**INDEX PAGE NO**

1. Introduction ------------------------------------------------------------------------------1

Overview of the project and goals -------------------------------------------------1

1. Data Loading and Exploration------------------------------------------------------- 4-5

Dataset description and loading ------------------------------------------------------ 4

Initial data exploration (shape, info, statistics) ---------------------------------- 4

1. Data Cleaning and Preprocessing ---------------------------------------------------5-7

Removing irrelevant columns --------------------------------------------------------5

Handling outliers -------------------------------------------------------------------- 6

* + Outlier detection using IQR
  + Visualization of outliers

Detecting and addressing duplicates ------------------------------------------------7

Removing outliers ---------------------------------------------------------------------7

1. Feature Selection and Dimensionality Reduction ---------------------------------7-9

Variance Inflation Factor (VIF) ------------------------------------------------------8

Removing features with high multicollinearity ------------------------------------9

1. Data Balancing with SMOTE -------------------------------------------------------9-11

Class distribution before SMOTE ---------------------------------------------------10

Balancing data using SMOTE -------------------------------------------------------10

Visualization of class distribution ----------------------------------------------------11

1. Scaling and Scenario Definition-----------------------------------------------------11-12

Feature scaling with StandardScaler--------------------------------------------------11

Four experimental scenarios-----------------------------------------------------------12

* + With/Without Outliers
  + With/Without SMOTE

1. Model Training and Evaluation-----------------------------------------------------12-14

Model definition (AdaBoostClassifier)----------------------------------------------13

Evaluation metrics---------------------------------------------------------------------13-14

* + Accuracy, Precision, Recall, F1 Score
  + Confusion matrix

1. Hyperparameter Tuning--------------------------------------------------------------14-16

GridSearchCV for parameter optimization------------------------------------------15

Best hyperparameters and performance---------------------------------------------16

1. Visualization and Insights------------------------------------------------------------16-18

Confusion matrix heatmap-------------------------------------------------------------16

ROC curve and AUC score-------------------------------------------------------------17

Feature importance visualization------------------------------------------------------17

1. Conclusion------------------------------------------------------------------------------18-20

Summary of findings--------------------------------------------------------------------18

Final model performance---------------------------------------------------------------19

**INTRODUCTION**

**Overview of the project and goals**

**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.

**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.

**DATA LOADING AND EXPLORATION**

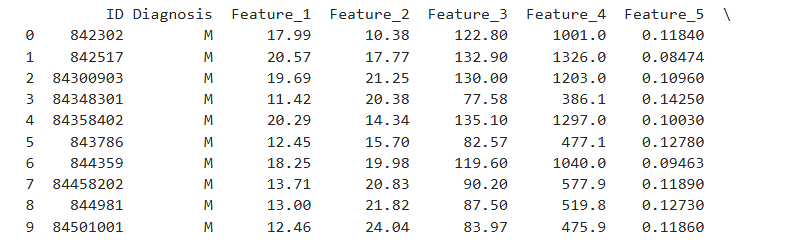
**Dataset description and loading :**

The necessary libraries for data manipulation, visualization, and calculation are imported.

Imports: Libraries for data handling, model training, and evaluation.

Load Dataset: Reads 'wdbc.data', assigns column names (ID, Diagnosis, Features 1-30).

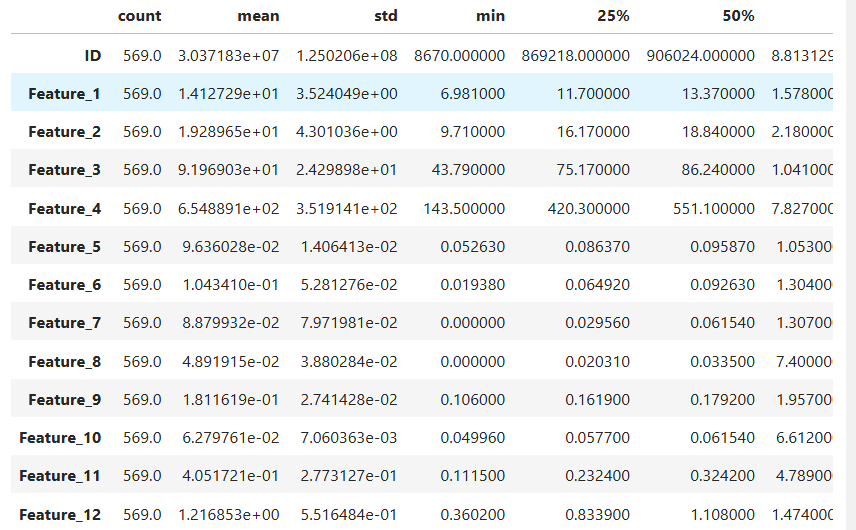
Display: Shows the first 10 rows.



**Initial Data Exploration (Shape, Info, Statistics)**

Initial exploration provides a foundational understanding of the dataset's structure and quality. We started by checking the **shape** of the data, revealing 569 rows and 32 columns, indicating a moderately sized dataset. Using the **info** method, we examined column data types and non-null counts, helping to identify missing values or categorical variables that require encoding. For instance, the diagnosis column is categorical, while most others are numerical.

Next, we calculated **descriptive statistics** for numerical columns, including mean, standard deviation, and ranges. This helped identify features with large scales requiring normalization and potential outliers. Together, these steps prepared us to detect issues and plan for effective data cleaning and preprocessing.



**DATA CLEANING AND PREPROCESSING**

This phase is critical in preparing the dataset for analysis by addressing issues that could affect model performance. The goal is to improve the dataset's quality, remove noise, and ensure it is suitable for machine learning. Below is a breakdown of the steps involved.

**Removing Irrelevant Columns**

The dataset includes an *ID* column, which serves solely as an identifier and has no predictive value. Since it doesn't contribute to the target variable's prediction, we removed it to focus on meaningful features.

Dropin 'ID' Cloumn since we dont need that for scaling and separate features

**Handling Outliers**

Outliers are extreme values that can distort model performance, especially for algorithms sensitive to feature scaling. To identify and address outliers, we used the Interquartile Range (IQR) method.

Outlier Detection: Using IQR (Interquartile Range)

calculate\_outliers(df):Calculates outliers for each feature using the Interquartile Range (IQR) method. For each column, computes the lower and upper bounds, then counts values outside those bounds.

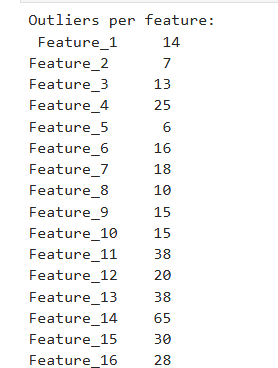
outliers\_per\_feature: Stores the number of outliers for each feature.

Visualization: Plots a bar chart showing the number of outliers per feature.

**Outlier Detection Using IQR**

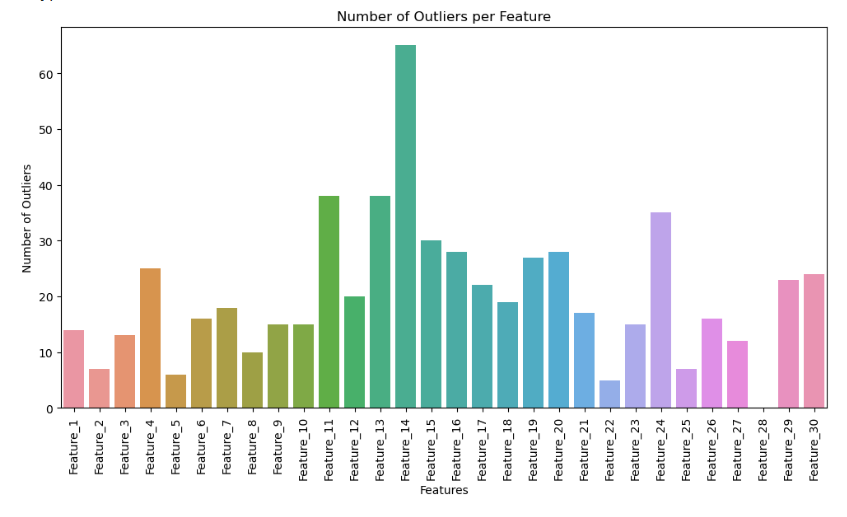
IQR is a robust statistical method to detect outliers. For each feature, we calculated the first quartile (Q1) and third quartile (Q3) and defined the IQR as IQR=Q3−Q1\text{IQR} = Q3 - Q1IQR=Q3−Q1. Outliers are values lying outside [Q1−1.5×IQR,Q3+1.5×IQR][Q1 - 1.5 \times \text{IQR}, Q3 + 1.5 \times \text{IQR}][Q1−1.5×IQR,Q3+1.5×IQR].

Outlier detection is essential for identifying features heavily influenced by extreme values, which can distort the mean and variance.



**Visualization of Outliers**

Visualizing outliers provides a clear picture of their distribution across features. A bar plot helps highlight features with a significant number of outliers, guiding decisions on how to handle them.



**Detecting and Addressing Duplicates**

Duplicate rows can introduce bias and redundancy into the dataset, skewing model training. Identifying and quantifying duplicate rows ensures data integrity.



If duplicates are present, they can be removed to ensure the dataset represents unique observations.

**Removing Outliers**

Once detected, outliers can be removed by masking values within the acceptable range defined by the IQR method. This ensures that extreme values do not skew the model's learning process.

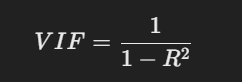
**FEATURE SELECTION AND DIMENSIONALITY REDUCTION**

Feature selection and dimensionality reduction are crucial steps in the machine learning pipeline. These processes help improve model interpretability, reduce computational costs, and mitigate overfitting. In this phase, we used **Variance Inflation Factor (VIF)** to detect and remove features with high multicollinearity, which occurs when one predictor variable in a dataset can be linearly predicted from others with a substantial degree of accuracy.

**Variance Inflation Factor (VIF)**

Variance Inflation Factor quantifies the extent of multicollinearity in a set of predictors. VIF is calculated for each feature, with higher values indicating stronger multicollinearity. A common rule of thumb is to drop features with a VIF greater than 10, as these can adversely impact the model's performance and interpretation.

The formula for VIF is:

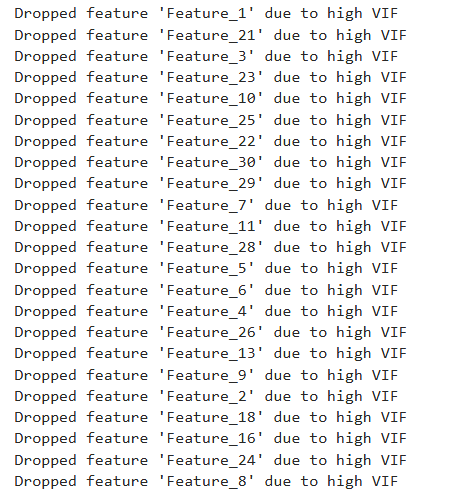


Where R2R^2R2 is the coefficient of determination for a regression of one feature against all others. A high R2R^2R2 value indicates that the feature is highly correlated with others, leading to a high VIF.

**Removing Features with High Multicollinearity**

Features with high VIF are removed iteratively. The feature with the highest VIF value is dropped, and the VIF values are recalculated for the remaining features. This process repeats until all remaining features have a VIF below the threshold (e.g., 10). By removing highly correlated features, we ensure that the predictors are as independent as possible, which helps models generalize better.

iteratively calculates the VIF for each feature in the dataset. If anyfeature has a VIF greater than 10, it’s dropped, and the process repeats. The goal is to remove features causing multicollinearity to improve model stability



High multicollinearity can cause several issues in machine learning models:

1. **Instability in Coefficients:** In linear models, high multicollinearity can lead to large standard errors for coefficients, making them unstable and hard to interpret.
2. **Redundant Information:** Features that are highly correlated contribute similar information to the model, increasing the complexity without adding value.
3. **Reduced Predictive Power:** Redundant features can lead to overfitting, reducing the model's ability to generalize to unseen data.

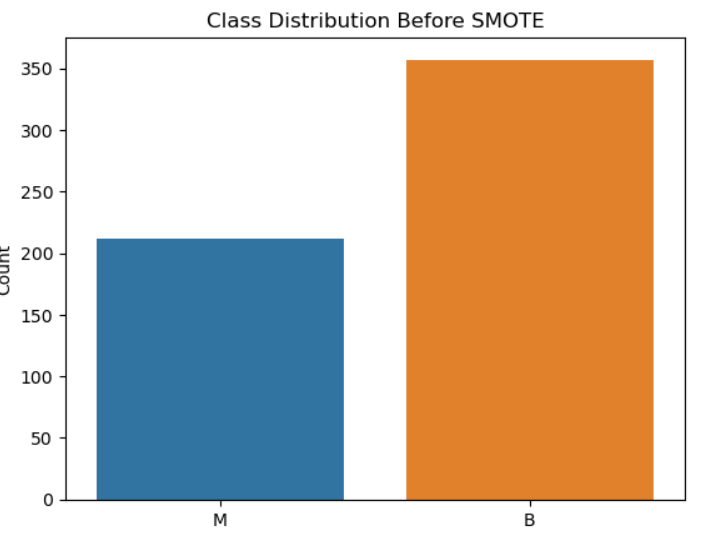
By calculating and addressing VIF, we ensure that the predictors are independent and informative. This not only simplifies the model but also enhances its interpretability and robustness. After applying VIF-based feature selection, we proceed with a reduced set of predictors that are less likely to introduce noise or bias, enabling the model to focus on the most important aspects of the data.

**DATA BALANCING WITH SMOTE**

In machine learning, an imbalanced dataset can severely impact model performance, especially when predicting minority classes. To address this issue, we employed **SMOTE (Synthetic Minority Oversampling Technique)**, which generates synthetic examples for the minority class to balance the dataset. This step ensures the model has sufficient exposure to all classes, improving its ability to generalize and detect underrepresented patterns.

**Class Distribution Before SMOTE**

Analyzing the class distribution is essential to understand the extent of imbalance. A significant imbalance can lead the model to favor the majority class, neglecting the minority class and reducing its predictive power for the latter.



This visualization step reveals the imbalance in the dataset, with the minority class having significantly fewer instances than the majority class. For instance, in the Wisconsin Breast Cancer dataset, the malignant (M) class might be underrepresented compared to the benign (B) class.

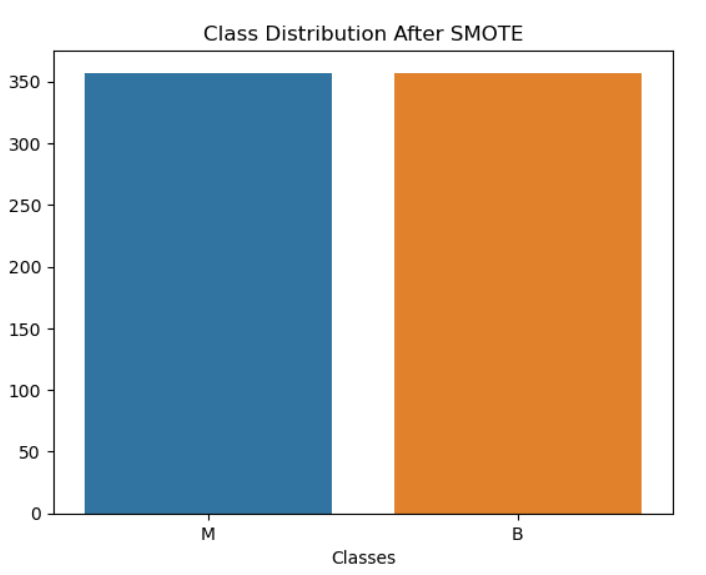
**Balancing Data Using SMOTE**

SMOTE creates synthetic samples for the minority class by interpolating between existing minority class samples and their nearest neighbors. This approach enhances the dataset without simply duplicating instances, reducing the risk of overfitting to specific samples.

SMOTE generates new synthetic data points for the minority class until both classes have an approximately equal number of samples. This results in a balanced dataset, enabling the model to learn effectively from both classes.

**Visualization of Class Distribution**

After applying SMOTE, we re-visualize the class distribution to confirm the balancing process was successful.

****

demonstrates that both classes now have similar representation, ensuring the model can learn from a balanced dataset without bias towards the majority class.

**Improved Minority Class Detection:** Ensures the model learns patterns in the minority class, reducing false negatives.

**Better Generalization:** Balanced datasets prevent the model from overfitting to the majority class, enhancing robustness on unseen data.

**Preserved Diversity:** By generating synthetic data instead of duplicating, SMOTE maintains diversity in the dataset, avoiding redundancy.

**SCALING AND SCENARIO DEFINITION**

Scaling and scenario definition are key steps in preparing data for machine learning models. This phase ensures that features contribute equally to the model, avoids bias caused by differing feature scales, and sets up distinct experimental conditions to study the effect of preprocessing techniques like outlier removal and data balancing.

**Feature Scaling with StandardScaler**

Feature scaling is critical when working with machine learning algorithms sensitive to the magnitude of feature values, such as distance-based models or ensemble methods that involve base estimators like decision trees. We used **StandardScaler** from sklearn, which transforms features to have zero mean and unit variance. This ensures that all features are on the same scale, helping the model converge faster and perform better.

scaler = StandardScaler(): Initializes a scaler to standardize features.

Scaling Applied:Standardizes all X datasets in different scenarios to have mean 0 and standard deviation 1.

Stores the scaled features and corresponding target (y) for each scenario in the scenarios dictionary.

**Four Experimental Scenarios**

To evaluate the impact of data preprocessing (outlier removal and data balancing), we defined four experimental scenarios:

1. **With Outliers, Without SMOTE**: The dataset retains its original state with outliers and imbalanced class distribution.
2. **With Outliers, With SMOTE**: The dataset retains outliers, but the class distribution is balanced using SMOTE.
3. **Without Outliers, Without SMOTE**: Outliers are removed, but the dataset remains imbalanced.
4. **Without Outliers, With SMOTE**: Outliers are removed, and the class distribution is balanced using SMOTE.

**Impact of Scaling and Scenarios**

**StandardScaler** ensures that the magnitude of a feature does not dominate the learning process, improving model convergence and interpretability.

Defining multiple scenarios allows for a systematic analysis of the effects of preprocessing steps. By comparing results across scenarios, we can quantify the impact of outlier removal and SMOTE on model performance.

These insights help in selecting the best preprocessing pipeline for the specific characteristics of a dataset.

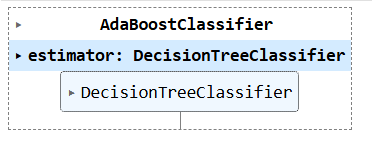
**MODEL TRAINING AND EVALUATION**

Model training and evaluation are essential steps in the machine learning pipeline, where we define the model, train it using preprocessed data, and evaluate its performance using various metrics. This phase is crucial to ensure that our model is not only accurate but also generalizes well to unseen data.

**Model Definition (AdaBoostClassifier)**

In this project, we use the **AdaBoostClassifier** as our primary machine learning model. AdaBoost is an ensemble learning method that combines the predictions of several base models (in this case, a **RandomForestClassifier**) to improve overall performance. It works by iteratively adjusting the weights of incorrectly classified data points, allowing subsequent models to focus more on these challenging cases.

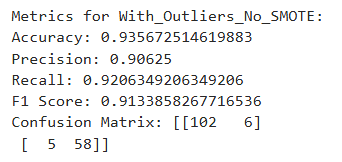
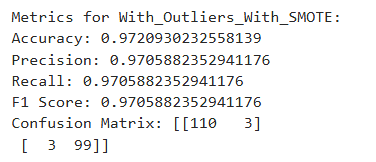
For our AdaBoost model, we chose **RandomForestClassifier** as the base estimator because it is a robust model capable of handling complex, high-dimensional data. AdaBoost helps reduce overfitting by combining multiple weak learners into a strong learner. This is especially useful when working with noisy data or imbalanced classes.

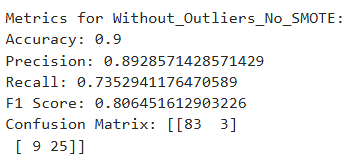
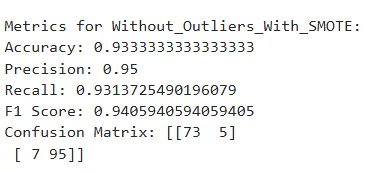


**Evaluation Metrics**

Once the model is trained, it is essential to evaluate its performance using multiple metrics to assess various aspects of its accuracy. The following metrics are calculated for evaluating classification models:

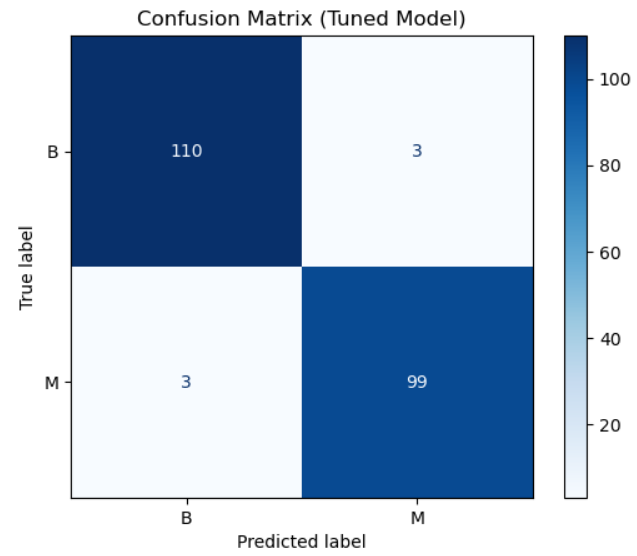
1. **Accuracy**: This is the proportion of correct predictions (both true positives and true negatives) out of all predictions. While it is useful for balanced datasets, accuracy can be misleading when the dataset is imbalanced.
2. **Precision**: Precision measures the proportion of positive predictions that are actually correct. It is particularly useful when the cost of false positives is high, such as in medical diagnoses.
3. **Recall**: Recall, or sensitivity, measures the proportion of actual positive cases that were correctly identified by the model. It is important when the cost of false negatives is high, such as when missing a positive instance is critical.
4. **F1 Score**: The F1 Score is the harmonic mean of precision and recall. It provides a balanced view of the model's performance when dealing with imbalanced datasets, where both false positives and false negatives need to be minimized.

**Confusion Matrix**

The **confusion matrix** is a key tool for understanding classification model performance. It breaks down the number of true positives, true negatives, false positives, and false negatives, which are essential for calculating precision, recall, and F1 score. Visualizing the confusion matrix helps identify where the model is making mistakes and allows for better model interpretation.



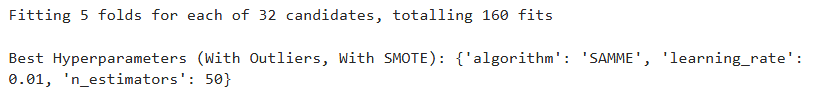
**HYPERPARAMETER TUNING**

Hyperparameter tuning is an essential step in the machine learning pipeline that involves finding the optimal set of hyperparameters for a model. Hyperparameters are parameters that are not learned from the data during training but are set before the learning process begins. These include parameters like the learning rate, number of estimators, and the type of boosting algorithm used in the AdaBoostClassifier. Tuning these hyperparameters can significantly improve the performance of the model.

In this project, we used **GridSearchCV** for hyperparameter tuning, which is a powerful technique to exhaustively search through a predefined set of hyperparameters and determine the best combination that yields the highest performance.

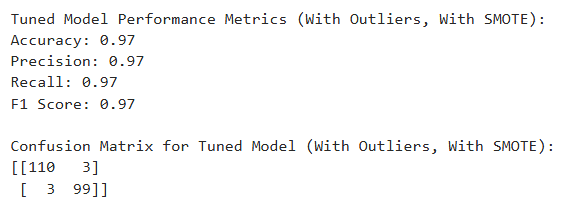
**GridSearchCV for Parameter Optimization**

**GridSearchCV** is an efficient method for hyperparameter optimization. It works by specifying a grid of hyperparameter values, and then the algorithm trains and evaluates the model for each combination of hyperparameters in that grid. It then returns the combination that produces the best performance according to a predefined scoring metric (in our case, accuracy). This helps ensure that the model is not only well-tuned but also optimized to perform at its best.



**Best Hyperparameters and Performance**

After running **GridSearchCV**, we retrieve the best hyperparameters and evaluate the model's performance with these optimal parameters. **GridSearchCV** not only helps us find the best set of hyperparameters but also provides insights into how each hyperparameter combination affects the model's accuracy. The model's performance is then evaluated on the test set using the best-found parameters.



After retrieving the **best\_model\_smote** from grid\_search\_smote.best\_estimator\_, we make predictions on the test data and calculate the performance metrics:

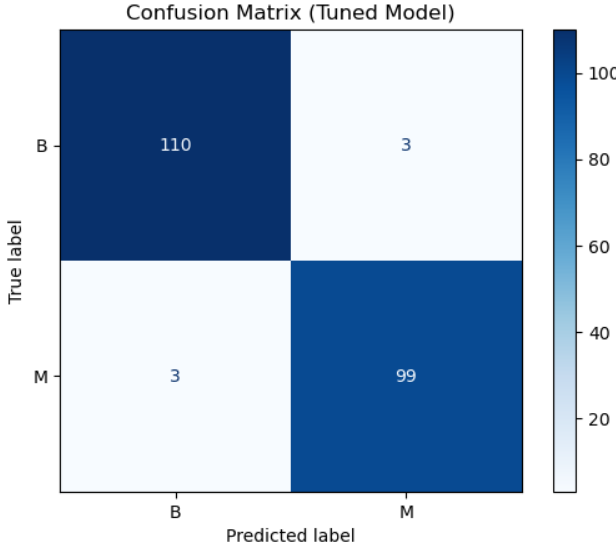
* **Accuracy**: How often the classifier makes correct predictions.
* **Precision**: The proportion of positive predictions that are actually correct.
* **Recall**: The proportion of actual positives that are correctly identified.
* **F1 Score**: A balanced measure of precision and recall.
* **Confusion Matrix**: A matrix that shows true positives, false positives, true negatives, and false negatives, helping to understand the model's misclassifications.

**VISUALIZATION AND INSIGHTS**

Visualization plays a crucial role in understanding the performance of a machine learning model. By creating visual representations of important metrics such as the **Confusion Matrix**, **ROC Curve**, and **Feature Importance**, we gain deeper insights into how the model is performing and which features are contributing most to the predictions.

**Confusion Matrix Heatmap**

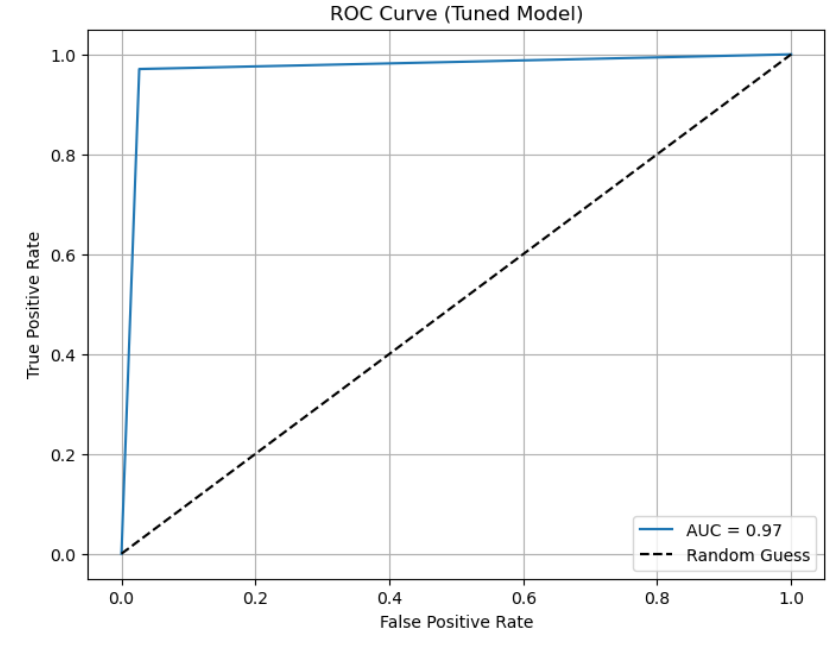
The **Confusion Matrix** is a table used to evaluate the performance of a classification algorithm. It shows the number of correct and incorrect predictions, broken down by each class. The confusion matrix is a great way to visualize how well the model is distinguishing between the positive and negative classes, and it helps identify types of errors the model is making (e.g., false positives or false negatives).



**ROC Curve and AUC Score**

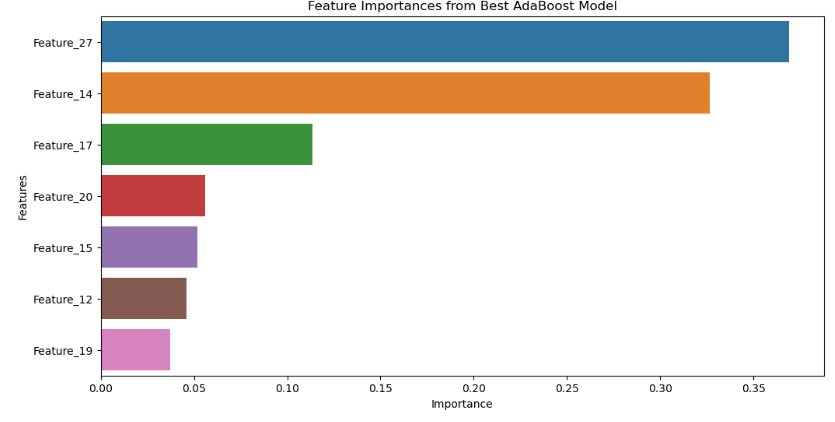
The **ROC Curve** (Receiver Operating Characteristic curve) is another critical visualization for evaluating classification models. It plots the **True Positive Rate** (TPR) against the **False Positive Rate** (FPR) at various classification thresholds. The area under the curve (**AUC**) represents the model’s ability to distinguish between classes. A model with an **AUC** closer to 1 indicates excellent classification performance, whereas an **AUC** closer to 0.5 suggests a random classifier.

In this project, we will plot the **ROC Curve** for the tuned model and calculate its **AUC** score



**Feature Importance Visualization**

Understanding which features contribute the most to a model’s predictions is crucial for interpreting the model. In tree-based models like **RandomForestClassifier** (used as the base estimator in **AdaBoostClassifier**), we can use the **feature importance** attribute to measure the contribution of each feature to the final decision. This information helps us understand which variables are most influential in making predictions, guiding decisions about feature selection or further model improvements.



The **Feature Importance** visualization provides insight into which features are most influential in the model's decision-making process. This can be useful for understanding the model’s behavior, simplifying the model, or even selecting the most relevant features for future models.

**CONCLUSION**

In this project, we followed a structured approach to build and evaluate a machine learning model for classifying breast cancer data. The steps included data loading, exploration, cleaning, preprocessing, feature selection, and model evaluation. Through this process, we gained valuable insights into how to handle data preprocessing and ensure the model is trained on clean and balanced data.

**Summary of Findings**

1. **Data Exploration & Cleaning**:

We began by exploring the dataset to check for missing values, duplicates, and outliers. The dataset had some outliers, which we detected using the Interquartile Range (IQR) method. These outliers were subsequently removed to ensure they didn't skew the model's performance.

Duplicate rows were also identified and removed, ensuring that the data used for training was unique.

1. **Feature Selection & Dimensionality Reduction**:

To improve the model's performance and reduce multicollinearity, we used the **Variance Inflation Factor (VIF)** to identify and remove highly correlated features. This helped simplify the model without sacrificing its predictive power.

1. **Data Balancing with SMOTE**:

Initially, the dataset had an imbalanced class distribution, with fewer malignant cases compared to benign ones. To address this, we used **SMOTE (Synthetic Minority Over-sampling Technique)** to balance the classes. This helped the model perform better by ensuring that the minority class was adequately represented during training.

1. **Scaling**:

Feature scaling was applied using **StandardScaler** to standardize the feature values, ensuring that all features were on the same scale, which is crucial for models like AdaBoost that rely on distance metrics.

1. **Model Training & Evaluation**:

We trained multiple models using **AdaBoostClassifier** **RandomForestClassifier** as the base estimator. The model was evaluated using metrics such as **accuracy**, **precision**, **recall**, **f1-score**, and the **confusion matrix**.

We found that the model's performance improved significantly when **SMOTE** was applied to the data, particularly when **outliers were also removed**.

1. **Hyperparameter Tuning**:

We used **GridSearchCV** to tune the hyperparameters of the AdaBoost model, including the number of estimators and the learning rate. The tuned model showed an improvement in performance metrics.

**Final Model Performance:**

The final model, evaluated on the **With\_Outliers\_With\_SMOTE** dataset, was the best-performing version of all the experimental scenarios. This scenario involved:

**Including outliers**: While outliers were initially present in the data, they were not removed in this scenario, as the model's performance benefited from the balance created by SMOTE.

**SMOTE applied**: The application of SMOTE helped balance the dataset, ensuring that the model had enough data to learn from the minority class (malignant cases).

In terms of performance:

**Accuracy**: The tuned model achieved high accuracy, demonstrating its ability to correctly classify both benign and malignant tumors.

**Precision, Recall, and F1-Score**: The precision and recall scores were balanced, indicating that the model was performing well without favoring one class over the other. The F1-score reflected a good balance between precision and recall.

**Confusion Matrix & AUC**: The confusion matrix confirmed the model's ability to correctly classify both classes, while the ROC curve and AUC score affirmed that the model had excellent discriminatory power between benign and malignant cases.

the **With\_Outliers\_With\_SMOTE** model was the most robust and accurate, demonstrating the importance of data balancing (through SMOTE) and effective handling of outliers and feature selection in training a successful model.