

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EPIC IMPOC COVID Project

Yannis Panagis, Selena Tabbara, Yichen Liu, Daniel Gallego Rubio, Ghajaanan Arunachalam

A report submitted for

3rd Year Group project

June 25, 2020

The final product can be found here : <https://github.com/bahp/django-epicimpoc-covid19-ai>
The testing Environment can be found here : <https://github.com/bahp/pysml>

0 Contents

1	Introduction	1
2	Literature Review	3
2.1	A systematic review of clinical decision support systems for antimicrobial management: are we failing to investigate these interventions appropriately? ²⁵	3
2.2	Risk Factors for severity and mortality in adult COVID-19 inpatients in Wuhan ¹⁹	3
2.3	Predictors of Mortality for Patients with COVID-19 Pneumonia Caused by SARS-CoV-2: A prospective Cohort Study ¹⁰	3
2.4	Characteristics of CoronaVirus Disease 2019 in China ¹³	4
2.4.1	Key Design Decisions	4
2.4.2	Key Results	4
2.4.3	Limitations of research	5
3	Design Criteria	6
4	System Architecture	7
4.1	Risk Assessment Using Probabilistic Inference	7
4.2	Patient Retrieval - Case Based Reasoning	7
4.3	Back-end	7
4.4	Front-end	8
4.5	Technologies used	8
5	Data Preprocessing	9
5.1	Data history	9
5.2	Key statistics and class distribution	9
5.2.1	Probabilistic Inference Module	9
5.2.2	CBR	9
5.3	Pathology datasets pre-processing	9
5.3.1	Data cleaning	10
5.3.2	Getting the daily records for each patient	10
5.3.3	Daily record selection	10
5.3.4	Merging the data with the labels	10
5.3.5	Row and features selection	10
5.4	Vital Signs dataset pre-processing	10
5.5	Features Selection	11
5.5.1	Filter-based selection	11
5.5.2	Wrapper-based selection	11
5.5.3	Embedded selection – using Random Forest	11
6	Probabilistic Inference	13
6.1	Estimators (Binary Classifiers) Research	13
6.1.1	Logistic Regression	13
6.1.2	Naive Bayes	13
6.1.3	K-Nearest Neighbors Classifier	13
6.1.4	Decision Trees	14
6.1.5	Random Forests	14
6.1.6	Support Vector Classifier	14
6.1.7	Extreme Gradient Boosting with XGBoost	15
6.1.8	Light Gradient Boosted Machine (LightGBM)	15
6.1.9	Other Estimators	15
6.2	Filtering Techniques Research	16
6.2.1	IQR filter	16
6.2.2	Isolation Forest (IsolationForestFilter)	16
6.2.3	Local Outlier Filter	16

6.2.4	Elliptical Envelope (EllipticalEnvelopeFilter)	16
6.3	Imputation Techniques Research	16
6.3.1	Uni-variate vs Multivariate Imputation	16
6.3.2	Uni-variate Feature Imputation	16
6.3.3	Multivariate Feature Imputation Using Iterative Imputation	17
6.3.4	Nearest Neighbors Imputation using KNN	17
6.4	Re-sampling Techniques Research	17
6.4.1	RandomUnderSampler	17
6.4.2	RandomOverSampler	17
6.4.3	SMOTE (Synthetic Minority Oversampling Technique)	18
6.4.4	TomekLinks	18
6.4.5	SMOTETomek	18
6.5	Feature Scaling Techniques Research	18
6.5.1	StandardScaler	18
6.5.2	MinMaxScaler	18
6.5.3	MaxAbsScaler	18
6.5.4	RobustScaler	18
6.6	Analysis of Results	19
6.7	Conclusion; Final Design and Implementation	21
6.7.1	Light Gradient Boosting Machine	21
6.7.2	Random Forest Classifier	22
6.7.3	Decision Tree Classifier	24
7	Case Base Reasoning	26
7.1	Supervised CBR	26
7.1.1	Introduction	26
7.1.2	Evaluation technique	27
7.1.3	Distance metrics	27
	Manhattan Distance	27
	Euclidean Distance	27
	Chebyshev Distance	27
	Minkowski Distance	28
	Canberra Distance	28
	Bray Curtis Dissimilarity	28
7.1.4	Results and discussion	28
	Distance Metrics	28
	Filtering	29
	Imputation	29
	Feature Scaling	30
7.1.5	The Implementation	31
	Vital signs Dataset	31
	Pathology Dataset	31
7.1.6	Limitations and Future Work	32
8	Unsupervised CBR	35
8.1	Motivations	35
8.2	Clustering Algorithms	35
8.2.1	Model Based Clustering	35
8.2.2	Deep Neural Network (DNN)	36
8.3	KMeans	36
8.3.1	The Elbow Method	37
8.3.2	Silhouette	37
8.3.3	Curse of dimensionality	39
8.4	HDBSCAN	39
8.4.1	Density based clustering	39

8.4.2	Hierarchical Clustering	40
8.4.3	Real HDBSCAN algorithm	40
8.4.4	Visualisation and Evaluation	40
8.5	Implementation of the Final Product	40
8.6	Future Work	42
9	Django Application	43
9.1	What is Django?	43
9.2	Design of User Interface	44
9.3	The App's views	44
9.3.1	CBR-Vital signs	44
	Vital signs home view	44
	Patient search results view	44
	Patient cases view	44
	Query similar cases view	44
9.3.2	CBR-Pathology	45
	Manual input form view	45
	Query similar cases view	46
9.3.3	Inference	46
	Manual input form	46
	CSV file upload	46
	Database retrieval	46
9.3.4	File Processing and Predictions	46
9.4	The API	47
10	Project Management	48
11	Conclusion and Evaluation	50
11.1	Report of Ethical Consequences	50
	11.1.1 Ethical Decisions	50
	11.1.2 Patient Privacy and Confidentiality	50
11.2	Model Evaluation	50
11.3	Future Work	51
12	Meeting Minute Summaries	52
12.1	Meetings with Client & Supervisor	52
	12.1.1 Friday, 1st May 2020	52
	12.1.2 Friday, 8th May 2020	54
	12.1.3 Friday, 15th May 2020	55
	12.1.4 Friday, 22nd May 2020	57
	12.1.5 Friday, 29th May 2020	58
	Questions made for the meeting in advance	58
	Client and Supervisor Meeting	59
	12.1.6 Friday, 5th June 2020	61
	12.1.7 Friday, 12th June 2020 (Questions with Answers from Bernard	63
	12.1.8 Friday, 19th June 2020	65
12.2	Group's Independent Progress Meetings	67
	12.2.1 Tuesday, 30th April 2020	67
	12.2.2 Monday, 4th May 2020	68
	12.2.3 Monday, 11th May 2020	69
	12.2.4 Thursday, 14th May 2020	70
	12.2.5 Thursday, 4th June 2020	72

A Appendix	74
A.1 Imputation Research Results	75
A.2 Re-sampling Research Results	76
A.3 Feature Scaling Research Results	77
References	79

Abstract

This report describes the development and implementation of a clinical decision support system module for COVID-19. **EPIC IMPOC** (Enhanced, Personalised and Integrated Care for Infection Management at the Point of Care) is a clinical decision support system used in the NHS to improve the management of infectious diseases by facilitating data collection, infection diagnostics and antimicrobial therapy advice at the point of care. This will allow clinicians and medical care workers not only to retrieve similar patients from a database, but also to use pathology tests to identify whether or not COVID-19 as well as perform a risk assessment. This module will be integrated within EPIC IMPOC in the near future and tested in Clinical Trials in early July.

Acknowledgements

This report and project would not have been possible without the guidance of Bernard Hernandez and Pau Herrero. Their continuous feedback and support was essential in the two months over which this project was developed and we cannot thank them enough.

We would also like to give a secondary thank you to Danah. Your tireless help proofreading this report and discussing the biological implications of this work was a valuable resource.

1 Introduction

The first case of COVID-19 was detected in Wuhan, China on the 31st of December, 2019. Within three months, the World Health Organization (WHO) characterized the viral disease as a pandemic because of its high reproduction rate and threat to global public health. With the UK's ever-increasing death toll and number infected, the government imposed a lockdown, shutting down all non-essential services on the 23rd of March 2020. Although restrictions across the globe are lifting, the UK and many other countries continue efforts to limit the spread of the virus and protect the National Health Service (NHS) from being overwhelmed.

According to the NHS, as of Saturday, the 23rd of May 2020, there have been 257,154 total lab-confirmed cases, 36,675 total associated deaths. Each day, the NHS estimates that there are another 2,959 daily lab-confirmed cases and another 282 daily associated deaths across the UK.²¹ The NHS reports the number of daily cases and associated deaths continues to decline. However, the implications of easing lock-down restrictions is important to consider in preventing the spread or resurgence of the virus.

Thus, with a second wave looming in the autumn, there is a crucial need for technical solutions such as clinical decision support systems to improve patient diagnosis, risk assessment and the distribution of medical resources for infection management.

A clinical decision support system (CDSS) is intended to improve healthcare delivery by enhancing medical decisions with targeted clinical knowledge, patient information, and other health information. CDSSs are composed of two main categories, knowledge based and non-knowledge based. In short, the knowledge based CDSS a predefined set of rules is created based on evidence from literature, practice, or patients. Alternatively, the non-knowledge based CDSS relies on artificial intelligence (AI), machine learning, or statistical pattern recognition.²⁸

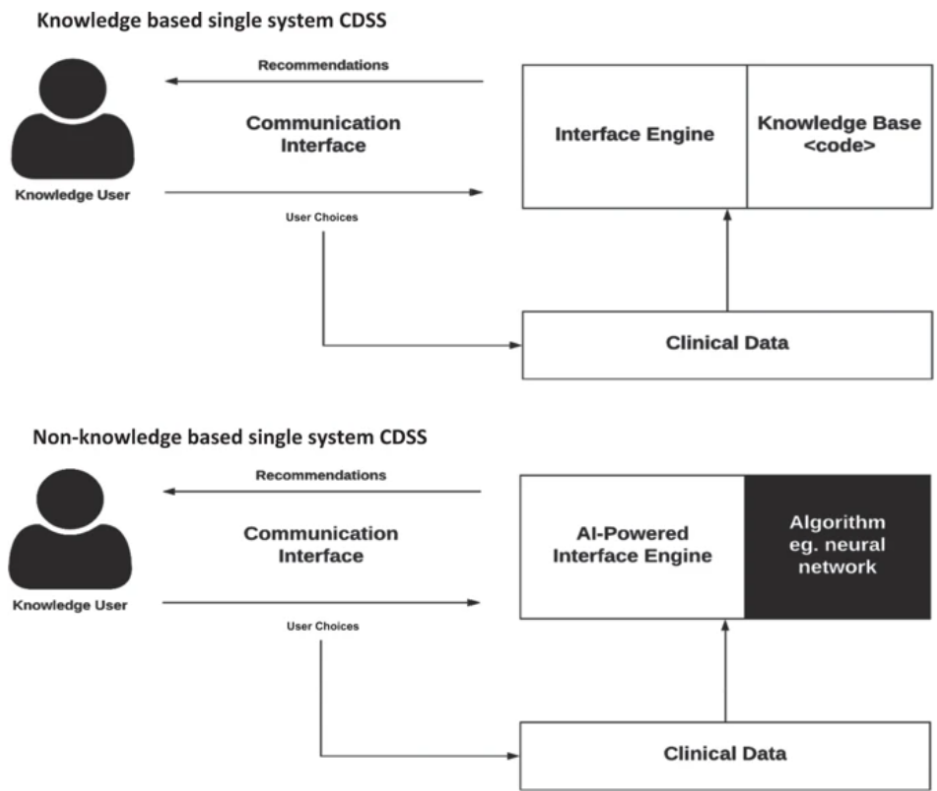


Figure 1.1: Diagram of Different Clinical Decision Support Systems (CDSSs)

The function of a CDSS ranges in complexity, with the simpler examples using knowledge based systems for administrative and clinical management. However, the use of non-knowledge based systems is necessary, because they can leverage models that describe the problem in terms of probability distributions to infer outcomes. There is an ever-growing need for health care professionals to diagnose and treat patients with infectious diseases as quickly as possible. With the strain put on the NHS under current conditions by COVID-19, there is an even greater need for better hospital resource allocation and patient diagnostics.

EPIC IMPOC (Enhanced, Personalised and Integrated Care for Infection Management at the Point of Care) is a clinical decision support system used in the NHS to improve the management of infectious diseases by facilitating data collection, infection diagnostics and antimicrobial therapy advice at the point of care.¹⁷

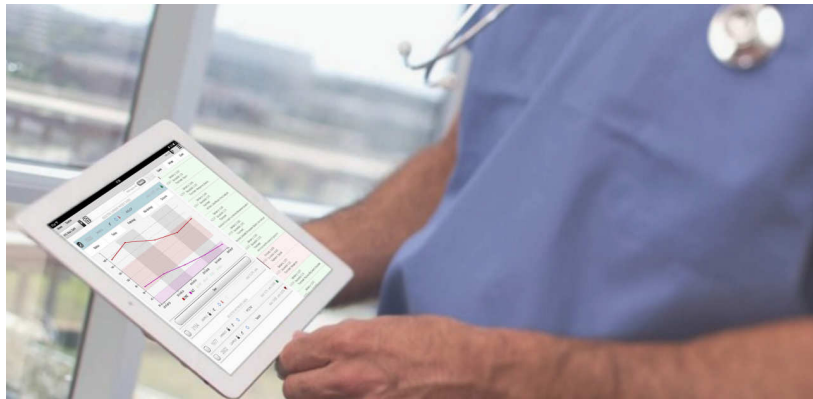


Figure 1.2: Photograph of EPIC IMPOC in use by clinicians

This module is a new component of EPIC IMPOC that focuses specifically on integrating current medical knowledge of COVID-19 and patient data with an inference engine to predict the likelihood of infection, the severity of infection, and to allow clinicians to retrieve the most similar patients to an individual patient at the point of care. Additionally, the patient retrieval system is packaged in a mobile and desktop friendly panel for clinicians within EPIC IMPOC to identify patients with similar severity levels, symptoms, and medical histories in the UK.

This EPIC IMPOC module is one of the first planned to undergo clinical trials in the NHS, starting in early July 2020. This solution will be a crucial part of the Imperial College Healthcare NHS Trust's COVID-19 response to the second wave in the pandemic. The Trust's five hospitals currently treat more than 1.125 million patients and employ 11,800 people annually.

Implementing a clinical decision support system for COVID-19 comes with a unique set of technical, social, and medical challenges. Mainly, the technical approach must adapt to the rapidly changing data and medical knowledge available. Furthermore, limited data creates challenges in developing robust machine learning algorithms to provide advanced, intelligent decision support for clinicians at the point of care. This required extensive research and testing of different algorithmic approaches, tuning their hyper-parameters and evaluating their limitations and strengths. The following sections of this report will include a literature review, the design criteria, a system architecture description, as well as a breakdown of the core components used in this module to overcome these challenges.

2 Literature Review

This section contains a literature review of some papers used to develop an improved understanding of the problem, biological context and informed the design decisions that were made in the development of this module. Studies, papers and approaches in English were selected based on their viability for implementation within the given time constraints as well as their relationship to the given brief.

2.1 A systematic review of clinical decision support systems for antimicrobial management: are we failing to investigate these interventions appropriately?²⁵

This paper investigates Clinical Decision Support Systems (CDSSs) as systems for antimicrobial management. CDSSs can be used to support clinicians to optimize antimicrobial therapy. Fifty-eight articles were included, describing 38 independent Clinical Decision Support Systems. Most systems that target antimicrobial prescription are platforms integrated with electronic medical records. These systems use a rule-based infrastructure to provide decision support. One of the biggest limitations of CDSS studies is that they fail to investigate and report any consideration of the non-expert, end-user. This literature review paper also suggests that this may be a primary factor in the poor engagement with CDSSs by clinicians. The article concludes by proposing that CDSS adoption can only become wide-spread when they are designed considering the non-expert end-user and their decision making processes.

2.2 Risk Factors for severity and mortality in adult COVID-19 inpatients in Wuhan¹⁹

This paper seeks to evaluate the severity on admission, complications, treatment, and outcomes of patients with COVID-19. Patients with COVID-19 admitted to Tongji Hospital from January 26, 2020, to February 5, 2020, were retrospectively enrolled and followed-up until March 3, 2020. Potential risk factors for severe COVID-19 were analysed by a multi-variable binary logistic model. Cox proportional hazard regression model was used for survival analysis in severe Patients.

Patients with older age, hypertension, and high lactate dehydrogenase level need careful observation and early intervention to prevent the potential development of severe COVID-19. Severe male patients with heart injury, hyperglycemia, and high-dose corticosteroid use may have a high risk of death. The mortality rate of patients with COVID-19 was 5.0% in Wuhan, which was close to that in the world (4.2%) and much higher than that in mainland China except Wuhan (2.4%).

This study aimed to describe and compare the epidemiological, demographic, clinical, laboratory, and radiological characteristics as well as the complications, treatment, and outcomes of hospitalized patients with non-severe and severe COVID-19. On the basis of whether or not requiring ventilatory support on admission, severe cases upon admission were divided into 2 cohorts, severely ill and critically ill cases. In the final logistic regression model, variables such as age 65 years or more (OR, 2.2; 95% CI, 1.5-3.5), hypertension (OR, 2.0; 95% CI, 1.3-3.2), LDH more than 445 U/L (OR, 4.4; 95% CI, 2.6- 7.6), and d-dimer more than 1 mg/L (OR, 2.2; 95% CI, 1.4-3.3) were significantly associated with cases with severe COVID-19

Multi-variable Cox proportional hazards regression analysis revealed that male sex (adjusted HR, 1.7; 95% CI, 1.0-2.8), age 65 years or more (adjusted HR, 1.7; 95% CI, 1.1-2.7), blood leukocyte count more than 10 cells/mm³ (adjusted HR, 2.0; 95% CI, 1.3-3.3), and LDH more than 445 U/L (adjusted HR, 2.0; 95% CI, 1.2-3.3) at admission, cardiac injury (adjusted HR, 2.9; 95% CI, 1.8-4.8), hyperglycemia (adjusted HR, 1.8; 95% CI, 1.1-2.8), and administration of high-dose corticosteroids (adjusted HR, 3.5; 95% CI, 1.8-6.9) during hospitalization were significant risk factors associated with death in cases with severe COVID-19 (Table IV).

2.3 Predictors of Mortality for Patients with COVID-19 Pneumonia Caused by SARS-CoV-2: A prospective Cohort Study¹⁰

The objective of this study is to identify factors associated with the death for patients with COVID-19 pneumonia caused by a novel Corona Virus SARS-CoV-2. All clinical and laboratory parameters were collected prospectively from a cohort of patients with COVID-19 pneumonia who were hospitalised to Wuhan Pulmonary Hospital, Wuhan City, Hubei Province, China, between December 25, 2019 and February 7, 2020. Uni-variate and multivariate logistic regression was performed to investigate the relationship between each variable and the risk for death of COVID-19 pneumonia patients. 179 patients with COVID-19 pneumonia (97 male and 82 female) were used.

Analysis revealed that age, preexisting concurrent cardiovascular or cerebrovascular diseases, CD3+CD8+ T cells, and cardiac troponin were associated with an increase in risk of mortality of COVID-19 pneumonia. In the sex [U+2012], age [U+2012], and comorbid illness-matched case study, CD3+CD8+ T cells, and cardiac troponin remained to be the predictors for high mortality of COVID-19 pneumonia. Four key risk factors were identified: age ≥ 65 , preexisting concurrent cardiovascular or cerebrovascular diseases, CD3+CD8+ T cells, and cardiac troponin, especially the latter two factors, were predictors for mortality of COVID-19 pneumonia patients.

2.4 Characteristics of CoronaVirus Disease 2019 in China¹³

Uses data regarding 1099 patients w/laboratory-confirmed Covid-19 from 552 hospitals in 30 provinces in mainland China through January 29, 2020. The objective of the paper was to evaluate patients and determine need for admission to an intensive care unit (ICU), use of mechanical ventilation, or death. The pathogen was identified as a novel enveloped RNA betacoronavirus, also known as severe acute respiratory syndrome Coronavirus 2 (SARS-CoV-2), which has a phylogenetic similarity to SARS-CoV.

2.4.1 Key Design Decisions

- The incubation period was defined as the interval between the potential earliest date of contact of the transmission source (wildlife or person with suspected or confirmed case) and the potential earliest date of symptom onset (i.e., cough, fever, fatigue, or myalgia).
- Fever was defined as an axillary temperature of 37.5°C or higher.
- Continuous variables were expressed as medians and interquartile ranges or simple ranges, as appropriate.
- No imputation was made for missing data.

2.4.2 Key Results

- Of the 7736 patients with Covid-19 who had been hospitalized at 552 sites as of January 29, 2020, this paper obtained data regarding clinical symptoms and outcomes for 1099 patients (14.2%).
- The most common patterns on chest CT were ground-glass opacity (56.4%) and bilateral patchy shadowing (51.8%).
- On admission, lymphocytopenia was present in 83.2% of the patients, thrombocytopenia in 36.2%, and leukopenia in 33.7
- Patients with severe disease were older than those with non-severe disease by a median of 7 years.
- Of 975 CT scans that were performed at the time of admission, 86.2% revealed abnormal results.
- A positive D-dimer indicates the presence of an abnormally high level of cross-linked fibrin degradation products in your body. It tells your doctor that there has been significant clot (thrombus) formation and breakdown in the body, but it does not identify the location or cause.
- Most of the patients had elevated levels of C-reactive protein; less common were elevated levels of alanine aminotransferase, aspartate aminotransferase, creatine kinase, and D-dimer.
- The median duration of hospitalization was 12.0 days (mean, 12.8). During hospital admission, most of the patients received a diagnosis of pneumonia from a physician (91.1%), followed by ARDS (3.4%) and shock (1.1%).
- Patients with severe disease had a higher incidence of physician-diagnosed pneumonia than those with non-severe disease (99.4% vs. 89.5%).
- Conventional routes of transmission of SARSCoV, MERS-CoV, and highly pathogenic influenza consist of respiratory droplets and direct contact, mechanisms that probably occur with SARS-CoV-2 as well.
- The absence of fever in Covid-19 is more frequent than in SARS-CoV (1%) and MERS-CoV infection (2%), so afebrile patients may be missed if the surveillance case definition focuses on fever detection.

The key conclusion in this paper is that early isolation, early diagnosis, and early management might have collectively contributed to the reduction in mortality in Guangdong.

2.4.3 Limitations of research

The limitations of this research include that the incubation period could only be estimated in 291 of the study patients who had documented information. This uncertainty of the exact dates (recall bias) might have inevitably affected the conclusions and generalisability of the results. Many patients also remained in the hospital at the time the paper was written so their outcomes were unknown at the time of data cut-off. This analysis also missed out patients who were asymptomatic or had mild cases and who were treated at home, and so the study may represent the more severe end of Covid-19.

3 Design Criteria

This section contains a summary of the design criteria used to guide the development of the COVID-19 CDSS module. The key design criteria are summarised in Table 3.1

Aspect	Specification
Performance	The probabilistic inference module has a balanced accuracy of approximately 81% with three distinctive models. Similarly, the Case-Based Reasoning system provides well above 77% confidence that the retrieved similar patients and input patient have the same outcomes at the end point of care.
Testing	The models and systems were intensively tested using a sample dataset with approximately 1187 patients. Testing methods include repetitive k-fold and leave-one-out cross-validation with deviations of each run recorded and compared. The result of each module has been sampled and monitored by clinical professionals.
Durability	The module considers cases from the first wave of the pandemic, and can be updated with more patients or refined with more models.
Maintenance	A fully documented library and API that allows continuous maintenance, development, and the addition of new features.
Sustainability	The documentation and API enable other developers and medical professionals to join and commit changes to the system under the expertise of medical advisors and the EPIC IMPOC team.
Compatibility	The module is designed to be fully compatible with the EPIC IMPOC CDSS by using similar user interface components, a similar dataset and database structure as well as the same core components for the probabilistic inference module implementation and design.
Ergonomics	This system is easy to use for non-technical background personnel with the detailed instruction and intuitive user interface. The API can be accessed using PCs, Tablets, and even mobile phones.
Justification	A detailed non-technical explanation of the models' results is provided to clinicians and healthcare personnel on their request. This is crucial for any clinical decision support systems with similar ethical concerns.

Table 3.1: Design Criteria

4 System Architecture

The module performs two functions: the retrieval of similar cases and the risk assessment of patients in terms of COVID-19 infection.

4.1 Risk Assessment Using Probabilistic Inference

The probabilistic inference (PI) function estimates the probability of COVID-19 infection in a patient based on key pathology biomarkers. It was trained and tested on a total of 1186 pathology (Table 4.1). New data can be input by form, existing patient records, or by uploading a CSV file with several patients' data. Three machine learning algorithms among other tested algorithms, each giving a different result, are used: LightGBM, Random Forest Classifier and Decision Tree Classifier. If a form or existing patient record is used, three different estimates are returned. In the case of a CSV file, an updated file with three new columns is returned, each column corresponding to a different estimator.

Data	Total	Covid-19 positive	Covid-19 negative
Pathology	77351	-	-
Labels	1675	976	699
Pathology \cap Labels	1186	758	428

Table 4.1: Class distribution for the important datasets used in the Probabilistic Inference Module

4.2 Patient Retrieval - Case Based Reasoning

The CBR system retrieves similar patients and provide them to the clinicians to support them with treatment prescription. The patients' retrieval is going to be done using the pathology dataset which contains 837 pathology profiles of and the vital signs dataset which contains the vital signs, symptoms, radiology results and medical history of 262 patients (Table 4.2). For the former, a supervised and an unsupervised method were used, while for the latter only the supervised methodology was followed.

In the supervised approach, the weighted distance between a given patient and the remainder of cases is computed and the most similar cases are selected. These weights are pre-computed using each feature's importance from the Random Forest. The system's performance was evaluated using the patient's outcome to verify and investigate patient similarities. Different distance metrics, filters, imputers and scalers were tested and selected to maximise the performance.

On another hand, the unsupervised approach for CBR, uses an HDBSCAN algorithm initialised with the same pre-computed weights to cluster similar patients by their biomarkers and symptoms. The primary advantage of using a clustering algorithm is that it does not require re-sampling, imputation, and most importantly, labels.

Data	Total	Admitted to the ICU	Not Admitted to the ICU	Died	Survived
Pathology	837	443	394	517	214
Vital signs	262	31	221	60	202

Table 4.2: Class distribution for the important datasets used in the CBR Module

Aside from the extensive functionality provided by the module, another key component of this system is the development of an intuitive app that clinicians can easily interact with at the point of care. Figure 4.1 depicts a high level view of the system architecture. The following sections investigate the development of each component of the system and explain their inner workings in more detail.

4.3 Back-end

The back-end was developed with a Python framework called Django, which is commonly used as a back-end for web-app development. The Django module interacts with the user via a front-end, developed with HTML, CSS and JavaScript. The requests from the user via the front-end are then used to either query a database or are evaluated using a Machine Learning-based back-end within the framework. The back-end maintains different views for interactions with different algorithms and features of the module. These views include, for example, a view for the application of probabilistic inference as well as a view that allows clinicians to retrieve similar patients based on the patients they select from the view. When the machine learning module is called, the pipelines developed in the research phase of this project are applied to the data, which is then returned via the Django framework to be rendered on the front-end.

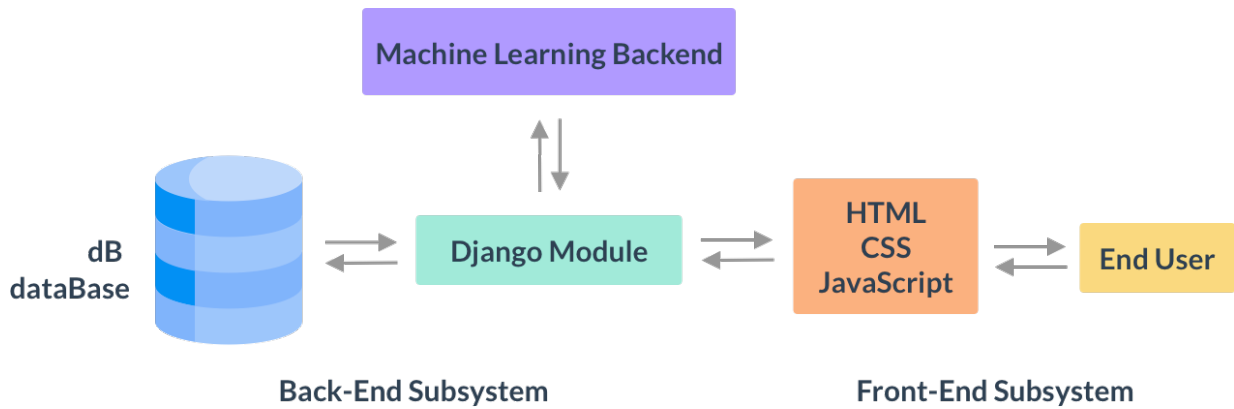


Figure 4.1: Pipeline of High Level Design

4.4 Front-end

The end-user, which is likely to be a clinician or medical care worker, interacts only with the front-end of the system. The front-end encapsulates all of the complexity of the system to present the information and functionality to the end-user in the simplest possible way via a desktop and mobile responsive panel. The User Interface (UI) was developed using CSS, HTML and JavaScript. The front-end has the ability to allow clinicians to customise and choose the way they want to control, view and input data. This includes the ability to fill out forms manually, upload large spreadsheets of cases, query databases for existing patients, and perform risk assessments for patients in terms of COVID-19 infection. The abstraction provided by the front-end removes the complexity of data pre-processing, machine learning and database interactions, to streamline the experience for medical care workers.

4.5 Technologies used

A variety of technologies were used in the development of this CDSS module including but not limited to Python, Sci-kit learn, Pandas, Matplotlib, NumPy, LightGBM, XGBoost, HTML, CSS, JavaScript, Django. float

5 Data Preprocessing

5.1 Data history

Throughout the project several datasets were available. The first data set was the ‘cbr-patient-admission-v03.csv’, accessible from the 01/05/2020 combines the symptoms, radiology results and outcomes (hospitalisation, admission to the ICU, and death) for 170 Covid-19 patients. On the 13/05/2020, new pathology and patient details were made available. They include the first include the results of all the blood test for each patient, and the second includes background information and the outcomes of the patients. The 2 datasets can be merged using the patients’ NHS Number giving a total of 838 patients.

On the 26/05/2020, 4 new datasets became accessible, the ‘bernard-pathology.csv’ dataset, which is similar to the pathology dataset discussed earlier, the ‘keira.csv’ dataset is manually and includes the vital signs, symptoms, medical history, radiology results, results to the Covid-19 test and outcomes of the patients. Additionally, ‘nisha.csv’ dataset includes the treatment of the patients and finally the ‘sid.csv’ which includes a list of patients who were tested positive to Covid-19. These were the dataset that were used for the training and testing for most of the models.

On the 05/06/2020, the ‘tim-damien-tofill-v0.0.1.xls’ dataset was available. It is an updated version of the ‘keira.csv’ dataset merged with the ‘nisha.csv’ (the treatments) dataset. However, the additional recorded patients in the ‘tim-damien-tofill-v0.0.1.xls’ dataset were poorly recorded. Therefore, it wasn’t pre-processed and used to train and evaluate the models.

5.2 Key statistics and class distribution

Using the datasets obtained on the 26/05/2020, it is important to study the overlap of these data sets and class distribution of the important labels.

5.2.1 Probabilistic Inference Module

Table 5.1 shows the distribution of the Covid-19 results that are used for the probabilistic inference module.

Data	Total	Covid-19 positive	Covid-19 negative
Pathology	77351	-	-
Vital signs	1675	976	699
Sid	1919	1919	0
Pathology \cap Vital signs	1186	758	428

Table 5.1: Class distribution for the important datasets used in the Probabilistic Inference Module

At the advice of the supervisors, only the labels of the intersection between Pathology and Vital signs datasets were considered and the "sid.csv" dataset was disregarded, because it was less reliable than the labels in Vital signs dataset.

5.2.2 CBR

Table 5.2 shows the distribution of the patients’ outcome used for the supervised CBR module:

Data	Total	Admitted to the ICU	Not Admitted to the ICU	Died	Survived
Pathology	837	443	394	517	214
Vital signs	262	31	221	60	202

Table 5.2: Class distribution for the important datasets used in the CBR Module

Note that for some rows has missing labels were included, and therefore the sums of the classes don’t always equal to the total.

For the CBR, the vital signs data of set was used (‘keira.csv’, accessible from the 26/05/2020) was used. Rows with no vital signs recorded were disregarded and only the manually recorded rows were kept. For the pathology data, the older version was used (13/05/2020) was used because the number of recorded outcomes for the new pathology dataset is too little.

5.3 Pathology datasets pre-processing

The preprocessing of the pathology datasets that became accessible on the 13/05/2020 and 26/05/2020, is represented by the diagram in Figure 5.1 :

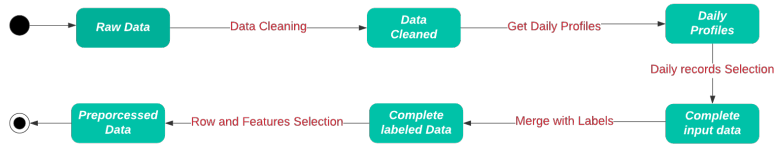


Figure 5.1: Preprocessing of the pathology dataset - MERGING PROCESS SHOULD BE ADDED

5.3.1 Data cleaning

The raw data is composed of one row for each test for every pathology test that was realised. Each row contains the patient’s identifier (‘_uid’), gender (‘GenderID’) and date of birth (‘patient_dob’), the test’s order code (‘orderCode’), date (‘dateResult’), result (‘result’) and abnormality status which indicates if the result to the test in or out of range (‘AbnormalStatus’). Furthermore, some of the rows show the final result and some show a result which should be corrected (‘ResultStatus’).

The raw data is cleaned by first filling the missing values for abnormal status based on the pathology results. ‘AbnormalStatus’ values ranges from -2 to 2, from very low to very high. Then, the values of ‘GenderID’ column are replaced with 0 and 1, for male and female, and the patient’s age is computed. Finally, the rows in which the result has been corrected have been disregarded. The cleaned data is now composed of one row for each realised test. It is composed of the patient’s ID, age and gender as well as the test’s result, abnormal status and data. The status dataset is, therefore, a discretized version of the value dataset.

5.3.2 Getting the daily records for each patient

As discussed previously, the cleaned data is composed of one row for each test. The goal of this step is to combine all the biomarkers (i.e. pathological characteristic) that have been tested for same patient on the same date. Therefore, the new data is composed of the daily record of each patient. Two possible datasets can be outputted, one which shows the exact result of the test (the ‘value’ data) and the other which shows the abnormal status instead (the ‘status’ data). Each of these datasets is useful depending of the algorithm used.

5.3.3 Daily record selection

Since the daily records aren’t long enough to make a times series, one daily record is selected for each patient. Using the results of only one daily record instead of combining them to get the biggest number of biomarkers tested per patient, allows us to have an atemporal algorithm. In other words, if the daily records were combined, all the inputs should have a matched timeline in order to have a robust prediction which is not feasible. Two algorithms were developed. The first selects the first daily record for each patient and is supposed to simulate the pathology results on admission. The second selects the daily record which has the highest number of recorded biomarkers per patient. The second algorithm was the final one that was selected because it has the most complete profiles that can be used to evaluate the models, it also has fewer missing values which needs to be computed and, therefore, it enabled the model to achieve higher accuracy.

5.3.4 Merging the data with the labels

Depending the model used, the data may need or not labels from other data sets. Therefore, the complete pre-processed data is merged with another dataset using the patients ID (‘_uid’) or NHS number (‘NHSNumber’) as discussed in section 5.1.

5.3.5 Row and features selection

In order to decrease the number of missing data, patients with too little biomarkers recorded are disregarded. Furthermore, only the most important features are selected using the methods discussed in section 5.5. This also allows us to decrease the complexity of the models and, therefore, assure a better generalisation.

5.4 Vital Signs dataset pre-processing

The vital signs dataset didn’t need a lot of pre-processing since it was already in the right format. Simple data cleaning was performed before feeding the data to the algorithm. Two main transformations were made. The first is to transform the radiology results to a Boolean, with 0 representing normal results and 1, abnormal results. The second one is to convert the categorical values used for the maximum respiratory support to integers which correspond to their severity based on Table 5.3:

Ventilation Requirements		
Nasal cannula	HC	0
Face mask	FM	1
Venturi	VENTURI	1
Non-rebreather	NRB	2
Optiflow	OPTIFLOW	3
Pos. Airway Pressure	CPAP	4
None invasive ventilation	NIV	5
Intubated	I&V	6

Table 5.3: Ventilation requirements severity according on the medical experts

5.5 Features Selection

Features selection is essential because it firstly reduce the dimension of the data and the complexity of the model built on top of it. This will ensure a better generalisation of the model and the model will be more interpretable.¹¹ Secondly, having less features will result with a higher-quality data since more complete profiles are available. Finally, computing the features importance is essential for CBR model in the computation of weighted distance. Three feature selection methods have been investigated:⁵

1. Filter-based selection
2. Wrapper-based selection
3. Embedded selection

5.5.1 Filter-based selection

The filter-based selection constitutes the first step of the features' selection process. Firstly, features that have been recorded for a limited number of patients were omitted. Then constant, and semi constant features were filtered out. This can be done by disregarding features which have a variance which is too small. Finally, the correlation between the features was calculated to test the independence of the features, and highly correlated features were disregarded.

The main limit of this method is that the it is not aggressive enough; the dimension of the data was still too big. Also, this method didn't give any indication of the feature's importance.

5.5.2 Wrapper-based selection

The wrapper-based searches for the combination of features that produces the highest accuracy. Two algorithms were tested:⁵

1. **Backwards Selection:** It start with all the feature and the most irrelevant features were removed at each iteration and then compute the mean of the model's performance over 100 iterations.
2. **Forward Selection:** It is the opposite than the previous method. It starts with no variable at all and it starts adding features that maximise the model's performance.

This method was disregarded because it didn't suit the context of the data. In fact, the results obtained were highly unstable. It can be explained by the fact that it was mainly used to select the biomarkers of the pathology dataset which are correlated to one another. Also, it was found that a combination of biomarkers provide more information that one biomarker on its own.

5.5.3 Embedded selection – using Random Forest

Embedded Features selection uses algorithms that have built-in features selection methods.¹² Random Forests is an ensemble learning method; it uses multiple decision trees to obtain better predictive performance than could be obtained from any of the decision trees alone by reducing the risk of overfitting. Each tree is constructed using a random sample of the patients and a random sample of features. Each tree is a sequence of yes-no question based on a single feature. At each node, the dataset is split into two subsets which are supposed to represent different classes. Consequently, the importance of each feature is derived from how pure each of the subset is using the 'Gini impurity' metric.

The limit of this method is that highly correlated features are given similar importance. Therefore, in order to optimise the selection, filter-based selection is applied first.

6 Probabilistic Inference

Probabilistic Inference is the process of deriving the probability of one or more random variables taking a specific value or set of values. In this particular context, the objective is to predict whether or not a patient has COVID-19. In this problem setup, a value of 1 or 0 can denote the two scenarios in which a patient either has or does not have COVID. Hence, this is a binary classification problem. Our COVID module uses Probabilistic Inference to estimate the probability that a patient has COVID-19 with custom-designed and optimised pipelines and estimators. To develop an effective classifier, one must consider filtering, imputation, data splitting, re-sampling, scaling, and the estimator. This section describes the experiments and investigation in the development of the solution developed. The investigation was broken down into a series of smaller experiments. The performance of each part of the "pipeline" was then optimised. Finally, to allow clinicians to compare and interpret different algorithms, the three best-performing algorithms were selected.

6.1 Estimators (Binary Classifiers) Research

A variety of estimators can be used as binary classifiers. This section summarises the theoretical and experimental findings for each of the different estimators investigated. Before beginning experimental tests, a theoretical evaluation and comparison of different estimators was conducted. The out-of-the-box performance of some models is better than others because the estimator has a significant effect on pipeline performance. A series of different estimators (binary classifiers) was investigated first before optimising other parts of the pipeline.

6.1.1 Logistic Regression

The logistic regression is a popular machine learning algorithm for binary classification. It is often used as a baseline algorithm to establish a performance baseline. However, while it is simple and quick to implement, it is used frequently in biology and does well for many tasks. A Logistic Regression measures the relationship between the dependent variable (the label), which, in this case, is whether or not the patient has COVID, and the independent variables, also known as features, by using a logistic function to estimate the probabilities. A logistic function, also known as a sigmoid function maps these real-valued numbers onto a value between 0 and 1. The classifier then uses this result to determine the outcome.⁸ The advantages and disadvantages of this estimator are shown in Figure 6.1.

Advantages	Disadvantages
Very efficient to train and implement	Often cannot solve non-linear problems since its decision surface is linear
Highly interpretable	High reliance on proper presentation of the data
Doesn't require input features to be scaled	Not useful unless important independent variables already identified
Easy to regularise	Output is discrete so can only predict a categorical outcome
Outputs well-calibrated predicted probabilities	Vulnerable to overfitting

Table 6.1: Advantages and disadvantages of a Logistic Regression estimator

6.1.2 Naive Bayes

A grid search was not performed for GaussianNB because there are no hyper-parameters to tune, so a grid search cannot be performed. This estimator makes strong independence assumptions, and so initial research showed it was unlikely to achieve high performance. The research and results have been omitted for the sake of brevity.

6.1.3 K-Nearest Neighbors Classifier

K-Nearest Neighbors Classifiers (KNNs) are another simple algorithm that can be used to solve classification problems. The "K" in KNN denotes the number of neighbors in the KNN. It is also a supervised learning algorithm and easy-to-implement, just like the logistic regression. The basic premise of this application of K-Nearest Neighbors is that similar patients would have similar features, while very different patients would have different features. The closer two patients are, the more similar they are. KNNs use this idea and can be used to cluster similar patients by their distance (sometimes called proximity or closeness).

There are many more parameters to tune in this model than the simple logistic regression. For example, the value of K can result in predictions becoming more or less stable. As the value of K decreases toward one,

the predictions become less stable. Conversely, as K 's value increases, the predictions become more stable due to majority voting, so they are also likely to make more accurate predictions up to a certain point. At a certain point if the value of K is pushed too far and, the number of errors increases because the value of K is too large to distinctly distinguish between the classes (in our case, whether or not a patient has COVID).¹⁵ The advantages and disadvantages of this estimator are shown in Figure 6.2.

Advantages	Disadvantages
Simple and easy to implement	Heavier computation load especially with higher dimensionality
No hyper-parameter tuning or additional assumptions needed	High memory requirement
No assumptions about data (no linearity assumption)	Prediction stage might be slow (with big N)

Table 6.2: Advantages and disadvantages of a KNN estimator

6.1.4 Decision Trees

Decision Trees are particularly popular in machine learning, particularly in a biological context. They work by transforming the data into a tree representation, in which each tree node represents a feature, and each leaf node represents a class label.¹⁴ They have a significant advantage over some of the other algorithms in that they are highly interpretable. This is because the decision tree formed can be shown to clinicians to be compared and evaluated using continuously updating medical knowledge and context. The advantages and disadvantages of this estimator are summarised in Figure 6.3.

Advantages	Disadvantages
Minimal pre-processing	Highly sensitive to outliers
Do not require normalization of data	More computationally intensive than other algorithms
Do not require feature scaling	Longer training time
Missing values in the data should NOT affect the process of building trees	
Highly interpretable for technical teams and medical care workers	

Table 6.3: Advantages and disadvantages of Decision Tree estimators

6.1.5 Random Forests

Random forests are a power algorithm in machine learning and especially popular in a biological context. They are based on the ensemble learning technique, also known as bagging.²⁹ This means that they split on a subset of the features on each split. Random Forests extend the basic concepts of decision trees by using bagged techniques.

Bootstrapping is a sampling technique in which one samples with replacement from the dataset.¹⁸ Random forests use Bagging to aggregate bootstrapped results. This means that they create bagged trees by creating many decision trees trained on an equal number of training sets. The value predicted is then taken as the average of all decisions made by the different decision trees. This overcomes the limitation of decision trees, which alone have high variance and tend to over-fit. By bagging and combining many weak learners, the result is a robust learning algorithm through majority voting. The advantages and disadvantages of this estimator are shown in Figure 6.4.

Advantages	Disadvantages
Can handle different feature types so little pre-processing needed	Model interpretability
Low Bias, moderate variance	For large data sets, the size of the trees can take up a lot of memory
Great with High dimensionality	tend to over-fit so require hyper-parameter tuning
Quick Prediction/Training speed	Difficult to optimise hyper-parameters
Good handling of unbalanced data	Long training period and high complexity
Robust to outliers and non linear data	

Table 6.4: Advantages and disadvantages of a Random Forest classifier

6.1.6 Support Vector Classifier

Support Vector Classifiers, also known as SVCs, use Support Vector Machines as classifiers for the patients to determine whether they are COVID-19 positive. They classify the data using a hyper-plane which acts as a decision boundary between the two classes.²³ The extreme data points from each class are called the

Support Vectors. SVCs try to find the optimal hyperplane, which has the maximum margin from each support-vector. The advantages and disadvantages of this estimator are shown in Figure 6.5.

Advantages	Disadvantages
Works well with clear margins of separation between classes	Require feature scaling and long training time
More effective in high dimensional spaces	Ineffective with noticeable noise and overlapping target classes
Effective for high dimensionality	No probabilistic explanation for the classification
Relatively memory efficient and stable in convergence	Choosing an appropriate kernel function is difficult
Regularization capabilities	Extensive memory requirement and Difficult to interpret

Table 6.5: Advantages and disadvantages of Support Vector Classifier (SVC) estimators

6.1.7 Extreme Gradient Boosting with XGBoost

XGBoost is the name of a library with an efficient implementation of the gradient boosting algorithm with high computational efficiency and model performance.⁷ It is, for short, an implementation of gradient boosted decision trees designed for speed and performance.

XGBoost offers a variety of model features including but not limited to

1. Regularised Gradient Boosting with both L1 and L2 regularization.
2. Parallelized tree construction using all CPU cores during training.
3. Cache Optimization of data structures and algorithm to make the best use of hardware.

XGBoost is comparatively very fast when compared to other gradient boosting implementations.⁹ In order to properly understand XGBoost, one must have an understanding of boosting techniques in general. Boosting is an ensemble technique that adds new models to correct the errors made by existing models until no improvements can be made. The most significant limitation of the XGBoost algorithm is that it is a "black box" because it is difficult to understand its inner workings. The black-box nature of the XGBoost algorithm is unfortunate because of its robustness, performance, and success with the problem. This may be an issue in terms of adoption for the medical community because it limits the interpretability⁷ of the model.

6.1.8 Light Gradient Boosted Machine (LightGBM)

LightGBM is another dynamic library of the gradient boosting developed at Microsoft. LightGBM, like XGBoost, is a form of boosting algorithm, in which at the end of the first model, it will identify the error and continue until the error is minimised. In short, it is a highly efficient gradient boosting tree. The algorithm is based on the decision tree algorithm, and it splits the tree leaf wise with the best fit. LightGBMs leaf-wise algorithm can further reduce the loss than boosting algorithms, and this often results in better performance.⁹ Experimentally LightGBM showed several advantages when compared to XGBoost. Firstly, it had noticeably faster training speed and higher efficiency. This is because LightGBM is a histogram-based algorithm that buckets continuous feature values into discrete bins makes the training process faster. Secondly, it has lower memory usage than XGBoost because it replaces continuous values with discrete bins to lower memory usage. While providing better accuracy than most boosting algorithms, generally, it is also capable of performing equally well with large data sets with a significant reduction in training time compared to XGBoost.

Although not used in the context of this project, XGBoost's parallel computing methods could further improve the speed of the algorithm if it were to be deployed for online learning in the future. Experimentally, LightGBM showed to have better accuracy than all other algorithms tested. It achieves this by producing many complex trees by following a leaf wise split approach rather than a level-wise approach, though sometimes this can lead to overfitting (but this can be avoided by setting the `max_depth` parameter).

6.1.9 Other Estimators

The above list is not a full list of the models investigated and tested throughout this project. Aside from the above models, an MLPClassifier, NuSVC, RadiusNeighborClassifier, LSTM, ANN, and a GradientBoostingClassifier were also investigated. As these were less effective models, their results have been omitted for brevity in this report.

6.2 Filtering Techniques Research

This section contains a summary of the research carried out on different filtering methods. After potentially useful estimators were identified, the performance of the pipeline was further improved through investigating other stages, starting with the filtering techniques. For effective generalisability of the model, it is crucial to distinguish between observations that belong to the standard distribution being investigated (i.e., inliers) and those that can be considered exceptions or different from that which is expected (i.e., outliers).

6.2.1 IQR filter

The initial filtering technique used was a simple IQR filter. This filtering method uses the inter-quartile range to filter out outliers based on the spread of the data. It is an effective filtering strategy because it is reasonably robust to slightly skewed distributions and can detect outliers even with small sample sizes. The median and inter-quartile range are more robust to outliers than the mean or standard deviation.

6.2.2 Isolation Forest (IsolationForestFilter)

Random forests are an efficient method of outlier detection for high-dimensional datasets. Using a recursively formed tree structure, different patients will have different path lengths from the root node to the bottom of the tree. By computing an average path length in the forest to each patient, the path length can be used to measure normality for the dataset. Through this approach, anomalies can be identified by shorter paths.

6.2.3 Local Outlier Filter

The Local Outlier Factor method provides another efficient method for outlier detection on high dimensional datasets. This algorithm measures the deviation in the local density of specific points to its neighbors.²³ This can detect samples with significantly lower densities than their neighbors, which can be used to identify outliers because they tend to be more sparse. Typical patients will then have similar local densities to their neighbors, while outliers will have much smaller local densities. The number of neighbors used can be modified in a similar way to a KNN algorithm. The number of neighbors used is chosen to be higher than the minimum number of patients in each cluster, but not too big so that the outliers are included in "local areas." This approach is effective because it considers both local and global properties of the patient database, and the patients are evaluated comparatively with respect to their neighbors, not independently. Unfortunately, this estimator could not be used because it cannot be applied when there are still missing values, but it could be investigated for a later stage in the pipeline in the future.

6.2.4 Elliptical Envelope (EllipticalEnvelopeFilter)

When trying to detect outliers, it can sometimes be efficient to assume that the data comes from a known distribution, either based on prior knowledge or a statistical assumption. This method can identify which data-points fit the distribution, and which are outliers. SciKit-learn has a method, `covariance.EllipticalEnvelope`,²³ that uses an ellipse to identify central data points and filter outliers. Unfortunately, because of the way the pipeline was set up, the isolation forest, local outlier filter, and elliptical envelope could not be used because there were missing values in the data set. For this reason, in the end, the simple IQR filter was used, which proved to be quite effective. Future work could find a workaround for the limitations of the other filters, which theoretically could achieve even better performance.

6.3 Imputation Techniques Research

6.3.1 Uni-variate vs Multivariate Imputation

There are two basic types of imputers: uni-variate and multivariate imputers. In uni-variate imputation, the i -th feature dimension is imputed using only non-missing values in that feature dimension (e.g., `impute.SimpleImputer`). On the other hand, a multivariate imputer uses the entire set of available feature dimensions to estimate the missing value (e.g., `impute.IterativeImputer`).

6.3.2 Uni-variate Feature Imputation

The `SimpleImputer` class imputes missing values for uni-variate feature imputation. Depending on preference, the missing values can either be imputed with constant values or statistical metrics like the mean, median, or mode of each feature.

6.3.3 Multivariate Feature Imputation Using Iterative Imputation

The `IterativeImputer` class²³ provides a more sophisticated imputation approach because it models each feature with missing values as a function of the others and uses that estimate for imputation. Iteratively, a feature column is designated as the output y , and the other feature columns are treated as inputs X at each step. A regression is then fit on (X, y) for known outputs y , and the regressor then predicts missing values. The `max_iter` can be used to specify the number of imputation rounds for this process to repeat itself. It may also be worth investigating multiple imputations, which is often used in the statistics community but is not discussed here.

6.3.4 Nearest Neighbors Imputation using KNN

Missing values can also be imputed using a k-Nearest Neighbors approach using sci-kits `KNNImputer` class.²³ It offers support for missing values and imputes values for each missing feature based on the k nearest neighbors (specified via the `n_neighbors` parameter). If there are multiple missing features, then the features can be imputed from different neighbors, depending on the feature being imputed.

6.4 Re-sampling Techniques Research

Re-sampling is a widely adopted technique for dealing with highly imbalanced datasets. It works by removing samples from the majority class (under-sampling) and adding more examples from the minority class (over-sampling), as shown in the figure. While this sounds like a care-free and straightforward way to improve the performance and generalisability of models, some disadvantages come with balancing classes. The simplest form of over-sampling is to duplicate random values from the minority class. The danger here is that this can cause overfitting. The most straightforward technique within under-sampling involves removing random records from the majority class, which can reduce the amount of information available to the model. The following is a brief description of the theoretical background and implementation of the different re-sampling techniques investigated.

6.4.1 RandomUnderSampler

The `RandomUnderSampler` from `imblearn` under-samples the majority classes by randomly picking samples with or without replacement.⁶ Random Under-Sampling involves randomly selecting examples from the majority class and removing from the dataset to balance the training set. This approach reduces the number of patients used for training, which can have negative repercussions on model training because it reduces the number of available examples for training. This approach would be more appropriate if there were a large number of examples, even in the minority class, which is not the case in the datasets used for this project.

This is a significant limitation for random under-sampling. Removing examples means that potentially valuable information, perhaps critical to a decision boundary, could be removed, affecting the generalisability of the model. Since the patients are removed randomly, there is no way to distinguish between whether or not the removed patients are information-rich. This means vast quantities of potentially useful information could be discarded, making the learning process more difficult and less reliable for small datasets.

6.4.2 RandomOverSampler

Random Oversampling is similar to Random Under-sampling, except it randomly duplicates examples in the minority class, resulting in a balanced and larger training set than under-sampling, which randomly removes examples from the majority class. In other words, Random oversampling involves randomly selecting examples from the minority class, with replacement, and adding them to the training dataset. Although experimentally, it proved to be the most effective re-sampling method. It is sometimes referred to as a naive re-sampling method because it does not make any assumptions about the data's structure and uses no heuristics. This makes it quick and straightforward to implement, which is very useful for large datasets like those involved in this project but does not take advantage of potential prior knowledge or more advanced techniques.

This technique is highly effective because the distribution of our dataset is highly skewed as the majority of patients had COVID, and there were few negative patient examples.⁶ However, the danger of this type of re-sampling is that the resulting models will over-fit the minority class, leading to an increased generalization error and increasing the likelihood of overfitting. This issue did not seem to be prevalent as the system was evaluated with 5-fold cross-validation.

6.4.3 SMOTE (Synthetic Minority Oversampling Technique)

SMOTE is another type of data augmentation. SMOTE is an abbreviation for Synthetic Minority Oversampling Technique, and it involves synthesizing new examples from existing examples in the minority class. SMOTE works by selecting an instance of the minority class at random and finding the k nearest neighbors in the minority class. It then creates a new synthetic patient by choosing one of the k nearest neighbors at random and generating a synthetic at a random point between the two examples in the feature space. This approach is particularly effective because the new synthetic examples generated from the minority class are realistic and reflect the structure and trends of the minority class. The disadvantage of synthetic minority oversampling techniques is that they are less effective when there is an overlap between the majority and minority classes in the feature space. Experimentally, this did not show to be a significantly noticeable issue when SMOTE was applied as a re-sampling technique. This may, however, be the reason why SMOTE performed marginally worse than random oversampling (which is generally considered more simplistic and less effective because it just duplicates patients instead of creating synthetic ones).

6.4.4 TomekLinks

Tomek Links are pairs of very close instances but of different classes. TomekLinks is a method of re-sampling that refers to identifying pairs on the edges between two classes.⁶ By removing one or both of the examples on the borderline between the majority and minority class, the decision boundary between the two classes becomes less noisy and ambiguous, improving the model's generalisability and performance. This method did not show any noticeable performance improvements during the experimental phase of this project.

6.4.5 SMOTETomek

SMOTETomek is a combination of SMOTE and Tomek Links.⁶ It uses a combination of oversampling and under-sampling methods to provide an effective re-sampling technique. Firstly, the SMOTE method is applied to over-sample the minority class to create a balanced dataset. Then the Tomek Links technique is used to increase the decision boundary between the two classes. This method also did not show any noticeable performance improvements during the experimental phase of this project.

6.5 Feature Scaling Techniques Research

The final step in this investigation is an experiment that evaluates the performance improvements offered by different feature scaling techniques. Feature scaling is the final stage of the pipeline before the estimator, and so it has the potential to result in significant performance improvements. The following is a brief evaluation of the different feature scaling techniques investigated.

6.5.1 StandardScaler

The StandardScaler used up to this point removes the mean and scales the data to unit variance.²³ However, this means that the outliers have an influence when computing the empirical mean and standard deviation, which shrinks the range of the features. The outliers on each feature have different magnitudes, and this results in a vastly different spread of the transformed data. This means that the StandardScaler is limited and cannot guarantee balanced feature scales in the presence of outliers. The StandardScaler was also tested with different metrics such as the most frequent value, the median, and the mean.

6.5.2 MinMaxScaler

The MinMaxScaler re-scales the data so that all feature values are in the range $[0, 1]$.²³ However, this compresses all inliers into a narrow range, and so it is susceptible to the presence of outliers, like the StandardScaler.

6.5.3 MaxAbsScaler

The MaxAbsScaler is different from the MinMaxScaler because the absolute values are mapped in the range $[0, 1]$.²³ When the data is only positive, this scaler behaves very similarly to the MinMaxScaler and is so also very sensitive to the presence of outliers. The context on this outlier suggests that it is unlikely to perform well within our pipeline.

6.5.4 RobustScaler

The RobustScaler, unlike the previous scalers, centers and scales the statistics based on percentiles and therefore is not influenced by large marginal outliers.²³ As a result, the range of the transformed feature

values is larger than for previous scalers and, more importantly, is approximately similar for all features. The advantage of this is that the outliers are still present in the transformed data. A non-linear transformation would be required for outlier clipping (though this was already done in the first pre-processing step in the pipeline).

6.6 Analysis of Results

The performance of 10 estimators was compared against different datasets using a basic pipeline combination. In the pre-processing phase, the pathology data was pre-processed in four different ways as part of the investigation. The pathology dataset, which had different rows for each pathology test, is transformed into a new data frame. Each row indicates a specific patient, and the results from the different tests become the columns for each patient. The admission datasets (value and status) use the bio-marker tests recorded for each patient at the point of admission. The best datasets (value and status) use the day for each patient, where the highest number of pathology biomarkers was recorded. Experimental testing showed that categorically encoding each of the bio-markers into the levels - very low (-2), low (-1), normal (0), high (1), and very high (2) - and using the "best" dataset, where the day where the highest number of biomarkers for each patient was recorded, resulted in the best performance. The following figures summarise the results and analysis obtained using the Best Status dataset.

Figure 6.1 shows the results obtained for the ROC and AUC score for the different estimators tested with the Best Status dataset and no hyper-parameter optimization. From this performance metric, the Radius Neighbours Classifier is the only noticeably worse classifier. All others seem to have achieved similar performance, except for XGB and the Random Forest, which performed slightly better "out-of-the-box" without any hyper-parameter optimization.

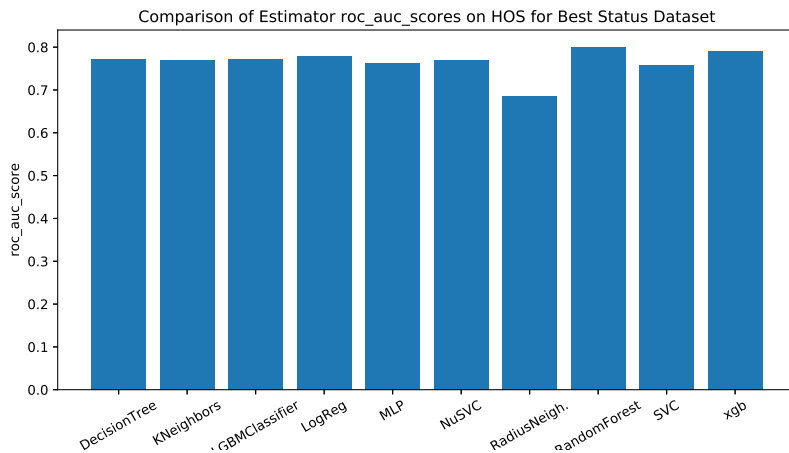


Figure 6.1: ROC and AUC Score for different estimators with Best Status dataset and no Hyper-parameter optimization

Figure 6.2 shows the results obtained for the ROC and PRAUC score for the different estimators tested with the Best Status dataset and no hyper-parameter optimization.

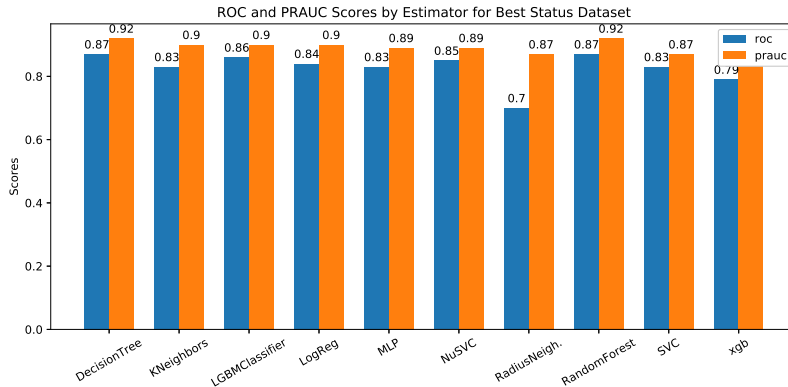


Figure 6.2: ROC and PRAUC Score for different estimators with Best Status Dataset and no Hyper-parameter optimization

Figure 6.3 shows both the sensitivity and specificity obtained for different estimators with no hyper-parameter optimization. These metrics are particularly important for evaluating the performance of different estimators because they can indicate of overfitting. Sensitivity, within a medical context, can be defined as the ability to correctly identify a positive case (in this case of COVID), or in other words, the true positive rate. Specificity similarly represents the ability of the model to correctly identify which patients do not have COVID (also known as the true negative rate). This is particularly important because the raw accuracy of a model does not always reveal very much and can be misleading. This means that sensitivity is the probability of a patient being COVID positive when the disease is present. Specificity is the probability of a patient being COVID negative when the disease is not.

Sensitivity and specificity are inversely proportional, which means that as the sensitivity increases, the specificity decreases conversely. An effective estimator and pipeline need a high sensitivity and specificity to be as useful as possible in a clinical application.

The worst sensitivity was achieved with the Radius Neighbours classifier. It proved to be particularly ineffective, and so it was discarded from further testing in later stages of the research phase of this project. The MLP and NuSVC were also disregarded from further testing because the test scripts took quite a long time to run. Reducing the number of estimators to test decreased the execution time of the testing scripts in later stages (which took several hours to run).

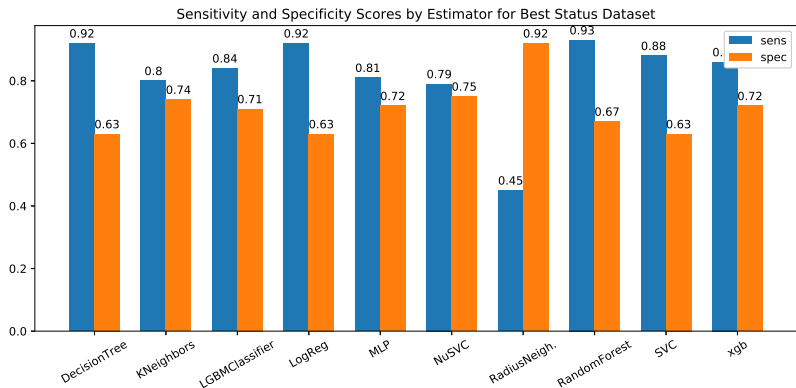


Figure 6.3: Sensitivity and Specificity for different estimators with Best Status Dataset and no Hyper-parameter optimization

Figure 6.4 shows the training and testing accuracies obtained for the different models. These can sometimes be used to diagnose models for overfitting if the training accuracy is very high, and the testing accuracy is comparatively low. From the figure, it is clear that the NuSVC is overfitting and so it was left out of further testing because it also had noticeably long training times. Although the XGB estimator has a high training accuracy, this is characteristic of boosting models, and although it is overfitting, it still achieves both good specificity and overfitting.

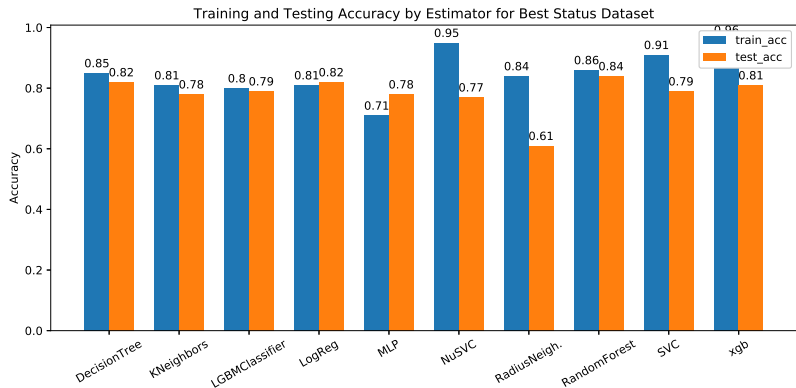


Figure 6.4: Training and Testing Score for different estimators with Best Status Dataset and no Hyper-parameter optimization

Figure 6.5 shows the log-loss for the different estimators tested with the best status dataset and no hyper-parameter optimization. From the figure, it is again clear that the worst classifier tested was the Radius

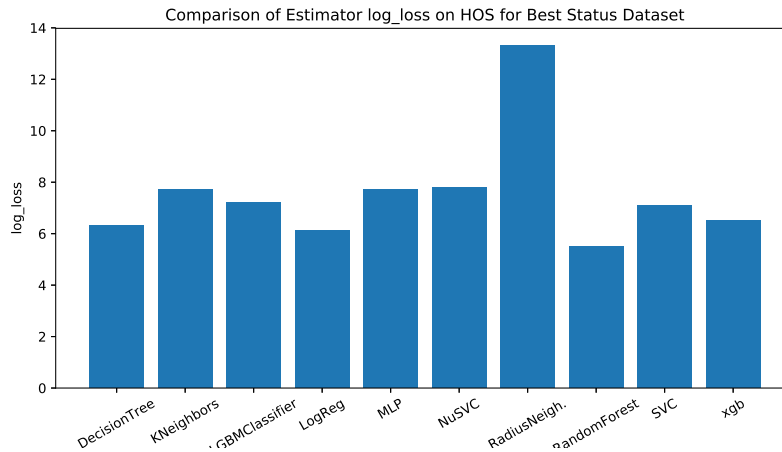


Figure 6.5: Log-loss for different estimators with Best Status Dataset and no Hyper-parameter optimization

Based on this initial analysis, further optimization of the pipeline was continued with a reduced set of estimators by first running grid searches to tune the filter, imputer, re-sampling method, and feature scaling. The results obtained for the best models from each grid search from each optimization experiment can be found in the appendix.

6.7 Conclusion; Final Design and Implementation

In the final design passed to the Django application from this probabilistic inference research, three models were used. This allows clinicians to test different models with different unique advantages. The best of these pipelines used a Light Gradient Boosting Machine (LightGBM), followed by the pipeline with the Random Forest and then the Decision Tree-based pipeline. While the LightGBM and Random Forest offer some performance improvements, the decision tree classifier was included for its transparency to clinicians, and all three pipelines achieved accuracies of approximately 80%. Within a clinical decision support system, clarity, and transparency on patient infection decisions and risk assessments.

6.7.1 Light Gradient Boosting Machine

Figure 6.6 shows the ROC curve, PR curve, Calibration curve, and threshold curves for the hyper-parameter optimized (via a grid search) Light Gradient Boosting Machine estimator pipeline.

The ROC curve, or Receiver Operating Characteristic curve, is a plot of the true positive rate against the false-positive rate. The line in grey indicates luck with random guessing, and models that perform well will have as high a ROC area as possible. From the ROC curve, one can investigate the trade-off between sensitivity and specificity because an increase in sensitivity will result in a corresponding decrease in specificity. The closer the curve is to the left-hand border, and the top left corner of the plot, the more accurate the estimator. A very poor estimator would lie on or close to the line in grey, indicating on the figure. The area under the curve can be used as a rough indication of accuracy. In this case, the ROC AUC is 0.92, or 92%. An area of 90% or above is generally indicative of excellent performance (a perfect test would have an area of 1). To be more precise, the area under the curve measures the discrimination of the model, which is its ability to classify patients, both with and without the disease correctly.

The PR curve, or Precision-recall curve, plot the positive prediction value (PPV) on the y-axis against the true positive rate (TPR) on the x-axis. These quantities can be defined, as shown in figure 6.7. PR curves are slightly less useful for model evaluation because they do not consider true negatives, but they can still be used to evaluate the model's sensitivity. They are often an effective alternative to the ROC curve because the ROC curve is often misleading for visual interpretation with imbalanced datasets. An ideal classifier would have an AUC PR of 1, and again 90% is a clear indication of excellent performance. A completely random classifier would, in theory, have an AUC PR score of 0.5, so poorer classifiers have a smaller area under the PR curve. An imbalanced dataset or ineffective model could have a much better ROC curve than the PR curve, but this is not the case here. This is another indication of effective model performance.

Calibration curves can be another powerful metric for evaluating the performance of classification models. In machine learning, most classification models produce outputs as predictions of outcomes with probabilities between 0 and 1. If a model is well-calibrated, then if the test samples were placed into bins based on their

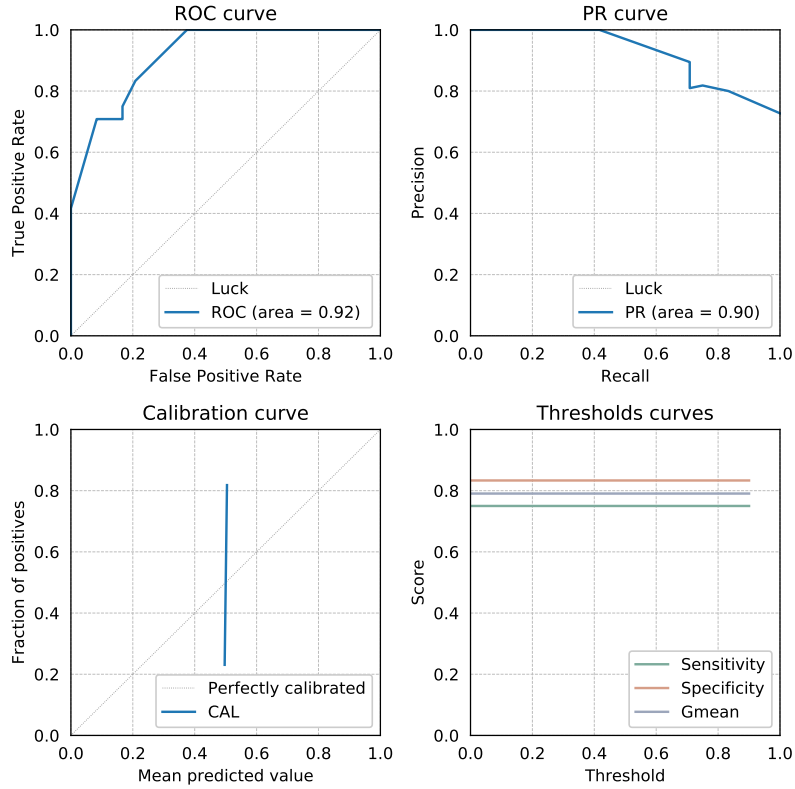


Figure 6.6: Performance evaluation of Light Gradient Boosting Machine Model

$$\text{precision} = PPV = \frac{TP}{TP + FP} \quad (6.1)$$

$$\text{recall} = TPR = \frac{TP}{TP + FN} \quad (6.2)$$

Figure 6.7: Equations for Precision and Recall

predicted probabilities, each bin's true outcome has a proportion close to the probabilities in that bin. Therefore, a well-calibrated model would have a line close to $y = x$, meaning that the true positive rate is proportional to the mean predicted value. The calibration plot is likely suffering because there are too few data points in the hold-out-set, and so one misclassification in a small bin can significantly change the fraction of positives after cross-validation.

The threshold curves show the value of the sensitivity, specificity, and Gmean for different thresholds. These values should not be constant, so it is unclear why this is the case in this plot. This sub-figure should be re-investigated and evaluated. Further, the performance of all metrics can be significantly improved if more data is available for training and testing. A larger hold-out set would result in improved performance. Based on the limited data available, this was the best performing pipeline tested. The final pipeline configuration used to generate figure 6.6 is shown in Figure 6.8.

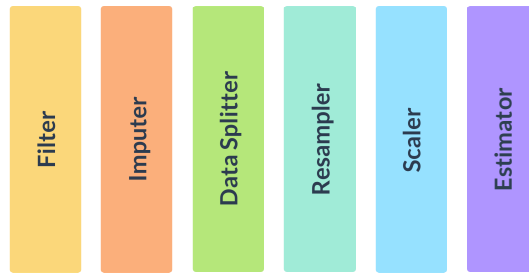
6.7.2 Random Forest Classifier

Figure 6.9 shows the ROC curve, PR curve, Calibration curve, and threshold curves for the hyper-parameter optimized (via a grid search) Random Forest estimator pipeline.

The ROC curve in figure 6.9 similarly shows a high ROC AUC of 0.90, which is indicative of excellent and comparable model performance. The random forest can be slightly faster to train than the Light Gradient Boosting Machine, which can be an essential feature if this model is to be used for online learning in clinical practice. As the plot shows, the ROC curve is close to the top-left corner, though it should be noted that the Hold Out Set should have been larger for a more interpretable set of plots. This is the reason why the ROC curve looks somewhat disjointed (rugged).

The PR curve in the figure similarly shows a PR AUC of 0.90, indicating comparable model performance

LightGBM



Probabilistic Inference

```

IQRFilter()
SimpleImputer(strategy='median')
StandardScaler()
RandomOverSampler()
    
```

Figure 6.8: Final Light Gradient Boosted Machine Model Architecture

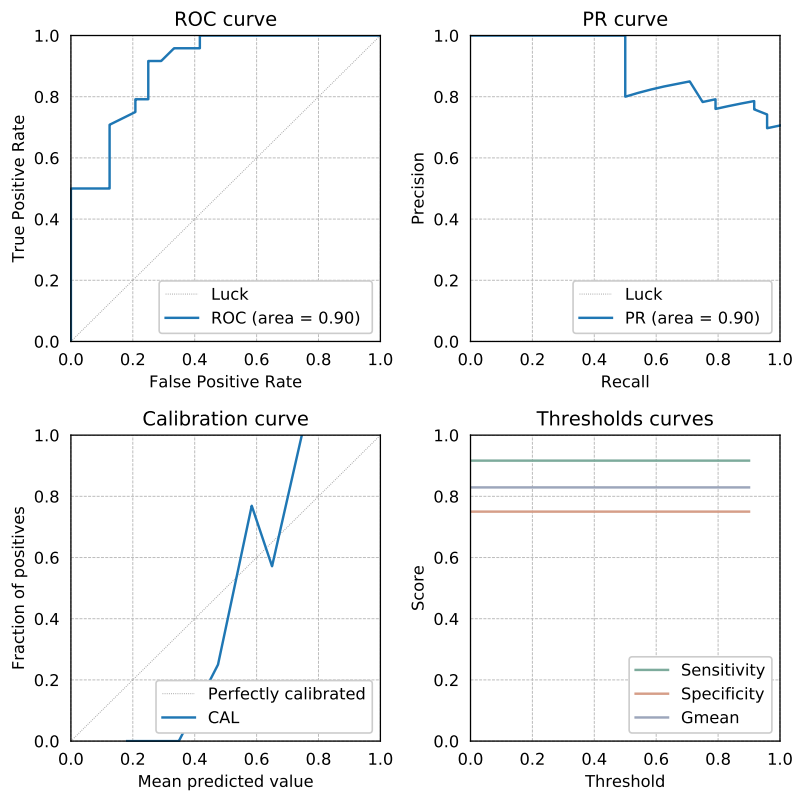


Figure 6.9: Performance evaluation of Random Forest Model

to the Light Gradient Boosting Machine. The line appears somewhat disjointed for the same reason as the ROC curve, and future work should use a larger Hold Out Set to evaluate the model's performance. The calibration curve is better for the random forest. This is an indication of excellent performance because the curve is closer to being perfectly calibrated and somewhat follows a proportional trend between the fraction of positives and the mean predicted value.

The threshold curves are unusual and constant, as mentioned before in the previous case and so this should be investigated further to evaluate the model's performance. From this plot, the mean sensitivity, specificity, and Gmean score can at least be seen and are all relatively good. The sensitivity being noticeably higher than the specificity may be an indication of overfitting. Nonetheless, the Gmean score shows that the overall accuracy of the model is marginally over 80%. Unfortunately, like the LightGBM, the random forest is also difficult for clinicians to interpret because it operates much like a black-box, where its inner workings are difficult to understand. This can have ethical consequences if the model is not performing well in the field. The clinicians may also be less willing to rely on the results generated by the clinical decision support system if they cannot explain or understand where they came from. The final pipeline configuration used to

generate Figure 6.9 is shown in Figure 6.10.

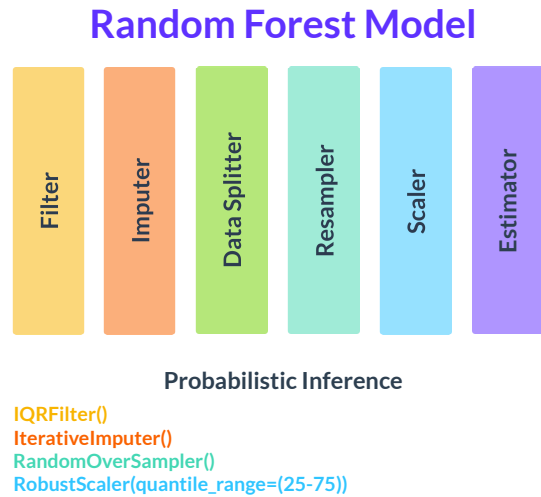


Figure 6.10: Final Random Forest Model Architecture

6.7.3 Decision Tree Classifier

Figure 6.11 ROC curve, PR curve, Calibration curve and threshold curves for the hyper-parameter optimised (via a grid search) decision tree estimator pipeline.

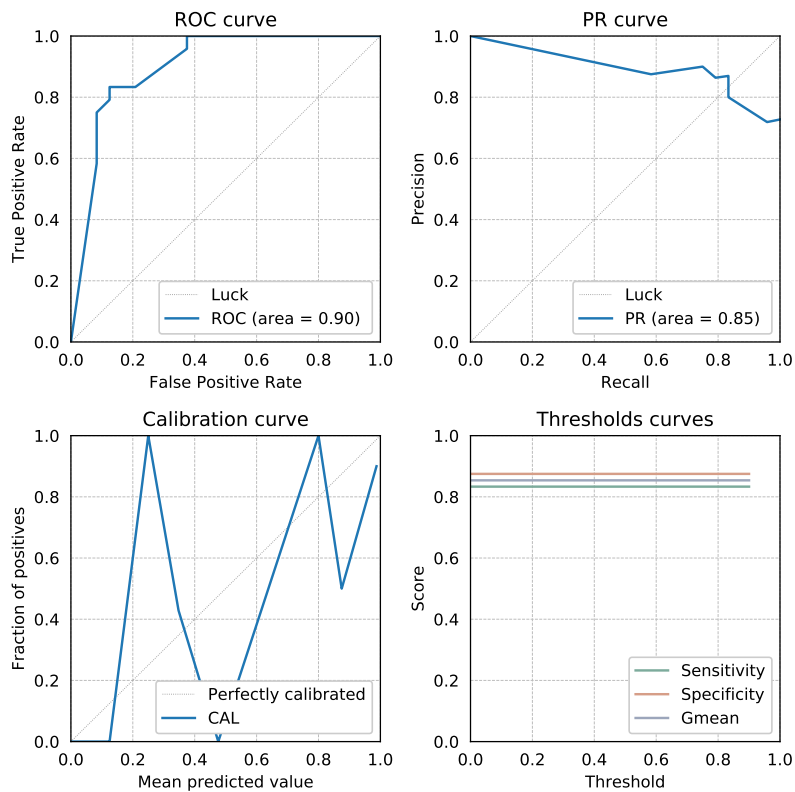


Figure 6.11: Performance evaluation of Decision Tree Model

The ROC curve in figure 6.3 shows that the curve is close to the top left corner and the left-hand side of the plot, as well as having a ROC AUC of 0.90. Again, this is an indication of excellent model performance as the line is far from the grey line, indicating luck on the plot. The PR curve is slightly worse for the Decision Tree Classifier and achieves a PR AUC of 0.85. This is still a high value for this performance metric, but it is marginally worse than the previous two models.

The calibration curve in figure 6.3 looks much more erratic and unpredictable. This may be because the decision tree is going to be very sensitive to each example in the small Hold Out Set used.

The threshold curve is again unusual because the sensitivity, specificity, and Gmean are all constant for

the thresholds. The values correspond to those computed as a mean of all five folds of cross-validation (from the "summean" file generated in our pysml repository). All three performance metrics are above 80%, which is again an indication of effective model performance. The sensitivity and specificity are much closer to the decision tree, which is potentially an indication of even better performance than the Random Forest. The biggest advantage offered by the Decision Tree Classifier is its transparency to clinicians. As the Decision Tree is constructed as a series of decisions made on the features passed into the model, this tree can be used and interpreted by clinicians. It gives a transparent and easy-to-understand view of the model's inner workings. As previously discussed, this can be useful for a model to increase its adoption in clinical practice. The final pipeline configuration used is shown in Figure 6.12.

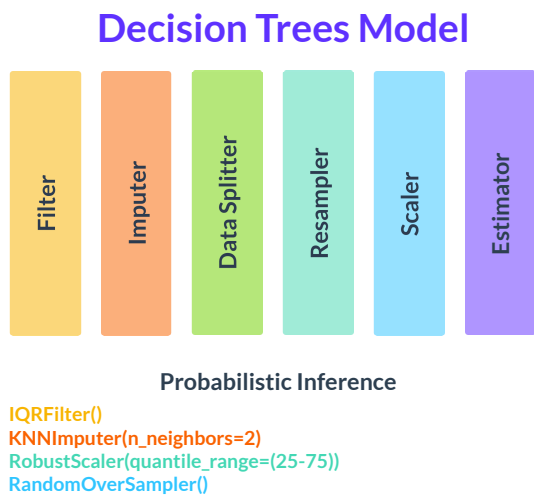


Figure 6.12: Final Decision Tree Model Architecture

7 Case Base Reasoning

The Case-based reasoning (CBR) is a methodology that uses previous experience in the form of a set of examples, denoted as case-base, in order to understand and solve new cases.²² To do so, for a specific new query the following steps are taken: searching and retrieving similar previously solved cases, adapting the solution to the current inquiry, evaluating and storing the newly solved case. This process is the CBR working cycle and it is illustrated in Figure 7.1.

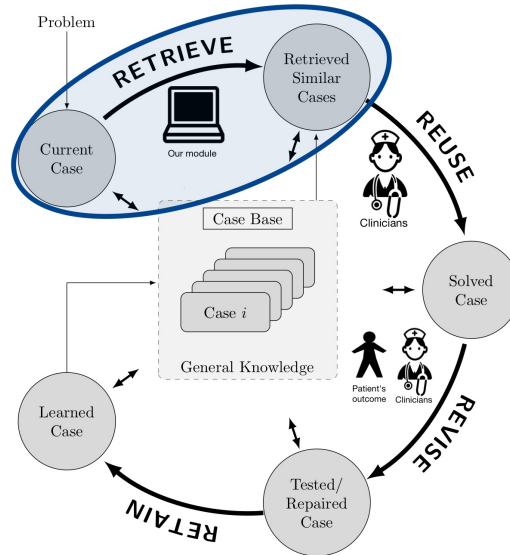


Figure 7.1: The CBR cycle - the 4 phases of the CBR cycle according to Aamodt and Plaza¹⁶

According to Aamodt and Plaza,⁴ the CBR working cycle is composed of four processing stages:

1. **RETRIEVE** the most similar cases by comparing the new patient to case-base using a preselected similarity or distance metric and selecting the most similar ones.
2. **REUSE** the case to attempt to solve the current query. This phase is one of the most challenging and it is going to be done by clinicians and medical experts.
3. **REVISE** the proposed solution. Based on the evolution and outcome of the patient's health, experts revise and correct possibly wrong solutions
4. **RETAIN** the new solution and add it to the case-base if it is useful.

The implementation is going to focus only the retrieval of similar patients and provide them to the clinicians to support them with treatment prescription. The patients' retrieval is going to be done using the dataset that contains patient's pathology profiles and the dataset that contains the vital signs, symptoms, radiology results and medical history of the patients. For the former, a supervised and an unsupervised method were used, while for the latter only the supervised methodology was followed.

7.1 Supervised CBR

7.1.1 Introduction

In the supervised learning CBR, a labelled dataset is used to train evaluate the performance of the CBR. The workflow of the CBR for the two datasets is represented in Figure 7.2.

First, the processing of the Case-base is done in a similar way as for the Probabilistic inference module. The data was not sampled because this would mean that some of the retrieved patients would be artificially made examples which is not desirable in the problem's context. Then, the new inquiry is also processed to match the format of the examples in the case base. Afterwards, the weighted distance between the new processed inquiry and the rest of case-base is computed using the weights importance found in during the features selection using random forest (Section 5.5)

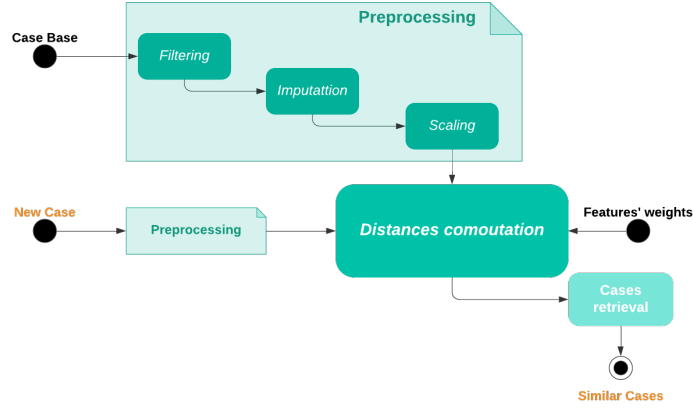


Figure 7.2: Workflow of the supervised CBR used for both the pathology and the vital signs datasets

7.1.2 Evaluation technique

Different metrics, filters, imputers and scalers were taken into account before selecting the final model using a supervised method. In fact, the patient’s outcomes (i.e. ICU admission and Death) were considered as binary labels in order to evaluate the performance of different models. These labels were used in order to compute the importance of the features.

In order to evaluate the quality of the patients retrieved, a simple voting system was used to predict the outcome of the patients. The percentage of retrieved patients that have been admitted to the ICU was compare to a predefined threshold. In fact, after observing that the classes are highly imbalanced for the vital signs’ dataset (Table 7.1), the threshold has been lowered to 0.3 for ICU and 0.4 for death.

Data	Total	Admitted to the ICU	Not Admitted to the ICU	Died	Survived
Pathology	837	443	394	517	214
Vital signs	262	31	221	60	202

Table 7.1: Class distribution for the important datasets used in the CBR Module

7.1.3 Distance metrics

The following distance metrics were considered:¹⁶

Manhattan Distance

It computes the sum of the absolute differences. In other words, it computes the $L_1 - norm$ of the difference, the Minkowski distance with $p=1$.

$$d^{(w)}(u, v) = \sum_i w_i |u_i - v_i| \quad (7.1)$$

Euclidean Distance

It computes the natural distance between two points in the euclidean space. In other words, it computes the $L_2 - norm$ of the difference, the Minkowski distance with $p=2$.

$$d^{(w)}(u, v) = \left(\sum_i w_i |u_i - v_i|^2 \right)^{1/2} \quad (7.2)$$

Chebyshev Distance

It computes distance between two vectors as the greatest if the differences along any dimension. In other words, it computes the $L_\infty - norm$ of the difference, the Minkowski distance with $p=\infty$.

$$d^{(w)}(u, v) = \max_i w_i |u_i - v_i| \quad (7.3)$$

Minkowski Distance

It computes distance between two points in the normed vector space. . In other words, it computes the L_p - norm of the difference.

$$d^{(w)}(u, v) = \left(\sum_i w_i |u_i - v_i|^p \right)^{1/p} \quad (7.4)$$

Canberra Distance

It computes a weighted version of the Manhattan distance. It is biased for data around the origin and very sensitive for values close to 0.

$$d^{(w)}(u, v) = \sum_i w_i \frac{|u_i - v_i|}{|u_i| + |v_i|} \quad (7.5)$$

Bray Curtis Dissimilarity

It computes the differences between samples based on abundance or count data. It ranges between 0 and 1.

$$d^{(w)}(u, v) = \sum_i w_i \frac{|u_i - v_i|}{|u_i + v_i|} \quad (7.6)$$

Two other dissimilarity metrics were soon disregarded:

- **Cosine similarity** which computes the cosine of the angle between the vectors. Therefore, it quantifies the difference based on the angle rather than the magnitude which doesn't fit in the problem's context.
- **Correlation** which compares shape of the response over time, rather than comparing the profiles.

7.1.4 Results and discussion

In this section investigates the impact of the distance metric, filtering, imputation, scaling on the performance of the model in predicting the patient's outcome for each dataset by analysing the ROC, PR and Calibration Curves (explained in section 6.7.1

Distance Metrics

The impact of choosing different distance metrics to predict the outcome of the patient (i.e. admission to the hospital and death) is presented in Figures 7.3 and 7.4 for both the vital signs and pathology datasets.

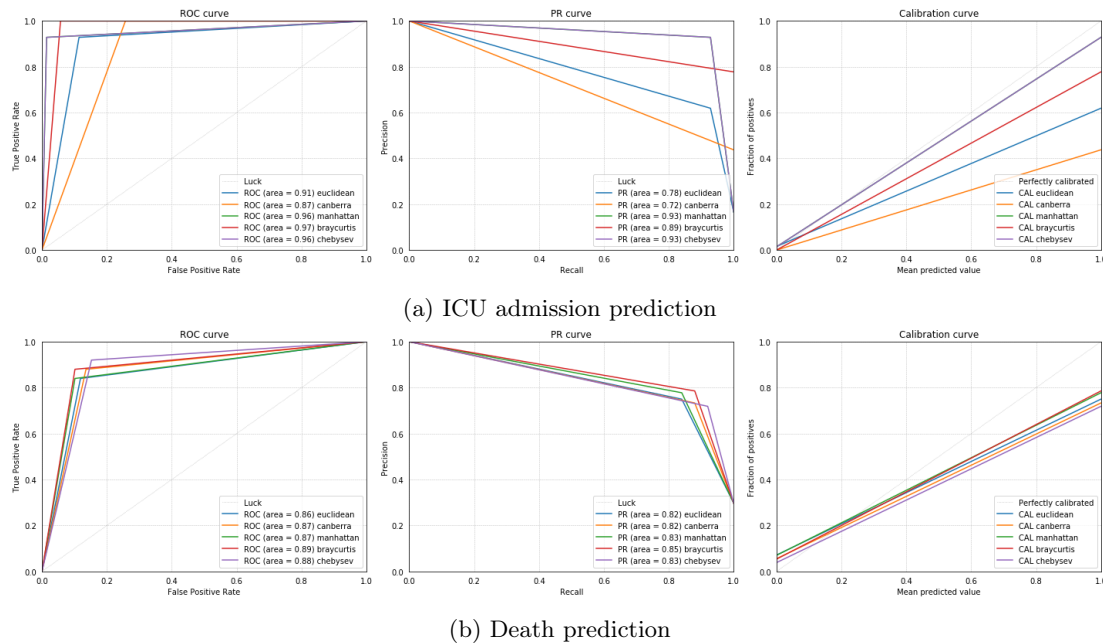


Figure 7.3: Impact of different distance metrics on the performance of the CBR module in predicting the outcome of the patient based on the vital signs dataset

For the vital signs dataset (Figure 7.3, both the models that uses Bray-Curtis dissimilarity and the Chebyshev distance performed the best to predict the patients' admission to the ICU. The model that uses Chebyshev distance has the best performance to predict the ICU admission since it has a high ROC AUC and PR AUC, and its calibration curve shows that the model will have a good ability to generalise. However, the model using the Bray Curtis dissimilarity has a better overall performance when the prediction of the possible death of the patient is taken into account. Therefore, the Bray Curtis dissimilarity was chosen to compute the distance between the profiles.

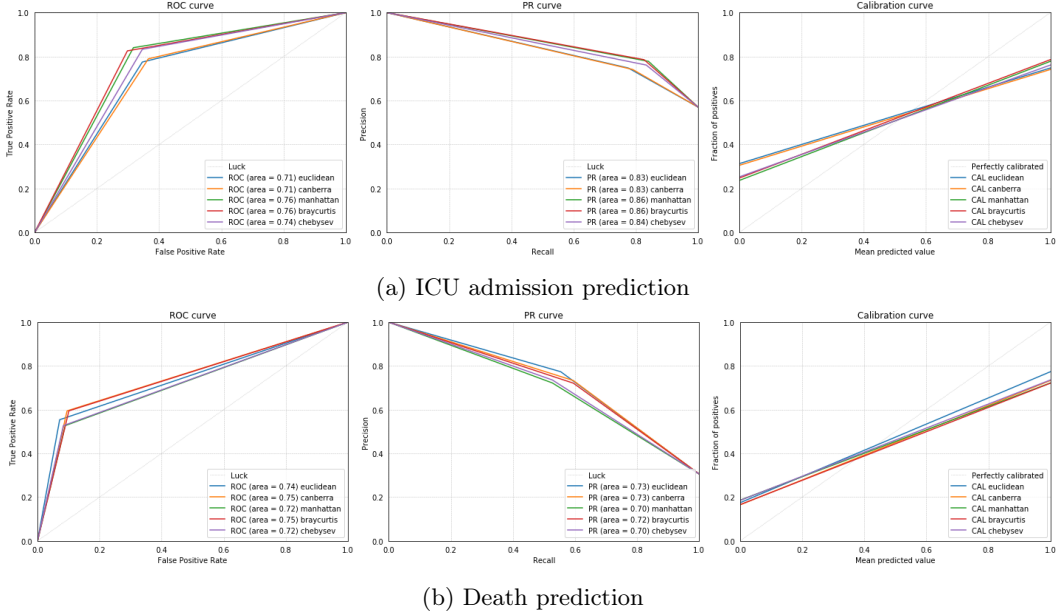


Figure 7.4: Impact of different distance metrics on the performance of the CBR module in predicting the outcome of the patient based on the pathology dataset

For the pathology dataset (Figure 7.4, the different distance metrics has less impact than for the vital signs dataset. However, the Bray Curtis dissimilarity has a consistent higher overall performance and was chosen to compute the distance between the profiles. The euclidean distance metric poor performance is due to the high dimensionality of the dataset (13) and the Chebyshev distance's one is due to its restrictiveness.

Filtering

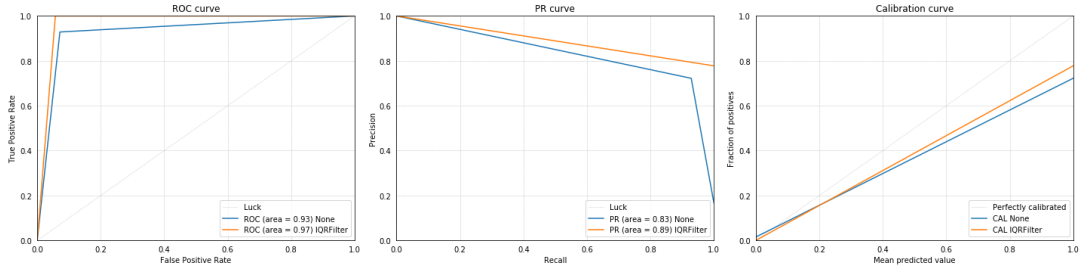
The impact of choosing different filtering discussed in Section 6.2 methods to predict the outcome of the patient (i.e. admission to the hospital and death) is presented in Figures 7.5 and 7.6 for both the vital signs and pathology datasets.

The filtering the data using the IQR Filter with a coefficient of 1.5 has a different impact on the two datasets which can be explained by their difference in the collection method and context. For instance, the vital signs dataset (Figure 7.5) has been collected manually which can induce some human error and the dataset suffers from sample bias since only the patients that were hospitalised were recorded. Moreover, serious cases have a higher probability of being recorded than a mild ones which also includes extreme cases. Therefore, the data set will be skewed and the prevalence of outliers is more important. This explain why using the IQR Filter has increased the model's performance. On the other hand, the pathology dataset (Figure 7.6) was collected automatically from the NHS dataset and has better distribution between different classes, which is why the filtering was as beneficial.

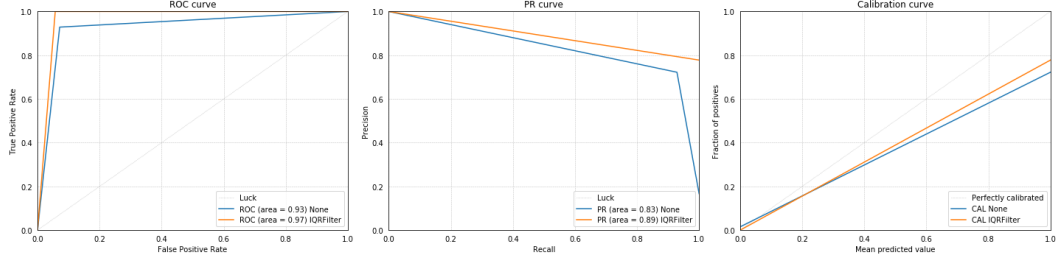
Imputation

The impact of choosing different filtering discussed in Section methods 6.3 to predict the outcome of the patient (i.e. admission to the hospital and death) is presented in Figures 7.7 and 7.8 for both the vital signs and pathology datasets.

The different imputations technique has a different impact on the two datasets. In fact, the vital signs dataset (Figure 7.7) has a total of only 6 features, presented in Section 7.1.5, that are little correlated to one another. Therefore, the missing data cannot be computed accurately using the values of the other features, thus a Uni-variate approach (i.e. imputing using the median) matches better this context. On the other hand, the pathology dataset (Figure 7.8) has a total of only 13 features, presented in Section 7.1.5, which are correlated to one another since it represent a set of connected bio markers (pathological or physiological

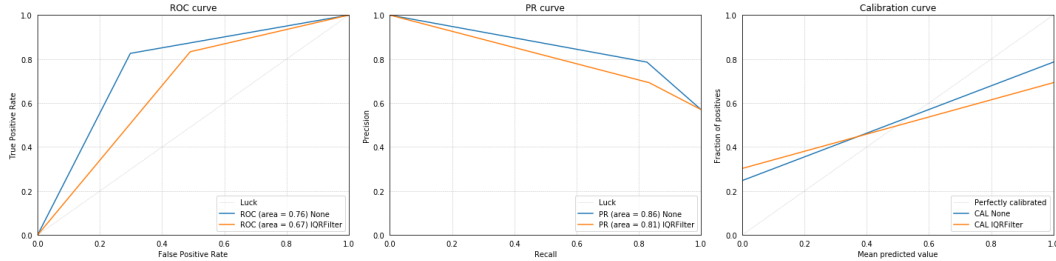


(a) ICU admission prediction

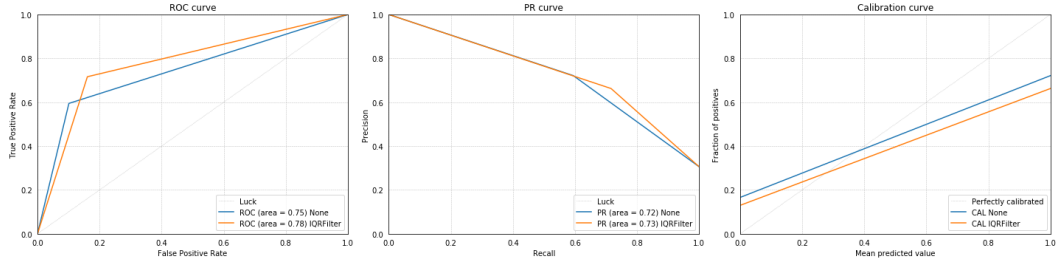


(b) Death prediction

Figure 7.5: Impact of different filtering methods on the performance of the CBR module in predicting the outcome of the patient based on the vital signs dataset



(a) ICU admission prediction



(b) Death prediction

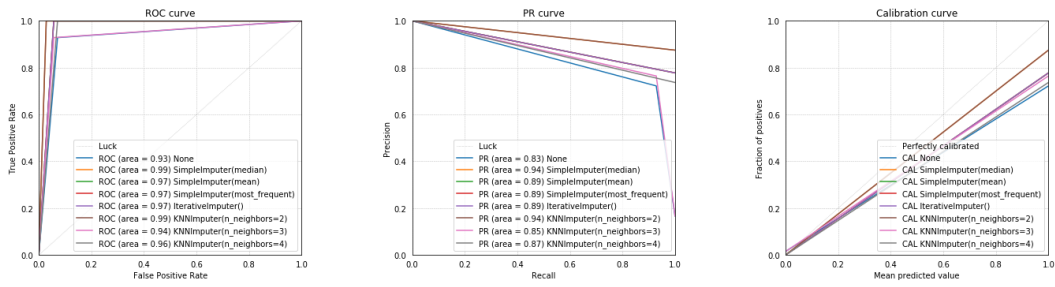
Figure 7.6: Impact of different filtering methods on the performance of the CBR module in predicting the outcome of the patient based on the pathology dataset

properties of a molecule in the patient’s blood). Consequently, the missing values can be imputed using the rest of the features, thus, a multi-variate (i.e. Iterative Imputation) approach was adopted.

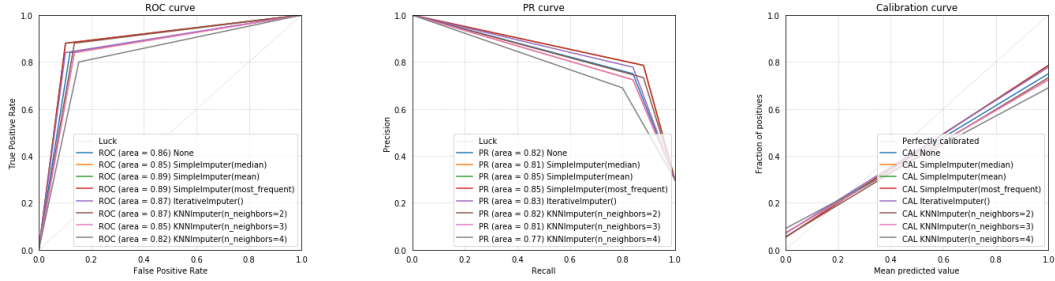
Feature Scaling

The impact of choosing different filtering discussed in Section methods 6.5 to predict the outcome of the patient (i.e. admission to the hospital and death) is presented in Figures ?? and 7.10 for both the vital signs and pathology datasets.

Having different scaling techniques didn’t affect the performance of the CBR a lot, since the filtering of the outliers out if necessary has already been done previously for both data sets. Therefore, the scaler with the best overall performance has been chosen for each data set; The MaxAbsScaler has been chose for the pathology dataset (Figure 7.10) and MinMaxScaler has been chosen for the vital signs dataset (Figure 7.9).

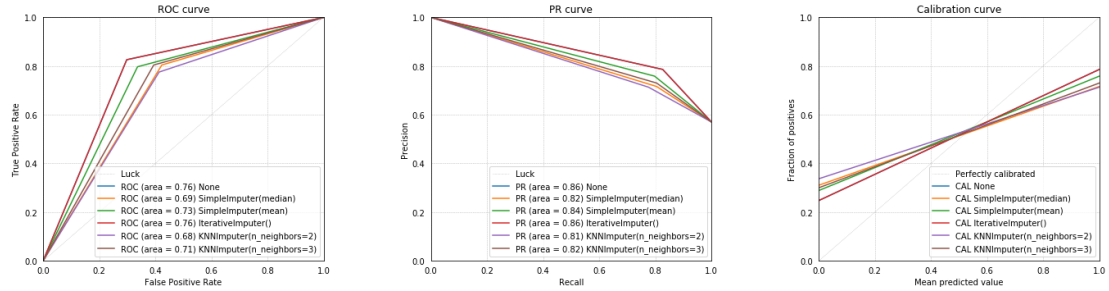


(a) ICU admission prediction

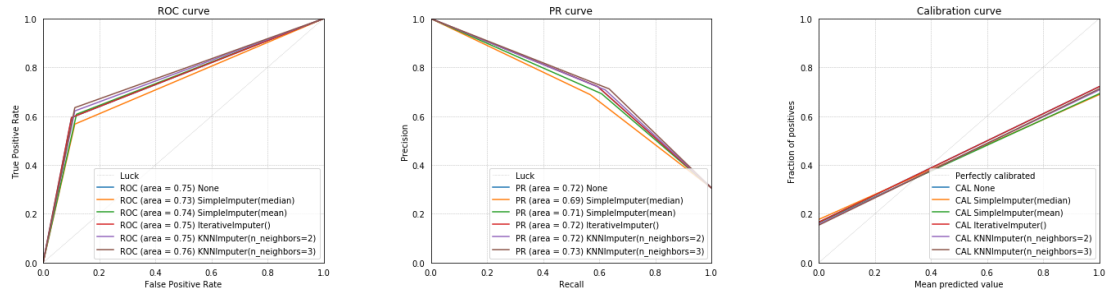


(b) Death prediction

Figure 7.7: Impact of different imputation methods on the performance of the CBR module in predicting the outcome of the patient based on the vital signs dataset



(a) ICU admission prediction



(b) Death prediction

Figure 7.8: Impact of different imputation methods on the performance of the CBR module in predicting the outcome of the patient based on the pathology dataset

7.1.5 The Implementation

Vital signs Dataset

Based on the results discussed in Section 7.1.4, the architecture in Figure 7.1.5 was chosen for the vital signs dataset, using the weights presented in Table 7.2.

The CBR model presented in Figure 7.1.5 have achieved the performance presented in Figure 7.12 and Table 7.3. The performance is promising with a specificity and sensitivity that are close to each other despite the class imbalance.

Pathology Dataset

Based on the results discussed in Section 7.1.4, the architecture in Figure 7.1.5 was chosen for the vital signs dataset, using the weights presented in Table 7.4.

The CBR model presented in Figure 7.1.5 have achieved a promising performance presented in Figure

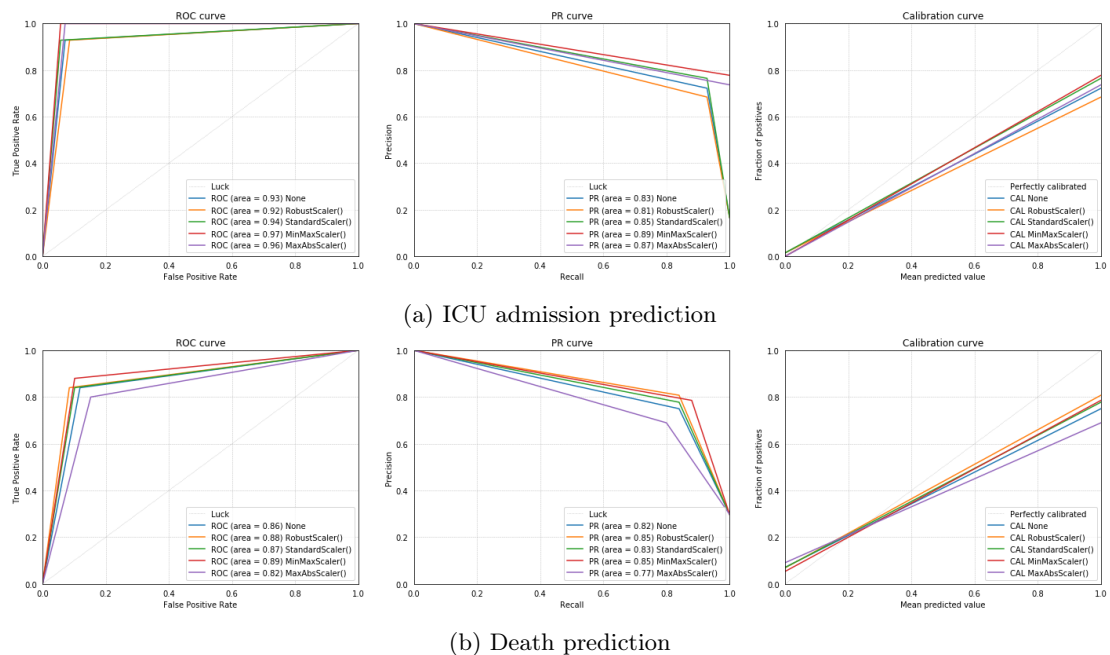


Figure 7.9: Impact of different scaling methods on the performance of the CBR module in predicting the outcome of the patient based on the keira dataset

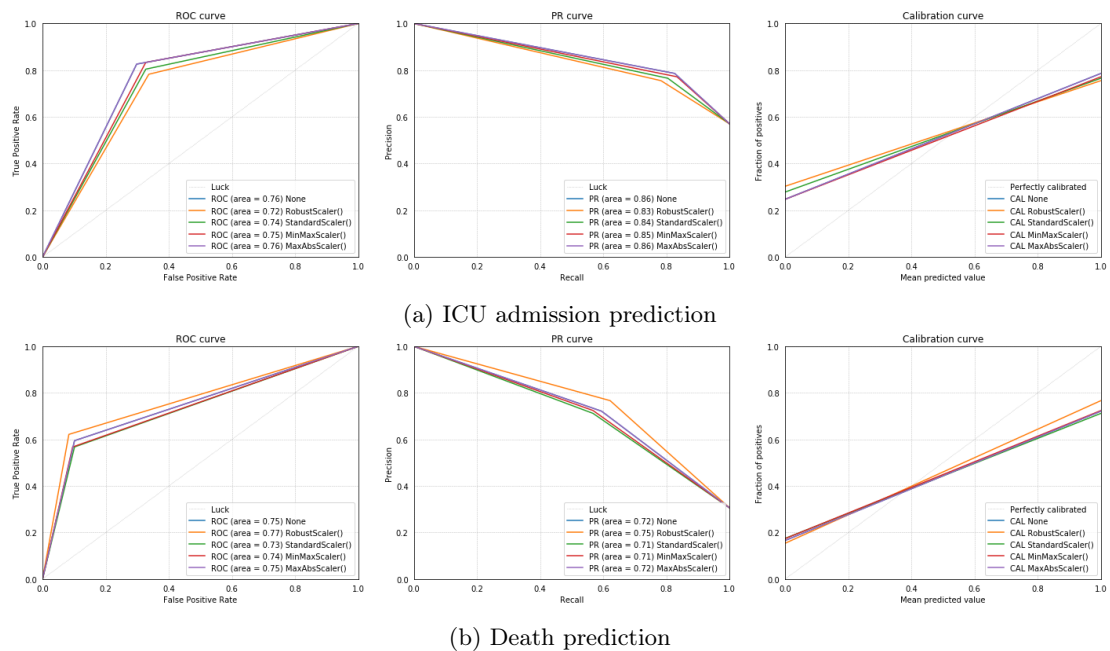


Figure 7.10: Impact of different scaling methods on the performance of the CBR module in predicting the outcome of the patient based on the pathology dataset

Name	Importance
Max resp support	0.5589
Age	0.0956
MuLBSTA	0.0646
SysBP	0.0534
Heart Rate	0.1106
Day of symptoms at presentation	0.1169

Table 7.2: Feature's Importance's used in the weighted distance computation for the vital signs dataset

7.14 and Table 7.5

7.1.6 Limitations and Future Work

The supervised model has the following limitations:

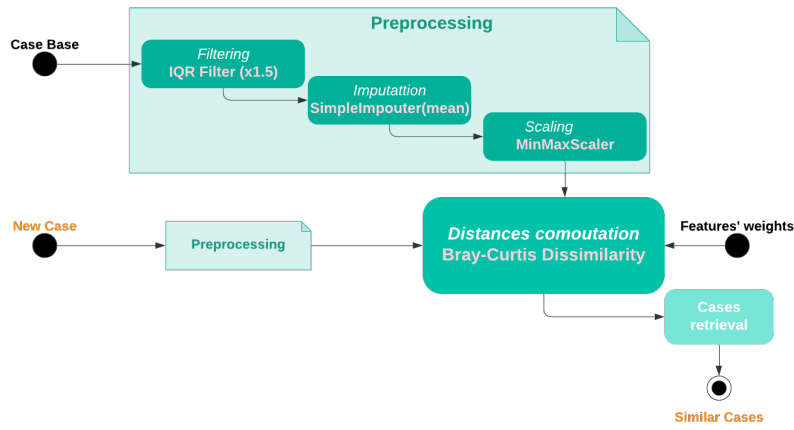
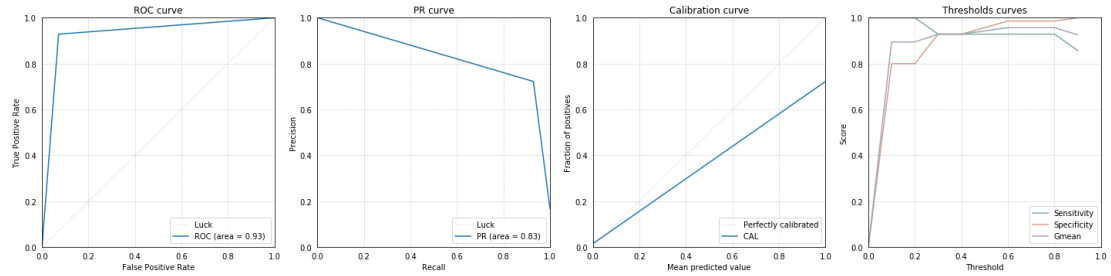
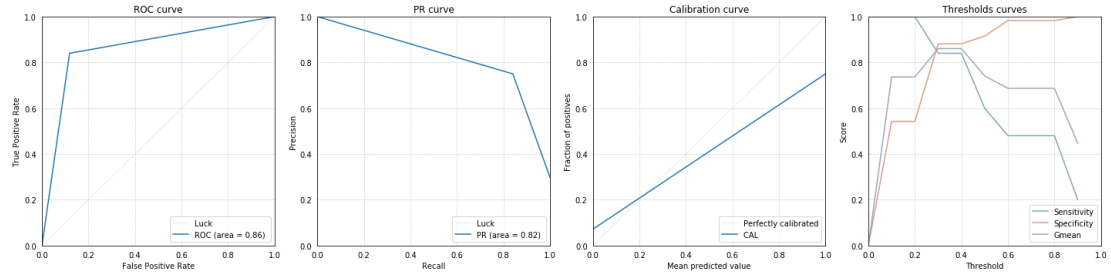


Figure 7.11: Workflow of the final supervised CBR used for the vital signs dataset



(a) ICU admission prediction



(b) Death prediction

Figure 7.12: The performance of the CBR module in predicting the outcome of the patient based on the vital signs dataset

ROC AUC	PR AUC	Speticivity	Sensitivity	Gmean	Accuracy
0.86	0.82	0.88	0.84	0.86	0.87

Table 7.3: Performance Table of the CBR using the the vital signs dataset

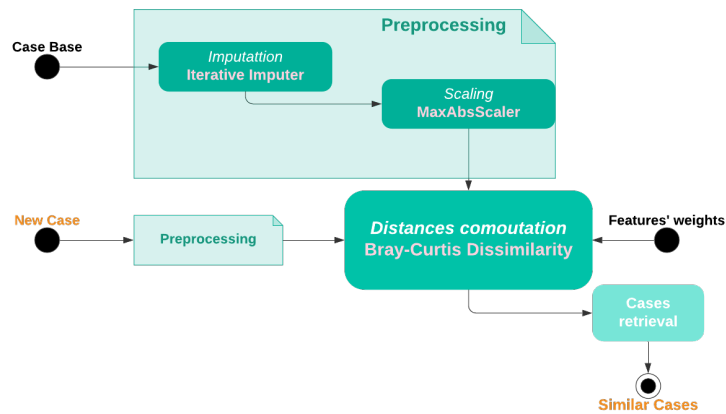


Figure 7.13: Workflow of the final supervised CBR used for the pathology dataset

Name	WFIO2	Age	UREA	CRP	PLT	CRE	HGB	RDW	MCH	WPH	NEUT	ALP	LY
Importance	0.083	0.1208	0.1056	0.1245	0.0806	0.0737	0.053	0.0627	0.0578	0.0533	0.0574	0.0675	0.0604

Table 7.4: Feature’s Importance’s used in the weighted distance computation for the pathology dataset

ROC AUC	PR AUC	Specificity	Sensitivity	Gmean	Accuracy
0.83	0.77	0.79	0.86	0.83	0.81

Table 7.5: Performance Table of the CBR using the the vital signs dataset

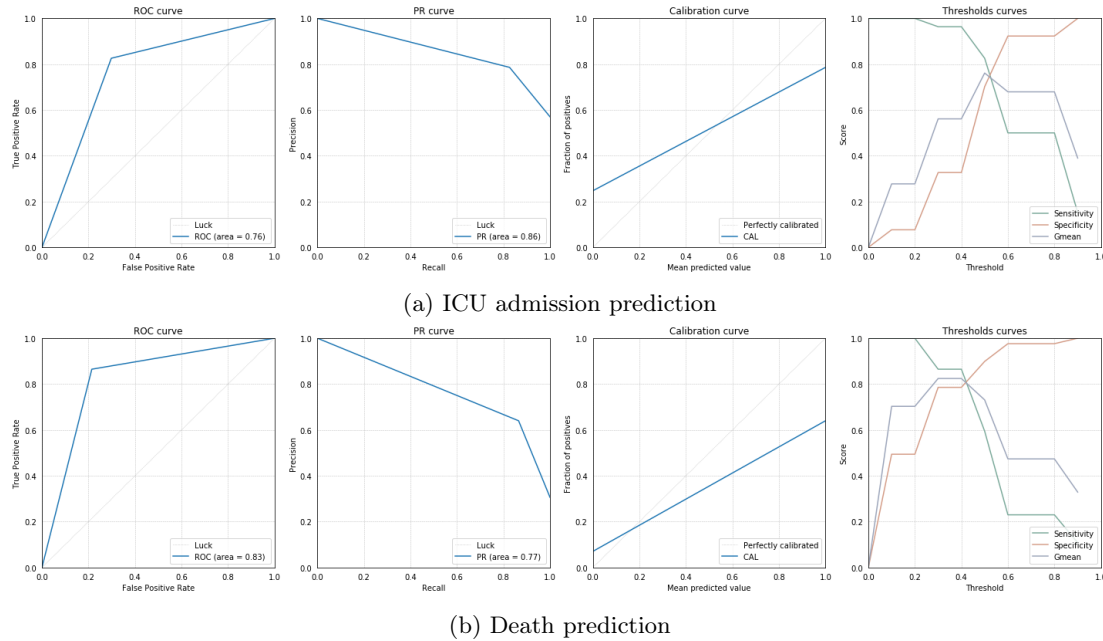


Figure 7.14: The performance of the CBR module in predicting the outcome of the patient based on the pathology dataset

- It is computationally expensive. In order to retrieve similar case to a specific patient, the distance is computed between this patient and all the examples in the case base which takes time when case base is large.
- The outcome of the patient were used as labels which might be biased. The outcome of the patient can be used to determine the severity of the patient, however, setting only admission to the ICU and death as labels may overlook important factors that determines the severity of the patients.

The supervised CBR model has shown promising performances, however, the following aspects can be investigated in order to overcome these limitations and evaluate model performances even further:

- Retrain the weights used in the distance computation and choose reselect the model based on continuous or multi-class label.
- Creating prototypes, generalised cases that represents a set of similar cases, to have a planned search retrieval and reduce the number of total distance computed. The centre of the clusters found using unsupervised learning in Section 8.
- Performing case maintenance by eliminating redundant cases and old cases.
- Evaluate the robustness of the proposed models against missing values

8 Unsupervised CBR

Unsupervised learning is a machine learning technique that aims to discover patterns without using labels in the input datasets. Cluster analysis is the most commonly used unsupervised learning technique, which is useful for grouping data points with similar properties.

For the CBR module, an unsupervised learning algorithm-based artificial intelligence function is developed aims to retrieve similar cases of the input patient by their demographic information, vital signs, and biomarkers without acquiring the outcomes of the cases in the database.

8.1 Motivations

There are two primary motivations for the unsupervised approach. Firstly, there are many patients with missing outcomes, including hospitalization, ICU admission, and death observation, in the database. For example, in the pre-processed pathology dataset, 106 out of a total of 943 patients have no information about the mortality status since they have not been released from the hospital, or the hospital did not follow up on the patient's status after they have been released.

The cases with missing labels are discarded when applying supervised learning algorithms, which leads to wasting data that may contribute valuable information. Secondly, the outcomes of the patients hardly always reflect the information of the patient. Apart from the demographic information and biomarkers, the outcomes can be frequently affected by various hospitals, clinicians, and times of admission. An improvement can be observed on all three outcomes since the first dataset. Especially the mortality rate has dropped from 85.1% in the first dataset to 29.3%, despite that there are no significant differences in the vital signs and biomarkers between the datasets. Therefore, the unsupervised CBR module facilitates the retrieval of similar patients at the point of care and provides an effective prediction of potential patient treatments and outcomes by comparing cases with both known and even unknown outcomes. The unsupervised learning approach does not require re-sampling and imputation and is capable of automatically detecting outliers. This is discussed in a later section.

8.2 Clustering Algorithms

This section aims to provide brief introductions to different categories of clustering algorithms and explanations of their pros and cons. Clustering algorithms can be categorized as follows;

- Euclidean Distance-Based Clustering
- Centroid-Based Clustering
- Model-Based Clustering
- Density-Based Clustering
- Hierarchical Clustering
- Deep Neural Networks

In reality, algorithms can be built by a combination of some of the categories. For example, KMeans is a centroid based clustering algorithm that primarily uses Euclidean distance as the metric.

Clustering algorithms are generally more challenging to evaluate and optimize due to the exemption of labels. Sometimes different metrics and evaluation methods are used for different clustering algorithms as well. This makes a grid search more difficult and sometimes unreliable. There are two primary focuses of the algorithms: KMeans clustering and HDBSCAN. Both algorithms are evaluated in the following sections.

8.2.1 Model Based Clustering

Model-based clustering is the method that optimally fits the data with some probabilistic models or geometries. The most commonly used is the Gaussian Mixture Models, namely based on the Gaussian distribution.

The dataset has a relatively large sample size, unknown and arbitrary geometry or probability density function of clusters, non-linear and skewed features, and many outliers. The combined results demonstrate that model-based clustering algorithms are not suitable for such purposes, and so other solutions should be investigated.

Advantages	Disadvantages
Good density estimation	Limited scalability as the sample size increases Difficult to optimise Limited geometry by the particular model Less effective when the dataset has a large skewness Sensitive to outliers

Table 8.1: Advantages and disadvantages of model based clustering algorithms

8.2.2 Deep Neural Network (DNN)

DNN approaches are commonly used in state-of-the-art cluster analysis. GAN/medGAN (Generative adversarial network) is the most popular model in the area of bioinformatics. A GAN consists of a generator and a discriminator. The generator tries to fool the discriminator, while the discriminator tries to catch the generator. A clustering GAN²⁷ is a method, designed explicitly for clustering in a latent space representation. It frequently outperforms other algorithms in many aspects with sufficient sample size, when properly tuned. However, the GANs are extremely difficult to optimize and consume considerable computational power. Moreover, the hidden layers are difficult to interpret, which can have ethical implications for a system deployed in a clinical setting. DNNs can also be used to impute missing data.

8.3 KMeans

KMeans is one of the most commonly known and used unsupervised learning algorithm. KMeans is a centroid based algorithm that exploits the Euclidean distances of the points to the centroid. To illustrate the metrics and evaluation method that is been used, it is good practice to understand the principles of KMeans. The algorithm can be described as follows. First, consider input feature space with some distance metric d which for most cases is the Euclidean distance, $X = \mathbb{R}^d$ and $d(x, y) = \|x - y\|^2$. Aiming to create k clusters C_1, \dots, C_k

1. Initialise cluster centres $\mu_1, \mu_1, \dots, \mu_k \in \mathbb{R}^d$ randomly
2. Repeat until convergence

- For $i = 1, \dots, n$, set

$$c_i = \operatorname{argmin}_j d(x_i, \mu_j) \quad (8.1)$$

- For $j = 1, \dots, k$, set

$$\mu_j = \frac{\sum_{i=1}^n L_2\{c_i = j\} x_i}{\sum_{i=1}^n L_2\{c_i = j\}} \quad (8.2)$$

The objective function is defined as:

$$\begin{aligned} J(C_1, C_2, \dots, C_k) &= \sum_{i=1}^k \sum_{x \in C_i} d(x, \mu_i)^2 \\ &= \sum_{i=1}^n \|x_i - \mu_{C_i}\|^2 \end{aligned} \quad (8.3)$$

The objective function is minimized at each iteration for every cluster. The objective function is also minimized for the cluster centers. Note that the objective function is monotonically decreasing but not convex. Hence the function must converge but not necessarily converges to the global minimum, which will be shown later in the silhouette plot. This depends highly on the random initialization. Therefore it is generally a good practice to run the algorithm multiple times.

Most clustering algorithms require prior specification of the number of clusters. Since there is no prior indication of the number of patient categories (clusters), a metric or visualization must be used to derive this for the clusters.

8.3.1 The Elbow Method

A simple and popular method to find the optimal number of clusters is by observing the drop in the objective function as the number of clusters increased. Figure 8.1 shows the objective function plotted (defined as Distortion) against the number of clusters. However, the Elbow Method does not always work as expected. In Figure 8.1, there is a clear joint point indicating the elbow, and the reduction of Distortion is not ideal for such a high number of clusters. Seeking a better evaluation method or even more, a suitable clustering algorithm is therefore encouraged.

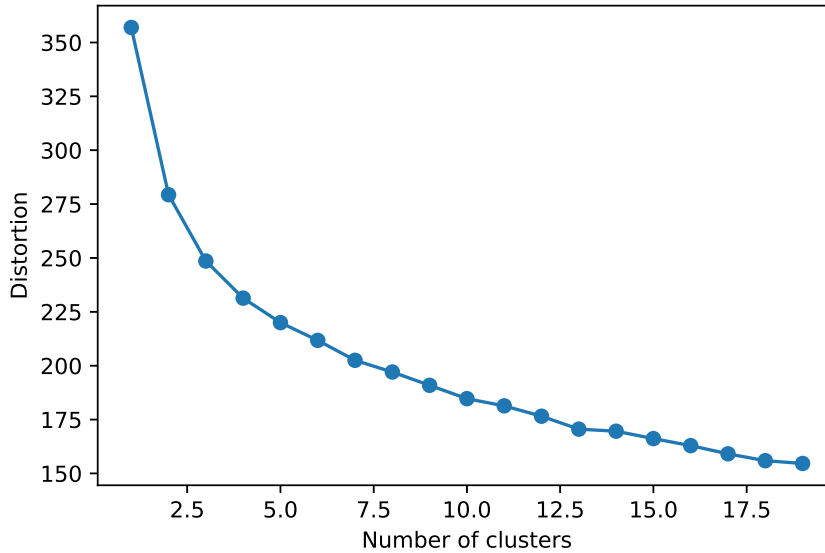


Figure 8.1: The Elbow Method of KMeans cluster

8.3.2 Silhouette

Another popular evaluation method for KMeans clustering is the so-called silhouette. A Silhouette is the measure of the separation of the clusters. The silhouette score indicates the distance metric applied (in this case, Euclidean distance) of each point to the neighboring cluster centroid. A higher silhouette score indicates a good separation of clusters, whereas a negative silhouette indicates the point was inadequately clustered. This is due to the random initialization, as mentioned previously, or suffering something called "the curse of dimensionality," which will be discussed in the later paragraphs. Too many negative values occur for the silhouette scores, and this indicates that KMeans could be an inappropriate algorithm for this task.

On the other hand, a silhouette score of above 0.5 is generally considered good performance. The silhouette is useful for selecting the optimal number of clusters for the Kmeans algorithm by observing the change in silhouette plots and scores for the various number of clusters. The silhouette is better for avoiding the danger of using too many clusters than the Elbow method.

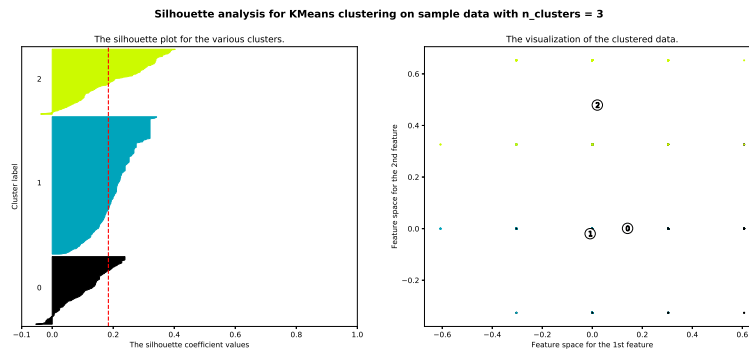


Figure 8.2: Silhouette plot for number of clusters = 3

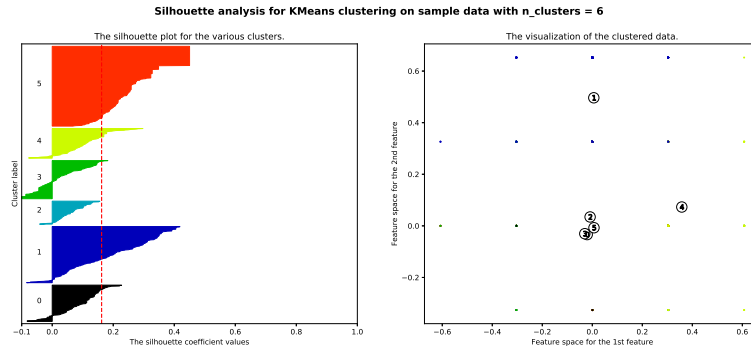


Figure 8.3: Silhouette plot for number of clusters = 6

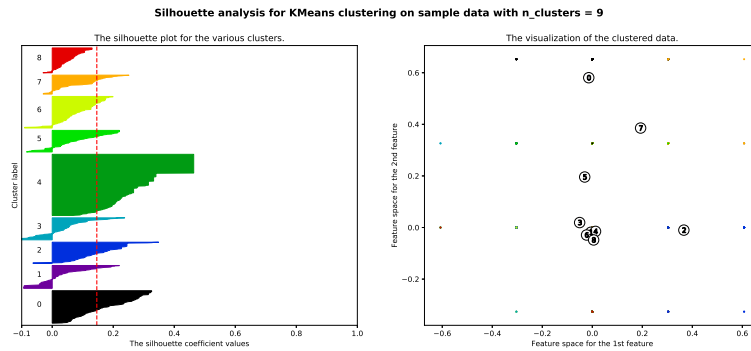


Figure 8.4: Silhouette plot for number of clusters = 9

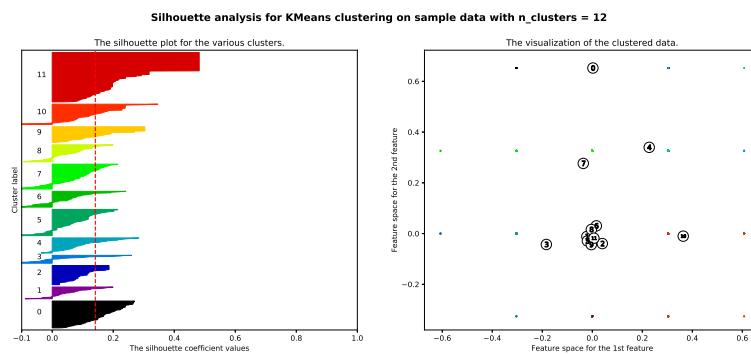


Figure 8.5: Silhouette plot for number of clusters = 12

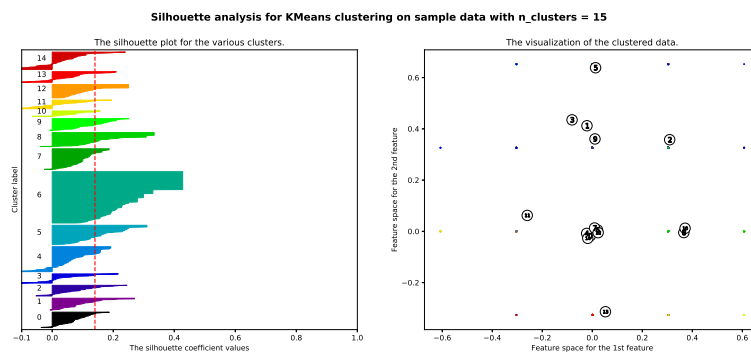


Figure 8.6: Silhouette plot for number of clusters = 15

Figure 8.2 to Figure 8.6 shows the silhouette plot of cluster number from 3 to 15. It can immediately be observed that the average silhouette score is low. The cluster center visualization shows that the clusters are largely overlapping. There are a significant amount of negative values for the silhouette score. None of these indicates KMeans algorithms are suitable for this dataset. The code was run 1000 times, and the results obtained are summarised in Table 8.2.

Number of Clusters	Average Silhouette Score
3	0.184
6	0.162
9	0.146
12	0.141
15	0.140

Table 8.2: Average number silhouette scores of 1000 trials for various number of clusters

The results are far from ideal for both "elbow" findings and silhouettes. From this, it can be concluded that the KMeans algorithm is not suitable for this task.

8.3.3 Curse of dimensionality

The dataset used has at least 11 features for each patient. This investigation concluded that these 11 features are the minimum required for any useful conclusions about the status of patients. The dataset was mapped to an 11-dimensional space, which is a relatively high dimensionality. Unfortunately, this results in the so-called "curse of dimensionality."

In short, this means that Euclidean distance-based clustering algorithms fail for high dimensional spaces. This is due to the sparseness and non-linearity of the data and means. It can be challenging to separate clusters from one another, and this may explain why the clusters clump together in the previous plots. The properties of the KMeans algorithm are summarised in Table 8.3. From this, it can be concluded that the next algorithm tested needs to use automated outlier detection to be capable of handle many clusters with diversified geometries and sizes and use a more complicated algorithm or metrics other than Euclidean distances.

Advantages	Disadvantages
Low requirement for computational power	Limited consideration apart from Euclidean distances
Relatively easy to tune	Even cluster size
Easy to use and familiar	Flat geometry
Good scalability	Only works with limited number of clusters
	Sensitive to outliers

Table 8.3: Advantages and disadvantages of KMeans clustering algorithms

8.4 HDBSCAN

Before the HDBSCAN²⁶ algorithm can be investigated, the two remaining categories of clustering algorithms - Density-based clustering and Hierarchical Clustering - have to be investigated.

8.4.1 Density based clustering

Density-based clustering algorithms consider clusters being dense regions separated by light regions. As a result, the clusters can form any arbitrary size, in contrast to the KMeans algorithm, which is likely to cluster into spheres or "high dimensional spheres." The other distinct properties of density-based clustering methods are

- Using the minimum cluster size rather than cluster number
- Deterministic instead of stochastic initialization
- Relatively robust to outliers especially local outliers since they are considered as the light regions that separate the clusters

The drawbacks of this type of algorithms are that they are relatively difficult to optimize and memory demanding. A typical example of density-based clustering is the DBSCAN²⁰ algorithm.

8.4.2 Hierarchical Clustering

This algorithm mainly focuses on the "relationship" of clusters. Agglomerative clustering is a typical example which builds up the cluster hierarchy from bottom to top. Such algorithms often require the minimum distances between clusters and, therefore, act as a reasonable control-mechanism of cluster size. Hierarchical clustering is frequently vulnerable to outliers. This will be discussed in more detail in later sections.

8.4.3 Real HDBSCAN algorithm

The HDBSCAN algorithm is a combination of the DBSCAN and agglomerative clustering. More precisely, it is a density-based clustering algorithm, built from bottom to top. This algorithm has almost made all the drawbacks of KMeans to its advantages. The cluster size, shape, and number of clusters should be flexible because most of the biomarkers and vital signs are non-linear and very distinct.

A density-based algorithm is also a promising approach because this is essentially the goal of retrieving similar patients. The tuning and optimization of the HDBSCAN algorithm are not trivial. Table 8.4 shows the effects of the different parameters of the HDBSCAN algorithm.

Parameter Name	Purpose	Value of Choice
min_cluster_size	smallest cluster size	5
min_samples	aggressiveness of clustering	2
cluster_selection_epsilon	threshold distance for further clustering	0.21
cluster_selection_method	how flat clusters are selected from the cluster tree hierarchy	'eom'

Table 8.4: Major hyper parameter tuning of HDBSCAN

Some of these parameters are tuned by comparing the visualization and metrics. Others are strategic decisions made to satisfy the design criteria. It was decided that relatively conservative approaches should be used to cluster the patients, which means that the algorithm clusters many relatively small clusters and leave a significant amount of patients as outliers. This allows clinicians to have more flexibility to look into more ambiguous cases. The built-in outlier detection and the robust nature of density-based algorithms, results in effective local outlier detection.

8.4.4 Visualisation and Evaluation

Tuning this model to satisfy the design criteria proved challenging, especially for complicated unsupervised learning algorithms. The evaluation process involved the following stages:

1. Visualisation with Principal component analysis (PCA) to briefly check the clustering performance and outlier detection
2. Cross validate with the outcome of the patient (if applicable) in the database
3. Consulted and monitored by medical professionals

Figure 8.7 and 8.8 show the clusters in the representation space with Principal Component Analysis (PCA). Despite only using 2 or 3 dimensions and having a huge number of clusters, some identifiable clusters can be observed by grouping their densities.

Figure 8.9 shows the outliers detected by the automated outlier detection algorithm. Since the main focus was on the local instead of global outliers, it is difficult to tell if the outlier detection has made a significant contribution. All results were cross-validated with the outcomes from the database. Despite the issues identified at the start of this chapter, a confidence level of more than 77% was achieved in predicting all three outcomes consistently. This is a very satisfying result considering the bias of the data and the fact that only an unsupervised learning approach was used. Finally, despite the good performance level achieved, all these results need to be verified and monitored by experienced medical professionals.

8.5 Implementation of the Final Product

The HDBSCAN algorithm was chosen for the unsupervised similar patient retrieval subsystem. This was then implemented in the pipeline, as shown in Figure 8.10.

The pipeline essentially does:

1. Retrieve patients from the NHS database

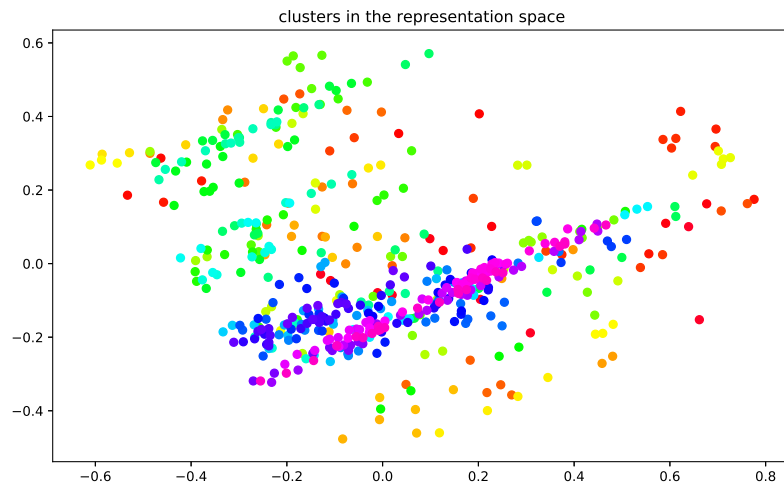


Figure 8.7: 2D cluster visualisation of the clusters

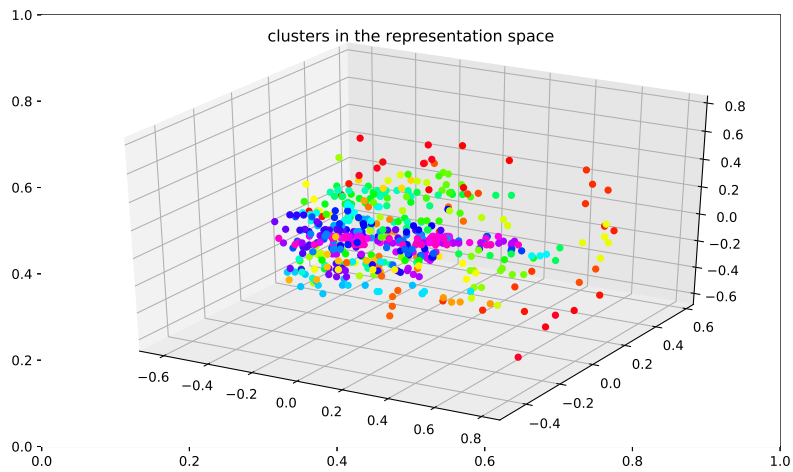


Figure 8.8: 3D cluster visualisation of the clusters

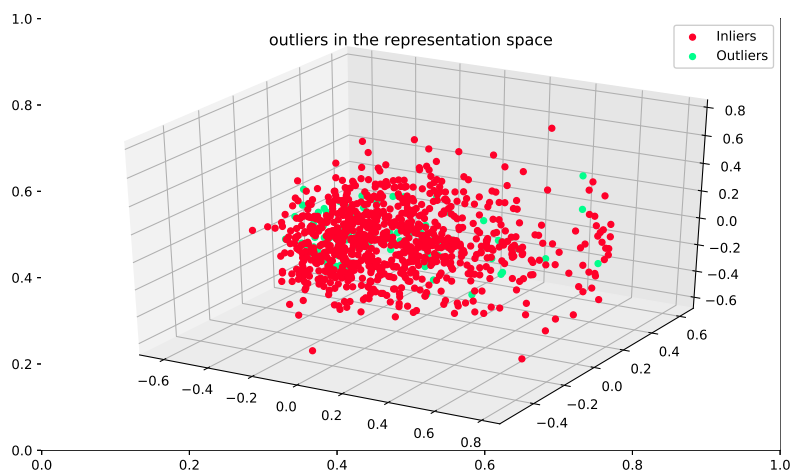


Figure 8.9: 3D outlier visualisation



Figure 8.10: Block diagram for the unsupervised similar patient retrieval pipeline

2. Pre-process the data to the desired form explained in the pre-processing section
3. Initialise all features by scaling with the square root of the weights pre-computed by random forest impurities
4. Cluster by HDBSCAN algorithm
5. Key in the new patient's detail
6. Predict on the new patient
7. Proceed either one of the two
 - Consider as an outlier and proceed to a case by case study by the clinicians
 - Receive a cluster label and return all other patients within that cluster.

8.6 Future Work

This pipeline has some promising performance but still far from perfection. As mentioned earlier, more complicated models could be experimented with, such as neural networks. Furthermore, an online learning feature could be implemented to comprehend the ever-expanding case base. If this were done, the pipeline would look as follows:

1. Retrieve patients from the NHS database
2. Pre-process the data to the desired form explained in the pre-processing section
3. Initialise all features by scaling with the square root of the weights pre-computed by random forest impurities
4. Cluster by HDBSCAN algorithm
5. Key in the new patient's detail
6. Predict on the new patient
7. Proceed either one of the two
 - Consider as an outlier and proceed to a case by case study by the clinicians
 - Receive a cluster label and return all other patients within that cluster.
8. Send the patient information together with the label or outlier detection outcome back to the database

The clustering is necessary every time a new patient is retrieved. This online learning pipeline requires constant monitoring of medical professionals as well as maintenance.

9 Django Application

9.1 What is Django?

Django is a Python framework that facilitates web-app development. Since EPIC IMPOC was built upon Django, it was used for the module for the sake of consistency and easy integration. It focuses on fast-paced development, security and scalability; all desirable traits for this project. Table 9.1 summarises some commonly used terms in Django, and explains their meaning in the context of this project. A more in-depth introduction to Django can be found in the official documentation.³

Term	Description
Apps	A Django app is an application that can perform a certain set of functions. A project can consist of several apps. In this case, the project is EPIC IMPOC, whereas the new apps are COVID_CBR and COVID_INFERENCE.
Databases	A database is a collection of data that allows easy access and editing e.g. a database of patient cases.
Models	A Django model is a python class that represents how data is stored in a database. Following on from the above example, a patient case model would have parameters such as: Gender, Ethnicity, Ventilation Requirements, Outcome, Date.
Views	A Django view is a function that takes a web request and returns a web response. In the case of the COVID apps, the views take in web requests and input data (e.g. from a database or CSV) and returns a web response in the form of output data and an HTML template.
Forms	A Django class that describes a form, how it works and what it looks like. Simplifies data management.
Serializers	Part of the Django REST Framework, a toolkit for building APIs. Allows models and querysets to be translated into other content types. Other developers can then easily implement the functions into other apps.
Templates	Templates are written in a quasi-HTML-Python language that facilitates human-readable display of any output data e.g. a list of patients. They also provide functionality to allow the end-user to interact with the apps e.g. via a form.
Admin Interface	The Django admin interface facilitates activities that require high-level permissions such as reading or editing databases. Provided the end-user had the correct username and password, they could easily read and edit patient details in a patient database.

Table 9.1: Summary of core components of the Django Application

9.2 Design of User Interface



Figure 9.1: Black arrows represent URL paths while purple arrows show user journey.

9.3 The App's views

9.3.1 CBR-Vital signs

Vital signs home view

Displays a form (see figure 9.2) that allows doctors to search for a patient in the app's CBR database. The user must fill at least one field of the form before it can be submitted.

Patient search works as follows: The form's fields are passed to the Django view, which filters patients that contain the input fields as a sub-string of their own fields. For example, if the input arguments are a single "H" in the "name" field, this function will return any patient whose name contains the letters "h" or "H". If the input arguments are "H" in the "name" field, and "Pa" in the surname field, it will return any patient whose name contains the letters "h" or "H", and whose surname contains "pa", "PA", "Pa", or "pA".

Patient search results view

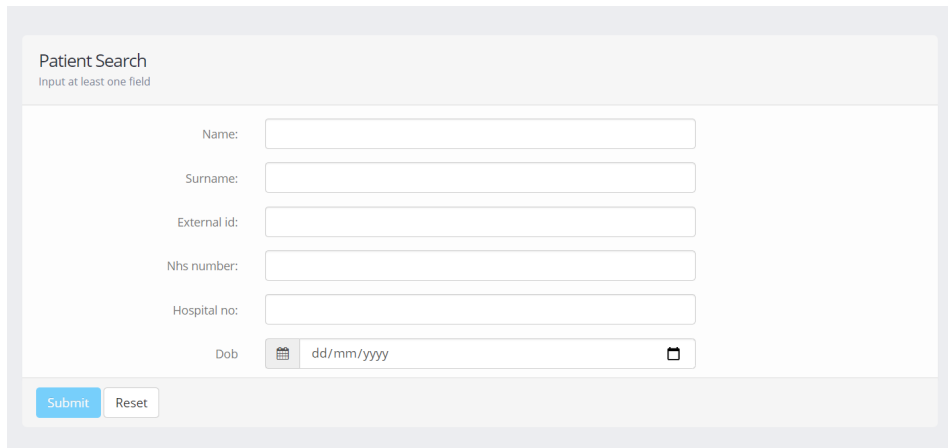
Displays a table (see figure 9.3) listing all patients that match the search as described above. A link is included for each patient that allows access to a following screen to view the patient's cases.

Patient cases view

Recorded cases for a given patient are displayed (see figure 9.4). The patient's primary key is given as part of the URL (see urls.py file). A link is included for each case that allows access to the following screen to view its most similar cases.

Query similar cases view

A case is given to this view via the URL (see file urls.py). The Django view then calls the models which give back the most similar cases (see figure 9.5) from the case base. The case bases used for both CBR-Pathology and CBR-Vital Signs can be found in the form of CSV files in the "data_pathology", "data_unsupervised",



Patient Search
Input at least one field

Name:

Surname:

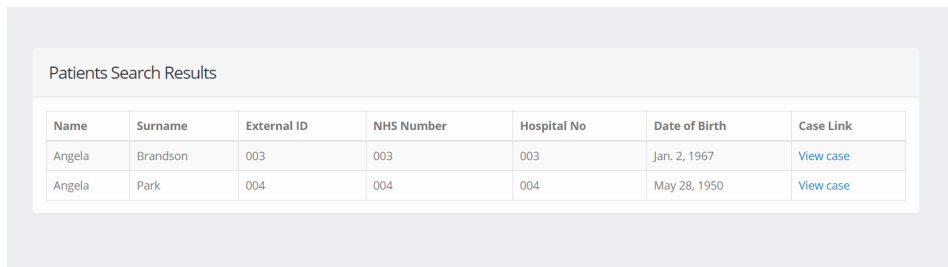
External id:

Nhs number:

Hospital no:

Dob:

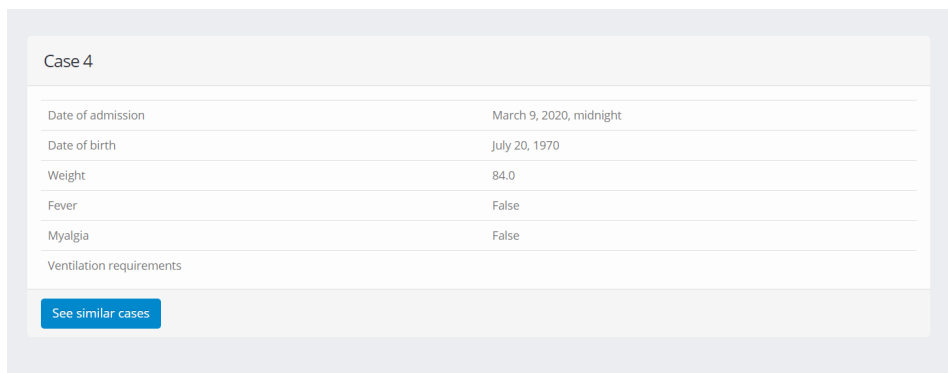
Figure 9.2: Patient Search Form



Patients Search Results

Name	Surname	External ID	NHS Number	Hospital No	Date of Birth	Case Link
Angela	Brandson	003	003	003	Jan. 2, 1967	View case
Angela	Park	004	004	004	May 28, 1950	View case

Figure 9.3: Patient Search Results

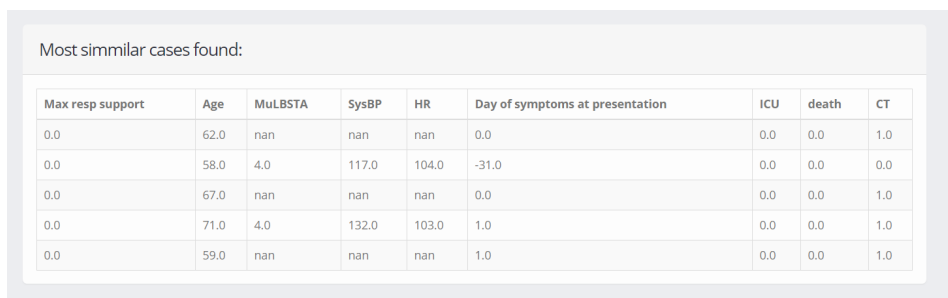


Case 4

Date of admission	March 9, 2020, midnight
Date of birth	July 20, 1970
Weight	84.0
Fever	False
Myalgia	False
Ventilation requirements	

Figure 9.4: View showing a patient's cases

and "data_vital" folders within the app. A task for the future would consist in using the app's database as case-base. This introduces new challenges, such as how to re-train the models when the database is updated, or how to use techniques such as indexing to deal with a much larger case base than the one currently being used.



Most similar cases found:

Max resp support	Age	MuLBSTA	SysBP	HR	Day of symptoms at presentation	ICU	death	CT
0.0	62.0	nan	nan	nan	0.0	0.0	0.0	1.0
0.0	58.0	4.0	117.0	104.0	-31.0	0.0	0.0	0.0
0.0	67.0	nan	nan	nan	0.0	0.0	0.0	1.0
0.0	71.0	4.0	132.0	103.0	1.0	0.0	0.0	1.0
0.0	59.0	nan	nan	nan	1.0	0.0	0.0	1.0

Figure 9.5: View showing a list of the most similar cases

9.3.2 CBR-Pathology

Manual input form view

A simple form view that allows users to manually input a pathology case.

Query similar cases view

The case provided in the manual input form view is given to the ML models. A list with the most similar cases are returned. For this section of the app, two models were produced (a supervised learning approach and an unsupervised learning approach). The result from each algorithm is displayed in the same page, thus allowing for easy comparison between the two models when testing.

9.3.3 Inference

As requested by the client, three different methods of inputting cases to the inference models were produced. Three different ML models are used to make predictions respectively. This allows for easy comparison by clinicians on the models' performance.

Manual input form

A simple form view (see figure 9.6) allows users to manually input a pathology case. After submitting, the predictions will appear below the form.

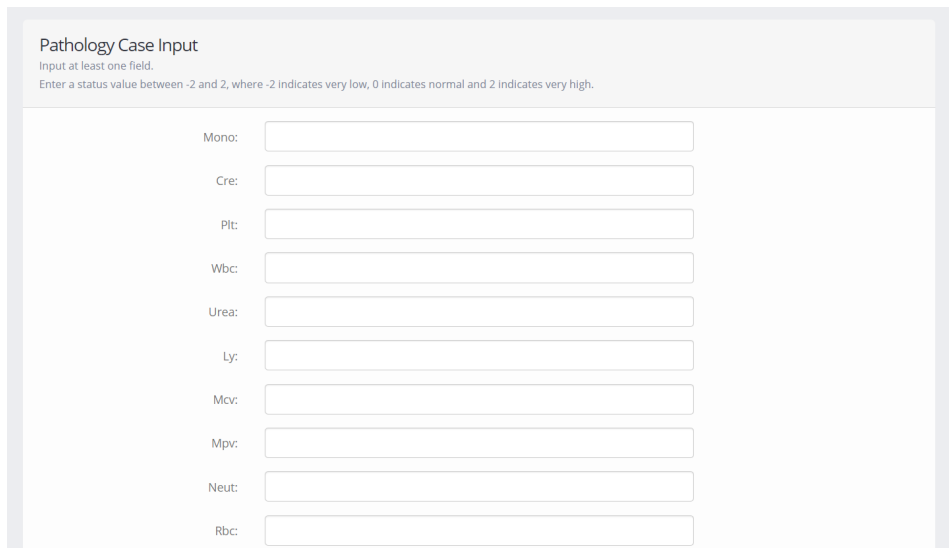


Figure 9.6: Form to manually input pathology cases.

CSV file upload

This view processes the inputted CSV file (see figure 9.7), by creating a single column for each patient, and adding predictions to each column. The resulting CSV file is then made available to download from the browser. There is also an extra ListView, "FileListView", that can be used to retrieve a list of all the CSV files uploaded in the past with their predictions.

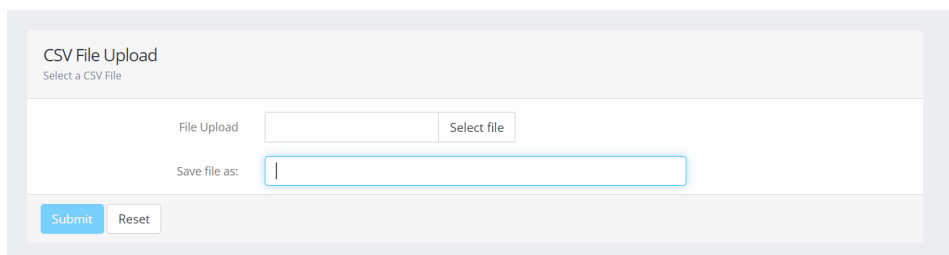


Figure 9.7: View that allows the user to upload CSV files with pathology data.

Database retrieval

A very similar user journey to that described in section 9.1.1 has been implemented: There are views to look for patients (this time using patients from the pathology database instead of the CBR database), to retrieve the newest records for each marker for a given patient, and to make predictions on a given case.

9.3.4 File Processing and Predictions

Instead of processing files and making predictions directly in views.py, some "predictors" modules have been created within both the Inference and the CBR apps. This modules contains classes to pre-process CSV files

with pathology records, as explained in previous sections of this report, and to standardise the data coming from the app's forms and databases in such way that it can be given to the models.

9.4 The API

As requested by the client, most views in the application are accompanied with an API "equivalent". This API has been built using another python library known as the Django REST Framework.² The idea of an API is to give the client the option of calling the algorithms from other applications, without the need to use the front-end. Some examples of HTML requests for each of these API Views can be found in the following link: <https://www.getpostman.com/collections/26300cd4c4d978dd4a39>

10 Project Management

This project was managed through a series of methods, tools and techniques from start to finish. The primary method of communication used was regular meetings, which took place at least twice a week, but usually much more frequently. The unique remote working situation under which this project took place resulted in some challenges such as team members on different continents and frequent internet connectivity issues. Our unique project management system overcame these issues by leveraging technological tools and popular project management techniques such as Kanban boards and an Agile workflow.



Figure 10.1: Example Scrum Diagram taken from HBO's hit show Silicon Valley

Given that meetings in person were not possible, a Facebook messenger group chat was used as the primary means of communication. This was effective not only because of quick responses and a record of communication, but also because it created transparency in the discussion of ideas. It should be mentioned that Microsoft Teams was used as well by mandate of the department but it often presented more technical difficulties than solutions.

The tasks that made up the project were allocated to different members of the group taking into account both personal preferences and previous experiences. The tasks were varied between administrative roles, technical work and research on each of the subsystems of the project, however, although individuals focused on specific subsystems, the team collaborated effectively and throughout the project.

The group was then split into two teams to further develop the different subsystems and parts of the project. The following list describes the actual split of the work in this project.

- Yannis, Selena and Yichen worked on researching machine learning methods for the project
- Yannis and Selena worked on the pre-processing steps of the project
- Yannis and Selena worked on the research relating to the Probabilistic Inference Module
- Yannis and Selena worked on developing the Probabilistic Inference module
- Yannis worked on the hyper-parameter optimisation for the Probabilistic Inference module
- Selena and Yichen worked on the research of different clustering methods for Case-based Reasoning
- Selena worked on developing the supervised Case-based Reasoning (retrieval only) system
- Yichen and Selena worked on developing the unsupervised Case-based Reasoning (retrieval only) system
- Daniel and Ghaj worked on developing the Django-based app and API for the COVID module
- Yannis kept the meeting minutes for all the meetings
- Selena, Daniel and Yannis worked on subsystem integration
- Yannis organised and produced the video to accompany the presentation
- Yannis and Selena formatted the Documentation

- Yannis set up, proofread and edited all content in the report and wrote the Literature Review, Probabilistic Inference, Project management, Introduction, Meeting Minutes, and Conclusion
- Yichen worked on the Unsupervised Learning and Design Criteria sections of the report
- Selena worked on the Supervised Case Based Reasoning and pre-processing sections of the report
- Ghaj and Daniel worked on the Django section of the report
- Yannis, Selena, Daniel and Ghaj worked on the System architecture of the report
- Everyone developed the slides together to accompany the presentation

In order to share documents and files in an effective and organised way, Google Drive was used to store all text based files while GitHub was for management of the code and technical work for the project.

A task management platform called Trello was further used for managing the tasks in the project according to a popular management scheme known as a Scrum Diagram (Fig. 10.1). Tasks are organised by subsystem. Once a task for the project was identified, it was added to the 'Ice Box', and once a member of the team began working on the task, it proceeded to the 'In progress' column until it was completed. Once any task was completed it underwent a group review where every group member's work was peer assessed and was placed in 'Testing' until the task was finally classified as completed.

For formal report submissions and formatting, the online LaTeX editor Overleaf was used for its excellent collaboration functionality. This allowed all progress on reports to be visible at all times to members of the group and made collaborating on report writing quick and effective.

11 Conclusion and Evaluation

Overall, this project has met the requirements set out by the initial brief from the client and supervisor. The following section provides a brief evaluation and conclusion to the last two months work.

11.1 Report of Ethical Consequences

In a project of this scope, ethical considerations are paramount. With several users potentially having access to live databases filled with sensitive data, privacy and confidentiality concerns should be raised. Furthermore, the ethical consequences of such a project must also be considered. Should the module have capacity to make other complex decisions? Would all patients be treated fairly?

11.1.1 Ethical Decisions

The main goal of the module is to aid clinicians in choosing the most appropriate treatment for a particular patient to better reallocate hospital resources. However, this becomes difficult in periods where needs for forms of medical treatment exceeds available resources; the times at which the app should be most useful. Any capacity the module could have to make complex ethical decisions was removed. This was done by disregarding factors such as hospital location, resources and percentage of max capacity. Instead, the module only assesses severity of illness, relevant comorbidities or other relevant factors (e.g. age). By providing only a risk score at the end, it remains the clinicians duty to consider other data and circumstances in making the final decision. This falls inline with the British Medical Association's ethical principles¹ which advises doctors prioritise treatment among patients when needs outweigh resources. The focus of the module remains not to make complex ethical decision, but to support those who are trained to do so.

Other measures were taken to ensure the module would not make complex decisions. Obvious health and safety concerns would arise if the app were to give a binary classification as to whether a patient should be put on mechanical ventilation or sent home. Instead, a variable risk score is provided, giving an indication to the clinician of the patient's medical requirements.

11.1.2 Patient Privacy and Confidentiality

Since the app has access to sensitive patient data, measures must be taken to ensure patient privacy and confidentiality. Admin privileges (username and password) are required to access patient and case databases. On top of this, Django has a user authentication system, meaning different users can be given different levels of permissions i.e. a clinician's level of access can be restricted to certain databases and functions.

The module accesses live NHS databases, thus it was ensured that no patient data would be locally cached on the front end. Furthermore, Django has SSL/HTTPS functionality, so authentication credentials and client-server data transfers can be hidden from other network users. Finally, throughout development, Machine Learning models were trained on anonymised patient data, with names and IDs having been removed. Developers had no access to patients' personal information.

11.2 Model Evaluation

The EPIC IMPOC COVID module is divided into three subsystems; a Probabilistic Inference system, a Case bBased Reasoning Retrieval system and a Django application.

The first subsystem is a Probabilistic Inference module that predicts whether or not a patient has COVID-19 based on their pathology biomarkers. Three custom-built pipelines with optimised hyperparameters and different estimators, including a Light Gradient Boosting Machine, a Random Forest Classifier, and a Decision Tree Classifiers, were integrated in the Django application to allow clinicians a choice between which algorithm to use as they each have their own respective advantages and disadvantages. The performance metrics used to evaluate these models showed that they all had balanced accuracies of approximately 80% and that they did not suffer from severe overfitting or the class imbalance in the initial dataset.

The second subsystem is the Case Based Reasoning system which retrieves similar cases to the patient based on their vital signs, symptoms, blood tests and medical history, to help the clinicians with the treatments prescription. The Case Based Reasoning System uses two separate approaches; a supervised and an unsupervised one. In the supervised approach, the weighted distance between a given patient and the remainder of cases is computed and the most similar cases are selected. These weights are pre-computed using each feature's importance from the Random Forest. The system's performance was evaluated using the patient's outcome to verify and investigate patient similarities. Different distance metrics, filters, imputers and scalers were tested and selected to maximise the performance. On another hand, the unsupervised approach

for CBR, uses an HDBSCAN algorithm initialised with the same pre-computed weights to cluster similar patients by their biomarkers and symptoms. The primary advantage of using a clustering algorithm is that it does not require re-sampling, imputation, and most importantly, labels.

Both of the previous two subsystems are packaged in a Django application that interacts directly with the NHS databases, clinicians and other end-users.

11.3 Future Work

The EPIC IMPOC COVID module has shown some promising results. However, some new aspects can be investigated. For example, for the supervised model, decreasing computational cost of the cases retrieval should be inspected. This can be done by performing the maintenance of the case base by eliminating redundant cases, and by creating generalised cases that represents a set of similar cases, to have plan and reduce the number of total distance computed. Furthermore an online learning feature can be implemented to adapt to database's expansion. On another hand, for the probabilistic inference module, the imputation of the missing values can be improved by using Generative Adversarial Networks since they can learn the hidden data distribution well. Additionally, Dropouts meet Multiple Additive Regression Trees (DART) can be used to improve the performance of the boosting algorithm that were tested (e.g. XGBoost) in order to avoid overfitting.²⁴

12 Meeting Minute Summaries

This section contains the meeting minutes taken over the course of the project. Each formal meeting is dated, fully documented and summarised over the coming pages.

12.1 Meetings with Client & Supervisor

12.1.1 Friday, 1st May 2020

Initial Supervisor and Client Meeting Summary

1. Introduction to Supervisor

- Bernard - postdoc - machine learning, ai, making clinical decisions with those
- Quick intro from the team to the supervisors Centre for BiT
- 4-5 yrs machine learning, control engineering to biomedical problems.
- Both supervisor and client and from department
- Transferring technology to real world
- Less clear who is industrial partner and who is supervisor because project will be very collaborative and involved

2. Project Discussion

- Explaining a bit of the system: clinical decision support system running in NHS network - demographics, patient data
- Finding patients with similar conditions using similarity measures in the past
- This project focuses on covid and specifically differentiating which patients should be admitted to hospital, sent home or directly to ICU
- System works in django - so easy to plug system into EPIC IMPOC
- Frontend html css javascript
- Shared this morning a small set of cases - start playing with them and identifying which parameters are important
- Try to either cluster them or predict for a new patient what should be done
- Predict relationships between parameters
- The idea is to use the data from a variety of sources:
- We have a small sample of 150 cases that we can use to start at this point to create models
- Models will not be very robust because we do not have a lot of data
- GDPR has been 'more relaxed' because of COVID's urgency
- Set up git repository for project management
- Assign people responsible for each part

3. Addressed Questions from Last Meeting:

- (a) Ask if he can help us draft a plan - we are not sure what time scale is realistic to set intermediate goals to make the project more manageable
- (b) How is the output going to be used? How interpretable the model should be?
 - Depends what you focus on - clustering you can see if patients are similar
 - If we have the treatments we can compare the treatments as well
 - More complex algorithms - usually have to send prospective results and send scores to doctors to react and judge the results
 - How to make sure NN is providing safe solution - Pau will send a paper on glucose predictions to make sure solution you are getting is safe predictions

- We are making a decision SUPPORT system - the final decision is still up to the doctors - you are just helping an expert make decisions and there is a safety layer of a human
 - Consideration of false positive vs false negative - depends on the data - try it out with
 - Data augmentation in biology becoming more popular - creating synthetic data
 - Adversarial networks would be very promising - could be a novel approach
- (c) How do we justify the ethical implications of the results of our research?
- (d) Missing data - should we use imputation to evaluate missing data? What imputation methods would he advise us to use? Implications of this in biological context / data augmentation

4. Other Project Related Items:

- Supervisors are continuously working with clinicians so we have access to experts from the clinical side and we can pass this on to supervisors to discuss with clinicians
- There is already a list in parameters in teams that clinicians were considering as important
- At the beginning don't worry about making a nice structure - just playing around, then start trying to create modules for python - set up the license, etc after the first month
- Setting minimum requirements is important: this module that predicts if a patient should need icu intervention, could add more beyond this - it also depends largely on the data available
- First step is the data, and cleaning the data
- Main specification is to have a system working and connected to EPiC IMPOC → System that connects server and client
- Data augmentation would be a very cool addition because it could also be used for other biological projects as well in the future
- Start with simple algorithms and increase complexity; Start with a linear classifier
- Pandas and numpy will be useful tools and scikit is easy for clustering
- **Logistics:**
 - Once a week catch-ups with supervisors - Friday at 10
 - Try to comment the code a lot - supervisors prefer to read code comments than documentations
 - Sphinx will create documentation for you based on specific format of method comment/descriptions
 - Continuous integration will become more important later on
- **Pathology & Data Information:**
 - Should be able to get pathology data for many more patients
 - Biomarkers - blood test - there are always 10 - when they do a blood test they give blood test counts and there is a panel – some of the biomarkers are related with age and gender - where the normal ranges should be for the biomarkers
 - Doctors otherwise only record what is relevant - they ignore a bunch of parameters which is why you have missing data sometimes
 - Demographic data such as age and gender are very important, ethnic and economic background can make a huge difference
 - Diabetes, heart disease, there is extreme correlation with disease progression
 - Dealing with data there is always an issue with privacy so we have to be careful - current dataset does not have an issue but be careful about storing on imperial servers
 - Recommended to do a literature search for people doing similar things to try to see what other people are already doing → this could be a good first step as well
 - Data we have is at admission
 - Might be able to get pathology data with time series but is likely to be not very long because they do not do the tests for very long

- Which are the most important parameters - minimum set of parameters needed - maybe a subset of them is enough
 - Hospital, survived means discharged after stay
 - If we know the discharge day we could also try to predict the number of days of stay in the hospital - for bed availability
 - Doctors in the ICU are sometimes very junior
5. After the meeting with Bernard and Pau, the team got together to decide how to split the work during the following week.
- Yichen and Daniel will start trying to implement simple models into the small dataset provided to us.
 - Selena and Yannis will work on finding research on this field through a peer literature review to learn about what other people are doing in similar projects.
 - Ghaj will do a literature review on the biological context of the project, and will work through a tutorial to familiarize with Django.
 - Yannis set up a Trello board was set up that will help us to manage our project. Different project management techniques such as “agile” could be used later on in the project.
 - Initial teamwork split - Yichen and Daniel playing with dataset, Ghaj working on biological background reading and literature review and working on a django tutorial, Selena and Yannis working on peer review of research
 - Trello + scrum + kanban board for project management + 2 week sprints

12.1.2 Friday, 8th May 2020

- Access to pathology, microbiology, patient databases
- 1000 patient database - data is clean because just patients
- Biomarker data next week
- First thing we should do is try to find out which markers are more common
- Used SMOTE to balance the data
- ICU decision - promising results without hyperparameter tweaking
- PCA - can visualise where synthetic points are generated
- Should do 10-fold validation for small dataset - at least 5 fold
- For covid patients increase in crp and neutrophil
- Chinese research paper - use ratio of neutrophil to lymphocytes
- Similarity patients bias the model - just use the median
- Other option is to use a regression to impute the model
- Could create a model to estimate the missing values based on the one we have
- Bernard to ask if there is a correlation between neutrophil, procalcitonin crp
- Could impute one biomarker value based on other biomarker values of same patient
- What metrics can we use to know that we are performing well? - 80% accuracy would be considered a success
- Radiology: from now on they will label it in different levels of risk; Currently record very complex radiology data but will be: Low medium and high

- Will get access to repository to work on a branch this weekend
- Xgboost - random forest and seems to be quite good but superior to random forest
- If output is very clear - plot probabilities of having false-positive false negative etc
- Evaluate and verify biomarker reference ranges for patients and use normal reference range to label patient severity
- Ward name sometimes has ICU so we can use that to tell if someone went to an ICU

12.1.3 Friday, 15th May 2020

- Questions regarding django repository
 - A lot of empty files
 - Repository is a template
 - Will need to pull data from different tables and then put it in the algorithm
 - Wednesday next week can cover django repository: 10AM; Bernard will send invite later and we will have to integrate algorithm with existing databases and cannot expect all data to be in the same format or in the same place
 - Bernard to look into the NHS numbers because they do not match perfectly - will check on Monday
 - Not necessarily assume that the same number of patients in the pathology database and the other dataset because some might not have had pathology tests
 - If there is no date of death then we can assume the patient did not die?
 - In the dataset, they record an episode, they don't have a date of end and the ending is in the next entry
 - Database has wardID and current wardID
 - * It might be the missing values that resulted in the discrepancy between the patient numbers
 - * Can use the wardID to identify which patients were in the ICU
 - We were thinking that in ICUs, doctors/nurses record vital signs in the ICU and secondary ward - we don't have access to that data yet; varies a lot based on biomarker - take the first / first two and average
 - Try to see which ones are more frequent - some of them are tested in a panel - assume that a few that are always present / checked together; Crp, pct, ddimer, ferritin
 - Look at feature importance - see if the performance of the classifier gets worse
 - Good to have a feel for how challenging the problem is / how difficult it is to do the prediction
 - If some of the less commonly tested biomarkers were counted as a comorbidity - summarise the other columns with one column - summarise meaning doing PCA?
 - One of the most challenging pieces of the process is data quality; in ML, 70% of the time is cleaning the data and making sense of the data, and 30% of the time is actually designing the algorithms
 - CRP plots: day of sample collection - the CRP is usually really high, then goes down and then it is sometimes low
 - PCT is stable at the beginning and goes up at the end
 - Clinician insight is that they died because of a blood infection
 - During Bernard's PHD he developed, ARIMA, etc - does not think they will work very well because we don't have a lot of datapoints
 - Will share two repositories with us today to help with this
 - Documented better than the django
 - With few data points a linear regression may be enough / cox regression too?

- Start with one day inference - based on time of admission - first readings
- Going straight to a time series may be a mistake
- Real problems the simplest approach is usually the best
- Patient similarity: 200-300 patients in this dataset - he collected whether or not patients were severe, stable etc
- Clinician Damien: He wrote also some comorbidities
- CBR chapter: patients that have similar conditions: do not need to do any prediction - just for the doctor to see similar patients - can compare the result of the patients - were they were admitted, length of their stay - so they can see what to do with the patients
- How to measure the similarity, need to find an external label or something to decide, for example how many biomarkers were outside the normal range -
- CBR in thesis uses KNN which uses euclidian distance
- Could try to do clustering, reduce the dimensionality to three so you can visualise it a little bit
- Measures that you use to evaluate the performance of a clustering algorithm are very different to the supervised learning problem
- Discussion of having different weights for each feature when doing case based reasoning
- Can we use the impurity scores
- Some people use impurity scores with decision trees or random forests - you can use those weights
- It is becoming a bit difficult to track progress
 - We need to have a repository so he can see what we are doing; It is a bit difficult to discuss things for bernard to see him
 - Every week can we have a summary with some slides with some tables and figures to show what we have actually look into the figures and details
 - Notebooks are really good for teaching but difficult for research
- Try to have a repository ready for next week monday/tuesday
- Playing with unsupervised is probably the best approach at the moment
- Yichen tried something similar with PCA to visualise them to see if there were clusters
- Probability of death for each cluster
- Synthetic way of generating data is maybe not the best approach?
- Clustering problem is right now just to get similar patients but using clustering to prediction is another level
- Plan for next few days
 - Get everything organised an on the repository
 - Choose development environment: pycharm, spyder + anaconda and get comfortable with it
 - Dealing with the new dataset
 - Getting our heads around django (wont happen until wednesday)
 - Start with some unsupervised approach on pathology dataset
 - Many biomarkers that are similar but they are the same thing

- For every test there is a range and the units - we could reduce the number of tests by trying to identify which tests are actually the same based on their names and units
- Try to output all the unique codenames and compare (manually)? By their ranges
- Individual assignments
 - Daniel and Ghaj focusing on Django
 - Divide the work in the preprocessing
 - Getting rid of one biomarker is repeated in different ways and having one patient per line
 - Yichen, Selena, Yannis focusing on the preprocessing, model design
 - Training random forest and using PCA to have ranking of different biomarkers
 - Visualisation in 3D
 - Yichen will try to make one element in df to be an array to potentially investigate time series data

12.1.4 Friday, 22nd May 2020

- Episode means from diagnosis of patient start
- Review of this weeks progress with powerpoint presentation
- Heat map
- Issues identified: Presence of NaN values and Effect of multiple corrected tests
- Clinicians would love to have data like/LOVE the kaplan meier fitter plots for survival plots
- Check if survival plot is correct based on time series for the pathology data
- Would use all biomarkers in the future to determine constant for different survival model curves
- Try to use complete profiles
- If there are 100 pathology markers - if there is 10 or 15 or 20 just focus on those ones, try to use it without you imputing anyone
- If half of rows are missing, then median affects results of algorithms
- Set threshold - keep biomarkers with only 90% of biomarkers
- Plot graph of number of biomarkers recorded
-
- Identifying covid infection - we would need negatives so we can do the testing with biomarkers so Bernard will send these
- Do a risk score because there would be more severe patients even within each category - this would be more useful
- Case based value with categorically encoded data
- Have reference on other groups doing similar relationships - not doing prediction, but comparing biomarkers to try to identify patients who are more likely to have covid
- Someone at Imperial College having promising results - doing some work - keep an eye on his research - probably not published yet
- How to determine the severity - would be based on hospitalisation length and time to death? Or time to discharge? Can consider if you have the biomarkers if they were out of the normal range and by how much

- To do over next few days
 - Send list with labels of different hospitals/icu identification to clinicians for confirmation
 - Select final list of biomarkers and Send final list to clinicians
 - Experiment with different models - decision tree, random forest
 - How to use survival models to predict patient survivals - assign risk score to patients
 - Need to make progress on Leaflet
 - Have a working pipe

12.1.5 Friday, 29th May 2020

Questions made for the meeting in advance

- We are a bit confused about what we are expected to deliver:
 - Inference module based ONLY on the pathology data set merged with the labels from the keira and sid
 - * Could the probabilistic inference module be a classification problem based on patient characteristics where we try to predict if a patient has covid from that information - i.e. predict the covid_confirmed column from the initial patient diagnosis?-> i think this has been answered it seems to be what they want
 - * I Know but it is based on the pathology only? He said to start with the keira
 - * I think he wants us to try look at them independently first? Let me ask him yes please
 - CBR : what is included?
 - * Should we include the medical treatments for the patients in order to evaluate the patient similarity ? Or is the patient similarity ranking a tool to help the doctors deciding with the treatment? How are we supposed to use neisha data set?
 - * We have been working on the cbr with different correlation metric - how to evaluate whether cbr is working?
 - * rank the patients that were most similar to most patients so we can start giving it a ranking when looking into bigger datasets?
 - * Is it just a patient ranking? Are we supposed to output something else ? Use the ranking for prediction?
 - * Correlation matrix at the beginning? Or should we every time calculate the correlation with the rest of the patients?
 - Risk score ? Are we still doing this ? if yes how ?
- Questions about the Keira dataset
 - We have only between 260 and 300 patient for which the all the columns/ symptoms were recorded - th rest only have date of admission and weather or not the patient is confirmed with Covid
 - * So should we use the covid confirmed labels; Sid dataset - just to add 400 new labels? and there are around 70 patients that are present in both data sets with different labels?
 - * If we are still trying to do a risk score, then should it be for those 260/300 patients ? (without the pathology)
 - * Severity ? how should we proceed?
 - * Should we try to cluster them/ autoencoder - have an unsupervised learning problem in order to solve it?
 - * Have a model that will combine the columns that we set as labels and try to cluster them ? or apply some dimensionality reduction?
 - How should we deal with patients with repeated rows?
 - We are unclear what the columns U, 65, CRB65, CURB65, Multi-lob.1, Lymph.1, Bac co, Smoker HTN.1, 60, MuLBSTA mean - they don't seem to be binary values so are these different levels of severity? For example Smoker has the following values : 0 2 3 or nan

- Ask Bernard about the data files because something weird happened - teams is weird
- Are we expecting more data? Are those the final data?
- Will Jcolibry be useful for us for the cbr module?

Client and Supervisor Meeting

- Yichen is working on Risk Scores
 - Random forest has optimal performance
 - Doctors input biomarker data and model estimates how severe the patients condition is to send them to the icu
 - Check how to verify a mortality score
- In pathology dataset every entry is a number: Number of biomarkers outside the normal reference range - using label encoded data to predict severity level of patients
- 3 or 5 biomarkers to differentiate between severity
- Sid is just a list of patients until the end of april
- Nisha has something similar as well - positives and negatives - prescriptions for a few of them
- Microbiology - although covid is a virus - also has the microbiology dataset - label if microbiology is positive or negative
- Missing urine
- Wouldnt rely much on micro but rely on covid
- Keira - would rely on keira now - not only numbers but has covid confirmed outcomes - those tests have been done for both positives and negatives
- Start by focusing on these 1500 patients
- Only about 300 that they have vital signs collected
- Clinicians started collecting vitals for negative patients so we have more of a balanced dataset
- Can create daily profiles for patients with keira and pathology
 - Try to differentiate between covid positives and negatives
 - Daily profile: with pandas for each patient, group biomarkers for each patient and each day
 - From daily profiles can see how many are 'full' profiles and how many are actually missing and then decide on which biomarkers to use based on which are missing
 - With daily profiles you can consider one patient on multiple occasions
 - How to treat different patients if you dont use daily profiles
 - If you have temporal data you can include it after a few days - but you have to take into account that if you need a feature that is the variability between two dates that you have to wait until the second date - this is a way to have more entries to the algorithm to consider each day for a patient as an entry instead of just taking each patient as a case
- For cbr patient history and how to interpret patient history
- What Bernard would do to start with
 - Get keira dataset - around 300 patients where we have the symptoms
 - Keira collected manually data on patients - only on 300 patients that were positive - doctors will do the same

- For PI focus on biomarkers - start with keira
- Focus on PI - get it for next week - show prediction and some roc curve
- For next week we will have new data and then we can do the cbr
- Have labels - get data from pathology
- Can count the positive patients as positive - assume the patients are positive
- Keira and sid - patients in keira marked as negative without being marked as patients
- Forget about sids and just go with keira's one
- Assume that if they are once they are positive
- Unsupervised learning is part of his idea for cbr - could start doing this for pathology
- They probably prefer to do the clustering with the vital signs because thats more clear to doctors
- Unsupervised - any good type of clustering
- List of uncertain variables - these seem to be computed aggregated metrics - focus on the actual raw data
- Smoker columns - dont focus on this too much
- How she had it - excel spreadsheet - two books one with negative and one with negative- different levels of severity in text - we can just use whether or not they were positives or negatives
- For those doing inference - have a look at pysml - it may take us a bit of time at the beginning to learn but it should work pretty straight away to configure the classifier if they are from scikit learn
- For training set - what he did is try to train only with full profiles
- Instead of 20 biomarkers he used only 6
- not bad if you use the median - use the median for positives and negatives
- Focus on biomarkers based on clinicians
- Created a file just counting the number of biomarkers
- Select the top 20 biomarkers that have more than 5000
- To do for Keira dataset
 - Keep only one _uid per patient
 - Keep only labels and uids from keira
 - Make all patients one line - temporally split patients by day of results recorded
 - Take the first day on admission biomarkers recorded
 - Split into train/test
 - Remove all biomarkers that werent recorded enough
 - Remove all patients that did not have enough biomarkers
 - Function rankColumns(df)
- To do by Saturday
 - Plan the documentation sections
 - Function keepColumns(df, keep)
 - Function to keep top biomarkers
 - Choose the day that has the most readings for each patient instead of first day for patient?

- Hour or 2 to look at pysml
- Basic functional probabilistic inference
- To do by Sunday
 - Do exactly the same thing as we did today but with Bernard's repository
 - Test set should be the patients that are mostly filled out
 - Finishing Leaflet
 - Build PI models
 - Check difference between using label encoded data vs values for bio-markers
- Install instructions on macOS Mojave
 - On
 - Check python is in the path with: `python --version`
 - Check pip is installed with: `pip --version`
 - If it is not installed: `brew install pip`
 - Install virtualenv: `sudo -H pip install virtualenv`

12.1.6 Friday, 5th June 2020

- Try to see with 10 biomarkers, acc will stay the same but we have lower hold out set
- Reduce the number of biomarkers we need that are more common so we have larger hold out set
- Try with XGBoost - should be superior to AdaBoost
- If you want to start with the html: django - can use html template
 - It has examples for forms and buttons etc
 - API currently working, for file retrieval still needs integration with front end
 - Purpose of front end is to make it look nicer
 - Front end uses html, css, javascript
 - Can have it separately in an html file or use the templates
 - Query to the server - refresh button
 - Form in CBR is adding something new (the one to actually add patients to the database)
- Selena: what are we expected to do with the cbr
 - Prediction based?
 - Retrieval - he has a number of parameters that they need to collect and we should narrow it down
 - Purpose of the cbr, is for prescription to decide what therapy to apply based on the similarity
 - If two patients had exactly the same treatment then they had a similarity of 1
 - Do not have that many patients with antibiotics
 - Computed similarity based on symptoms but optimised it based on the treatments
 - Started doing cbr to determine which of the three outputs icu, go home or hospitalisation
 - Use hospital, icu, send home but would have a much larger dataset if you use pathology
 - As for checking if it really works, could only really do using clustering and silhouettes
 - To try to verify it you can consider the severity of their outcome, if they were in the icu
 - Daniel to merge his branch with Ghajs and then to merge to master today for django
- Presentation on the 18th June

- Get some nice graphs because we have a lot of research
- Make the django a bit nicer and that already looks like an app with everything working together
- Make sure it is clear in which part you have been working on in the presentation
- They are going to be marking and there will be two other markers
- Send presentation on the last friday before the final presentation
- Final deadline is on the 25th June
- Probabilistic Inference: Yannis
 - Try some Keras networks ANNs
 - Try XGBoost
 - Try RandomForest
 - Reduce number of biomarkers
 - Increase the size of the hold out set
 - Documentation in code
- Case Based Reasoning: Yichen and Selena
 - Actual cbr code
 - Documentation in code
- Django: Ghaj and Daniel
 - Front end css, html, templates
 - Edge cases
 - Error handling
 - Documentation in code
- Documentation
 - Introduction: Yannis
 - Problem Description: Yannis
 - Literature Review: Yichen
 - Design Criteria: Yichen
 - High Level Description: Ghaj (add diagram of pipeline)
 - Preprocessing: Selena
 - Probabilistic Inference: leave for later
 - Case Based Reasoning: leave for later
 - Django Application: Daniel (only)
 - Conclusion and Evaluation: leave for later
 - Report of ethical consequences: Ghaj
 - Sustainability report: leave for later ask clare description makes no sense
 - Record of meetings held: Yannis
 - Future Work: leave for later

12.1.7 Friday, 12th June 2020 (Questions with Answers from Bernard)

- How would you expect to run the algorithm to run:
 - Constructing a correlation/distance matrix every time a new data is uploaded and then retrieve the n most similar patients when the algorithm is asked to?
 - Or should we compute all the distances between the patient and the rest of the database every time the doctor makes the request.
 - * You will need to compute the distance between the query case (patient) and the rest of the cases in the database every time the doctor queries a patient. You will need to select the features and format them before inputting it into the algorithm. To save time you can have a numpy with the case base already formatted stored in the server.
- The weights of the features for the computation of the distance are initialised using the data and labels we have in the dataset. Should we rerun this algorithm with the new data or just use the weights that we have already computed?
 - Identify a set of weights initially and these will remain constant during the trial.
- Are we the one who defines the features/and input the value of the test patient if needed or do the doctor select the feature the algorithm needs to take into account while computing the distance?
 - Define a number of variables that will be required and the doctor will have to input them, otherwise the None value will be used.
- Should we select the n most similar patients or should we find a threshold of the distance which is then used to retrieve the patients? Is it supposed to be a default value or should the doctor set it?
 - Do the most simple one to start with (n most similar cases).
- For the keira dataset we are computing the weights and selecting the features based on the following labels: death, ICU, CRB65 and CURB65. For the pathology dataset we are considering using the old pathology dataset so we can use the values we have in the patient data set and have more training points, but the weights of the feature are only based on the ICU/death. Is there any other evaluation method you would suggest?
 - Death, ICU, severity, length of stay?
- We have been tuning the HDBSCAN algorithms for clustering the pathology data. We now can make it cluster aggressively (bigger clusters, less outliers) or conservatively (more and smaller clusters, more outliers) depending on the need. What does cbr aim for in the end? Should we leave more outliers for the doctors to look into case by case (the conservative approach) or we provide a more generic picture of patient similarity for the doctors (the aggressive approach)
 - Will need to see how it behaves, have you attempted to plot them? I presume conservative is better.
- Most of the clustering algorithms are applied on the status dataset (based on in range, out of range and severely out of range), the imputation for such dataset is tricky especially when the results are difficult to evaluate. Any suggestions for the imputation or is it a good idea to just never impute data for clustering?
 - Dont input data for clustering. The value hence will be none. You can chose whether to consider two cases with one parameter as None as similar (hence assuming that data missing might actually contain information, for instance doctors may ignore the collection of certain data if not relevant) or just ignore them.
- The results are difficult to evaluate since the only references we have are the outcomes of the patient (hospitalisation, icu admin, and mortality) which does not entirely reflect the similarity of the patient from the pathology dataset.

- Yes, CBR is difficult to evaluate. For pathology dataset you can create some labels counting the number of markers out of the normal reference range, creating a bool array where 0 indicates within range and 1 out of range [0, 1, 0..] and counting similarities, looking at absolute value, distance between values...
- Apart from the HDBSCAN, we also tried a few basic clustering algorithms based on euclidean distance or centroid like kmeans, the number of cluster is difficult to determine despite using the ‘elbow’ distortion and silhouette and had pretty bad performance for the discrete dataset. Any improvements can be made or should we stick to the more advanced (density based) clustering methods which are more difficult to optimise.
 - Don’t know.
- Probabilistic Inference; just a discussion of Results no more questions
- DJANGO
 - The project route plan document mentions three CBR tasks: “Only pathology”, “Only vital signs” and “Combo”. Should each of these algorithms have their own corresponding screens in our app?
 - * Those were the possible options, there is no need to finalise all of them. Each algorithm should have its own screen (since the features to input -data entry- will vary).
 - For the CBR module, are we aiming to let doctors only select a patient from the app’s database? Or do we want to be able to make predictions using new cases introduced either manually or using a CSV file, as we are doing in the inference module?
 - * Lets assume they will select a patient from the database. The other functionality would be good but not needed. They can always use the admin to create cases for a patient and test them.
- Two different parts: display models, inputs, outputs and etc
- Different view for each of the algorithm and integrate in the interface
- 3 options:
 - form and manually input data
 - excel attach outputs in extra column -> could have one column per algorithm
 - querying the database
 - Can have two or three top models
 - Have a folder with probabilistic inference algorithms, have a readme and pickle file
 - * Input parameters, labels, description of algorithm, configuration
 - * Get pickle file and explaining
 - * Can write a script that plots the pickle file in a json file
- Video presentation (5 minutes)
 - Put things in context
 - What are we trying to solve
 - Challenges
 - Problem
 - Different things weve worked on
 - Put some of the graphs
 - Last minute or minute and a half on django app
 - Probability score
- Send a preview on the video and we can get some feedback

- To do:
 - Probabilistic Inference: finished so just some more code comments
 - Yichens pipeline by tomorrow evening
 - Yannis 3 pipelines by tomorrow evening
 - Selena - pathology cbr (yannis to help with filtering, imputation)
- Video
 - Plan the structure
 - 5 slides
 - Finish the video by Tuesday
 - Make a folder in the google drive
 - Take screen recordings
 - Graphs
 - Voice notes/recordings to the drive
 - Yichens script

12.1.8 Friday, 19th June 2020

- Agreed with other markers that we did a lot of work, it was difficult, highly technical
- Delivery aspects to improve:
 - Some people don't know about machine learning (need to explain things at a bit a lower level)
 - Explain the basic concepts very quickly: probabilistic inference, binary classification
 - Some slides with way too much text: try to explain it with an image especially remotely, picture or animation would be much more visually effective
 - If you put bullet points then at least try to animate them Presentation more about the delivery and not the content
- Good presentation but for guys that did not know the project, they didnt know how it would be used, they didnt know how things would be put together
- Video was good
- In the presentation it was a bit chaotic - slides were a bit chaotic
- In terms of technicality, probably the most sophisticated/best project
- This mark is more about the delivery and not the content
- There will be another mark on the documentation: there would probably have the highest mark because
- Spend one slide or two: literally say what is machine learning, e.g. when you are presenting to clinicians
- Three concepts before presenting the rest of the ideas
- Machine Learning is a technology with like 10 years, some professors that maybe do not understand what's going on
- One of the points: if we were using layman's terms, Industrial project: talk in less scientific way
- 5 slides is crazy - might take longer to prepare a 5 minute presentation than a 30 minute presentation
- A lot of material online about how to prepare slides, tutorials on how to prepare slides, types of colors, fonts
- Next steps: Documentation with deadline thursday next weel

- Focus on the documentation because it is what is going to be evaluated
- Need to have a report
- Explain where they are going to be used and explain where they are from PI probability of infection
- CBR so doctor can see similar patients
- STATE THE PROBLEM very well at the beginning
- Introduction: should understand immediately what you are trying to do with the project
- Very nice tables and very nice graphs
- Try to put figures
- Make sure the captions of the figures are explained as much as possible
- Some supervisors obsessed with captions so figure+caption should be self explanatory
- Try to send a skeleton of the document
- Can give more detailed feedback before we submit
- Can comment on alternatives that you investigated too not just the one you selected, but don't focus too much on what you did not use
- Schematic from the beginning that shows how the system is going to work: nhs servers, pulling the data,
- Bernard to send a screenshot of the final app and how all the information is presented
- Bernard can send diagram of NHS side of the system - three different databases that you have to manage together
- Should we call it CBR for the report because it could lead to confusion
- Introducing new terms: Similar Patient Retrieval
- Explain the 4 steps of CBR but say we are focusing on just the first step of the CBR: similar retrieval
- Can do a diagram: reuse, revise, retain
- Bernard has the latex graph for the CBR and said he can send it
- Common for CBR when you have massive case bases, you organise your case base, you may discard two thirds of the database for efficiency and search a subset of cases
- Prototypes approach (further work)
- One big problem in CBR is called case based maintenance - deleting outdated cases
- There are no very good python libraries for cbr
- Making an algorithm is hard because it is more like a framework
- discussion/future work section is very very important

12.2 Group's Independent Progress Meetings

12.2.1 Tuesday, 30th April 2020

- Deciding on method of taking minutes; options Otter transcription software, recording zoom calls and using another transcription software
- Discussion of need for continuous integration; this will probably only become relevant later in the project
- preliminary meeting tomorrow: more details on problems we will solve, ethical considerations
- Discussion of deliverables
- Documentation should be collected as we go
- Source code is kept on Github
- Branches used for independent demonstration
- Video should cater to technically aware users and also be more general
- Leaflet can be designed in Sketch - again closer to the time of the deadline
- Report to be made in Latex
- Discussion of Interpretability of results - neural networks a bit of a black box
- Proportion of marks allocated to portfolio is unclear
- Vaguely the way we understand the objective is: what kind of patients need what kind of treatment
- Multiple models for multiple datasets
- Using multiple datasets would mean identifying which covariates are important across datasets
- Mention of AI for Medical Prognosis Course - whether it is worth investigating this - Selena has already started going through this
- None of the datasets available has time series style data available which makes prediction for individuals more difficult
- Yichen found some datasets from China but they are in Chinese - still to be uploaded
- Plan is to use Keras and Tensorflow because it will mean less problems with environments and less setup
- Trello board - some kind of scrum/sprint/project management
- Daily standup every morning - especially at the beginning to know what everyone is working on

Specific questions for tomorrow:

- Ask if he can help us draft a plan - we are not sure what time scale is realistic to set intermediate goals to make the project more manageable
- How is the output going to be used? How interpretable the model should be ?
- How do we justify the ethical implications of the results of our research?
- Missing data - should we use imputation to evaluate missing data? What imputation methods would he advise us to use? Implications of this in biological context / data augmentation

12.2.2 Monday, 4th May 2020

Recap of the last few days:

1. Yichen & Daniel

- trimmed the data and uploaded to google drive
- Daniel did some trimming with pandas
- Making data binary
- Converting things like lymphocytes to standardised or binary data to feed into training
- It is okay to combine numerical and binary data
- Trying to use SMOTE
- Limitation is that we only have one or two survivors - very imbalanced
- From Bernard - we will have more data
- Desire to explore other datasets; warned that incoming datasets will also be imbalanced
- Found Towards Data Science very helpful resource
- You can extract certain meaning from different ranges of the numbers
- Tried to use PCA for dimensionality reduction
- Look at notebook on git - Daniel's is on colab and Yichen too
- Yichen looked into the first data set that we thought would be useful but it also does not have useful information
- Scikit learn and
- Tried Stochastic Gradient Descent and did not work very well

2. Ghaj

- looking at COVID related biological data
- Looked into different types of ventilators being used
- Prevalence of obesity in people that needed respirators
- Got in contact with a few other people who are working on COVID research
- Reaching out to people who may have access to other data which we could potentially use - someone at U Chicago
- Still waiting to start Django tutorials

3. Selena & Yannis

- Gaussian Bayes Classifier
 - assume independence, compute mean and variance of each variable and based on that classify risk
 - simplest approach, does not need a lot of data, relatively low accuracy
- Decision Trees & Random Forests
 - non-linear data
 - Could use decision trees to initialise weights to shallow neural network
- SVM Classifiers
 - Require more data so may be a bit too early
 - NN - could change loss function to account for minority classes to deal with imbalanced classes

Next Steps

- Each of us will explore an avenue and pre-process

- Daniel will send a Python ML preprocessing tutorial
- Daniel: Random forests - many decision trees and patient similarity ranking, K-Nearest Neighbours
- Yichen: preprocessing and SMOTE - synthetic upsampling
- Selena: Random forests
- Yannis: Multivariate logistic regressions Two-layer Random Forest - check Titanic, imputation, tutorial, identifying outliers
- Ghaj: tutorials and reaching out to other research groups

For Later

- Gaussian Naive Bayes
- SVMs
- Autoencoders
- Probability based models: Bayesian networks
- Try to get a risk score

12.2.3 Monday, 11th May 2020

Do not have everything we need New dataset Repository access **What have we done:**

1. Ghaj:
 - Same position as before on Friday ; working on django stuff, should be done with tutorial in next Friday
2. Daniel:
 - Finishing app, reading blog articles from towardsdatascience, reading through other people's notebooks
3. Yannis:
 - ROC and AUC: used about
 - Hyperparameter grid search on multivariate logistic regression, random forests
 - Simple SMOTE and 5/10 fold cross validation
4. Yichen:
 - How the data is suitable for clustering method
 - Will: do more about how to use justified way of synthesising the data
 - Synthesised data only for training not also for testing
 - N-fold cross validation
5. Selena:
 - Random Forests
 - Imputation
 - Gini impurity: something already in scikit learning, gives pretty good and robust ranking of all the features
 - Decide which features we want to keep
 - Less values = bigger probability in not imputing values in the columns; can get better indication of how the imputation works

Things to do/think about:

- Way to have a risk score as opposed to having 3 probabilities
- Using grid search on different algorithms being tested
- **C-index**: coursera course, coming up with a risk score

THIS WEEK ON THE COVID-SHOW

1. Daniel: patient ranking
2. Yichen: cross validation, balancing, imputation
3. Yannis & Selena:
 - Random Forests
 - ROC AUC
 - Hyperparameters tuning
 - Features Selection
 - Different algorithms:
 - XG-Boost
 - Light GBM
 - Gradient Boost
 - Research on finding risk scores
Further Information on work necessary:
 - Estimators; Logistic regression, xgb, light gbm, gradient boost, random forest, decision tree, knn, svm, ann
 - Hyperparameter (hosp, ICU, death) tuning with and without smote; plot curve for each one separately
 - All the figures for each estimator together
 - Performance metrics to use include test accuracy, precision, recall, F1-score, and AUC.

12.2.4 Thursday, 14th May 2020

Current Work:

1. Daniel
 - Looking django at repository - but the repository is messy we should ask the supervisor
 - Ranking patient - but he is a bit lost → how should we do it in python
 - Why would we use it ?
 - * Classification
 - * Imputation
 - Maybe we can work on it later
2. Yichen
 - Smote and Cross Validation
 - Weird results :
 - * Having the same accuracy with when we train the random forest
 - For the new data:
 - * NHS doesn't match - 837 nhs numbers out of the 1079
 - * We have 2 datasets
 - Pathology
 - If it was to be used - unsupervised learning
 - Time series which varies from patient to patient (some of them are recorded daily)

- Incorrectly label - human error → outliers
- * Dates - that can be useful survival
- He thinks that 2 people should work on the project

3. Selena & Yannis

- Comparing the performance of different estimator - with/without SMOTE + with and without the hyperparameters tuning
 - Have them saved in data frame
 - Draw: roc pr and calibration curves
 - Promising result despite the small initial data set
 - Tried different estimators:
 - * Xgboost
 - * Random forest
 - * Logistic regression
 - Potential estimator:
 - * ANN
 - * SVM
 - * Light gbm
 - * Gradient boost
 - * Testing the performance should be automatic now
- Working on the data set :
 - Non pathological data:
 - * Since there are not many missing values those assumptions were made:
 - People who have a discharge date but no death date have been considered to to have at least lived until the were discharged
 - People who don't have a discharge date is because the might still be at the hospital
 - Key information that were extracted from the data:
 - * Hospitalisation length
 - * Episode length
 - * Time to death + a bool that says if the death was observed
 - The fact if the patient was admitted to the icu might be deduced from the warren number but we don't how to interpret them
 - Pathological Data:
 - * Not many info is present
 - * Each patient is represented by several rows which indicates ever biomarker that was tested
 - * We have time series for some patients and some not
 - * There are many biomarker over 250 labels and some of the labels represent the same biomarker but written in a different way
 - Using the units of the test might be used in order to group biomarkers that represent the same thing
 - * As part of the first preprocessing we have chose to keep only the first test of each biomarker, disregarding the time series:
 - Are we supposed to deal with time series knowing that our inputs are just going to be on admission results?
 - Using random forests to classify the importance of biomarkers

- Also we can make column out of the most important biomarker and have a last column which regroups the rest:
- For example the last column can represent the presence of comorbidities but we need to find a way to have a weighted sum

We can use smaller sets to train and test our models.

Questions for Meeting with Client and Supervisor

- the repository is messy ? If he can walk us through the repository.
- Ranking patients
- New data sets :
 - NHS doesn't match 837 nhs numbers out of the 1079
 - 74 patients with the same NHS Number have 2 or 3 different patient IDs - ?
 - We don't have people have enough people that were sent home
 - Warren id → the number might be that icu but we don't know about it
 - * NHS number don't match
 - * The fact if the patient was admitted to the icu might be deduced from the warren number but we don't how to interpret them
 - * Only sent home only 64 spent 0 days at the hospital
 - * Can we disregard patient who weren't discharged yet
 - * Due to the little information we have about the label that we have - should we tackle the problem as an unsupervised learning problem and then use the data point we found in a way to test it
 - Also if wanted to rank the severity of the patients based only on the pathological data - is there a way to do it? so we can develop a risk score and find the threshold later
 - Make sure that the assumptions we made are correct
 - * Time series which varies from patient to patient (some of them are recorded daily) - Are we supposed to deal with time series knowing that our inputs are just going to be on admission results?
 - * We have many biomarkers can make column out of the most important/most frequently found (tested at least once to most patients) biomarker and have a last column which regroups the rest:
 - For example the last column can represent the presence of comorbidities but we need to find a way to have a weighted sum?

12.2.5 Thursday, 4th June 2020

1. Things that we still need to work on
 - (a) Loading pickle files
 - (b) Getting model hyper-parameters
 - (c) Inference part of Django in progress: Aim for MVP tomorrow
 - (d) CBR Django part is working but we need algorithm working

- (e) Need a way to evaluate the similarity ranking
- (f) Labels for patient severity - ask about it again on Friday
- (g) Investigate viability of using time series data (though this is of low importance based on time constraints)
- (h) Try to work on Keras models - NNs and LSTMs - maybe to think about a bit later
- (i) Bernard and Pau advised to leave a small part in the report about things that people could do in the future (we need to have some research if we want to do this in the future)
- (j) Discussion of models investigated for the report and presentation
- (k) Daniel: NNs performance not great w/best results acc; with completely balanced dataset, best acc 0.6, sens 0.4 specificity 0.8, may be overfitting
- (l) Added a threshold which disregards a patient that have too many values imputed
- (m) PassiveAgressive: worse version of SVM
- (n) OnevsOne: way of dealing with multi0class classification
- (o) Leaflet is good, need to fix a typo

2. Next checkpoint: by the next time we meet we want to accomplish the following things:

- (a) Daniel finishing MVP for inference part on Django
- (b) Selena is making pathology transpose for pipeline and pre-processing (integration)
- (c) Yannis: Gradient Boosting and Ada + plotting
- (d) Yannis: further investigation for probabilistic inference
- (e) Yannis: Pipeline tuning optimisation scripts
- (f) Yichen: plots and boosting and Ada
- (g) Ghaj: finish up inference module

A Appendix

A.1 Imputation Research Results

Estimator	Imputer	id_estm	roc	prauc	roc_auc_score	seis	spec	acc	log_loss	train_acc
2	LGBMClassifier	SimpleImputer('median')	0.8697222222222222	0.9161681555262020	0.7836111111111111	0.8088888888888890	0.7583333333333333	0.7913043478260870	7.208159677562450	0.82038883495145630
19	KNeighbors	IterativeImputer()	0.8060185185185190	0.9017025871228040	0.775	0.8	0.75	0.78240086956521740	7.508499181278860	0.8398058252427180
35	xgb	KNNImputer(n_neighbors=3)	0.880462962962963	0.9148654579651660	0.8066666666666670	0.8800000000000000	0.7333333333333333	0.8289855072463770	5.906705491065210	0.941747572815534
24	DecisionTree	KNNImputer(n_neighbors=2)	0.8786111111111110	0.9276257669279630	0.8050000000000000	0.8933333333333333	0.7166666666666670	0.8318840579710150	5.806597731065060	0.8640776690929130
22	RandomForest	IterativeImputer()	0.8725925925925930	0.9216678636212750	0.8033333333333333	0.9066666666666670	0.7000000000000000	0.8347826086956520	5.706489971064920	0.8398058252427180
27	LogReg	KNNImputer(n_neighbors=2)	0.8325925925925930	0.8631238136233400	0.7583333333333333	0.8666666666666670	0.65	0.791304347826087	7.208189807320690	0.8106796116504850
0	DecisionTree	SimpleImputer('median')	0.8498148148148150	0.9078819704860200	0.7541666666666670	0.8666666666666670	0.6416666666666670	0.7884057971014490	7.3083045203419700	0.8592223009708740
15	LogReg	SimpleImputer('most_frequent')	0.8453703703703700	0.899637848397087	0.7697222222222222	0.8977777777777778	0.6416666666666670	0.8086956521739130	6.6075177529099000	0.8058252427184470

Table A.1: Results obtained for varying imputation methods (best model for each estimator from the grid search is shown)

A.2 Re-sampling Research Results

Estimator	Sampler	id_estm	roc	prauc	roc_auc_score	seis	spec	acc	log_loss	train_acc
0	DecisionTreeClassifier	RandomUnderSampler()	0.917593539927142	0.8897222222222222	0.7977777777777778	0.7955555555555556	0.8	0.7971011492753623	7.0079232984987625	0.8058252427184166
1	KNeighborsClassifier	RandomUnderSampler()	0.8812610191912185	0.7477777777777778	0.7477777777777778	0.7955555555555556	0.7	0.7623188405797101	8.2092909854754098	0.8308625242718416
2	LGBMClassifier	RandomUnderSampler()	0.8266221607877213	0.6527777777777778	0.6527777777777778	0.6888888888888889	0.6166666666666667	0.6637681159420289	11.61314447330849	0.5937087378640777
3	RandomForestClassifier	RandomUnderSampler()	0.9306020700445824	0.9306020700445824	0.8080555555555556	0.8577777777777778	0.7983333333333334	0.8231884057971015	6.1069233287392199	0.7864077669902912
4	xgb	RandomUnderSampler()	0.9081828924894201	0.7644444444444444	0.7644444444444444	0.7288888888888889	0.8	0.7536231884057971	8.509600922871227	0.7864077669902912
5	DecisionTreeClassifier	RandomOverSampler()	0.926280907809284	0.7916666666666666	0.7916666666666666	0.8177777777777778	0.7500000000000001	0.7971011492753623	7.0079232984987625	0.7961165048543889
6	KNeighborsClassifier	RandomOverSampler()	0.8981120761247947	0.7816666666666666	0.7816666666666666	0.7733333333333333	0.7500000000000001	0.76521739139043476	8.109173553364267	0.8155339805825242
7	LGBMClassifier	RandomOverSampler()	0.9163137896105994	0.8019444444444444	0.8019444444444444	0.7955555555555556	0.8083333333333333	0.8	6.907808585477482	0.8155339805825242
8	RandomForestClassifier	RandomOverSampler()	0.9319053905097908	0.8061111111111111	0.8061111111111111	0.8622222222222222	0.7500000000000001	0.8231884057971015	6.1069233287392199	0.8349514563106796
9	xgb	RandomOverSampler()	0.900547378069922	0.8044444444444444	0.8044444444444444	0.8755555555555556	0.7300000000000001	0.820980956217392	6.006817886412773	0.961165048543889
10	DecisionTreeClassifier	SMOTE()	0.9229295017069696	0.7883333333333333	0.7883333333333333	0.8933333333333333	0.6833333333333333	0.820289850724639	6.207056583150173	0.89805825242718416
11	KNeighborsClassifier	SMOTE()	0.906242795073691	0.7694444444444444	0.7694444444444444	0.8088888888888889	0.725	0.7797101449275362	7.60861852964756	0.8058252427184166
12	LGBMClassifier	SMOTE()	0.9027273476232154	0.7680555555555556	0.7680555555555556	0.8444444444444444	0.6916666666666667	0.7913043478260869	7.208178218952135	0.83009708737864077
13	RandomForestClassifier	SMOTE()	0.9069379563141904	0.8019444444444444	0.8019444444444444	0.9288888888888888	0.675	0.8405797101449275	5.506272133390914	0.8592233000708737
14	xgb	SMOTE()	0.923921841492272	0.7963888888888888	0.7963888888888888	0.8844444444444444	0.7083333333333334	0.8231884057971014	6.1069233287392199	0.9514563106796117
15	DecisionTreeClassifier	TomekLinks(sampling_strategy='majority')	0.9344746257414822	0.8052777777777778	0.8052777777777778	0.9088888888888888	0.6416666666666666	0.8590724637681159	5.0057194273479215	0.844660141747572
16	KNeighborsClassifier	TomekLinks(sampling_strategy='majority')	0.8917814364480577	0.7461111111111111	0.7461111111111111	0.8088888888888888	0.6833333333333333	0.76521739139043478	8.109192094753952	0.83009708737864077
17	LGBMClassifier	TomekLinks(sampling_strategy='majority')	0.881412208768103	0.5	0.5	1.0	0.0	0.65217391390434783	12.01376562553349	0.7378640776699029
18	RandomForestClassifier	TomekLinks(sampling_strategy='majority')	0.8804629629629629	0.9294576061551048	0.9294576061551048	0.9911111111111111	0.6333333333333333	0.8666666666666666	4.605272163631362	0.8786407766990292
19	xgb	TomekLinks(sampling_strategy='majority')	0.8799074074074074	0.9248016475444444	0.9248016475444444	0.7852777777777777	0.8622222222222222	0.8086956321739129	6.6074992115193885	0.9514563106796117
20	DecisionTreeClassifier	SMOTE Tomek(sampling_strategy='auto')	0.9214927136666664	0.8011111111111111	0.8011111111111111	0.9022222222222222	0.7	0.8318840579710145	5.866602866412182	0.8592233000708737
21	KNeighborsClassifier	SMOTE Tomek(sampling_strategy='auto')	0.90519183098258	0.90519183098258	0.90519183098258	0.7777777777777778	0.75	0.7681159420289854	8.009061158016697	0.8349514563106796
22	LGBMClassifier	SMOTE Tomek(sampling_strategy='auto')	0.908571022985034	0.908571022985034	0.908571022985034	0.8533333333333333	0.7333333333333334	0.811594202898508	6.507379863150612	0.8009708737864077
23	RandomForestClassifier	SMOTE Tomek(sampling_strategy='auto')	0.8785185185185185	0.9192667953368407	0.9192667953368407	0.9466666666666666	0.675	0.85217391390434783	5.116582552000645	0.883495145631068
24	xgb	SMOTE Tomek(sampling_strategy='auto')	0.8746296296296296	0.910044786582512	0.910044786582512	0.8666666666666666	0.675	0.81739139043478259	6.307157390129187	0.912921359223301
25	DecisionTreeClassifier	SMOTE Tomek(sampling_strategy='all')	0.9219392184615458	0.9219392184615458	0.9219392184615458	0.8888888888888889	0.6916666666666667	0.820289850724639	6.207054265476462	0.854389320388349
26	KNeighborsClassifier	SMOTE Tomek(sampling_strategy='all')	0.8940932375465487	0.8940932375465487	0.8940932375465487	0.7955555555555555	0.7166666666666667	0.7681159420289856	8.009070428711539	0.8349514563106796
27	LGBMClassifier	SMOTE Tomek(sampling_strategy='all')	0.8451851851851853	0.894101869367099	0.894101869367099	0.8022222222222222	0.6916666666666667	0.820289850724639	6.80728637951807	0.8640776699029126
28	RandomForestClassifier	SMOTE Tomek(sampling_strategy='all')	0.9092309571406144	0.7902777777777778	0.7902777777777778	0.8888888888888889	0.6916666666666667	0.820289850724639	6.207054265476462	0.8155339805825242
29	xgb	SMOTE Tomek(sampling_strategy='all')	0.8935058213585556	0.9135058213585556	0.9135058213585556	0.8666666666666666	0.7249999999999999	0.81739139043478261	6.307157390129189	0.941747572815534

Table A.2: Results obtained for different scaling methods applied (best model for each estimator from the grid search is shown)

A.3 Feature Scaling Research Results

Estimator	Inputter	Sampler	Scaler	id_estm	roc	prauc	roc_auc_score	seus	spcc	acc	log_loss	train_acc
0	LogitClassifier	SimpleInputter(strategy='median')	StandardScaler()	301	0.89759296296296	0.94011060422580	0.8075900000000000	0.7733333333333333	0.5416666666666670	0.797104492753620	7.007931710130210	0.8009708375364080
1	xgb	KNNInputter(n_neighbors=3)	StandardScaler()	1	0.83796296296296	0.900338176271030	0.8105555555555556	0.8711111111111111	0.75	0.82985557236377	5.90670655717790	0.9466019147475730
2	RandomForestClassifier	IterativeInputter()	StandardScaler()	47	0.89435185185185	0.9384505147871230	0.8244444444444444	0.9155555555555556	0.7333333333333333	0.8521239130434780	3.10580632828467	0.8349514663106800
3	KNNeighborsClassifier	RandomOverSampler()	StandardScaler()	14	0.8150000000000000	0.8964515353382950	0.7533333333333333	0.7733333333333333	0.7333333333333333	0.7394292836590720	8.309402939406820	0.79611650948546600
4	DecisionTreeClassifier	RandomOverSampler()	StandardScaler()	26	0.8444444444444444	0.905468327464944	0.7702777777777778	0.8166666666666667	0.75	0.8202828282828283	7.11082952681150	0.8454352628282828
5	LogitClassifier	KNNInputter(strategy='median')	MinMaxScaler()	261	0.8407222222222222	0.9288409871447010	0.8166666666666667	0.8755555555555556	0.7583333333333333	0.7847926486945630	5.49829965694600	0.8454352628282828
6	xgb	IterativeInputter(n_neighbors=3)	MinMaxScaler()	43	0.8407222222222222	0.9288409871447010	0.8166666666666667	0.8755555555555556	0.7583333333333333	0.8347926486945630	5.706473717294800	0.8466019147475730
7	RandomForestClassifier	RandomOverSampler()	MinMaxScaler()	40	0.8962962962962963	0.940818201689088	0.8311111111111111	0.9288888888888889	0.7333333333333333	0.8688666652173910	4.805460142241070	0.8466019147475730
8	KNNeighborsClassifier	IterativeInputter()	MinMaxScaler()	0	0.7845259259259260	0.876715566960040	0.7152777777777778	0.7555555555555556	0.675	0.727536218840580	4.11065551946830	0.8009708375364080
9	DecisionTreeClassifier	KNNInputter(n_neighbors=2)	MinMaxScaler()	271	0.8057407407407410	0.943771563728729	0.8052777777777778	0.8444444444444444	0.7666666666666667	0.8173913043478260	6.310745801760630	0.8252427184466020
10	LogitClassifier	SimpleInputter(strategy='median')	MinMaxScaler()	23	0.8821296296296300	0.922478729290110	0.8036111111111111	0.7688888888888889	0.8166666666666667	0.79429028085307250	7.108024105477750	0.7961076696902010
11	xgb	KNNInputter(n_neighbors=3)	MinMaxScaler()	301	0.8821296296296300	0.922478729290110	0.8036111111111111	0.7688888888888889	0.8166666666666667	0.8173913043478260	6.310745801760630	0.7961076696902010
12	RandomForestClassifier	IterativeInputter()	MinMaxScaler()	32	0.8821296296296300	0.922478729290110	0.8036111111111111	0.7688888888888889	0.8166666666666667	0.8173913043478260	6.310745801760630	0.7961076696902010
13	KNNeighborsClassifier	RandomOverSampler()	MinMaxScaler()	4	0.7937037037037040	0.8741793531559870	0.7233333333333333	0.7466666666666667	0.75	0.849273623188410	5.205014088284820	0.8252427184466020
14	DecisionTreeClassifier	RandomOverSampler()	MinMaxScaler()	26	0.8587962962962963	0.917656526829610	0.7950000000000000	0.8400000000000000	0.7000000000000000	0.7943478296806960	3.1053620357340	0.7766690291292140
15	LogitClassifier	SimpleInputter(strategy='median')	RobustScaler(quantile_range=(25-75))	181	0.9049074074074070	0.948672246141130	0.8011111111111111	0.8355555555555556	0.7666666666666667	0.8086956521739130	6.607487923150790	0.82038383495145630
16	xgb	IterativeInputter(n_neighbors=3)	RobustScaler(quantile_range=(25-75))	151	0.8733037037037040	0.9124484115756830	0.8191666666666667	0.8800000000000000	0.7583333333333333	0.811584202808510	6.507759592455770	0.82038383495145630
17	RandomForestClassifier	RandomOverSampler()	RobustScaler(quantile_range=(25-75))	30	0.8937037037037040	0.9400576884025430	0.8180555555555556	0.9111111111111111	0.725	0.8376811594202900	5.6063613520013700	0.95145463106796120
18	KNNeighborsClassifier	IterativeInputter()	RobustScaler(quantile_range=(25-75))	2	0.8314814814814820	0.9076002640325430	0.7844444444444444	0.7688888888888889	0.8	0.8463378115942303	7.608597076584170	0.8466019147475730
19	DecisionTreeClassifier	RandomOverSampler()	RobustScaler(quantile_range=(25-75))	5	0.8759292929292930	0.9283541230717710	0.7963333333333333	0.8177777777777778	0.775	0.7797101449275360	6.807705460824790	0.81553398068292240

Table A.3: Results obtained for different scaling methods applied (best model for each estimator from the grid search is shown)

A Bibliography

- [1] Bma: Faqs about ethics. <https://www.bma.org.uk/advice-and-support/covid-19/ethics/covid-19-faqs-about-ethics>, 2020. (Online; accessed 25th-May-2020).
- [2] Django rest framework. <https://www.django-rest-framework.org/>, 2020. (Online; accessed 25th-May-2020).
- [3] Django tutorial. <https://docs.djangoproject.com/en/3.0/intro/tutorial01/>, 2020. (Online; accessed 25th-May-2020).
- [4] . P. E. Aamodt, A. Case-based reasoning: Foundational issues, methodological variations, and system approaches. <https://www.iiia.csic.es/~enric/papers/AICom.pdf>, 1994.
- [5] R. Agarwal. The 5 feature selection algorithms every data scientist should know. <https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3>, 2019. (Online; accessed 12-June-2020).
- [6] R. Alencar. Resampling strategies for imbalanced datasets. <https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets>, 2017. (Online; accessed 12-June-2020).
- [7] J. Brownlee. A gentle introduction to xgboost for applied machine learning. <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>, 2016. (Online; accessed 12-June-2020).
- [8] J. Brownlee. Logistic regression for machine learning. <https://machinelearningmastery.com/logistic-regression-for-machine-learning/>, 2016. (Online; accessed 12-June-2020).
- [9] J. Brownlee. Gradient boosting with scikit-learn, xgboost, lightgbm, and catboost. <https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-lightgbm-and-catboost/>, 2020. (Online; accessed 12-June-2020).
- [10] R.-H. Du, L.-R. Liang, C.-Q. Yang, W. Wang, T.-Z. Cao, M. Li, G.-Y. Guo, J. Du, C.-L. Zheng, Q. Zhu, M. Hu, X.-Y. Li, P. Peng, and H.-Z. Shi. Predictors of mortality for patients with covid-19 pneumonia caused by sars-cov-2: A prospective cohort study. *European Respiratory Journal*, 2020.
- [11] A. Dubey. Why feature selection? <https://medium.com/analytics-vidhya/why-feature-selection-144816f05ee8>, 2018. (Online; accessed 12-June-2020).
- [12] S. Gajare. Feature selection using random forest. <https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f>, 2019. (Online; accessed 12-June-2020).
- [13] W.-j. Guan, Z.-y. Ni, Y. Hu, W.-h. Liang, C.-q. Ou, J.-x. He, L. Liu, H. Shan, C.-l. Lei, D. S. Hui, B. Du, L.-j. Li, G. Zeng, K.-Y. Yuen, R.-c. Chen, C.-l. Tang, T. Wang, P.-y. Chen, J. Xiang, S.-y. Li, J.-l. Wang, Z.-j. Liang, Y.-x. Peng, L. Wei, Y. Liu, Y.-h. Hu, P. Peng, J.-m. Wang, J.-y. Liu, Z. Chen, G. Li, Z.-j. Zheng, S.-q. Qiu, J. Luo, C.-j. Ye, S.-y. Zhu, and N.-s. Zhong. Clinical characteristics of coronavirus disease 2019 in china. *New England Journal of Medicine*, 382(18):1708–1720, 2020.
- [14] P. Gupta. Decision trees in machine learning. <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>, 2017. (Online; accessed 12-June-2020).
- [15] O. Harrison. Machine learning basics with the k-nearest neighbors algorithm. <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d>, 2018. (Online; accessed 12-June-2020).
- [16] B. Hernandez. Data-driven web-based intelligent decision support system for infection management at point of care. <https://spiral.imperial.ac.uk/handle/10044/1/73000>, 10 2018.
- [17] B. Hernandez, P. Herrero, T. Rawson, L. Moore, E. Charani, A. Holmes, and P. Georgiou. Data-driven web-based intelligent decision support system for infection management at point-of-care: Case-based reasoning benefits and limitations. pages 119–127, 01 2017.
- [18] P. Joshi. What is bootstrap sampling in statistics and machine learning? <https://www.analyticsvidhya.com/blog/2020/02/what-is-bootstrap-sampling-in-statistics-and-machine-learning/>, 2020. (Online; accessed 12-June-2020).
- [19] Y. M. e. a. Li X, Xu S. Risk factors for severity and mortality in adult covid-19 inpatients in wuhan. 04 2020.

- [20] J. S. X. X. Martin Ester, Hans-Peter Kriegel. A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.
- [21] NHS. Covid-19 daily deaths. <https://www.england.nhs.uk/statistics/statistical-work-areas/covid-19-daily-deaths/>, 2020. (Online; accessed 23-May-2020).
- [22] M. Pantic. Introduction to machine learning and case based reasoning. <https://ibug.doc.ic.ac.uk/media/uploads/documents/courses/syllabus-CBR.pdf>, 2017.
- [23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [24] K. Rashmi and R. Gilad-Bachrach. Dart: Dropouts meet multiple additive regression trees. <http://proceedings.mlr.press/v38/korlakaivinayak15.pdf>, 2015.
- [25] H. B. e. a. Rawson TM, Moore LSP. A systematic review of clinical decision support systems for antimicrobial management: are we failing to investigate these interventions appropriately? pages 524–532, 08 2017.
- [26] R. J. G. B. C. M. Sander. Density-based clustering based on hierarchical density estimates. 2013.
- [27] E. L. S. K. Sudipto Mukherjee, Himanshu Asnani. Clustergan : Latent space clustering in generative adversarial networks. 2019.
- [28] R. T. Sutton, D. Pincock, D. C. Baumgart, D. C. Sadowski, R. N. Fedorak, and K. I. Kroeker. An overview of clinical decision support systems: benefits, risks, and strategies for success. 02 2020.
- [29] T. Yiu. Understanding random forest: How the algorithm works and why it is so effective. <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>, 2019. (Online; accessed 12-June-2020).