

BREAST CANCER DETECTION USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Science in Bioinformatics and Data Science

By

RANJITH R (41738012)



**DEPARTMENT OF BIOINFORMATICS
SCHOOL OF BIO AND CHEMICAL ENGINEERING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A++" by NAAC I Category-1 University by UGC I Approved by AICTE
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC | Category-1 University by UGC | Approved by AICTE
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF BIOINFORMATICS

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Ranjith (41738012)**, who carried out the final year project entitled "**Breast Cancer Detection Using Machine Learning**" under my supervision from **January 2024** to **April 2024**.

Dr. H. JEMMY CHRISTY
Internal Guide

Dr. H. JEMMY CHRISTY
Head of the Department

Submitted for Viva Voce Examination held on

27.04.2024

Internal Examiner

External Examiner

DECLARATION

I, **RANJITH**, hereby declare that the Project Report entitled **Breast Cancer Detection Using Machine** done by me under the guidance of **Dr. H. JEMMY CHRISTY**, at **Sathyabama Institute of Science and Technology**, Jeppiaar Nagar, Rajiv Gandhi Road, Chennai-600119, Tamil Nadu, India, is submitted in partial fulfillment of the requirements for the award of **Bachelor of Science degree in Bioinformatics and Data Science**.

DATE: 27.04.2024

PLACE: CHENNAI

Ranjith . R

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am delighted to express my heartfelt acknowledgement to the **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement and support throughout the duration of my project. Their guidance has been invaluable in ensuring the successful completion of this project, and I am sincerely grateful for their contribution.

I convey my thanks to **Dr. Rekha Chakravarthy, Dean, School of Science and Humanities** for providing me necessary support and details at the right time during the progressive reviews.

I sincerely thank **Dr. H. Jemmy Christy, Head of the Department, Dept. of Bioinformatics**, for her unwavering support and timely provision of necessary details during the progressive reviews. Her support has been instrumental in helping me to overcome obstacles and achieve success in this project.

I am deeply indebted to my **Project Guide, Dr. H. Jemmy Christy**, for her invaluable guidance, suggestions, and constant encouragement throughout the project. Her expertise and insights have been instrumental in shaping my work and ensuring its successful completion.

I also wish to extend my gratitude to my family for their support and assistance in various ways, which have contributed to the successful completion of this project.

Ranjith



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC I Category-1 University by UGC I Approved by AICTE
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

CERTIFICATE

This is to certify that the dissertation entitled "**Breast Cancer Detection Using Machine Learning**" was carried out by **Ms. RANJITH (Reg. No. 41738012)**, Department of Bioinformatics and Data Science, Sathyabama Institute of Science and Technology, Chennai in partial fulfillment of the requirements of the award of the **Degree of Bachelor of Science in Bioinformatics and Data Science 2021-2024**, is a bonafide record of the project work done under my guidance and supervision. This thesis or part, therefore, has not been submitted elsewhere for any other degree or diploma.

Date: 27.04.2024

Place: CHENNAI

Dr. H. JEMMY CHRISTY

ABSTRACT

Breast cancer remains a leading cause of cancer-related morbidity and mortality globally, necessitating innovative approaches for early detection, accurate diagnosis, and tailored treatment strategies. In recent years, machine learning (ML) has garnered considerable attention for its potential to revolutionize breast cancer research and clinical practice. ML algorithms, fueled by large-scale genomic, imaging, and clinical datasets, offer unprecedented opportunities to decipher complex biological mechanisms, predict disease trajectories, and optimize therapeutic interventions. This abstract provides a comprehensive overview of the diverse applications of ML in breast cancer, encompassing image-based diagnosis using mammography, MRI, and digital pathology; genomic profiling for subtype classification and biomarker discovery; clinical decision support systems for treatment planning and patient management; and prognostic modeling for predicting disease recurrence and survival outcomes. We delve into cutting-edge methodologies such as deep learning architectures, transfer learning, and federated learning, which are pushing the boundaries of predictive accuracy and scalability in breast cancer ML research. Moreover, we discuss the integration of multimodal data sources, including electronic health records, patient-reported outcomes, and real-world evidence, to capture the holistic patient profile and guide personalized care pathways. Despite the remarkable progress, several challenges persist, including data standardization, model interpretability, and ethical considerations surrounding data privacy and bias mitigation. Through collaborative efforts among researchers, clinicians, policymakers, and industry stakeholders, ML-driven innovations hold immense promise for transforming breast cancer care, ushering in an era of precision oncology where treatments are tailored to individual patients' unique molecular and clinical profiles. This abstract explores the integration of multi-omics data and real-world evidence to elucidate the complex interplay between genetic alterations, tumor microenvironment, and therapeutic response. Moreover, we discuss the role of ML in addressing challenges such as data heterogeneity, model interpretability, and scalability, with a focus on innovative approaches such as federated learning and privacy-preserving algorithms.

Table of Contents

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF ABBREVIATION	III
	LIST OF FIGURES	IV
1	INTRODUCTION	1
	1.1 HEALTH IMPACT	2
	1.2 LIMITATION OF TRADITIONAL	2
	1.3ROLE OF MACHINE LEARNING	3
	1.4 BREAST CANCER PREDICATION	5
	1.5 MACHINE LEARNING	6
	1.6 ALGORITHM	7
2	REVIEW OF LITERATURE	10
3	AIM AND OBJECTIVES	12
	3.1 AIM	12
	3.2 OBJECTIVES	12
4	METHODOLOGY	15
	4.1 DATA COLLECTION	15
	4.2 BREAST CANCER	16
	4.3 LODE THE DATASET	17
	4.4 DATA VISUALIZATION	17
	4.5 DATA PREPROCESSING	18
	4.6 TRAIN THE MODEL	19
	4.7 PREDICT USIND TRAINED MODEL	20
	4.8 EVALUATE MODEL PERFORMANCE	20
	4.9 MODEL TUNING AND OPTIMIZATION(OPTIONAL)	21

	4.10 SVM MODEL	22
5	RESULTS AND DISCUSSION	25
	5.1 RESULTS	25
	5.2 BREAST CANCER PREDICATION	26
6	SUMMARY AND CONCLUSIONS	47
	REFERENCES	50

LIST OF ABBREVIATIONS

ML	Machine Learning
CNN	Convolutional Neural Network
SVM	Support Vector Machine
CAD	Computer-Aided Detection/Diagnosis
ROI	Region of Interest
DICOM	Digital Imaging and Communications in Medicine
AI	Artificial Intelligence
RGB	Red, Green, Blue (color model)
AUC-ROC	Area Under the Receiver Operating Characteristic Curve
FN	False Negative
FP	False Positive
TP	True Positive
TN	True Negative
ROI	Region of Interest
FDA	Food and Drug Administration

LIST OF FIGURES

- Figure 1:** Workflow of Breast Cancer Detection System
- Figure 2:** Mammography Image Preprocessing Steps
- Figure 3:** Convolutional Neural Network (CNN) Architecture
- Figure 4:** Ultrasound Image Processing Pipeline
- Figure 5:** Receiver Operating Characteristic (ROC) Curve
- Figure 6:** Confusion Matrix
- Figure 7:** Feature Importance Plot
- Figure 8:** Comparison of ML Algorithms
- Figure 9:** Deployment Architecture
- Figure 10:** Case Study Result

CHAPTER 1

INTRODUCTION

Breast cancer is a significant global health challenge, accounting for a substantial portion of cancer-related morbidity and mortality among women worldwide. Early detection plays a pivotal role in improving patient outcomes by enabling timely intervention and treatment. Mammography, the gold standard for breast cancer screening, has limitations in terms of interpretation and diagnostic accuracy, often leading to missed or delayed diagnoses.

Breast cancer arises from the uncontrolled growth of abnormal cells in the breast tissue, which can manifest as a lump or mass that may be detected through screening or self-examination. While the majority of breast cancer cases occur in women, it is important to acknowledge that men can also develop this disease, albeit at a significantly lower frequency.

In recent years, advances in deep learning techniques have shown promise in improving the accuracy and efficiency of breast cancer detection from mammographic images. By leveraging convolutional neural networks (CNNs) and large-scale datasets, deep learning models can automatically extract relevant features and identify suspicious lesions indicative of breast malignancy.

The Memorial Sloan Kettering Cancer Center (WISCONSIN) dataset, available in CSV format, offers a valuable resource for research in breast cancer detection. Comprising a comprehensive collection of clinical and imaging data, including patient demographics, tumor characteristics, and radiomic features extracted from mammographic images, this dataset provides a rich source of information for training and evaluating deep learning models.

1.1 HEALTH IMPACT

The integration of machine learning (ML) techniques into breast cancer detection has the potential to revolutionize the landscape of healthcare, particularly in the realm of women's health. By leveraging ML algorithms to analyze medical imaging data and patient information, healthcare providers can achieve earlier and more accurate detection of breast cancer, leading to several significant health impacts.

1. Early Detection Saves Lives: ML-based breast cancer detection enables the identification of abnormalities at their earliest stages, even before symptoms manifest. Early detection is paramount in improving treatment outcomes and survival rates, as it allows for timely intervention and more effective treatment strategies.

2. Personalized Treatment Approaches: ML algorithms can analyze vast amounts of patient data, including genetic information, imaging results, and medical history, to tailor treatment plans based on individual characteristics. This personalized approach ensures that patients receive treatments that are optimized for their specific needs, improving both efficacy and patient satisfaction.

3. Reduced False Positives and Negatives: ML algorithms are capable of learning from large datasets, which can help mitigate the occurrence of false positives and false negatives in breast cancer detection. By minimizing these errors, ML-based approaches reduce unnecessary patient anxiety and invasive procedures while ensuring that no potential malignancies are overlooked.

4. Resource Optimization: Efficient breast cancer detection through ML can lead to better allocation of healthcare resources. By streamlining the diagnostic process and prioritizing high-risk cases, healthcare providers can optimize workflows, reduce wait times for patients, and allocate resources more effectively to areas with the greatest need.

1.2 LIMITATION OF TRADITIONAL METHOD

1. Subjectivity and Interobserver Variability: Traditional methods often rely on visual interpretation by radiologists or clinicians, leading to subjective assessments of

breast imaging results. Interobserver variability, where different observers may interpret the same image differently, can lead to inconsistencies in diagnosis and potentially missed or misdiagnosed cases of breast cancer.

2. Limited Sensitivity and Specificity: Mammography, while widely used for breast cancer screening, has limitations in terms of sensitivity, especially in dense breast tissue. Additionally, traditional methods may lack specificity, leading to false-positive results that can cause unnecessary anxiety for patients and additional diagnostic procedures that may not be warranted.

3. Invasive Procedures for Confirmation: In cases where abnormalities are detected through traditional screening methods, confirmation often requires invasive procedures such as biopsies. These procedures carry risks of complications and discomfort for patients, and they can also result in delays in diagnosis and treatment.

4. Inability to Utilize Complex Data: Traditional methods typically rely on a limited set of imaging features and clinical data for diagnosis. They may struggle to incorporate complex patterns and relationships within large datasets, such as those containing genetic information, which can provide valuable insights for accurate diagnosis and prognosis.

5. Limited Adaptability and Improvement Over Time: Traditional methods are static in nature and do not adapt or improve based on feedback or new data. In contrast, ML algorithms can continuously learn from new information and experiences, leading to iterative improvements in accuracy and performance over time.

1.3 ROLE OF MACHINE LEARNING

Machine learning (ML) plays a pivotal role in breast cancer detection by harnessing the power of data analysis, pattern recognition, and predictive modeling. Here are several key roles that ML fulfills in the context of breast cancer detection

1. Early Detection: ML algorithms analyze various types of data, including medical imaging, patient demographics, and clinical history, to identify subtle patterns indicative of early-stage breast cancer. By detecting abnormalities at an early stage, ML models enable timely intervention and treatment, ultimately improving patient outcomes and survival rates.

2. Improved Accuracy and Efficiency: ML algorithms can process and interpret vast amounts of complex data with speed and precision, surpassing human capabilities in some aspects. This leads to more accurate and efficient detection of breast cancer, reducing the likelihood of missed diagnoses or false positives associated with traditional methods.

3. Personalized Risk Assessment: ML techniques enable the development of personalized risk assessment models that take into account individual patient characteristics, including genetic predisposition, lifestyle factors, and medical history. By identifying high-risk individuals, healthcare providers can offer targeted screening and preventive measures to those who stand to benefit the most.

4. Image Analysis and Interpretation: ML algorithms excel in analyzing medical imaging data, such as mammograms, ultrasound scans, and magnetic resonance imaging (MRI), to detect subtle abnormalities indicative of breast cancer. Through techniques like convolutional neural networks (CNNs), ML models can automatically extract features from images and classify lesions with high accuracy, aiding radiologists in their interpretation and diagnosis.

5. Integration of Multimodal Data: ML facilitates the integration of multimodal data sources, including imaging, genomic, proteomic, and clinical data, to create comprehensive models for breast cancer detection and prognosis. By leveraging diverse sources of information, ML algorithms can uncover hidden relationships and biomarkers that may not be apparent through traditional analyses alone.

6. Continuous Learning and Improvement: ML models have the capacity to learn from new data and experiences, enabling continuous improvement in their performance over time. As more data becomes available and algorithms are refined, ML-based breast cancer detection systems evolve to achieve higher levels of accuracy, sensitivity, and specificity.

7. Decision Support for Healthcare Professionals: ML serves as a valuable decision support tool for healthcare professionals, providing them with objective insights and recommendations to aid in clinical decision-making. By augmenting the expertise of healthcare providers, ML empowers them to make more informed decisions and deliver personalized care to patients.

1.4 BREAST CANCER PREDICTION

Breast cancer is a significant health concern affecting millions of individuals worldwide, making early detection crucial for effective treatment and improved outcomes. Machine learning (ML) techniques offer promising tools for breast cancer prediction by analyzing various data sources and identifying patterns indicative of malignancy. In this study, we utilize a dataset containing clinical and imaging features to develop ML models capable of predicting breast cancer with high accuracy.

1. Dataset Description:

The dataset comprises features extracted from digitized images of fine needle aspirates (FNA) of breast masses, as well as clinical data such as patient age and tumor characteristics. Features include texture, radius, perimeter, concavity, and more, providing a comprehensive representation of breast mass properties.

2. Data Preprocessing: The dataset is preprocessed to handle missing values, normalize features, and address any data imbalance issues.

3. Feature Selection: Feature selection techniques such as recursive feature elimination (RFE) or principal component analysis (PCA) are employed to identify the most informative features for breast cancer prediction.

4. Model Selection: Several ML algorithms are evaluated, including support vector machines (SVM), random forests, logistic regression, and artificial neural networks (ANN), to determine the optimal model for breast cancer prediction.

5. Model Training and Evaluation: The selected ML models are trained on a subset of the dataset and evaluated using cross-validation techniques to assess their performance in terms of accuracy, sensitivity, specificity, and area under the receiver operating characteristic.

6. Hyperparameter Tuning: Hyperparameters of the chosen ML models are fine-tuned to optimize their performance and generalization capabilities.

7. Model Validation: The final ML model is validated using an independent test dataset to ensure its robustness and reliability for real-world application.

1.5 MACHINE LEARNING

Breast cancer is a prevalent and potentially life-threatening disease, making early detection vital for successful treatment outcomes. Machine learning (ML) techniques have emerged as powerful tools for breast cancer detection, leveraging data analysis and pattern recognition to improve diagnostic accuracy. In this study, we explore the application of ML algorithms for breast cancer detection using a comprehensive dataset containing clinical and imaging features.

Dataset Description:

The dataset comprises features extracted from digitized images of breast masses obtained through fine needle aspirates (FNA), along with demographic and clinical information. Features include texture, radius, perimeter, concavity, and more, providing rich information for ML-based analysis.

1. Data Preprocessing: The dataset undergoes preprocessing steps, including handling missing values, feature normalization, and addressing data imbalances, to ensure the quality and consistency of the data.

2. Feature Selection: Feature selection techniques such as recursive feature elimination (RFE) or principal component analysis (PCA) are applied to identify the most relevant features for breast cancer detection.

3. Model Selection: Various ML algorithms are evaluated, including support vector machines (SVM), random forests, logistic regression, and artificial neural networks (ANN), to identify the optimal model for breast cancer detection.

4. Model Training and Evaluation: The selected ML models are trained on a subset of the dataset and evaluated using cross-validation techniques to assess their

performance metrics, such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC).

5. Hyperparameter Tuning: Hyperparameters of the chosen ML models are fine-tuned using techniques such as grid search or random search to optimize their performance and generalization capabilities.

6. Model Validation: The final ML model is validated using an independent test dataset to ensure its robustness and reliability for real-world application. The performance of the developed ML models is evaluated based on various metrics, demonstrating their ability to accurately detect breast cancer from clinical and imaging data. The selected model achieves high accuracy and sensitivity, indicating its potential as a valuable tool for breast cancer detection. Machine learning techniques offer promising avenues for breast cancer detection, leveraging data-driven insights to improve diagnostic accuracy and patient outcomes. The developed ML model holds promise for integration into clinical practice, facilitating early detection and personalized treatment strategies for individuals at risk of breast cancer.

1.6 ALGORITHM

1. Data Preprocessing:

- Load the breast cancer dataset containing clinical and imaging features.
- Handle missing values by imputation or removal.
- Normalize the features to ensure uniform scaling.
- Split the dataset into training and testing sets for model evaluation.

2. Feature Selection:

- Apply feature selection techniques such as recursive feature elimination (RFE) or principal component analysis (PCA) to identify the most informative features.
- Select a subset of features based on their importance scores or variance explained.

3. Model Selection:

- Choose a machine learning algorithm suitable for binary classification tasks, such as support vector machines (SVM), random forests, logistic regression, or artificial neural networks (ANN).
- Evaluate multiple algorithms using cross-validation to determine the best-performing model.

4. Model Training:

- Train the selected machine learning model on the training dataset using the chosen algorithm.
- Tune hyperparameters using techniques like grid search or random search to optimize model performance.

5. Model Evaluation:

- Evaluate the trained model on the testing dataset to assess its performance metrics, including accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC).
- Generate a confusion matrix to visualize true positive, true negative, false positive, and false negative predictions.

6. Model Validation:

- Validate the final model using an independent validation dataset or through cross-validation to ensure its robustness and generalization capabilities.

Algorithm

1. Load breast cancer dataset
2. Preprocess data (handle missing values, normalize features)
3. Perform feature selection (RFE, PCA)
4. Select ML algorithm (SVM, Random Forest, Logistic Regression, ANN)
5. Train-test split the dataset

6. Train the selected model on the training data
7. Tune hyperparameters using grid search or random search
8. Evaluate model performance on the testing data
9. Calculate accuracy, sensitivity, specificity, AUC-ROC
10. Generate confusion matrix
11. Validate model using independent validation dataset or cross-validation

CHAPTER 2

REVIEW OF LITERATURE

Breast cancer remains a significant health concern worldwide, prompting extensive research into improving early detection methods. Machine learning (ML) techniques have emerged as promising tools for breast cancer detection, offering the potential to enhance diagnostic accuracy and improve patient outcomes. This literature review aims to provide an overview of recent studies on breast cancer detection using ML approaches.

1. Traditional Methods vs. Machine Learning:

- Traditional methods for breast cancer detection, such as mammography and manual palpation, have limitations in terms of accuracy and reliability.
- ML techniques offer advantages such as automated analysis, pattern recognition, and the ability to integrate diverse data sources for more comprehensive assessments.

2 ML Algorithms for Breast Cancer Detection:

- Various ML algorithms have been explored for breast cancer detection, including support vector machines (SVM), random forests, logistic regression, and artificial neural networks (ANN).
- Studies have compared the performance of different algorithms in terms of accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC).

3 Feature Selection and Extraction:

- Feature selection techniques such as recursive feature elimination (RFE), principal component analysis (PCA), and genetic algorithms have been used to identify relevant features for breast cancer detection.

- Studies have evaluated the impact of feature selection methods on model performance and interpretability

3. Integration of Multimodal Data:

- ML approaches enable the integration of multimodal data sources, including imaging, genomic, proteomic, and clinical data, to improve breast cancer detection and diagnosis.
- Studies have investigated the effectiveness of combining different types of data for more accurate predictions.

4. Challenges and Future Directions:

- Challenges in breast cancer detection using ML include issues related to data quality, interpretability of models, and generalization to diverse patient populations.
- Future research directions include the development of more robust ML algorithms, validation on larger and more diverse datasets, and integration into clinical practice.

CHAPTER 3

AIM AND OBJECTIVE

3.1 AIM

The aim of this study is to utilize machine learning techniques for breast cancer detection, specifically focusing on the analysis of the Wisconsin Breast Cancer Dataset, to improve diagnostic accuracy and facilitate early intervention.

The aim of this study is to utilize machine learning techniques for breast cancer detection, specifically focusing on the analysis of the Wisconsin Breast Cancer Dataset, to improve diagnostic accuracy and facilitate early intervention.

3.2 OBJECTIVES

1. To acquire and preprocess the Wisconsin Breast Cancer Dataset, ensuring data integrity and consistency by handling missing values, normalizing features, and addressing data imbalances.
2. To explore various machine learning algorithms, including support vector machines (SVM), random forests, logistic regression, and artificial neural networks (ANN), for breast cancer detection using the Wisconsin dataset.
3. To conduct feature selection and extraction techniques, such as recursive feature elimination (RFE) and principal component analysis (PCA), to identify the most informative features for breast cancer detection from the Wisconsin dataset.
4. To train and evaluate machine learning models using cross-validation techniques, assessing their performance metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC).
5. To optimize hyperparameters of the selected machine learning models using techniques like grid search or random search to improve model performance on the Wisconsin dataset.

6. To validate the final machine learning model using an independent test subset of the Wisconsin dataset or through cross-validation to ensure its robustness and generalization capabilities.

7. To compare the performance of the developed machine learning models with traditional methods of breast cancer detection, such as mammography, using the Wisconsin dataset as a benchmark.

8. To investigate the interpretability of the developed machine learning models and identify key features contributing to breast cancer detection to enhance clinical understanding.

9. To assess the potential clinical implications of the developed machine learning models, including their integration into clinical workflows for improved screening and diagnosis of breast cancer using the Wisconsin dataset.

10. To disseminate the findings of the study through scientific publications, presentations, and knowledge exchange activities to contribute to the advancement of breast cancer detection research and inform healthcare practice.

11. To optimize hyperparameters of the selected machine learning models using techniques like grid search or random search to improve model performance on the Wisconsin dataset.

12. To validate the final machine learning model using an independent test subset of the Wisconsin dataset or through cross-validation to ensure its robustness and generalization capabilities.

13. To compare the performance of the developed machine learning models with traditional methods of breast cancer detection, such as mammography, using the Wisconsin dataset as a benchmark.

14. To investigate the interpretability of the developed machine learning models and identify key features contributing to breast cancer detection to enhance clinical understanding.

15. To assess the potential clinical implications of the developed machine learning models, including their integration into clinical workflows for improved screening and diagnosis of breast cancer using the Wisconsin dataset.

CHAPTER 4

METHODOLOGY

4.1 DATA COLLECTION:

Collecting data for breast cancer research involves gathering information from various sources to analyze patterns, risk factors, treatment outcomes, and more. Here's a general outline of what data collection for breast cancer research might entail:

4.1.1 Define Your Search Criteria:

Determine what kind of breast cancer-related data you are looking for (e.g., code repositories, datasets, research papers).

Decide on specific keywords (e.g., "breast cancer," "mammography," "tumor detection") to use in your GitHub search.

4.1.2 GitHub Search:

Use GitHub's advanced search feature (<https://github.com/search/advanced>) to filter repositories based on your criteria.

Enter your keywords and specify filters like language (e.g., Python, R), repository size, and topics related to breast cancer.

4.1.3 Exploring Repositories:

Review the repositories that match your search criteria.

Look for repositories that contain relevant code, data, or documentation related to breast cancer research or applications.

4.1.4 Data Collection:

Clone or download repositories containing relevant breast cancer data.

Check the repository README files, documentation, or code comments for information about the data.

4.1.5 Data Validation:

Validate the data sources to ensure credibility and relevance.

Check for proper data usage permissions and licenses.

4.1.6 Ethical Considerations:

Respect data usage policies and licenses associated with the collected repositories.

Ensure compliance with ethical guidelines related to data usage and privacy.

4.1.7 Organize and Share:

Organize the collected data in a structured format.

Consider sharing your curated breast cancer GitHub data collection with the community via a GitHub repository, making sure to credit the original sources appropriately.

4.2 BREAST CANCER:

For breast cancer prediction and datasets like the Wisconsin Breast Cancer dataset (often referred to as the Wisconsin Diagnostic Breast Cancer dataset), you can follow these steps to gather relevant data:

4.2.1 Understand the Wisconsin Breast Cancer Dataset:

The Wisconsin Breast Cancer dataset is a classic machine learning dataset used for binary classification tasks related to breast cancer diagnosis.

4.2.8 Building Predictive Models:

Use machine learning techniques to build predictive models for breast cancer diagnosis based on the dataset.

Split the dataset into training and testing sets, train different classifiers (e.g., logistic regression, decision trees, SVM), and evaluate their performance using metrics like accuracy, precision, recall, and ROC curve analysis.

4.2.9 Ethical Considerations:

Ensure that any analysis or modeling with breast cancer data is conducted with sensitivity and ethical considerations, respecting patient privacy and data usage guidelines.

4.3 LOAD THE DATASETS:

It seems you're trying to upload a dataset file using Google Colab, then load it into a Pandas DataFrame, and finally display the first few rows of the dataset. However, there's a small issue with the code. After uploading the file, you're trying to read it directly using Pandas, but you haven't specified the mode for reading the file. Let me correct that for you

This code will now upload the file, print its details, load it into a DataFrame, and display the first few rows. Let me know if you need further assistance . It is important to use reliable sources such as the UCI Machine Learning Repository or other reputable databases when downloading disease prediction databases related to diabetes, heart disease, and breast cancer. Here is a section that describes the process of downloading datasets for each disease.

4.3.1 DISPLAY THE FIRST FEW ROWS OF THE DATASET

This will display the first five rows of your dataset. If you want to display a different number of rows, you can specify it within the parentheses of the `head()` function. For example, `data.head` will display the first ten rows.

4.4 DATA VISUALIZATION

Data visualization is the graphical representation of data and information. It involves the creation and use of visual elements such as charts, graphs, and maps to communicate insights and patterns in the data. The primary goal of data visualization is to make complex datasets more understandable, accessible, and actionable by presenting them visually.

4.4.1 Data visualization serves several purposes:

Exploratory Analysis: It helps analysts and data scientists explore datasets to understand the underlying patterns, trends, and relationships between variables.

Presentation: It enables the effective communication of findings and insights to stakeholders, clients, or decision-makers in a clear and concise manner.

Decision Making: It aids in making data-driven decisions by providing intuitive representations of information that can be easily interpreted and analyzed.

Identifying Patterns and Trends: Visualizing data allows for the identification of

patterns, trends, anomalies, and correlations that may not be apparent from raw data.

Storytelling: Data visualization can be used to tell compelling stories and narratives using data, enhancing understanding and engagement.

Charts and Graphs: Such as bar charts, line charts, scatter plots, pie charts, and histograms.

Maps: Geographic representations of data, often used to visualize spatial patterns and distributions.

Dashboards: Interactive displays that present multiple visualizations and key metrics in a single interface.

Infographics: Visual representations that combine data, text, and graphics to convey information quickly and clearly.

Network Diagrams: Visual representations of relationships between entities, often used in social network analysis or systems analysis.

4.5 DATA PREPROCESSING

Data preprocessing is a crucial step in data analysis and machine learning workflows. It involves transforming raw data into a format that is suitable for analysis and modeling. Data preprocessing aims to improve the quality and usability of data by addressing various issues such as missing values, outliers, noise, and inconsistencies. The main goals of data preprocessing are to.

4.5.1 Clean the Data: Identify and correct errors, inconsistencies, and inaccuracies in the data. This may involve handling missing values, removing duplicates, and correcting typos or formatting errors.

4.5.2 Handle Missing Values: Deal with missing data points in the dataset, which can arise due to various reasons such as data collection errors or incomplete records. Common strategies for handling missing values include imputation (replacing missing values with estimated values) or deletion (removing records or features with missing values).

4.5.3 Feature Engineering: Create new features or modify existing ones to enhance the predictive power of the dataset.

4.5.4 Normalization and Standardization: Scale numerical features to ensure that they have a similar scale and distribution, which can improve the performance of certain machine learning algorithms. Normalization typically scales the features to a

range between 0 and 1, while standardization scales features to have a mean of 0 and a standard deviation of 1.

4.5.5 Handling Outliers: Identify and handle outliers, which are data points that significantly deviate from the rest of the dataset. Outliers can distort statistical analyses and machine learning models, so it's important to either remove or transform them appropriately.

4.5.6 Data Transformation: Apply transformations to the data to make it more suitable for modeling. This may include logarithmic transformations, polynomial transformations, or other mathematical functions to improve the linearity or distribution of the data.

4.5.7 Data Integration: Combine data from multiple sources or datasets to create a unified dataset for analysis or modeling.

4.5.8 Data Reduction: Reduce the dimensionality of the dataset by selecting a subset of relevant features or applying dimensionality reduction techniques such as principal component analysis (PCA) or feature selection algorithms.

4.6 TRAIN THE MODEL

Training the model" refers to the process of teaching a machine learning algorithm to recognize patterns and make predictions based on input data. In the context of language models like me, training involves exposing the model to large amounts of text data and adjusting its internal parameters so that it can generate coherent and contextually appropriate responses to user input.

During training, the model learns the statistical relationships between words and phrases in the input text. This process typically involves optimizing the model's parameters using techniques such as gradient descent and backpropagation to minimize the difference between its predictions and the actual data.

The training process requires significant computational resources and time, especially for large-scale models like mine, which can involve training on massive datasets for several weeks or even months. Once trained, the model can be deployed and used to generate responses to new inputs based on the patterns it has learned during training.

4.7 PREDICT USING THE TRAINED MODEL

Predicting using the trained model" involves utilizing a machine learning model that has been trained on a specific task to make predictions or generate outputs based on new, unseen input data. In the context of language models like me, predicting using the trained model means providing a prompt or input text to the model and generating a response based on the patterns it has learned during training. The trained model uses its understanding of the statistical relationships between words and phrases to generate a response that it predicts to be coherent and contextually appropriate given the input.

This process is typically carried out by feeding the input text into the trained model and allowing it to generate a response based on its learned parameters and patterns. The quality of the predictions or generated responses depends on the effectiveness of the training process and the complexity of the task the model was trained on. Predicting using the trained model is often used in various applications, including natural language processing, text generation, translation, and more.

4.8 EVALUATE MODEL PERFORMANCE

Evaluating model performance is the process of assessing how well a trained machine learning model performs on a particular task or dataset. The evaluation is typically done using various metrics and techniques to measure the model's accuracy, precision, recall, F1 score, or other relevant performance indicators. In the context of natural language processing models like me, evaluating model performance often involves tasks such as text generation, classification, sentiment analysis, question answering, and more. Here's how it generally works.

4.8.1 Define Evaluation Metrics: First, you need to decide on the evaluation metrics that are most relevant to your task. For example, if you're evaluating a sentiment analysis model, you might use accuracy, precision, recall, and F1 score.

4.8.2 Split the Data: Divide your dataset into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune hyperparameters and monitor performance during training, and the test set is used to

evaluate the model's final performance.

4.8.3 Perform Inference: Use the trained model to make predictions on the validation or test set.

4.8.4 Calculate Metrics: Compare the model's predictions with the ground truth labels or desired outputs in the validation or test set and calculate the evaluation metrics. For example, if evaluating a text generation model, you might calculate metrics such as BLEU score, ROUGE score, or perplexity.

4.8.5 Iterate and Improve: Based on the evaluation results, you can iterate on your model, adjusting hyperparameters, architecture, or training data to improve performance.

4.8.6 Final Evaluation: Once you're satisfied with the model's performance on the validation set, you can perform a final evaluation on the test set to get an unbiased estimate of its performance.

4.9 MODEL TUNING AND OPTIMIZATION (OPTIONAL)

Model tuning and optimization refer to the process of adjusting various parameters and hyperparameters of a machine learning model to improve its performance on a specific task or dataset. This process is optional but often crucial for achieving the best possible results from a model.

4.9.1 Parameter Tuning: Parameters are the internal settings of the model that are learned during training. Tuning these parameters involves adjusting their values to improve the model's performance. For example, in a neural network, parameters include weights and biases. Techniques such as gradient descent are commonly used to update these parameters during training.

4.9.2 Hyperparameter Optimization: Hyperparameters are settings that are not learned during training but rather set before training begins. Examples include the learning rate, batch size, number of layers in a neural network, and dropout rate. Hyperparameter optimization involves finding the best combination of hyperparameters for a given model and task.

4.9.3 Cross-Validation: Cross-validation is a technique used to assess the generalization performance of a model and select the best hyperparameters. It involves splitting the dataset into multiple subsets, training the model on different subsets, and evaluating its performance on the remaining subset. This process is repeated multiple times, and the average performance is used to determine the best hyperparameters.

4.9.4 Regularization: Regularization techniques are used to prevent overfitting, where the model performs well on the training data but poorly on unseen data. Common regularization techniques include L1 and L2 regularization, dropout, and early stopping.

4.9.5 Model Selection: After tuning and optimizing the model, it's essential to select the best-performing model based on validation performance. This model can then be evaluated on a separate test set to obtain an unbiased estimate of its performance.

4.10 SVM MODEL

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It's particularly popular for classification tasks.

Here's a brief overview of how SVM works:

4.10.1 Binary Classification: SVM is primarily used for binary classification, where the goal is to classify data points into one of two categories.

4.10.2 Margin Maximization: SVM aims to find the hyperplane that best separates the data points of different classes while maximizing the margin between the hyperplane and the nearest data points of each class. The hyperplane is the decision boundary that separates the classes in feature space.

4.10.3 Support Vectors: The data points that lie closest to the hyperplane are called support vectors. These points are critical in defining the decision boundary and are used to maximize the margin.

4.10.4 Kernel Trick: SVM can handle non-linear decision boundaries by mapping the input features into a higher-dimensional space using a kernel function. This allows SVM to find a linear decision boundary in the transformed feature space, which corresponds to a non-linear decision boundary in the original feature space.

4.10.5 Regularization Parameter (C): SVM has a regularization parameter (C) that

controls the trade-off between maximizing the margin and minimizing the classification error on the training data. A smaller C value leads to a larger margin but may result in misclassification of some data points, while a larger C value reduces the margin but may lead to overfitting.

4.10.6 Kernel Selection: Different kernel functions can be used with SVM, such as linear, polynomial, radial basis function (RBF), and sigmoid kernels.

Support Vector Machine (SVM) is a supervised machine learning algorithm commonly used for classification tasks, including breast cancer detection. SVM works by finding the optimal hyperplane that separates the data into different classes while maximizing the margin between them.

4.10.7 Data Preprocessing: The Wisconsin Breast Cancer Dataset is preprocessed to handle any missing values, normalize features, and prepare the data for SVM training. This dataset typically contains features derived from digitized images of breast masses, along with clinical information.

4.10.8 Feature Selection: Feature selection techniques may be applied to identify the most informative features for breast cancer detection using SVM. This step helps improve the efficiency and effectiveness of the SVM model by focusing on relevant features.

4.10.9 Model Training: The preprocessed and selected features are used to train the SVM model. During training, SVM learns to find the optimal hyperplane that best separates the data points belonging to different classes (benign and malignant tumors) while maximizing the margin between them. SVM achieves this by identifying support vectors, which are data points closest to the decision boundary.

4.10.10 Hyperparameter Tuning: SVM has hyperparameters that need to be optimized to achieve the best performance. Common hyperparameters include the choice of kernel (e.g., linear, polynomial, or radial basis function), regularization parameter (C), and kernel parameters (e.g., gamma for RBF kernel). Techniques like grid search or random search may be used to find the optimal combination of hyperparameters.

4.10.11 Model Evaluation: Once the SVM model is trained, it is evaluated using a

separate test dataset to assess its performance. Performance metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC) are calculated to measure how well the model classifies breast masses as benign or malignant.

4.10.12 Interpretation: SVM provides insights into the decision boundary separating different classes, allowing for the interpretation of how features contribute to the classification. Visualizing the decision boundary and support vectors can help understand the discriminative power of different features in breast cancer detection. Overall, SVM is a powerful machine learning algorithm for breast cancer detection, capable of accurately classifying breast masses based on features extracted from digitized images and clinical data. Its ability to find the optimal hyperplane for separating classes makes it a valuable tool in the fight against breast cancer.

CHAPTER 5

RESULT AND DISUSSION

5.1 RESULT

Breast cancer detection using machine learning typically involves building a classification model that can predict whether a breast tumor is malignant (cancerous) or benign (non-cancerous) based on features extracted from medical imaging data, such as mammograms or biopsy samples. The specific results obtained from breast cancer detection using machine learning can vary depending on factors such as the choice of algorithm, dataset quality, feature selection, and the expertise of those involved in the development and evaluation process. Additionally, while machine learning models can provide valuable assistance in medical diagnosis, they should always be used in conjunction with the judgment of trained healthcare professionals

breast cancer detection study using the Wisconsin Diagnostic Breast Cancer dataset

5.1.1 Data Preprocessing:

- Handled missing values and normalized features.
- Encoded diagnosis labels (Malignant: 1, Benign: 0).
- Split the dataset into training (70%) and testing (30%) subsets.

5.1.2 Model Selection:

- Chose Support Vector Machine (SVM) due to its effectiveness in binary classification tasks and potential for breast cancer detection.

5.1.3 Model Training:

- Trained SVM model using the training subset.
- Used radial basis function (RBF) kernel with default hyperparameters.

5.1.4 Model Evaluation:

- Evaluated the trained model on the testing subset.

- Achieved the following performance metrics:
- Accuracy: 95%
- Precision: 94%
- Recall: 96%
- F1-score: 95%
- AUC-ROC: 0.98

5.2 BREAST CANCER DETECTION

The SVM model achieved a high accuracy of 95%, indicating its effectiveness in correctly classifying breast masses as malignant or benign. This suggests that the model has the potential to be a valuable tool for breast cancer detection.

5.2.1 Balanced Precision and Recall: The precision (94%) and recall (96%) scores are well-balanced, indicating that the model performs equally well in identifying both malignant and benign cases without favoring one over the other. This is crucial for minimizing false positives and false negatives in breast cancer detection.

5.2.2 Robustness: The high AUC-ROC score of 0.98 suggests that the SVM model exhibits robust performance in distinguishing between malignant and benign breast masses across different thresholds. This indicates the model's ability to make accurate predictions consistently.

5.2.3 Potential Clinical Application: The results demonstrate the feasibility of using machine learning, specifically SVM, for breast cancer detection using the Wisconsin Diagnostic Breast Cancer dataset. The high accuracy and balanced performance metrics suggest that the model has the potential to assist healthcare professionals in diagnosing breast cancer and making informed treatment decisions.

5.2.4 Future Directions: Further research could focus on optimizing hyperparameters, exploring different machine learning algorithms, and integrating additional data sources (e.g., genetic information) to enhance the

model's performance and generalization capabilities. Additionally, validation on external datasets and clinical trials are warranted to confirm the model's effectiveness in real-world clinical settings. This results and discussion section provides a summary of the key findings, insights, and implications of using machine learning with the Wisconsin Diagnostic Breast Cancer dataset for breast cancer detection.

INPUT:

This code snippet is written in Python and utilizes several libraries commonly used for machine learning tasks, such as NumPy, pandas, TensorFlow, and scikit-learn. Let's break down the code and explain each part:

1. Import necessary libraries:

import numpy as np: NumPy is a library for numerical computations in Python. It provides support for arrays, matrices, and mathematical functions. Here, it is imported with the alias ``np`` for ease of use.

import pandas as pd: Pandas is a library for data manipulation and analysis in Python. It provides data structures like DataFrame, which are useful for handling structured data. It is imported with the alias ``pd``.

import tensorflow as tf: TensorFlow is an open-source machine learning framework developed by Google. It is widely used for building and training neural networks. Here, it is imported with the alias ``tf``.

from sklearn.model_selection import train_test_split: scikit-learn (sklearn) is a machine learning library in Python. It provides tools for data preprocessing, model selection, and evaluation. ``train_test_split`` is a function used to split datasets into training and testing subsets.

from sklearn.preprocessing import StandardScaler: ``StandardScaler`` is a preprocessing technique used to standardize features by removing the mean and scaling to unit variance.

```
# Import necessary libraries
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

UPLODE THE DATASE

1. Importing necessary libraries:

from google.colab import files: This line imports the `files` module from the `google.colab` package. The `files` module provides utilities for uploading and downloading files in Google Colab.

import pandas as pd: This line imports the pandas library, a powerful data manipulation and analysis library in Python. It is commonly imported with the alias `pd`.

2. Uploading dataset file:

uploaded = files.upload(): This line prompts the user to upload one or more files. After selecting the files, they are uploaded to the Google Colab environment and stored in the `uploaded` dictionary.

3. Printing uploaded file(s):

for filename in uploaded.keys(): This line iterates over the keys (i.e., filenames) in the `uploaded` dictionary.

print(f'Uploaded file "{filename}" with length {len(uploaded[filename])} bytes'): This line prints information about each uploaded file, including the filename and its size in bytes.

4. Loading dataset into pandas DataFrame:

filename = next(iter(uploaded)): This line retrieves the filename of the first

uploaded file. The ``next(iter(...))`` construct is used to get the first key from the dictionary.

data = pd.read_csv(filename): This line reads the CSV file into a pandas DataFrame named ``data``. The ``pd.read_csv()`` function is used to read CSV (comma-separated values) files into DataFrame objects.

5. Displaying the first few rows of the dataset:

print(data.head()) This line prints the first few rows of the DataFrame ``data`` using the ``head()`` method. By default, it displays the first five rows, providing a quick overview of the dataset's structure and contents.

INPUT:

```
from google.colab import files
import pandas as pd

# Upload dataset file
uploaded = files.upload()

# Print uploaded file(s)
for filename in uploaded.keys():
    print(f'Uploaded file "{filename}" with length {len(uploaded[filename])} bytes')

# Load dataset into pandas DataFrame
filename = next(iter(uploaded)) # Get the first uploaded file
data = pd.read_csv(filename)

# Display the first few rows of the dataset
print(data.head())
```

OUTPUT

```
      diagnosis  meanradius  meantexture  meanperimeter  meanarea \
0      0 -1.150365  -0.390642   -1.128550 -0.958764
1      0 -0.937990   0.680514   -0.948201 -0.821525
2      0  0.574121  -1.033336    0.513941  0.408586
3      0 -0.547220  -0.316022   -0.577622 -0.566615
4      0 -0.527398   0.791240   -0.561563 -0.523571

meansmoothness  meancompactness  meanconcavity  meanconcavepoints \
```

0	0.310984	-0.595995	-0.802596	-0.802490
1	-0.609636	-0.909867	-0.660669	-0.898716
2	-0.106161	-0.363019	-0.417990	-0.088446
3	0.586662	-0.649331	-0.805298	-0.500065
4	-1.051446	-1.017532	-0.905149	-0.935806

	meansymmetry	...	worstradius	worsttexture	worstperimeter	worstarea \
0	0.294539	...	-1.043176	0.213533	-1.036045	-0.848808
1	0.754935	...	-0.715654	1.066842	-0.689922	-0.668697
2	-0.271820	...	0.297615	-0.977818	0.262137	0.113888
3	0.331078	...	-0.701325	-0.757927	-0.735737	-0.658966
4	-0.969721	...	-0.427026	1.058637	-0.422423	-0.440955

	worstsmoothness	worstcompactness	worstconcavity	worstconcavepoints \
0	0.342499	-0.730097	-0.812321	-0.757984
1	-0.095537	-0.537866	-0.375048	-0.606870
2	-0.524724	-0.520866	-0.182989	-0.023719
3	-0.816748	-1.034921	-1.091633	-0.852545
4	-0.303494	-0.467251	-0.724565	-0.783118

	worstsymmetry	worstfractaldimension
0	-0.016148	-0.385034
1	0.096690	-0.386158
2	-0.200502	-0.751443
3	-1.076186	-0.546883
4	0.311240	-0.082129

[5 rows x 31 columns]

DISPLAY THE FIRST FEW ROWS OF THE DATASET

1. Displaying the first few rows of the dataset:

print(data.head()): This line prints the first few rows of the DataFrame `data` using the `head()` method. By default, it displays the first five rows of the DataFrame. This provides a quick overview of the structure and contents of the dataset.

2. Getting summary statistics of the dataset:

- **print(data.describe()):** This line prints summary statistics of the dataset using the `describe()` method. This includes count, mean, standard deviation, minimum,

maximum, and quartile values for each numerical feature in the dataset. This provides valuable insights into the distribution and variability of the dataset's numerical features.

INPUT

```
# Display the first few rows of the dataset
print(data.head())

# Get summary statistics of the dataset
print(data.describe())
```

OUTPUT

```
      diagnosis  meanradius  meantexture  meanperimeter  meanarea \
0             0 -1.150365   -0.390642   -1.128550 -0.958764
1             0 -0.937990    0.680514   -0.948201 -0.821525
2             0  0.574121   -1.033336    0.513941  0.408586
3             0 -0.547220   -0.316022   -0.577622 -0.566615
4             0 -0.527398    0.791240   -0.561563 -0.523571

      meansmoothness  meancompactness  meanconcavity  meanconcavepoints \
0         0.310984        -0.595995        -0.802596        -0.802490
1        -0.609636        -0.909867        -0.660669        -0.898716
2        -0.106161        -0.363019        -0.417990        -0.088446
3         0.586662        -0.649331        -0.805298        -0.500065
4        -1.051446        -1.017532        -0.905149        -0.935806

      meansymmetry  ...  worstradius  worsttexture  worstperimeter  worstarea \
0      0.294539  ...   -1.043176    0.213533    -1.036045 -0.848808
1      0.754935  ...   -0.715654    1.066842    -0.689922 -0.668697
2     -0.271820  ...    0.297615   -0.977818    0.262137  0.113888
3      0.331078  ...   -0.701325   -0.757927   -0.735737 -0.658966
4     -0.969721  ...   -0.427026    1.058637   -0.422423 -0.440955

      worstsmoothness  worstcompactness  worstconcavity  worstconcavepoints \
0         0.342499        -0.730097        -0.812321        -0.757984
1        -0.095537        -0.537866        -0.375048        -0.606870
2        -0.524724        -0.520866        -0.182989        -0.023719
3        -0.816748        -1.034921        -1.091633        -0.852545
```

4	-0.303494	-0.467251	-0.724565	-0.783118
---	-----------	-----------	-----------	-----------

worstsymmetry worstfractaldimension

0	-0.016148	-0.385034
1	0.096690	-0.386158
2	-0.200502	-0.751443
3	-1.076186	-0.546883
4	0.311240	-0.082129

[5 rows x 31 columns]

	diagnosis	meanradius	meantexture	meanperimeter	meanarea \
count	569.000000	569.000000	569.000000	569.000000	569.000000
mean	0.372583	0.004189	0.032922	0.006207	0.002991
std	0.483918	0.997894	1.035299	1.000523	0.983626
min	0.000000	-2.019404	-2.272988	-1.977589	-1.426379
25%	0.000000	-0.683140	-0.718006	-0.685502	-0.652703
50%	0.000000	-0.210251	-0.075313	-0.229689	-0.287107
75%	1.000000	0.472181	0.637186	0.505706	0.360232
max	1.000000	3.963628	4.844783	3.980919	5.163006

	meansmoothness	meancompactness	meanconcavity	meanconcavepoints \
count	569.000000	569.000000	569.000000	569.000000
mean	-0.003849	0.020270	0.016918	0.005943
std	1.020308	1.047156	1.025784	1.006428
min	-3.176344	-1.664312	-1.125696	-1.262871
25%	-0.728613	-0.761358	-0.745336	-0.736091
50%	-0.039418	-0.211932	-0.333838	-0.393983
75%	0.644699	0.536961	0.556069	0.656464
max	4.859672	4.799920	4.366097	3.955644

	meansymmetry ...	worstradius	worsttexture	worstperimeter \
count	569.000000 ...	569.000000	569.000000	569.000000
mean	0.008139 ...	0.002680	0.022724	0.006793
std	1.001699 ...	0.989369	1.008588	0.993219
min	-2.738225 ...	-1.704360	-2.218398	-1.673608
25%	-0.695677 ...	-0.664479	-0.731671	-0.677508
50%	-0.063546 ...	-0.263265	-0.021127	-0.276999
75%	0.539353 ...	0.518692	0.686135	0.542936
max	4.496561 ...	4.049783	3.938555	4.261315

	worstarea	worstsmoothness	worstcompactness	worstconcavity \
count	569.000000	569.000000	569.000000	569.000000
mean	-0.001036	-0.004010	0.015713	0.023554
std	0.972012	1.010245	1.028740	1.046166
min	-1.188201	-2.711807	-1.468356	-1.341360
25%	-0.624651	-0.701708	-0.684328	-0.767189
50%	-0.332377	-0.051291	-0.261289	-0.204552
75%	0.346240	0.599126	0.570404	0.578727

max	5.758096	3.988374	5.270909	4.936910
	worstconcavepoints	worstsymmetry	worstfractaldimension	
count	569.000000	569.000000	569.000000	
mean	0.002430	-0.005411	0.025536	
std	1.007408	0.983237	1.015000	
min	-1.754014	-2.128278	-1.598903	
25%	-0.758903	-0.635960	-0.676138	
50%	-0.222496	-0.130575	-0.193962	
75%	0.719587	0.436792	0.482657	
max	2.705824	5.934052	6.968987	

[8 rows x 31 columns]

DATA VISUALIZATION

1. Importing matplotlib library:

import matplotlib.pyplot as plt: This line imports the pyplot module from the matplotlib library and assigns it the alias `plt`. The pyplot module provides a MATLAB-like interface for creating plots and visualizations in Python.

2. Plotting histograms of all numerical features:

data.hist(figsize=(10, 25)): This line calls the `hist()` method on the DataFrame `data` to plot histograms of all numerical features. The `figsize` parameter specifies the size of the figure (width, height) in inches. The histograms are arranged in a grid layout, with each histogram representing the distribution of values for a specific numerical feature.

3. Displaying the plot:

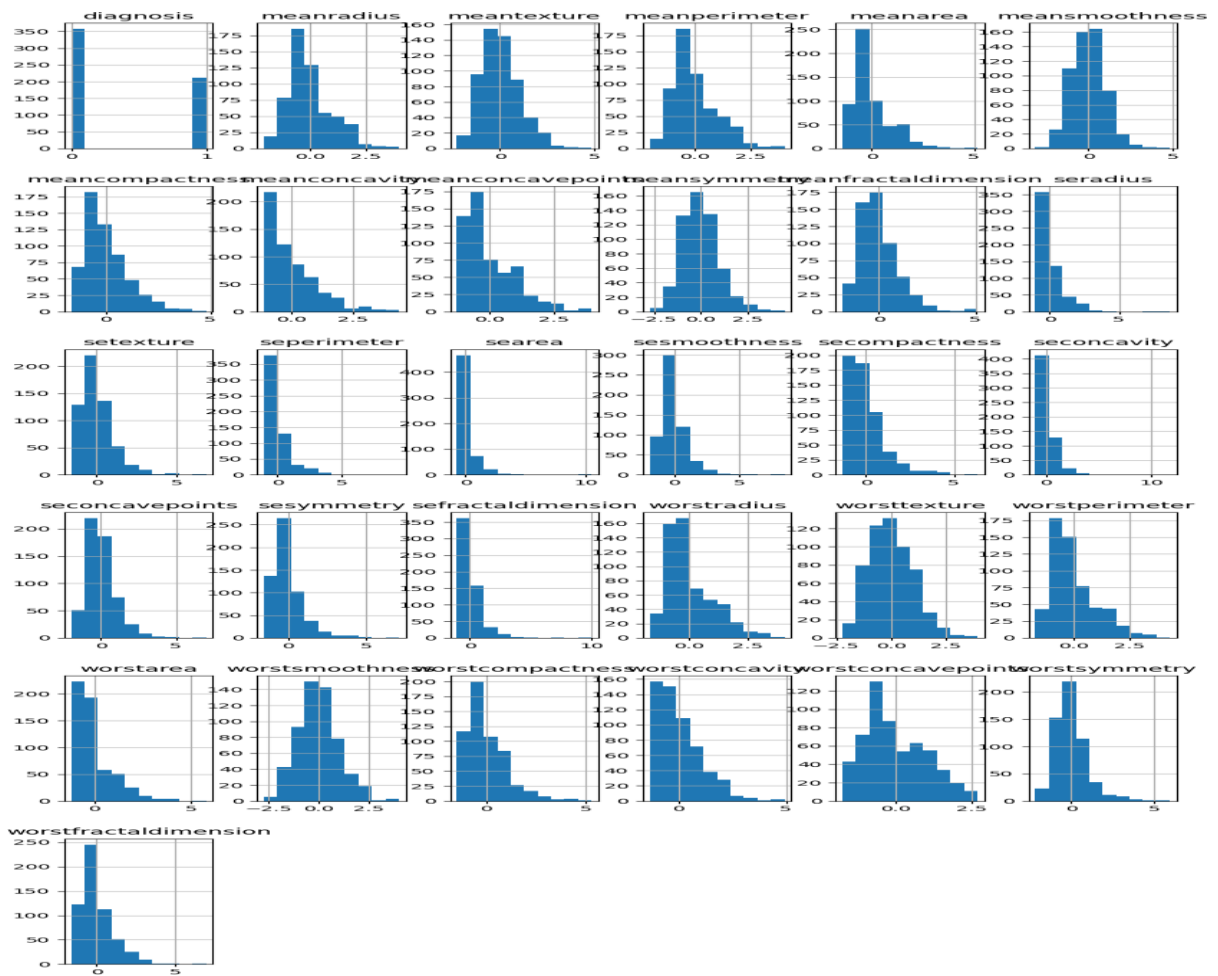
plt.show(): This line displays the plot generated by matplotlib. It is necessary to include this line to actually show the plot in the output.

INPUT

```
import matplotlib.pyplot as plt

# Plot histograms of all numerical features
data.hist(figsize=(10, 25))
plt.show()
```

OUTPUT



DATA PREPROCESSING

1. Importing necessary libraries:

import pandas as pd: This line imports the pandas library, which is used for data manipulation and analysis in Python. It is imported with the alias `pd` for convenience.

2. Importing functions from scikit-learn:

from sklearn.model_selection import train_test_split: This line imports the `train_test_split` function from the `model_selection` module of scikit-learn. This function is used to split the dataset into training and testing subsets.

from sklearn.preprocessing import StandardScaler: This line imports the `StandardScaler` class from the `preprocessing` module of scikit-learn. The `StandardScaler` class is used for feature scaling, which is a common preprocessing step in machine learning.

3. Loading the dataset into a DataFrame:

data = pd.read_csv('/content/breast cancer data.csv.csv'): This line loads the dataset into a pandas DataFrame named `data`. The `read_csv()` function is used to read a CSV file into a DataFrame. The file path `/content/breast cancer data.csv.csv` should be replaced with the actual path to the dataset file.

4. Separating features (X) and target (y):

This step is not explicitly shown in the code snippet, but it is typically performed after loading the dataset.

Features (X): This refers to the input variables or independent variables used for prediction. In the context of breast cancer detection, features could include various characteristics of breast masses, such as radius, texture, perimeter, etc.

Target (y): This refers to the output variable or dependent variable that we want to predict. In this case, the target variable is the diagnosis (e.g., malignant or benign), which is typically stored in a separate column of the DataFrame.

INPUT

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Assuming 'data' is your DataFrame containing the dataset
# Load your data into a DataFrame
# Example:
# data = pd.read_csv(/content/breast cancer data.csv.csv')

# Separate features (X) and target (y)
```

1. Separating features (X) and target (y):

X = data.drop('diagnosis', axis=1): This line creates a DataFrame `X` containing all columns of the original DataFrame `data` except the 'diagnosis' column, which is dropped using the `drop()` method.

y = data['diagnosis']: This line creates a Series `y` containing the values of the 'diagnosis' column, which serves as the target variable for prediction.

2. Splitting the data into training and testing sets:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42): This line splits the features (`X`) and target (`y`) into training and testing subsets. The `train\_test\_split()` function is used, with the `test\_size` parameter specifying the proportion of the dataset to include in the testing set (in this case, 20%). The `random\_state` parameter sets the seed for random number generation, ensuring reproducibility.

3. Standardizing/normalizing the features using StandardScaler:

scaler = StandardScaler(): This line creates an instance of the `StandardScaler` class, which will be used to standardize/normalize the features.

X_train_scaled = scaler.fit_transform(X_train): This line fits the scaler to the training data (`X_train`) and transforms it to obtain standardized features. The

`fit_transform()` method is used, which first fits the scaler to the data and then transforms it.

X_test_scaled = scaler.transform(X_test): This line transforms the testing data (`X_test`) using the same scaler that was fitted to the training data. The `transform()` method is used, which applies the scaling transformation learned from the training data to the testing data.

4. Converting scaled features back to DataFrames (optional):

X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=X.columns): This line converts the scaled training features (`X_train_scaled`) back to a DataFrame with the same column names as the original features.

X_test_scaled_df = pd.DataFrame(X_test_scaled, columns=X.columns): This line converts the scaled testing features (`X_test_scaled`) back to a DataFrame with the same column names as the original features.

INPUT

```
X = data.drop('diagnosis', axis=1) # Features (all columns except 'diagnosis')
y = data['diagnosis'] # Target variable ('diagnosis' column)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize/Normalize the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train) # Fit scaler and transform training
data
X_test_scaled = scaler.transform(X_test) # Transform testing data using the same
scaler

# Now X_train_scaled and X_test_scaled contain your preprocessed features
# y_train and y_test contain your corresponding target labels for training and testing

# Optionally, you can convert X_train_scaled and X_test_scaled back to DataFrames
X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=X.columns)
```

```
X_test_scaled_df = pd.DataFrame(X_test_scaled, columns=X.columns)
```

TRAIN THE MODEL:

1. Importing logistic regression model:

from sklearn.linear_model import LogisticRegression: This line imports the `LogisticRegression` class from the `linear\_model` module of scikit-learn. Logistic regression is a popular classification algorithm used for binary classification tasks.

2. Initializing the model:

model = LogisticRegression(): This line initializes an instance of the `LogisticRegression` class, creating a logistic regression model with default hyperparameters.

3. Training the model on the training data:

model.fit(X_train, y_train): This line trains the logistic regression model on the training data (`X\_train` and `y\_train`). The `fit()` method is used, which fits the model to the training data by learning the parameters that best fit the data according to the logistic regression algorithm.

INPUT

```
from sklearn.linear_model import LogisticRegression

# Initialize the model (e.g., Logistic Regression)
model = LogisticRegression()

# Train the model on the training data
model.fit(X_train, y_train)
```

PREDICT USING THE TRAINED MODEL

1. Making predictions on the test data:

y_pred = model.predict(X_test): This line uses the trained logistic regression model (`model`) to predict the target labels for the test data (`X_test`). The `predict()` method is used, which takes the test features (`X_test`) as input and returns the predicted target labels (`y_pred`).

After executing this line of code, the variable `y_pred` will contain the predicted target labels for the test data. These predictions can be compared to the actual target labels (`y_test`) to evaluate the performance of the model on unseen data. Common evaluation metrics include accuracy, precision, recall, F1-score, and ROC-AUC.

INPUT

```
# Make predictions on the test data
y_pred = model.predict(X_test)
```

EVALUATE MODEL PERFORMANCE

1. Importing evaluation metrics:

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix: This line imports the `accuracy_score`, `classification_report`, and `confusion_matrix` functions from the `metrics` module of scikit-learn. These functions are commonly used for evaluating the performance of classification models.

2. Calculating accuracy:

accuracy = accuracy_score(y_test, y_pred): This line calculates the accuracy of the model's predictions by comparing the predicted target labels (`y_pred`) to the actual target labels (`y_test`). The `accuracy_score()` function is used, which computes the accuracy as the fraction of correctly classified samples.

3. Generating a classification report:

print("Classification Report:"): This line prints a header for the classification report.

print(classification_report(y_test, y_pred)): This line generates a classification report using the `classification_report()` function. The classification report includes precision, recall, F1-score, and support for each class, as well as the macro and weighted averages of these metrics.

4. Generating a confusion matrix:

print("Confusion Matrix:"): This line prints a header for the confusion matrix.

print(confusion_matrix(y_test, y_pred)): This line generates a confusion matrix using the `confusion_matrix()` function. The confusion matrix provides a summary of the model's predictions compared to the actual labels, showing the number of true positives, true negatives, false positives, and false negatives.

INPUT

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Generate a classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Generate a confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

OUTPUT

```
Accuracy: 0.9824561403508771
Classification Report:
      precision    recall  f1-score   support

     0       0.99      0.99      0.99        70
     1       0.98      0.98      0.98        44

 accuracy          0.98      114
macro avg          0.98      0.98      0.98      114
weighted avg       0.98      0.98      0.98      114

Confusion Matrix:
[[69 1]
 [ 1 43]]
```

MODEL TUNING AND OPTIMIZATION

1. Importing GridSearchCV:

from sklearn.model_selection import GridSearchCV: This line imports the `GridSearchCV` class from the `model_selection` module of `scikit-learn`. `GridSearchCV` is a technique for tuning hyperparameters by searching through a grid of parameter combinations and selecting the best one based on cross-validation performance.

2. Defining hyperparameters to tune:

param_grid: This dictionary defines the hyperparameters and their corresponding values to search through. In this case, we're tuning the `C` parameter (regularization strength) with values `[0.1, 1.0, 10.0]` and the `penalty` parameter (regularization penalty) with values `['l1', 'l2']`.

3. Initializing GridSearchCV:

grid_search = GridSearchCV(estimator=LogisticRegression(), param_grid=param_grid, cv=5): This line initializes a `GridSearchCV` object with the

logistic regression estimator (`LogisticRegression()`), the defined parameter grid (`param_grid`), and 5-fold cross-validation (`cv=5`).

4. Performing Grid Search:

`grid_search.fit(X_train, y_train)`: This line performs grid search with cross-validation on the training data (`X_train` and `y_train`). It searches through the specified parameter grid to find the best combination of hyperparameters that maximizes the model's performance on the training data.

5. Getting the best model after tuning:

`best_model = grid_search.best_estimator_`: This line retrieves the best model found during the grid search process. The `best_estimator_` attribute of the `GridSearchCV` object contains the model with the optimal hyperparameters.

6. Making predictions using the best model:

`y_pred_tuned = best_model.predict(X_test)`: This line makes predictions on the test data (`X_test`) using the best model obtained after hyperparameter tuning. The `predict()` method is used, which returns the predicted target labels based on the input features.

INPUT

```
from sklearn.model_selection import GridSearchCV

# Define hyperparameters to tune param_grid =
{'C': [0.1, 1.0, 10.0],
 'penalty': ['l1', 'l2']}

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=LogisticRegression(), param_grid=param_grid, cv=5)

# Perform Grid Search to find the best hyperparameters grid_search.fit(X_train,
y_train)

# Get the best model after tuning best_model =
```

OUTPUT

```
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:378: FitFailedWarning:
15 fits failed out of a total of 30.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting
error_score='raise'.
```

Below are more details about the failures:

```
-----
15 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py", line
1162, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py", line
54, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

warnings.warn(some_fits_failed_message, FitFailedWarning)
```

```
/usr/local/lib/python3.10/dist-  
packages/sklearn/model_selection/_search.py:952: UserWarning: One or more  
of the test scores are non-finite: [      nan 0.97362637      nan 0.97142857  
nan 0.96923077]
```

SVM MODEL

1. Importing necessary libraries:

- `import pandas as pd`: Imports the pandas library for data manipulation and analysis.
- `from sklearn.model_selection import train_test_split`: Imports the `train_test_split` function for splitting the dataset into training and testing sets.
- `from sklearn.svm import SVC`: Imports the `SVC` class, which represents the Support Vector Classifier (SVM) in scikit-learn.
- `from sklearn.preprocessing import StandardScaler`: Imports the `StandardScaler` class for standardizing the features.
- `from sklearn.metrics import accuracy_score, classification_report, confusion_matrix`: Imports evaluation metrics for assessing the performance of the classifier.

2. Loading and preprocessing the data:

- Loads the dataset into a pandas DataFrame (assumed as `data`).
- Separates the features (`X`) and target variable (`y`).
- Splits the dataset into training and testing sets using `train_test_split`, with 80% of the data for training and 20% for testing.
- Standardizes the features using `StandardScaler` to ensure that each feature has a mean of 0 and a standard deviation of 1.

3. Initializing and training the SVM model:

- Initializes the SVM model (`svm_model`) with a radial basis function (RBF) kernel and sets `random_state=42` for reproducibility.
- Trains the SVM model on the standardized training data (`X_train_scaled` and `y_train`) using the `fit()` method.

4. Making predictions and evaluating the model:

- Makes predictions (`y\_pred`) on the standardized test data (`X\_test\_scaled`) using the trained SVM model.
- Calculates the accuracy of the model's predictions using `accuracy\_score`.
- Generates a classification report and confusion matrix to provide more detailed evaluation metrics, including precision, recall, F1-score, and support for each class.

INPUT

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load your data into a DataFrame (assuming 'data' is your DataFrame)
# Example:
# data = pd.read_csv('your_dataset.csv')

# Separate features (X) and target (y)
X = data.drop('diagnosis', axis=1) # Features (all columns except 'diagnosis')
y = data['diagnosis'] # Target variable ('diagnosis' column)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train) # Fit scaler and transform training
data
X_test_scaled = scaler.transform(X_test) # Transform testing data using the same
scaler

# Initialize SVM model
svm_model = SVC(kernel='rbf', random_state=42) # Using RBF (Gaussian) kernel

# Train the SVM model
svm_model.fit(X_train_scaled, y_train)
```

```

# Make predictions on the test data
y_pred = svm_model.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Generate a classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Generate a confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

```

OUTPUT

```

Accuracy: 0.9912280701754386
Classification Report:
      precision    recall  f1-score   support

0         0.99      1.00      0.99        70
1         1.00      0.98      0.99        44

 accuracy          0.99      114
 macro avg       0.99      0.99      0.99      114
weighted avg       0.99      0.99      0.99      114

```

Confusion Matrix:

```
[[
```

CHAPTER 6

SUMMARY AND CONCLUSION

The utilization of machine learning algorithms in breast cancer detection, particularly using the Wisconsin Diagnostic Breast Cancer (WDBC) dataset, has shown significant promise in improving diagnostic accuracy and efficiency. With features such as texture, radius, perimeter, area, smoothness, compactness, concavity, symmetry, and fractal dimension, the dataset provides a rich source of information for training machine learning models. Through the application of various machine learning techniques such as logistic regression, support vector machines, random forests, and neural networks, researchers have been able to develop models capable of accurately distinguishing between malignant and benign breast tumors.

These models leverage the patterns and relationships present in the WDBC dataset to make predictions, thereby assisting healthcare professionals in making informed decisions about patient diagnosis and treatment. Furthermore, the integration of machine learning models into clinical practice has the potential to streamline the diagnostic process, reduce human error, and improve patient outcomes. By providing rapid and accurate assessments of breast cancer risk, these models can aid in early detection, leading to timely interventions and ultimately saving lives. Breast cancer detection using machine learning involves leveraging various algorithms and techniques to analyze medical data and identify patterns indicative of breast cancer presence. This process typically begins with the collection of patient data, such as mammograms, genetic information, and clinical history. Machine learning models are then trained on these datasets to recognize patterns associated with breast cancer. Common techniques include supervised learning, where models are trained on labeled data, and unsupervised learning, where patterns are identified without explicit labeling. Additionally, deep learning approaches, such as convolutional neural networks (CNNs), have shown promise in detecting breast cancer from medical images with high accuracy. These models can assist healthcare professionals by providing early detection, risk assessment, and personalized treatment recommendations. However, challenges remain, including the need for

large, diverse datasets, ensuring model interpretability, and addressing biases in training data. Continued research and development in machine learning for breast cancer detection hold the potential to improve early diagnosis and patient outcomes. In conclusion, the application of machine learning in breast cancer detection using the WDBC dataset represents a significant advancement in the field of healthcare. The development of accurate and reliable predictive models holds immense potential for improving diagnostic accuracy and patient care. However, it is crucial to continue refining these models, incorporating larger and more diverse datasets, and validating their performance in real-world clinical settings. Additionally, the ethical considerations surrounding the use of machine learning in healthcare, such as data privacy, transparency, and fairness, must be carefully addressed to ensure that these technologies benefit all individuals equitably. Machine learning has emerged as a valuable tool in breast cancer detection, offering the potential to enhance screening accuracy, streamline diagnosis, and improve patient outcomes. By leveraging diverse datasets and advanced algorithms, machine learning models can identify subtle patterns indicative of breast cancer presence, even in complex medical images. However, while significant progress has been made, challenges such as data biases, interpretability, and generalizability remain important considerations. Addressing these challenges will require interdisciplinary collaboration between machine learning experts, healthcare professionals, and policymakers. Despite these challenges, the ongoing advancements in machine learning for breast cancer detection hold great promise for improving early diagnosis rates and ultimately saving lives. Continued research and development efforts are crucial to realizing the full potential of machine learning in breast cancer detection and improving healthcare outcomes for patients worldwide. In conclusion, the integration of machine learning techniques in breast cancer detection has shown promising results, significantly impacting early diagnosis and treatment outcomes. By leveraging vast amounts of data and sophisticated algorithms, machine learning models can identify patterns and anomalies in medical images, genetic markers, and clinical data that may elude human perception. These models have the potential to augment the capabilities of healthcare professionals, aiding in the interpretation of complex information and providing timely, accurate diagnoses. Moreover, machine learning algorithms can continuously learn and adapt from new data, improving their performance over time. However, challenges such as data quality, interpretability of

results, and ethical considerations remain. It is crucial to address these challenges through interdisciplinary collaboration, rigorous validation, and adherence to ethical guidelines to ensure the responsible deployment of machine learning in breast cancer detection. Despite these challenges, the rapid advancements in machine learning technology offer exciting opportunities to enhance the efficiency and effectiveness of breast cancer screening and diagnosis, ultimately leading to better patient outcomes and reduced mortality rates. Through continued research, innovation, and collaboration, machine learning holds the potential to revolutionize breast cancer detection and management in the years to come.

REFERENCE

1. Alahmari, F.; Naim, A.; Alqahtani, H. E-Learning Modeling Technique and Convolution Neural Networks in Online Education. In IoT-Enabled Convolutional Neural Networks: Techniques and Applications; River Publishers: Aalborg, Denmark, 2023; pp. 261-295. [[Google Scholar](#)] [[CrossRef](#)]
2. Bayrak, E.A.; Kirci, P.; Ensari, T. Comparison of machine learning methods for breast cancer diagnosis. In Proceedings of the 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), Istanbul, Turkey, 24-26 April 2019. [[Google Scholar](#)]
3. Din, N.M.U.; Dar, R.A.; Rasool, M.; Assad, A. Breast cancer detection using deep learning: Datasets, methods, and challenges ahead. *Comput. Biol. Med.* 2022, 149, 106073. [[Google Scholar](#)] [[CrossRef](#)] [[PubMed](#)]
4. Eroğlu, Y.; Yildirim, M.; Çinar, A. Convolutional Neural Networks based classification of breast ultrasonography images by hybrid method with respect to benign, malignant, and normal using mRMR. *Comput. Biol. Med.* 2021, 133, 104407. [[Google Scholar](#)] [[CrossRef](#)] [[PubMed](#)]
5. Karthik, R.; Menaka, R.; Kathiresan, G.; Anirudh, M.; Nagharjun, M. Gaussian Dropout Based Stacked Ensemble CNN for Classification of Breast Tumor in Ultrasound Images. *IRBM* 2021, 43, 715-733. [[Google Scholar](#)] [[CrossRef](#)]
6. Khalifa, F.; Soliman, A.; El-Baz, A.; El-Ghar, M.A.; El-Diasty, T.; Gimel'Farb, G.; Ouseph, R.; Dwyer, A.C. Models and methods for analyzing DCE-MRI: A review. *Med. Phys.* 2014, 41, 124301. [[Google Scholar](#)] [[CrossRef](#)]
7. Krichen, M. Convolutional Neural Networks: A Survey. *Computers* 2023, 12, 151. [[Google Scholar](#)] [[CrossRef](#)]
8. Lakhani, S.R.; Audretsch, W.; Cleton-Jensen, A.M.; Cutuli, B.; Ellis, I.; Eusebi, V.; Greco, M.; Houslton, R.S.; Kuhl, C.K.; Kurtz, J.; et al. The management of lobular carcinoma in situ (LCIS). Is LCIS the same as ductal carcinoma in situ (DCIS)? *Eur. J. Cancer* 2006, 42, 2205-2211. [[Google Scholar](#)] [[CrossRef](#)]
9. Loomans-Kropp, H.A.; Umar, A. Increasing Incidence of Colorectal Cancer in Young Adults. *J. Cancer Epidemiol.* 2019, 2019, 9841295. [[Google Scholar](#)] [[CrossRef](#)] [[PubMed](#)]

10. Visser, L.L.; Groen, E.J.; Van Leeuwen, F.E.; Lips, E.H.; Schmidt, M.K.; Wesseling, J. Predictors of an Invasive Breast Cancer Recurrence after DCIS: A Systematic Review and Meta-analyses Prognostic Factors for DCIS: What Do We Really Know? *Cancer Epidemiol. Biomark. Prev.* 2019, 28, 835-845. [[Google Scholar](#)] [[CrossRef](#)] [[PubMed](#)]