GenomicRL: A DRL Framework for Cancer Treatment Recommendation using Genomic and Metastatic Markers

by

Heebatullah Beg

B.Eng., SSM College of Engg, University of Kashmir

A Project Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in the Department of Electrical and Computer Engineering

University of Victoria

GenomicRL: A DRL Framework for Cancer Treatment Recommendation using Genomic and Metastatic Markers

by

Heebatullah Beg

B.Eng., SSM College of Engg, University of Kashmir

Supervisory Committee

Prof. Hong-Chuan Yang, Supervisor

(Department of Electrical and Computer Engineering)

Prof. Fayez Gebali, Departmental Member

(Department of Electrical and Computer Engineering)

Abstract

Even with significant advances in cancer treatment, Gastroesophageal Junction (GEJ) cancers continue to present therapeutic challenges with a 5-year survival rate of just 21%. This work develops GenomicRL, a deep reinforcement learning (DRL) framework that integrates genomic, metastatic, and clinical markers to optimize treatment recommendations. Initially, a supervised learning (SL) baseline using ElasticNet achieves 64% exact match ratio (EMR) with clinician decisions. Augmenting training with synthetic data improves EMR to 70%, demonstrating generated data's utility in mitigating limited real-world samples. However, SL's reliance on historical decisions neglects post-treatment outcomes. To address this, a novel outcome-driven DRL agent is trained. Although the approach, incorporating survival, metastasis, and genomic stability into its reward function, reduces EMR from 99% (for clinician-mimicking reward function) to 73%, it achieves a higher average reward. Incorporating post-treatment signals, however, leads the agent to deviate from historical choices in ways that improve long-term outcome metrics—trading some immediate agreement for better anticipated patient benefit. This shift from pure imitation to outcome-oriented optimization highlights the promise of data-driven recommendation strategies that leverage diverse clinical and molecular information. Importantly, the proposed framework is not designed to supplant medical professionals but to assist them in refining treatment planning through personalized insights that account for individual patient variability.

Contents

List of Tables

List of Figures

List of Acronyms

AI	Artificial Intelligence
AWD	Alive With Disease
BCE	Binary Cross Entropy
CT	Computed Tomography
DOD	Dead of Disease
DNA	Deoxyribonucleic Acid
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
EEG	ElectroEncephaloGram
ELT	Extract, Load, Transform
FGA	Fraction Genome Altered
FISH	Fluorescence In Situ Hybridization
GAN	Generative Adversarial Networks
GEJ	Gastroesophageal Junction
HER	Human Epidermal growth factor Receptor
H-Pylori	Helicobacter Pylori
IHC	Immunohistochemistry
L1	Lasso Regression
L2	Ridge Regression

LeakyReLU	Leaky Rectified Linear Unit
LR	Low Risk
MDP	Markov Decision Process
ML	Machine Learning
MRA	Magnetic Resonance Angiography
MRI	Magnetic Resonance Imaging
MSE	Mean Squared Error
MSI	Microsatellite instability
MSK	Memorial Sloan Kettering
NED	No Evidence of Disease
PPO	Proximal Policy Optimization
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
SAC	Soft Actor-Critic
SL	Supervised Learning
UV	Ultra Violet
TD3	Twin Delayed DDPG
TMB	Tumor Mutational Burden
WES	Whole Exome Sequencing

Acknowledgments

First and foremost, I am deeply grateful to Allah (سُبْحَانَهُ وَتَعَالَى), the Most Gracious and Most Merciful, who has blessed me with the strength, patience, and perseverance to complete this endeavor and has carried me through every challenge. It is with His infinite mercy and guidance this milestone has been possible.

I am deeply indebted to my parents, whose patience, wisdom, and prayers have been a great source of motivation for me. To my siblings, thank you for standing by me through every step of this journey with both comfort and nagging. Your support has been my anchor, and your dua (دعاء) my shield.

I also extend my sincere thanks to my supervisor, Professor Hong Chuan Yang for this opportunity and also for his invaluable guidance, insightful feedback, and unwavering support throughout this journey. His mentorship has enriched this project and also significantly shaped my academic growth.

I would also like to extend my sincere thanks to my professors at UVic for the last 2 years. This project is a small expression of gratitude for the immense blessings I have received and a reflection of the support surrounding me.

Dedication

This project is wholeheartedly dedicated to my parents, who have been my unwavering pillars of strength and inspiration. To my beloved father (Baba), who faced cancer with unmatched courage and resilience. To my mother, whose prayers, patience, and sacrifices have been my greatest blessings—this achievement is as much yours as it is mine. To my siblings, my first friends, and lifelong companions, thank you for your boundless encouragement and belief in me, and for keeping life lively even when the stakes were high.

Chapter 1: Introduction

Cancer is the uncontrollable or rogue growth of the body's cells that can start almost anywhere and spread to other parts of the body. These cells may form tumors which can be cancerous (malignant) or non-cancerous (benign) [1]. The malignant ones can, oftentimes, travel and invade tissues nearby and then move on to distant places in the body to form new tumors resulting in metastasis. Although most cancers seem to be caused by a combination of several factors ranging from age to smoking, cancer is primarily believed to be a genetic disease caused by changes in our genes. These genetic changes can occur due to errors during cell division, DNA damage due to UV exposure or smoke, or something that we inherit from our parents.

The complexity of cancer treatment lies not only in its genetic foundations but also in its remarkable diversity, with over 200 distinct types requiring uniquely tailored approaches. The challenge becomes particularly evident in cases of GEJ tumors. Despite advances in multimodal treatment approaches, the prognosis for GEJ cancers remains poor, with a mere 21% five-year relative survival rate [2]. The National Cancer Institute's SEER (Surveillance, Epidemiology, and End Results Program) database gave an even more sobering statistic: approximately 39% of GEJ cancers are diagnosed at distant stages, with the five-year survival rate plummeting to 5.3% for metastatic cases [2]. Despite the severity of this cancer, there is a lack of proper open-source data available to further research in this domain.

In response to these challenges, this project is focused on developing a cancer treatment recommendation system specifically designed for GEJ cancers. Leveraging patient-specific data such as genomic profiles, metastatic patterns, and clinical histories, the system is designed to generate personalized treatment recommendations. Importantly, the intent of this system is not to supplant medical professionals, but rather to assist them in refining treatment planning through personalized insights that account for individual patient variability. Traditional treatment modalities, primarily aimed at maximal cell eradication, often encounter limitations when addressing metastatic cancers due to drug resistance. This project aims to follow the footsteps of precision medicine approach—grounded in the molecular characterization of tumors and individual genetic profiles [3] to tailor therapeutic strategies. This recommendation system,

therefore, represents a crucial step toward integrating advanced AI methodologies with clinical expertise to enhance treatment outcomes in cancer care.

1.1 AI in Oncology

Over the years there have been several advancements in cancer treatment with AI and Machine Learning (ML) models. Researchers have adopted supervised and unsupervised learning algorithms for diagnosis and prognosis. There are several studies investigating the role DRL agents can play in cancer treatment recommendations. A lot of those studies are focused on Breast and lung cancer as they form around 27.2% of cancer incidences globally [4]. The application of DRL in cancer treatment has shown promising results across various domains:

Liu et al. proposed PPORank, a DRL-based framework for personalized cancer treatment recommendations. By modeling drug ranking as a Markov Decision Process (MDP), PPORank adaptively identifies optimal therapies based on individual molecular and clinical profiles, outperforming traditional SL approaches in large-scale cancer cell line datasets [5]. In another paper by Yauney and Shah (2018), an RL-based optimization of chemotherapy regimens was suggested to achieve better outcomes in clinical trials while reducing adverse effects [6].

Yoon et al. (2022) proposed an adaptive dosing strategy using Deep Q-Networks (DQN), demonstrating improved progression-free survival in simulated breast cancer cases [9]. Chen and others. (2023) introduced a time-sensing mechanism in the DRL model for dynamic therapy adjustment. It shows 30% better results compared to the static protocol [7]. In a real-time study, Park and others. (2023) conducted one of the first prospective trials using DRL for prostate cancer treatment, by examining the safety and effectiveness of the approach in healthcare [8].

Owen et al. (2018) emphasized the importance of tailoring treatments to individual patient characteristics [10], highlighting the potential of RL in oncology. Although RL applications in cancer care are relatively new and underexplored, studies have shown promise in optimizing chemotherapy dosages, radiation planning, and dynamic treatment regimens. Gallagher et al. leveraged DRL to design adaptive cancer treatment protocols, demonstrating its ability to extend progression-free [11] survival in prostate cancer models.

While these advancements demonstrate the versatility of AI in oncology, the specific challenges of GEJ cancers such as genomic heterogeneity and limited therapeutic targets, demand a tailored approach that integrates both molecular profiling and dynamic treatment optimization.

1.2 Challenges and Solutions

The development of a cancer treatment recommendation system presents the following complex challenges:

1. High-quality clinical data for GEJ cancers is limited, and the available data often suffers from imbalance and bias. This scarcity not only makes it challenging to maintain clinical validity but also increases the risk of over- or underfitting in data-driven models.

2. The inherent variability in patient characteristics encompassing genomic profiles, metastatic patterns, and clinical characteristics complicates the formulation of effective treatment recommendations.

The proposed solution includes the following steps:

1. Address the challenge of data scarcity in medical research by employing data augmentation techniques to generate proper synthetic data. This is followed by a two-stage training approach, pre-training on synthetic data and fine-tuning on real patient data.
2. Develop and implement a DRL framework that can effectively process and learn from complex patient data to provide cancer treatment recommendations; all while taking before and after treatment results into account.

1.3 Structure of the report

This project report is divided into six chapters. Chapter 2 provides an overview of ML, SL and Reinforcement Learning (RL). Chapter 3 explores the dataset utilized in this project and the different categories of its features. Chapter 4 explores an SL algorithm for a two-stage training process highlighting the significance of GAN and its impact on key metrics. Chapter 5 describes the DQN-based RL approach for this recommendation system, including MDP formulation and

metrics to analyze the performance of the DQN agent. Chapter 6 suggests future research directions.

Chapter 2: Machine Learning

ML is a subset of AI that has revolutionized several aspects of our lives, from weather predictions to business decisions. It plays a key role in healthcare by enabling disease prediction through pattern recognition and data analysis. Its ability to process complex clinical datasets, allows for the identification of subtle patterns and risk factors that traditional methods might overlook. For cancer research, the ML algorithms can be used to analyze imaging and genetic profiles and comprehensive patient histories in order to predict disease possibility, progression, and prognosis which can inform treatment decisions at an early stage, thereby facilitating timely interventions and improving outcomes.

2.1 Supervised Learning

In SL, models are trained on labeled data to learn the mapping between input features and known outcomes. The process is underpinned by the belief that patterns identified in historical data can generalize to unseen instances if the model has sufficient data to represent this and is equipped with appropriate regularization techniques to mitigate overfitting. This approach is essential for both classification tasks, where data are sorted into discrete categories, and regression tasks, which predict continuous values. The accuracy of the predictions depends on the quality and quantity of the labelled data available for training purposes [12]. The formalization of an SL algorithm typically involves a dataset as:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \tag{2.1}$$

where D represents the dataset which comprises an input feature vector $x_i \in R^d$ (where R^d represents the d-dimensional real space) and a corresponding target variable y_i. The central objective becomes the approximation of a function $f\colon R^d \rightarrow R$ (or R^c for multi-output scenarios) that minimizes a predefined loss function $L(f(x_i),\ y_i)$ across the training distribution.

Classification

Classification is a specialized branch of SL focused on assigning discrete labels to input instances [13]. The theoretical foundation rests on the estimation of decision boundaries within the feature space. From a probabilistic standpoint, the goal is to model the conditional probability $P(y|x)$

and select the target label y that maximizes this probability for a given input x. In practice, classification tasks involve mapping an input vector $x_i \in R^d$ to a discrete output space $y \in \{1,2,\dots, C\}$, where C denotes the number of classes. While binary classification (with C=2) is conceptually simpler, multi-class classification requires more nuanced approaches such as one-versus-all or one-versus-one strategies. Classification finds its application in tasks such as sentiment analysis, spam email detection, etc.

Regression

Regression analysis is another fundamental component of SL that deals with predicting continuous output variables from input features [13]. Unlike classification, regression focuses on forecasting numerical values along a continuous spectrum, the effectiveness of which depends on robust feature selection and precise preprocessing. The aim is to approximate a function $f: R^d \rightarrow R$ such as to minimize the difference between predicted and observed values, typically measured using loss functions like squared error or absolute deviation. The classic linear regression model is defined by the equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d + \epsilon \tag{2.2}$$

where y is the predicted or dependent variable, β_0, β_1, β_2, ... are called the regression coefficients where β_0 is the intercept, β_1 is the coefficient of independent variable x_1, and so on and ϵ is a random error component. Regression techniques are widely used in fields such as financial forecasting to predict stock prices or economic indicators based on historical data and various market factors. A popular algorithm in regression is linear regression, a model that estimates the linear relationship between two variables by fitting a linear equation to observed data.

ElasticNet

The ElasticNet model, implemented in Chapter 4 of this project, is a regularized linear regression that combines the properties of L1 and L2 regularization, where L1 regularization (Lasso) encourages sparsity by driving some coefficients to zero and is key in feature selection and L2 regularization (Ridge) shrinks coefficients towards zero (not to exactly 0) thereby preventing overfitting and handling multicollinearity (when predictor variables are highly correlated). The mathematical formulation of ElasticNet loss can be written as:

$$L = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\left(\alpha\sum_{j=1}^{p}|\beta_j| + (1-\alpha)\sum_{j=1}^{p}\beta_j^2\right) \quad (2.3)$$

Where n is the number of observations, $\hat{y}_i$ and y_i are the predicted and actual values respectively; α is the mixing parameter to determine the balance between L1 and L2 penalties; $\sum_{j=1}^{p}|\beta_j|$ is the L1 penalty inducing sparsity and $\sum_{j=1}^{p}\beta_j^2$ is the L2 penalty encouraging small coefficients with β_j representing the weights or coefficients of the regression model and λ is the regularization parameter to control the overall strength of the regularization that helps control the complexity of the model [14]. When $\lambda = 0$, no regularization is applied and the model only focuses on minimizing the prediction error and when $\lambda \gg 0$, the penalty term dominates that helps prevent overfitting [14]. As scikit-learn is used to call the linear model ElasticNet, the key parameters λ corresponds to alpha (default = 1.0) and α corresponds to l1_ratio ($\in[0,1]$). If l1_ratio = 1, the penalty would be L1 penalty. If l1_ratio = 0, the penalty would be an L2 penalty. If the value of l1 ratio is between 0 and 1, the penalty would be a combination of L1 and L2.

2.2 Reinforcement Learning

RL is a powerful ML paradigm that employs the interaction of an agent with an environment to learn to make optimal decisions based on the feedback it receives from the environment. This learning process is typically modeled using MDPs, characterized by the tuple (S, A, P, R, γ), where S is the state space, A is the action space, P defines state transition probabilities, R is the reward function, and γ is the discount factor that balances immediate and future rewards [15]. The agent observes the current state s_t, selects an action a_t based on a policy $\pi: S \rightarrow A$, receives a reward r_t, and transitions to a new state s_{t+1}. Through this cycle, the agent iteratively refines its policy to maximize the expected cumulative reward. With each algorithmic iteration, the output is presented to the interpreter, responsible for determining the favorability of the outcome. When the algorithm produces a correct solution, the interpreter reinforces this success by administering a reward. The two main players in an RL setting are Environment and Agent.

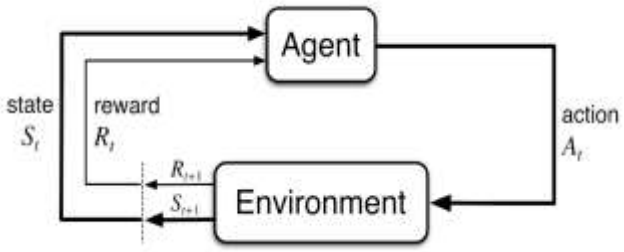

Figure 2.1: Agent-environment Interaction in an MDP

In Figure 2.1 “Environment” denotes the external setting or framework wherein an “agent” operates, acquires knowledge, and executes decisions. RL can be implemented using various algorithmic approaches, including value-based methods (e.g., Q-learning, DQN), policy gradient techniques (e.g., REINFORCE, Proximal Policy Optimization), and actor-critic architectures that combine elements of both paradigms. This framework has been successfully applied to various domains, including robotic control, autonomous navigation, and resource allocation. Its strength lies in the agents’ ability to learn optimal strategies through exploration and exploitation without explicit instructions.

2.3 Google Colaboratory

Google Colaboratory (Colab) is a high performance, zero-configuration, cloud-based environment typically used for ML research. As an integrated Jupyter notebook platform, Colab allows the execution of Python code with the added advantage of free access to powerful Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). The system’s architecture includes a user-friendly web interface for code editing and visualization, backed by virtual machine instances that handle intensive computational workloads. It also integrates seamlessly with Google Drive and GitHub, supporting collaborative research and version control. It has played a crucial role in accelerating innovation in ML by bridging the gap between exploratory data analysis and production deployment.

Its architectural framework comprises several interdependent components: a frontend web interface for code editing and result visualization, backend virtual machine instances that execute computational workloads, persistent storage mechanisms for notebooks and associated data assets, and hardware acceleration infrastructure for intensive mathematical operations common

in ML applications. The hardware specifications typically include NVIDIA Tesla K80, T4, or P100 GPUs with corresponding memory capacities ranging from 12GB to 16GB, enabling the training of moderately complex neural network architectures without prohibitive computational bottlenecks. Furthermore, the integration with Google Drive and GitHub provides flexible options for data storage, code versioning, and distribution of completed projects to the broader scientific community. The programmatic interface of Colab extends beyond standard Python execution to include system-level commands through magic functions and shell access, enabling comprehensive environment configuration and dependency management.

Pandas is a Python library for data manipulation and analysis that simplifies working with tabular data, offers data frames and series for tasks like cleaning, exploration, and transformation [16]. Matplotlib is another Python library for 2D plotting, used to create static and interactive plots [16]. NumPy is a Python library for scientific computing that supports large multi-dimensional arrays and offers mathematical functions for array operations [16]. Scikit-learn is a library for ML that provides efficient tools for data mining and data analysis, including algorithms for classification, regression, clustering, and more [16]. PyTorch is an open-source ML library focused on deep learning. It offers dynamic computational graphs and GPU acceleration, making it a popular choice for both research and production [17].

Chapter 3: Dataset Preparation

3.1 Dataset Description

The primary dataset used for this project is the Metastatic Esophagogastric Cancer publicly accessible from the cBioPortal for Cancer Genomics [18] hosted by the Center for Molecular Oncology at Memorial Sloan Kettering Cancer Center (MSK). The dataset contains targeted sequencing of 341 samples for metastatic esophagogastric cancer patients [18]. The dataset contains information regarding the clinical, genomic and metastatic parameters before and after administering cancer treatments. In addition to this, another set of data is extracted from the Esophageal Cancer-TRAP Project [18] with targeted sequencing of tumor samples from Esophageal cancer patients, increasing the number of samples to 409. The majority of the patients are males (70%) with four common cancer therapies (chemotherapy, targeted molecular therapy, radiation therapy, immunotherapy) being administered.

3.2 Dataset Preparation

This is the very first stage of data preprocessing where the data is gathered from multiple sources into a central repository (Google drive) for storage, processing and model building. The features (clinical, genomic and metastatic parameters), are extracted and appended (from the aforementioned datasets) to create a combined real-world dataset. The dataset originally contains 26 features for different stages of GEJ cancer patients ranging from stomach to gastroesophageal junction adenocarcinoma. To align the dataset with the requirements for this project, relevant information is extracted from these columns: Received IO (Immunotherapy) after First Line and Treatment in order to determine the specific therapies the patients' received, thus creating Treatment columns – Chemotherapy, Immunotherapy, Radiation Therapy and Targeted Molecular Therapy. The columns are rearranged and Received IO after First Line and Treatment columns are dropped, giving us an updated set of 28 features.

An additional advantage of this dataset is that it not only contains the 28 features for before treatment is administered but also after the treatment is administered. This information is flagged by IMPACT pre/post column as PRE-TX (indicating the row is pre-treatment data for a patient) and POST-TX (indicating the row is post-treatment data for a patient). The combined dataset [18,

19] is built in such a way as to offer comprehensive insights into patient characteristics, clinical metrics, and treatment outcomes. The majority of patients received chemotherapy (~75%) either as the only treatment or in combination with targeted or immunotherapy while only 7 patients received Radiation therapy. Additionally, the data contains valuable information about the patients' overall vital status (AWD - *Alive with Disease*, DOD - *Dead of Disease*, LTF - *Lost to follow-up*, NED - *No Evidence of Disease*. An overview of all the features in the dataset used for this project:

Table 3.1: Features of the Dataset Used

Feature Name	**Description**
Patient ID	Unique identifier assigned to each patient.
Sample ID	Unique identifier assigned to each sample collected per patient
Cancer Type	Broad category or type of cancer (e.g., esophageal, esophagogastric).
Cancer Type Detailed	Detailed classification of cancer type (Adenocarcinoma of the Gastroesophageal Junction, Stomach Adenocarcinoma, Esophageal Adenocarcinoma).
Mutation Count	Total number of genetic mutations identified in the sample.
Fraction Genome Altered	Proportion of the genome affected by mutations.
Age at Diagnosis	Patient's age at the time of cancer diagnosis.
HER2 IHC or FISH	Diagnostic assay to determine HER2 protein expression and gene amplification.
H-PYLORI	Whether the patient has gastric infections (presence can influence treatment response).
HER2 IMPACT	Assesses the functional effect of HER2 alterations on tumor aggressiveness.
HER2 IMPACT LR	Refined evaluation of HER2 impact to quantify its influence on tumor behavior.

IHC-HER2	Immunohistochemical test measuring HER2 protein levels in tumor tissue.
IMPACT pre/post	IMPACT analysis results before and after treatment.
Liver Metastasis	Presence of liver metastasis.
Lung Metastasis	Presence of lung metastasis.
MSIsensor Score	Microsatellite instability (MSI) sensor score.
Mutation Rate	Frequency of mutations per genomic region.
Vital Status	Current survival status of the patient.
Peritoneum Metastasis	Presence of peritoneal metastasis.
Somatic Status	Status of somatic mutations.
Stage At Diagnosis	Stage of cancer at the time of diagnosis.
TMB (nonsynonymous)	Tumor mutational burden for nonsynonymous mutations.
Tumor Grade	Histological grade of the tumor.
Overall Survival (Months)	Total survival time from diagnosis (months).
Chemotherapy	Has the patient had chemotherapy?
Radiation Therapy	Has the patient had Radiation Therapy?
Immunotherapy	Has the patient had Immunotherapy?
Targeted Molecular Therapy	Has the patient had Targeted Molecular Therapy?

To prepare the data for the SL model (Chapter 4) and DRL agent (Chapter 5), the features are categorized into groups to have a smooth data pipeline before the training begins. The dataset features can be broadly categorized into 3 groups:

3.2.1 Categorical Features

These discrete variables capture specific characteristics of the patient and their treatment:

- **Cancer type classifications, Stage at Diagnosis and Tumor Grade:** This includes both general and detailed cancer, essential for identifying the best treatment options based on the cancer's nature and subtypes.

- **IMPACT pre/post:** This column indicates whether the row corresponds to a patients' pre-treatment data (PRE-TX) or post-treatment data (POST-TX).

- **Vital status:** This feature combines information regarding the patient's survival status and the status of their disease, which is crucial for evaluating treatment success. This feature is further categorized into four statuses, each providing significant information regarding the patient's current status: No Evidence of Disease (NED), Alive with Disease (AWD), Dead of Disease (DOD) and Lost to Follow-up (LTF).

- **Somatic status, HER2 IHC or FISH, H-Pylori, HER2 Impact, IHC-HER2:** This tracks genetic alterations in the tumor, providing insights into how these mutations may influence treatment efficacy.

3.2.2 Binary Features

These features focus on treatment and metastasis states:

- **Treatment options:** These binary indicators capture whether patients have undergone specific therapies, such as chemotherapy, radiation therapy, targeted molecular therapy, or immunotherapy. These variables are essential for understanding treatment regimens and their impact on patient outcomes.

- **Metastasis sites:** By identifying the presence of metastasis in organs such as the liver, lung, or peritoneum, this feature helps in understanding the extent of disease spread and its influence on treatment strategies.

3.2.3 Numerical Clinical Measurements

These features provide quantitative data for patient health and genomic profiles:

- **Patient demographics (Age at Diagnosis):** Age at diagnosis is a critical factor influencing treatment choices and prognosis.

- **Genomic metrics (Fraction Genome Altered, Mutation Count, HER2 IMPACT LR, and Mutation Rate):** These biomarkers reveal mutation-driven patterns that can significantly influence how effective a treatment is, thereby informing personalized clinical decision-making.

- **Microsatellite Instability scores (MSI sensor Score):** This score helps in determining the genetic instability of the tumor, which is critical for selecting therapies that target genomic instability.

- **Treatment response indicators (TMB nonsynonymous):** Tumor mutational burden (TMB) serves as an important biomarker to assess the likelihood of a favorable response to immunotherapy.

- **Overall Survival in Months:** This feature tracks the survival duration post-treatment, which is essential for evaluating the long-term efficacy of cancer therapies.

As a part of preprocessing, the categorical variables are transformed via label encoding where each category is assigned an integer value to help the ML models process categorical information effectively (example: for Vital status $NED \rightarrow 0$, $AWD \rightarrow 1$, $DOD \rightarrow 2$ and $LTF \rightarrow 3$); binary features are assigned two possible values 0/1 (example: 0 indicating absence of liver metastasis and 1 indicating presence of liver metastasis); and lastly, numerical variables are replaced using median values followed by standard scaling. This normalization process is essential to mitigate scale differences and missing value effects, ensuring that the subsequent modeling stages are not confounded by extraneous variability. The fit transform of Scaler (applying fitted scaling technique such as standardization to adjust feature values) computes the necessary parameters (mean and standard deviation) and applies scaling to standardize the feature values. For given a feature x, the normalized value z is computed as:

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

where μ is the mean and σ is the standard deviation of the numerical value for the feature; this is to ensure the features have 0 mean and a unit variance. Median imputation was selected over mean imputation to preserve the skewness inherent in clinical biomarkers, while standardization

(eq. 3.1) ensures that features such as TMB and mutation count (features that operate on divergent scales) contribute equally to the model's objective function. Additionally, as data privacy is important when dealing with medical data, the columns with patient information (Patient_ID and Sample_ID) are anonymized using a random key (original Patient_IDs are replaced by a standard format P_000001) to ensure anonymity, after removing any duplicates. The original ID columns are then dropped. The datatype of the columns is converted from object type to numerical type. Any non-numerical values are first replaced by a generic NaN and then replaced using median imputation.

Now that the ELT (Extract, Load and Transform - a data processing approach where raw data is first extracted from source systems, loaded into a storage system, and then transformed for analysis) stage is completed, the output label (treatments) is assigned to variable y and dropped from the feature set X (all the features mentioned in table 3.1 excluding the treatment labels). The data is split into train (80%) denoted by X_{train}, y_{train} and test (20%) denoted by X_{test}, y_{test}. The features are used as input to the model and the target labels are what the supervised model will try to predict (Chapter 4) and what the agent will decide (Chapter 5).

Chapter 4: Treatment Recommendation Using SL

One way to approach the problem of treatment recommendation systems using ML is the use of SL algorithms trained on labeled data, where the model learns mapping from input data to known output labels. One such algorithm is ElasticNet (discussed in Chapter 2), a linear regression model that is good at handling datasets with correlated features. Our treatment predictor employs ElasticNet regression (as mentioned in Chapter 2) to model recommendations by capturing predictive features for each treatment. As mentioned in Chapter 2, ElasticNet balances L1 and L2 regularization by minimizing the loss function (eq. 2.3) with GridSearchCV being used for hyperparameter optimization.

4.1 Data Augmentation Strategy

To overcome the limitations posed by scarce clinical data, we adopt a two-stage training process. This section explores the integration of synthetic data generation techniques with ElasticNet for our treatment recommendation system. The primary objective is to quantitatively assess if this approach of synthetic data training and real data fine-tuning would be effective in achieving congruence between the agent-generated recommendations and those provided by clinicians. One key technique is the Generative Adversarial Network (GAN) that enables the generation of synthetic data that mimics real patient profiles.

4.1.1 Generative Adversarial Network

As mentioned above, to curtail the limited availability of real-world patient data, a GAN architecture will be explored for a a two-stage training process with the purpose of getting as close to the doctor's recommended treatments as possible. The GAN architecture has two primary components that work in a feedback loop to generate synthetic data that closely resembles real data: the generator and the discriminator, the basic architecture for which is mentioned below:

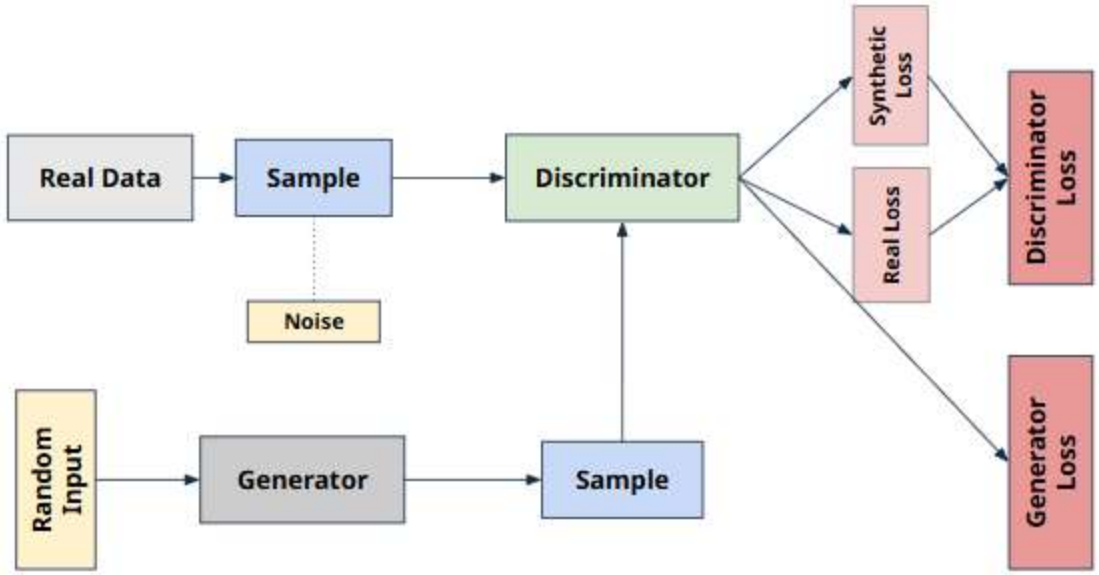

Figure 4.1: GAN Architecture [20]

Generator Architecture

The generator is a neural network responsible for producing synthetic data to mimic real-world medical data, thereby fooling the discriminator. It begins with an input layer that accepts a latent vector z sampled from a standard normal distribution $N(0,1)$. This latent vector serves as the source of randomness, which the generator learns to transform into data that resembles the characteristics of real patient data. Following the input, the generator includes several hidden layers with 2^n neurons, utilizing activation functions such as ReLU or LeakyReLU to introduce non-linearity and mitigate issues like dead neurons. The final output layer applies an activation function (Tanh or Linear) to ensure that the synthetic data has the appropriate dimensionality and characteristics.

Discriminator Architecture

In parallel, the discriminator operates as a neural network that differentiates between real and synthetic data. It accepts an input vector corresponding to the real data's dimensionality. It processes this information through hidden layers that typically decrease in 2^n neurons size, again using activation functions that allow small gradients for negative inputs. The discriminator has an output layer that maps the processed features to a single probability value, usually through a

Sigmoid activation function, which indicates whether the input sample is real (with a high probability) or synthetic (with a low probability). The commonly implemented loss function for a GAN is the minimax loss [21] expressed as:

$$\mathcal{L} = \min_{G} \max_{D} \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [(\log(1 - D(G(z)))] \tag{4.1}$$

with the generator trying to minimize it and the discriminator trying to maximize it [21]. $E_{x \sim p_{data}(x)}[\log D(x)]$ is maximized when *D* assigns high probabilities to data drawn from the true distribution p_{data}. If *D* is good at recognizing real samples, $\log D(x)$ is large. $\mathbb{E}_{z \sim p_z(z)}[(\log(1 - D(G(z)))]$ term is maximized when *D* assigns low probabilities to data generated by *G*. For the purpose of this project, Binary Cross Entropy (BCE) [21] loss is employed, as it naturally fits the binary classification task inherent in the GAN's adversarial training, while also aligning with the theoretical minimax loss formulation. The BCE loss is defined as:

$$L_{BCE} = -(y \log(p) + (1 - y) \log(1 - p)) \tag{4.2}$$

where y represents the true label ($y = 1$ for real data and $y = 0$ for synthetic data) and p is the probability predicted by the *D* that the same is real. For real samples, the term $y \log(p)$ encourages the discriminator to assign a high probability, whereas for synthetic samples, the term $(1 - y) \log(1 - p)$ pushes *D* to assign a low probability.

4.2 Two-stage Model Training

We begin with a comprehensive data augmentation stage that leverages a GAN to generate synthetic data. In this system, the real dataset consists of 28 features across 409 datapoints, and the GAN is used to create a much larger synthetic dataset—over 10,000 samples—after training for 1000 epochs.

As previously mentioned, the GAN framework is built on two neural networks, Generator (G) and Discriminator (D), which operate in an adversarial manner. The generator's task is to map a latent vector $z \in \boldsymbol{R}^{100}$ to the data space R^d, where d is the number of features, in order to produce synthetic data that closely approximates the real data distribution. This mapping is achieved through a network that begins with a fully connected input layer. It progresses through

three hidden layers, and utilizes LeakyReLU activation functions to introduce the non-linearity. The output layer employs a Tanh activation function to ensure that the output values conform to the range and dimensionality of the real data.

The discriminator, on the other hand, is designed to differentiate between real and synthetic samples. It receives data samples and processes them through hidden layers that typically reduce the input dimensionality and also applies LeakyReLU activation functions. The output layer of *D* is a fully connected layer that produces a single probability value via a Sigmoid activation function. This probability indicates the likelihood that the input data is real, with values closer to 1 representing real data and values closer to 0 representing synthetic data.

An essential part of the training is the loss function, for which Binary Cross Entropy (BCE) was used. This is to quantify the difference between the Discriminator's predictions and the actual labels for both real and synthetic data. The overall loss for *D* is a combination of the loss on real data and the loss on synthetic data. Specifically, the loss on real data (L_{real}) is calculated as expectation of the BCE loss for real samples:

$$L_{real} = -\mathbb{E}_{x\sim p_{data}}[\log D(x)] \tag{4.3}$$

where $D(x)$ is *D*'s output probability for a real sample x. Conversely, the loss on synthetic data L_{synth} is computed as:

$$L_{synth} = -\mathbb{E}_{z\sim p_z}[(\log(1 - D(G(z))))] \tag{4.4}$$

where $G(z)$ represents the synthetic data generated from the latent vector z drawn from the distribution p_z and $D(G(z))$ is the discriminator's predicted probability that the generated sample is real. The total discriminator loss is then given by:

$$L_D = L_{real} + L_{synth} \tag{4.5}$$

The Generator is trained to minimize this same objective, thereby producing synthetic samples that increasingly challenge the Discriminator's ability to distinguish them from real data. A DataLoader, B = 32, plays a crucial role in managing real data samples during training, ensuring that the generator learns the underlying distribution effectively. The Adam optimizer, with a learning rate of 0.0002, is employed to facilitate stable convergence during training. Through

iterative adversarial training, the Generator becomes proficient in synthesizing data that mimics the real distribution, which is vital for augmenting the dataset and improving the robustness of the treatment recommendation system. Below is the algorithm for the generation:

4.2.1 Synthetic Data Generation Algorithm

Algorithm 1 Synthetic Data Generation Algorithm

Input: original_data D_{real}, n_samples, epochs
Output: Synthetic data D_{synth}
Normalize original_data to [-1, 1] then convert to tensor and create a DataLoader ($\mathcal{B} = 32$)
Define latent_dim (100) and data_dim (number of features)
Instantiate the Generator (latent_dim) and Discriminator models (data_dim)
Define optimizers for both networks with learning rates
Set BCE as the loss criterion
Define a noise factor (0.05) to add small perturbations to real data
for epoch = 1 to epochs **do**
    Train Discriminator:
    Set real labels and synthetic labels
    Add noise to real data
    Compute discriminator loss on real data:

$$L_{real} = -\mathbb{E}_{x\sim p_{data}}[\log D(x)]$$

    Generate noise vector z for the current batch and produce synthetic data:

$$synth\_data = G(z)$$

    Compute discriminator loss on synthetic data:

$$L_{synth} = -\mathbb{E}_{z\sim p_z}[(\log(1 - D(G(z))))]$$

    Total discriminator loss:

$$L_D = L_{real} + L_{synth}$$

    Backpropagate L_D and update the Discriminator parameters
    Train Generator:
    for each generator update **do**
        Generate new noise z and compute synthetic data
        Compute generator loss by comparing discriminator output to real labels*:*

$$L_G = BCE\big(D\big(G(z)\big), 1.0\big)$$

        Backpropagate L_G and update the Generator parameters
    end for
end for
Set Generator to evaluation mode
Generate synthetic data = Generator(noise)
Inverse scale to original_data range
return synthetic data as a DataFrame

A quick summary of the parameters:

Table 4.1: GAN Parameters

Component	**Parameters**
GAN Generator	$Input \rightarrow 256 \rightarrow 512 \rightarrow 1024 \rightarrow N\text{umber } of\ features$
G Hidden Layers	LeakyReLU activation
G Output Layer	Tanh output activation
GAN Discriminator	$Input \rightarrow\ 512 \rightarrow\ 256 \rightarrow O\text{utput}$
D Hidden Layers	LeakyReLU activation
D Output Layer	Sigmoid output activation

ElasticNet Model Training

The ElasticNet model training occurs in two stages, first with the synthetic data followed by fine-tuning on real data.

Hyperparameter Tuning with GridSearchCV

GridSearchCV is an automated method that systematically evaluates every combination in a predefined grid of hyperparameter values. The two key hyperparameters being tuned here are alpha (in scikit-learn; it corresponds to λ in eq. 2.3) and l1_ratio (in scikit-learn; it corresponds to α in eq. 2.3, determining the balance between L1 and L2 regularization). The process starts by partitioning the available data to perform k-fold cross-validation $(k = 5)$ for each combination of hyperparameters. For every combination in the parameter grid, the model is trained on a subset of the data (4-folds) and validated on the remaining portion (1-fold), ensuring that the hyperparameter tuning is robust against overfitting. This process is repeated k times with each fold serving as the validation set once. The performance is averaged over k folds:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.6}$$

where y_i is the actual output variable and $\hat{y}_i$ is the predicted output. As GridSearchCV maximizes the score, the negative MSE is used (lower MSE becomes a higher negative value). Once all possible hyperparameter combinations are evaluated, GridSearchCV selects the combination that

achieved the best average score across the folds. The best parameters are used to re-fit the model on the full training set to finalize the model. This is to ensure that the parameters are not just tuned for a single data split but are robust across different partitions of the training data.

Synthetic Data Training

This stage acts as a form of pre-training where the ElasticNet models are trained on synthetic data using GridSearchCV with a wide range of hyperparameters, allowing the models to learn coarse feature relationships. The phase ends once all the possible combinations are evaluated and the best estimator hyperparameter combination is selected. The weights learned during this training phase are preserved to be utilized during the real data training phase.

Real Data Fine-tuning

The synthetic data training is followed by fine-tuning the model on the real data which helps the model to learn from the actual, observed patterns and distributions in the real world. The pre-training phase acts as a warm-start for the fine-tuning phase with the parameters being adjusted to better capture the nuances of the real data, thereby improving the accuracy of the results post real data training. The weights preserved during the synthetic data training phase are further refined for a more targeted grid search. Once the training is complete, the model for each treatment is saved for later use during prediction.

4.2.2 Two-stage SL Training

Algorithm 2 Two-stage ElasticNet Training Algorithm

Input: Synthetic data, real data, treatments T
Output: Trained models and evaluation metrics
procedure PREPAREDATA(data)
    Define features X by dropping treatment columns
    Define target vector y as the treatment columns
    return X, y
end procedure
Split real data into train and test sets (80/20)
Initialize metrics tracker.
Stage 1: Pre-training on Synthetic Data
Start training timer.
for each treatment in T **do**
    Extract features X_{synth} and target y_{synth} for the current treatment from synthetic data.
    Initialize base_model = ElasticNet

15: Define parameter grid for grid search.
16: Run GridSearchCV (with 5-fold CV and scoring neg MSE) on X_{synth}, y_{synth}.
17: Retrieve best hyperparameters from the grid search.
18: Initialize best model with the best hyperparameters.
19: Train best model on X_{synth}, y_{synth}.
20: Save best model in predictor.treatment models.
21: **end for**
22: Stop training timer.
23: **Stage 2: Fine-Tuning on Real Data**
24: **for** each treatment in T **do**
25: Extract features X_{real} and target y_{real} for the treatment from training data.
26: Clone the pre-trained model for the current treatment and retrieve current alpha and l1_ratio from the model.
27: Define fine-tuning parameter grid:
{'alpha': [0.1×current alpha, current alpha, 10×current alpha], 'l1 ratio': [max(0, current ratio−0.2), current_ratio, min(1, current_ratio+0.2)]}
28: Initialize base_model = ElasticNet
29: Run GridSearchCV (estimator = base_model, param_grid = grid, cv = 5, scoring = 'neg_mean_squared_error')on (X_{real}, y_{real}).
30: Retrieve best hyperparameters from the grid search.
31: Update the cloned model with the best hyperparameters.
32: Train the updated model on (X_{real}, y_{real}).
33: Save the updated model in predictor.treatment models[treatment].
34: **end for**
35: Evaluation on real test data
36: **return** trained model and evaluation metrics

A quick overview of the ElasticNet parameters:

Table 4.2: ElasticNet Treatment Predictor Parameters

Parameter	**Description**
Model Type	ElasticNet
Cross-validation	5-fold
Alpha Range	[0.001, 0.01, 0.1, 1.0]
L1 Ratio Range	[0.1, 0.3, 0.5, 0.7, 0.9]

Prediction

For a new patient, their relevant features (such as demographic data, medical history, etc.) are extracted and prepared in the same way as the training data. The trained model converts regression outputs into probability (if the score is > 0.5, the treatment is predicted; otherwise, it is not) using

the models' internal calibration and the final predictions for all treatments are returned for the new patient. Additionally, multiple treatments can be recommended simultaneously, if appropriate.

4.3 Evaluation Metrics

The evaluation phase compares the agents' multi-label treatment recommendations against the clinician's decisions. Each recommendation is represented as a 4-dimensional binary vector. The following key metrics are computed:

Mean Squared Error (MSE)

The MSE is calculated as:

$$MSE = \frac{1}{N \times 4}\sum_{i=1}^{N}\sum_{j=1}^{4}\left(y_{ij} - \hat{y}_{ij}\right)^2 \tag{4.7}$$

where y_{ij} is the ground truth (doctor's recommendation) for patient i and treatment j, $\hat{y}_{ij}$ is the agent's predicted recommendation.

Exact Match Ratio

This metric represents the proportion of patients for whom all four treatment recommendations are predicted exactly:

$$EMR = \frac{1}{N}\sum_{i=1}^{N}\mathbf{I}(\hat{y}_i = y_i) \tag{4.8}$$

where N is the number of samples in the test data, $\mathbf{I}\{\hat{y}_i = y_i\}$ is an indicator function that returns **I** if the entire predicted vector $\hat{y}_i$ matches the ground truth vector y_i and 0 otherwise.

4.4 Results and Analysis

The evaluation phase compares the agent's multi-label treatment recommendations against the clinician's decisions and the metrics mentioned in section 4.2 are calculated. A quick comparison of the results with and without a two-stage training process are as follows:

Table 4.3: SL Results Comparison

Metrics	**Model Training (only real data)**	**Two-stage Training**
MSE	0.1986	0.1631
Exact Match Ratio	64.70%	70.08%

The numbers indicate that the two-stage training approach outperforms the model trained when solely on real data. With only real data, the MSE of 0.1986 suggests that, on average, the predicted treatment probabilities are somewhat close to the doctor's recommendations, EMR of 64.70% shows that just over half of the complete treatment recommendations match exactly with those made by doctors.

In contrast, the two-stage training process yields a lower MSE of 0.1631, meaning the predictions are, on average, much closer to the clinician's recommendations. Moreover, the EMR increases to 70.08%, demonstrating that nearly 70% of the recommendation vectors produced by the model perfectly align with those from the clinician. This improvement in both metrics implies that incorporating synthetic data in the initial training phase, followed by fine-tuning on real data, effectively enhances the model's performance, likely by providing a more balanced and enriched learning environment.

Per-Treatment Accuracy

For each treatment j, the accuracy is:

$$Accuracy_j = \frac{1}{N}\sum_{i=1}^{N} \mathbf{I}\left(y_{ij} = \hat{y}_{ij}\right) \tag{4.9}$$

where $I(\cdot)$ is the indicator function and y_{ij} is the actual treatment administered by the doctors and $\hat{y}_{ij}$ is the model recommended treatment. Per-treatment accuracy yielded values of 81.41% for Chemotherapy, 100% for Radiation Therapy, 96.85% for Targeted Molecular Therapy, and 64.66% for Immunotherapy.

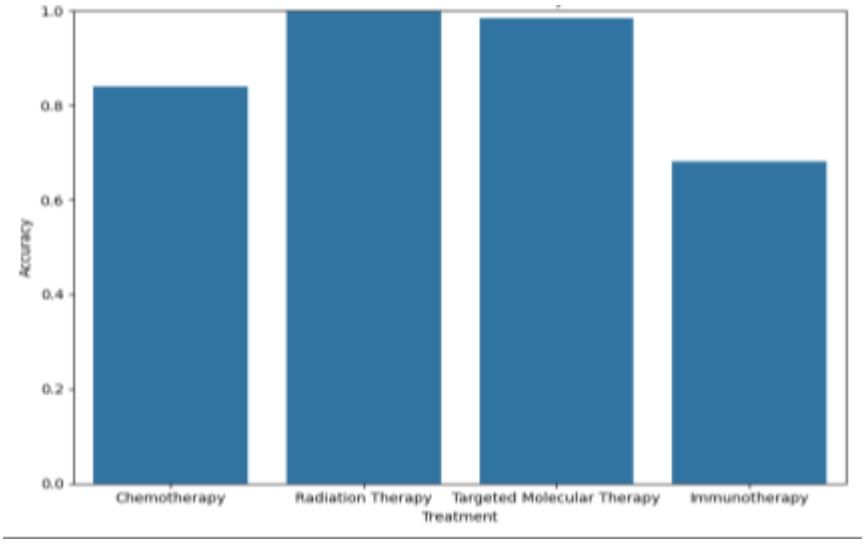

Figure 4.2: Per Treatment Accuracy

This tells us that the model performs very well for Radiation Therapy and Targeted Molecular Therapy, but it struggles relatively more with Immunotherapy, which might be due to either less training data for that treatment.

The proposed treatment recommendation system successfully integrates synthetic data generation and ElasticNet modeling to produce multi-label predictions that closely align with clinician recommendations. By employing GAN to generate synthetic samples, the system alleviates some of the challenges posed by limited and imbalanced clinical data. The evaluation metrics, including low MSE, and a respectable EMR, demonstrate the models' ability to replicate the doctor's recommendations.

The results achieved by SL so far seem to be fair, but the system seems to have a moderate EMR which can probably be enhanced by using other complex SL algorithms. However, this approach relies on historical clinical decisions as static labels, which may not capture long-term treatment

effectiveness. Additionally, it does not take into account how the features change before and after the recommended treatments are administered, which is crucial in medical decision-making.

Chapter 5: Treatment Recommendation Using DRL

SL models, even when bolstered by synthetic data augmentation, are inherently limited by their dependence on static historical labels. As mentioned in the previous chapter, SL does not capture the underlying long-term effectiveness of those decisions. The key limitation lies in its inability to account for the clinical outcomes that unfold after a treatment is administered. DRL offers a compelling alternative to overcome this limitation. With a flexible reward structure, the agent can adapt its decision-making process based on individual patient outcomes. This dynamic capability is crucial for developing personalized treatment recommendations that account for the complex variability in patient profiles. This approach represents a shift from merely mimicking historical decisions to proactively optimizing recommendations based on their clinical impact, offering an effective strategy for a more personalized approach.

5.1 MDP Formulation

The treatment recommendation problem was formulated as a decision-making task within an MDP framework defined by the tuple: (S, A, R, γ), where S represents the state space, A is the action space, R is the reward function and γ is the discount factor. At each time step t, the agent (treatment recommender) receives the current patient state s_t, selects a treatment action a_t according to its policy π and then observes a reward r_t.

State Space: The state space S is the input to our agent. The state is composed of the patient's current clinical and genomic features and metastasis status. The complete state space S consists of multiple feature vectors representing patient status:

$$s_t = \begin{bmatrix} g_t \\ c_t \\ m_t \end{bmatrix} \tag{5.1}$$

where $g_t \in R^5$ represents the genomic features vector with Fraction Genome Altered (FGA), Mutation Count, MSIsensor Score, TMB (nonsynonymous), HER2 IMPACT LR; $m_t \in \{0,1\}^3$ is a binary vector which represents metastasis status, c_t denotes the clinical features such as the age at diagnosis, somatic status, etc.

Action Space: The action space was designed to represent all possible treatment combinations excluding an all-zero vector. Each action represents a specific treatment protocol, encoded in a way that captures both individual treatments and their combinations. The action space consists of 15 possible treatment combinations, represented as a discrete 4-bit binary number. Each bit encodes the presence or absence of a specific treatment modality - chemotherapy, radiation therapy, targeted molecular therapy, and immunotherapy:

$$A = \{4 - bit\ binary\ vectors\} \backslash \{(0,0,0,0)\} \tag{5.2}$$

The action space A consists of 15 possible treatment combinations represented as a 4-bit binary vector: $A = 2^4 - 1 = 15$. For instance, an action $a = (1,0,0,1)$ represents administering Chemotherapy and Immunotherapy together.

Reward Function: Two types of reward functions are built.

Reward Function I: As in Chapter 4, this reward function is curated to mimic the clinicians' immediate decisions:

$$R(s,a) = \begin{cases} +1, & \text{if the agent's decision matches the doctor's recommendations} \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

Reward Function II: As mentioned before, an interesting feature of the dataset is that it contains the patients' response to the treatment recommended by the doctor. This is flagged as POST-TX in the IMPACT pre/post column.

The doctors' decisions, while expert-informed, might not always be optimal in every case. Post-treatment data. And as the data has post treatment patient information, it's fair to use an alternate reward function to analyze the impact of post-treatment data on the performance of the agent. To utilize this feature, the data is sorted by the IMPACT pre/post column. This column flags whether the corresponding data was collected before the treatment was administered or after the treatment was administered. The data is then grouped using the Patient_ID. This is, of course, followed by the same set of data preprocessing as mentioned in Chapter 3 and the same training process. The reward $R(s,a)$ is a weighted sum of clinical outcomes (vital component, genomic change and metastasis component):

$$R(s, a, s') = \alpha \cdot v(s') + \beta \cdot \Delta m(s, s') + \gamma \cdot \Delta g(s, s') \quad (5.4)$$

Where $\alpha = 1,\ \beta = 0.5,\ \gamma = 0.3$ are assigned to the vital status $\big(v(s')\big)$ (NED= +2.0, AWD = +1.0, DOD = -2.0, LTF = 0.0) change in metastasis $\big(\Delta m(s, s')\big)$ and change in genomics $(\Delta g(s, s'))$. The highest weight is assigned to vital status. As metastasis risk is next on the hierarchy it is weighted at 0.5 to encourage treatments that lower the risk of metastasis, followed by genomic features. The hierarchy of the weights aligns with clinical guidelines where survival supersedes secondary biomarkers in therapeutic decision-making.

In both the cases, the process is modeled as a 1-step transition so the episode terminates immediately after the reward is received.

5.2 DQN-based Solution

Each patient encounter represents a discrete episode comprising a pre–treatment state and a treatment decision where the goal for the agent is to match the doctor's recommendations. The problem is solved using a DRL approach, specifically the DQN algorithm using Python in the Google Colaboratory environment. Our DQN agent, trained with real data, learns to optimize treatment recommendations to improve clinical outcomes while adhering to established guidelines.

The agent in our DRL framework functions analogously to a multidisciplinary team of oncologists, radiologists, and other specialists who analyze a patient's clinical data, genomic profile, metastatic status, along with other factors to determine optimal treatment regimens. At each decision epoch t, the agent receives a state vector s_t (i.e., patient features) and chooses an action a_t (treatment combinations) based on its current policy. Once an action is taken, the environment responds with a reward r_t.

As discussed in Chapter 2, several DRL algorithms are available, but for this project we implement DQN, a model-free RL algorithm. Traditional Q-learning uses a Q-table to store and update the quality of actions for each state–action pair, but this approach becomes impractical when the state and action spaces grow large—as in the case of cancer treatment recommendation.

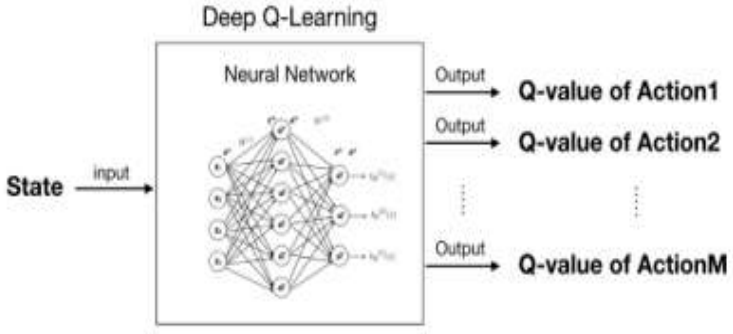

Figure 5.1: DQN Working

DQN employs a neural network ($Q(s, a; \theta)$) to the traditional Q-learning framework to approximate the Q-value function instead of a Q-table, thereby addressing the scalability issue. It takes a state s as input and outputs estimated Q-values for all possible actions a. This combination of deep learning with Q-learning enables the agent to generalize across large state spaces. In our formulation, the DQN acts as the automated agent that processes patient data and outputs a recommended combination of treatments. At each decision epoch t, the agent chooses an action based on its current policy:

$$a_t^* = \ argmax\ Q(s_t, a; \theta) \tag{5.5}$$

where θ are the parameters of a target network that is periodically updated.

5.2.1 Environment

The environment in our framework represents the clinical setting where cancer treatment decisions are made. It is modeled as a single, complex system that abstracts the interactions between various biological processes, treatment modalities, and individual patient characteristics into a unified framework. The environment provides the observation space (the state) to the agent as an array of patient features, including genomic markers and metastatic patterns. In each episode, the environment provides a pre–treatment state s_t (a vector of patient features), receives the agent's treatment action a_t, and then returns a reward r_t. The reward reflects clinical outcomes such as improvements in genomic markers or adverse events resulting from treatment. Although many factors are at play, for simplicity we model the process as a one-step transition, meaning

that once an action is taken and the immediate reward is received, the episode is considered complete.

5.2.2 Policy

The policy π is a mapping from states to actions:

$$\pi: s \rightarrow a \tag{5.6}$$

In DQN, this policy is implicitly derived from the Q-function. After executing the action a_t and the agent observes the immediate reward r_t. The objective is to learn the optimal Q-function $Q^*(s, a)$ satisfying the Bellman optimality equation:

$$Q(s, a; \theta) = r + \gamma \max_{a'} Q(s', a'; \theta) \tag{5.7}$$

where: s is the current state, a is the action taken, r is the immediate reward, $\gamma (= 0$ for this project), $Q(s, a; \theta)$ is the action-value function. The learning process involves minimizing the following loss:

$$\mathcal{L}(\theta) = \left(\frac{1}{2M}\right) \sum_{i} (Q(S_i, A_i;\ \theta) -\ y_i)^2 \tag{5.8}$$

where M is the mini-batch size. and $(Q(s_i, a_i; \theta) - y_i)^2$ is the MSE between the online network's Q-value and the target Q-value. This loss is minimized via gradient descent, and the DQN parameters are updated accordingly. A target network is used to stabilize training by periodically copying the online network's parameters. In the current problem, the DQN agent processes patient data and outputs a recommended combination of treatments. Over repeated interactions with the environment, the DQN agent learns to maximize the immediate reward, thereby adapting its treatment recommendations to follow the doctor's recommendations. The policy is learned using DQN with the following model parameters:

Table 5.1: DQN Parameters

Parameter	**Value**
Network Architecture	$Input \rightarrow 128 \rightarrow 256 \rightarrow 128 \rightarrow Output$

Experience Replay Buffer Size	20,000
Learning Rate	lr = 0.0005
Total Training Timesteps	3000
Target Network Updates	Every 5 episodes
Batch Size	32

5.3 DQN Training Process

The DQN architecture comprises three fully connected hidden layers and an output layer corresponding to the action space. A separate target network, used to stabilize the learning, is also initialized and periodically updated to mirror the online network. Additionally, a replay buffer is set up to store transition tuples (s, a, r). For each training episode, a patient is randomly selected. The state vector is constructed from pre-processed clinical, metastatic, and genomic features. The agent uses DQN to compute Q-values $Q(s, a; \theta)$ for all actions and selects the one with the highest value. After decoding the action into a treatment vector, the reward is computed. The loss is defined as the MSE between the current Q-values and the target values as mentioned in eq. 5.8. The running reward (an exponentially weighted moving average of rewards) is logged over episodes. This metric is later visualized to assess the learning progress. The process continues until the agent is trained over 3000 episodes. The goal is to achieve a high EMR indicating that the agent's recommendations closely align with those made by clinicians. The trained DQN agent is evaluated on the test_set and evaluation metrics such as the MSE and EMR are calculated using eq. 4.7 and 4.8 respectively. In addition to this, the average reward is also calculated.

5.3.1 DQN Training Algorithm

Algorithm 3 General DQN Training Algorithm

1: Input: Real data, treatments T, episodes, mini-batch size M, discount factor $\gamma = 0$, learning rate lr
2: Output: Trained DQN parameters θ, treatment_recommendations.csv
3: Split data into train and test sets (80/20)
4: Initialize the DQN network with parameters θ.
5: Initialize the target DQN network with parameters $\theta_t \leftarrow \theta$.
6: Initialize replay buffer D with a set capacity (e.g., 20,000).
7: **for** episode $k = 1\ to\ N$:

8: Randomly sample a patient's sample (single sample or a pre/post pair, for 5.3 and 5.4 respectively).
9: Set current state $S \leftarrow s$.
10: Select action A = argmax(Q(s, a; θ)).
11: Decode action A into a treatment vector.
12: Compute reward R.
13: Store the experience tuple in the replay buffer E.
14: If the size of E is at least M:
15: Sample a mini-batch from E
16: **for** each sample, compute target:

$$y_i = R_i \text{ (since } \gamma = 0\text{, future rewards are ignored)}$$

17: Compute the loss:

$$L = \left(\frac{1}{2M}\right)\sum_i (Q(S_i, A_i;\ \theta) - y_i)^2$$

18: Update the network parameters θ using the Adam optimizer with gradient clipping.
19: Every T episodes (target update frequency), update the target network:

$$\theta_t \leftarrow \theta.$$

20: **end for**
21: **end for**
21: Evaluate the trained DQN on test data
22: Save each patient's ID and treatment recommendation in treatment_recommendations.csv.
23: Save the trained DQN parameters θ
24: **return** θ and treatment_recommendations.csv

5.4 DQN Learning Curve

The learning curves are quite typical and suggest that the training process is working.

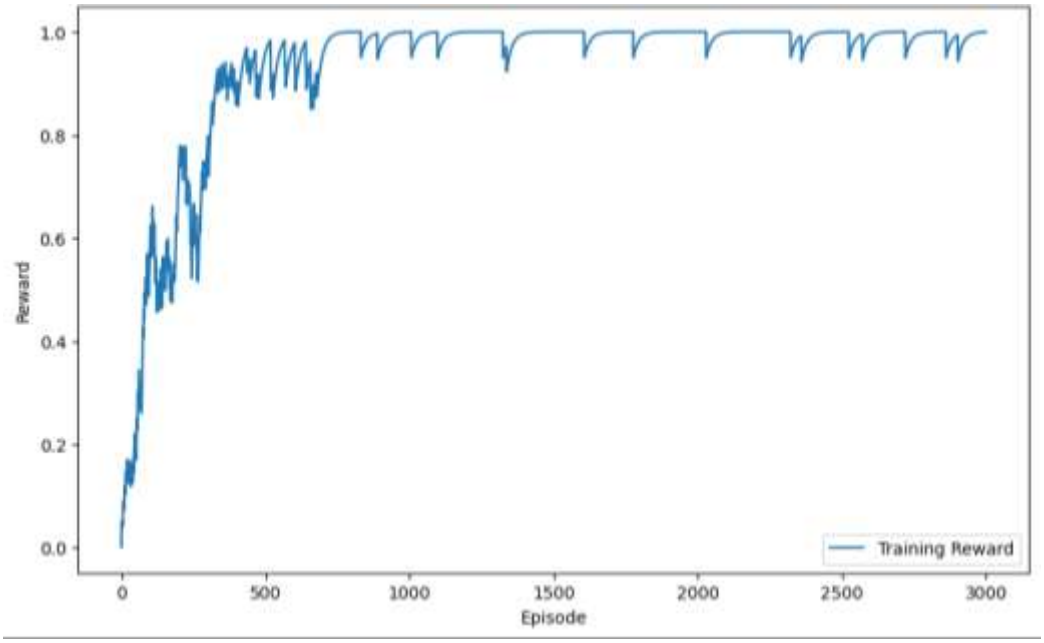

Figure 5.2: Learning Curve: Training Includes Only Pre-treatment Data

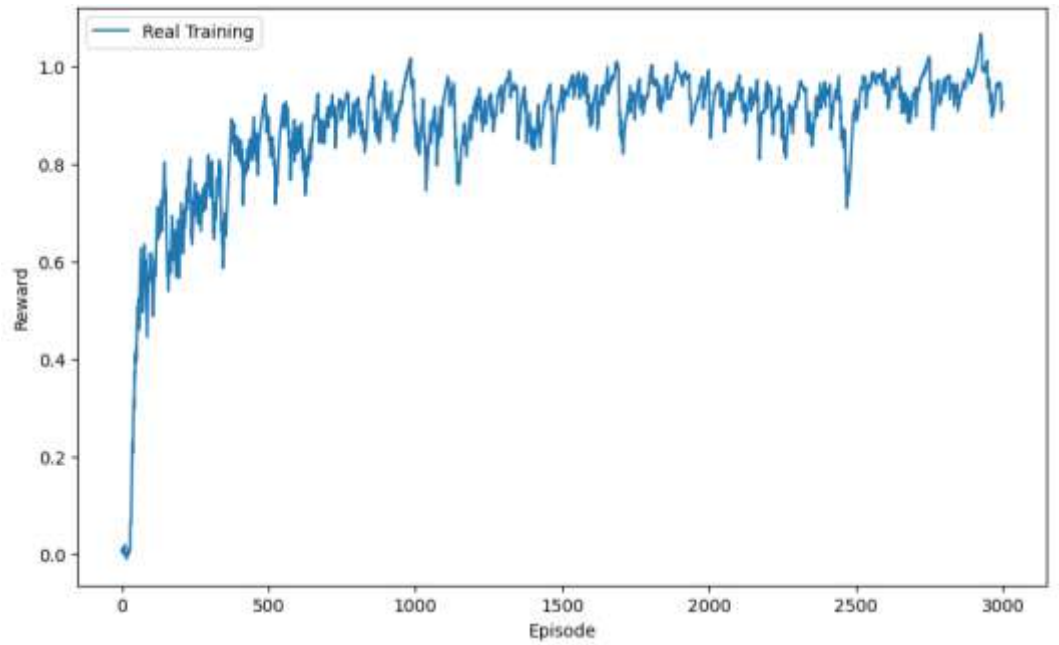

Figure 5.3: Learning Curve: Training Includes Post-treatment Data

In figure 5.2, the curve shows the agent's reward steadily increasing and eventually stabilizing near 1.0. There is a rapid rise in the reward, indicating that the agent quickly learns to match or closely replicate the doctor's pre-treatment recommendations. The plateau near the top suggests that once the agent has discovered the "best" action (i.e., matching the doctor's recommendation), there is little incentive to explore alternatives.

In figure 5.3, although the learning curve still trends upward, the behavior may show more variability or take longer to converge. By factoring in post-treatment data, the reward signal seems to be influenced by outcome-related information.

5.5 Evaluation Metrics and Results

The following metrics were calculated to evaluate the performance of the DQN agent:

Exact Match Ratio

For the action space (encoded as a 4-bit integer), the EMR is calculated for the training set as:

$$EMR = \frac{Number\ of\ episodes\ where\ a_{agent} = a_{doctor}}{Total\ no.of\ episodes} \tag{5.9}$$

Table 5.2: DQN Results

Metrics	DQN (only Pre-treatment)	DQN (incl. Post-treatment)
EMR	99.06%	73.04%
MSE	0.0002	0.145

With only pre-treatment data, the agent closely mimics the doctor's decisions (nearly 99% exact match). This high alignment is natural because the reward function encourages agreement with the doctor. When post-treatment data is introduced, the agent's alignment with the doctor's recommendations drops (73.04%). This indicates that the agent is sometimes deviating from the doctor's pre-treatment choices. The reason can be that it sees signals from post-treatment outcomes that suggest different actions may be more beneficial in the long run. To properly evaluate the agent's performance we consider the following rewards:

Full-test Reward (0.6508): This was calculated by scoring the agent's recommendations on the post-treatment reward, so it reflects outcome-driven performance rather than simple decision concordance.

Doctor-agreement Reward (0.9750): This was obtained by computing average reward only on those test cases where the agent's recommendation matched the doctor's; by restricting evaluation to "agreement" cases.

Together, these show that incorporating post-treatment signals leads the agent to select treatments that yield substantially higher outcome-driven rewards—even though this may reduce exact agreement with the doctor's initial plan. In other words, the agent moves beyond imitation to optimize for longer-term patient benefits as evidenced by the higher average reward. However, it must be noted that in the case of cancer treatments, the latter might not always align with overall treatment success. Additionally, the results need to be further reviewed and assessed by an oncologist to take the final call on a patient's treatment.

Chapter 6: Conclusion and Future Work

At its core, the environment design represents the patient's state through a comprehensive feature vector that includes demographic information, clinical history, and genetic markers. This state representation was crafted to capture all clinically relevant information while remaining computationally tractable. The core of the implementation revolves around the DQN architecture, tuned to handle the complexities of cancer treatment decisions. Experience replay is implemented to maintain a balance between different patient types and outcomes. The results of this approach demonstrate the potential of DRL in medical decision support systems by having good generalization capabilities. Furthermore, this approach could serve as a valuable tool for supporting clinical decision-making in oncology. While not intended to replace human expertise, the system can provide recommendations and help clinicians consider treatment options and therapy combinations.

Future Work

Although the project yields good results and succeeds in laying a solid foundation, there are several promising directions for future work. Firstly, the enhancement of reward function to incorporate more sophisticated temporal patterns in disease progression. A more nuanced approach would include, but would not be limited to, a continuous cycle of treatment responses along with the factors that could indicate treatment toxicity in patients. This would, in addition to the curation of the reward function, require longitudinal data. A more granular approach could be leveraged for the action space (which treats treatments as binary decisions) to include treatment dosages and timings. For this, a better-suited algorithm would be SAC (Soft Actor Critic) or TD3 (Twin Delayed DDPG).

While GAN-based synthetic data augmentation addressed data scarcity in this work, future iterations could explore transfer learning (TL) to reduce reliance on synthetic data and improve generalizability. Pre-training models on large, related oncology datasets (e.g., pan-cancer genomic repositories like TCGA or clinical outcome databases) could capture foundational biomarkers (e.g., TMB, MSI) and treatment-response patterns, enabling robust initialization for GEJ-specific fine-tuning. TL would leverage existing knowledge from well-studied cancers (e.g., gastric or esophageal) while avoiding risks of synthetic data biases such as implausible feature

correlations. For instance, models pre-trained on immunotherapy response data could better predict outcomes for GEJ patients with high TMB, even with limited training samples. Integrating TL with multi-modal data (genomic plus imaging) could further enhance personalized recommendations while minimizing computational overhead compared to GANs. This approach would prioritize biologically validated patterns over synthetic generation, aligning with clinical trust and scalability needs.

Additionally, other data modalities, such as imaging data (CT Scan, MRI, MRA, EEG), additional genetic markers, and daily patient health statistics could enhance the model's predictive capabilities. This can also be combined with IoT devices (specifically smartwatches) to monitor patient treatment response and alarm the caregivers in case of treatment toxicity. According to a paper published in BJC [27] reports and PubMed Central, treatments like chemotherapy and radiation therapy are associated with several life-threatening side effects. In fact, patients with recurrent advanced cancer (non-adenocarcinoma) who undergo chemotherapy show a deteriorated baseline status and are more likely to experience higher toxicity [26]. Knowing about it well in advance could potentially help with dosage adjustments sooner rather than later. Future improvements could also include expanding the knowledge base to cover more rare cancer types and recommending drugs [4] along with therapies wherever necessary.

Bibliography

[1] How Do Genetics Play a Role in the Development of Cancer. Virginia Cancer Specialists & Research Institute blog [Internet]. [cited 2025 Apr 24]. Available from: https://blog.virginiacancer.com/how-do-genetics-play-a-role-in-the-development-of-cancer

[2] SEER Cancer Stat Facts: Esophageal Cancer. National Cancer Institute, Bethesda, MD [Internet]. [cited 2025 Apr 24]. Available from: https://seer.cancer.gov/statfacts/html/esoph.html

[3] Liu M, Shen X, Pan W. Deep reinforcement learning for personalized treatment recommendation. Stat Med. 2022;41. doi:10.1002/sim.9491

[4] World Cancer Research Fund. Worldwide Cancer Data [Internet]. [cited 2025 Apr 24]. Available from: https://www.wcrf.org/preventing-cancer/cancer-statistics/worldwide-cancer-data/.

[5] Gudiol C, Albasanz-Puig A, Cuervo G, Carratalà J. Understanding and Managing Sepsis in Patients With Cancer in the Era of Antimicrobial Resistance. Front Med (Lausanne). 2021 Mar 31;8:636547. doi:10.3389/fmed.2021.636547.

[6] Yauney G, Shah P. Reinforcement Learning with Action-Derived Rewards for Chemotherapy and Clinical Trial Dosing Regimen Selection. In: Proceedings of the 3rd Machine Learning for Healthcare Conference, Proceedings Research. 2018;85:161–226. Available from: https://proceedings.mlr.press/v85/yauney18a.html.

[7] Wang Y, Gao W, Sun M, Feng B, Shen H, Zhu J, Chen X, Yu S. A filter-electrochemical microfluidic chip for multiple surface protein analysis of exosomes to detect and classify breast cancer. Biosens Bioelectron. 2023;239:115590. doi:10.1016/j.bios.2023.115590

[8] Friedl TWP, Fehm T, Müller V, et al. Prognosis of Patients With Early Breast Cancer Receiving 5 Years vs 2 Years of Adjuvant Bisphosphonate Treatment: A Phase 3 Randomized Clinical Trial. JAMA Oncol. 2021;7(8):1149–1157. doi:10.1001/jamaoncol.2021.1854

[9] Demetri GD, De Braud F, Drilon A, et al. Updated Integrated Analysis of the Efficacy and Safety of Entrectinib in Patients With NTRK Fusion-Positive Solid Tumors. Clin Cancer Res. 2022;28(7):1302–1312. doi:10.1158/1078-0432.CCR-21-3597

[10] Owen DR, Boonstra PS, Viglianti BL, Balter J, Schipper MJ, Jackson W, El Naqa I, Jolly S, Ten Haken R, Kong FM, Matuszak MM. Erratum to: Modeling Patient-Specific Dose-Function Response Using Perfusion SPECT/CT for Enhanced Characterization of Personalized Functional Changes. Int J Radiat Oncol Biol Phys. 2021;110(5):1552. doi:10.1016/j.ijrobp.2021.04.011.

[11] Gallagher K, Strobl MAR, Park DS, Spoendlin FC, Gatenby RA, Maini PK, Anderson ARA. Mathematical Model-Driven Deep Learning Enables Personalized Adaptive Therapy. Cancer Res. 2024 Jun 1;84(11):1929–1941. doi:10.1158/0008-5472.CAN-23-2040

[12] National Comprehensive Cancer Network. NCCN Guidelines for Colon Cancer [Internet]. [cited 2025 Apr 24]. Available from: https://www.nccn.org/guidelines/guidelines-detail?category=1&id=1428

[13] Di Nunzio GM, Sordoni A. Picturing Bayesian Classifiers: A Visual Data Mining Approach to Parameters Optimization. In: Zhao Y, Cen Y, editors. Data Mining Applications with R. Academic Press; 2014. p. 35–61. doi:10.1016/B978-0-12-411511-8.00002-5.

[14] Jain A. Elastic Net Regression: Combined Features of L1 and L2 Regularization. Medium [Internet]. [cited 2025 Mar 16]. Available from: https://medium.com/@abhishekjainindore24/elastic-net-regression-combined-features-of-l1-and-l2-regularization-6181a660c3a5.

[15] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. Cambridge, MA: A Bradford Book; 2018.

[16] McKinney WA. Pandas: A Foundational Python Library for Data Analysis and Statistics. 2011 [Internet]. Available from: https://api.semanticscholar.org/CorpusID:61539023.

[17] Ketkar N. Introduction to PyTorch. In: Deep Learning with Python. Apress; 2017. p. 241–264. doi:10.1007/978-1-4842-2766-4_12.

[18] cBioPortal for Cancer Genomics [Internet]. [cited 2025 Apr 24]. Available from: https://www.cbioportal.org/

[19] cBioPortal for Cancer Genomics. Clinical Data: STAD TCGA GDC [Internet]. [cited 2025 Apr 24]. Available from: https://www.cbioportal.org/study/clinicalData?id=egc_msk_2017.

[20] Google Developers. GAN Structure [Internet]. Google. [cited 2025 Mar 16]. Available from: https://developers.google.com/machine-learning/gan/gan_structure.

[21] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Advances in Neural Information Processing Systems. 2014;27.

[22] Omolola S, Kehinde SO, Adeoluwa AA, Gabriel TF, Joshua AO. Chemotherapy induced neutropenia and febrile neutropenia among breast cancer patients in a tertiary hospital in Nigeria. ecancermedicalscience. 2021;15:1188.

[23] Cancer Research UK. Sepsis and Infections in Cancer [Internet]. [cited 2025 Apr 24]. Available from: https://www.cancerresearchuk.org/about-cancer/coping/physically/sepsis-infection-cancer.

[24] Chan TA, Yarchoan M, Jaffee E, Swanton C, Quezada SA, Stenzinger A, Peters S, et al. Ann Oncol. 2019;30(1):44–56. doi:10.1093/annonc/mdy495.

[25] Le DT, Durham JN, Smith KN, et al. Mismatch repair deficiency predicts response of solid tumors to PD-1 blockade. Science. 2017;357(6349):409–413.

[26] Juthani R, Punatar S, Mittra I. New light on chemotherapy toxicity and its prevention. BJC Rep. 2024;2:41. doi:10.1038/s44276-024-00064-8.

[27] Lee EM, Jiménez-Fonseca P, Galán-Moral R, Coca-Membribes S, Fernández-Montes A, Sorribes E, García-Torralba E, Puntí-Brun L, Gil-Raga M, Cano-Cano J, Calderon C. Toxicities and Quality of Life during Cancer Treatment in Advanced Solid Tumors. Curr Oncol. 2023 Oct 19;30(10):9205–9216. doi:10.3390/curroncol30100665.