6BUIS020W – Business Information Systems Project Report

Breast Cancer Mortality and Survival Prediction using Machine Learning

Prepared by: Ben Gajadar

Supervisor: Mahmoud Aldraimli

This report is submitted in partial fulfilment of the requirements for the BSc (Hons) Business Information Systems

School of Computer Science & Engineering
University of Westminster

6th May 2025

Declaration

This report has been prepared based on my own work. Where other published and unpublished source materials have been used, these have been acknowledged in references.

Word Count: 7479

Student Name: Ben Gajadar

Date of Submission: 6th May 2025

Abstract

This project investigates the development of a Breast Cancer Mortality and Survival Prediction Model using machine learning. Breast Cancer remains the leading cause of death among women, and accurate predictions of mortality and survival outcomes can assist healthcare professionals in treatment planning, resource allocation, and early intervention. The goal of this study was to create a machine learning model capable of predicting both mortality status and survival time. A real-world dataset of 4,024 breast cancer patients, containing data on age, race, marital status and grade, was used to train the model. Various machine learning algorithms such as random forest and logistic regression, were used. The best performing model was XGB Regressor and Classifier, which achieved a classification accuracy of 89% and a regression Mean Absolute Error of 19.93. Key predictors of mortality included age, positive regional nodes, and grade. These results highlight the importance of machine learning in advancing healthcare. Future work will focus on the model's functionality, saving predictions to a database, utilising new data for model training, and integrating the model into healthcare institutions.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Mahmoud Aldraimli, for his support, guidance throughout this project. His insights and suggestions were detrimental in its successful implementation and completion. Furthermore, I extend my deepest appreciation to my manager, Soroush Khadem, and the entire team at AES for their support during the project. I also want to thank my sister, Anaya Gajadar, for her encouragement and patience, and to my family and friends for their continued support and understanding.

Table of Contents

List of Figures

List of Tables

Project Overview

1. The Purpose of the Project

1.1 Background of the Project Effort

This project is developed with the primary goal of using artificial intelligence to improve healthcare decision making. Breast Cancer is a worldwide health issue with almost 150 cases a day in the UK alone (Breast Cancer UK, 2024). Despite diagnosis and treatment improvements, the long-term prediction of patient survival and mortality is challenging to forecast.

Majority of healthcare professionals and institutions currently rely on clinical staging and statistical averages to estimate patient's survival time. However, these methods overlook trends and patterns in patient data. With the advancements in artificial intelligence and machine learning, this technology provides promising opportunities to impact the healthcare industry by offering more individualised treatment, earlier intervention and how patient's outcomes are predicted.

This project is intended for use by healthcare professionals, institutions, researchers and government agencies who require fast, reliable, and data-driven tool to aid in breast cancer patient's decision making.

1.2 Project Aim

This project aims to deliver a machine learning model with the capability to predict a patient's breast cancer mortality and survival rates based on past patient data. This model has the potential to enhance how healthcare professionals plan treatments and decide actions for their patients.

1.3 Project Goals

- To develop a predictive model which estimates breast cancer patients' mortality status and expected survival time based on real-world patient data

- To build an interactive user interface that allows users (e.g., healthcare professionals) to input patient details and receive predictions

- To implement visualisations to make prediction results understandable for healthcare professionals and stakeholders

- To ensure the model is functional, scalable and ethically developed, with attention to patient privacy and use of data.

1.4 Project Objectives

- The model should produce clear and comprehensive predictions which can be used by healthcare professionals to create or adjust a patient's treatment plans and understand an individual's diagnosis.

- Ensure the model uses machine learning to constantly learn and adapt in detecting patterns in patient's data which may be overlooked by doctors.

- Develop a fully working, trained machine learning model which has the ability to analyse patient data with high accuracy and precision

Ensure the model outputs are comprehensible and in an accessible way for all healthcare professionals of varying levels of expertise

2. The Scope of the work

2.1 The Current Situation

As it stands, healthcare professionals depend on clinical staging, pathology reports and statistical averages to determine breast cancer severity and estimate patient survival. These standardised methods lack consideration of deeper patterns hidden within patient's personal and medical data such as genetic profiles, confidential patient data, or treatment response variables. Additionally, the process is time consuming and heavily dependent on a healthcare professional's experiences, which may lead to inconsistent prediction. The integration of advanced machine learning technology could support medical personnel provide quicker and more accurate predictions improving workflow, patient workflow and resource allocation.

2.2 The Context of the Work

This project uses a machine learning model to provide data-driven insights, and it fits within the larger healthcare decision-making context. The final version of the machine learning model will integrate with external hospital systems and will be accessible to researchers or healthcare professionals. External factors influencing the model include medical databases or systems, ethical data governance and end-users such as oncologist, researchers and healthcare personnel. Identifying these factors is crucial for the model's clinical applicability and regulatory compliance

2.3 Work Partitioning

The model is designed to respond to key user events that support the prediction of patient mortality and survival outcomes. Each event defines how the model functions and interacts with users and patient data.

Event Name	**Input from adjacent systems**	**Output from adjacent systems**	**Use Case Summary**
Input patient data	User input patient data	Data sent to ML model	The user inputs patient's attributes (e.g. age, grade, stage. Etc) into the prediction model
Invalid Input	User inputs invalid data format or leaves incomplete	Message prompt will be shown to user	The Streamlit interface will notify user if any field is incomplete or has an incorrect value
Output Prediction	The processed result from prediction model	Prediction results and visualisation is displayed to user	The model returns the predicted survival time, mortality status and a visualisation
Evaluate Model Performance (Not user-facing)	Test ML Model	Model performance statistics (MAE, Classification report, accuracy, precision. Etc)	An internal ML Model evaluation to assess performance statistics.

Table 1: Work Partitioning

2.4 Competing Products

Deepserv is a learning-based survival analysis model which applies deep neural networks for personalised survival calculations based on patient's medical data (Jared lee Katzman, n.d.). It improved clinical decision-making by identifying intricate non-linear patterns which are often missed by traditional statistical methods (Katzman, 2018). The model allows user to input patient data and offers detailed visual graphs highlighting high-risk cases effectively. The model was built using python and integrates with cloud or hospital systems however it requires technical know-how to operate, making it inaccessible for other users.

OncoPredict is a machine learning oncology system which forecast patient survival outcomes and treatment responses. It generates predictions based on clinical trials, patient data and genomic profiles (HuangLabUMN, 2024). Additionally, it uses the OncoPredict.R package to help identify the most effective drugs for induvial patients (Maeser, 2021). The system provides detailed survival projections, personalised treatment reports and highlights high-risk cases helping healthcare professionals prioritise care. The user interface features interactive dashboards, real-time data entry and export options (CSV and PDF). The outputs include risk scores, bar charts and survival curves.

2.5 Gap Analysis

- DeepSurv is technically advanced but lack user accessibility due to complexity

- OncoPredict is a powerful system but has a dependency on complex data and a patient's bioinformation

- This project offers a user-friendly tool for all general healthcare staff focused on accessibility and comprehensive predictions

2.6 Key Deliverables

- A trained machine learning model which can predict breast cancer mortality status and survival time

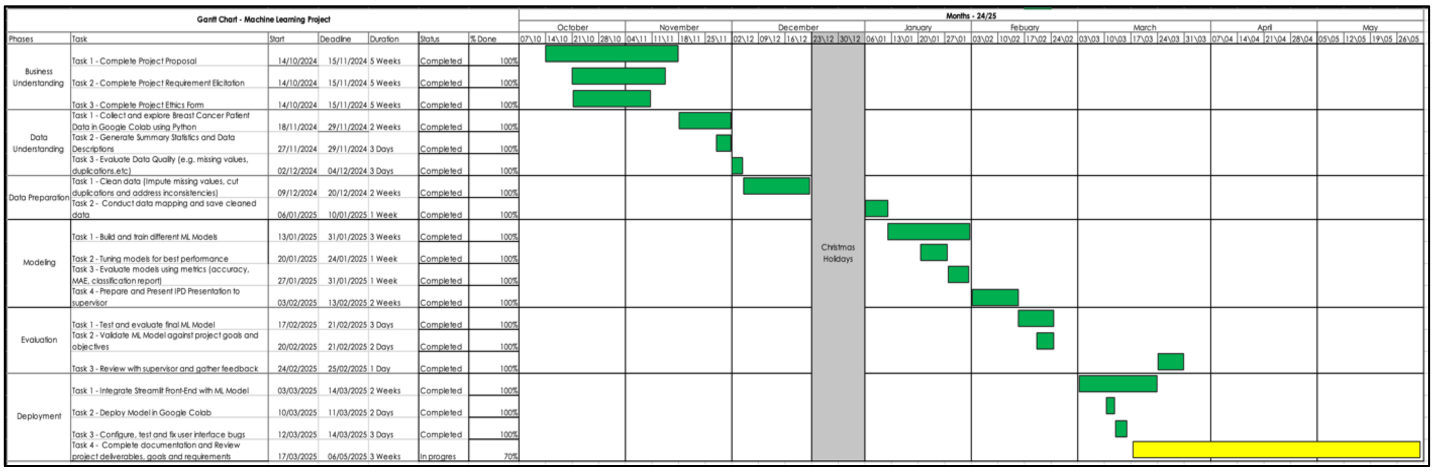
- An interactive prototype user interface for inputting patient data and receive prediction results

- A model performance evaluation report including accuracy, precision and MAE

- A data visualisation output alongside survival and mortality prediction

- Documentation covering the Project Overview, Requirements, Design and Implementation

2.7 Project Timeline



Phases	Task	Start	Deadline	Duration	Status	% Done
Business Understanding	Task 1 - Complete Project Proposal	14/10/2024	15/11/2024	5 Weeks	Completed	100%
	Task 2 - Complete Project Requirement Elicitation	14/10/2024	15/11/2024	5 Weeks	Completed	100%
	Task 3 - Complete Project Ethics Form	14/10/2024	15/11/2024	5 Weeks	Completed	100%
Data Understanding	Task 1 - Collect and explore Breast Cancer Patient Data in Google Colab using Python	18/11/2024	29/11/2024	2 Weeks	Completed	100%
	Task 2 - Generate Summary Statistics and Data Descriptions	27/11/2024	29/11/2024	3 Days	Completed	100%
	Task 3 - Evaluate Data Quality (e.g. missing values, duplications.etc)	02/12/2024	04/12/2024	3 Days	Completed	100%
Data Preparation	Task 1 - Clean data (impute missing values, cut duplications and address inconsistencies)	09/12/2024	20/12/2024	2 Weeks	Completed	100%
	Task 2 - Conduct data mapping and save cleaned data	06/01/2025	10/01/2025	1 Week	Completed	100%
Modeling	Task 1 - Build and train different ML Models	13/01/2025	31/01/2025	3 Weeks	Completed	100%
	Task 2 - Tuning models for best performance	20/01/2025	24/01/2025	1 Week	Completed	100%
	Task 3 - Evaluate models using metrics (accuracy, MAE, classification report)	27/01/2025	31/01/2025	1 Week	Completed	100%
	Task 4 - Prepare and Present IPD Presentation to supervisor	03/02/2025	13/02/2025	2 Weeks	Completed	100%
Evaluation	Task 1 - Test and evaluate final ML Model	17/02/2025	21/02/2025	3 Days	Completed	100%
	Task 2 - Validate ML Model against project goals and objectives	20/02/2025	21/02/2025	2 Days	Completed	100%
	Task 3 - Review with supervisor and gather feedback	24/02/2025	25/02/2025	1 Day	Completed	100%
Deployment	Task 1 - Integrate Streamlit Front-End with ML Model	03/03/2025	14/03/2025	2 Weeks	Completed	100%
	Task 2 - Deploy Model in Google Colab	10/03/2025	11/03/2025	2 Days	Completed	100%
	Task 3 - Configure, test and fix user interface bugs	12/03/2025	14/03/2025	3 Days	Completed	100%
	Task 4 - Complete documentation and Review project deliverables, goals and requirements	17/03/2025	06/05/2025	3 Weeks	In progres	70%

Figure 1: Project Timeline

3. Product Scenarios

3.1 Individual Product Scenarios

Scenario 1: Dr. Aldraimli, an oncologist, is preparing to consult a patient recently diagnosed with breast cancer. He opens the prediction system and inputs patient data including age, cancer grade, stages and the number of positive regional nodes. The ML model returns a clear prediction of the patient's mortality status, survival time and a gauge visualisation. This insight helps Dr. Aldraimli to guide patient's treatment plan and discuss appropriate options.

Scenario 2: Dr. Fang, a university researcher specialising in oncology data science, is conducting research on survival trends among breast cancer patients. Using the ML Model, she inputs anonymous patient data and generate estimated survival times. She identifies a correlation between marital status, positive regional nodes, and survival outcomes. The model's outputs and visualisations were published as part of her study as it supported her findings.

Scenario 3: Alex, a public health analyst at the UK Department of Health, is reviewing the UK's breast cancer survival statistics. He inputs various patient data from different boroughs and generates mortality and survival time predictions. This will aid in policy planning and identifying high-risk demographics or areas with poorer outcomes. These insights support policy planning, awareness campaigns, and funding decisions.

4. Stakeholders

4.1 The Client

The primary client for this project is my supervisor, Dr. Mahmoud Aldraimli, who oversees the project, provides guidance, and assesses my work to ensure it is structured, ethical and innovative.

The intended real-world clients for this project include healthcare institutions, such as hospitals, cancer research centres, and government health organisations. These stakeholders would benefit from a machine learning prediction tool that supports clinical decision-making, improve resource allocation, and enhance patient communication. By investing in and implementing this tool, they can improve treatment planning and boost patient outcomes.

4.2 The Customer

The primary customers are healthcare professionals (regardless of experience level), including oncologists, nurses, general practitioners, and public health personnel who require a data-driven tool to support breast cancer diagnosis, patient treatment planning, and survival analysis. Academic researchers could use the system for oncology studies or research. The tool is designed to be user-friendly, making it easy for all users to interpret predictions from patient data.

4.3 Hands-On Users of the Product

Username / Category	**User Role**	**Healthcare or Medical Expertise**	**Technological Experience**	**Other User Characteristics**
Oncologists & Doctors	Inputs patient information into the model and uses the	Master	Journeyman	Highly educated, responsible, aged 30-65,

	prediction for treatment planning.			driven towards improving patient care, strategic and problem-solving thinkers
Nurse & General Practitioners	Uses model predictions to monitor and support patient's diagnosis	Journeyman	Novice to Journeyman	Strong patient-facing roles, under 35, limited time for complex interfaces and varying levels of technical expertise and medical experience.
Medical Students / Trainees	Uses the model for educational purposes and simulations	Novice	Journeyman	Learning focused, adaptive, aged 20-30, low medical experience and require simplicity.
Academic Researchers	Uses model predictions to analyse survival trends, test hypotheses and support research.	Master	Master	Highly educated, Technically skilled, aged 40 – 80, research-driven, and motivated by accuracy, consistency and publication standards.
Public Health Analysts & Government Personnel	Uses model predictions to identify high risk population and support national health planning and policy making.	Journeyman	Journeyman	Investigative, Data-driven, outcome-driven, high computer literacy and technological experience, aged 25-55

Table 2: Hands-On Users of the Product

5. Constraints

5.1 Implementation Environment of the Current System

My model will be implemented in a digital healthcare environment, such as cloud-based platforms, hospital IT systems, or academic research labs. The model's uses a Streamlit-based interface that requires internet connectivity and a web enabled device. The implementation environment consists of non-human systems such as electronic health record (EHR) platforms, cloud storage or local servers, and secure networks. The model must be compatibility with standard browsers, responsiveness across devices, and integrate into existing healthcare workflows. Additionally, my model must adhere to data protection regulations or policies while addressing privacy concerns.

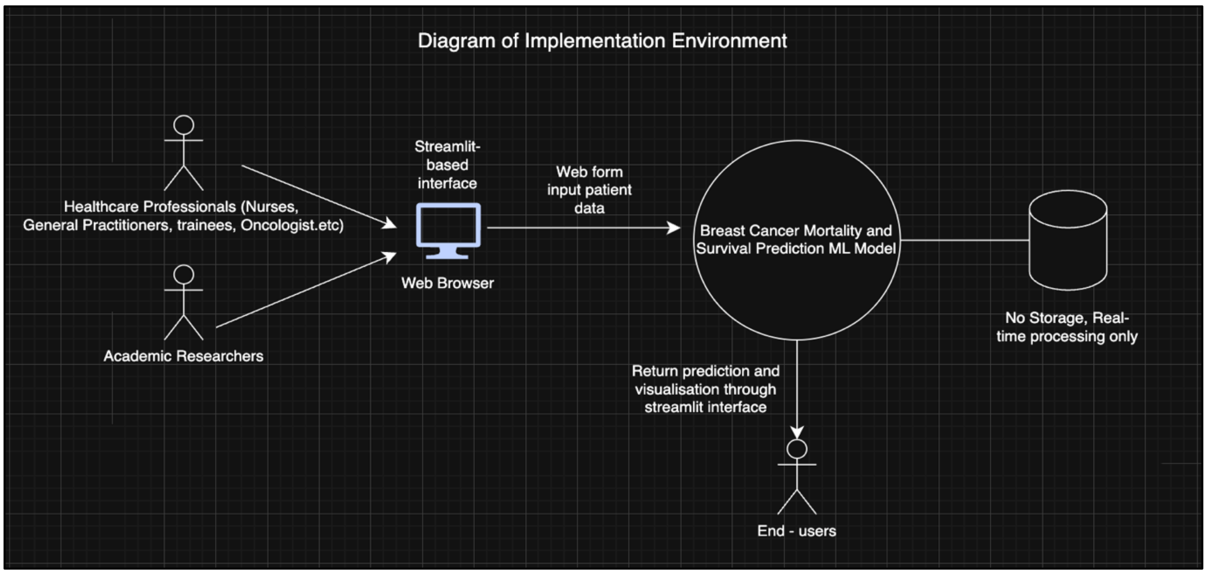


Diagram of Implementation Environment
Healthcare Professionals (Nurses, General Practitioners, trainees, Oncologist.etc)
Academic Researchers
Streamlit-based interface
Web Browser
Web form input patient data
Breast Cancer Mortality and Survival Prediction ML Model
No Storage, Real-time processing only
Return prediction and visualisation through streamlit interface
End - users

Figure 2: Diagram of Implementation Environment

5.2 Partner or Collaborative Applications

My model does depend on external tools and services which influence its design and functionality however these partner applications are not part of the core model but are a necessity for prediction, data analysis, model deployment and user interaction. Key collaborative tools are :

Partner Applications	Type	Description	Role
Pandas & NumPy	Libraries	A python library used for data preparation and transformation	Used for data exploration, manipulation and preparation before training model
Scikit-learn	Library	A python library used for machine learning	Used to develop and train the machine learning model

Matplotlib / Seaborn	Libraries	Python libraries used for generating visualisation	Used to display visualisation like prediction insights and interpretability
Streamlit	Web App Framework	An open-source python framework used for developing frontend interface	Used for user to interact with the model and receive prediction
Google Colab	Integrated Development Environment (IDE)	Used to execute implementation including data exploration, preparation, model development, training and evaluation	Used to execute the entire implementation
Healthcare EHR systems (Future Integration)	Internal hospital systems	Hospital database, network and EMR systems	Used for secure automated data input/output and saving prediction to a database

Table 3: Partner Applications

Using partner applications imposes the following design constraints :

- Dependency Management : It is essential for Scikit-learn, Pandas, and Streamlit to maintain compatibility across all deployment settings

- Performance Consideration : Current development with Streamlit is suitable however for future integration with large scale systems like a hospital database, API integration may have to be considered.

- Security and Privacy Concerns : At the moment, no data is stored however for future integration with electronic health records (EHRs), data protection and encryption will be required.

- Deployment Limitation : Currently, Google Colab is suitable for development however is not reliable for future long-term deployment

Requirements

6. Product Use Cases

6.1 Use Case Diagrams

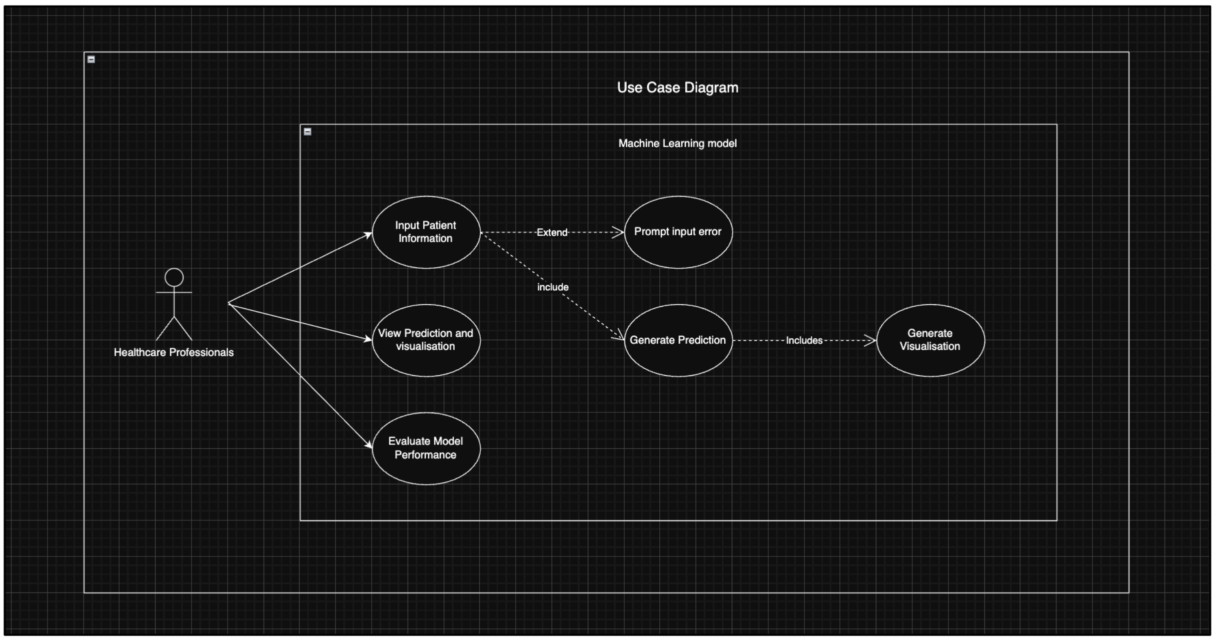


Figure 3: Functional Model

6.2 Individual Product Use Cases

Use Case Name :	Input Patient Information
Participating Actors :	Initiated by healthcare professionals
Flow of events :	1. Healthcare Professional accesses the Streamlit interface 2. Healthcare Professional enters patient data 3. Interface input form validates inputs and prompts input error if present 4. Upon valid input, the machine learning model generates prediction and visualisation 5. Prediction is displayed
Entry Conditions :	• The user interface is publicly accessible without the need of user authentication
Exit Conditions :	• Input is successfully submitted • Prediction and visualisation are generated and displayed to user
Quality Requirements :	• Interface must be user friendly and suitable for all levels of expertise in healthcare staff

	• The Machine Learning Model must respond within 1 minute • The Machine Learning Model must achieve a minimum of 85% accuracy rate • Inputs should be validated instantly

Table 4: Individual Product Use Cases

7. Functional & Non-functional Requirements

7.1 Functional | Non-functional Requirements

Requirement #	User Requirement	Priority	(Non) Functional Area	Requirements Owner
R01	Model must provide trustworthy and reliable predictions with an accuracy score more than or equal to 85%	High	Functional	Healthcare Institutions, Healthcare Professional
R02	Results must be delivered with 1 minute of data input to support real-time decision-making	Medium	Functional	Healthcare Institutions
R03	User interface must display clear prediction results and visualisations for users	High	Functional	Healthcare Professional, Patients
R04	Progress updates must be regularly communicated by either email or in-person meetings to stakeholders	Low	Non - Functional	Supervisor
R05	Project must adhere to the project timeline	Low	Non - Functional	Supervisor
R06	User interface must display error messages and prompts clearly	Medium	Non - Functional	Healthcare Institutions, Healthcare Professional
R07	Codebase must be well-documented and explained with comments for future development and maintenance	High	Non - Functional	Supervisor, Healthcare Institutions
R08	The system must not collect, store, or process any personally identifiable information (PII)	High	Non - Functional	Healthcare Institutions
R09	Only anonymised data should be processed by the ML Model	High	Functional	Healthcare Institutions
R10	Model must reject invalid or incomplete input data	High	Functional	Healthcare Professionals
R11	Model must display a disclaimer alerting users that predictions are experimental	High	Non- Functional	Healthcare Institutions, Healthcare Professional, Supervisor
R12	The system should be easily navigable with standard keyboard and mouse	Low	Non- Functional	Healthcare Professional

R13	The interface must have a restart session button to reset input fields	Medium	Functional	Healthcare Professional

Table 5: Requirements Table

8. Data Requirements

8.1 Subject Matter & Domain Model

My model is designed to assist healthcare professionals in predicting a breast cancer's patient survival time and mortality status using a machine learning model. The model provides predictions based on input data such as grade, progesterone Status, N Stage, tumour size and other patient biomedical data. The model was trained using a publicly available dataset called the SEER Breast Cancer Dataset, obtained by the National Cancer institution (TENG, 2019).

8.2 Essential Data Entities

- **Patient Record**: Input data is anonymised
- **Prediction Output**: A patient's mortality status and survival time
- **Visualisation Output**: Graph, chart or figure showing a visualisation representation of prediction
- **Model Performance Report**: Accuracy, Precision, MAE and Classification report

8.3 Data Handling Considerations

- **Patient's Anonymity**: No personally identifiable information (PII) is stored or collected
- **Data Validation**: The interface alerts user if incorrect value is entered ensuring accuracy and correctness
- **Data Storage**: Input and output data is not stored and is cleared after the session ends
- **User – friendly Interface**: The interface should be easy to use and clear for health professionals of all varying levels of expertise to use and understand.

8.4 Safety, Ethical & Critical Requirements

Category	Requirement
Ethical Standards	• Must comply with GDPR and the Uk Data Protection Act 2018

	• No personally identifiable information must be stored or collected • Only anonymised data can be processed by the model
Clinical Safety	• The Model should have a disclaimer like "This model is experimental and not a substitute for medical advice. Verify all results with a qualified professional."
Accuracy Expectation	• The ML Model must have an accuracy score of at least 85%, with planned improvement effort through retraining to avoid false predictions
Operation Safety	• The ML Model should respond within 1 minute of form submission
Transparency & Interpretability	• The ML Model should produce understandable visualisation by medical staff
Data Storage Policy	• No long-term storage or export functionality and data should be cleared after session is terminated
User Access Control	• No authentication in current implementation, but must be considered for future iteration and implementation with hospital databases for security and data protection purposes
Ethical Bias	• The model must be tested for bias with focus on age, race, and marital status • Data imbalances must be addressed • Any data bias must be documented, and disclaimers must be displayed to users

Table 6: Safety, Ethical & Critical Requirements

9. Security Requirements

9.1 Access Requirements

- The Model should have user authentication, such as NHS or OAuth, to restrict access to sensitive patient or hospital data (**Future Iterations**)

- Role-based access control (RBAC) should be implemented to distinguish between different users and restrict access to specific users (e.g., Only Oncologist can edit patient treatment whereas nurses can view only) (**Future Iterations**)

9.2 Integrity Requirements

- The interface should validate all input data to ensure accuracy and reduce chances of system errors

- The interface should reject form submission if input is either invalid or incomplete

- The ML Model or generated outputs should not be subject to modification during execution

- The system should not allow any unauthorized users to write or overwrite the models programming or configurations (**Future Iterations**)

9.3 Privacy Requirements

- The system must process only anonymised data and must not store, share or process any personally identifiable information (PII)

- No data (input or output) will be collected or stored on any database after the session ends

9.4 Accessibility Requirements

- The interface should be clear and navigable by all healthcare professionals regardless of level of digital or technical literacy

- The interface should use design elements that improve readability, such as typography, colour contrast, and appropriate spacing

- All interactive elements such as buttons or fields should be clear and functional

- The interface must be fully navigable using standard keyboard and mouse

10. Legal Requirements

10.1 Compliance Requirements

- **General Data Protection Regulation (GDPR) & UK Data Protection Act 2018**: In adherence to these regulations, no personal or identifiable patient data shall be stored, shared or processed by the machine learning model. All input data must be anonymised before processed.

- **Copyright & Dataset Usage**: The model will be trained using the publicly available SEER Breast Cancer dataset (TENG, 2019), which is licensed for academic and research purposes. The project must ensure the dataset is referenced appropriately.

- **Software Licensing**: All frameworks and libraries used in the implementation - included but not limited to Pandas, Matplotlib and Streamlit - are either open source or licensed for academic use.

- **Medical Device Regulations (MDR)**: As the model is not classified as a certified medical device, the interface must display a disclaimer stating the model is experimental and should not be used as a substitute for professional medical judgement or advice.

Compliance with these legal frameworks ensures the protection of patient data, fosters user trust, protects developers from legal liability, and helps prevent the misuse of predictions in clinical decision-making.

Design

11. Proposed Software Architecture

11.1 Object Model

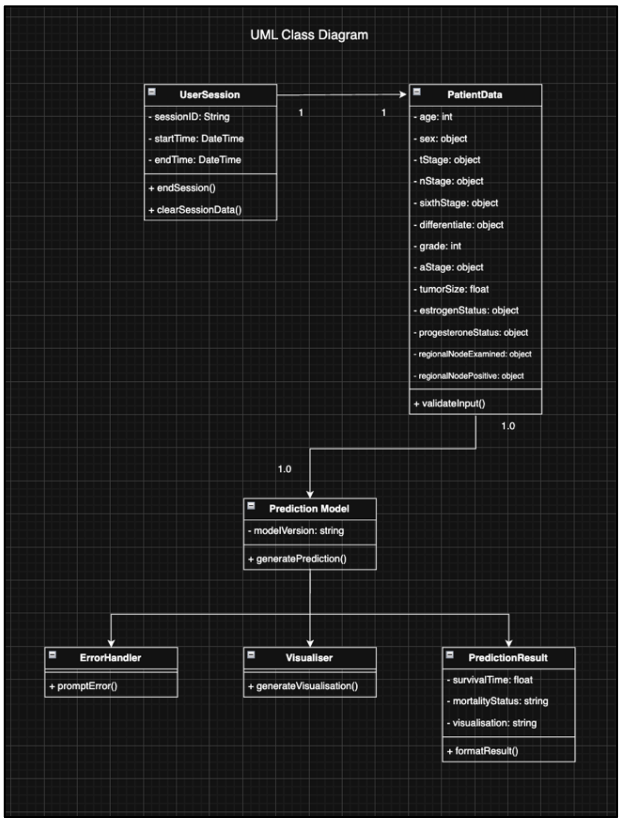


Figure 4: Object Model

This class diagram represents my breast cancer mortality and survival prediction model. The current implementation does not use a database and supports only one patient at a time. Once a session begins, the user inputs patient data, which is sent to the prediction model to generate and display results.

The sequence diagram illustrates the flow of interactions between the user and the prediction model. It shows how the user inputs patient data and receives a prediction, while also providing insight on the technological steps in the process.

11.2 Dynamic Model

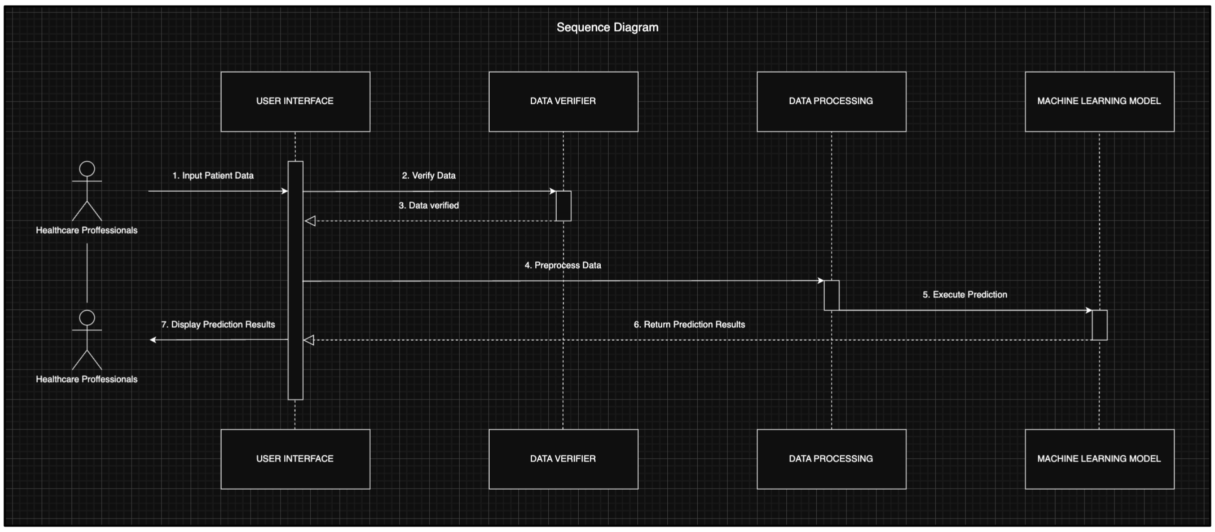


Figure 5: Dynamic Model

11.3 System Model and Decomposition

The system model for my breast cancer survival and mortality prediction model includes a functional model, object model and dynamic model. Each diagram addresses different aspects of the model's design and architecture.

- Functional Model (Use Case Diagram): This diagram highlights the models core functions from a user's perspective, such as entering patient data, error messages, and viewing prediction results.

- Object Model (Class Diagram): This diagram defines key classes and their relationships. All classes were derived from the use cases ensuring structural support for all functionalities.

- Dynamic Model (Sequence Diagram): This diagram illustrates the real-time interaction between the user and the model during a single session.

This use case-driven approach ensures that:

- Classes can be tracked back to functional requirements

- Use cases define what the model must achieve

- Sequence diagrams show the user's actions alongside the systems

- Class diagram shows structural design while the sequence diagram emphasises user and system interactions

With all models combined to create a system model, this ensures the system design is clear, consistent and traceable.

11.4 Persistent Data Management

My current implementation of the prediction model uses ephemeral (non-persistent) data. All processed data, including inputs and model outputs, is temporary and is cleared as soon as session is over. This current design complies with legal and ethical standards for handling sensitive healthcare data under the UK Data Protection Act of 2018 and GDPR.

For future iterations, persistent data management will be required for :

- Audit logging to ensure adherence to healthcare regulations

- Model retraining using real-world data to improve predictions

- Saving past prediction histories for clinical reference (Subject to hospital approval)

Potential persistent data storage technologies include:

- Relational Databases such as MySQL or PostgreSQL for organised patient and prediction records
- Cloud Storage Solutions such as AWS or Google Cloud for model logs and backups
- Secured Distributed File Systems such as Hassop HDFS for large-scale data analysis and insights

Only anonymised data approved by ethics boards will be stored. All persistent data will be encrypted and protected by robust access controls and policies.

11.5 Access Control and Security

Access control is crucial for ensuring that only authorised personnel can access or interact with the model's functionalities. My current implementation does not require authentication as it is open for public demo use. However, for future iterations, strict access controls measures will be integrated.

Key security measures include:

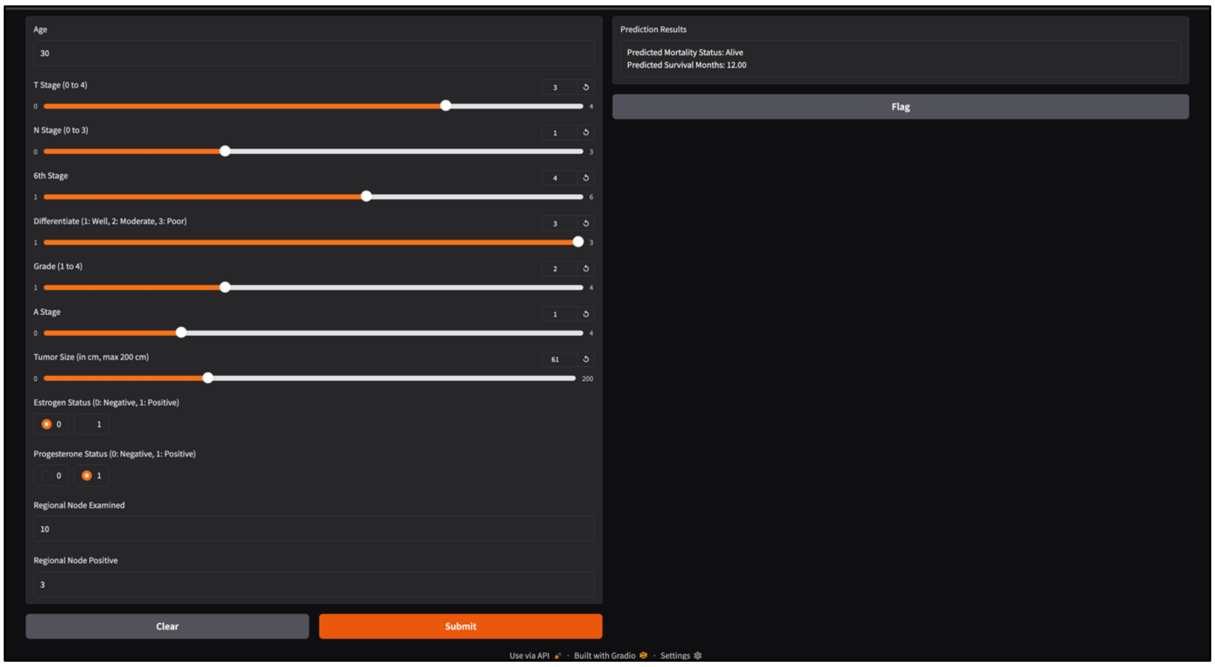
- **Role-Based Access (RBAC)**: Users will be assigned different permissions based on their role (e.g. Oncologists, nurses, researchers. Etc). For example, oncologists can access the prediction functionalities, while a nurse may be limited to viewing results only.
- **Authentication Mechanisms**: Future iterations of my implementation will need secure authentication integration such as OAuth 2.0 or NHS Single Sign-On (SSO). This ensures only authorised healthcare professionals can access features related to patient data or predictions.
- **Data Transmission**: All data communication over the interface should use HTTPS with TLS encryption, preventing cyber-attacks such as man-in-the-middle (MITM) attacks or the interception of sensitive information (RAPID7, n.d.).
- **Session Management**: Each session will have a timeout feature that automatically deletes input and output data after being inactive for a set period of time. No cookies or tokens will continue beyond the session.
- **Security Standards Compliance**: Future iterations will adhere to NHS Digital Guidelines, OWASP, and GDPR practices. All future deployments will subject to penetration testing and vulnerability scans.

These measures are crucial to minimise security risks, maintain user trust, and ensuring scalability and suitability for real-world clinical use.

12. User Interface

12.1 Interface Development using Gradio

To develop my initial prototype interface, I chose Gradio for its simplicity and quick deployment. I initially decided to use sliders and text input fields to quickly test model prediction capabilities with minimal programming. However, Gradio frequently crashed during development when implementing visualisations, and the average response time was 2-3 minutes, which did not meet performance requirements. These issues alongside the limited flexibility for customisation and scalability, led me to explore more suitable and advanced user interface platforms.

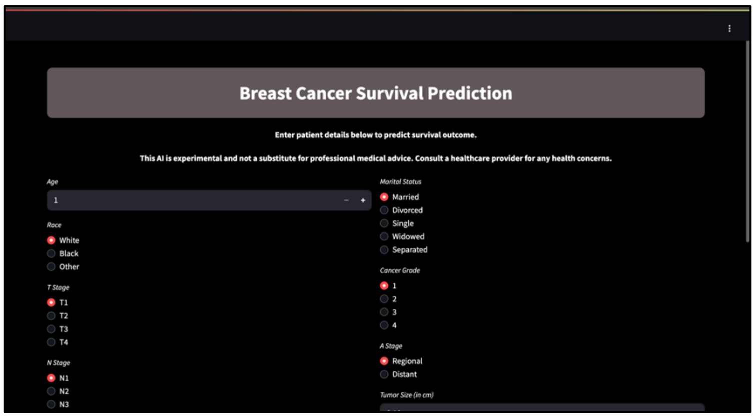

Age: 30
T Stage (0 to 4): 3
N Stage (0 to 3): 1
6th Stage: 4
Differentiate (1: Well, 2: Moderate, 3: Poor): 3
Grade (1 to 4): 2
A Stage: 1
Tumor Size (in cm, max 200 cm): 61
Estrogen Status (0: Negative, 1: Positive): [✓ 0] 1
Progesterone Status (0: Negative, 1: Positive): 0 [✓ 1]
Regional Node Examined: 10
Regional Node Positive: 3

Prediction Results
Predicted Mortality Status: Alive
Predicted Survival Months: 12.00

Figure 6: Gradio User Interface

12.2 Interface Development using Streamlit

Once I discussed the problems, I was having with Gradio with my supervisor, I was advised to try implementing Streamlit. This proved to be a better solution for my machine learning model, and I used it in the final implementation. Streamlit is more flexible in terms of layout and visuals compared to Gradio.


Breast Cancer Survival Prediction

Enter patient details below to predict survival outcome.

This AI is experimental and not a substitute for professional medical advice. Consult a healthcare provider for any health concerns.

Age: 1
Race: [✓ White] Black Other
T Stage: [✓ T1] T2 T3 T4
N Stage: [✓ N1] N2 N3
Marital Status: [✓ Married] Divorced Single Widowed Separated
Cancer Grade: [✓ 1] 2 3 4
A Stage: [✓ Regional] Distant
Tumor Size (in cm)

Figure 7: Streamlit User Interface - Form

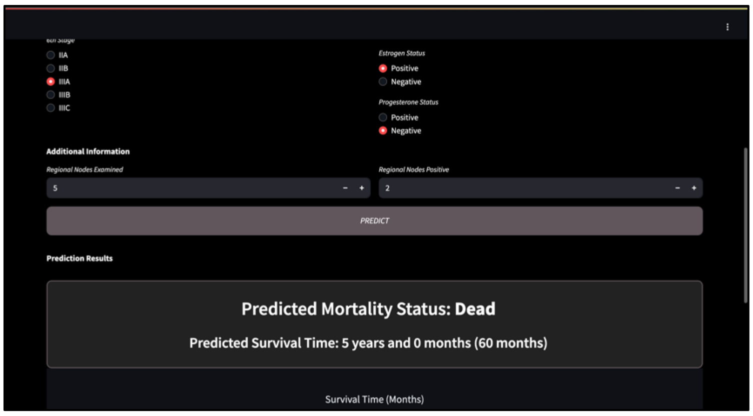
I made numerous adjustments to improve the interface design:

- Replaced sliders with buttons for better precision and control

- Changed colour scheme to look more professional and presentable

- Changed layout to columns for improved alignment, spacing and readability

- Implemented visualisation (Gauge) to support the model's interpretability and engagement



6th Stage
- IIA
- IIB
- [✓ IIIA]
- IIIB
- IIIC

Estrogen Status
- [✓ Positive]
- Negative

Progesterone Status
- Positive
- [✓ Negative]

Additional Information

Regional Nodes Examined: 5

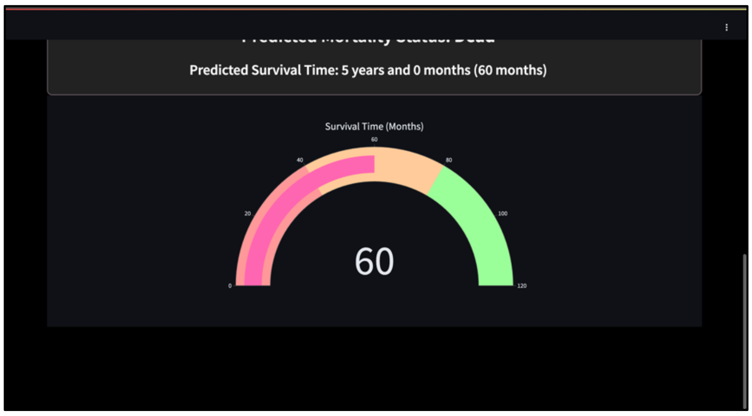
Regional Nodes Positive: 2

Prediction Results

Predicted Mortality Status: **Dead**

Predicted Survival Time: 5 years and 0 months (60 months)

Figure 8: Streamlit User Interface - Prediction



Predicted Survival Time: 5 years and 0 months (60 months)

Figure 9: Streamlit User Interface - Visualisation

While Streamlit was much more complex to set up and slightly more time-consuming than Gradio, it offers better control over the user interface. This allowed me to apply design principles more effectively and deliver a more refined and user-friendly interface that best showcases the capabilities of my model.

12.3 Interface Design Principles

Key User interface design decisions focused on applying core design principles:

- User Control: The interface allows users to interact with the model by manually selecting buttons, filling fields, and viewing predictions and visualisations. This ensures users have full control of inputs and of their interactions with the model.

- Consistency: The interface is uniform, readable and uses font enhancements, ensuring easy navigation for users.

- Feedback: The interface gives clear, immediate messages to the user regarding input form errors and responses to submitted forms by displaying predictions and visualisations.

- Accessibility & Clarity: The user interface is structured in columns, utilising all available screen space with readable text. It has a sleek design and is understandable to all healthcare professionals, regardless of their level of expertise.

- Hierarchy: The key elements of my interface, such as inputs, outputs and buttons, are highlighted by their placements

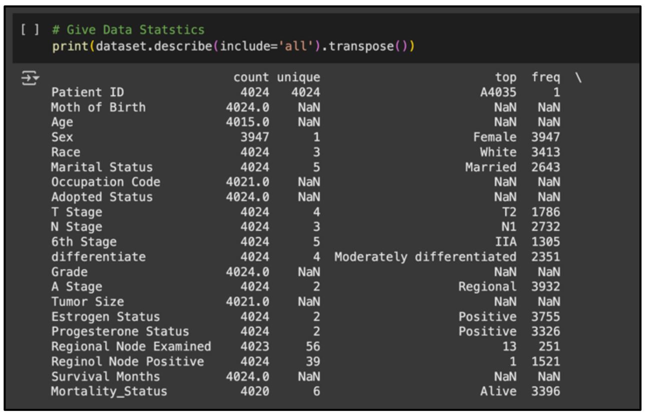
These key design principles are necessary for maintain a functional, user friendly and structured interface regardless of changes in interface python libraries.

Implementation

13. Coding

13.1 Understanding Dataset

To start development, I needed to understand my dataset. The SEER Breast Cancer Dataset I used contained 4024 patient records and was diverse across different demographics. I loaded the dataset and explored its structure using pandas library. The dataset consisted of 21 columns, including variables such as Age, Sex, Race, Marital Status, T Stage, Tumor size, and more. The primary variables for classification were Mortality_Status, and for regression, Survival Months.



```
[ ] # Give Data Statstics
    print(dataset.describe(include='all').transpose())
```

```
                          count unique                        top  freq  \
Patient ID                 4024   4024                      A4035     1
Moth of Birth            4024.0    NaN                        NaN   NaN
Age                      4015.0    NaN                        NaN   NaN
Sex                        3947      1                     Female  3947
Race                       4024      3                      White  3413
Marital Status             4024      5                    Married  2643
Occupation Code          4021.0    NaN                        NaN   NaN
Adopted Status           4024.0    NaN                        NaN   NaN
T Stage                    4024      4                         T2  1786
N Stage                    4024      3                         N1  2732
6th Stage                  4024      5                        IIA  1305
differentiate              4024      4  Moderately differentiated  2351
Grade                    4024.0    NaN                        NaN   NaN
A Stage                    4024      2                   Regional  3932
Tumor Size               4021.0    NaN                        NaN   NaN
Estrogen Status            4024      2                   Positive  3755
Progesterone Status        4024      2                   Positive  3326
Regional Node Examined     4023     56                         13   251
Reginol Node Positive      4024     39                          1  1521
Survival Months          4024.0    NaN                        NaN   NaN
Mortality_Status           4020      6                      Alive  3396
```

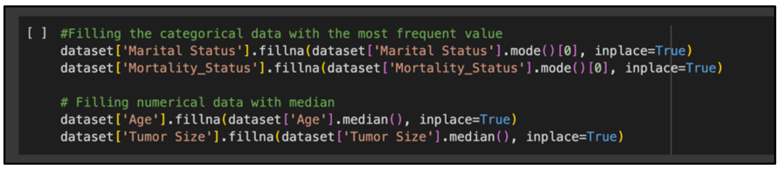
Figure 10: Implementation - Dataset Statistics

13.2 Cleaning & Preparing Data

Before using data for machine learning training, it is important to ensure the data is cleaned and prepared to ensure accuracy, reliability, consistency, and to reduce the chance of errors. This step usually consists of:

- Dropping non-contributory columns: From the initial 21 columns, 6 columns were dropped based on the belief that they would not affect predictions:
    - Month of birth – age was present, making column redundant
    - Differentiate – grade was present, suggesting this column was irrelevant
    - Occupation code – It was not believed that occupation will affect prediction
    - Adopted status – Adoption status is not considered a relevant factor for prediction
    - patient ID - this was an irrelevant column for predictions
    - Sex – As only biological women can have breast cancer; this column was deemed irrelevant

- Handling Missing values: Numerical columns such Age and Tumor size contained missing values, which were imputed using median (the middle value in the dataset). Categorical columns like marital Status and Mortality Status were filled with the mode (most frequent value).

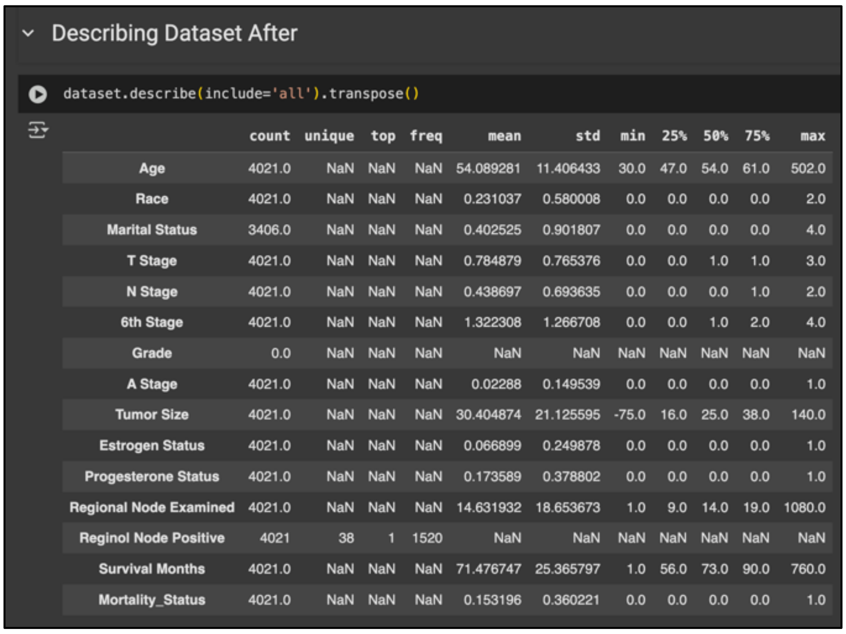


```
[ ] #Filling the categorical data with the most frequent value
    dataset['Marital Status'].fillna(dataset['Marital Status'].mode()[0], inplace=True)
    dataset['Mortality_Status'].fillna(dataset['Mortality_Status'].mode()[0], inplace=True)

    # Filling numerical data with median
    dataset['Age'].fillna(dataset['Age'].median(), inplace=True)
    dataset['Tumor Size'].fillna(dataset['Tumor Size'].median(), inplace=True)
```

Figure 11: Implementation - Handling Missing Values in Dataset

- Handling Duplicates: Only 1 duplication was found and dropped.
- Label Encoding / Mapping: To ensure the model compatibility, instead of using a label encoder, manual mapping was conducted for 7 categorical variables.

After cleaning, the processed dataset consisted of 4021 records with fully numeric features, which was exported and saved as "cleaned_breast_cancer_data.csv".

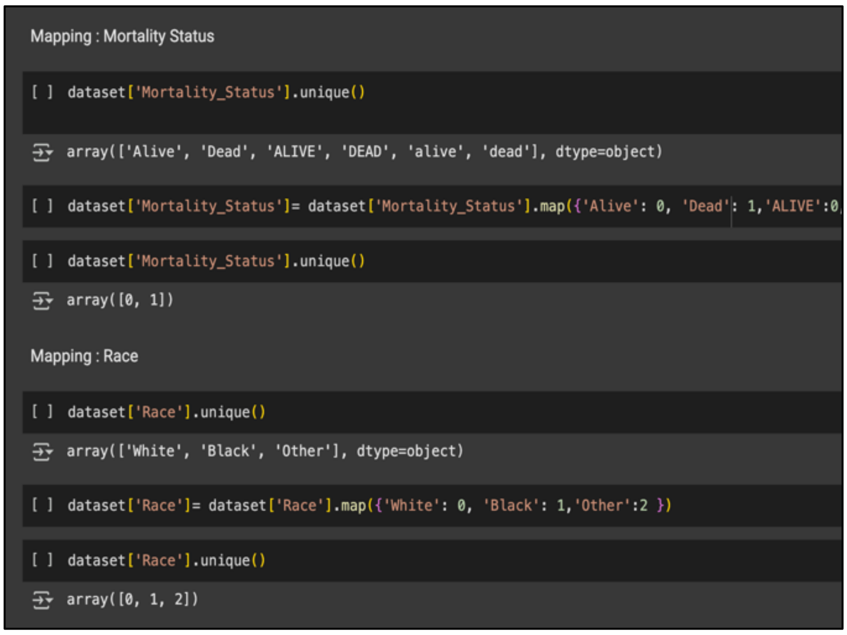


Describing Dataset After

```
dataset.describe(include='all').transpose()
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Age	4021.0	NaN	NaN	NaN	54.089281	11.406433	30.0	47.0	54.0	61.0	502.0
Race	4021.0	NaN	NaN	NaN	0.231037	0.580008	0.0	0.0	0.0	0.0	2.0
Marital Status	3406.0	NaN	NaN	NaN	0.402525	0.901807	0.0	0.0	0.0	0.0	4.0
T Stage	4021.0	NaN	NaN	NaN	0.784879	0.765376	0.0	0.0	1.0	1.0	3.0
N Stage	4021.0	NaN	NaN	NaN	0.438697	0.693635	0.0	0.0	0.0	1.0	2.0
6th Stage	4021.0	NaN	NaN	NaN	1.322308	1.266708	0.0	0.0	1.0	2.0	4.0
Grade	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
A Stage	4021.0	NaN	NaN	NaN	0.02288	0.149539	0.0	0.0	0.0	0.0	1.0
Tumor Size	4021.0	NaN	NaN	NaN	30.404874	21.125595	-75.0	16.0	25.0	38.0	140.0
Estrogen Status	4021.0	NaN	NaN	NaN	0.066899	0.249878	0.0	0.0	0.0	0.0	1.0
Progesterone Status	4021.0	NaN	NaN	NaN	0.173589	0.378802	0.0	0.0	0.0	0.0	1.0
Regional Node Examined	4021.0	NaN	NaN	NaN	14.631932	18.653673	1.0	9.0	14.0	19.0	1080.0
Reginol Node Positive	4021	38	1	1520	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Survival Months	4021.0	NaN	NaN	NaN	71.476747	25.365797	1.0	56.0	73.0	90.0	760.0
Mortality_Status	4021.0	NaN	NaN	NaN	0.153196	0.360221	0.0	0.0	0.0	0.0	1.0

Figure 12: Implementation - Dataset after Cleaning



Mapping : Mortality Status

```
[ ] dataset['Mortality_Status'].unique()

array(['Alive', 'Dead', 'ALIVE', 'DEAD', 'alive', 'dead'], dtype=object)

[ ] dataset['Mortality_Status']= dataset['Mortality_Status'].map({'Alive': 0, 'Dead': 1,'ALIVE':0,

[ ] dataset['Mortality_Status'].unique()
array([0, 1])
```

Mapping : Race

```
[ ] dataset['Race'].unique()
array(['White', 'Black', 'Other'], dtype=object)

[ ] dataset['Race']= dataset['Race'].map({'White': 0, 'Black': 1,'Other':2 })

[ ] dataset['Race'].unique()
array([0, 1, 2])
```

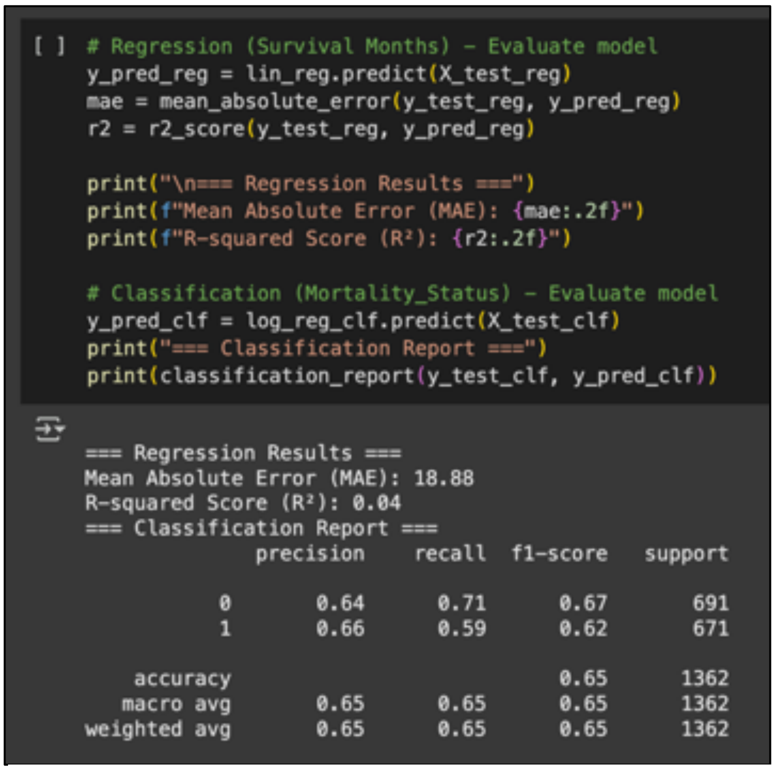
Figure 13: Implementation - Mapping Categorical Variables

13.3 Creating & Training Model

Given that no single model is the best model in machine learning, I explored and evaluated various models to pinpoint the most suitable model for my specific problem. The models were tasked with predicting Mortality Status (Classification) and Survival months (Regression). Below is a table of my findings and explanations for each model I created, trained and evaluated:

Model	Findings	Evidence
Logistic Regression & Linear Regression	This was the first model I created and trained. I used a Logistic Regression model for classification, applying SMOTE to address class imbalances before training to ensure a balanced dataset. For Regression, I used Linear Regression model after applying a MinMaxScaler to normalise data. Unfortunately, this model was too simple and underperformed severely in all aspects.	See Figures 14–16 below
Random Forest Classifier & Regressor	The Random Forest model, being more complex than the Logistic Regression, was better suited for both classification and regression tasks. I used RandomForestClassifer for predicting mortality status and RandomForestRegressor for predicting survival months. This model showed a significant improvement in classification accuracy and robustness. However, the regression performance was not much difference than the previous model.	See Figures 17–18 below
XGBoost Classifier & Regressor	From all the models tested, XGBoost performed the best for both classification and regression tasks. For classification, the dataset was balanced using SMOTE and split 80/20 for training and testing. For regression, the data was normalised using MinMaxScaler before training. Unfortunately, the regression Mean Absolute Error (MAE) was similar to that of Random Forest, XGBoost achieved a higher accuracy and delivered more stable and consistent predictions across both tasks.	See Figures 19–21 below

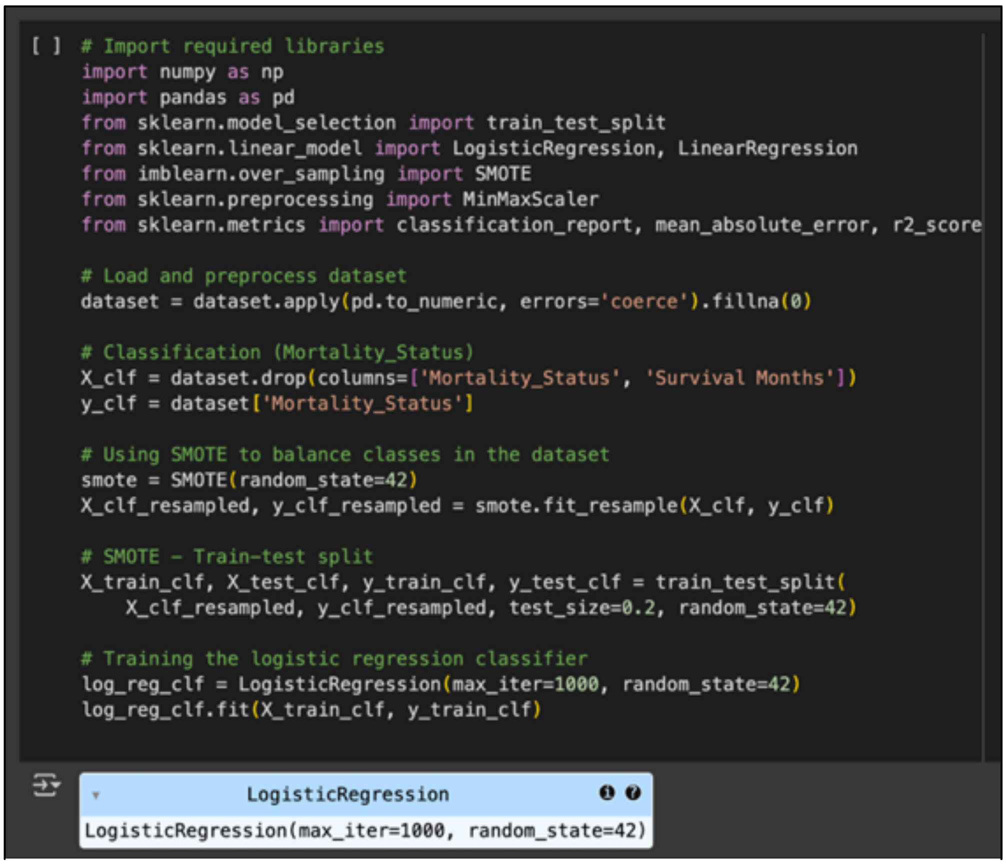
Table 7: Model Summary Table



```
[ ] # Regression (Survival Months) - Evaluate model
    y_pred_reg = lin_reg.predict(X_test_reg)
    mae = mean_absolute_error(y_test_reg, y_pred_reg)
    r2 = r2_score(y_test_reg, y_pred_reg)

    print("\n=== Regression Results ===")
    print(f"Mean Absolute Error (MAE): {mae:.2f}")
    print(f"R-squared Score (R²): {r2:.2f}")

    # Classification (Mortality_Status) - Evaluate model
    y_pred_clf = log_reg_clf.predict(X_test_clf)
    print("=== Classification Report ===")
    print(classification_report(y_test_clf, y_pred_clf))
```

```
=== Regression Results ===
Mean Absolute Error (MAE): 18.88
R-squared Score (R²): 0.04
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.64      0.71      0.67       691
           1       0.66      0.59      0.62       671

    accuracy                           0.65      1362
   macro avg       0.65      0.65      0.65      1362
weighted avg       0.65      0.65      0.65      1362
```

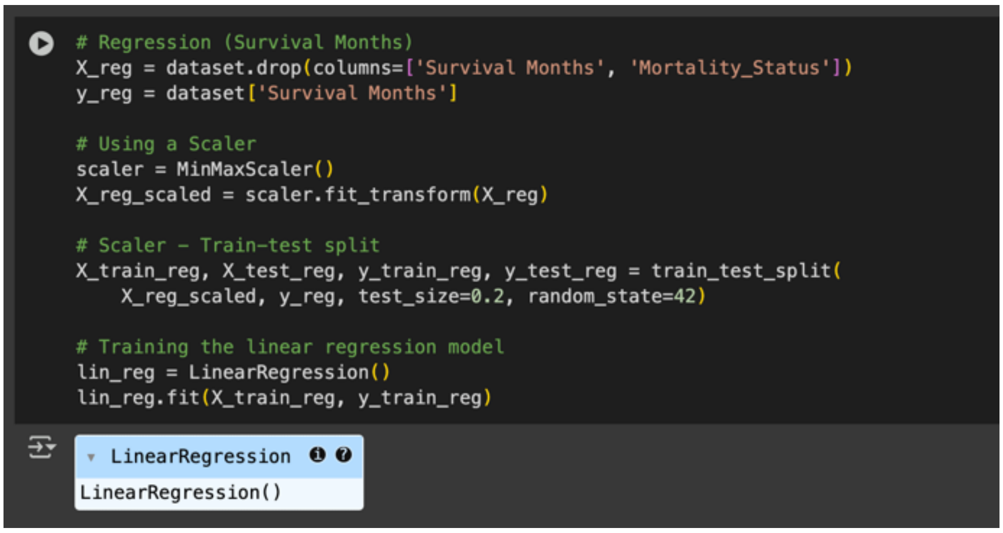
Figure 14: Implementation - Logistic Regression & Linear Regression Model Evaluation



```
[ ] # Import required libraries
    import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression, LinearRegression
    from imblearn.over_sampling import SMOTE
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.metrics import classification_report, mean_absolute_error, r2_score

    # Load and preprocess dataset
    dataset = dataset.apply(pd.to_numeric, errors='coerce').fillna(0)

    # Classification (Mortality_Status)
    X_clf = dataset.drop(columns=['Mortality_Status', 'Survival Months'])
    y_clf = dataset['Mortality_Status']

    # Using SMOTE to balance classes in the dataset
    smote = SMOTE(random_state=42)
    X_clf_resampled, y_clf_resampled = smote.fit_resample(X_clf, y_clf)

    # SMOTE - Train-test split
    X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(
        X_clf_resampled, y_clf_resampled, test_size=0.2, random_state=42)

    # Training the logistic regression classifier
    log_reg_clf = LogisticRegression(max_iter=1000, random_state=42)
    log_reg_clf.fit(X_train_clf, y_train_clf)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

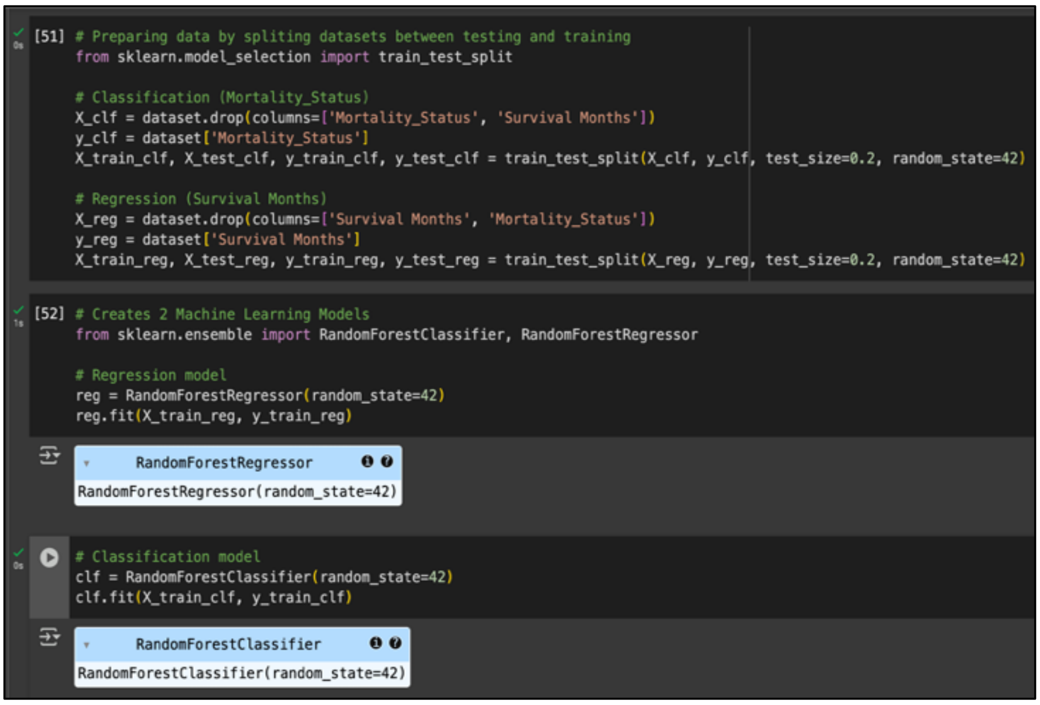
Figure 15: Implementation - Logistic Regression Model Development & Training



```
# Regression (Survival Months)
X_reg = dataset.drop(columns=['Survival Months', 'Mortality_Status'])
y_reg = dataset['Survival Months']

# Using a Scaler
scaler = MinMaxScaler()
X_reg_scaled = scaler.fit_transform(X_reg)

# Scaler - Train-test split
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(
    X_reg_scaled, y_reg, test_size=0.2, random_state=42)

# Training the linear regression model
lin_reg = LinearRegression()
lin_reg.fit(X_train_reg, y_train_reg)
```

```
LinearRegression
LinearRegression()
```

Figure 16: Implementation - Linear Regression Model Development & Training



```
[51] # Preparing data by spliting datasets between testing and training
     from sklearn.model_selection import train_test_split

     # Classification (Mortality_Status)
     X_clf = dataset.drop(columns=['Mortality_Status', 'Survival Months'])
     y_clf = dataset['Mortality_Status']
     X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(X_clf, y_clf, test_size=0.2, random_state=42)

     # Regression (Survival Months)
     X_reg = dataset.drop(columns=['Survival Months', 'Mortality_Status'])
     y_reg = dataset['Survival Months']
     X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg, test_size=0.2, random_state=42)
```

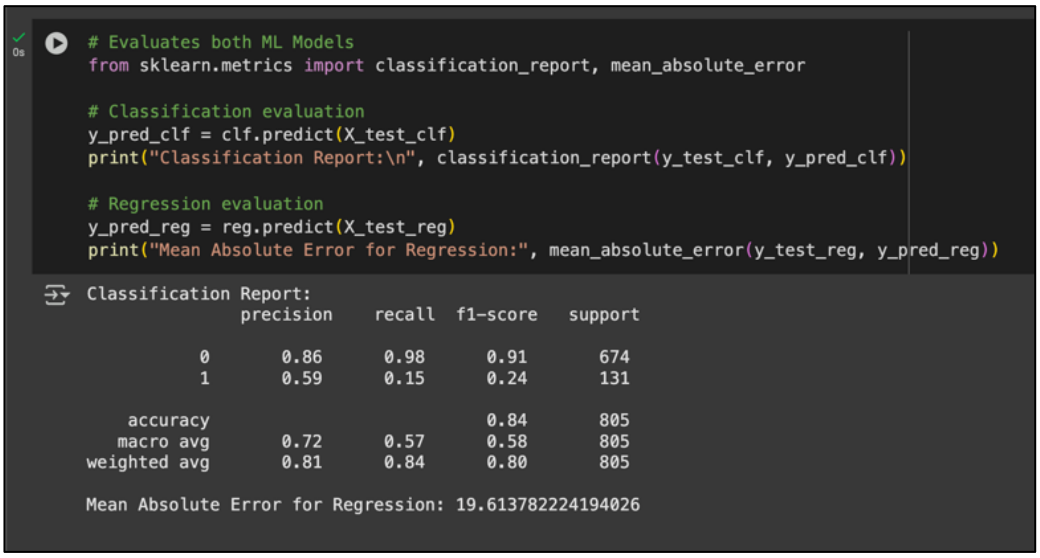
```
[52] # Creates 2 Machine Learning Models
     from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

     # Regression model
     reg = RandomForestRegressor(random_state=42)
     reg.fit(X_train_reg, y_train_reg)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
# Classification model
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_clf, y_train_clf)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

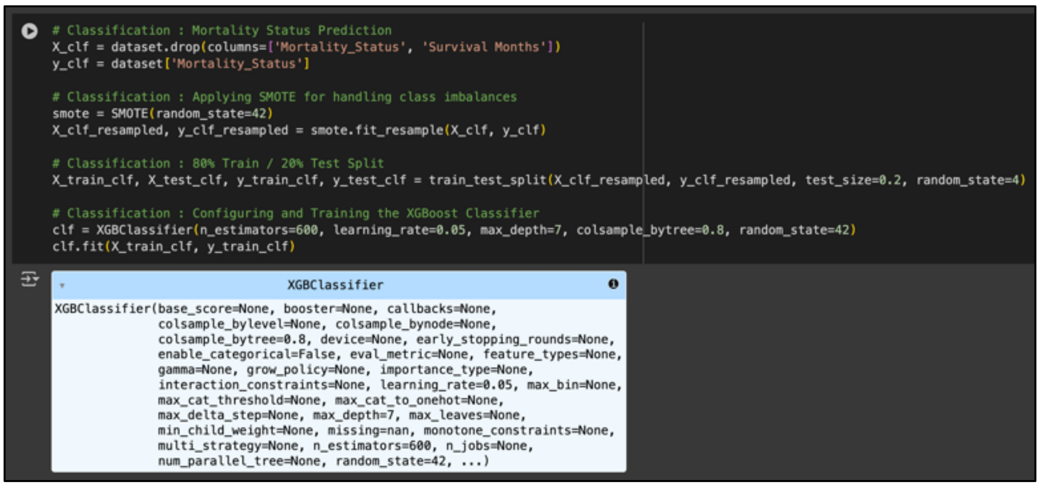
Figure 17: Implementation - Random Forest Models Development & Training



```
# Evaluates both ML Models
from sklearn.metrics import classification_report, mean_absolute_error

# Classification evaluation
y_pred_clf = clf.predict(X_test_clf)
print("Classification Report:\n", classification_report(y_test_clf, y_pred_clf))

# Regression evaluation
y_pred_reg = reg.predict(X_test_reg)
print("Mean Absolute Error for Regression:", mean_absolute_error(y_test_reg, y_pred_reg))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.98      0.91       674
           1       0.59      0.15      0.24       131

    accuracy                           0.84       805
   macro avg       0.72      0.57      0.58       805
weighted avg       0.81      0.84      0.80       805

Mean Absolute Error for Regression: 19.613782224194026
```

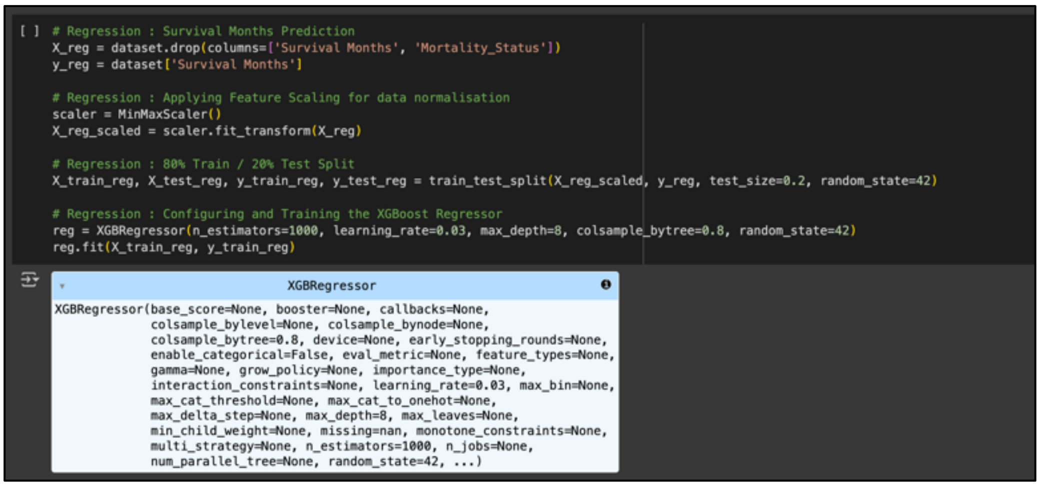
Figure 18: Implementation - Random Forest Models Development & Training



```
# Classification : Mortality Status Prediction
X_clf = dataset.drop(columns=['Mortality_Status', 'Survival Months'])
y_clf = dataset['Mortality_Status']

# Classification : Applying SMOTE for handling class imbalances
smote = SMOTE(random_state=42)
X_clf_resampled, y_clf_resampled = smote.fit_resample(X_clf, y_clf)

# Classification : 80% Train / 20% Test Split
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(X_clf_resampled, y_clf_resampled, test_size=0.2, random_state=4)

# Classification : Configuring and Training the XGBoost Classifier
clf = XGBClassifier(n_estimators=600, learning_rate=0.05, max_depth=7, colsample_bytree=0.8, random_state=42)
clf.fit(X_train_clf, y_train_clf)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.05, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=7, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=600, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

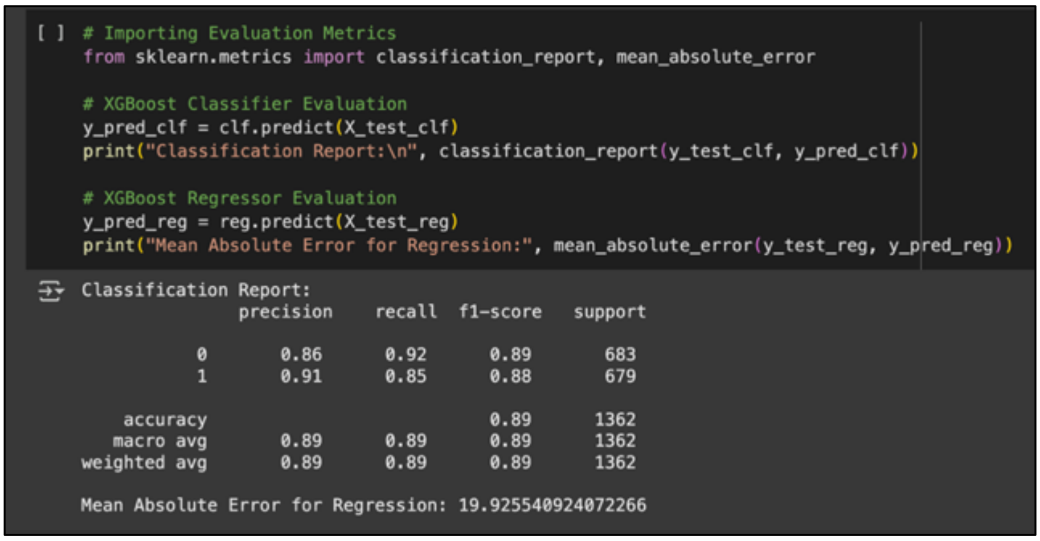
Figure 19: Implementation - XGBClassifier Development and Training



```
# Regression : Survival Months Prediction
X_reg = dataset.drop(columns=['Survival Months', 'Mortality_Status'])
y_reg = dataset['Survival Months']

# Regression : Applying Feature Scaling for data normalisation
scaler = MinMaxScaler()
X_reg_scaled = scaler.fit_transform(X_reg)

# Regression : 80% Train / 20% Test Split
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg_scaled, y_reg, test_size=0.2, random_state=42)

# Regression : Configuring and Training the XGBoost Regressor
reg = XGBRegressor(n_estimators=1000, learning_rate=0.03, max_depth=8, colsample_bytree=0.8, random_state=42)
reg.fit(X_train_reg, y_train_reg)
```

```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.8, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.03, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=8, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=1000, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

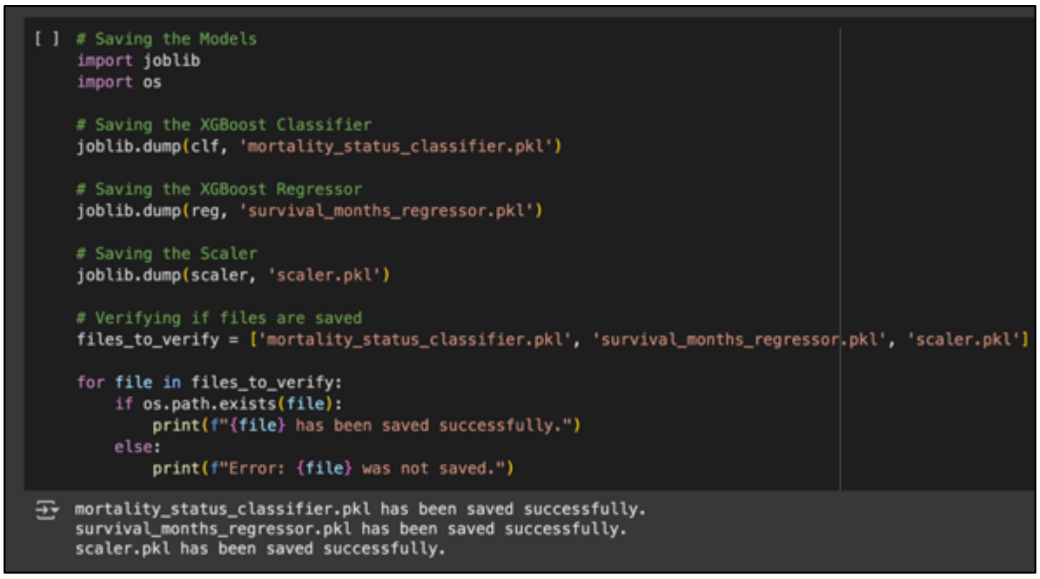
Figure 20: Implementation – XGBRegressor Development and Training



```
# Importing Evaluation Metrics
from sklearn.metrics import classification_report, mean_absolute_error

# XGBoost Classifier Evaluation
y_pred_clf = clf.predict(X_test_clf)
print("Classification Report:\n", classification_report(y_test_clf, y_pred_clf))

# XGBoost Regressor Evaluation
y_pred_reg = reg.predict(X_test_reg)
print("Mean Absolute Error for Regression:", mean_absolute_error(y_test_reg, y_pred_reg))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.92      0.89       683
           1       0.91      0.85      0.88       679

    accuracy                           0.89      1362
   macro avg       0.89      0.89      0.89      1362
weighted avg       0.89      0.89      0.89      1362

Mean Absolute Error for Regression: 19.925540924072266
```

Figure 21: Implementation – XGBoost Models Evaluation



```
# Saving the Models
import joblib
import os

# Saving the XGBoost Classifier
joblib.dump(clf, 'mortality_status_classifier.pkl')

# Saving the XGBoost Regressor
joblib.dump(reg, 'survival_months_regressor.pkl')

# Saving the Scaler
joblib.dump(scaler, 'scaler.pkl')

# Verifying if files are saved
files_to_verify = ['mortality_status_classifier.pkl', 'survival_months_regressor.pkl', 'scaler.pkl']

for file in files_to_verify:
    if os.path.exists(file):
        print(f"{file} has been saved successfully.")
    else:
        print(f"Error: {file} was not saved.")
```

```
mortality_status_classifier.pkl has been saved successfully.
survival_months_regressor.pkl has been saved successfully.
scaler.pkl has been saved successfully.
```

Figure 22: Implementation – Saving Models for Interface Integration

After extensive experimentation and comparison with other models, I decided to use XGBoost as the final model for deployment. It provided the best accuracy, computational efficiency and interpretability. The trained XGBoost model and scaler were saved and integrated into the Streamlit user interface for real-time predictions.

14. Features to be tested | Not to be tested

<u>Features to be Tested</u>

Features	Description	Requirement Addressed
Mortality Status Prediction	Ensure the XGBoost Classifier returns mortality predictions with >= 85% accuracy.	R01
Survival Months Prediction	Ensure that the XGBoost Regressor returns a Mean Absolute Error (MAE) of 20 or lower.	R01
Model Prediction Speed	Ensure the model delivers predictions within 1 minute of user input and submission	R02
User Input Validation	Ensure the model rejects invalid inputs	R10
Display Predictions / Visualisation in interface	Ensure the user interface clearly displays prediction outputs and visualisations in a structured format.	R03
Error Messages	Ensure the UI displays user- friendly and informative error messages when required.	R06
Model Predictions Disclaimer	Ensure a visible disclaimer displayed, stating that the model is experimental and not for clinical decisions.	R11
Restart Session Function	Ensure users can reset the interface to clear input fields and predictions.	R13

Table 8: Features to be Tested

<u>Features Not to Be Tested</u>

- PII Collection and Storage: Not tested as the model is not designed to collect, store or process any personally identifiable information. There is no database integration, and only anonymised data is used. Session data is cleared after use.

- Project Timeline Adherence: This is a project management task, not a functional feature therefore no testing is needed.

- Supervisor Communication: Regular stakeholder communication (via email or meetings) is a non-functional administrative task, therefore no testing needed.

- Security Features: There are no login, authentication, or encryption mechanisms in the current implementation. Therefore, security testing is out of scope for this project.

- Scalability: The current implementation is an independent prototype intended for local or small-scale use. Therefore, testing backend infrastructure scalability is not relevant at this stage.

15. Test Cases

Test Case ID	Test Objective	Expected Result	Actual Result	Verdict
TC001	Ensure the XGBoost Classifier returns mortality predictions with >= 85% accuracy	85% accuracy or higher	0.89	PASS
TC002	Ensure that the XGBoost Regressor returns a Mean Absolute Error (MAE) of 20 or lower.	20 MAE or lower	19.93	PASS
TC003	Ensure the model delivers predictions within 1 minute of user input and submission	Prediction Results returned in <=60 seconds	3 seconds	PASS
TC004	Ensure the model rejects invalid inputs	To reject invalid submission	Would not allow numbers to be inputted, only numeric values. Converted letter to numeric value and submitted the form	PASS
TC005	Ensure the user interface clearly displays prediction outputs and visualisations in a structured format.	UI shows a Guage visualisation with mortality and survival prediction	Displayed a Guage Visualisation with mortality status and survival time	PASS
TC006	Ensure the UI displays user-friendly and informative	An informative error message must display	Error Message display "Value must be greater than or equal to 0.1"	PASS

	error messages when required.			
TC007	Ensure a visible disclaimer displayed, stating that the model is experimental and not for clinical decisions.	Disclaimer is visible and states the model is experimental and not an alternative for clinical advice	Visible at top	PASS
TC008	Ensure users can reset the interface to clear input fields and predictions.	All fields and outputs are reset, and previous prediction is cleared	“Rerun” Streamlit button on top left of page	PASS

Table 9: Test Case Table

Project Summary

16. Delivering Sustainability

There are three pillars of sustainability: environmental, social and economic (enel, 2024). Throughout this project, I have aimed to contribute to a more sustainable healthcare future by demonstrating how machine learning can reduce costs, improve patient care, and minimise environmental impact.

- **Environmental Sustainability**: My current implementation is a cloud-based model using Google Colab and Streamlit. As the model is fully digital, it reduces the need for repeated in-person reviews, thereby lowering material usage and energy consumption. Additionally, efficient computation helps minimises the carbon footprint compared to traditional infrastructure.

- **Social Sustainability**: The model predicts a breast cancer patient's mortality status and survival time, supporting early intervention and improving patient's outcomes while reducing strain on healthcare services. The web interface and simple prediction outputs ensures usability across a range of healthcare environments and among professionals regardless of experience to reduce the chance of wrongfully interpreted prediction, which can lead to inappropriate clinical decisions.

- **Economic Sustainability**: Accurate mortality and survival predictions facilitates resources optimisation, helping healthcare professionals reduce unnecessary and time-consuming testing and intervention. Using open-source solutions and scalable frameworks, such as XGBoost, pandas and Streamlit, supports long-term sustainability and cost-efficiency.

17. Conclusions and Open Issues

<u>17.1 Open Issues</u>

- **Negative Tumour Size Values**: Some data values in the "Tumour Size" column contain negative invalid values. Unfortunately, this was not detected in data cleaning. This must be removed or corrected as it may impact predictions.

- **Incorrect Grade Encoding**: Unfortunately, in preprocessing, the Tumour grades were not properly mapped, affecting the model's accuracy. This must be fixed in data cleaning section.

- **High MAE in Regression (Survival Months)**: Despite using the XGBoost model, the MAE was still high for the survival months prediction model. Further improvements or configuration may be required.

- **Class Imbalance Risk**: While SMOTE was used to balance data in the final implementation, there is a risk of overfitting, reducing generalisation in real-world scenarios.

17.2 Future Enhancements

- **Integrate with a hospital patient database**: Allow healthcare professionals to save prediction results to a patient's record

- **Model Evaluation Button**: Allow users to view model performance metrics (e.g. Classification report, accuracy, MAE. Etc)

- **Save Prediction Button**: Allow users to save predictions to device in different export options such as PDF and CSV.

- **User Access Control**: Implement login and role-based access for healthcare staff

- **Autofill Feature**: To automatically input patient data into the prediction model

- **Device Compatibility**: Configure UI for mobile/tablet compatibility when desktop use is not available

- **Real-world Deployment:** Conduct regular GDPR and security audits, data encryption, bias testing and government approval before deployment

17.3 Lessons Learned

- **Technical Skills**: Throughout this project, I enhanced my knowledge of machine learning programming, data cleaning, and front-end interface development through self-directed learning.

- **Problem-Solving**: I refined my programming and adaptability skills by dealing with numerous unexpected issues, which also strengthened my research and troubleshooting abilities.

- **Project Management**: I developed stronger time management, organisation, planning, and strategic thinking skills, while also becoming more adaptable to the project.

17.4 Challenges Faced

- **Streamlit Integration**: As someone who has never used Streamlit before, I struggled with setting up and designing the interface. After research and troubleshooting, I built a functional and interactive front-end.

- **Machine Learning Models Development**: Besides experiences with simple ML Models in university, all knowledge used to build more complex models were self-taught. My initial errors taught me the importance of scaling and how to properly evaluate a model performance.

- **Data Cleaning**: I faced many issues when it came to mapping. I tried using a label encoder, but it didn't work. Therefore, I manually mapped the columns which simplified the process and ensured cleaner data.

17.5 Conclusion

This project successfully met its aim of developing a functional breast cancer mortality and survival prediction model using machine learning and Streamlit integration. The final model achieved 89% classification accuracy, aligning with the performance requirement of 85% or above. However, open issues remain - such as data cleaning errors, model underperformance in regression and ethical concerns - all must be addressed before real-world deployment or further development beyond the current scope.

Despite these challenges, the project truly improved my technical, problem-solving, and project management skills. Overall, this project contributes to knowledge by demonstrating how machine learning can be applied to assist in healthcare tasks and improve clinical workflows in a hospital setting. It also highlights the importances of technological innovation, ethical design, and transparency in building responsible clinical support tools.

Glossary

Term	Definition
Accuracy	The condition of correct prediction without error
Anonymised Data	Data with no personally identifiable information (PII)
Classification Model	ML Model which predicts categorical outcomes like "Alive" or "Dead"
Clinical Decision Support (CDS)	A tool helping healthcare professionals making decisions
Data Preprocessing	Cleaning and preparing data for machine learning
Electronic Health Record (EHR)	Digital patient health data system
GDPR	A UK/EU Data Privacy Law
Machine Learning (ML)	An Algorithm used to learn patterns or trends in data
Mortality Status	Predicted patient outcome (Alive/Dead)
Streamlit	A Python framework for developing front-end interfaces
XGBoost	A Machine Learning Algorithm used for predictions

Bibliography

Breast Cancer UK, 2024. *Facts and Figures.* [Online]
Available at: https://www.breastcanceruk.org.uk/about-breast-cancer/facts-figures-and-qas/facts-and-figures/#:~:text=There%20are%20around%2056%2C000%20new,in%20the%20UK%20(17).

enel, 2024. *The 3 pillars of sustainability: environmental, social and economic.* [Online]
Available at: https://www.enel.com/company/stories/articles/2023/06/three-pillars-sustainability

HuangLabUMN, 2024. *GitHub / oncoPredict.* [Online]
Available at: https://github.com/HuangLabUMN/oncoPredict

Jared lee Katzman, n.d. *DeepSurv GitHub Repository.* [Online]
Available at: https://github.com/jaredleekatzman/DeepSurv

Katzman, J. S. U. C. A. B. J. J. T. &. K. Y., 2018. *DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network.* [Online]
Available at: https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-018-0482-1?utm_source=chatgpt.com

Maeser, D. G. R. &. H. R., 2021. *OncoPredict: an R package for predicting in vivo or cancer patient drug response and biomarkers from cell line screening data..* [Online]
Available at: https://academic.oup.com/bib/article/22/6/bbab260/6321360

RAPID7, n.d. *Man in the Middle (MITM) Attacks.* [Online]
Available at: https://www.rapid7.com/fundamentals/man-in-the-middle-attacks/

TENG, J., 2019. *SEER BREAST CANCER DATA.* [Online]
Available at: https://ieee-dataport.org/open-access/seer-breast-cancer-data

[THE END]