**Problem Statement :**

Breast Cancer Detection Using AdaBoost Classifier .This documentation outlines a project focused on detecting breast cancer using the AdaBoost classifier. The project is designed to help students learn about machine learning for medical diagnosis by implementing and training the AdaBoost model on breast cancer data.

**Module 1: Introduction to Breast Cancer and Medical Diagnosis**  
**What is Breast Cancer?**  
At breast cancer, the uncontrolled growth of the cells in the breast into a lump or mass, most commonly known as a tumour. If the tumour is malignant, that is, cancerous, it is capable of invading the surrounding tissues or spreading to other parts of the body. so, it is essential to find this disease early on for effective treatment.  
  
**Types of Breast Cancer:**  
**Malignant (cancerous):** This type contains tumour cells that effect the surrounding tissue and grow rapidly, which allows them to easily spread to other organs without treatment.  
**Benign (non-cancerous):** For the most part, these lumps are considered harmless. They do not spread but will require monitoring or, at least, removal if they become large enough.

**Early Detection is Important:**  
If breast cancer is diagnosed early, it will drastically influence treatment results. Some early-stage cancers are easier to treat and offer a more favourable prognosis. Early detection methods include imaging tests (mammograms and ultrasounds), biopsies (where a small sample of tissue is removed), and laboratory tests. Regular self-examinations and checkups are also very important.  
  
**Machine Learning in Medical Diagnosis**  
**How Does Machine Learning Help?**  
Machine learning usage is growing healthcare through the application of algorithms to medical data that identify patterns. The technology helps in diagnosing breast cancer by finding subtle indicators in the data, such as on mammogram images and patient histories.  
  
**Machine Learning In Breast Cancer Detection Examples:**  
Classification Model: These models treat the distinction between malignant (cancerous) and benign (non-cancerous) tumours.  
Predictive Model: These models assess the chance of cancer recurring or predicting the alternative chance of survival of a patient based on some other aspects.

**Module 2: Data Collection and Preprocessing**

**Importing Required Libraries**

A screenshot of a computer program

Description automatically generated

**pandas:** For handling data in DataFrame format.

**numpy:** For numerical operations.

**sklearn.preprocessing.StandardScaler:** For scaling features.

**sklearn.ensemble.AdaBoostClassifier:** For using the AdaBoost algorithm for classification.

**sklearn.tree.DecisionTreeClassifier:** For building a decision tree model.

**sklearn.model\_selection.train\_test\_split**: For splitting the data into training and testing sets.

**sklearn.metrics:** For evaluating the model's performance

**Loading the Dataset**

A close up of a word

Description automatically generated

The code defines the column names for the dataset, which includes an 'ID', 'Diagnosis' and 30 features

It reads the data from a CSV file called wdbc.data and stores it in a DataFrame named data

**Inspecting the Dataset**

**Data.head(10)** displays first 10 rows

A screenshot of a computer

Description automatically generated

**data.tail(10)** will displays last 10 rows of the dataset

A screenshot of a computer

Description automatically generated

**data.shape** shows the number of rows and columns

**A screenshot of a computer code

Description automatically generated**

**data.info()** gives a summary of the dataset, including column names, data types

**A screenshot of a computer

Description automatically generated**

**data.describe().T** gives statistical summaries of numerical columns, such as count, mean, and standard deviation. T is used to display as transpose

**A table of numbers and percentages

Description automatically generated**

**Checking for Missing Values**

**A screenshot of a computer

Description automatically generated**

As we can see , there are no missing values , so no need for data cleaning

**Data Analysis**

**Check for Duplicates: Identify the number and percentage of duplicate rows.**

**A screenshot of a computer code

Description automatically generated**

**Count Total Rows**: The code calculates the total number of rows in the Data Frame and stores it in num\_total\_rows.

**Count Duplicate Rows**: It checks for duplicate rows in the Data Frame and counts how many there are, storing the result in num\_duplicate\_rows.

**Calculate Percentage of Duplicates:** The code computes the percentage of duplicate rows by dividing the number of duplicates by the total rows and multiplying by 100.

**Print Results:** Finally, it prints the number and percentage of duplicate rows to the console.

**Data Preprocessing :**

**Note**:There are no missing values and no changes in datatypes

**Remove the 'ID' Column**

**A black and white screen with colorful text

Description automatically generated**

Remove the 'ID' Column: It deletes the column named 'ID' from the DataFrame data. The 'ID' column is typically used to uniquely identify each row and does not provide useful information for analysis or model training

**axis=1**: This specifies that you're dropping a column

**inplace=True**: This means that the changes will be applied directly to the data

**Feature Selection**

A close up of a word

Description automatically generated

**Define Features and Target Variable**: Separate the features (X) from the target variable (y)

The code defines X as the Data Frame containing all features except the 'Diagnosis' column

y as the Series containing the 'Diagnosis' column, which is the target variable for prediction

**Data Splitting**

A computer screen shot of a program code

Description automatically generated

**Dataset Splitting**: The code divides the dataset into a training set and a testing set.

**Training and Testing Proportions**: It uses 70% of the data for training the model and 30% for testing its performance.

**Reproducibility**: The random\_state=42 ensures that the split is consistent each time the code is run, allowing for reproducible results.

**Output Dimensions**: The code prints the number of samples in both the training and testing sets, providing insight into their sizes