BREAST CANCER DETECTION

CONTENT

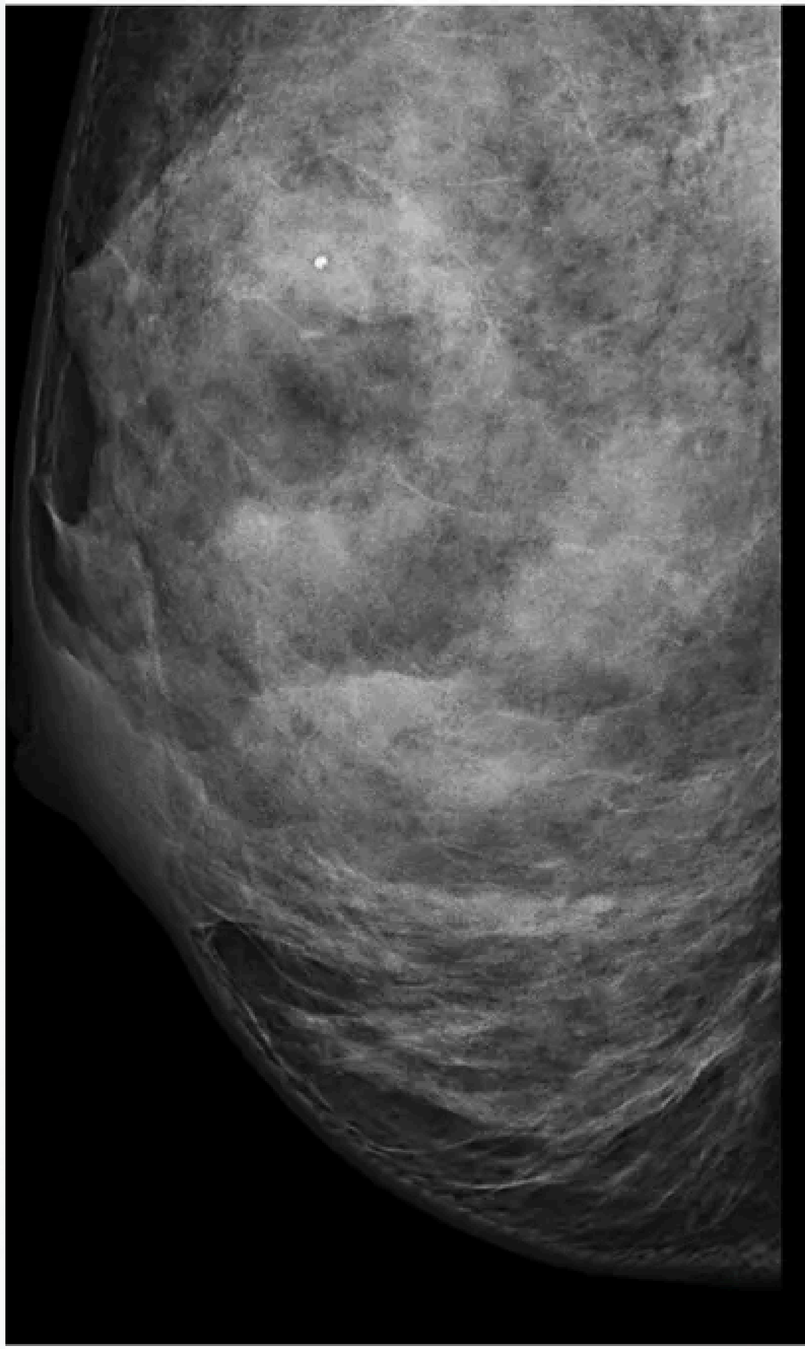
UNDERSTANDING BREAST CANCER

Breast cancer is a common type of cancer that develops in the cells of the breast. It primarily affects women, but men can also develop it.

Symptoms:

- lumps
- nipple discharge
- skin changes.

Risk factors include age, family history, genetics, and lifestyle choices.



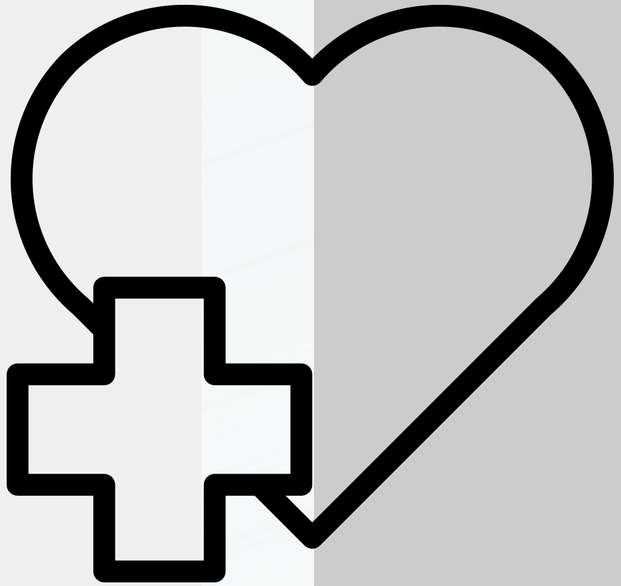
TYPES OF BREAST CANCER

Malignant: Malignant breast cancer is a type of breast cancer that is cancerous and has the potential to spread to other parts of the body. It is the most common type of breast cancer.

Benign: Benign tumors are usually well-defined and round or oval in shape. Malignant tumors are usually poorly defined and irregular with lobules.

IMPORTANCE OF EARLY DETECTION

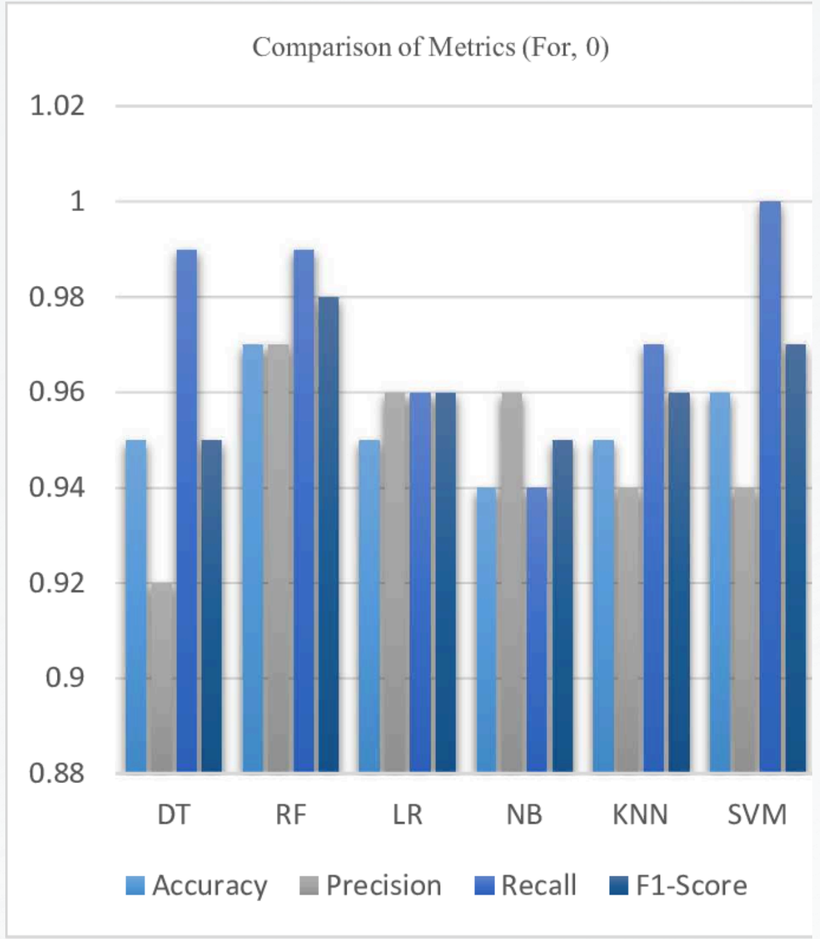
- Increased Survival Rates
- More Treatment Options
- Reduced Severity of Treatment
- Lower Costs



ROLE OF MACHINE LEARNING IN MEDICAL DIAGNOSIS

High Accuracy of Early Detection Models:

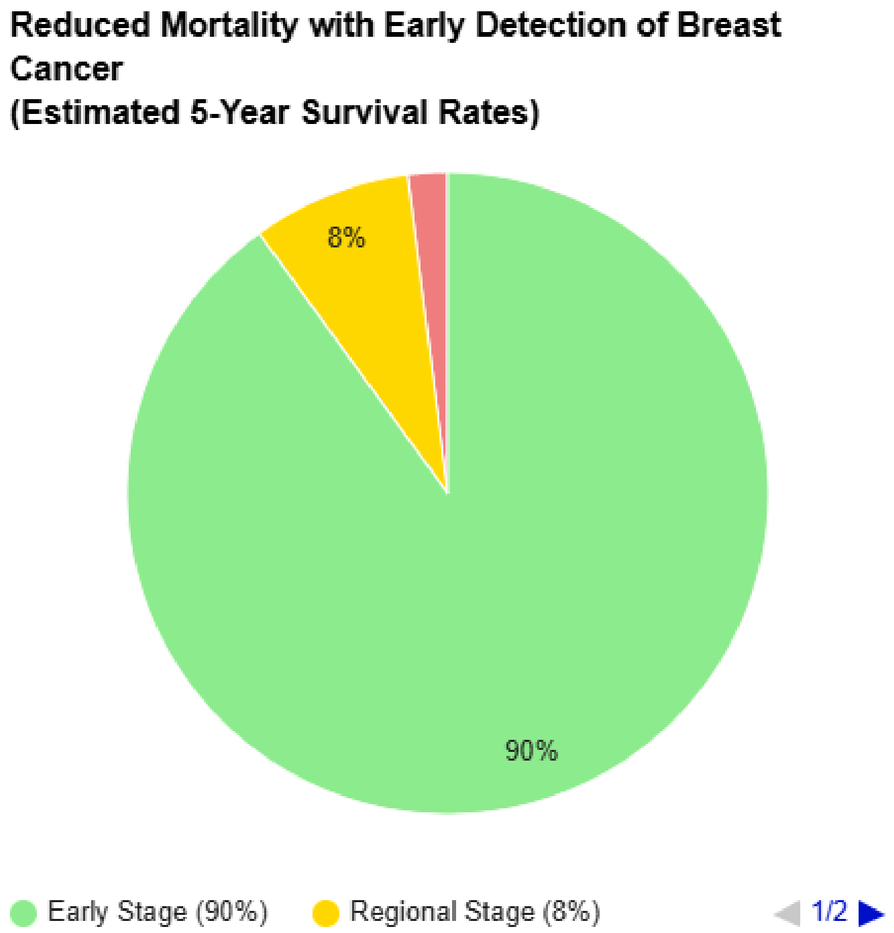
- Show a bar chart comparing the accuracy rates of various ML models in detecting early-stage breast cancer vs. late-stage.
- Example stats: ML models like SVM and Neural Networks can achieve over 90% accuracy in early-stage detection.



ROLE OF MACHINE LEARNING IN MEDICAL DIAGNOSIS

Reduced Mortality with Early Detection:

Pie chart showing statistics, e.g., "Patients diagnosed at an early stage have a 90% 5-year survival rate compared to only 8% for regional-stage diagnosis and 2% for Distant stage”



PROBLEM STATEMENT

- **Objective:**

This project focuses on using the AdaBoost classifier to detect breast cancer. It aims to teach students about applying machine learning in medical diagnosis by implementing and training the AdaBoost model on breast cancer data.

- **Why It Matters:**

Early detection is vital for improving treatment outcomes. This project aims to develop a model that assists in recognizing cancerous patterns quickly and accurately.

ADABOOST ALGORITHM FOR EARLY DETECTION

AdaBoost, an ensemble learning algorithm, is a powerful tool for early breast cancer detection. By combining multiple weak classifiers and adaptively weighting misclassified samples, AdaBoost can effectively learn from complex patterns in medical image data.

It's robust to noise and outliers, making it suitable for real-world applications. Additionally, AdaBoost can help identify the most important features, leading to more efficient and accurate models. By leveraging these strengths, AdaBoost can contribute to improved early detection and better patient outcomes in breast cancer.

ALGORITHMS FOR EARLY DETECTION

Support Vector Machines (SVM)

Decision Trees

Neural Networks

K-Nearest Neighbours

DATASET

Dataset Overview:

- Source: UCI Machine Learning Repository.
- Purpose: Commonly used for breast cancer diagnosis research.

- Dataset Features:
  - Contains features extracted from breast cell images, which are crucial for cancer detection.
  - Includes both malignant and benign samples for training the model effectively.

IMPLEMENTATION

IMPORT

Import the Dataset. Check the information and gain the insights of the dataset

DATA PREPROCESSING

Key steps in data preparation include handling missing values, feature scaling, and data splitting into training and testing sets. These preprocessing steps ensure the model learns from clean, structured data, leading to improved accuracy in predictions.

TRAIN

Train the data according to the model designed

TEST

Test the data and check the accuracy, error and other metric info

Preprocessing:

IMPORT

Importing Required dataset

- Pandas: Data manipulation and analysis.
- Seaborn: Statistical data visualization.
- Matplotlib: General-purpose plotting library.
- NumPy: Numerical operations on arrays.
- Scikit-learn: Machine learning algorithms.

	ID	Diagnosis	Feature_1	Feature_2	Feature_3	Feature_4	Feature_5	\
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	

	Feature_6	Feature_7	Feature_8	...	Feature_21	Feature_22	Feature_23	\
0	0.27760	0.3001	0.14710	...	25.38	17.33	184.60	
1	0.07864	0.0869	0.07017	...	24.99	23.41	158.80	
2	0.15990	0.1974	0.12790	...	23.57	25.53	152.50	
3	0.28390	0.2414	0.10520	...	14.91	26.50	98.87	
4	0.13280	0.1980	0.10430	...	22.54	16.67	152.20	

	Feature_24	Feature_25	Feature_26	Feature_27	Feature_28	Feature_29	\
0	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	
1	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	
2	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	
3	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	
4	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	

	Feature_30
0	0.11890
1	0.08902
2	0.08758
3	0.17300
4	0.07678

1st few rows of data

PREPROCESSING

```
(569, 32)
```

Shape of the dataset
the dataset has 569 rows of patient records, each described by 38 columns of features.

Checking any null value present
there are no null values

```
ID             0
Diagnosis      0
Feature_1      0
Feature_2      0
Feature_3      0
Feature_4      0
Feature_5      0
Feature_6      0
Feature_7      0
Feature_8      0
Feature_9      0
Feature_10     0
Feature_11     0
Feature_12     0
Feature_13     0
Feature_14     0
Feature_15     0
Feature_16     0
Feature_17     0
Feature_18     0
Feature_19     0
Feature_20     0
Feature_21     0
Feature_22     0
Feature_23     0
Feature_24     0
Feature_25     0
Feature_26     0
Feature_27     0
Feature_28     0
Feature_29     0
```

```
[4]:  0
```

Checking any duplicate value present
there are no duplicate values

PREPROCESSING

	Diagnosis	Feature_1	Feature_2	Feature_3	Feature_4	Feature_5	Feature_6	Feature_7	Feature_8	Feature_9	...	Feature_21	Feature_22	Feature_23	Feature_24	Fe
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	...	25.38	17.33	184.60	2019.0	
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	...	24.99	23.41	158.80	1956.0	
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	...	23.57	25.53	152.50	1709.0	
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	...	14.91	26.50	98.87	567.7	
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	...	22.54	16.67	152.20	1575.0	

5 rows × 31 columns

converting categorical values in the 'Diagnosis' column of a dataset into numerical values

Mapping

Malignant = 1

Benign = 0

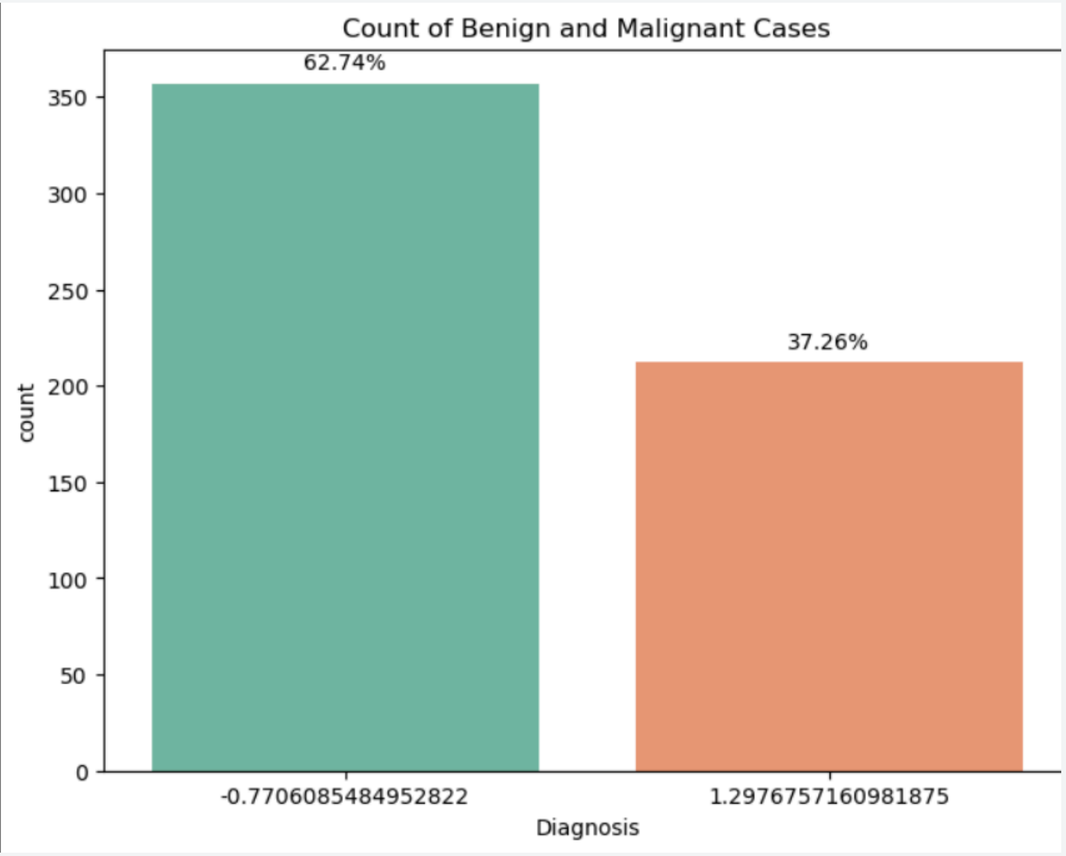
statistical summary of the dataset,

```
[9]: data.describe().T #statistical summary of a dataset
```

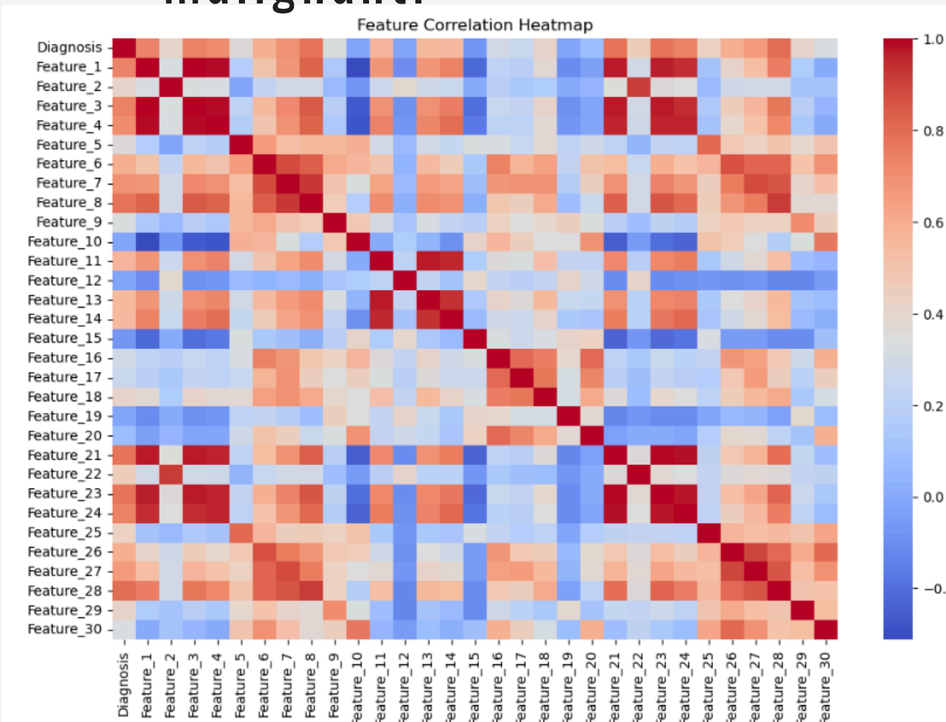
[9]:

	count	mean	std	min	25%	50%	75%	max
Diagnosis	569.0	0.372583	0.483918	0.000000	0.000000	0.000000	1.000000	1.00000
Feature_1	569.0	14.127292	3.524049	6.981000	11.700000	13.370000	15.780000	28.11000
Feature_2	569.0	19.289649	4.301036	9.710000	16.170000	18.840000	21.800000	39.28000
Feature_3	569.0	91.969033	24.298981	43.790000	75.170000	86.240000	104.100000	188.50000
Feature_4	569.0	654.889104	351.914129	143.500000	420.300000	551.100000	782.700000	2501.00000
Feature_5	569.0	0.096360	0.014064	0.052630	0.086370	0.095870	0.105300	0.16340
Feature_6	569.0	0.104341	0.052813	0.019380	0.064920	0.092630	0.130400	0.34540
Feature_7	569.0	0.088799	0.079720	0.000000	0.029560	0.061540	0.130700	0.42680
Feature_8	569.0	0.048919	0.038803	0.000000	0.020310	0.033500	0.074000	0.20120
Feature_9	569.0	0.181162	0.027414	0.106000	0.161900	0.179200	0.195700	0.30400
Feature_10	569.0	0.062798	0.007060	0.049960	0.057700	0.061540	0.066120	0.09744
Feature_11	569.0	0.405172	0.277313	0.111500	0.232400	0.324200	0.478900	2.87300
Feature_12	569.0	1.216853	0.551648	0.360200	0.833900	1.108000	1.474000	4.88500
Feature_13	569.0	2.866059	2.021855	0.757000	1.606000	2.287000	3.357000	21.98000
Feature_14	569.0	40.337079	45.491006	6.802000	17.850000	24.530000	45.190000	542.20000
Feature_15	569.0	0.007041	0.003003	0.001713	0.005169	0.006380	0.008146	0.03113
Feature_16	569.0	0.025478	0.017908	0.002252	0.013080	0.020450	0.032450	0.13540

VISUALIZATION



creates and displays a count plot using Seaborn to visualize the distribution of the target variable "Diagnosis" in DataFrame, showing the counts of majority of "Benign" over "Malignant" cases. Approximately 62.74% of the cases are benign, while 37.26% are malignant.



generates and displays a correlation heatmap using Seaborn,The diagonal line from the top-left to the bottom-right represents the correlation of each feature with itself, which is always 1.

SCALING

Z-score scaling standardizes all numerical features in the data DataFrame to have a mean of 0 and a standard deviation of 1 using Z-score scaling.Z-score scaling is less sensitive to outliers, making it a good choice for datasets with outliers

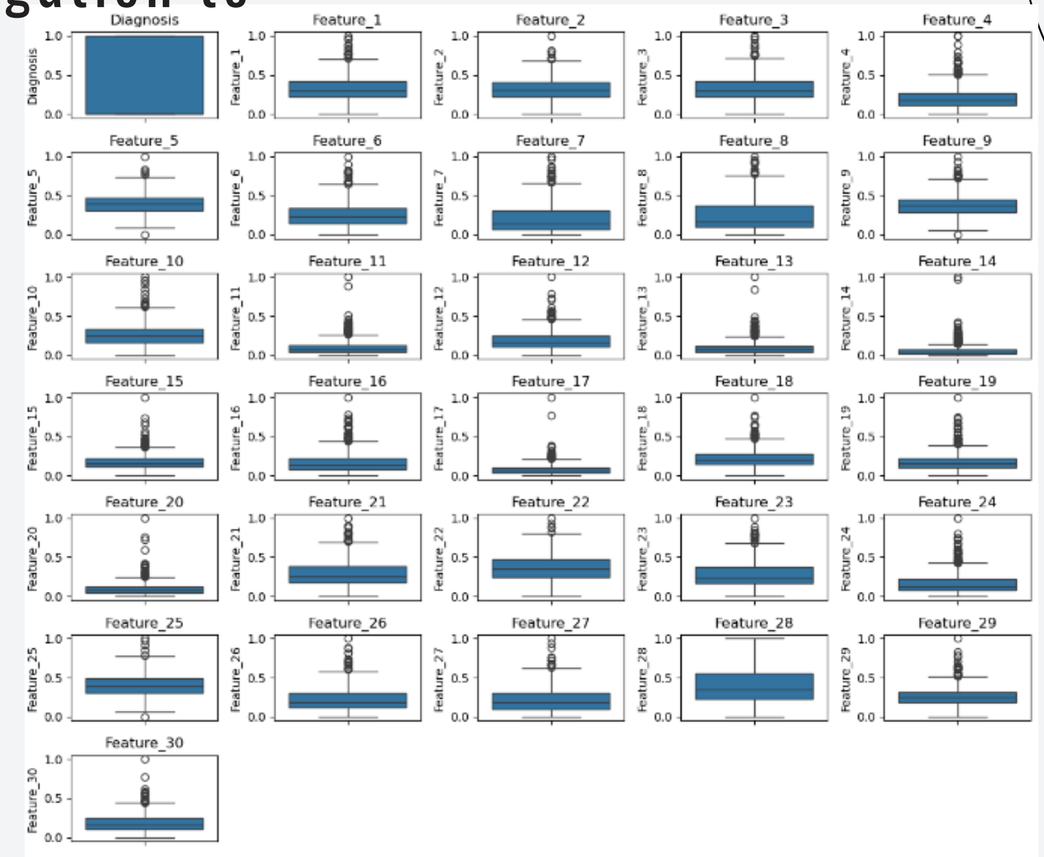
```
Diagnosis  Feature_1  Feature_2  Feature_3  Feature_4  Feature_5  \
 1.297676   1.097064  -2.073335   1.269934   0.984375   1.568466
 1.297676   1.829821  -0.353632   1.685955   1.908708  -0.826962
 1.297676   1.579888   0.456187   1.566503   1.558884   0.942210
 1.297676  -0.768909   0.253732  -0.592687  -0.764464   3.283553
 1.297676   1.750297  -1.151816   1.776573   1.826229   0.280372

Feature_6  Feature_7  Feature_8  Feature_9  ...  Feature_21  Feature_22
 3.283515   2.652874   2.532475   2.217515  ...    1.886690   -1.359293
-0.487072  -0.023846   0.548144   0.001392  ...    1.805927   -0.369203
 1.052926   1.363478   2.037231   0.939685  ...    1.511870   -0.023974
 3.402909   1.915897   1.451707   2.867383  ...   -0.281464    0.133984
 0.539340   1.371011   1.428493  -0.009560  ...    1.298575   -1.466770

Feature_23  Feature_24  Feature_25  Feature_26  Feature_27  Feature_28
  2.303601    2.001237    1.307686    2.616665    2.109526    2.296076
  1.535126    1.890489   -0.375612   -0.430444   -0.146749    1.087084
  1.347475    1.456285    0.527407    1.082932    0.854974    1.955000
 -0.249939   -0.550021    3.394275    3.893397    1.989588    2.175786
  1.338539    1.220724    0.220556   -0.313395    0.613179    0.729259

Feature_29  Feature_30
  2.750622    1.937015
 -0.243890    0.281190
  1.152255    0.201391
  6.046041    4.935010
 -0.868353   -0.397100

rows x 31 columns]
```

OUTLIERS

```
Count of Outliers for Each Numerical Column:
{'Diagnosis': 0, 'Feature_1': 14, 'Feature_2': 7, 'Feature_3': 13, 'Feature_4': 25, 'Feature_5': 6, 'Feature_6': 16, 'Feature_7': 18, 'Feature_8': 10,
'Feature_9': 15, 'Feature_10': 15, 'Feature_11': 38, 'Feature_12': 20, 'Feature_13': 38, 'Feature_14': 65, 'Feature_15': 30, 'Feature_16': 28, 'Featur
e_17': 22, 'Feature_18': 19, 'Feature_19': 27, 'Feature_20': 28, 'Feature_21': 17, 'Feature_22': 5, 'Feature_23': 15, 'Feature_24': 35, 'Feature_25':
7, 'Feature_26': 16, 'Feature_27': 12, 'Feature_28': 0, 'Feature_29': 23, 'Feature_30': 24}
```

Checking outliers using IQR Method number of outliers present in each feature,outlier needs further investigation to check valid or invalid

checking outliers using BoxPlots



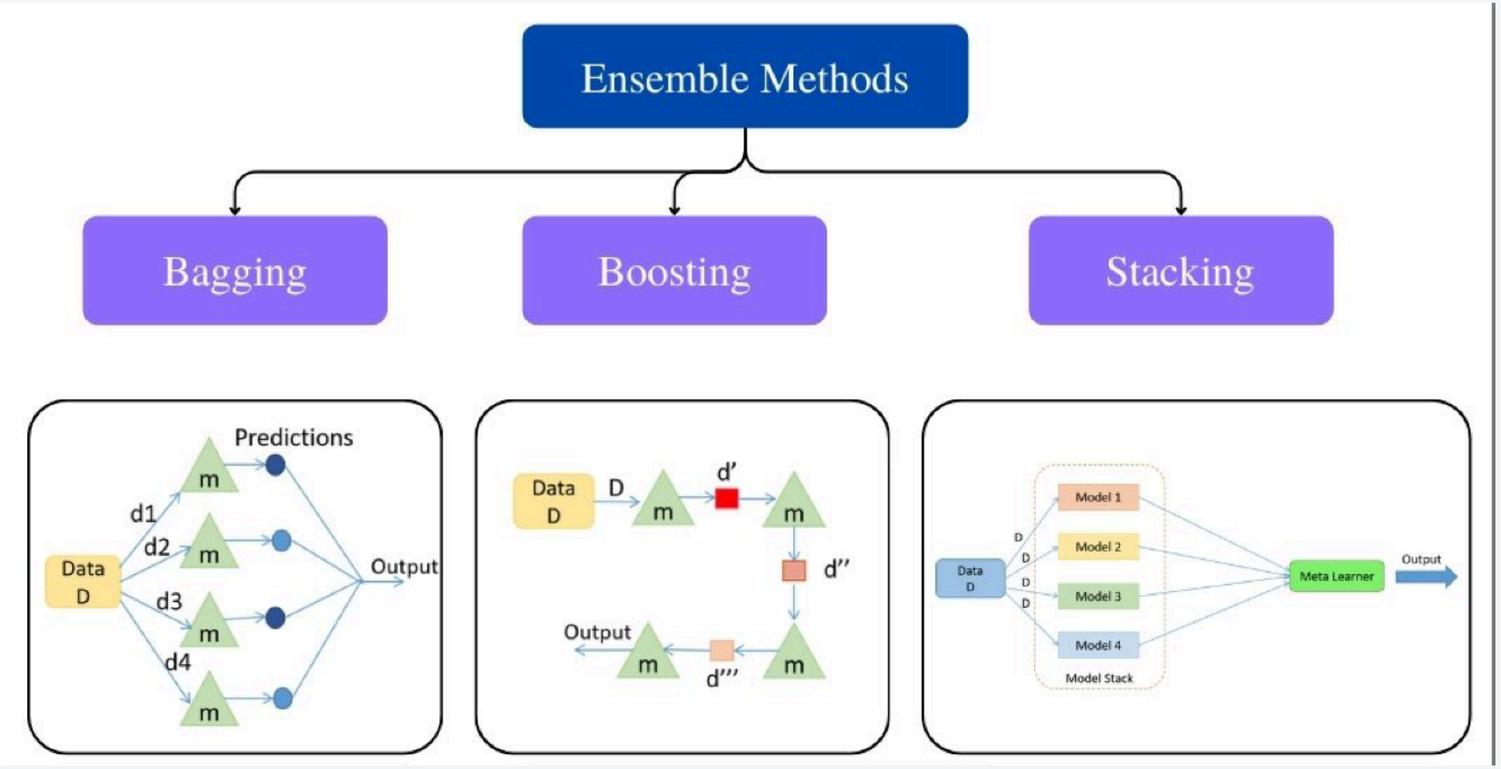
SPLITTING OF DATA

```
Training data shape: (455, 30)
Testing data shape: (114, 30)
```

Splitting the data for training and testing
we have split the data in ratio 80:20

ENSEMBLE LEARNING AND ADABOOST

Ensemble learning is a powerful technique that combines multiple models to improve predictive performance. By combining diverse models, ensemble methods can often outperform individual models.

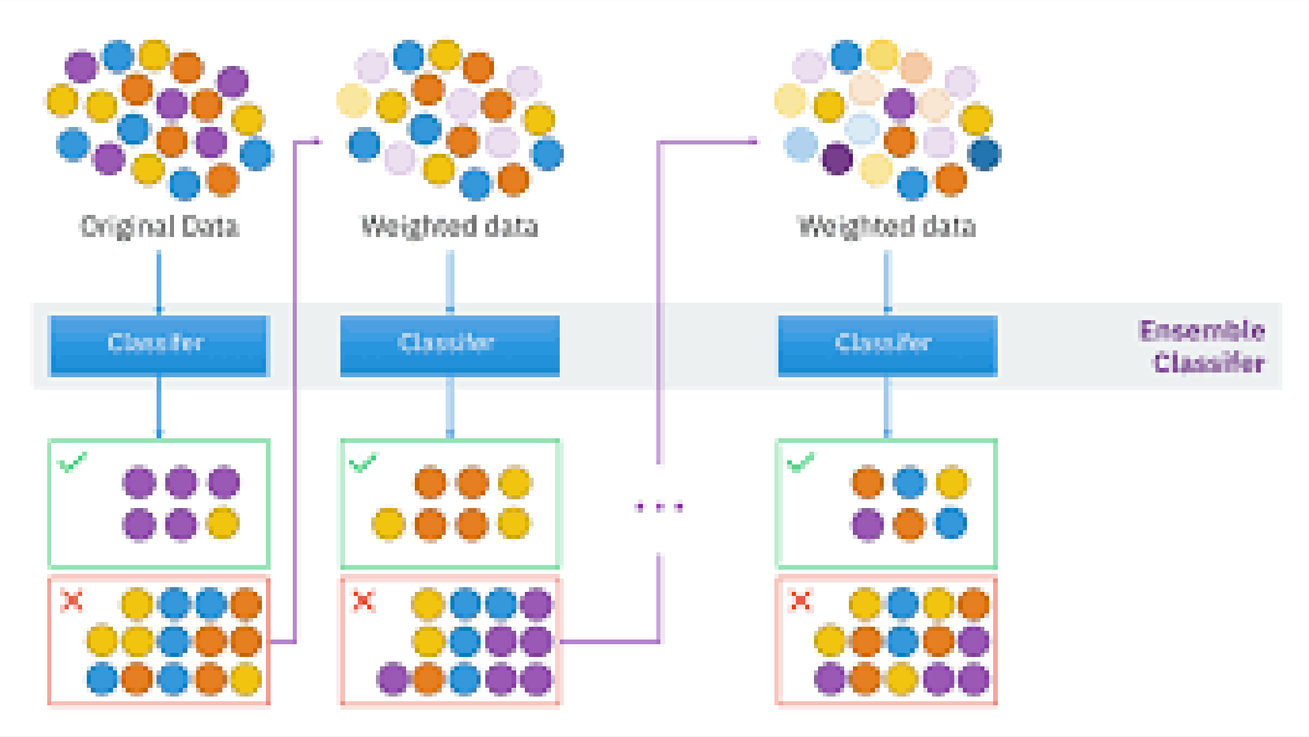


- Bagging: Train multiple decision trees on different bootstrapped samples of the data and average their predictions. Example:Random forest
- Boosting : Sequentially train weak models, focusing on misclassified instances, and combine them into a strong model.
  Example:AdaBoost,Gradient Boosting
- Stacking : Train multiple base models, use their predictions as features for a meta-model, and train the meta-model to make the final prediction.

IMPLEMENTING ADABOOST CLASSIFIER

```
AdaBoostClassifier(random_state=42)
```

- Adaboost classifier is implemented which combines multiple weak learners to create a strong classifier.



- Adaboost iteratively trains weak learners, focusing on misclassified samples by assigning higher weights to them. The final classifier is a weighted combination of these weak learners, with weights determined by their accuracy. This adaptive approach allows AdaBoost to effectively learn from its mistakes and achieve high predictive performance.

PRECISION AND RECALL IN BREAST CANCER DIAGNOSIS

- Precision: This metric measures the accuracy of positive predictions. It indicates the proportion of correctly identified malignant tumors out of all predicted malignant tumors. A high precision score is crucial to minimize false positives, which could lead to unnecessary biopsies and patient anxiety.

- Recall: This metric measures the model's ability to correctly identify all positive cases. It indicates the proportion of actual malignant tumors that are correctly identified by the model. A high recall score is essential to minimize false negatives, which could delay diagnosis and treatment.

PERFORMANCE METRICS

(WITH OUTLIERS)

	Metric	Score
0	Accuracy	0.982456
1	Precision	0.976744
2	Recall	0.976744
3	F1 Score	0.976744

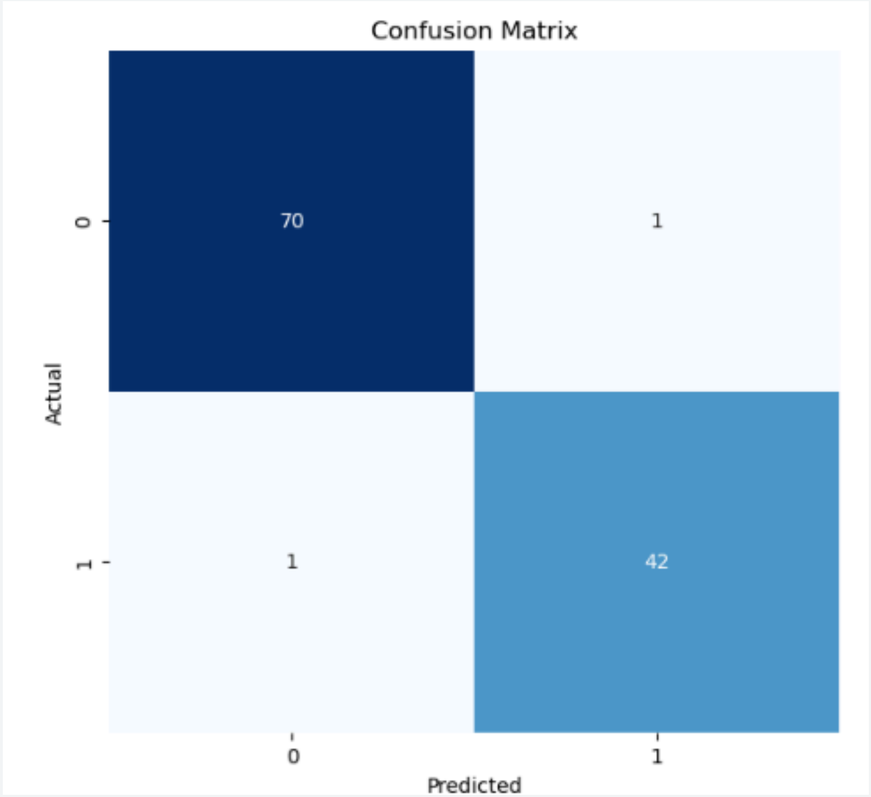
High Accuracy, Precision, Recall, and F1-Score: This model exhibits excellent performance across all metrics. It is highly accurate in its predictions, and it effectively identifies both positive and negative cases.

(WITHOUT OUTLIERS)

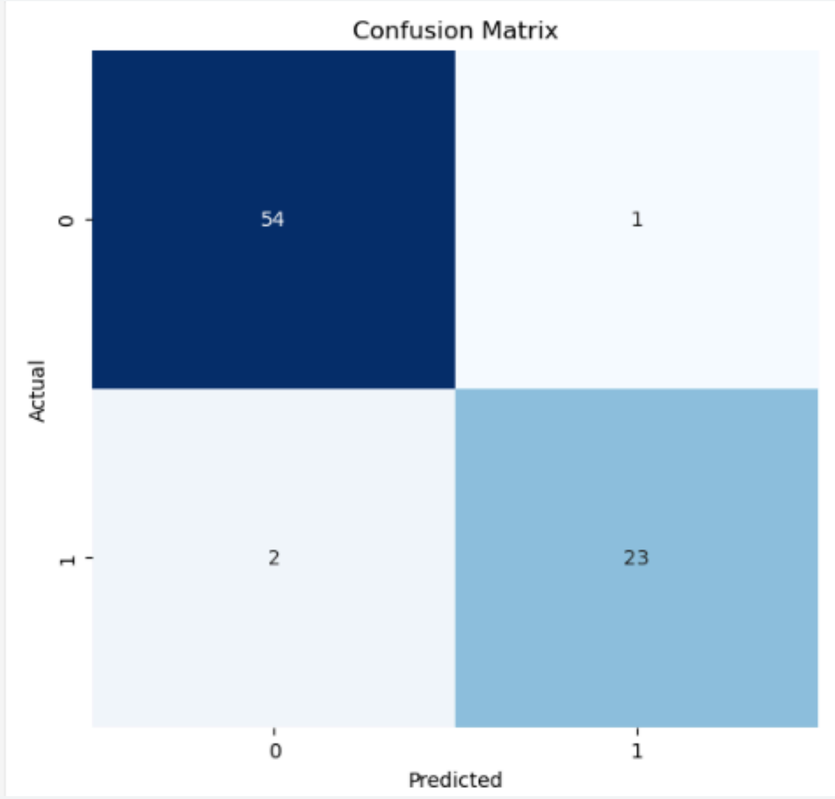
	Metric	Score
0	Accuracy	0.9625
1	Precision	0.958333
2	Recall	0.92
3	F1 Score	0.938776

It has a high precision and lower recall value indicates that the model might miss some actual positive cases. This could be a concern depending on the specific application and the relative importance of false positives and false negatives.

CONFUSION MATRIX EVALUATION



WITH OUTLIERS



WITHOUT OUTLIERS

IN OUR CASE, THE MODEL HAS A HIGH ACCURACY, PRECISION, AND RECALL WITH OUTLIERS. THIS SUGGESTS THAT THE MODEL WITH OUTLIERS IS PERFORMING WELL IN CORRECTLY CLASSIFYING BOTH NEGATIVE AND POSITIVE INSTANCES.

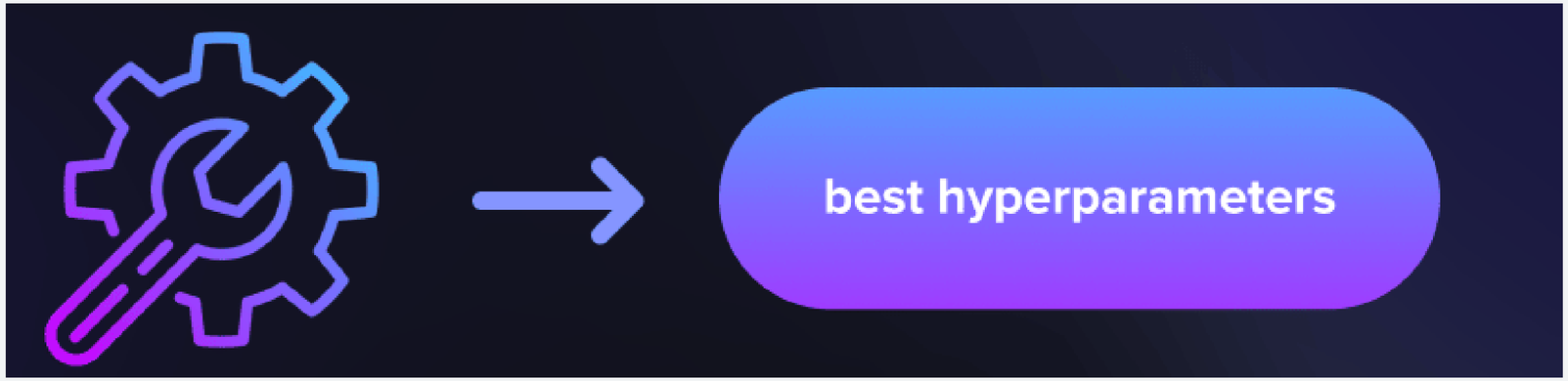
HYPERPARAMETER TUNING

Handling Imbalanced Data:

- Class Imbalance: The dataset may have an imbalance between malignant and benign cases.
- Addressing Imbalance: Proper hyperparameter tuning can help mitigate the impact of class imbalance and improve the model's ability to accurately classify minority classes.

Preventing Overfitting and Underfitting:

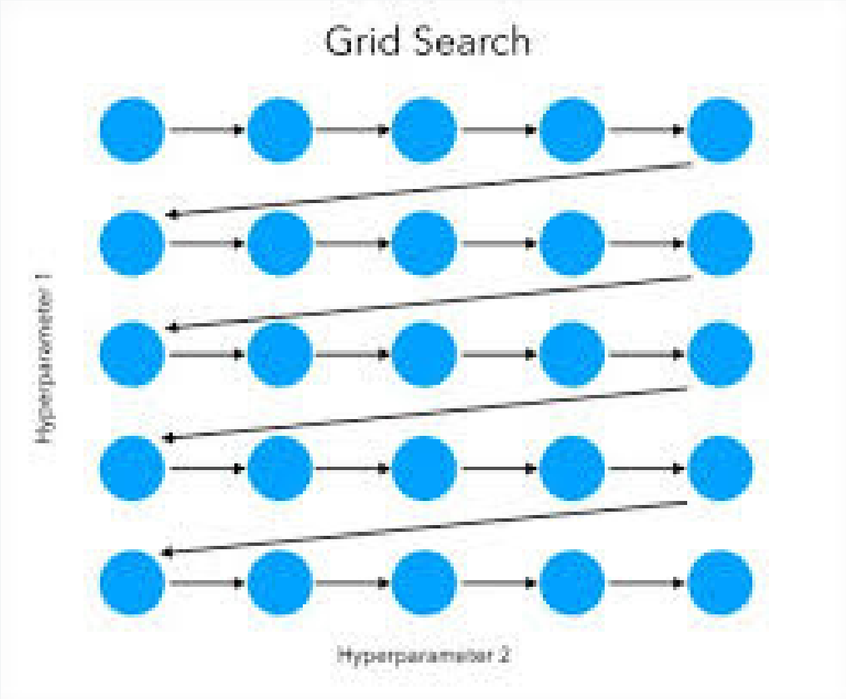
- Regularization: Hyperparameters like regularization strength can control model complexity and prevent overfitting.
- Model Capacity: The choice of the model architecture and its complexity can be influenced by hyperparameter tuning.



GRID SEARCH

```
grid_search = GridSearchCV(estimator=ada_classifier,
                           param_grid=param_grid,
                           scoring='accuracy',
                           cv=5,
                           n_jobs=-1,
                           verbose=1)
grid_search.fit(X_train_smote, y_train_smote)
```

The code snippet is using GridSearchCV to systematically evaluate different combinations of hyperparameters for the AdaBoost classifier. The goal is to find the best-performing model on the given dataset. By tuning the hyperparameters, we can improve the model's accuracy and generalization ability.



GRID SERACH:

- Defines a grid of hyperparameter values.
- Trains the model for each combination.
- Evaluates the model's performance on a validation set.
- Selects the best-performing combination.

CLASSIFICATION REPORT

BEFORE HYPERTUNING

Classification Report (Without Hyperparameter Tuning):

	precision	recall	f1-score	support
0	0.97	0.98	0.98	107
1	0.97	0.95	0.96	64
accuracy			0.97	171
macro avg	0.97	0.97	0.97	171
weighted avg	0.97	0.97	0.97	171

Final Accuracy (Without Hyperparameter Tuning): 0.97

AFTER HYPERTUNING

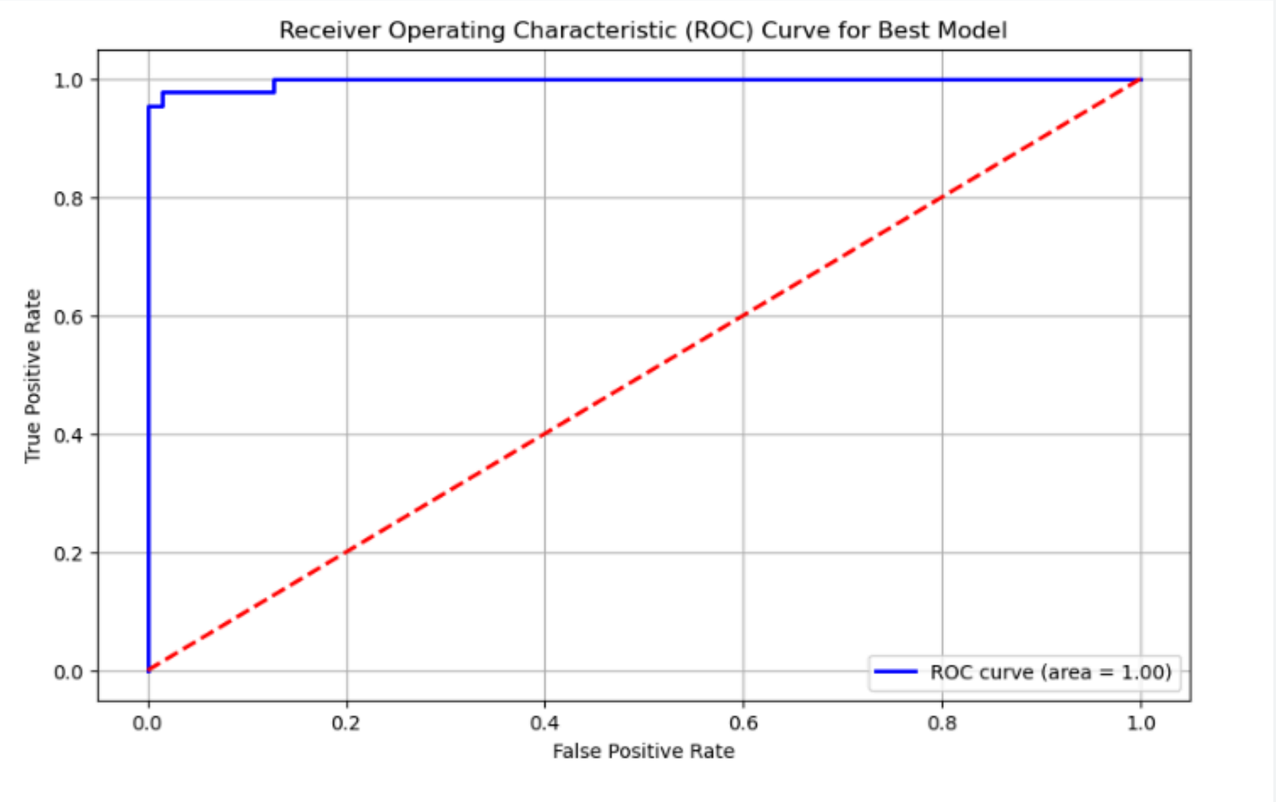
Classification Report (With Hyperparameter Tuning):

	precision	recall	f1-score	support
0	0.96	1.00	0.98	107
1	1.00	0.94	0.97	64
accuracy			0.98	171
macro avg	0.98	0.97	0.97	171
weighted avg	0.98	0.98	0.98	171

Final Accuracy (With Hyperparameter Tuning): 0.98

The model with hyperparameter tuning outperformed the baseline model. It achieved an accuracy of 0.98, a macro-average F1-score of 0.97, and a weighted-average F1-score of 0.98.This indicates a significant improvement in overall performance and a better balance between precision and recall for both classes.

ROC CURVE



The ROC curve demonstrates that the model exhibits excellent performance after hypertuning. It has a high ability to correctly classify both positive and negative instances. Such a model would be highly valuable in breast cancer detection, especially where accurate predictions are critical.

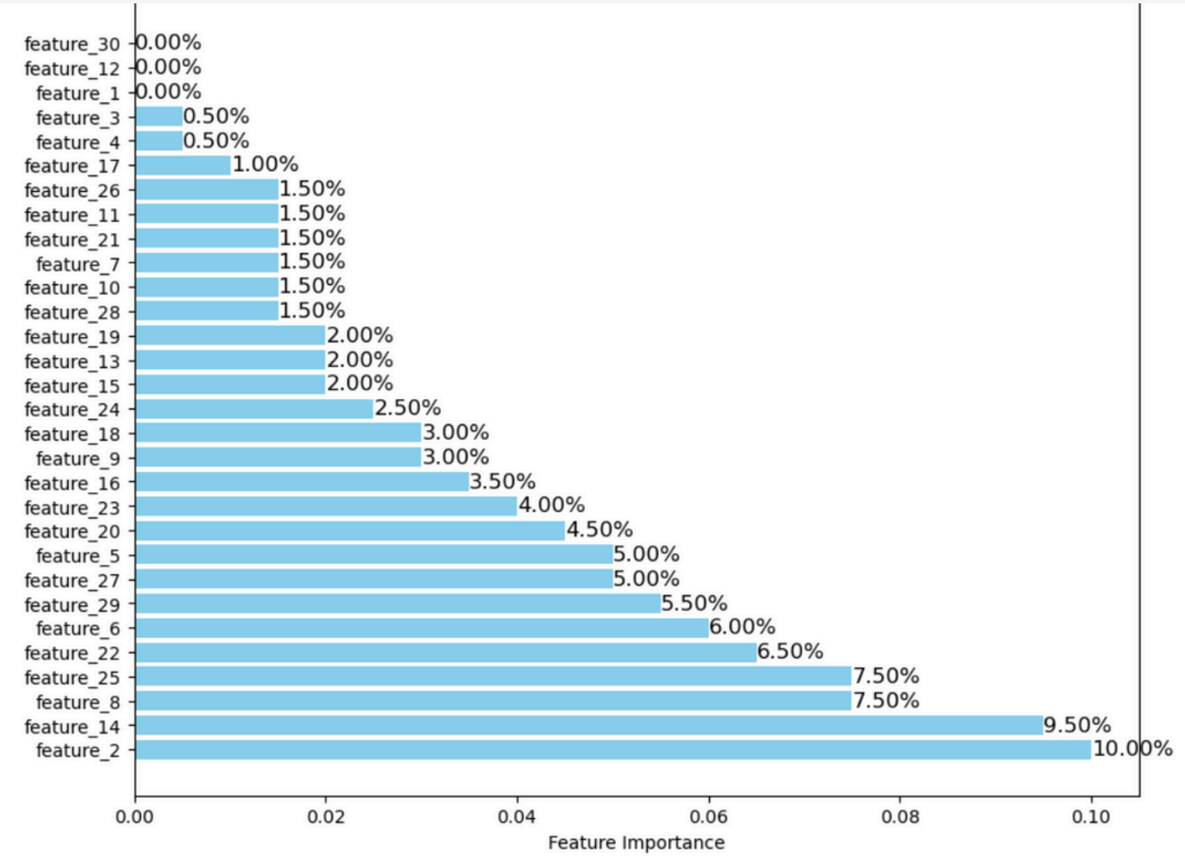
FEATURE IMPORTANCE

```
# Create a DataFrame to hold the feature names and their importance scores
feature_importance_df = pd.DataFrame({
'Feature': X.columns,
'Importance': importances
})
```

```
# Sort the DataFrame by importance in descending order
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
```

The code snippet creates a DataFrame to store feature names and their corresponding importance scores. It then sorts the DataFrame by importance in descending order, allowing for easy visualization and analysis. This technique is useful for understanding the contribution of each feature to the model's predictions, aiding in feature selection, model interpretation, and potential feature engineering.

FEATURE IMPORTANCE PLOT



The feature importance plot provides valuable insights into the factors that significantly influence the prediction of breast cancer malignancy. Features 2, 14, 8, 22, and 25 emerge as the most critical predictors. These features likely capture key characteristics of the tumor cells that are strongly associated with malignancy.

OUR TEAM

V Sharmila

Everest
Cantu

Priyadharshini R

Ceo Of Ingoude
Company

THANK YOU