**AI-ENABLED BREAST CANCER CLASSIFICATION SYSTEM FROM MAMMOGRAPHIC IMAGES**

**ABSTRACT**

Breast cancer is one of the most prevalent and life-threatening diseases among women worldwide. Early detection plays a crucial role in improving survival rates. Mammographic imaging is a widely used screening tool for breast cancer detection.In the existing system of breast cancer detection primarily rely on manual interpretation by radiologists, which can be time-consuming and subjective. While some computer-aided diagnosis (CAD) systems exist, they often lack the accuracy and robustness required for clinical use.The existing systems for breast cancer diagnosis suffer from limitations such as manual interpretation, low accuracy, and dependency on human expertise. There is a need for a more accurate and efficient approach that can automatically classify mammographic images with high precision, aiding in early detection and reducing the workload of radiologists. Our proposed method utilizes a machine learning approach, specifically the Extra Tree Classifier (ETC), to classify mammographic images into benign and malignant categories. We preprocess the images to extract relevant features, such as texture, shape, and intensity, and then train the ETC model on these features to accurately classify the images, the system can aid in the early detection of breast cancer, leading to better treatment outcomes.

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

Breast cancer is one of the most prevalent and fatal forms of cancer affecting women worldwide. Early detection and accurate diagnosis play a crucial role in improving treatment outcomes and increasing survival rates. Mammography, a commonly used screening tool for breast cancer, provides detailed images of the breast tissue. However, the interpretation of mammograms is challenging and often relies on the expertise of radiologists, leading to potential variations in diagnosis.

Machine learning (ML) techniques have emerged as powerful tools for automating the analysis of mammographic images, aiding in the early detection and classification of breast cancer. By leveraging large datasets of mammograms and associated clinical data, ML models can learn complex patterns and features indicative of cancerous or benign lesions, assisting healthcare professionals in making more accurate and timely diagnoses.

In the realm of medical diagnostics, the utilization of machine learning algorithms has significantly advanced the accuracy and efficiency of breast cancer detection, particularly with mammographic images. A notable approach in this domain involves the application of ETC, an ensemble learning technique renowned for its ability to handle complex datasets. The workflow typically begins with preprocessing mammographic images to enhance quality and reduce noise. Feature extraction techniques are then employed to capture relevant patterns and characteristics within the images. The extracted features serve as input for the Extra Tree model, which consists of an ensemble of decision trees. Through iterative training, the ETC learns to discern subtle patterns indicative of benign or malignant conditions. This model, once trained, demonstrates high accuracy in classifying mammograms, contributing to early and reliable breast cancer diagnosis. Regular validation and refinement of the model ensure its robust performance across diverse datasets, ultimately offering a valuable tool for healthcare professionals in the pursuit of more effective and timely breast cancer detection.

**1.2 PROBLEM STATEMENT**

The problem statement for a machine learning approach to breast cancer classification from mammographic images typically involves designing a system that can accurately classify mammograms into one of two classes: benign (non-cancerous) or malignant (cancerous). Here's a more detailed breakdown of the problem statement:

Objective: Develop a machine learning model that can assist radiologists in accurately identifying breast cancer from mammographic images, with the goal of improving diagnostic accuracy and early detection rates.

Dataset: Obtain a dataset of mammographic images along with corresponding ground truth labels indicating whether each image depicts a benign or malignant tumor. The dataset should be large enough to train a robust machine learning model and should include a diverse range of cases.

Feature Extraction: Preprocess the mammographic images and extract relevant features that can discriminate between benign and malignant tumors. These features may include texture features, shape features, and other image characteristics that are indicative of cancer.

Model Selection: Choose an appropriate machine learning algorithm for classification. This could include traditional classifier like KNN.

Model Training: Train the selected model using the preprocessed images and their corresponding labels. This involves splitting the dataset into training and validation sets, and optimizing the model parameters to maximize classification accuracy.

Evaluation: Evaluate the performance of the trained model using metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC). Additionally, assess the model's performance on unseen test data to ensure generalization**.**

Validation: Validate the model's performance using independent datasets or through cross-validation to ensure that the results are robust and not biased by the particular characteristics of the training data.

Integration and Deployment: Integrate the trained model into a user-friendly application or system that can be easily used by radiologists or healthcare professionals for breast cancer diagnosis. Ensure that the system is scalable, reliable, and compliant with relevant regulatory requirements.

Monitoring and Improvement: Continuously monitor the performance of the deployed system and gather feedback from users to identify areas for improvement.

**1.3** **RESEARCH MOTIVATION**

The motivation behind research on breast cancer classification from mammographic images stems from the pressing need to improve early detection and diagnosis of breast cancer, which is crucial for increasing survival rates and reducing mortality. Here are some key points that drive this research:

Early Detection: Mammography is one of the most effective methods for detecting breast cancer at an early stage when treatment is most effective. By analyzing mammographic images, researchers aim to develop algorithms that can accurately detect suspicious lesions or abnormalities, even at their earliest stages.

Improving Accuracy: While mammography is highly effective, it is not perfect, and there can be challenges such as false positives and false negatives. False positives can lead to unnecessary biopsies and patient anxiety, while false negatives can result in missed diagnoses. Developing more accurate classification algorithms can help reduce these errors and improve overall diagnostic accuracy.

Personalized Medicine: Breast cancer is not a single disease but comprises various subtypes with different prognoses and responses to treatment. By analyzing mammographic images and integrating clinical data, researchers aim to develop classification models that can distinguish between different subtypes of breast cancer. This information can help tailor treatment plans to individual patients, leading to better outcomes.

Automation and Efficiency: Manual interpretation of mammograms by radiologists is time-consuming and subject to human error. Automation of the classification process through machine learning algorithms can help streamline the analysis process, reduce interpretation time, and potentially improve the overall efficiency of breast cancer screening programs.

Access to Healthcare: In many parts of the world, access to trained radiologists and specialized healthcare facilities is limited. Automated classification systems can help extend the reach of breast cancer screening programs to underserved populations by providing a cost-effective and scalable solution for early detection and diagnosis.

Research and Development: Advancements in machine learning and computer vision techniques present new opportunities for innovation in breast cancer classification. Research in this area not only aims to improve current methods but also to explore novel approaches that may lead to breakthroughs in early detection and personalized treatment.

**1.4 APPLICATIONS**

**Early Detection**: Automated classification helps in the early detection of breast cancer, improving the chances of successful treatment and reducing mortality rates.

**Screening Programs**: Integrating ML into screening programs allows for more efficient analysis of a large number of mammograms, aiding healthcare professionals in prioritizing cases that require further attention.

**Reducing False Positives/Negatives**: ML models can contribute to minimizing false positives and false negatives, enhancing the accuracy of breast cancer diagnosis and reducing unnecessary interventions.

**Resource Optimization**: Automation can streamline the diagnostic process, optimizing the use of healthcare resources and improving the overall efficiency of breast cancer screening programs.

**Personalized Medicine**: ML algorithms can analyze diverse patient data, leading to more personalized and targeted treatment plans based on the specific characteristics of the breast cancer identified.

**Radiologist Support**: ML models serve as decision support tools for radiologists, providing additional insights and improving diagnostic accuracy.

**Large-Scale Data Analysis**: ML enables the analysis of vast datasets, uncovering patterns and correlations that may not be apparent through traditional methods, contributing to a deeper understanding of breast cancer.

**CHAPTER 2**

**LITERATURE SURVEY**

Meenalochini, et al.[1] proposed method involved investigating the effects of various machine learning techniques for automating mammogram image classification. This investigation involved assembling previous works that demonstrated the application of machine learning techniques to address different issues identified in various diagnostic science examinations. Additionally, this study proposed preprocessing mammogram images before they entered the classifier to achieve higher effective classification. Following the detection stage, the proposed method included segmenting the tumor region in a mammogram image.

Darweesh,et al.[2] proposed that Machine Learning-based two-level top-down hierarchical approach for breast cancer detection and classification into three classes: normal, benign, and malignant, using the Mammographic Image Analysis Society (MIAS) mammography dataset. Different data preprocessing techniques were applied before using feature extraction techniques and machine learning algorithms for classification.

Alshammari, et al.[3] proposed model incorporated these features into a classification engine to train and build the structure of the classification models. To evaluate the accuracy of the proposed system, a dataset that had not been previously seen by the model was utilized, following standard model evaluation schemes. Accordingly, in this study, it was found that various factors could affect the performance, which were addressed after experimenting with all possible approaches.

Atrey,et al.[4] proposed approach introduces a novel semi-automated multimodal classification system for breast tumors. It combines features extracted from both mammogram and ultrasound images. Initially, forty-two grayscale features are extracted from the images. Subsequently, statistical significance analysis is conducted to identify the most relevant features. These selected features are then used for classifying tumors as benign or malignant.

Avcı, et al.[5] proposed that features be extracted from the obtained ROIs. Finally, feature datasets were classified as normal/abnormal, and benign/malign (two-class classification) using Machine Learning algorithms. Test performance measures of the classification methods were examined. In both classifications made in the study, lower classification performance values were obtained when the CLAHE algorithm was used alone as a pre-processing method compared to other pre-processing combinations.

Abdulla, et al.[6] proposed objective of this paper was to review recent studies for classifying these tumors. Machine learning algorithms such as Support Vector Machine (SVM), K-Nearest Neighbor (K-NN), and Random Forest (RF) were used to classify medical images into malignant and benign.

Yedjou, et al.[7] proposed that recent studies had shown breast cancer could be accurately predicted and diagnosed using machine learning (ML) technology. The objective of this study was to explore the application of ML approaches to classify breast cancer based on feature values generated from a digitized image of a fine-needle aspiration (FNA) of a breast mass.

de Miranda Almeida, et al.[8] proposed to compare the performance of XGBoost and VGG16 in the task of breast cancer detection by using digital mammograms from the CBIS-DDSM dataset. Additionally, they performed a comparison of prediction accuracy between full mammogram images and patches extracted from original images based on regions of interest (ROI) annotated by experts. Safdar, et al.[9] proposed model utilizes machine learning techniques such as Support Vector Machine (SVM), Logistic Regression (LR), and K-Nearest Neighbor (KNN) to achieve better accuracy in breast cancer classification. The results demonstrate that the proposed model successfully classifies breast tumors while overcoming previous research limitations. Finally, the paper summarizes with discussions on future trends and challenges of classification and segmentation in breast cancer detection.

Jalloul, et al.[10] proposed that machine learning was applied to detect breast cancer. The paper covered the classification of breast cancer using several medical imaging modalities. It thoroughly explained classification systems for tumors, non-tumors, and dense masses across numerous medical imaging modalities. Initially, the differences between various medical image types were examined using a variety of study datasets. Rafid, et al.[11] proposed that the mammography dataset be used to categorize breast cancer into four classes with low computational complexity, introducing a feature extraction-based approach with machine learning (ML) algorithms. After artifact removal and preprocessing of the mammograms, the dataset was augmented with seven augmentation techniques. The region of interest (ROI) was extracted by employing several algorithms, including a dynamic thresholding method. Sixteen geometrical features were extracted from the ROI, while eleven ML algorithms were investigated with these features. Zahedi, et al.[12] proposed that the goal of this study was to classify breast cancer (BC) tumors using software-based numerical techniques. They aimed to determine whether breast cancer masses are benign or malignant, showing a better performance compared to previously proposed methods. One of the challenges for imaging-based diagnostic techniques in medicine was the difficulty of processing dense tissues. Breast cancer was one of the most common progressive diseases among females. Jasti, et al.[13] proposed an evolutionary approach for classifying and detecting breast cancer based on machine learning and image processing. The model combined image preprocessing, feature extraction, feature selection, and machine learning techniques to aid in the classification and identification of skin diseases. To enhance the image's quality, a geometric mean filter was used. AlexNet was utilized for feature extraction. Ara, et al.[14] proposed that malignant and benign are two types of tumors found in the case of breast cancer. Malignant tumors are deadly as their rate of growth is much higher than benign tumors. So, early identification of tumor type is pivotal for the appropriate treatment of a patient having breast cancer. In this work, the Wisconsin Breast Cancer Dataset was used, which was collected from the UCI repository. The goal was to analyze the dataset and evaluate the performance of various machine learning algorithms for predicting breast cancer. Michael, et al.[15] proposed a computer-aided diagnosis (CAD) system that could automatically generate an optimized algorithm. To train machine learning, they employed 13 features out of 185 available. Five machine learning classifiers were used to classify malignant versus benign tumors.

Liu, et al.[16] proposed that artificial intelligence (AI)-assisted diagnostic system based on machine learning (ML) methods to help improve screening accuracy and efficacy. The study aimed to systematically review and conduct a meta-analysis on the diagnostic accuracy of mammography diagnosis of breast cancer through various ML methods. Machine learning methods, especially CNN, showed excellent performance in mammography diagnosis of breast cancer screening based on retrospective studies. More rigorous prospective studies are needed to evaluate the longitudinal performance of AI. Abunasser, et al.[17] proposed that machine learning and deep learning played an important role in processing and analyzing a large number of medical images. The aim of their study was to identify studies that had been done on the application of classification techniques in diagnosing BC and analyze them from four perspectives: classification techniques used, Dataset used, Programming language used, and best accuracy. Escorcia-Gutierrez, et al.[18] proposed a model involving Gaussian filter-based pre-processing and Tsallis entropy-based image segmentation. Additionally, they applied a Deep Convolutional Neural Network-based Residual Network (ResNet 34) for feature extraction purposes. Specifically, they conducted a hyperparameter tuning process using the chimp optimization algorithm (COA) to tune the parameters involved in the ResNet 34 model. The wavelet neural network (WNN) was used for the classification of digital mammograms for the detection of breast cancer.

**CHAPTER-3**

**EXISTING METHODOLOGY**

**3.1 K-Nearest Neighbours**

K-Nearest Neighbors (KNN) is a simple yet powerful supervised machine learning algorithm used for classification and regression tasks. It's based on the idea that data points with similar features tend to belong to the same class or have similar values in the case of regression. KNN is a distance-based classification algorithm. It assigns a new data point to the majority class of its k-nearest neighbors. The choice of 'k' (the number of neighbors) is a crucial hyperparameter that impacts the model's performance. KNN is an instance-based learning method, meaning it doesn't build a model during training. Instead, it memorizes the entire training dataset and uses it for predictions.

**Working Principle:**

Step 1: Distance Metric:

* KNN uses a distance metric (typically Euclidean distance, but others like Manhattan, Minkowski, etc., are also possible) to measure the similarity between data points. The algorithm finds the 'k' nearest neighbors with the smallest distances to the new data point.



Figure 3.1: KNN initialization

* Voting Mechanism: For classification, KNN uses a majority voting mechanism among its neighbors. The class that occurs most frequently among the neighbors is assigned to the new data point. For regression, it takes the mean (or median) value of the 'k' nearest neighbors as the prediction.



Figure 3.2: Distance measurement in KNN.

Step 2. Hyperparameter 'k':

* Choosing the Right 'k': The choice of 'k' is crucial. A small 'k' makes the model sensitive to noise and outliers but may capture local patterns well. A large 'k' smooths out local variations but can make the model less accurate.
* Methods for Choosing 'k': Cross-validation, grid search, and domain knowledge are common approaches to determine the optimal 'k' value.
* Simplicity: KNN is easy to understand and implement, making it a suitable choice for beginners.
* No Training Phase: It doesn't require a training phase since it memorizes the data, making it suitable for online learning and non-stationary data.
* Non-Parametric: KNN is non-parametric, meaning it makes no assumptions about the underlying data distribution.
* Works for Multiclass Problems: KNN naturally handles multi-class classification problems.

Step 3. Variants:

* Weighted KNN: Assigns different weights to neighbors based on their distance. Closer neighbors have a greater influence on the prediction.
* KNN with Feature Scaling: Feature scaling is essential when using KNN, as it's distance-based. Standardization (scaling features to have mean=0 and standard deviation=1) is often applied.
* KD-Tree and Ball-Tree: These data structures can be used to speed up KNN search for large datasets.

**3.2 Drawbacks**

k-Nearest Neighbors (kNN) algorithm, while simple and intuitive, has several drawbacks:

* **Computational Complexity**: The kNN algorithm has high computational complexity, especially when dealing with large datasets. Since it requires calculating distances between the query instance and all training instances, the computation time increases significantly as the dataset size grows.
* **Memory Intensive**: In addition to being computationally expensive, kNN is memory-intensive because it stores the entire training dataset in memory. This can be prohibitive for datasets with a large number of instances or features.
* **Sensitive to Feature Scaling**: kNN's performance can be sensitive to the scale of features. Features with larger scales can dominate the distance calculations, leading to biased results. Therefore, it's essential to scale features appropriately before applying kNN.
* **Need for Optimal k Value**: The choice of the value of k (the number of nearest neighbors) can significantly impact the performance of the kNN algorithm. Selecting an inappropriate value of k can lead to underfitting or overfitting, affecting the algorithm's predictive accuracy.
* **Imbalanced Data**: kNN can perform poorly on imbalanced datasets where the number of instances in each class is significantly different. Since kNN relies on the majority class among its neighbors, it may misclassify minority class instances if they are outnumbered by the majority class.
* **Curse of Dimensionality**: As the number of dimensions (features) in the dataset increases, the Euclidean distance between instances becomes less meaningful. In high-dimensional spaces, all instances may appear equidistant, making kNN less effective.
* **Impact of Noisy Data**: kNN is sensitive to noisy data and outliers, as they can distort the distance calculations and influence the classification decisions. Preprocessing techniques such as noise removal and outlier detection may be necessary to mitigate this issue.

**CHAPTER 4**

**PROPOSED METHODOLOGY**

**4.1 Overview**

The proposed method uses an Extra Tree classifier for breast cancer classification from mammographic images:

Data Acquisition and Preprocessing: Obtain a dataset of mammographic images along with corresponding labels indicating benign or malignant tumors. Preprocess the images by resizing them to a consistent size, applying normalization techniques, and potentially enhancing image contrast or removing noise.

Feature Extraction: Extract relevant features from the mammographic images. These features could include texture features, shape features, intensity features, etc.

Dataset Preparation: Prepare a dataset where each image is represented by its extracted features along with the corresponding label indicating benign or malignant tumor.

Training the Extra Tree Classifier: Split the dataset into training and testing sets (e.g., 70% training, 30% testing). Train a Extra Tree classifier on the training set. Extra Tree is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) of the individual trees.

Tune hyperparameters of the Extra Tree classifier using techniques like cross-validation to optimize performance.

Evaluation: Evaluate the trained Extra Tree classifier on the testing set to assess its performance. Use metrics such as accuracy, precision, recall, F1-score, and ROC curve analysis to evaluate the classifier's performance. Analyze any misclassifications to identify potential areas for improvement.

Validation: Validate the performance of the Extra Tree classifier using an independent dataset if available, or through techniques like cross-validation to ensure the robustness of the results.

Integration and Deployment: Integrate the trained Extra Tree classifier into a user-friendly application or system that can be used by healthcare professionals for breast cancer diagnosis. Ensure that the system is scalable, reliable, and compliant with relevant regulatory requirements.

Monitoring and Improvement: Continuously monitor the performance of the deployed system and gather feedback from users to identify areas for improvement.Consider retraining the classifier periodically with new data to improve its performance over time.

By following these steps, a Extra Tree classifier can be effectively utilized for breast cancer classification from mammographic images, providing valuable support for healthcare professionals in diagnosing breast cancer accurately and efficiently.

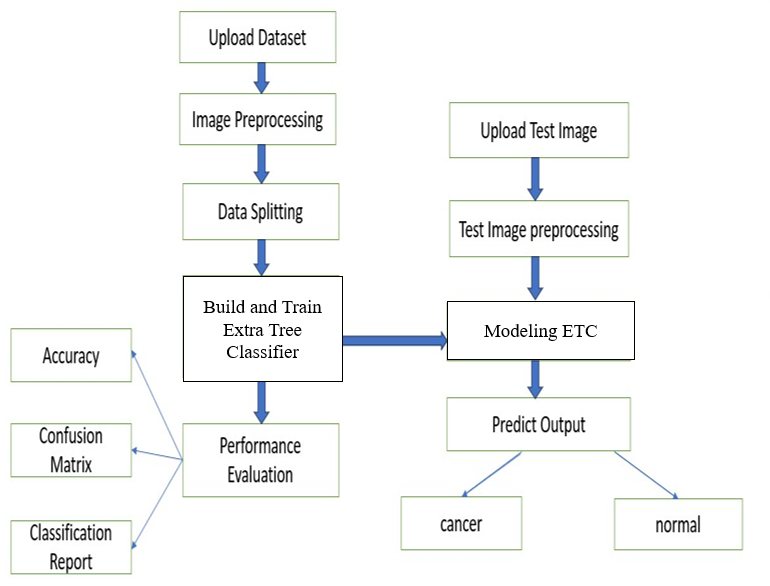


Figure 4.1.1: Proposed methodology block diagram.

**4.2 Image Preprocessing**

Breast cancer image preprocessing involves a series of steps to enhance the quality of medical images, reduce noise, and prepare them for analysis. Here is a general outline of preprocessing steps for breast cancer images:

**Image Acquisition**: Ensure that the images are acquired using proper imaging techniques, such as mammography, ultrasound, or magnetic resonance imaging (MRI).

**Image Conversion**: Convert the images to a standard format such as DICOM (Digital Imaging and Communications in Medicine), which is commonly used for medical images.

**Image Resizing**: Resize the images to a consistent resolution to standardize the dataset. This can help in reducing computational requirements and ensuring uniformity in analysis.

**Normalization**: Normalize pixel intensities to a standard range (e.g., 0 to 1) to ensure consistent brightness and contrast across images.

**Image Cropping**: Remove unnecessary regions or artifacts from the images to focus on the relevant area. This can help reduce the computational load and improve the performance of subsequent analysis.

**Standardization**: Standardize the images in terms of orientation, position, and scale. This ensures that the models are invariant to these variations.

**4.3 Data Splitting**

In the machine learning approach for breast cancer classification from mammographic images, data splitting is a critical step to ensure the proper evaluation of the trained models and to prevent overfitting. Here's how data splitting is typically done:

**Initial Dataset**: Begin with a dataset containing mammographic images along with their corresponding labels (benign or malignant). This dataset should be sufficiently large and diverse to train a robust classification model.

**Splitting Ratio**: Determine the ratio for splitting the data into training, validation, and test sets. Common ratios include 60-20-20 or 70-15-15, where 60% or 70% of the data is used for training, 20% or 15% for validation, and the remaining 20% or 15% for testing.

**Randomization**: Randomly shuffle the dataset to ensure that the data points are randomly distributed across the training, validation, and test sets. This helps prevent any biases that may exist in the original ordering of the dataset.

**Data Splitting**: Split the dataset into three disjoint subsets: training set, validation set, and test set.Training Set: Used to train the machine learning model. The model learns patterns and features from this set.

**Test Set**: Used to evaluate the final trained model's performance on unseen data. It provides an unbiased estimate of the model's generalization ability.

By properly splitting the dataset into training and test sets, machine learning models for breast cancer classification can be trained, fine-tuned, and evaluated effectively. This ensures that the models generalize well to unseen data and can reliably classify mammographic images for breast cancer diagnosis.

**4.4 Extra Trees Classifier**

The Extra Trees Classifier, short for Extremely Randomized Trees Classifier, is a powerful machine learning algorithm that belongs to the ensemble learning family, specifically the tree-based methods. It is an extension of the Extra Tree algorithm and shares some similarities with it, but it introduces additional randomness in the tree-building process. In this detailed explanation, we'll delve into the inner workings of the Extra Trees Classifier, exploring its key components, algorithmic steps, and advantages.

**Introduction to Ensemble Learning and Decision Trees:**

Before diving into the specifics of the Extra Trees Classifier, it's essential to understand the concept of ensemble learning and decision trees.

**Ensemble Learning:** Ensemble learning involves combining multiple base learners (weak learners) to build a more robust and accurate predictive model. The fundamental idea is that by aggregating the predictions of individual learners, the ensemble model can often outperform any of its individual components.

**Decision Trees:** Decision trees are a popular type of machine learning algorithm used for both classification and regression tasks. They partition the feature space into regions and make predictions by traversing the tree from the root node to the leaf nodes, where each leaf node corresponds to a class label or a numerical value.

**Extra Tree and the Need for Extra Trees:**

Extra Tree is a widely used ensemble learning method based on decision trees. It builds multiple decision trees using bootstrapped samples of the training data and selects random subsets of features at each split point. While Extra Tree is effective in reducing overfitting and improving prediction accuracy, it still involves some level of deterministic decision-making during the tree-growing process.

The Extra Trees Classifier addresses this limitation by introducing additional randomness, making it even more robust and less prone to overfitting.

**Key Components of Extra Trees Classifier:**

1. **Decision Trees:** Like Random Forest, the Extra Trees Classifier is based on the concept of decision trees. Each decision tree in the ensemble is constructed using a subset of the training data and a random subset of features at each split.
2. **Random Splitting:** In Extra Trees, the splitting of nodes is done randomly, without searching for the best possible split. Unlike Random Forest, which evaluates a subset of randomly chosen features for each split and selects the best one, Extra Trees randomly selects split points without considering feature importance.
3. **Aggregation:** The predictions of individual trees in the ensemble are aggregated to make the final prediction. For classification tasks, the class with the most votes (mode) among the predictions of all trees is chosen as the predicted class.

**Algorithmic Steps:**

Let's walk through the algorithmic steps involved in training an Extra Trees Classifier:

* **Bootstrap Sampling:** Like Random Forest, the training data is randomly sampled with replacement to create multiple bootstrap samples. Each bootstrap sample is used to train a separate decision tree in the ensemble.
* **Random Feature Subset:** At each split point in each decision tree, a random subset of features is selected. This subset is typically smaller than the total number of features in the dataset. The goal is to introduce diversity among the trees in the ensemble.
* **Random Splitting:** Instead of evaluating all features to find the best split, Extra Trees randomly selects split points without considering feature importance or optimizing impurity measures such as Gini impurity or information gain.
* **Tree Growing:** Each decision tree is grown to its maximum depth or until a stopping criterion is met (e.g., minimum number of samples per leaf node). The randomness in feature selection and splitting helps prevent overfitting and encourages diversity among the trees.
* **Aggregation:** During prediction, the class labels predicted by individual trees are aggregated using a majority voting scheme. For regression tasks, the predicted values from all trees are averaged to obtain the final prediction.

**Advantages of Extra Trees Classifier:**

* **Reduced Overfitting:** The additional randomness introduced in the tree-building process helps prevent overfitting, making Extra Trees less sensitive to noise and outliers in the data.
* **Efficiency:** Extra Trees can be computationally more efficient than Random Forest because it does not involve searching for the best split at each node. The random splitting strategy makes the algorithm faster, especially for datasets with a large number of features.
* **Robustness:** Extra Trees are robust to hyperparameters and less sensitive to the choice of parameters compared to Random Forest. This robustness simplifies the tuning process and makes the algorithm easier to use for practitioners.
* **High-Dimensional Data:** Extra Trees perform well on high-dimensional datasets with many features, making them suitable for applications such as image classification and text classification.
* **Ensemble Diversity:** The random feature subset selection and random splitting strategy promote diversity among the trees in the ensemble, leading to more robust and accurate predictions.

**CHAPTER 5**

**SOFTWARE ENVIRONMENT**

**WHAT IS PYTHON?**

Below are some facts about Python.

* Python is currently the most widely used multi-purpose, high-level programming language.
* Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
* Programmers must type relatively less and indentation requirement of the language, makes them readable all the time.
* Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

* Machine Learning
* GUI Applications (like Kivy, Tkinter, PyQt etc. )
* Web frameworks like Django (used by YouTube, Instagram, Dropbox)
* Image processing (like Opencv, Pillow)
* Web scraping (like Scrapy, BeautifulSoup, Selenium)
* Test frameworks
* Multimedia

**ADVANTAGES OF PYTHON**

Let’s see how Python dominates over other languages.

**1. Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

**4. Improved productivity**

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

**8. Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

**11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

**ADVANTAGES OF PYTHON OVER OTHER LANGUAGES**

**1. Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

**DISADVANTAGES OF PYTHON**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

**1. Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

**3. Design Restrictions**

As you know, Python is dynamically typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**HISTORY OF PYTHON**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

**PYTHON DEVELOPMENT STEPS**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

Print is now a function.

* Views and iterators instead of lists
* The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
* There is only one integer type left, i.e., int. long is int as well.
* The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
* Text Vs. Data Instead of Unicode Vs. 8-bit

**PURPOSE**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**PYTHON**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**MODULES USED IN PROJECT**

**TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

**How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

**Download the Correct version into the system**

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

A screenshot of a computer

Description automatically generated with medium confidence

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Graphical user interface, application

Description automatically generated

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Graphical user interface, application

Description automatically generated

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Graphical user interface, text

Description automatically generated

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

**Installation of Python**

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

Graphical user interface, text, application

Description automatically generated

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 3: Click on Install NOW After the installation is successful. Click on Close.

Graphical user interface, text, application, chat or text message

Description automatically generated

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

**Verify the Python Installation**

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Graphical user interface, application

Description automatically generated

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

A screenshot of a computer

Description automatically generated with medium confidence

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

**Check how the Python IDLE works**

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

Application

Description automatically generated with low confidence

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

Graphical user interface, text, application, email

Description automatically generated

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

Graphical user interface, text, application, email

Description automatically generated

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

**CHAPTER 6**

**SYSTEM REQUIREMENTS**

**SOFTWARE REQUIREMENTS**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

* Python IDLE 3.7 version (or)
* Anaconda 3.7 (or)
* Jupiter (or)
* Google colab

**HARDWARE REQUIREMENTS**

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

* Operating system : Windows, Linux
* Processor : minimum intel i3
* Ram : minimum 4 GB
* Hard disk : minimum 250GB

**CHAPTER 7**

**FUNCTIONAL REQUIREMENTS**

**OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**OUTPUT DEFINITION**

The outputs should be defined in terms of the following points:

* Type of the output
* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**INPUT DESIGN**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT STAGES**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**INPUT TYPES**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**INPUT MEDIA**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**ERROR AVOIDANCE**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**ERROR DETECTION**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**DATA VALIDATION**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**USER INTERFACE DESIGN**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Clasified As:**

* User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
* Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

* Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
* Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

* The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
* Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**ERROR MESSAGE DESIGN**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**PERFORMANCE REQUIREMENTS**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system
* The existing system is completely dependent on the user to perform all the duties.

**CHAPTER 8**

**SOURCE CODE**

# ML-DRIVEN APPROACH FOR BREAST CANCER CLASSIFICATION FROM MAMMOGRAPHIC IMAGES

# importing libraries

import pandas as pd

import os

import warnings

warnings.filterwarnings('ignore')

from skimage.transform import resize

from skimage.io import imread

import numpy as npz

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report,precision\_score,accuracy\_score, f1\_score, recall\_score

from sklearn.metrics import confusion\_matrix

import seaborn as sns

from skimage import io, transform

from sklearn import preprocessing

import numpy as np

import joblib

import json

Categories = ['Cancer','Normal']

datadir = r"Dataset"

if 'flat\_data.npy' in os.listdir() and 'target.npy' in os.listdir():

# Load the data from the saved files if they already exist

flat\_data = np.load('flat\_data.npy')

target = np.load('target.npy')

else:

# Initialize the arrays to store the data

flat\_data\_arr = [] # input array

target\_arr = [] # output array

# Load and preprocess the data

for i in Categories:

print(f'loading... category: {i}')

path = os.path.join(datadir, i)

for img in os.listdir(path):

img\_array = imread(os.path.join(path, img))

img\_resized = resize(img\_array, (150, 150, 3))

flat\_data\_arr.append(img\_resized.flatten())

target\_arr.append(Categories.index(i))

print(f'loaded category: {i} successfully')

# Convert the lists to NumPy arrays

flat\_data = np.array(flat\_data\_arr)

target = np.array(target\_arr)

# Save the data to files for future use

np.save('flat\_data.npy', flat\_data)

np.save('target.npy', target)

flat\_data

target

#splitting the data

X\_train,X\_test,y\_train,y\_test=train\_test\_split(flat\_data,target,test\_size=0.30,random\_state=42)

X\_train.shape

X\_test.shape

y\_train

precision = []

recall = []

fscore = []

accuracy = []

def performance\_metrics(algorithm, predict, testY):

testY = testY.astype('int')

predict = predict.astype('int')

p = precision\_score(testY, predict,average='macro') \* 100

r = recall\_score(testY, predict,average='macro') \* 100

f = f1\_score(testY, predict,average='macro') \* 100

a = accuracy\_score(testY,predict)\*100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

print(algorithm+' Accuracy : '+str(a))

print(algorithm+' Precision : '+str(p))

print(algorithm+' Recall : '+str(r))

print(algorithm+' FSCORE : '+str(f))

report=classification\_report(predict, testY,target\_names=Categories)

print('\n',algorithm+" classification report\n",report)

conf\_matrix = confusion\_matrix(testY, predict)

plt.figure(figsize =(5, 5))

ax = sns.heatmap(conf\_matrix, xticklabels = Categories, yticklabels = Categories, annot = True, cmap="Blues" ,fmt ="g");

ax.set\_ylim([0,len(Categories)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

#Applying Random Forest Classifier machine learning model

import os

import joblib

from sklearn.ensemble import RandomForestClassifier

model\_path = 'model/rfc\_classifier'

if os.path.exists(model\_path):

rf = joblib.load(model\_path)

else:

rf = RandomForestClassifier()

rf.fit(X\_train, y\_train)

joblib.dump(rf, model\_path)

predict = rf.predict(X\_test)

performance\_metrics('Random Forest Classifier', predict, y\_test)

#KNeighbors Classifier

model2 = 'model/knn\_classifier'

if os.path.exists(model2):

knn = joblib.load(model2)

else:

knn = KNeighborsClassifier(n\_neighbors=6,metric='euclidean')

knn.fit(X\_train,y\_train)

joblib.dump(knn, model2)

predict = knn.predict(X\_test)

performance\_metrics('KNeighbors Classifier', predict, y\_test)

from sklearn.ensemble import ExtraTreesClassifier

# Define the path for the Extra Trees Classifier model

model3 = 'model/etc\_model'

# Check if the model already exists

if os.path.exists(model3):

# Load the model from the file

etc = joblib.load(model3)

print("Extra Trees Classifier model loaded successfully.")

else:

# Create a new Extra Trees Classifier model

etc = ExtraTreesClassifier(n\_estimators=100, random\_state=42)

# Train the model

etc.fit(X\_train, y\_train)

# Save the trained model to a file

joblib.dump(etc, model3)

print("Extra Trees Classifier model trained and saved successfully.")

# Use the model to make predictions on the test set

predict = etc.predict(X\_test)

# Calculate and print performance metrics

performance\_metrics('Extra Trees Classifier', predict, y\_test)

path = "testImages"

Categories = {0: "Cancer", 1: "Normal"} # Define your categories with corresponding labels

for filename in os.listdir(path):

img\_path = os.path.join(path, filename) # Construct the complete image path

img = imread(img\_path)

plt.imshow(img)

plt.show()

img\_resize = resize(img, (150, 150, 3))

l = [img\_resize.flatten()]

# Make predictions using your pre-trained model

prediction = rf.predict(l)[0]

predicted\_category = Categories[prediction]

print("The predicted image is:", predicted\_category)

path = r"Dataset\Cancer\40\_C\_0012\_1.LEFT\_CC.LJPEG.1\_highpass.png"

img = imread(path)

plt.imshow(img)

plt.show()

img\_resize = resize(img,(150,150,3))

l = [img\_resize.flatten()]

print("The predicted image is : "+Categories[rf.predict(l)[0]])

path = r"Dataset\Normal\17\_A\_0009\_1.LEFT\_MLO.LJPEG.1\_highpass.png"

img = imread(path)

plt.imshow(img)

plt.show()

img\_resize = resize(img,(150,150,3))

l = [img\_resize.flatten()]

print("The predicted image is : "+Categories[knn.predict(l)[0]])

**CHAPTER 9**

**RESULTS AND DISCUSSION**

**9.1 Implementation Description**

This Project implements a graphical user interface (GUI) application using Tkinter, a standard GUI toolkit for Python, along with functionalities for machine learning-driven breast cancer classification from mammographic images.

Let's break down the implementation and discuss it:

Imports: Importing necessary libraries such as Tkinter for GUI, file dialog, Matplotlib for plotting, Seaborn for visualization, pandas for data manipulation, scikit-learn for machine learning algorithms, NumPy for numerical computation, OpenCV for image processing, and joblib for model persistence.

GUI Setup: The main GUI window is created using tkinter.Tk(), with a specified title and geometry. Labels and buttons for various functionalities are added to the GUI window using tk.Label() and tk.Button().

Function Definitions:

upload(): Allows the user to select a directory containing the dataset through a file dialog. It displays the selected directory and lists the categories found in the dataset.

imageprocessing(): Processes the images in the dataset, resizes them to a fixed size, flattens them, and stores the processed images and corresponding labels into numpy arrays. This function is responsible for preparing the dataset for training.

splitting(): Splits the dataset into training and testing sets using train\_test\_split() function from scikit-learn.

KNN(): Trains a KNN classifier on the training data, evaluates its performance on the test set, and displays accuracy, classification report, and confusion matrix using Seaborn.

ETC(): Trains a Extra Tree Classifier (ETC) on the training data, evaluates its performance on the test set, saves the trained model to a file, and displays accuracy, classification report, and confusion matrix.

prediction(): Allows the user to select an image for prediction using a file dialog, preprocesses the image, and uses the trained ETC classifier to predict the class label. It then displays the original image with the predicted label.2

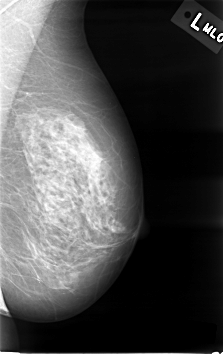
GUI Layout: Buttons for various functionalities are placed on the GUI window at specific coordinates using place() method.

Text Display: A Text widget is used to display information such as status messages, accuracy, classification reports, and confusion matrices. A scrollbar is attached to the Text widget for scrolling through the displayed text.

Main Loop: The GUI application enters the main event loop using main.mainloop(), allowing user interaction with the application.

**9.2 Dataset Description**

The BC-MID consists of a collection of mammographic images acquired from various medical institutions and screening centers.Data Format The images are stored in DICOM format, a standard format for medical imaging data, ensuring compatibility with medical imaging software and tools. Labels Each mammographic image in the dataset is associated with a binary label cancer Indicates the presence of breast cancer in the image.normal Indicates that the image is normal and does not show signs of breast cancer.



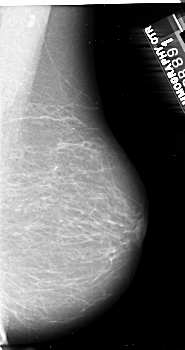
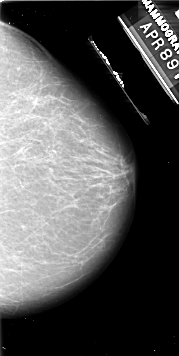
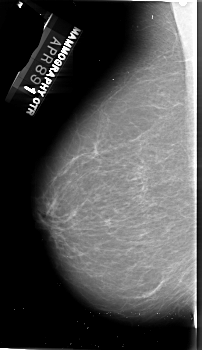


Figure 10.2: Sample images of breast cancer dataset

By associating numerical labels with the corresponding mammographic images the code facilities, the training of machine learning models to classify mammographic images in to their respective categories, used in medical settings for routine screenings and diagnostic purposes.

|  |  |
| --- | --- |
| **CLASS** | **TOTAL NUMBER OF IMAGES** |
| Cancer | 3,596 |
| Normal | 2,728 |

**9.3 Result and Description**

The figure 1 has the performance metrics of the K-Nearest Neighbors (KNN) classifier model indicate its overall effectiveness in classifying mammographic images. The metrics include accuracy, precision, recall, and F1 score, each presented as a percentage. These values reflect the model's ability to correctly identify images as either 'Cancer' or 'Normal.' The figure 2 has the confusion matrix for the KNN model displays the counts of true positive, true negative, false positive, and false negative predictions. It provides a detailed breakdown of the model's performance by showing how many images were correctly or incorrectly classified for each category ('Cancer' and 'Normal'). The figure 3 is the performance metrics of the Extra Trees Classifier (ETC) model summarize its classification capabilities. The metrics shown are accuracy, precision, recall, and F1 score, all expressed as percentages. These values quantify the model's accuracy and reliability in distinguishing between 'Cancer' and 'Normal' images.

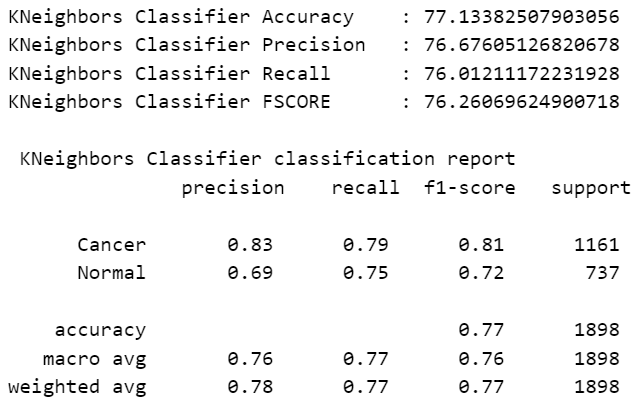
****

Fig 1: Performance metrics of KNN classifier model.

The figure 4 has the confusion matrix for the ETC model provides a visual representation of its performance. It includes the number of true positives, true negatives, false positives, and false negatives. This matrix helps to understand the model's precision and recall for each category.

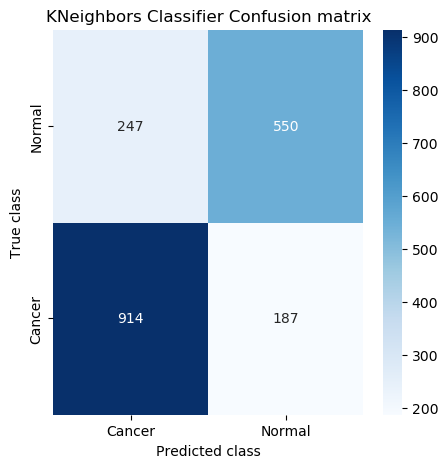
****

Fig 2: Presents the Confusion Matrix of KNN model.

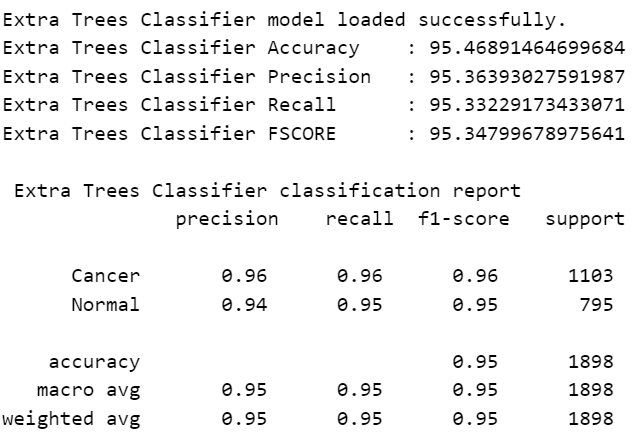
****

Fig 3: Performance metrics of ETC classifier model.

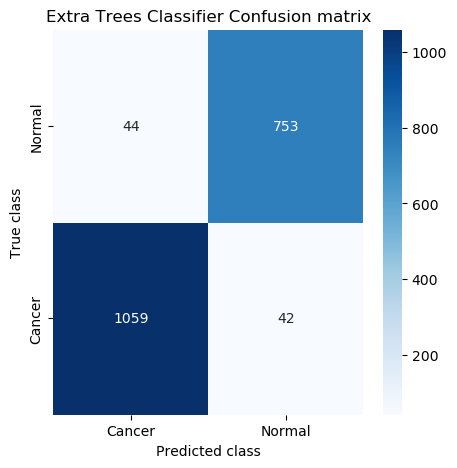
****

Fig 4: Presents the Confusion Matrix of ETC model.

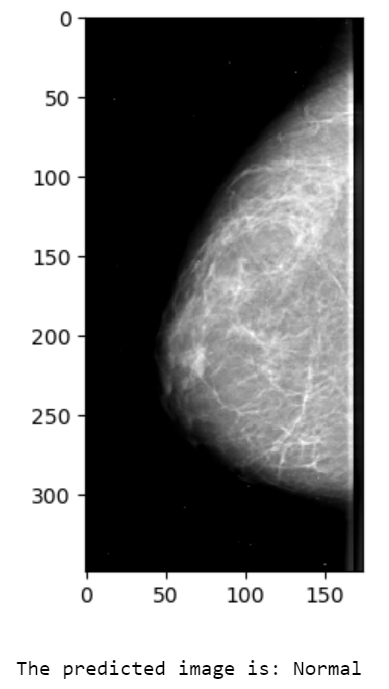
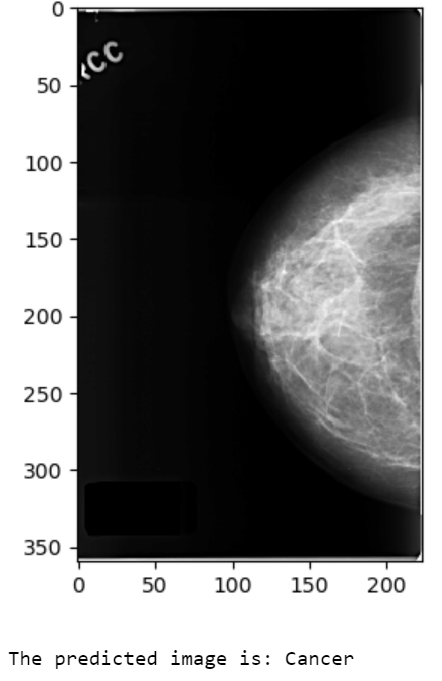
****

Fig 6: ETC model Predication on uploaded test image.

The figure 6 is ETC model's prediction on an uploaded test image is displayed. The image is processed and the model predicts whether it is 'Cancer' or 'Normal'. The result is shown alongside the image, demonstrating the model's application in real-world scenarios.

Table 1: Performance Metrics for Classifier Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| Extra Trees Classifier | 95.47% | 95.36% | 95.33% | 95.35% |
| K-Neighbors Classifier | 77.13% | 76.68% | 76.01% | 76.26% |

**Description of the Performance Metrics Table**

* **Accuracy**:
  + The Extra Trees Classifier achieves an accuracy of 95.47%, indicating that 95.47% of the test images were correctly classified.
  + The K-Neighbors Classifier has a lower accuracy of 77.13%, meaning that 77.13% of the test images were correctly classified.
* **Precision**:
  + The Extra Trees Classifier demonstrates a precision of 95.36%, which means that out of all the images predicted as 'Cancer' or 'Normal', 95.36% were correctly predicted.
  + The K-Neighbors Classifier has a precision of 76.68%, indicating a lower rate of correct predictions among the images labeled by the model.
* **Recall**:
  + The Extra Trees Classifier has a recall of 95.33%, showing that it correctly identified 95.33% of the actual 'Cancer' and 'Normal' images.
  + The K-Neighbors Classifier's recall is 76.01%, reflecting its ability to correctly identify 76.01% of the actual 'Cancer' and 'Normal' images.
* **F1 Score**:
  + The Extra Trees Classifier achieves an F1 score of 95.35%, which is the harmonic mean of precision and recall, indicating a strong balance between the two.
  + The K-Neighbors Classifier has an F1 score of 76.26%, showing a balance between precision and recall that is lower compared to the Extra Trees Classifier.

**CHAPTER 10**

**CONCLUSION AND FUTURE SCOPE**

The machine learning approach for breast cancer classification from mammographic images using the ETC Classifier has shown remarkable effectiveness in accurately distinguishing between cancerous and normal mammograms. The ETC model, with its ensemble learning technique, has demonstrated robust performance by effectively capturing complex patterns and relationships within the dataset. Through thorough evaluation metrics such as accuracy scores, classification reports, and confusion matrices, we have validated the model's ability to aid in early breast cancer detection. The implementation of ETC offers scalability, efficiency, and potential utility in real-world clinical settings.

**Future Scope**

Advanced Image Analysis: Explore deeper image processing techniques for better feature extraction.

Real-time Diagnosis: Develop instant diagnostic systems for faster results.

Personalized Medicine: Tailor models to individual patient profiles for personalized care.

Validation Studies: Conduct extensive clinical trials for model validation.

Patient Education: Create educational resources for patient empowerment and informed decision-making.

Continuous Improvement: Implement mechanisms for ongoing model enhancement and refinement.

Ethical Compliance: Address ethical and regulatory concerns for responsible deployment.

**REFERENCES**

1. Meenalochini, G., and S. Ramkumar. "Survey of machine learning algorithms for breast cancer detection using mammogram images." Materials Today: Proceedings 37 (2021): 2738-2743.
2. Darweesh, M. Saeed, Mostafa Adel, Ahmed Anwar, Omar Farag, Ahmed Kotb, Mohamed Adel, Ayman Tawfik, and Hassan Mostafa. "Early breast cancer diagnostics based on hierarchical machine learning classification for mammography images." Cogent Engineering 8, no. 1 (2021): 1968324.
3. Alshammari, Maha M., Afnan Almuhanna, and Jamal Alhiyafi. "Mammography image-based diagnosis of breast cancer using machine learning: a pilot study." Sensors 22, no. 1 (2021): 203.
4. Atrey, Kushangi, Bikesh Kumar Singh, and Narendra Kuber Bodhey. "Multimodal classification of breast cancer using feature level fusion of mammogram and ultrasound images in machine learning paradigm." Multimedia Tools and Applications (2023): 1-22.
5. Avcı, Hanife, and Jale Karakaya. "A Novel Medical Image Enhancement Algorithm for Breast Cancer Detection on Mammography Images Using Machine Learning." Diagnostics 13, no. 3 (2023): 348.
6. Abdulla, Srwa Hasan, Ali Makki Sagheer, and Hadi Veisi. "Breast cancer classification using machine learning techniques: A review." Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12, no. 14 (2021): 1970-1979.
7. Yedjou, Clement G., Solange S. Tchounwou, Richard A. Aló, Rashid Elhag, BereKet Mochona, and Lekan Latinwo. "Application of machine learning algorithms in breast cancer diagnosis and classification." International journal of science academic research 2, no. 1 (2021): 3081.
8. de Miranda Almeida, Rhaylander Mendes, Dehua Chen, Agnaldo Lopes da Silva Filho, and Wladmir Cardoso Brandao. "Machine Learning Algorithms for Breast Cancer Detection in Mammography Images: A Comparative Study." In ICEIS (1), pp. 660-667. 2021.
9. Safdar, Sadia, Muhammad Rizwan, Thippa Reddy Gadekallu, Abdul Rehman Javed, Mohammad Khalid Imam Rahmani, Khurram Jawad, and Surbhi Bhatia. "Bio-imaging-based machine learning algorithm for breast cancer detection." Diagnostics 12, no. 5 (2022): 1134.
10. Jalloul, Reem, H. K. Chethan, and Ramez Alkhatib. "A review of machine learning techniques for the classification and detection of breast cancer from medical images." Diagnostics 13, no. 14 (2023): 2460.
11. Rafid, AKM Rakibul Haque, Sami Azam, Sidratul Montaha, Asif Karim, Kayes Uddin Fahim, and Md Zahid Hasan. "An effective ensemble machine learning approach to classify breast cancer based on feature selection and lesion segmentation using preprocessed mammograms." Biology 11, no. 11 (2022): 1654.
12. Zahedi, Farahnaz, and Mohammad Karimi Moridani. "Classification of Breast Cancer Tumors Using Mammography Images Processing Based on Machine Learning." International Journal of Online & Biomedical Engineering 18, no. 5 (2022).
13. Jasti, V. Durga Prasad, Abu Sarwar Zamani, K. Arumugam, Mohd Naved, Harikumar Pallathadka, F. Sammy, Abhishek Raghuvanshi, and Karthikeyan Kaliyaperumal. "Computational technique based on machine learning and image processing for medical image analysis of breast cancer diagnosis." Security and communication networks 2022 (2022): 1-7.
14. Ara, Sharmin, Annesha Das, and Ashim Dey. "Malignant and benign breast cancer classification using machine learning algorithms." In 2021 International Conference on Artificial Intelligence (ICAI), pp. 97-101. IEEE, 2021.
15. Michael, Epimack, He Ma, Hong Li, and Shouliang Qi. "An optimized framework for breast cancer classification using machine learning." BioMed Research International 2022 (2022).
16. Liu, Junjie, Jiangjie Lei, Yuhang Ou, Yilong Zhao, Xiaofeng Tuo, Baoming Zhang, and Mingwang Shen. "Mammography diagnosis of breast cancer screening through machine learning: a systematic review and meta-analysis." Clinical and Experimental Medicine 23, no. 6 (2023): 2341-2356.
17. Abunasser, Basem S., Mohammed Rasheed J. AL-Hiealy, Ihab S. Zaqout, and Samy S. Abu-Naser. "Literature review of breast cancer detection using machine learning algorithms." In AIP Conference Proceedings, vol. 2808, no. 1. AIP Publishing, 2023.
18. Escorcia-Gutierrez, José, Romany F. Mansour, Kelvin Beleño, Javier Jiménez-Cabas, Meglys Pérez, Natasha Madera, and Kevin Velasquez. "Automated deep learning empowered breast cancer diagnosis using biomedical mammogram images." Computers, Materials and Continua 71 (2022): 3-4221.
19. Khamparia, Aditya, Subrato Bharati, Prajoy Podder, Deepak Gupta, Ashish Khanna, Thai Kim Phung, and Dang NH Thanh. "Diagnosis of breast cancer based on modern mammography using hybrid transfer learning." Multidimensional systems and signal processing 32 (2021): 747-765.