PREDICTIVE MODELING OF BRAIN AND LUNGS CANCER

A PROJECT REPORT SUBMITTED TO

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

AWARD OF THE DEGREE OF

MASTER OF SCIENCE IN APPLIED DATA SCIENCE

BY

SHRADDHA SUMAN GURU

REG NO. RA2332014010172

UNDER THE GUIDANCE OF

DR. BALAMURUGAN S, MCA, M.Phil, Ph.D

DEPARTMENT OF COMPUTER APPLICATIONS

FACULTY OF SCIENCE AND HUMANITIES

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur – 603 203

Chennai, Tamil Nadu

April – 2025

BONAFIDE CERTIFICATE

This is to certify that the project report titled "**PREDICTIVE MODELING OF BRAIN AND LUNGS CANCER**" is a bonafide work carried out by **SHRADDHA SUMAN GURU** (**RA2332014010172**) under my supervision for the award of the Degree of Master of Applied Data Science. To my knowledge the work reported here in is the original work done by the student.

Dr. Balamurugan S
Assistant Professor,
Department of Computer Applications
(GUIDE)

Dr. Jayashree R
Associate Professor and Head,
Department Of Computer Application

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION OF ASSOCIATION OF RESEARCH PROJECT WITH SUSTINABLE DEVELOPMENT GOALS

This is to certify that the research project entitled "**PREDICTIVE MODELING OF BRAIN AND LUNGS CANCER**" carried out by **SHRADDHA SUMAN GURU** (RA2332014010172) under the supervision of **Dr. Balamurugan S,** Assistant Professor of Department of Computer Applications in partial fulfilment of the requirement for the award of Post Graduation program has been significantly or potentially associated with SDG Goal **3 (THREE)** titled **GOOD HEALTH AND WELL BEING**.

This study has clearly shown the extent to which its goals and objectives have been met in of filling the research gaps, identifying needs, resolving problems, and developing innovative solutions locally for achieving the above-mentioned SDG on a National and/or on an international level.

SIGNATURE OF THE STUDENT **GUIDE**

HEAD OF THE DEPARTMENT

ACKNOWLEDGEMENT

With profound gratitude to the ALMIGHTY, I take this chance to thank the people who helped me to complete this project.

I take this as a right opportunity to say THANKS to my parents who are there to stand with me always with the words "YOU CAN".

I wish to express my sincere gratitude to **Dr. T. R. Paarivendhar**, Chancellor, SRM Institute of Science and Technology, who gave me the platform to establish me to reach greater heights.

I am thanful to **Prof. A. Vinay Kumar,** Pro Vice-Chancellor (SBL) and **Dr. A. Duraisamy,** Dean, Faculty of Science and Humanities, SRM Institute of Science and Technology for their unwavering support throughout my project.

I earnestly thank **Dr. S. Albert Antony Raj**, Professor and Deputy Dean, College of Sciences, Faculty of Science and Humanities who always encourage me to do novel things.

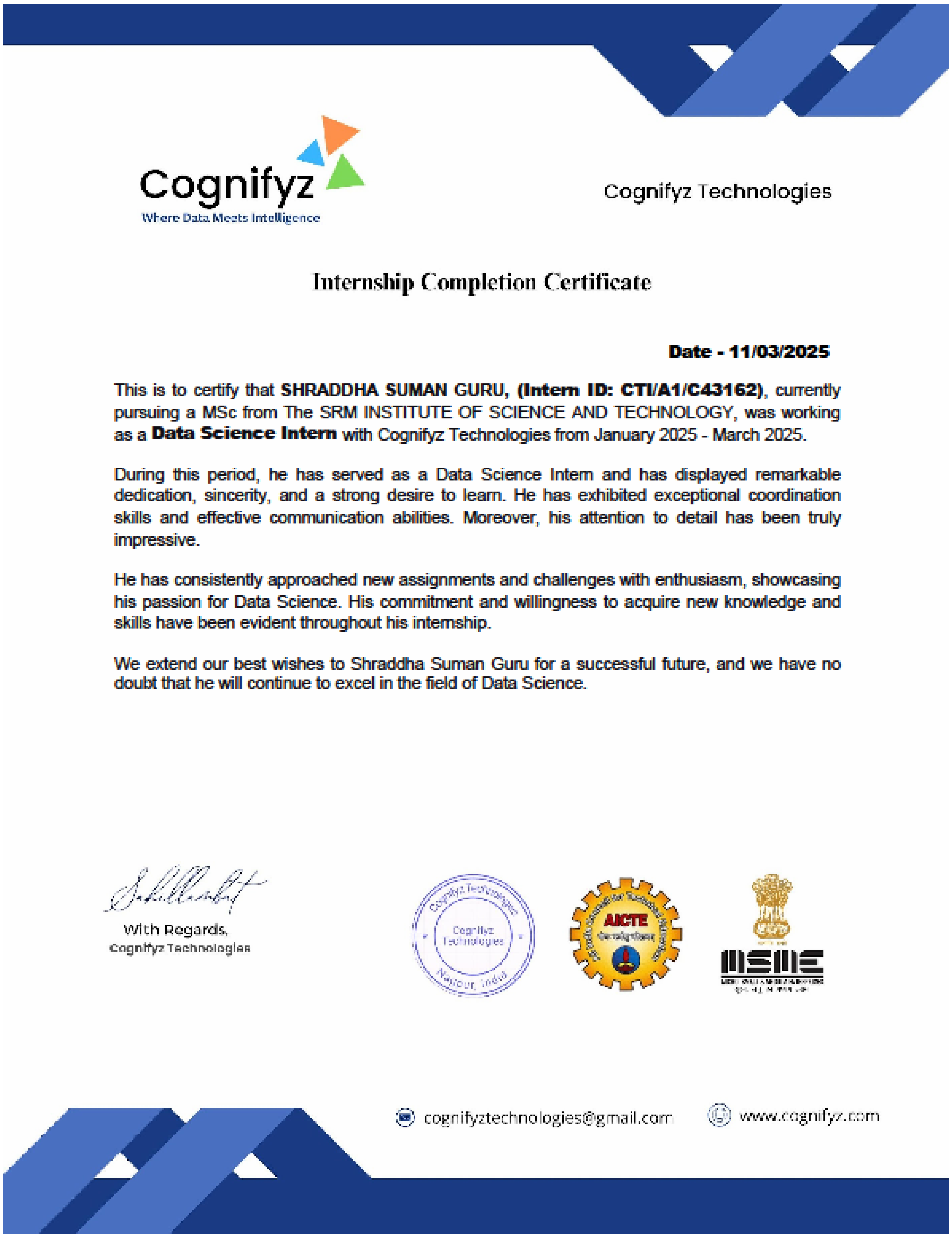
I express my sincere thanks to **Dr. R. Jayashree**, **Ph.D.,** Associate Professor and Head, Department of Computer Applications, Faculty of Science and Humanities for her valuable guidance and support to execute all incline in learning.

It is my delight to thank my project guide **Dr. S. Balamurugan,** Assistant Professor, Department of Computer Applications for his help, support, encouragement, suggestions and guidance throughout the development phases of the project.

I convey my gratitude to all the faculty members of the department who extended their support through valuable comments and suggestions during the reviews.

My gratitude to friends and people who are known and unknown to me who helped in carrying out this project work a successful one.

SHRADDHA SUMAN GURU



Cognifyz
Where Data Meets Intelligence

Cognifyz Technologies

Internship Completion Certificate

Date - 11/03/2025

This is to certify that **SHRADDHA SUMAN GURU, (Intern ID: CTI/A1/C43162)**, currently pursuing a MSc from The SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, was working as a **Data Science Intern** with Cognifyz Technologies from January 2025 - March 2025.

During this period, he has served as a Data Science Intern and has displayed remarkable dedication, sincerity, and a strong desire to learn. He has exhibited exceptional coordination skills and effective communication abilities. Moreover, his attention to detail has been truly impressive.

He has consistently approached new assignments and challenges with enthusiasm, showcasing his passion for Data Science. His commitment and willingness to acquire new knowledge and skills have been evident throughout his internship.

We extend our best wishes to Shraddha Suman Guru for a successful future, and we have no doubt that he will continue to excel in the field of Data Science.

[illegible]

With Regards,
Cognifyz Technologies

Cognifyz Technologies
Cognifyz Technologies
Nagpur, India

AICTE

MSME

cognifyztechnologies@gmail.com www.cognifyz.com

0% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

0 Not Cited or Quoted 0%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

0%	Internet sources
0%	Publications
0%	Submitted works (Student Papers)

TABLE OF CONTENTS

ABSTRACT

Around 5.69 percent of the population is diagnosed with lungs cancer at some point during their lifetime. Brain metastasis and Lung metastasis in cancer reports about the spread of cancer to other organs of the body and in most cases linked with low rates of survival and quality of life. This emphasizes the need of early detection of high-risk patients. Brain cancer is an uncommon form of cancer found rarely in patients. Hence chances of the patient being alive given the patient suffers from this uncommon kind of cancer aka brain cancer highly depends on the type of tumor. Survival chances falls if tumor extends beyond the original site. Many patients were diagnosed from 2018 to 2021 and their data is recorded in the SEER (Surveillance, Epidemiology, and End Results) Stat database. Three Machine Learning Ensemble techniques are used for the **SEER Cancer data**, with cross-validation accuracy scores considered in the selection process. The **Gradient Boosting Classifier** (GBC) outperformed the other models and was selected for further analysis. Precision, recall and F1-score were calculated to assess model performance. Insights from GBC model helped identify risk factors associated with lung and brain cancer diagnoses. Feature importance scores from the **Random Forest** (RF) algorithm revealed the most significant predictors, providing over 99 percent predictive accuracy for the GBC model. **Exploratory Data Analysis** (EDA) was used to gain additional insights into the dataset. This prediction can aid healthcare provider to take well-informed decisions for the patients which can effectively slow down cancer progression. Also, potentially enhance survival chances by determining vital status (alive or dead).

LIST OF FIGURES

LIST OF SCREENSHOTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Predictive modeling has become a powerful tool in the field of healthcare, particularly in cancer diagnosis and prognosis. Lung and brain cancers are among the leading causes of death globally, and early detection and accurate prediction of outcomes are crucial in improving survival rates and patient quality of life. In this study, we explore the use of **Gradient Boosting**, a powerful ensemble learning algorithm, to predict the outcomes of **brain and lung cancer** patients.

This approach incorporates various techniques such as **Data Visualization** in leveraging advanced visualization tools to explore and interpret the patterns within the data, facilitating better understanding and insight into key features related to cancer diagnosis, **Feature Engineering** in transforming raw data into meaningful features that improve model accuracy and robustness. Feature selection and extraction are critical in identifying the most significant predictors for cancer prognosis. Finally **Dimensionality Reduction** techniques like computing feature importance using Random Forest model to reduce the number of variables, improving the efficiency and predictive power of the model by focusing on the most relevant features while reducing noise and computational complexity.

The **Gradient Boosting Classifier (GBC)** model was selected as the final model due to its superior performance in terms of cross validation accuracy score. This model is trained on the collected data and tuned to identify the key factors influencing cancer outcomes, ultimately offering a tool for clinicians to predict patient prognosis and make informed decisions.

By employing these techniques, this study aims to enhance early detection, improve treatment planning, and contribute to the ongoing effort to combat brain and lung cancer through data driven-insights.

OBJECTIVE

This project aims to apply predictive analytics and risk assessment models and data visualisation techniques to study the cancer scenario in US, with a focus on:

- **To enhance Decision Making in Clinical Settings**: Provide healthcare professionals with a tool that supports decision-making by predicting patient outcomes, potentially improving treatment planning and prognosis for brain and lung cancer patients.
- **To Identify Key Risk Factors for Brain and Lung Cancer**: Analyze patient data (e.g., age, gender, smoking history, tumor type, genetic factors) to identify the most significant factors associated with the development or spread of brain and lung cancer.
- **Developing a Predictive Classification Model**: Using ensemble ML techniques to categorize individuals based on their vital status.

SCOPE

- Implementing classification models (e.g., Logistic Regression, Decision Trees, Support Vector Machines, Neural Networks) to identify individuals at high risk of mortality.
- Assessing how cancer programs across US influenced infection and fatality rates.
- Providing insights to help governments and health organizations

improve cancer preparedness.

- Identifying trends and its role in preventing severe illness and hospitalization.
- Providing early warnings for policymakers and healthcare institutions to allocate resources effectively.

CHAPTER 2

SYSTEM ANALYSIS

EXISTING SYSTEM

The description is about the previous researches about lungs and brain cancer and its health risks, economic impact and social consequences. Various studies have explored machine learning models, statistical approaches and epidemiological theories to assess risks and predict trends.

This study evaluates the conceptual designs to identify the most suitable approach for the proposed project. The selected design incorporates software implementation along with potential hardware implementation.

PROPOSED SYSTEM

According to (Wang et al., 2021)**,** predictive analytics is performed by processing historical and real-time data through machine learning algorithms and risk assessment frameworks**.** The system is designed to enhance accuracy by preprocessing raw data, reducing inconsistencies, and applying feature engineering techniques to improve predictive accuracy. Additionally, the system can be adapted to various applications, such as public health planning, and business risk management, providing policymakers and stakeholders with data-driven insights for informed decision-making.

SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENT

OS	:	Windows 7 or Higher,
		Mac OS 10.12.6 (Sierra) or later
Application	:	Python 3.6–3.8,

pip 19.0 or later

HARDWARE REQUIREMENT

The performance of the cancer dataset is significantly influenced by the underlying hardware usedduring development and deployment. Below are the recommended hardware specifications necessary for running the dataset effectively:

1. **CPU Specifications**
   - **Processor**: A multi-core processor (e.g., Intel i5 or i7, AMD Ryzen 5 or 7) is recommended to handle multiple threads during training and inference processes.
   - **Cores**: At least 4 to 8 cores to ensure smooth multi-tasking and efficient model training.

2. **Memory (RAM)**
   - **Minimum Requirement**: 8 GB of RAM for basic operations and small-scale model training.
   - **Recommended**: 16 GB or more to handle larger datasets, more complex models and multiple applications running concurrently.

3. **Storage**
   - **SSD (Solid State Drive)**: A minimum of 256 GB SSD for faster data access and improved performance during model training and inference.
   - **Data Storage**: Sufficient storage space for datasets, model weights, and logs. As dataset grows, consider

expanding storage capacity to atleast 1TB.

4. **Graphics Processing Unit (GPU)**
    - **GPU**: A dedicated GPU (e.g., NVIDIA GTX 1060, RTX 2060 or higher) is essential for accelerating the training of machine learning models.
    - **VRAM**: At least 6 GB of VRAM to accommodate larger models and datasets during training.

5. **Operating System**
    - **OS**: The software stack is compatible with Windows, mac OS and Linux distributions. Linux is often preferred for server deployment due to its stability and performance in handling machine learning workloads.

6. **Network Requirements**
    - **Internet Connectivity**: Reliable internet access is necessary for downloading libraries, accessing datasets and facilitating cloud deployments if needed.

CHAPTER 3

SOFTWARE DESCRIPTION

PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as the other languages use punctuations. It has fewer syntactical constructions than other languages.

Python is a powerful programming language ideal for scripting and rapid application development. It is used in web development (like: Django and Bottle), scientific and mathematical computing (Orange, Sym Py, Num Py) to desktop graphical user Interfaces (Pygame, Panda3D).

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

The Python interpreter is developed under an OSI-approved open source license, making it free to install, use, and distribute, even for commercial purposes. Aversion of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows, mac OS, smartphones and tablets, and probably anything else you ever heard of. Aversion even exists for the half dozen people remaining who use OS/2.

Reasons for using PYTHON

- **Easy-to-learn** − Python has few keywords, simple structure and a clearly defined syntax. This allows a student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of UNIX.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

STATS MODELS

Statsmodels is a comprehensive Python library for statistical modeling and data analysis, built on top of NumPy, SciPy, and pandas. It is widely used in fields like economics, social sciences, and biology, where statistical tests and modeling are crucial. Statsmodels offers a wide range of models for estimation and testing, with a focus on linear regression, generalized linear models, time-series analysis, and more. It provides users with powerful tools for analyzing data and testing hypotheses, making it invaluable in both academic research and practical applications. It is a powerful library for users who require statistical analysis, model fitting, and hypothesis testing in their data science workflows. Whether you're working with simple linear regressions or more complex time-series models, Statsmodels provides the tools you need to conduct rigorous statistical analyses.

Features

Statsmodels comes with an array of features for statistical analysis which include:

- Implements a variety of regression techniques, including ordinary least squares (OLS), generalized least squares (GLS), and robust linear models.
- Supports various distributions and link functions, enabling flexible modeling for different types of data.
- Includes tools for working with time-series data, such as ARIMA models, seasonal decomposition, and autocorrelation tests.
- Offers various statistical tests, including t-tests, chi-square tests, and F-tests, for evaluating the significance of model parameters.
- Provides a set of visualization tools for exploring relationships between variables and the results of statistical tests.

- Model Diagnostics: Includes diagnostic tests for evaluating model fit and assumptions, such as residual plots, influence measures, and multicollinearity checks.

Benefits

- Comprehensive Statistical Support: It provides a wide range of statistical models and tests, making it suitable for a variety of research and analysis scenarios.
- In-depth Statistical Output: Statsmodels provides rich summary tables with detailed statistical information (e.g., p-values, confidence intervals, AIC/BIC), making it ideal for in-depth analysis.
- Flexible and Extensible: The library is highly customizable and allows users to implement their own models and estimators if necessary.
- Integration with pandas and NumPy: Statsmodels integrates seamlessly with pandas for data manipulation and NumPy for numerical computation, making it easy to use with existing workflows.
- Open-Source and Well-Documented: As an open-source package, Statsmodels is well-supported by the community, with thorough documentation and examples.

PANDAS

Pandas is a Python library for data manipulation and analysis, widely used for handling structured data in machine learning projects. It provides powerful tools for loading, processing, and analyzing data efficiently, making it an essential part of any data science workflow. Pandas allows users to work with datasets in tabular formats through its primary data structures: Data Frame and Series. These structures enable efficient data organization, transformation, and analysis.

This library is particularly useful for data preprocessing tasks such as handling missing values, filtering, grouping, and merging datasets. It also integrates well with other Python libraries such as Num Py, Matplotlib, and Scikit-learn, enabling a smooth workflow from data preparation to model training.

Pandas is a cornerstone library for data analysis in Python. Its versatility, ease of use, and wide range of features make it indispensable for anyone working with structured data. Whether you're cleaning data, conducting exploratory analysis, or preparing datasets for machine learning, Pandas provides the necessary tools to handle and analyze data efficiently and effectively.

Features

Pandas includes several capabilities that make it a highly effective tool for working with structured data:

- Provides Data Frame and Series objects for efficient data representation and manipulation.
- Supports operations such as filtering, sorting, and aggregating data with minimal code.
- Enables handling of missing values through functions like fillna and dropna.
- Allows reading and writing data from multiple file formats, including Comma Separated Values (CSV), Excel, JSON, and SQL.
- Integrates seamlessly with visualization libraries for creating data plots and summaries.

Benefits

Pandas is widely used for its ease of use and efficiency in data analysis. Some key benefits include:

- Simplifies data manipulation, making it easier to preprocess datasets for machine learning.
- Provides fast performance using optimized functions built on Num Py and Cython.
- Supports a wide range of data sources, making it flexible for different applications.
- Facilitates easy data visualization with built-in plotting functions.
- Works well with other Python libraries, creating a comprehensive data analysis ecosystem.

SCIKIT-LEARN

Scikit-learn, commonly known as sklearn, is a machine learning library in Python that provides simple and efficient tools for building predictive models. It is built on top of NumPy, Sci Py, and Matplotlib, offering a comprehensive set of machine learning algorithms for tasks such as classification, regression, clustering, and dimensionality reduction.

The library is widely used in both academia and industry due to its ease of use, efficiency, and extensive documentation. It provides utilities for data preprocessing, model selection, and evaluation, making it a valuable tool for beginners and experienced practitioners alike.

Scikit-learn is an essential tool for machine learning practitioners due to its comprehensive set of algorithms, ease of use, and integration with the broader Python ecosystem. Whether you're building a predictive model, performing data analysis, or experimenting with machine learning algorithms, scikit-learn provides a robust and user-friendly framework to meet your needs.

Features

Scikit-learn includes a wide range of functionalities that support the development of machine learning models:

- Provides a variety of supervised and unsupervised learning algorithms, including decision trees, support vector machines, and k-means clustering.
- Supports cross-validation techniques for assessing model performance.
- Includes feature selection and dimensionality reduction techniques such as Principal Component Analysis (PCA).
- Offers built-in tools for data preprocessing, including scaling, encoding, and imputation.
- Allows the creation of machine learning pipelines for automated model training workflows.

Benefits

Scikit-learn is known for its simplicity and efficiency, offering several benefits:

- Provides an easy-to-use API that allows quick experimentation with machine learning models.
- Offers a wide selection of algorithms suitable for various machine learning tasks.
- Optimized for performance using Num Py and Sci Py, ensuring efficient computations.
- Includes built-in tools for feature engineering and data transformation.
- Has strong community support and well-maintained documentation, making it accessible for learners and working professionals.

NUMPY, SCIPY and MATPLOTLIB

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multi dimensional array objects and a collection of routines for processing of array. Num Py is often used along with packages like **Sci Py** (Scientific Python) and **Matplotlib** (plotting library).This combination is widely used as a replacement for Mat Lab, a popular platform for technical computing. However, Python alternative to Mat Lab is now seen as a more complete programming language. It is open source, which is an added advantage of NumPy.

Using Num Py, a developer can perform the following operations−

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. Num Py has in-built functions for linear algebra and random number generation.
- Numpy is the most useful library for Data Science to perform basic calculations.
- Numpy contains nothing but array data type which performs the most basic operation like sorting, shaping, indexing, etc.

SciPy, a scientific library for Python is an open source, BSD-licensed library for mathematics, science and engineering. The Sci Py library depends on Num Py, which provides convenient and fast N-dimensional array manipulation.The main reason for building the Sci Py library is that, it should work with Num Py arrays. It provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization.

- Sci Py contains varieties of subpackages which help to solve the most common issue related to Scientific Computation.

- Sci Py package in Python is the most used Scientific library only second to GNU Scientific Library for C/C++ or Matlab's.
- Easy to use and understand as well as fast computational power.
- It can operate on an array of Num Py library.
- Sci Py is built on top of the Num Py
- Sci Py module in Python is a fully-featured version of Linear Algebra while Numpy contains only a few features.

Matplotlib is a Python plotting package that makes it simple to create two-dimensional plots from data stored in a variety of data structures including lists, numpy arrays and pandas data Frames. Matplotlib uses an object oriented approach to plotting. This means that plots can be built step-by-step by adding new elements to the plot.

There are two primary objects associated with a matplotlib plot:

- Figure object: The overall figure space that can contain one or more plots.
- Axis objects: The individual plots that are rendered with in the figure.

You can think of the figure object as your plot canvas.You can think about the axis object as an individual plot.

Matplotlib is one of the most popular Python packages used for data visualization. It is a crossplatform libraryfor making2D plots from data in arrays. Matplotlib iswritten in Python and makes use of Num Py, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as Py Qt, Wx Pythonot Tkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers.

JUPYTER NOTEBOOK

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.

IPython notebook was developed by Fernando Perez as a web based front end to IPython kernel. As an effort to make an integrated interactive computing environment for multiple languages, notebook project was shifted under Project Jupyter providing frontend for programming environments Juila and R in addition to Python.

A notebook document consists of rich text elements with HTML formatted text, figures, mathematical equations etc. The notebook is also an executable document consisting of code blocks in Python or other supporting languages. You can interact with Jupyter Notebook using your web browser. The notebook program creates a “web server” locally on your computer that you can then connect to. On Windows you can find a launcher for Jupyter Notebook under Anaconda in the Start menu.

On Linux or OS X, you can start Jupyter Notebook from the command line. First open a terminal window, used to navigate to the directory where you want to store your Python files and notebook document files.

One major feature of the Jupyter notebook is the ability to displayplots that are the output of running code cells.The IPython kernel is designed to work seamlessly with plotting library to provide this functionality. Specific plotting library integration is a feature of the kernel.

- In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.

- The ability to execute code from the browser, with the results of computations attached to the code which generated them.

- Displaying the result of computation using rich media representations, such as HTML, La Te X, PNG, SVG, etc. For example, publication-quality figures rendered by the matplotlib library can be included inline.

- In-browser editing for rich text using the Markdown mark up language, which can provide commentary for the code, is not limited to plain text.

- The ability to easily include mathematical notation within mark downcells using La Te X, and rendered natively by Math Jax.

CHAPTER 4

SYSTEM DESIGN

DATA FLOW DIAGRAM

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. UML is a very important part of developing objects oriented software and the software development process. UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

The UML's five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML's behavioral diagrams are roughly organized around the major ways which can model the dynamics of a system. Behavioral diagrams consists of Usecase Diagram, Sequence Diagram, Collaboration Diagram, Statechart Diagram and Activity Diagram.

It usually begins with a context diagram as level 0 of the DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower-level functions decomposed from the major functions of the system.This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to levels 3, 4 and soon is possible but anything beyond level 3 is not very common. Please bear in mind that the level of detail for decomposing a particular function depending on the complexity that function.

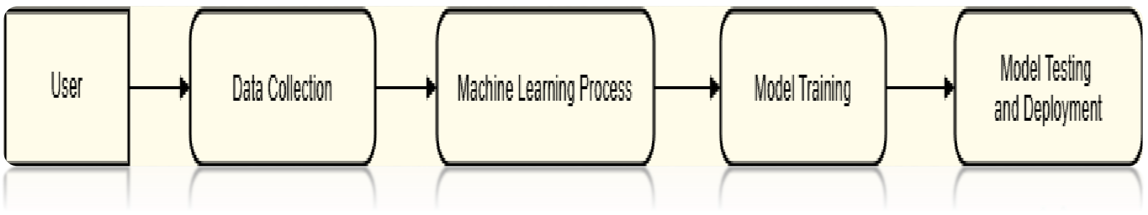


Fig 4.1: 0th level DFD

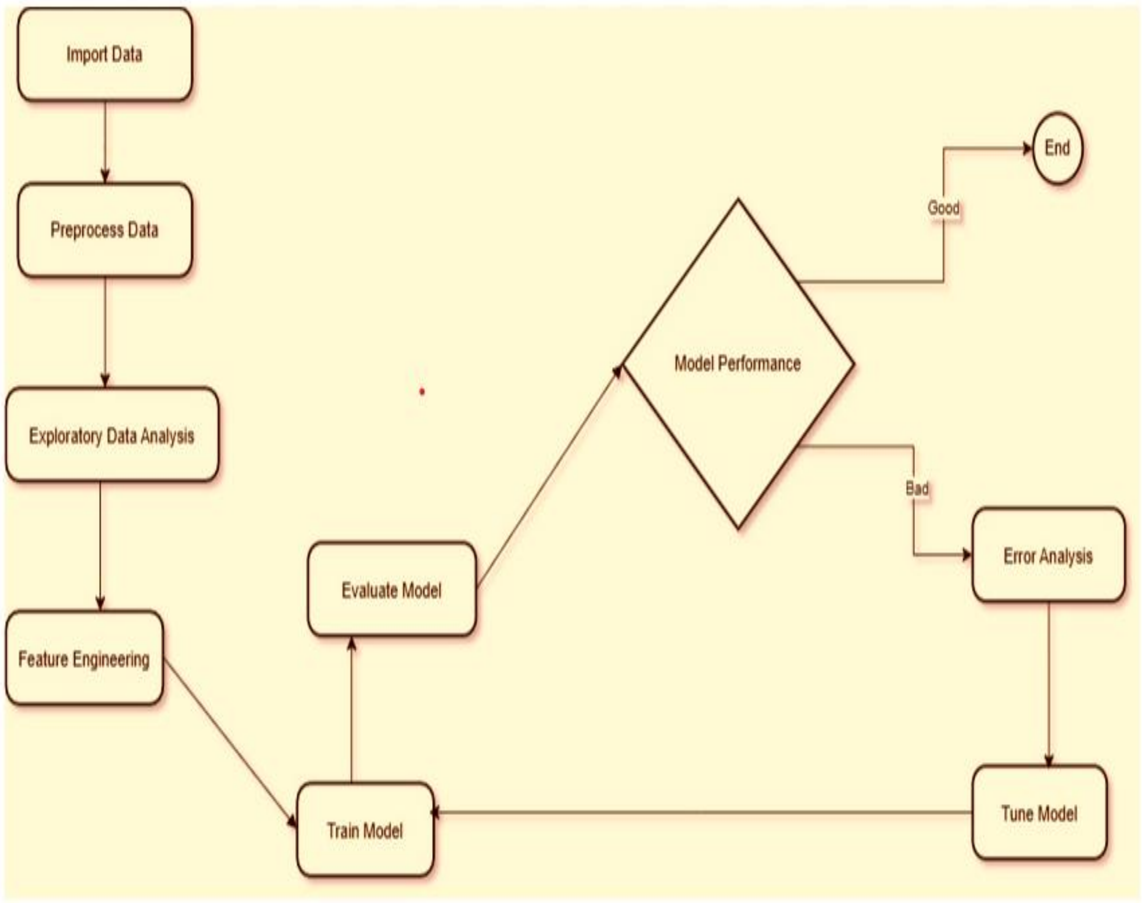


Fig 4.2: 1st level DFD

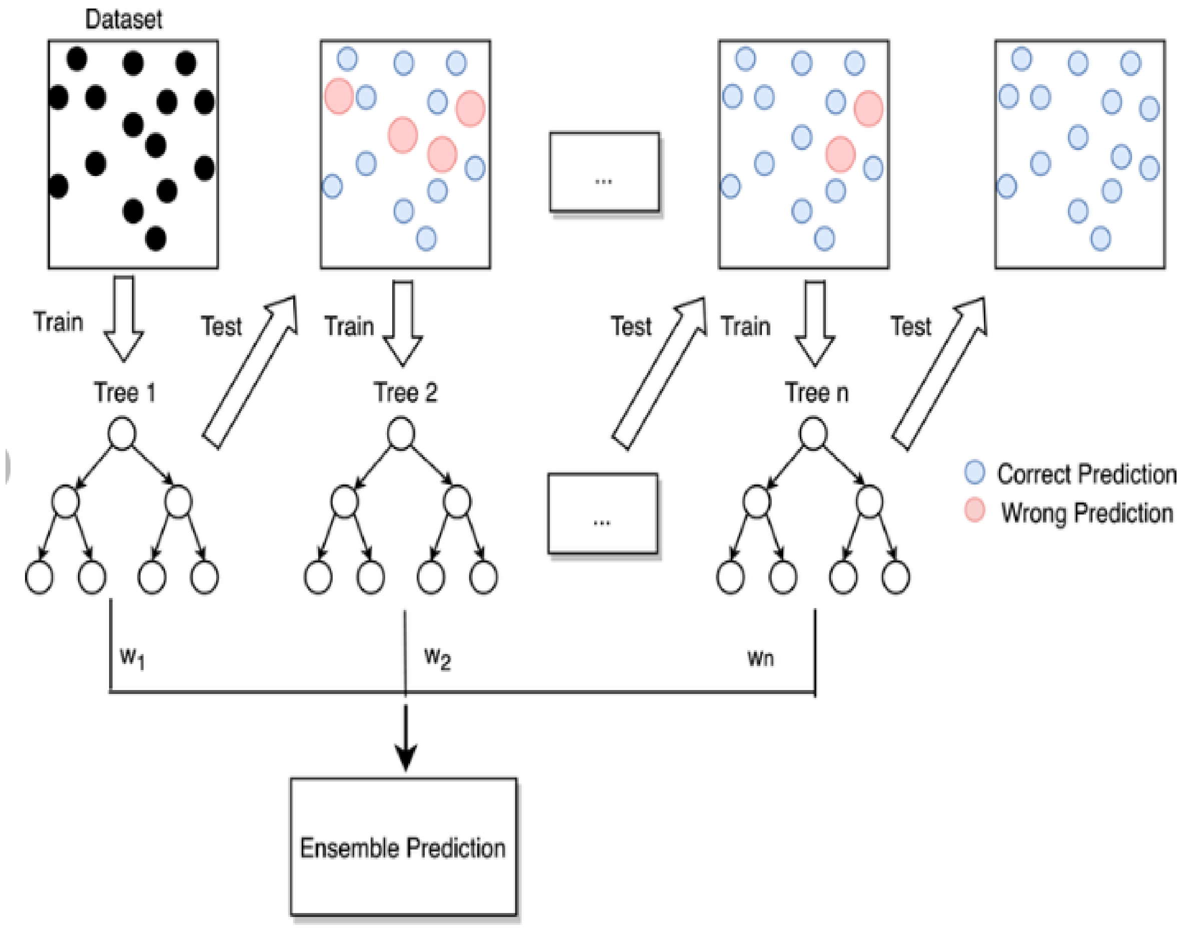


Fig 4.3: 2nd level DFD

USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. A Use Case Diagram is a graphical representation of interactions within a system. In the case of Brain and Lung Cancer Classification, the system aims to analyze medical imaging and patient data to provide an accurate diagnosis using Gradient Boosting Classifier.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference or performing a consumer service-oriented task. The system processes MRI, CT scans, and patient medical records to detect abnormalities in brain and lung tissues. By extracting crucial features like tumor size, shape, and texture, the classifier predicts whether a patient has lung or brain cancer. A use case diagram contains four components.

- Creating an Integral Image, Gradient Boost Training Classifiers.
- Guarantee that all independent paths have been executed.
- Execute internal data structures to ensure their validity
- Execute all logical decisions on their true and false sides.

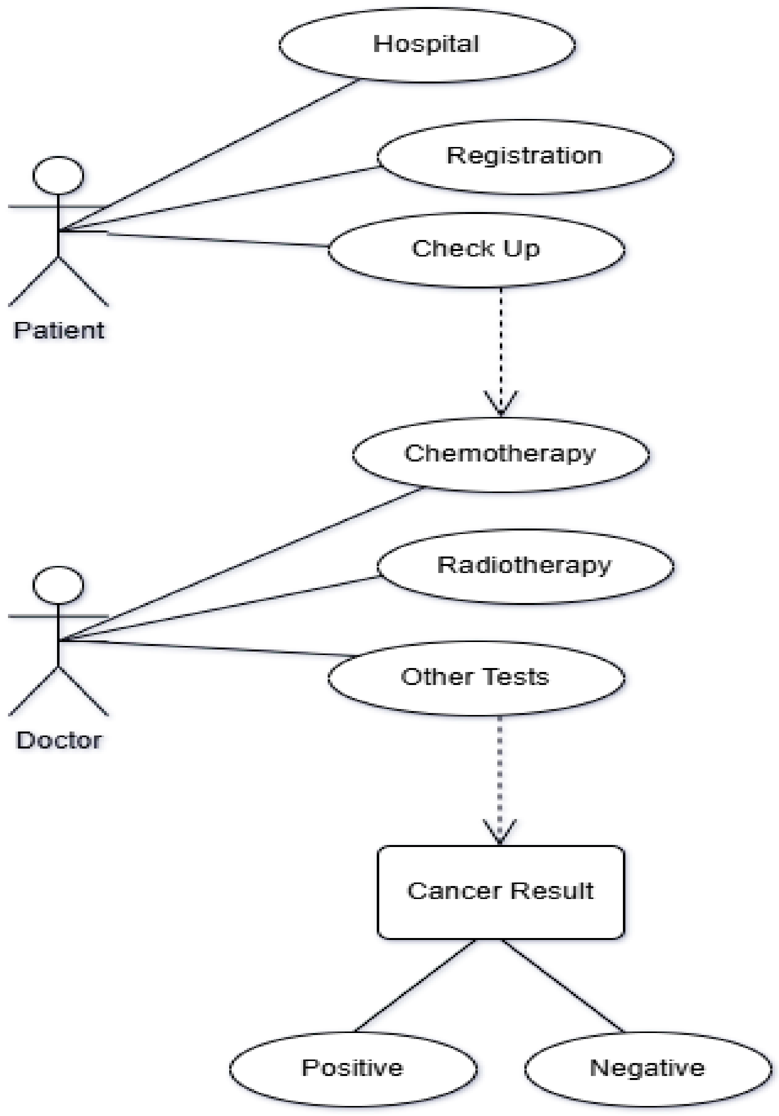


Fig 4.4: Use Case diagram

CHAPTER 5

MODULE ANALYSIS

IMPORT DATASET

The Dataset has over 1.9 Lakh records with 17 attributes. The target or the dependent column consists of the vital status of the patient that is alive or deceased based on different medical conditions and the age of the patients which plays a major role of determining if the disease will spread in the body of the patient. The dataset is in spreadsheet format and hence imported using the pandas library in python.

DATA PRE-PROCESSING

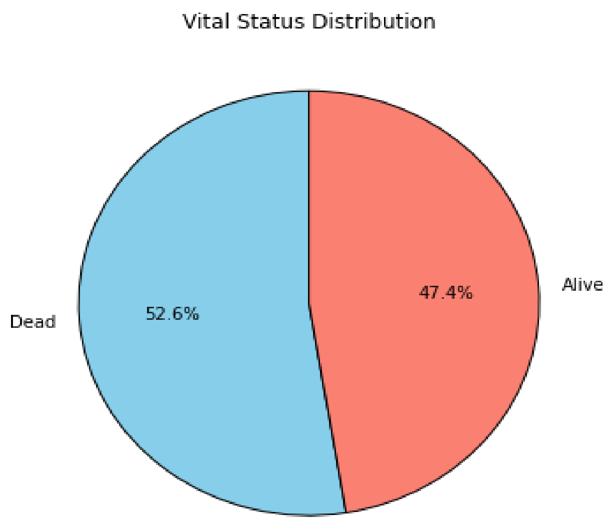
Data preprocessing is an essential step in preparing a cancer dataset for analysis. It involves cleaning, transforming, and organizing the data to improve its quality and ensure it is suitable for statistical analysis or machine learning models.

Data cleaning follows, where errors, inconsistencies, and duplicate records are identified and corrected. For example, categorical variables such as cancer stages or treatment types should have consistent labels to avoid discrepancies. If different formats exist for the same category, standardizing them ensures uniformity. The process begins with handling missing values, which are common in medical datasets. Missing values can occur due to incomplete patient records, errors in data collection, or inconsistencies in reporting. Various methods can be used to address missing data, such as removing records with too many gaps, filling in missing values using the mean or median for numerical data.

Data transformation is another crucial step. Numerical features such as tumor size may need to be normalized or scaled to ensure they are on a comparable scale, preventing certain variables from dominating the analysis. Categorical variables, such as radiation type or primary site, may need to be encoded into numerical values using techniques like one-hot encoding or label encoding.

Feature engineering enhances the dataset by creating meaningful new features or modifying existing ones. This can involve grouping tumor sizes into categories, calculating time intervals such as the duration between diagnosis and treatment, or deriving survival rate classifications from raw survival months data. These additional features can improve the predictive power of a model.

We also checked the imbalance in the target column which shows that the target labels are not imbalanced as they both are close to 50 percent. If they might have been imbalanced we would have used the IMBLearn which stands for imbalanced learn. Imbalanced Learn is best for normalizing the imbalanced labels in a datatset.



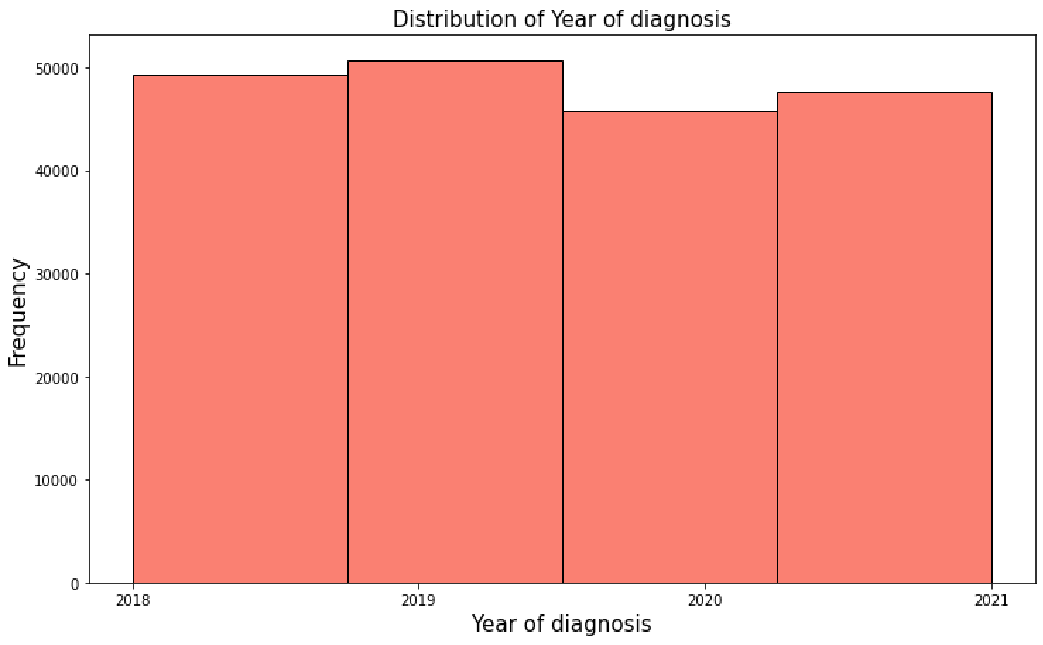
5.1: Distribution of Target Variable Labels

EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis, or EDA, is the process of analyzing and summarizing a dataset to understand its main characteristics before applying machine learning models or statistical methods. It helps in identifying patterns, spotting anomalies, and gaining insights into the relationships between variables.

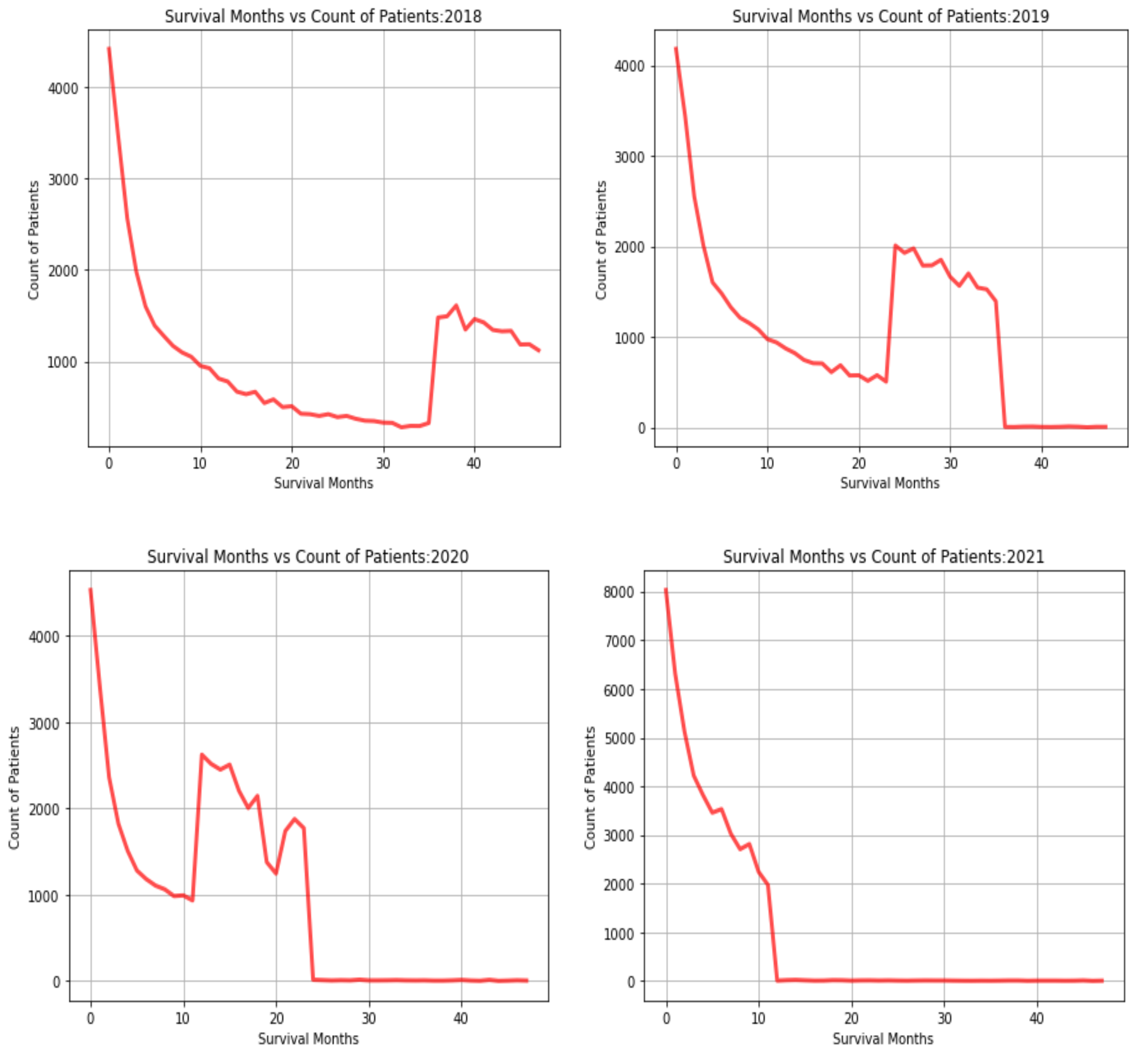
Visualizing the distribution of data is important to detect skewness or unusual patterns. Histograms and density plots are commonly used to examine the spread of numerical variables like tumor size or survival months. Box plots help in identifying outliers, while bar charts are useful for categorical variables like primary site or treatment type.

Another aspect of EDA is examining relationships between different variables. Correlation analysis is used to measure the strength of relationships between numerical features. A heatmap can visually represent these correlations, showing how factors like tumor size, metastasis, and survival time interact with each other. For categorical variables, cross-tabulations and grouped bar charts can reveal trends in treatment patterns or survival outcomes.



5.2: Distribution of Year of Diagnosis

Insights: This histogram shows the distribution of year of diagnosis of the patients in the dataset and provides insights into the diagnosis year distribution, which can be helpful in analyzing the number of distribution of patients. It may reveal information about the cohort of patients in the dataset. Almost equal numbers of people were diagnosed each year from 2018 to 2021.



5.3: Survival Months vs Count of Patients each year

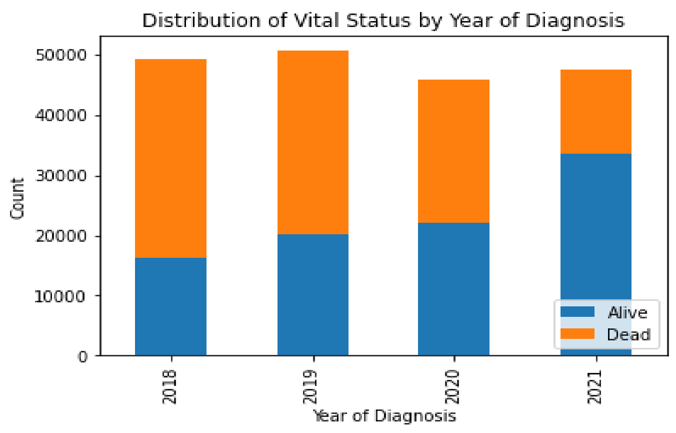
Insights: We observe an exponential downward graph of number of patients with respect to their survival months (0-10) in every plot.

In the year 2018, the count of patients for more than 34 months survival has a positive growth.

In the year 2019, the count of patients for more than 23 months survival has a positive growth.

In the year 2020, the count of patients for more than 11 months survival has a positive growth.

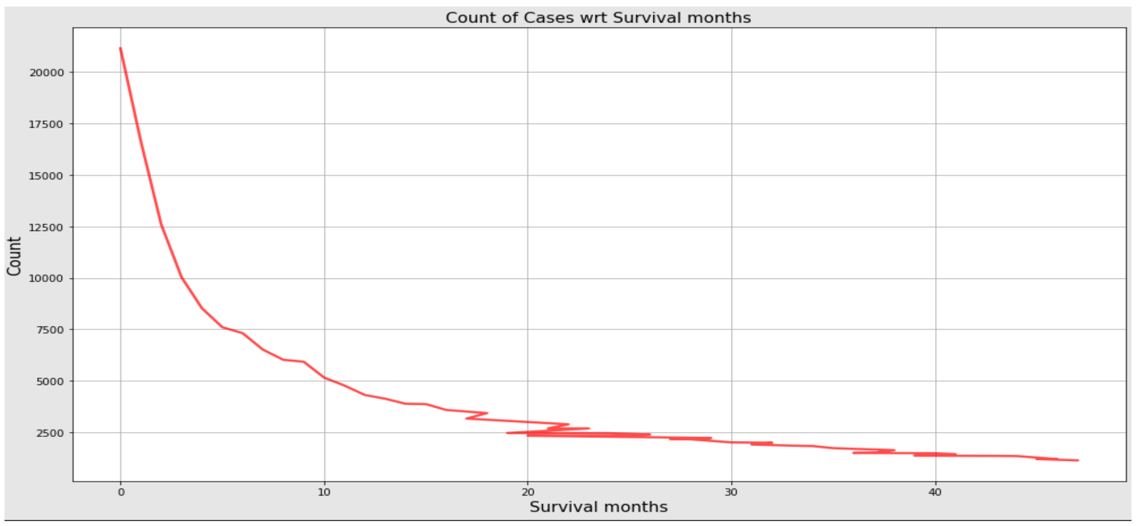
In the year 2021, the patients survived for a maximum of 11 months.



5.4: Distribution of Vital Status by Year of diagnosis

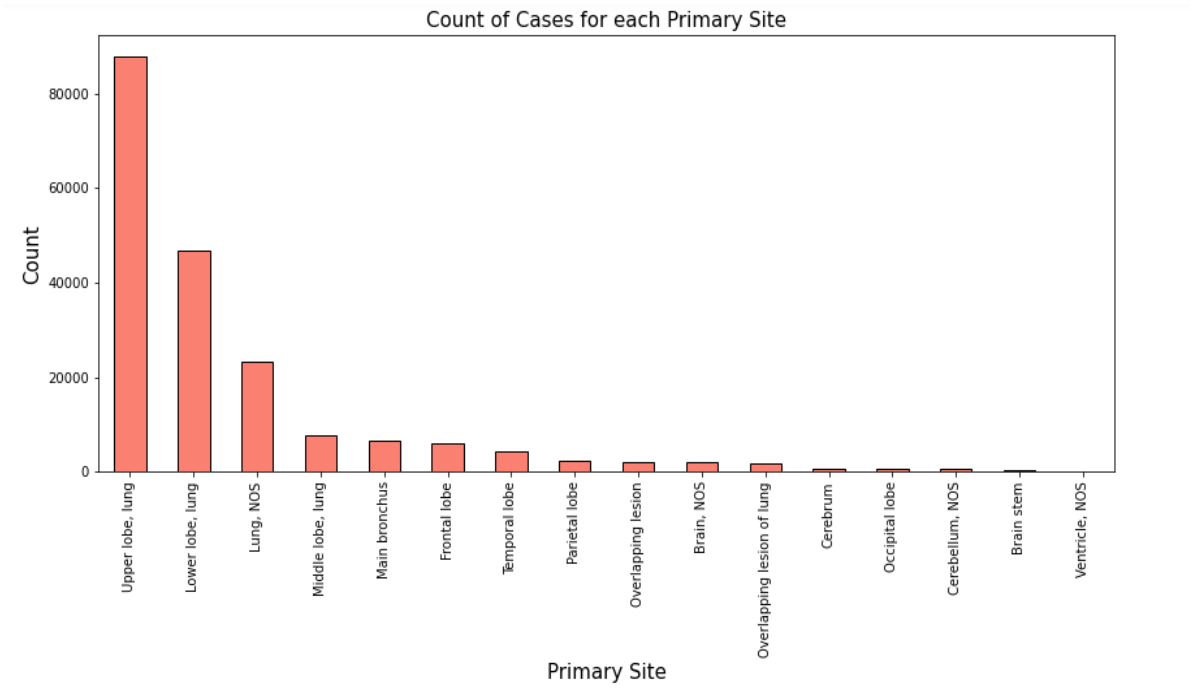
Year of diagnosis	Alive	Dead
2018	0.328421	0.671579
2019	0.395071	0.604929
2020	0.479941	0.520059
2021	0.703865	0.296135

Insights: The year 2018 has highest number of dead patients while 2021 has the least number of dead patients.



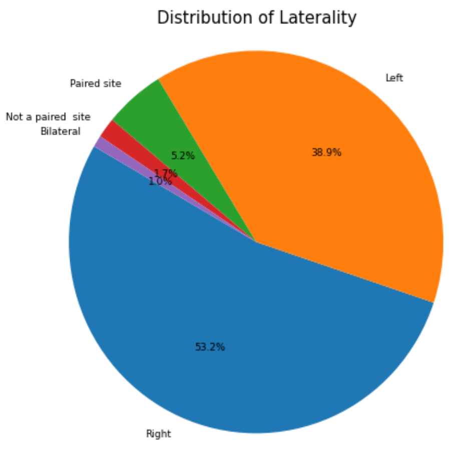
5.5: Count of cases with respect to Survival Months

Insights: We can observe that the number of patients with respect to the survival months has a high negative correlation.



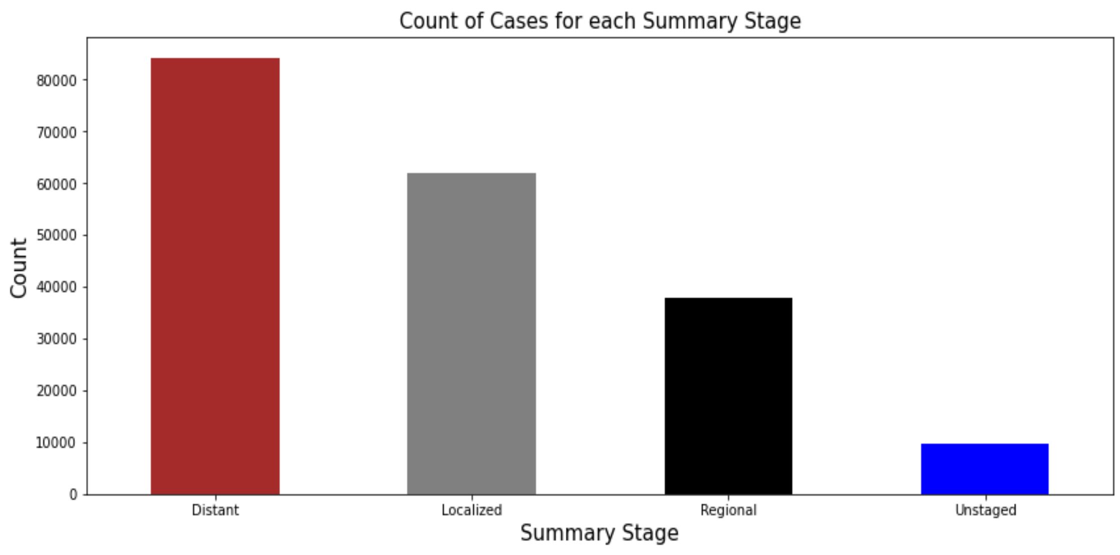
5.6: Count of cases for Primary Site

Insights: The bar chart helps identify the primary sites with the highest and lowest occurrence of cancer cases. In case of Lungs, Upper Lobe is the primary area whereas in Brain, Frontal Lobe is the primary area of attack.



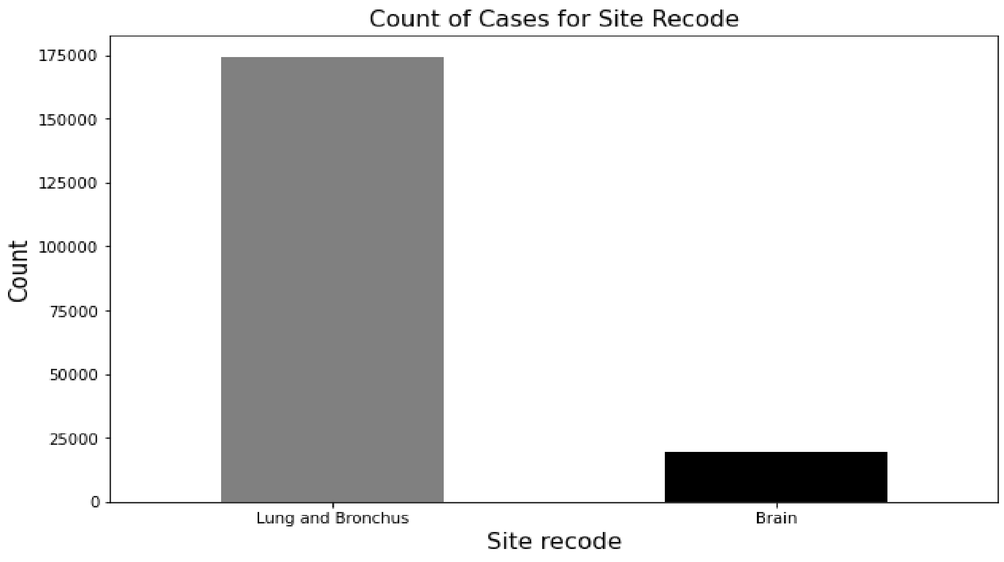
5.7: Distribution of Laterality

Insights: The pie chart offers a visual representation of the relative frequencies of different laterality recoded categories. It can be seen that 53.2% cancer has appeared in right side whereas 38.9% cancer has appeared in left side. A very small percentage of 1% appeared from both the sides which is bilateral.



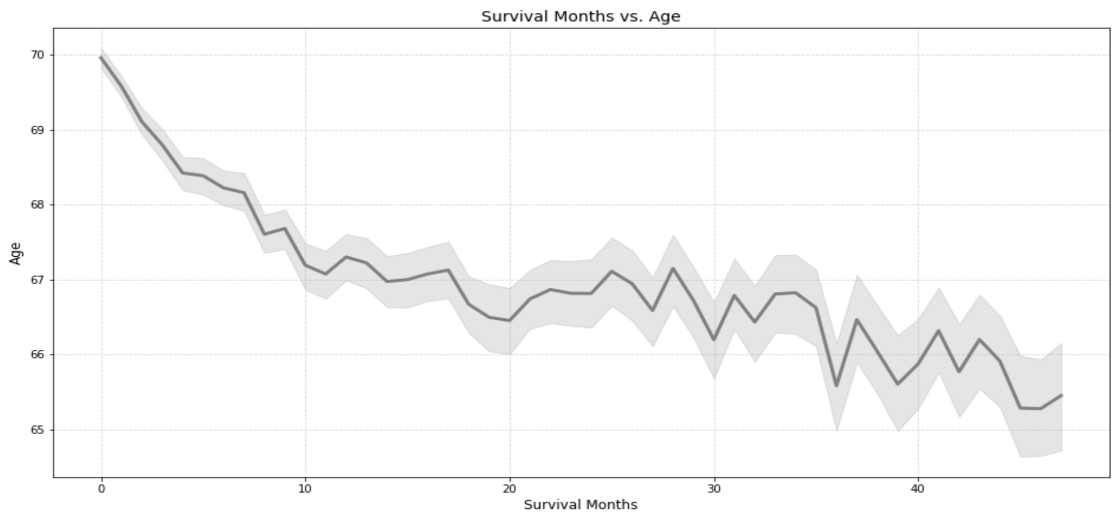
5.8: Count of cases for Summary Stage

Insights: The bar graph provides a visual representation of the distribution of cases across different stages. We observe that the nummber of high risk patients (distant cancer) is significantly more than the number of regional stage cancer patients.



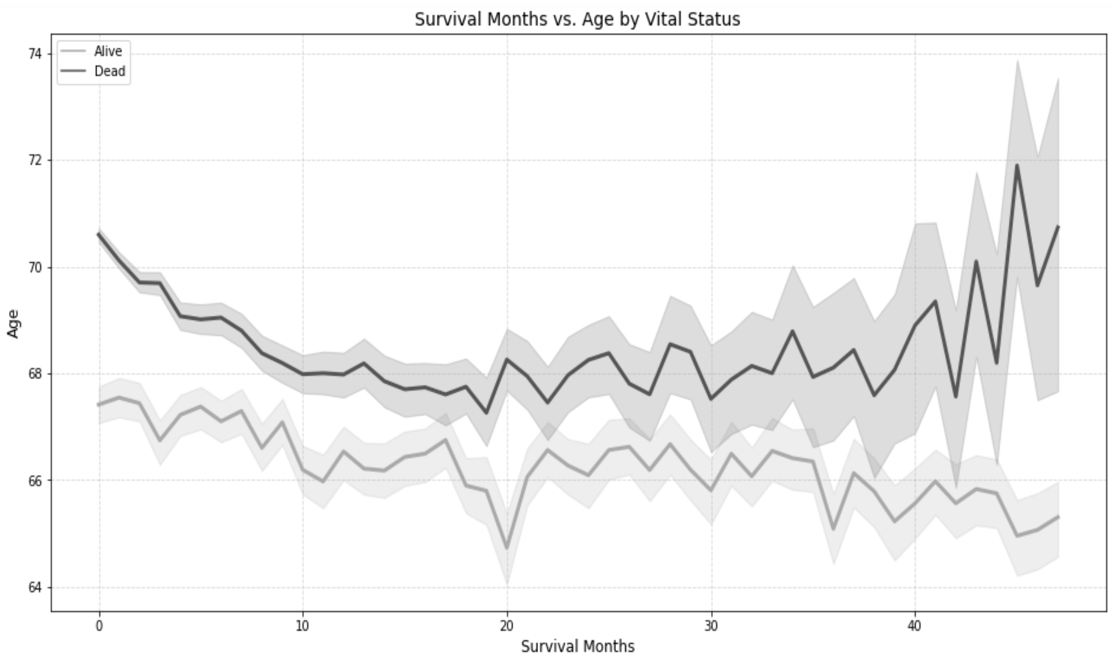
5.9: Count of cases for Site Recode

Insights: There is a data imbalance for site cancers of 'Lungs and Bronchus' and 'Brain'. But this will not affect the accuracy of the data.



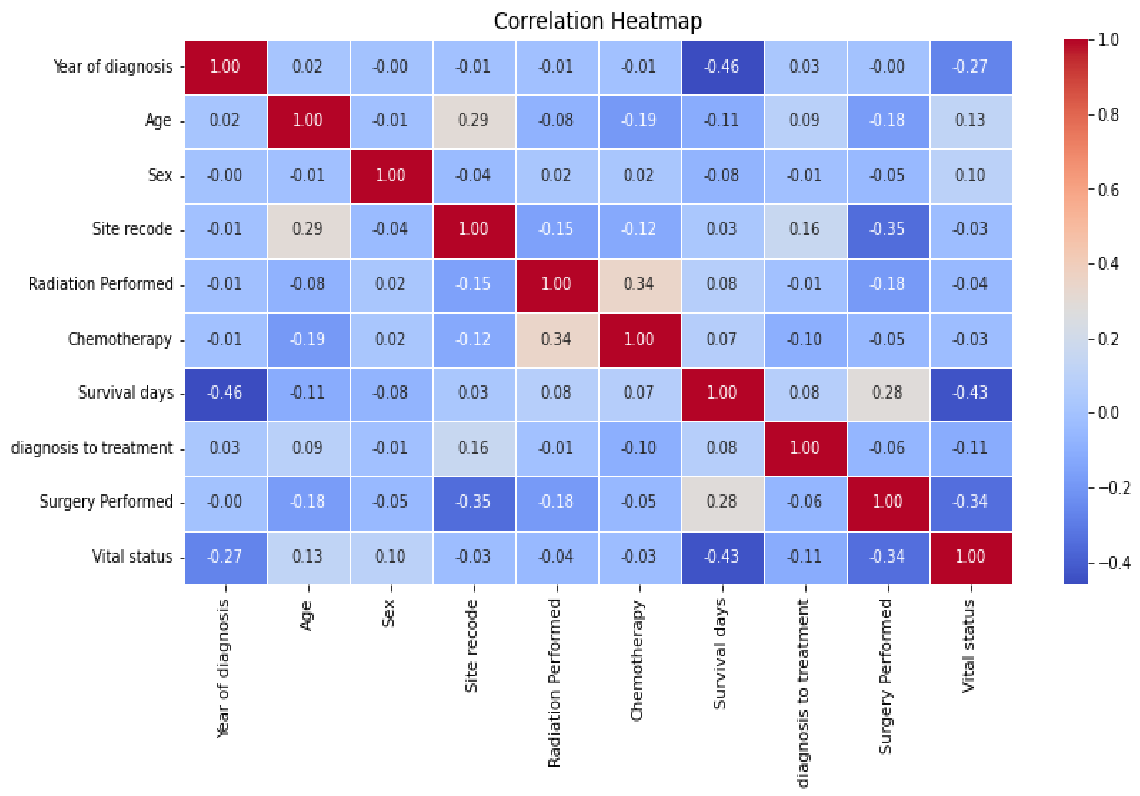
5.10: Survival Months vs Age of Patients

Insights: A negative correlation between Survival months and Age. As survival months increases, age of the patients decreases.



5.11: Survival Months vs Age of Patients by Vital Status

Insights: Trend line for alive patients with respect to Survival months vs Age shows a simple straight line which means alive patients show no correlation to Survival months vs Age. On the other hand, for the dead patients, first there is a negative correlation and then at the end, we observe a positive correlation which may depict the improvement of condition of the patient which helped him to live longer.



	Year of diagnosis	Age	Sex	Site recode	Radiation Performed	Chemotherapy	Survival days	diagnosis to treatment	Surgery Performed	Vital status
Year of diagnosis	1.00	0.02	-0.00	-0.01	-0.01	-0.01	-0.46	0.03	-0.00	-0.27
Age	0.02	1.00	-0.01	0.29	-0.08	-0.19	-0.11	0.09	-0.18	0.13
Sex	-0.00	-0.01	1.00	-0.04	0.02	0.02	-0.08	-0.01	-0.05	0.10
Site recode	-0.01	0.29	-0.04	1.00	-0.15	-0.12	0.03	0.16	-0.35	-0.03
Radiation Performed	-0.01	-0.08	0.02	-0.15	1.00	0.34	0.08	-0.01	-0.18	-0.04
Chemotherapy	-0.01	-0.19	0.02	-0.12	0.34	1.00	0.07	-0.10	-0.05	-0.03
Survival days	-0.46	-0.11	-0.08	0.03	0.08	0.07	1.00	0.08	0.28	-0.43
diagnosis to treatment	0.03	0.09	-0.01	0.16	-0.01	-0.10	0.08	1.00	-0.06	-0.11
Surgery Performed	-0.00	-0.18	-0.05	-0.35	-0.18	-0.05	0.28	-0.06	1.00	-0.34
Vital status	-0.27	0.13	0.10	-0.03	-0.04	-0.03	-0.43	-0.11	-0.34	1.00

5.12: Correlation Heatmap with Vital Status

Insights: We can observe a few moderately correlated variables

FEATURE SELECTION

Feature selection is an important step in machine learning that involves identifying the most relevant variables in a dataset to improve model performance, reduce complexity, and prevent overfitting. Random forest, a widely used ensemble learning method, provides a powerful way to perform feature selection based on its ability to measure feature importance.

Random forest consists of multiple decision trees trained on different subsets of the data. Each tree is built using a random subset of features, and the final prediction is obtained by averaging the predictions (for regression) or using majority voting (for classification). While constructing the trees, random forest naturally ranks features based on how much they contribute to reducing uncertainty or improving decision-making within the trees.

One of the key measures for feature selection in random forest is the importance score, which can be determined by using the mean decrease in accuracy, also called permutation importance. In this approach, the model's accuracy is first recorded using the original dataset.

Then, one feature at a time is randomly shuffled while keeping all other features unchanged. The drop in accuracy after shuffling indicates the importance of that feature. A significant decrease suggests the feature is important, while little or no change implies it may not be necessary. This method is more robust but requires additional computation.

After calculating feature importance scores, the least important features can be removed to simplify the model. Removing irrelevant features reduces noise in the data, speeds up model training, and improves generalization on new data. Feature selection using random forest is particularly useful in high-dimensional datasets, such as cancer research data, where numerous clinical, genetic, or imaging features may be present. By selecting the most informative features, researchers can build more interpretable models while maintaining strong predictive performance. The total importance of all the features in the dataset having feature importance greater than 0.02 (base importance) is 99.16%. Hence we have 99.16% predictive power which will help in most effective prediction.

Below are the feature importances of the top five features of the dataset:

Features	**Importance (in %)**
Surgery Performed	4.93
Age	8.60
Diagnosis to treatment	8.64
Year of diagnosis	18.51
Survival in days	28.83

Table 5.1 – Feature Importance from RF Feature Selection

MODEL BUILDING

Model building is a critical step in developing a predictive system using cancer datasets. It involves training a machine learning model on historical data and validating its performance to ensure it generalizes well to unseen cases.

The process starts with selecting a suitable machine learning algorithm based on the nature of the dataset and the problem at hand. Common algorithms for cancer prediction and classification include **random forest**, **gradient boost**, and **adaptive boost**. The choice depends on factors like dataset size, complexity, and interpretability requirements.

Once the algorithm is selected, the dataset is divided into training and validation sets. The training set consists of around 80 percent of the data, while the remaining 20 percent is used for testing. The training set is used to fit the model, meaning the algorithm learns patterns and relationships between input features (such as tumor size, metastasis status, and treatment history) and the target variable (such as survival rate or disease progression).

Further, **K-fold cross validation** is applied in validation to access the performance of the models by observing the mean validation accuracy over 10 folds of the train dataset. Cross validation is an essential step and the validation set helps detect issues like overfitting, where the model memorizes the training data instead of learning generalizable patterns. Performance metrics such as accuracy is used to evaluate the model.

MODEL SELECTION

Random Forest

Random forest classifier is an ensemble learning algorithm used for classification tasks. It builds multiple decision trees and combines their outputs to make more accurate and stable predictions by randomly selecting subsets of the training data and features to construct multiple decision trees. Each tree is trained independently, and during classification, each tree votes for a class label. The final prediction is determined by majority voting, meaning the class with the most votes is chosen as the final output.

One of the main advantages of a random forest classifier is its ability to handle high-dimensional data and reduce overfitting. Since multiple trees are trained on different subsets of data, the overall model is less likely to memorize noise or irrelevant patterns in the dataset. Random forest is widely used in medical diagnosis tasks due to its robustness and interpretability. It performs well even with unstructured or noisy data and is less sensitive to small changes in the dataset compared to single decision trees.

Gradient Boost

Gradient Boosting Classifier is a machine learning algorithm that builds an ensemble of weak predictive models, typically decision trees, to improve classification accuracy. It belongs to the family of boosting algorithms, where multiple models are trained sequentially, and each new model attempts to correct the errors of the previous ones.

The process begins by training an initial decision tree on the dataset. The errors or residuals from this model are then analyzed, and a second tree is trained to predict these residuals. This process continues for multiple iterations, with each new tree focusing on reducing the mistakes made by the previous trees. The final prediction is made by combining the outputs of all the trees, usually through a weighted sum.

One of the main advantages of Gradient Boosting is its ability to handle complex relationships in data, making it effective for both small and large datasets. It is also highly customizable, allowing users to tune parameters such as the number of trees, learning rate, maximum depth of trees, and subsampling rate to improve performance. However, Gradient Boosting can be computationally expensive, especially with large datasets, and is prone to overfitting if not properly regularized. Techniques such as early stopping, tree pruning, and regularization parameters like learning rate and maximum depth help in preventing overfitting. Gradient Boosting is widely used in various applications, including medical diagnosis due to its high predictive accuracy and flexibility.

Adaptive Boost

Adaptive Boost, commonly known as AdaBoost, is a machine learning algorithm used for classification and regression tasks. It is an ensemble learning technique that builds multiple weak classifiers and combines them to create a strong classifier.

The algorithm starts by training a base model, typically a decision stump, on the given dataset. Initially, all training samples are assigned equal weights. After the first model is trained, AdaBoost evaluates its performance and identifies misclassified instances. It then increases the weights of these misclassified samples so that they receive more attention in the next iteration.

A new model is trained with the updated weights, focusing more on the difficult cases. This process continues for a predefined number of iterations or until the model achieves a desired level of accuracy. Each weak model contributes to the final prediction based on its accuracy, with more reliable models having a greater influence on the final decision.

The final prediction is made by combining the outputs of all weak models through a weighted voting mechanism. In classification problems, the class with the highest combined score is chosen as the final output. AdaBoost is particularly effective in reducing bias and improving model performance on complex datasets.This algorithm is widely used in applications such as medical diagnosis due to its ability to enhance weak classifiers and improve overall predictive performance.

Final Model

Grid Search Cross-Validation (Grid Search CV) is a technique used to find the optimal hyperparameters for a machine learning model. All of the three classifier models that is **random forest**, **gradient boost**, and **adaptive boost** output a validation accuracy of 89.78%, 90.1% and 90% respectively.So, Grid Search CV is applied to **GBC** model due to its best performance of all three models. Hyperparameters are settings that are not learned from the data but are chosen before training.

CHAPTER 6

SAMPLE CODING

SAMPLE CODING FOR FEATURE SELECTION

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.ensemble import RandomForestClassifier
X2 = df_dummycopy3.drop('Vital status', axis=1)
y2 = df_dummycopy3.pop('Vital status')
model = RandomForestClassifier()
model.fit(X2, y2)
feature_importances = model.feature_importances_
```

SAMPLE CODING FOR MODEL SELECTION

```
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, make_scorer
# Random Forest Classifier
RF_classifier =  RandomForestClassifier(n_estimators = 100,
class_weight='balanced', random_state = 42)
cvscore_RF = cross_val_score(RF_classifier,
X_train_scaled, y_train, cv=cv, scoring=scoring_metric)

# Gradient Boosting Classifier
GBC_classifier = GradientBoostingClassifier(n_estimators=100,
learning_rate=0.1,      max_depth=3,      random_state=42,
subsample=0.8)
cvscore_GBC = cross_val_score(GBC_classifier,
X_train_scaled,
y_train, cv=cv, scoring=scoring_metric)
best_GBC_classifier = grid_search.best_estimator_
y_test_pred = best_GBC_classifier.predict(X_test)
test_accuracy = accuracy_score(y_test, y_test_pred)

# Adaptive Boost Classifier
base_learner = DecisionTreeClassifier(max_depth=3)
ADC_classifier =
AdaBoostClassifier(base_estimator=base_learner,
n_estimators=100, learning_rate=0.1, random_state=42)
cvscore_ADC = cross_val_score(ADC_classifier,
X_train_scaled,
y_train, cv=cv, scoring=scoring_metric)

# Grid Search CV
GBC_classifier =
GradientBoostingClassifier(random_state=42)
param_grid = {
   'n_estimators': [100, 150],
   'learning_rate': [0.01, 0.1],
   'max_depth': [3, 5],
   'subsample': [0.8, 1.0]
}
grid_search = GridSearchCV(GBC_classifier,
param_grid, cv=cv, scoring=scoring_metric, n_jobs = -1)
```

CHAPTER 7

MODEL TESTING

SAMPLE CODING FOR TESTING

```
best_GBC_classifier = GradientBoostingClassifier(**grid_search.best_params_)
best_GBC_classifier.fit(X_train_scaled, y_train)
y_pred_GBC = best_GBC_classifier.predict(X_test_scaled)
accu_GBC = accuracy_score(y_test, y_pred_GBC)
```

TESTING

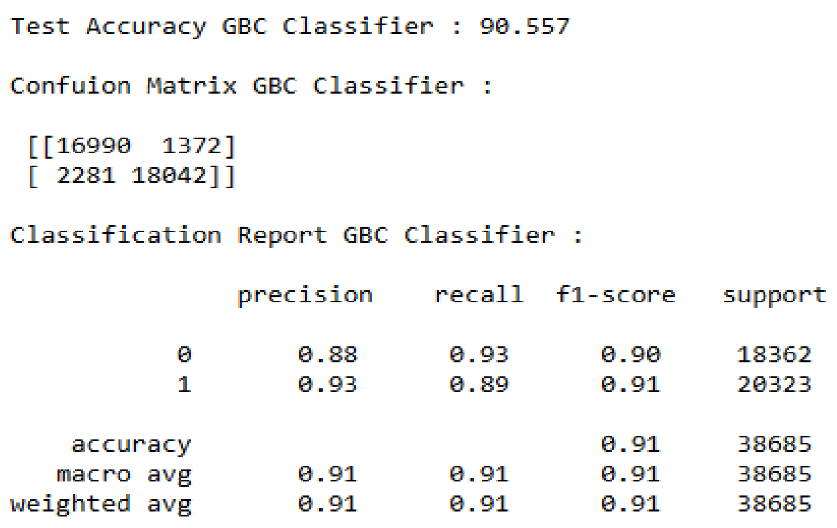
Model testing is an essential step in evaluating the performance of a trained machine learning model on unseen data. It helps determine how well the model generalizes to new cases and ensures its reliability before deployment. The process begins by selecting a test dataset, which consists of data that was not used during the model's training or validation phases. This dataset represents real-world scenarios and allows for an unbiased assessment of the model's predictive ability.

Once the model is trained, it is applied to the test dataset to generate predictions. These predictions are then compared to the actual values in the dataset using appropriate evaluation metrics. The choice of metrics depends on the problem type. For classification models, common metrics include accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve. For regression models, evaluation may involve mean absolute error, mean squared error, or root mean squared error.

Overfitting and underfitting are key concerns during model testing. If a model performs exceptionally well on the training data but poorly on the test data, it may be overfitting, meaning it has memorized patterns rather than learning generalizable trends. If it underperforms on both training and test data, it may be underfitting, indicating that the model is too simple to capture important relationships.

Cross-validation is often used to improve model evaluation. This technique divides the dataset into multiple subsets, training and testing the model on different splits to ensure stability and reduce variability in performance estimates.

The final step is interpreting the results to determine if the model is ready for real-world use. If it meets performance criteria, it can be deployed for practical applications like medical diagnosis or cancer prediction. The test accuracy for the GBC model is given below with the confusion matrix and other metrics such as recall, precision and F1 score.



```
Test Accuracy GBC Classifier : 90.557

Confuion Matrix GBC Classifier :

 [[16990  1372]
 [ 2281 18042]]

Classification Report GBC Classifier :

               precision    recall  f1-score   support

           0       0.88      0.93      0.90     18362
           1       0.93      0.89      0.91     20323

    accuracy                           0.91     38685
   macro avg       0.91      0.91      0.91     38685
weighted avg       0.91      0.91      0.91     38685
```

7.1 – Classification Report of Gradient Boost Classifier

Gradient Boosting Classifier stood out with a test accuracy of 90.56 percent with hyper parameter tuning (by Grid Search CV). The test accuracy represents the proportion of correctly predicted instances in the test data. The models with higher accuracy values indicate better performance in predicting the vital status. It's important to note that accuracy alone might not provide a complete picture of model performance, and other evaluation metrics should be considered as well (e.g., precision, recall, F1-score) to assess the models' overall performance and suitability for the specific task.

In summary, the EDA helped us understand the data distribution and identify patterns, outliers, and relationships between variables. The multivariate analysis provided insights into the associations between different variables and the vital status.

CHAPTER 8

CONCLUSION

The analysis of the lung and brain cancer dataset provides valuable insights into the characteristics, treatment patterns, and survival outcomes of patients diagnosed with these cancers. By preprocessing and analyzing the data, important trends related to tumor size, metastasis, treatment effectiveness, and patient survival can be identified.

For lung cancer, factors such as tumor stage, metastasis status, and treatment methods like chemotherapy and radiation play a crucial role in determining patient survival. Patients with early-stage lung cancer generally have better survival outcomes compared to those diagnosed at later stages. The presence of metastasis significantly reduces survival rates, emphasizing the importance of early detection and timely intervention.

For brain cancer, tumor location, size, and treatment options such as surgery, radiation, and chemotherapy impact patient prognosis. Unlike lung cancer, brain tumors are often more complex due to their location and limited treatment options. The survival rate for brain cancer varies significantly depending on tumor type, with malignant tumors showing lower survival rates compared to benign ones.

Overall, the findings from this dataset highlight the importance of early diagnosis, personalized treatment plans, and continuous advancements in medical research to improve survival rates. Finally, the EDA and high test accuracy of the GBC model will help the medical team diagnose cancer and conclude their findings and insights with higher precision.

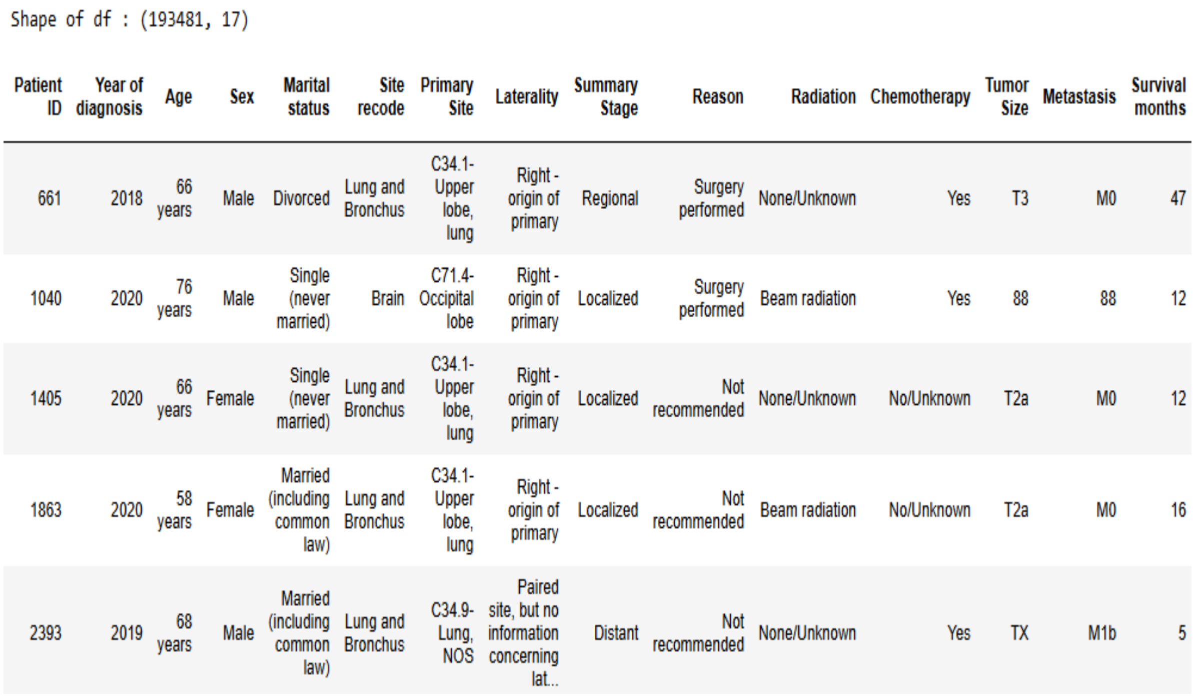
CHAPTER 9

APPENDIX

The appendix is an optional section that can provide additional technical information, large datasets, codesamples and supplementary materials that support the project report. Below are some suggested contents for the appendix:

Appendix A:

Raw Data Table:



```
Shape of df : (193481, 17)
```

Patient ID	Year of diagnosis	Age	Sex	Marital status	Site recode	Primary Site	Laterality	Summary Stage	Reason	Radiation	Chemotherapy	Tumor Size	Metastasis	Survival months
661	2018	66 years	Male	Divorced	Lung and Bronchus	C34.1-Upper lobe, lung	Right - origin of primary	Regional	Surgery performed	None/Unknown	Yes	T3	M0	47
1040	2020	76 years	Male	Single (never married)	Brain	C71.4-Occipital lobe	Right - origin of primary	Localized	Surgery performed	Beam radiation	Yes	88	88	12
1405	2020	66 years	Female	Single (never married)	Lung and Bronchus	C34.1-Upper lobe, lung	Right - origin of primary	Localized	Not recommended	None/Unknown	No/Unknown	T2a	M0	12
1863	2020	58 years	Female	Married (including common law)	Lung and Bronchus	C34.1-Upper lobe, lung	Right - origin of primary	Localized	Not recommended	Beam radiation	No/Unknown	T2a	M0	16
2393	2019	68 years	Male	Married (including common law)	Lung and Bronchus	C34.9-Lung, NOS	Paired site, but no information concerning lat...	Distant	Not recommended	None/Unknown	Yes	TX	M1b	5

9.1: Raw Data Table

Code Snippet for Feature Engineering:

Feature Transformation

```
df['Survival days'] = df['Survival months'] * 30
df['Surgery Performed'] = df['Reason'].apply(lambda x: 1 if x == 'Surgery
performed' else 0)
df['Radiation Performed'] = df['Radiation'].apply(lambda x: 0 if x in
no_categories else 1)
```

Feature Encoding

```
df['Surgery Performed'] = df['Surgery Performed'].astype('category').cat.codes
df['Radiation Performed'] =
df['RadiationPerformed'].astype('category').cat.codes
df['Chemotherapy'] = df['Chemotherapy'].astype('category').cat.codes
df['Site recode'] = df['Site recode'].astype('category').cat.codes
df_dummy = pd.get_dummies(df, columns = ['Marital status', 'Primary Site',
'Laterality', 'Summary Stage', 'Tumor Size', 'Metastasis'], drop_first=True)
```

Appendix B:

Glossary of Terms

ML	Machine Learning
GBC	Gradient Boosting Classifier
RF	Random Forest
EDA	Exploratory Data Analysis
CV	Cross Validation
SEER	Surveillanc, Epidemology and End Results
SciPy	Scientific Python

NumPy	Numerical Pytho**n**

9.2: Glossary of Terms

Data Dictionary:

The **Lungs and Cancer Dataset** seeks to classify the patients on their vital status based on the different medical conditions of the patients based on their age and year of diagnosis. The list of all the columns used in the dataset and their description is given below:

Attributes	**Description**
Patient ID	Represents ID of the patient
Year of Diagnosis	Represents the year when patient was daignosed
Age	Age of the patient
Sex	Whether the patient is male or female
Marital Status	If the patient is married, single or divorced
Site Recode	Attack on lungs or brain of the patient
Primary Site	Represents primary site of the tumor
Laterality	Indicates whether the cancer occurred on the left, right, or a paired organ
Summary Stage	Represents the cancer progression of the patient
Reason	Explanation for specific treatment decisions
Radiation	Indicates whether the patient received radiation therapy

Chemotherapy	Indicates whether the patient was administered chemotherapy
Tumor Size	The measured size of the tumor at the time of diagnosis
Metastasis	Information on whether the cancer has spread to distant parts of the body
Survival Months	The number of months the patient survived after diagnosis
Diagnosis to treatment	The time interval (in days) from the initial cancer diagnosis to the start of the treatment of the patient
Vital Status	The patient's status i.e. alive or deceased

9.3: Attributes and Description

CHAPTER 10

REFERENCE

Books:

1) Mastering Machine Learning Algorithms (Second Edition): Giuseppe B

2) Python for Data Analysis: Data Wrangling with Pandas, Num Py and Ipython 2nd Edition : Wes Mc Kinney

Websites:

1) UML Diagrams:
   - Creately (creately.com) – Offers a drag-and-drop interface for UML diagrams.
   - Draw.io (app.diagrams.net) – Free and open-source, great for creating UML diagrams.
   - Lucidchart (lucidchart.com) – A user-friendly tool with templates for UML diagrams.
   - PlantUML (plantuml.com) – A text-based UML diagram generator.

2) Dataset
   - National Cancer Institute, United States - Surveillance, Epidemiology, and End Results Program

3) Journals
   - **Lung Cancer Detection: A Deep Learning Approach** - Examines deep learning methods for lung cancer classification
   - **ML-based early detection of lung cancer** - Focuses on a machine learning model for predicting lung cancer