancer registry. URL: <https://www.isdscotland.org/Health-Topics/Cancer/Scottish-Cancer-Registry/> [accessed: 2019-12-10].
- [27] Carlisle, S. (2018). Software: Tableau and microsoft power bi. *Technology| Architecture+ Design*, 2(2):256–259.
- [28] Cassandras, C. G. and Lafortune, S. (1999). *Introduction to Discrete-Event Simulation*, pages 591–661. Springer US, Boston, MA.
- [29] Celery (2020). Distributed Task Queue. URL: <https://docs.celeryproject.org/en/stable/> [accessed: 2020-10-03].
- [30] chemoSideEffects (2019). Chemotherapy - Side effects - NHS. URL: <https://www.nhs.uk/conditions/chemotherapy/side-effects/> [accessed: 2018-27-08].

- [31] Chen, H., Yang, B., Liu, J., and Liu, D.-Y. (2011). A support vector machine classifier with rough set-based feature selection for breast cancer diagnosis. *Expert Systems with Applications*, 38:9014–9022.
- [32] Christidis, A., Davies, R., and Moschoyiannis, S. (2019). Serving machine learning workloads in resource constrained environments: a serverless deployment example. In *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)*, pages 55–63.
- [33] Clippinger, R. F. (1962). COBOL. *The Computer Journal*, 5(3):177–180.
- [34] COVID (2021). Coronavirus (COVID-19) NHS. URL: <https://www.nhs.uk/conditions/coronavirus-covid-19/> [accessed: 2021-10-02].
- [35] Croarkin, C., Tobias, P., Filliben, J., Hembree, B., Guthrie, W., et al. (2006). Nist/sematech e-handbook of statistical methods. *NIST/SEMATECH, July*. Available online: <http://www.itl.nist.gov/div898/handbook>.
- [36] D3 (2020). Data-Driven Documents. URL: <https://d3js.org/> [accessed: 2020-10-03].
- [37] Dandekar, A., Zen, R. A. M., and Bressan, S. (2017). Comparative evaluation of synthetic data generation methods. In *Deep Learning Security Workshop Singapore*.
- [38] Davidson-Pilon, C. (2019). lifelines: survival analysis in python. *Journal of Open Source Software*, 4:1317.
- [39] Der, G. and Everitt, B. S. (2008). *A handbook of statistical analyses using SAS*. CRC Press.
- [40] Di Girolamo, C., Walters, S., Gildea, C., Benitez Majano, S., Rachet, B., and Morris, M. (2018). Can we assess Cancer Waiting Time targets with cancer survival? A population-based study of individually linked data from the National Cancer Waiting Times monitoring dataset in England, 2009-2013. *PLoS ONE*, 13(8):e0201288.
- [41] django (2019). The Web framework for perfectionists with deadlines. URL: <https://www.djangoproject.com/> [accessed: 2019-23-09].
- [42] Django (2020). Writing your first Django app. URL: <https://docs.djangoproject.com/en/3.0/intro/tutorial01/> [accessed: 2020-16-03].
- [43] Django REST Framework (2020). Django REST Framework. URL: <https://www.django-rest-framework.org/> [accessed: 2020-16-03].
- [44] Dua, S., Acharya, U. R., and Dua, P. (2014). *Machine learning in healthcare informatics*, volume 56. Springer.
- [45] Efron, B. (1988). Logistic regression, survival analysis, and the kaplan-meier curve. *Journal of the American statistical Association*, 83(402):414–425.
- [46] eHealth (2020). Digital Care for Scotland. URL: <https://www.ehealth.scot/> [accessed: 2020-16-03].

- [47] Eisner, J. (2002). An interactive spreadsheet for teaching the forward-backward algorithm. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, pages 10–18.
- [48] Extermann, M., Boler, I., Reich, R. R., Lyman, G. H., Brown, R. H., DeFelice, J., Levine, R. M., Lubiner, E. T., Reyes, P., Schreiber III, F. J., et al. (2012). Predicting the risk of chemotherapy toxicity in older patients: The chemotherapy risk assessment scale for high-age patients (crash) score. *Cancer*, 118(13):3377–3386.
- [49] Fielding, R. (2000a). *Representational state transfer (REST). CHAPTER 3 Network-based Architectural Styles*. PhD thesis, Ph. D. Thesis, University of California, Irvine, CA.
- [50] Fielding, R. (2000b). *Representational state transfer (REST). Chapter 5 in Architectural Styles and the Design of Networkbased Software Architectures*. PhD thesis, Ph. D. Thesis, University of California, Irvine, CA.
- [51] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext transfer protocol–http/1.1.
- [52] Fisher, M., Ellis, J., and Bruce, J. (2003). *JDBC API tutorial and reference*. Addison-Wesley Professional.
- [53] Freeman, A. (2014). *Putting AngularJS in Context*, pages 45–54. Apress, Berkeley, CA.
- [54] Friedman, J., Hastie, T., and Tibshirani, R. (2001a). *The elements of statistical learning*, volume 1. Springer series in statistics New York.
- [55] Friedman, J., Hastie, T., and Tibshirani, R. (2001b). *The elements of statistical learning*, volume 1. Springer series in statistics New York.
- [56] Gatling (2019). Gatling Open-Source Load Testing. URL: <https://gatling.io/> [accessed: 2019-23-09].
- [57] GEV (2020). Generalized Extreme Value distribution and calculation of Return value. URL: <https://gmao.gsfc.nasa.gov/research/subseasonal/atlas/GEV-RV-html/GEV-RV-description.html> [accessed: 2020-18-03].
- [58] Ghahramani, Z. (2003). Unsupervised learning. In *Summer School on Machine Learning*, pages 72–112. Springer.
- [59] Goel, M. K., Khanna, P., and Kishore, J. (2010). Understanding survival analysis: Kaplan-meier estimate. *International journal of Ayurveda research*, 1(4):274.
- [60] Google Developers (2020). Constraint Optimization. URL: <https://developers.google.com/optimization/cp> [accessed: 2020-30-03].
- [61] googleHealth (2019). Deep Learning for Electronic Health Records. URL: <https://ai.googleblog.com/2018/05/deep-learning-for-electronic-health.html> [accessed: 2019-18-11].

- [62] Gorman, K., Hirt, A., Noderer, D., Rowland-Jones, J., Sirpal, A., Ryan, D., and Woody, B. (2019). *Introducing Microsoft SQL Server 2019*. Packt Publishing.
- [63] gradle (2019). Gradle User Manual. URL: <https://docs.gradle.org/current/userguide/userguide.html> [accessed: 2019-29-10].
- [64] Gradle (2020). Developing Custom Gradle Task Types. URL: https://docs.gradle.org/current/userguide/custom_tasks.html [accessed: 2020-10-03].
- [65] Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385.
- [66] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2016). Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.
- [67] Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.
- [68] Hadjigeorgiou, C. et al. (2013). Rdbms vs nosql: Performance and scaling comparison. *MSc in High*.
- [69] Havlicek, L. L. and Peterson, N. L. (1976). Robustness of the pearson correlation against violations of assumptions. *Perceptual and Motor Skills*, 43(3_suppl):1319–1334.
- [70] Hendler, J. (2014). Data integration for heterogenous datasets. *Big Data*, 2(4):205–215. PMID: 25553272.
- [71] Heydt, M. (2015). *D3.js by Example*. Packt Publishing Ltd.
- [72] Hu, X., Cammann, H., Meyer, H.-A., Miller, K., Jung, K., and Stephan, C. (2013). Artificial neural networks and prostate cancer-tools for diagnosis and management. *Nature reviews. Urology*, 10.
- [73] Hunter J., D. D. (2007). The matplotlib user’s guid. *Universidado de Sao Paulo*.
- [74] IBM (2020a). IBM Research | Haifa. URL: <http://www.research.ibm.com/labs/haifa/> [accessed: 2020-25-05].
- [75] IBM (2020b). What are message brokers? URL: <https://www.ibm.com/cloud/learn/message-brokers> [accessed: 2020-10-03].
- [76] Jadhav, M. A., Sawant, B. R., and Deshmukh, A. (2015). Single page application using angularjs. *International Journal of Computer Science and Information Technologies*, 6(3):2876–2879.
- [77] Janjic, V., Bowles, J., Vermeulen, A. F., Silvina, A., Belk, M., Fidas, C., Pitsillides, A., Kumar, M., Rossbory, M., Vinov, M., et al. (2019). The serums tool-chain: Ensuring security and privacy of medical data in smart patient-centric healthcare systems. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2726–2735. IEEE.
- [78] JDBC (2019). Introducing JDBC. URL: <https://docs.oracle.com/database/121/JJDBC/overvw.htm#JJDBC28025> [accessed: 2019-29-10].

- [79] Jun, J., Jacobson, S., and Swisher, J. (1999). Application of discrete-event simulation in health care clinics: A survey. *Journal of the Operational Research Society*, 50:109–123.
- [80] Jurafsky, D. and Martin, J. H. (2014). *Speech and language processing*. Pearson.
- [81] Kalin, M. (2015). Concurrent and parallel programming concepts.
- [82] Karpathy, A. (2016). Lecture 10: Recurrent neural networks, image captioning, lstm.
- [83] Keras (2020). Keras: The Python Deep Learning Library. URL: <https://keras.io/> [accessed: 2020-16-03].
- [84] Kerr, G. (2014). Dco database manual.
- [85] Komorowski, M. and Raffa, J. (2016). Markov models and cost effectiveness analysis: applications in medical research. In *Secondary Analysis of Electronic Health Records*, pages 351–367. Springer.
- [86] Kotz, S. and Nadarajah, S. (2000). *Extreme value distributions: theory and applications*. World Scientific.
- [87] Kreibich, J. (2010). *Using SQLite*. " O'Reilly Media, Inc."
- [88] Kupiec, J. (1992). Robust part-of-speech tagging using a hidden markov model. *Computer speech & language*, 6(3):225–242.
- [89] LaMorte, W. W. (2016). Cox proportional hazards regression analysis.
- [90] lazyprogrammer (2020). Machine Learning Example. URL: https://github.com/lazyprogrammer/machine_learning_examples/blob/master/supervised_class2/bagging_classification.py [accessed: 2020-16-03].
- [91] Leff, A. and Rayfield, J. T. (2001). Web-application development using the model/view/controller design pattern. In *Proceedings fifth ieee international enterprise distributed object computing conference*, pages 118–127. IEEE.
- [92] Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.
- [93] lifeline (2020). Generalized Extreme Value Distribution. URL: <https://lifelines.readthedocs.io/en/latest/Survival%20Regression.html> [accessed: 2020-18-03].
- [94] Lifelines (2019). Estimating hazard rate using Nelson-Aalen. URL: <https://lifelines.readthedocs.io/en/latest/Survival%20analysis%20with%20lifelines.html#estimating-hazard-rates-using-nelson-aalen> [accessed: 2020-16-03].
- [95] Lilley, C. (2016). Cancer information programme plan nhs lothian.
- [96] Lopes, R. H. C. (2011). *Kolmogorov-Smirnov Test*, pages 718–720. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [97] LSTM (2020). Understanding LSTM Network. URL: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> [accessed: 2020-16-03].

- [98] Malin, J. L. (2013). Envisioning watson as a rapid-learning system for oncology. *Journal of Oncology Practice*, 9(3):155–157. PMID: 23942497.
- [99] Mani, S., Chen, Y., Li, X., Arlinghaus, L., Chakravarthy, B., Abramson, V., Bhave, S., Levy, M., Xu, H., and Yankeelov, T. (2013). Machine learning for predicting the response of breast cancer to neoadjuvant chemotherapy. *Journal of the American Medical Informatics Association : JAMIA*, 20.
- [100] Masel, N., Haesendonckx, S., and Papadopoulou, A. (2019). An automation proof of concept of periodic reporting via r shiny.
- [101] Massa, M. S. (2016). Lecture 13: Kolmogorov smirnov test and power of tests.
- [102] Mase, M. (2011). *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. " O'Reilly Media, Inc."
- [103] Massey Jr, F. J. (1951). The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78.
- [104] mathwave (2020). Extreme Value Distribution. URL: <http://www.mathwave.com/articles/extreme-value-distributions.html> [accessed: 2020-18-03].
- [105] mathWorks (2020). Generalized Extreme Value Distribution. URL: <https://uk.mathworks.com/help/stats/generalized-extreme-value-distribution.html> [accessed: 2020-18-03].
- [106] MATLAB (2020). MATLAB Simulink. URL: <https://www.mathworks.com/products/matlab.html> [accessed: 2020-10-03].
- [107] Matplotlib (2020). Python plotting. URL: <https://matplotlib.org/> [accessed: 2020-16-03].
- [108] McFarland, D. S. (2011). *Javascript & jQuery: the missing manual*. " O'Reilly Media, Inc."
- [109] McKinney, W. (2013). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly, Beijing, first edition.
- [110] McKinney, W. G. (2010). Data structures for statistical computing in python.
- [111] Microsoft Access (2020). Microsoft Access Database Software. URL: <https://products.office.com/en-gb/access> [accessed: 2020-18-03].
- [112] Microsoft Data Migration Guide (2020). Migrate Access to SQL Server. URL: <https://datamigration.microsoft.com/scenario/access-to-sqlserver?step=1> [accessed: 2020-10-03].
- [113] Miller Jr, R. G. (1983). What price kaplan-meier? *Biometrics*, pages 1077–1081.
- [114] MS Access (2020). Microsoft Access Database Software. URL: <https://products.office.com/en-gb/access> [accessed: 2020-10-03].

- [115] Mumbaikar, S., Padiya, P., et al. (2013). Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, 3(5):1–4.
- [116] Murray, D. G. (2013). *Tableau your data!: fast and easy visual analysis with tableau software*. John Wiley & Sons.
- [117] MVC (2020). Definitions of Web-related terms. URL: <https://developer.mozilla.org/en-US/docs/Glossary/MVC> [accessed: 2020-30-03].
- [118] National Records (2019). Vital events - death | national records of scotland. URL: <https://www.nrscotland.gov.uk/statisticsanddata/statistics/statisticsbytheme/vitalevents/deaths> [accessed: 2019-12-10].
- [119] Nguyen, C., Wang, Y., and Nguyen, H.-N. (2013). Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic. *Journal of Biomedical Science and Engineering*, 06:551–560.
- [120] NHS-Lothian (2019). Rie scan lung cancer pathway.
- [121] NHS-Scotland (2019). Cancer waiting time standards definitions manual. *NHS*.
- [122] NumPy (2019). NumPy. URL: <https://numpy.org/> [accessed: 2020-16-03].
- [123] OECD (2005). Health technologies and decision making.
- [124] pandas (2019). pandas: Powerful python data analysis toolkit. URL: <https://pandas.pydata.org/pandas-docs/stable/> [accessed: 2019-29-10].
- [125] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., and Louppe, G. (2012). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12.
- [126] Perez-Marin, A. M. (2008). Empirical comparison between the nelson-aalen estimator and the naive local constant estimator. *SORT: statistics and operations research transactions*, 32(1):0067–76.
- [127] pickle (2020). pickle: Python object serialization. URL: <https://docs.python.org/3/library/pickle.html> [accessed: 2020-16-03].
- [128] Plaisant, C., Mushlin, R., Snyder, A., Li, J., Heller, D., and Shneiderman, B. (2003). Lifelines: using visualization to enhance navigation and analysis of patient records. *The craft of information visualization*, pages 308–312.
- [129] Power BI (2020). Microsoft Power BI: Data Visualization. URL: <https://powerbi.microsoft.com/en-us/> [accessed: 2020-10-03].
- [130] projectHanover (2019). Project Hanover. URL: <https://hanover.azurewebsites.net/> [accessed: 2019-23-09].
- [131] pyodbc (2019). Connecting to Microsoft Access. URL: <https://github.com/mkleehammer/pyodbc/wiki/Connecting-to-Microsoft-Access> [accessed: 2019-29-10].

- [132] python (2019). What is Python. URL: <https://www.python.org/doc/essays/blurb/> [accessed: 2019-29-10].
- [133] R Shiny (2020). Shiny from R Studio. URL: <https://shiny.rstudio.com/> [accessed: 2020-10-03].
- [134] Rad, R. (2018). *Pro Power BI Architecture: Sharing, Security, and Deployment Options for Microsoft Power BI Solutions*. Apress.
- [135] Redis (2020). Redis. URL: <https://redis.io/> [accessed: 2020-10-03].
- [136] Reiter, J. (2005). Using cart to generate partially synthetic public use microdata. *Journal of Official Statistics*, 21(3):441–462.
- [137] Rinne, H. (2008). *The Weibull distribution: a handbook*. CRC press.
- [138] RNN (2020). CS230 Recurrent Neural Network Cheatsheet . URL: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks> [accessed: 2020-16-03].
- [139] Rodriguez, G. (2005). Non-parametric estimation in survival models. *cited on*, page 20.
- [140] Roffman, C., Buchanan, J., and Allison, G. (2016). Charlson comorbidities index. *Journal of physiotherapy*, 62.
- [141] Rossi, F., Van Beek, P., and Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.
- [142] Rubin, D. B. (1993). Discussion statistical disclosure limitation. *Journal of Official Statistics*, 9(2):461–468.
- [143] Rubio, D. (2017). Rest services with django. In *Beginning Django*, pages 549–566. Springer.
- [144] Safe Havens (2019). Safe havens. URL: <https://www.nhsresearchscotland.org.uk/research-in-scotland/data/safe-havens> [accessed: 2019-12-10].
- [145] sas (2020). Analytics, Business Intelligence and Data Management. URL: <https://www.sas.com/> [accessed: 2020-10-03].
- [146] Sathya, R. and Abraham, A. (2013). Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2(2):34–38.
- [147] Scala (2020). The Scala Programming Language. URL: <https://www.scala-lang.org/> [accessed: 2020-08-06].
- [148] Schober, P., Boer, C., and Schwarte, L. A. (2018). Correlation coefficients: appropriate use and interpretation. *Anesthesia & Analgesia*, 126(5):1763–1768.
- [149] scikit-learn (2020). scikit-learn: machine learning in Python. URL: <https://scikit-learn.org/stable/> [accessed: 2020-16-03].

- [150] SciPy.org (2019). Scientific computing tool for Python. URL: <https://www.scipy.org/about.html> [accessed: 2020-16-03].
- [151] Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.
- [152] Sherstinsky, A. (2018a). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *arXiv preprint arXiv:1808.03314*.
- [153] Sherstinsky, A. (2018b). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314.
- [154] Shneiderman, B., Plaisant, C., and Hesse, B. W. (2013). Improving healthcare with interactive visualization. *Computer*, 46(5):58–66.
- [155] Silvina, A., Bowles, J., and Hall, P. (2019). On predicting the outcomes of chemotherapy treatments in breast cancer. In Riaño, D., Wilk, S., and ten Teije, A., editors, *Artificial Intelligence in Medicine*, pages 180–190, Cham. Springer International Publishing.
- [156] SMR Datasets (2020). Patient Identification and Demographic Information | Community Health Index (CHI) Number. URL: <https://www.ndc.scot.nhs.uk/Data-Dictionary/SMR-Datasets/Patient-Identification-and-Demographic-Information/Community-Health-Index-Number/> [accessed: 2020-18-03].
- [157] Spurlock, J. (2013). *Bootstrap: Responsive Web Development*. " O'Reilly Media, Inc."
- [158] sqlite (2019). About SQLite. URL: <https://www.sqlite.org/about.html> [accessed: 2019-29-10].
- [159] SSRS (2020). SQL Server Reporting Services (SSRS). URL: <https://docs.microsoft.com/en-gb/sql/reporting-services/> [accessed: 2020-10-03].
- [160] Steinberg, M. D., Juliano, M. A., and Wise, L. (1985). Psychological outcome of lumpectomy versus mastectomy in the treatment of breast cancer. *The American journal of psychiatry*.
- [161] Stigler, S. and Burdack, M. (2020). A practical approach of different programming techniques to implement a real-time application using django. *ATHENS JOURNAL OF SCIENCES*, 7:43–66.
- [162] Stimulsoft (2020). Stimulsoft Reports and Dashboards. URL: <https://www.stimulsoft.com/en> [accessed: 2020-16-03].
- [163] Sutton, R. S. and Barto, A. G. (2011). Reinforcement learning: An introduction.
- [164] Tableau (2020). Tableau Software. URL: <https://www.tableau.com> [accessed: 2020-10-03].
- [165] TensorFlow (2020). TensorFlow. URL: <https://www.tensorflow.org/> [accessed: 2020-16-03].

- [166] Tobias, J. and Hochhauser, D. (2005). *Cancer and its management*. John Wiley and Sons, Inc.
- [167] transactSQL (2019). Transact-SQL Reference (Database Engine). URL: <https://docs.microsoft.com/en-gb/sql/t-sql/language-reference?view=sql-server-2017> [accessed: 2019-29-10].
- [168] TSQL (2020). Transact-SQL. URL: <https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15> [accessed: 2021-11-08].
- [169] Valderas, J., Starfield, B., Sibbald, B., Salisbury, C., and Roland, M. (2009). Defining comorbidity: Implications for understanding health and health services. *Annals of family medicine*, 7:357–63.
- [170] van der Ham, R. (2018). Salabim: Open source discrete event simulation and animation in python. In *Proceedings of the 2018 Winter Simulation Conference, WSC '18*, page 4186–4187. IEEE Press.
- [171] Virtanen, P., Gommers, R., Oliphant, T., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S., Brett, M., Wilson, J., Millman, K., Mayorov, N., Nelson, A., Jones, E., Kern, R., Larson, E., and Pingel, T. (2019). Scipy 1.0—fundamental algorithms for scientific computing in python.
- [172] vis.js (2020). vis.js community edition*. URL: <https://visjs.org/> [accessed: 2020-16-03].
- [173] Walt, S. v. d., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science and Engg.*, 13(2):22–30.
- [174] Walters, S. J. (1999). *What is a Cox model?* Citeseer.
- [175] Wishart, G., Azzato, E., Greenberg, D., Rashbass, J., Kearins, O., Lawrence, G., Caldas, C., and Pharoah, P. (2010). Predict: A new uk prognostic model that predicts survival following surgery for invasive breast cancer. *Breast cancer research : BCR*, 12:R1.
- [176] Wohlgethan, E. (2018). *Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue. js*. PhD thesis, Hochschule für Angewandte Wissenschaften Hamburg.
- [177] Young, I. T. (1977). Proof without prejudice: use of the kolmogorov-smirnov test for the analysis of histograms from flow systems and other sources. *Journal of Histochemistry & Cytochemistry*, 25(7):935–941.
- [178] Zappa, C. and Mousa, S. A. (2016). Non-small cell lung cancer: current treatment and future advances. *Translational Lung Cancer Research*, 5(3).
- [179] Zhan, H. N. and Hall, P. (2016). Using real-world data to discover predictors of chemotherapy toxicity in breast cancer patients.
- [180] Zhang, X. (2018). Application of discrete event simulation in health care: a systematic review. *BMC health services research*, 18(1):1–11.

Appendix A

Architecture and Design Pattern

A.1 Representational state transfer (REST) API



Fig. A.1 A Web API interaction. Source: [102]

Representational State Transfer (REST) architectural style, a hybrid style derived from several network-based architectural styles [50] (e.g. data-flow styles, replication styles) [49], can be applied to the design application program interface (API) for web services. Web services are web servers that support the need for a site or any other client application. The client will use the API to communicate with web services. Figure A.1 shows an interaction between a client and a server via web API. A web API with REST architectural style is called REST API. The REST API is an API which utilises HTTP request (e.g. GET, PUT, POST, DELETE [51]) to read, update, write, and delete data. It is more preferred than the Simple Object Access Protocol (SOAP) communications because of network traffic, higher latency and delay caused by SOAP. As REST API leverages less bandwidth than SOAP, it is more suitable for internet uses [115].

A.2 Model-View-Controller

Model-View-Controller (MVC) is a design pattern that processes the data model, presentation information and control information separately. The model represents the data and does not

depend on the controller or the view. The view displays the model data while the controller processes the user's interaction and comprises the application's user interface [91]. Because of this separation, it allows the application to be more modular. With MVC, the application can be constructed and maintained with different interfaces, such as multiple languages, or different sets of user permissions.

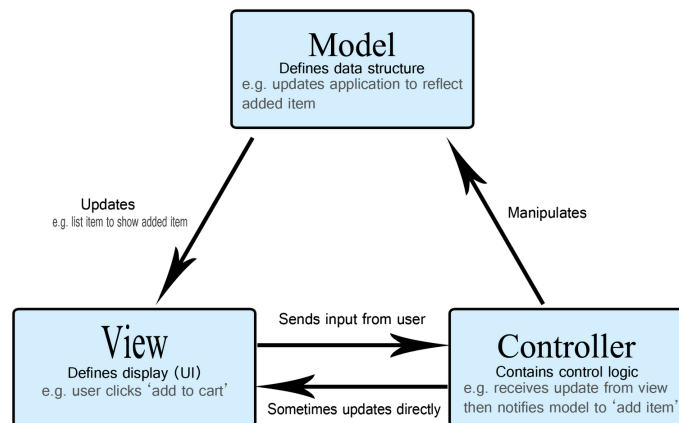


Fig. A.2 An example of an MVC application. Source: [117]

Figure A.2 shows an example of MVC for a simple shopping list application. The model specifies what data the list items should contain (e.g. item, price). The view defines the application GUI. The controller handles the logic for the user's interactions, for example, adding or removing an item in the shopping list.

The MVC pattern is most common for web application development. In the early days of the web, MVC architecture was commonly implemented on the server-side. However, today, more of the logic is pushed to the client with the advent of client-side data stores, and Http requests allowing partial page updates as required. Web frameworks such as AngularJS implement an MVC architecture [117].

Appendix B

Statistical Tools

B.1 Weibull Distribution

Weibull distribution is a member of extreme value distribution [86]. It is a continuous probability distribution and widely used in reliability engineering due to its versatility. The general expression of Weibull density function (i.e. for a random variable X) is given by the three-parameter Weibull distribution expression, as shown in equation B.1, is noted as $X \sim W_e(a, b, c)$ for short [137].

$$f_X(x|a, b, c) = \frac{c}{b} \left(\frac{x-a}{b} \right)^{c-1} e^{-\left(\frac{x-a}{b}\right)^c}, x \geq a \quad (\text{B.1})$$

Where:

$$f_X \geq 0, x \geq 0 \text{ or } a, -\infty < a < +\infty, b > 0, c > 0 \quad (\text{B.2})$$

and:

- a is the location parameter.
- b is the scale parameter.
- c is the shape parameter, also known as the Weibull slope

Different value of parameters can create mark effects on the behaviour of the distribution, as shown in Figure B.1, B.2, B.3.

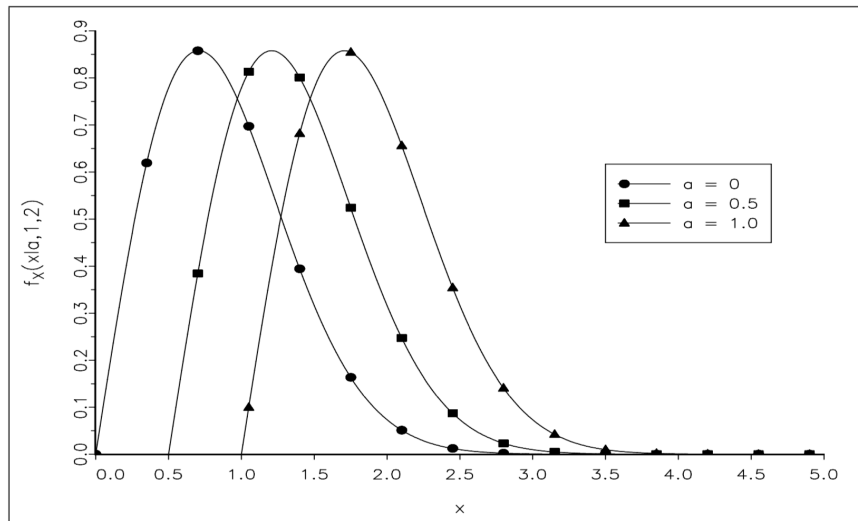


Fig. B.1 Weibull densities with differing value of the location parameter

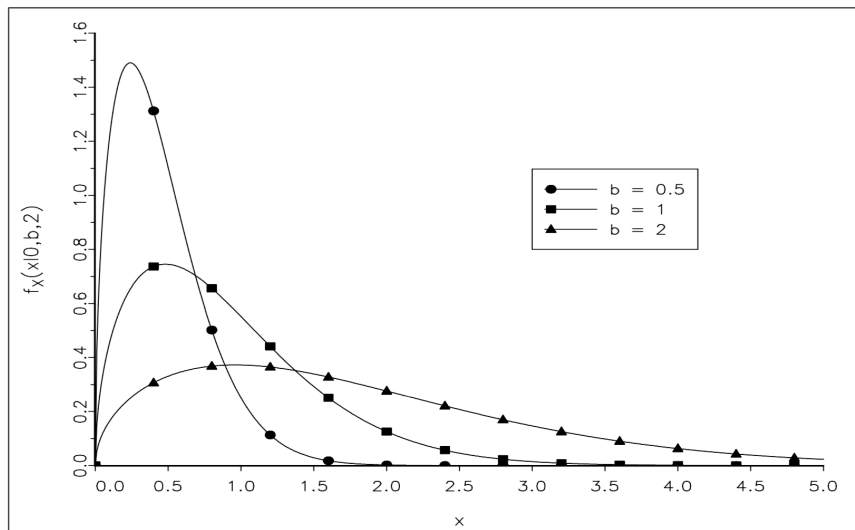


Fig. B.2 Weibull densities with differing value of the scale parameter

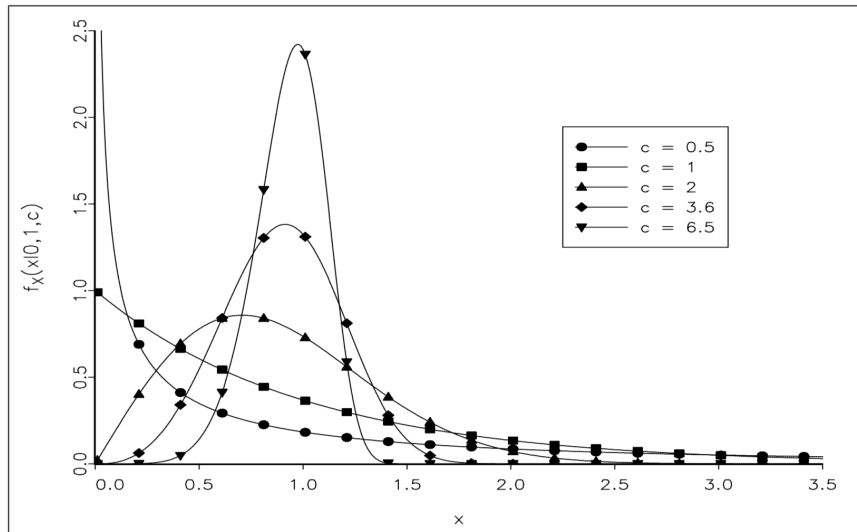


Fig. B.3 Weibull densities with differing value of the shape parameter

B.2 Generalized Extreme Value Distribution

The Generalized extreme value (GEV) [57] distribution is a family of continuous probability distribution developed within the field of extreme value theory. The extreme value theory allows inferences about the probability of very rare or extreme events. The GEV unites the Gumbel (Type I), Fréchet (Type II), and Weibull (Type III) extreme value distributions.

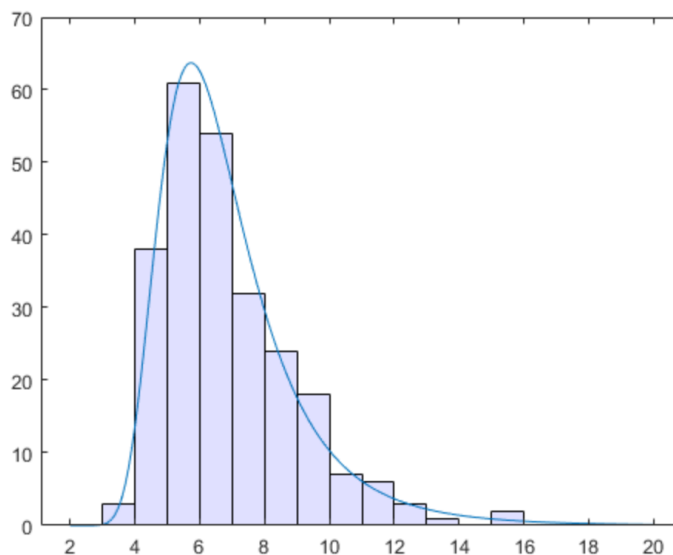


Fig. B.4 An example of GEV random sample [105]

Figure B.4 shows an example of a random sample with GEV distribution. The GEV has the following probability density function (PDF) as follow [86, 104, 105],

$$f(x) = \begin{cases} \frac{1}{\sigma} e^{-(1+kz)^{-\frac{1}{k}}} (1+kz)^{-1-\frac{1}{k}} & k \neq 0 \\ \frac{1}{\sigma} e^{(-z-e^{-z})} & k = 0 \end{cases} \quad (\text{B.3})$$

where $z = \frac{(x-\mu)}{\sigma}$, and k , σ , μ are the shape, scale, and location parameters.

The scale must be positive while the shape and location can take on any real value. The range of definition of the GEV distribution depends on k :

$$\text{range} = \begin{cases} (1+kz) > 0 & k \neq 0 \\ -\infty < x < +\infty & k = 0 \end{cases} \quad (\text{B.4})$$

B.3 Kaplan-Meier Estimator

The *Kaplan-Meier* (KM) [45] survival estimator is a tool for analysing censored data. For example, we want to examine the survival rate between two different treatments (e.g. comparing radiation therapy treatment (A) versus radiation plus chemotherapy treatment (B)).

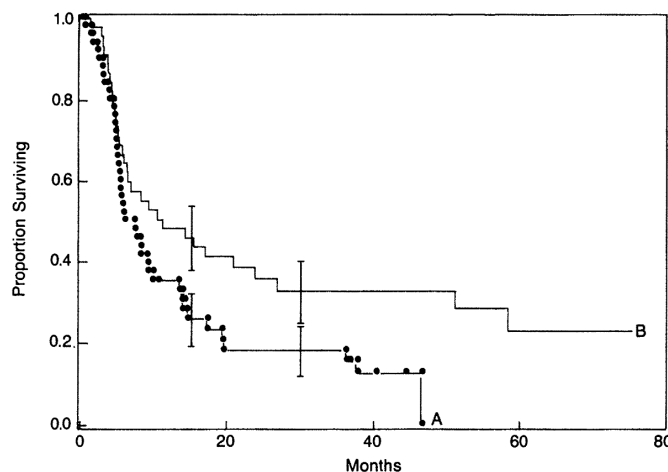


Fig. B.5 An example of *Kaplan-Meier* estimated survival curve. Source: [45]

By using the KM curve, we can compare the survival probability of the population. Here, as shown in Figure B.5, treatment B is significantly better as it has significantly higher surviving proportion than treatment A. The KM estimate simplifies the survival analysis computation despite all these difficulties associated with subjects or situations. The curve can be easily calculated and being totally non-parametric. Because it requires few assumptions,

it can be inefficient compare to the parametric estimators of survival analysis (e.g. less precision) [113]. It is also difficult to compare the hazard rate of the variables, even in the absence of statistical noise. Despite its limitation, the KM is still widely used. The appeal of this estimator is the ease of usage, faster calculation, and its few assumption requirements [59].

B.4 Nelson-Allen Estimator

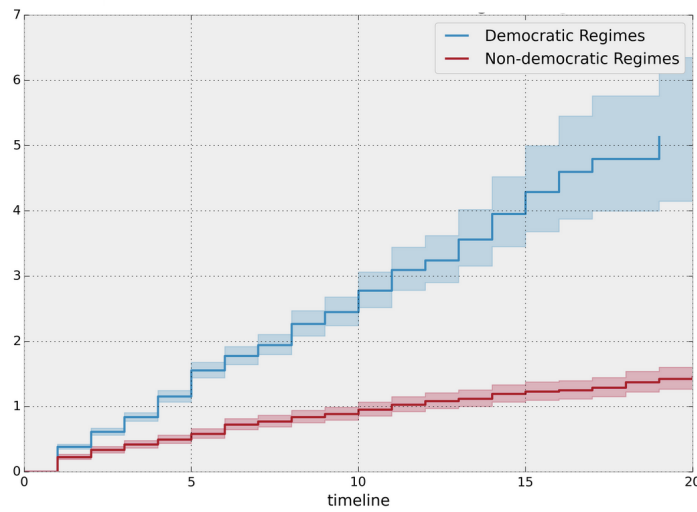


Fig. B.6 Cumulative hazard function of global regimes. Source: [94]

The Nelson-Aalen estimator was initially introduced by Nelson (1972) and later on rediscovered by Aalen (1978) who derived the estimator using modern counting process techniques [126]. It is a non-parametric estimator of the cumulative hazard rate function for censored or incomplete data. The cumulative hazard is used (e.g. in survival theory) to estimate the increasing number of the expected event (e.g. death, failure event). The KA estimator does not work well enough to observe the hazard function. Hence, the NA estimator is an alternative estimator to the KM estimator if we want to calculate the hazard function of variables [139]. To calculate the NA non-parametric cumulative hazard function with the equation, as shown below [94].

$$\hat{H}(t) = \sum_{t_i \leq t} \frac{d_i}{n_i} \quad (\text{B.5})$$

With the equation B.5, it is possible to calculate the hazard function over time. Figure B.6 shows an example of the NA estimate cumulative hazard. It is possible to identify whether the hazard of several variables are proportional by observing the NA curve.

B.5 Shapiro-Wilk Test

The Shapiro-Wilk test [151] is a test of normality. It tests whether a random sample derives from a normal distribution by calculating the W statistic. The null hypothesis of this test, H_0 , is as follows: the sample drawn from the normally-distributed population, while equation B.6 shows the formula for calculating W [35].

$$W = \frac{(\sum_{i=1}^n a_i x_i)^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (\text{B.6})$$

Here, x_i are the ordered sample values and a_i are constants generated from the means, variances and covariances of the order statistics of a sample of size n from a normal distribution. If the distribution is normal, both the denominator and nominator of the equation value is an estimate of σ^2 . Hence, if H_0 true, W should be equal to 1. The small differences between the values of denominator and nominator are evidence of departure from normality.

B.6 Kolmogorov–Smirnov Test

The Kolmogorov-Smirnov Test is a non-parametric test to check the difference between two sample distribution or to compare a sample to an expected statistical distribution [96]. Being a non-parametric test, it does not require the data to follow a normal distribution.

Consider two different observation X_1, X_2, \dots, X_n assumed to come from distribution P .

The Kolmogorov-Smirnov test hypotheses are as follows [103]:

- H_0 : the samples follow a specified distribution P
- H_1 : the samples do not follow a specified distribution P

We get the Kolmogorov-Smirnov statistic (D) by calculating the maximum distance between the empirical distribution function of the data ($F_{obs}(x)$), with the cumulative distribution function associated with the null hypothesis ($F_{exp}(x)$) as shown in equation B.7 [101].

$$D_n = \max_x |F_{exp}(x) - F_{obs}(x)| \quad (\text{B.7})$$

Figure B.7 illustrates the method to calculate Kolmogorov-Smirnov statistic. The green lines show the absolute distance between the expected and observed distribution function.

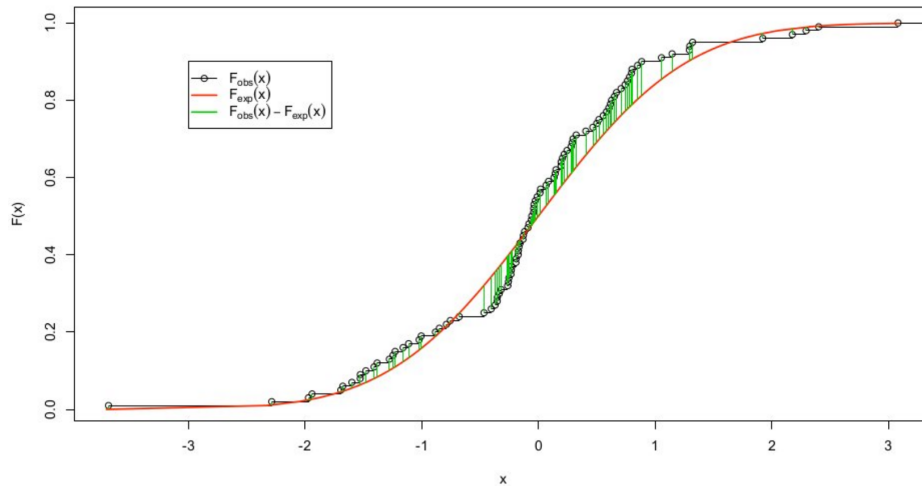


Fig. B.7 Kolmogorov-Smirnov statistic. Source: [101]

We then compare the maximum Kolmogorov-Smirnov statistic to the critical value ($D_{n,\alpha}$) based on the information from Figure B.8. If the D value is less than the $D_{n,\alpha}$, we do not reject the null hypothesis.

Sample size (N)	Level of significance (α)				
	0.20	0.15	0.10	0.05	0.01
1	0.900	0.925	0.950	0.975	0.995
2	0.684	0.726	0.776	0.842	0.929
3	0.565	0.597	0.642	0.708	0.828
4	0.494	0.525	0.564	0.624	0.733
5	0.446	0.474	0.510	0.565	0.669
6	0.410	0.436	0.470	0.521	0.618
7	0.381	0.405	0.438	0.486	0.577
8	0.358	0.381	0.411	0.457	0.543
9	0.339	0.360	0.388	0.432	0.514
10	0.322	0.342	0.368	0.410	0.490
11	0.307	0.326	0.352	0.391	0.468
12	0.295	0.313	0.338	0.375	0.450
13	0.284	0.302	0.325	0.361	0.433
14	0.274	0.292	0.314	0.349	0.418
15	0.266	0.283	0.304	0.338	0.404
16	0.258	0.274	0.295	0.328	0.392
17	0.250	0.266	0.286	0.318	0.381
18	0.244	0.259	0.278	0.309	0.371
19	0.237	0.252	0.272	0.301	0.363
20	0.231	0.246	0.264	0.294	0.356
25	0.21	0.22	0.24	0.27	0.32
30	0.19	0.20	0.22	0.24	0.29
35	0.18	0.19	0.21	0.23	0.27
over 35	1.07	1.14	1.22	1.36	1.63
	\sqrt{N}	\sqrt{N}	\sqrt{N}	\sqrt{N}	\sqrt{N}

Fig. B.8 Kolmogorov-Smirnov critical values. Source: [103]

Appendix C

Databases

C.1 Microsoft SQL Server

Microsoft SQL Server is a relational database management system developed by Microsoft. The latest version, as per March 2020 (i.e. Microsoft SQL Server 2019) is built on a database engine that supports On-Line Transaction Processing Benchmark (TPC-E) and Decision Support Benchmark (TCP-H). Furthermore, the use of Storage Class Memory (SCM) and in-memory database operations improve the transaction processing rate. It allows near-real-time data analysis/manipulation without having to move the transaction data to another temporary data warehouse for reporting or other purposes [62].

C.1.1 Transact-SQL (T-SQL)

is a set of programming extensions from Sybase and Microsoft that add several features to the Structured Query Language (SQL), including transaction control, exception and error handling, row processing and declared variables [167]. Instead of using vanilla SQL, we use T-SQL to interact with the Microsoft SQL Server database. The T-SQL are designed to extend SQL's abilities while being able to integrate well with SQL. Several features such as local variables and string/data processing are added.

C.2 Microsoft Access

Microsoft Access [111] is a database management system from Microsoft. It is a member of the Microsoft office suite. It is designed to create databases and organise reports locally. Microsoft Access can import or link directly to data stored in other applications and databases

(e.g. Microsoft Azure SQL, SQL Server). Currently, Microsoft Access is only available on Window PC.

C.3 SQLite

SQLite [158] is an open-source C-language library that implements a SQL database engine. The library provides a relational database management system (RDBMS) with several other features as follow [87].

- Serverless, the SQLite does not require a separate server or system to operate. The library can directly access the storage file.
- Cross-Platform.
- The library allows safe access from multiple processor or threads as the SQLite transactions are fully ACID (atomicity, consistency, isolation, durability)-compliant.
- SQLite support most of the query language features found in the SQL-92.

C.4 Redis

Redis [135] is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. *Redis* supports various data structures (e.g. strings, hashes, lists, sets, sorted sets with range queries). A message broker is software that enables systems to communicate with each other and exchange information. It mediates communication among applications by translating messages between formal messaging protocols. This allows interdependent services to directly interact with one another, even if the services were implemented on different platforms [75].

Appendix D

Framework

D.1 AngularJS

Angular is a google-maintained app-design framework and development platform for creating a single page application (SPA) using HTML and TypeScript[10]. A single page application is a web application with components that can be updated/replaced independently without refreshing/reloading the entire page [76]. The SPA client-server request-response cycle allows a flexible way of dealing with data (e.g. using JSON). Figure D.1 shows the difference between traditional and SPA response-request cycle.

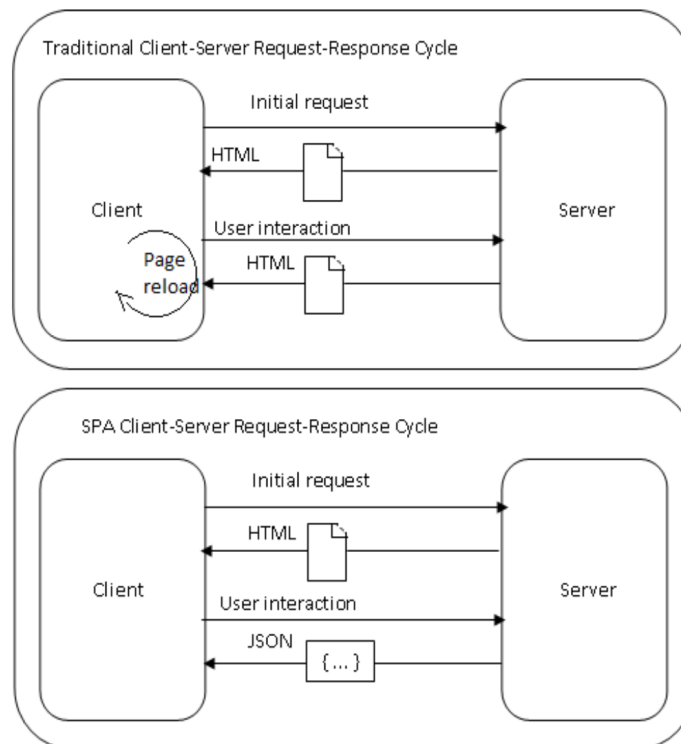


Fig. D.1 Client-server request-response cycle. Source: [76]

With its Model-View-Controller (MVC), sometimes refer as Model-View-Whatever (MV*), architectural pattern, Angular provides a separation of concerns between the user interface and business logic [53]. Furthermore, as shown in Figure D.2, the two way data binding features allows data to get simultaneously updated whenever there is a change in the Views.

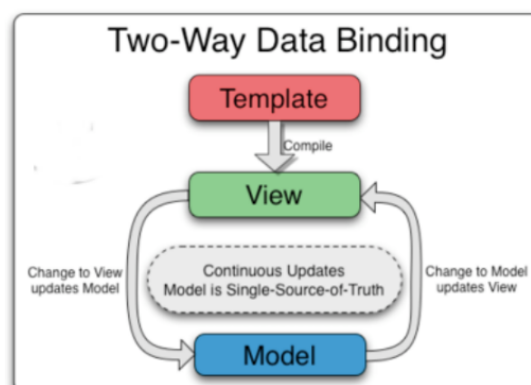


Fig. D.2 Two ways data binding [12]

D.2 Bootstrap

Bootstrap [22] is an open-source framework for developing the graphical user interface (GUI) of websites with HTML, CSS, and JavaScript. It was developed due to the demand for front-end toolset standardisation on Twitter [157]. Bootstrap streamlines the process of prototyping and building websites with its provided features such as Sass variables and mixins, responsive grid system, extensive prebuilt components, and various plugins built on jQuery [22].

D.3 Django

Django [41] is a high-level Python Web framework. It loosely implements the Model-View-Controller (MVC) pattern [6].

- The Django model is responsible for managing data and core business logic.
- The Django view displays the data to the user. It provides a template language for the developer for this purpose.
- The Django controller accepts users' input and performs the tasks/logic specific to the Django app.

A typical Django project is shown below [42].

```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    settings.py  
    urls.py  
    wsgi.py  
  app1/  
    apps.py  
    model.py  
    urls.py  
    views.py  
    ...  
  app2/
```

The Django project, as shown above, has a root directory (i.e. *mysite*). Inside the root directory, *manage.py* allows the interaction (e.g. run server, create a new app) to the Django projects via command-line. The inner folder *mysite* is the Python package of the project. It has *setting.py* containing the project configuration. The *urls.py* in this directory acts as the table of content of the Django site while *wsgi.py* is an entry-point for Web Server Gateway Interface (WSGI)-compatible web servers to serve the project. It is possible to have other types of configurations. A Django project may have multiple apps. Here we have *app1*, *app2*. The app has python sources to manage the display, data, business logic and URL specific to the app.

D.4 Gatling

Gatling [56] is an open-source load testing framework based on Scala, Akka, and Netty. Gatling comes with the support of the HTTP protocol that streamlines the process of load testing any HTTP server. The Gatling's load-testing consists of:

- A large number of users simulation by generating request on HTTP protocol or other types of protocol.
- All requests' response time aggregation.
- Response-request data analysing and report generation.

Gatling has its Domain Specific Language (DSL) for creating the test scenarios and is designed for continuous load testing, which can be integrated with the development pipeline.

Appendix E

Libraries and Toolkit

E.1 NumPy

Numerical Python (*NumPy*) [109] is one of the most common python libraries for numerical computing in python. The *NumPy* arrays (e.g. *ndarray*) [173] have become the standard representation for numerical data and data exchange for performing statistical analysis with python. The *NumPy* library has several features that help with scientific computing, as shown below [109, 122].

- Multidimensional array objects.
- Mathematical functions for processing entire data arrays without having to write loops.
- Functions to perform linear algebra, random number generation, and Fourier transform capabilities.
- A C-API for connecting *NumPy* with libraries written in C, C++, or *FORTRAN*.

Functions to perform linear algebra, random number generation, and Fourier transform capabilities. A C-API for connecting NumPy with libraries written in C, C++, or FORTRAN

E.2 pandas

pandas [124] is a python open sources data analysis and manipulation library. Panda provides a data structure placeholder for statistical datasets in the tabular format (i.e. *DataFrame*). This data structure has flexibility for its size, which is one of the important features when assembling a collection of data [110]. *pandas* library allows the modifications of many different kinds of data, such as [124]:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and un-ordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels.
- Any other form of observational/statistical data sets. The data doesn't need to be labelled at all to be placed into a pandas data structure.

pandas also has functions to streamline (e.g. *read_sql*, *read_csv*) the process of data loading from various sources. It has several built-in functionalities to perform various data manipulation and plotting (e.g. merging, concatenation, histogram plotting) [109]. Furthermore, *pandas* allows seamless/continuous integration with many python libraries for statistical analysis, such as *scipy*, *scikit-learn*.

E.3 SciPy

SciPy is a python-based open-source scientific computing library. It contains a collection of numerical algorithms and domain-specific toolboxes (e.g. for signal processing, optimisation statistics) [150]. It aims to provide complete coverage for mathematical analysis in linear algebra. The *SciPy* library provides statistical tools that one expects to find in a statistics textbook, such as probability distributions, hypothesis tests, frequency statistics, correlation functions [171].

E.4 scikit-learn

scikit-learn is a python module integrating extensive state-of-the-art machine learning algorithms for supervised and unsupervised learning [125]. This package provides a high-level API for predictive data analysis to non-specialists. The *scikit-learn* is built on *NumPy*, *Scipy*, *Matplotlib* and emphasis on ease of use, performance, documentation, and API consistency [125, 149].

E.5 Matplotlib

Matplotlib [107] is a python library for creating static, animated, and interactive visualisation. Although it was previously emulating the *MATLAB* [106] graphics commands, it does not

require *MATLAB* [73]. As a python library, it can be used in a *pythonic*, object-oriented way. Similarly, the *Matplotlib* uses *NumPy* to improve the performance for plotting large array.

E.6 Lifelines

lifelines [128] is a *python* library for performing survival analysis. It built on top of *pandas*. Previously, when performing survival analysis, it is common for data scientists to shuffle between two different programming languages (e.g. perform the analysis in *R* and deploy the application in *python*) [38]. As a *python* library with extensive capacity for performing various survival analysis techniques (e.g. *Kaplan-Meier* estimator, *COX* regression), it eases the process of deployment in the *python* production system as well as allows the integration with other *python* libraries.

E.7 Keras

Keras [83] is a high-level neural network *python* library which capable of running on the top of *TensorFlow*. It was developed with a motivation to enable fast neural network experimentation. *Keras* supports convolutional networks, recurrent networks and the combinations of the two. The API can run seamlessly on central processing unit (CPU) and graphics processing unit (GPU).

E.8 Tensorflow

TensorFlow [165] is an open-source platform for machine learning. It offers multiple levels of abstraction (e.g. the users can build and train models by using the high-level *Keras API* [83]). A *TensorFlow* computation is described by a set of nodes that construct a directed graph. The graph represents a data flow computation. The users typically build a computational graph using one of the supported front-end languages (i.e. *C++*, *Python*) [1]. This helps the clients to represents some machine learning algorithm (e.g. Neural Network) in their native representation.

E.9 Django REST Framework

Django REST Framework (DRF) [43] is a python toolkit for building web API. It streamlines the process of writing REST API. The DRF provides a browsable web interface as well as

integrated authentication mechanisms (i.e. to control the REST services access). It integrate well with an authentication mechanism, such as *OAuth2*, HTTP Signature, HTTP digest authentication, and JSON Web Token Authentication [143]. The DRF supports serialisation for both Object Relational Mapping (ORM) and non-ORM data sources.

E.10 Celery

Celery [29] is a task queue package which provides the realisation of the asynchronous computational workload. Task queues are used as a mechanism to distribute work across threads and machines. The input of task queues is a unit called task which is monitor by *Celery*'s workers. The worker is triggered by the information contained in the messages explicitly specified as a celery task. Celery used a storage solution (i.e. message broker) to handle the upcoming task queues. The message broker is used to store messages from the celery queue and storage once the system finishes executing the message. *Celery* can run on a single machine, multiple machines, or even across data centres.

E.11 pickle

pickle [127] is a *python* object serialisation module. It implements binary protocols for serialising and de-serialising (i.e. "pickling" and "unpickling") a Python object structure. During the serialisation process, the Python object hierarchy is converted into a byte stream and vice versa. The data format used by pickle is Python-specific. It imposes no restriction compared to the standard external procedure for record data/information such as JSON.

E.12 Salabim

Salabim [170] is an open-source object-oriented developed for discrete event simulation of complex control in logistics and production environments. It follows the methodology of process description as demonstrated in *Simula* and later in *Prosim*, *Must* and *Tomas*. As a python package, it allows the use of other powerful python libraries (e.g. for statistical processing, presentation, machine learning). It also has an integrated animation engine which eases the manual observation of process flow for the user (here, the healthcare professionals). Because of these reasons, we use *Salabim* for simulating lung cancer waiting time.

E.13 pyodbc

pyodbc [131] is an open-source Python module for accessing the Open Database Connectivity (ODBC) databases. It implements the Python Database API v2.0 specification. The *pyodbc* can establish a connection to many databases (e.g. Microsoft Access, SQLite, Microsoft SQL Server). The pyodbc module provides a connect function to the database specified in the connection string. Before using the pyodbc module to read/write to the database, the client needs to have an appropriate driver to establish the connection (e.g. *Microsoft Access Driver (*.mdb)*, *Microsoft Access Driver (*.mdb, *.acldb)* for Microsoft Access).

E.14 JDBC

Java Database Connectivity (JDBC) [52] is the industry standard for database-independent connectivity between the Java programming language and SQL databases and other tabular data sources, such as spreadsheets or flat files. It is part of the Java Standard Edition library. The JDBC provides a call-level API for SQL-based database access [78]. It uses JDBC drivers (e.g. JDBC-ODBC bridge driver, native driver, network protocol driver) to establish the connection. The JDBC API simplifies the process of sending SQL statements to relational database systems as it supports almost all dialects of SQL. The JDBC API allows applications to virtually access any data sources and runs on any platform with a Java Virtual Machine.

E.15 vis.js

vis.js [172] is a JavaScript dynamic, browser-based visualisation library. It designs to allow the manipulation of and interaction with the data in the web-client. The library has several components with different functionalities such as timeline creation (i.e. *Timeline*), 2D/3D graphs generation (i.e. *Graph2d*, *Graph3d*).

E.16 Data-Driven-Documents (D3)

Data-Driven-Documents (D3.js) [36] is a JavaScript library for manipulating documents based on data. The D3.js enables data visualisation and interaction using HTML, Scalable Vector Graphics (SVG), and Cascading Style Sheets (CSS). It combines visualisation component and data-driven approach to Document Object Model (DOM) manipulation. The library provides features similar to *jQuery* [108] (i.e. a popular JavaScript library to perform HTML DOM tree traversal and manipulation). It extends them to allow extensive DOM

modification. With the extensions, it will enable the creation of visuals based on the structure of data instead of only being a framework for low-level DOM manipulation [71].

E.17 Gradle

Gradle is a general purpose build management system. Gradle supports the automatic download and configuration of dependencies or other libraries from online repositories (e.g., Maven, Ivy) [63]. It allows reusing the artifacts of existing build systems.