BREAST CANCER DETECTION

Using AdaBoost Classifier

SUBMITTED BY

TEAM - 1

Gayathri R
Krishna Raja Sree B
Naveen S
Shaik Ayesha Parveen
Joseph Boban

INDEX

List of Tables:

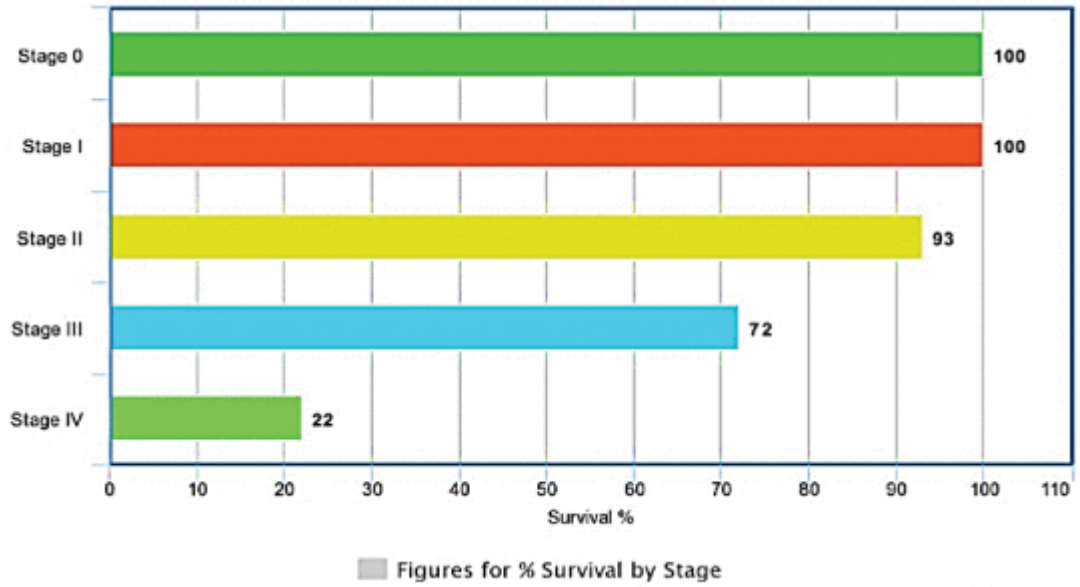
List of Graphs:

1. Introduction to Breast Cancer and Medical Diagnosis

Breast cancer is one of the most common cancers affecting women worldwide, contributing significantly to cancer-related deaths. Early detection is crucial for improving survival rates, as timely treatment can lead to better outcomes.

Traditional breast cancer diagnosis relies on physical exams, imaging techniques like mammograms, and laboratory tests such as biopsies. While effective, these methods can be expensive, time-consuming, and occasionally prone to errors.

Machine learning offers a modern approach to assist in breast cancer detection by analyzing large medical datasets to identify patterns and make predictions. This project uses machine learning techniques to develop a reliable model for detecting breast cancer early, helping improve accuracy and supporting healthcare systems in providing better patient care.



2. Dataset Information

The UCI Breast Cancer Wisconsin Diagnostic Dataset is a widely used open-source dataset for breast cancer research and diagnostics. It contains detailed measurements of cell features, including size, shape, texture, and more, extracted from digitized images of fine needle aspirates of breast masses.

The experimental results have been demonstrated with the help of the dataset Link: https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic

Introduction

The **Diagnostic Wisconsin Breast Cancer Database** is a multivariate dataset widely used in the health and medicine domain for research and diagnostic purposes. It is designed for **classification tasks**, helping differentiate between benign and malignant tumors. The dataset consists of **569 instances** and **30 real-valued features** that describe various physical

characteristics of cell nuclei, such as radius, texture, perimeter, area, and smoothness. Derived from fine needle aspirate images of breast tissue, these features provide valuable insights into tumor properties. Its structured nature and relevance to medical diagnostics make it a benchmark for developing machine learning models in breast cancer detection.

3. Data Cleaning and Preprocessing

- **shape** - dimensions of the dataset, providing the number of rows and columns as a tuple. The dataset consists of 569 rows and 32 columns.

```
(569, 32)
```

- **Columns** and their names

  The dataset consists of 32 columns, out of which 31 are numeric (int64 and float64) and 1 is categorical (object) variables.

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave_points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave_points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave_points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

- Null values represent **missing or incomplete data** in the dataset, which can affect the accuracy and performance of the AI model.

  There are **no missing values** in the Dataset.

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
```

```
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

- The dataset contains **no duplicate rows**, ensuring 100% unique data for accurate analysis.

```
Number of Rows with Duplicates: 0
Percentage of Duplicate Rows: 0.00%
```

- **info()** - provides a quick summary of a DataFrame. It is a useful tool for understanding the structure and quality of the dataset at a glance.

  The number of rows and columns, column names, their data types, the count of non-null values in each column, and the memory usage is displayed.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      569 non-null    int64
 1   diagnosis               569 non-null    object
 2   radius_mean             569 non-null    float64
 3   texture_mean            569 non-null    float64
 4   perimeter_mean          569 non-null    float64
 5   area_mean               569 non-null    float64
 6   smoothness_mean         569 non-null    float64
 7   compactness_mean        569 non-null    float64
 8   concavity_mean          569 non-null    float64
 9   concave points_mean     569 non-null    float64
 10  symmetry_mean           569 non-null    float64
 11  fractal_dimension_mean  569 non-null    float64
 12  radius_se               569 non-null    float64
 13  texture_se              569 non-null    float64
 14  perimeter_se            569 non-null    float64
 15  area_se                 569 non-null    float64
 16  smoothness_se           569 non-null    float64
 17  compactness_se          569 non-null    float64
 18  concavity_se            569 non-null    float64
 19  concave points_se       569 non-null    float64
 20  symmetry_se             569 non-null    float64
 21  fractal_dimension_se    569 non-null    float64
 22  radius_worst            569 non-null    float64
 23  texture_worst           569 non-null    float64
 24  perimeter_worst         569 non-null    float64
 25  area_worst              569 non-null    float64
 26  smoothness_worst        569 non-null    float64
 27  compactness_worst       569 non-null    float64
 28  concavity_worst         569 non-null    float64
 29  concave points_worst    569 non-null    float64
 30  symmetry_worst          569 non-null    float64
 31  fractal_dimension_worst 569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

- **head -** Displays the **first five rows** of the dataset, giving a quick preview of the initial records and structure of the data.

```
         0  1      2      3       4       5        6        7        8   \
0    842302  M  17.99  10.38  122.80  1001.0  0.11840  0.27760  0.3001
1    842517  M  20.57  17.77  132.90  1326.0  0.08474  0.07864  0.0869
2  84300903  M  19.69  21.25  130.00  1203.0  0.10960  0.15990  0.1974
3  84348301  M  11.42  20.38   77.58   386.1  0.14250  0.28390  0.2414
4  84358402  M  20.29  14.34  135.10  1297.0  0.10030  0.13280  0.1980

         9   ...     22     23      24      25      26      27      28      29  \
0  0.14710  ...  25.38  17.33  184.60  2019.0  0.1622  0.6656  0.7119  0.2654
1  0.07017  ...  24.99  23.41  158.80  1956.0  0.1238  0.1866  0.2416  0.1860
2  0.12790  ...  23.57  25.53  152.50  1709.0  0.1444  0.4245  0.4504  0.2430
3  0.10520  ...  14.91  26.50   98.87   567.7  0.2098  0.8663  0.6869  0.2575
4  0.10430  ...  22.54  16.67  152.20  1575.0  0.1374  0.2050  0.4000  0.1625

       30       31
0  0.4601  0.11890
1  0.2750  0.08902
2  0.3613  0.08758
3  0.6638  0.17300
4  0.2364  0.07678

[5 rows x 32 columns]
```

- **tail** - Shows the **last five rows** of the dataset, providing a glimpse of the final records and structure of the data.

	0	1	2	3	4	5	6	7	8	9	...	22	23	24	25	26	27	28	29	30	31
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039

5 rows × 32 columns

- **describe().T** - generates summary statistics for numerical columns in the dataset and transposes the result making it easier to interpret the data.

	count	mean	std	min	25%	50%	75%	max
id	569.0	3.037183e+07	1.250206e+08	8670.000000	869218.000000	906024.000000	8.813129e+06	9.113205e+08
radius_mean	569.0	1.412729e+01	3.524049e+00	6.981000	11.700000	13.370000	1.578000e+01	2.811000e+01
texture_mean	569.0	1.928965e+01	4.301036e+00	9.710000	16.170000	18.840000	2.180000e+01	3.928000e+01
perimeter_mean	569.0	9.196903e+01	2.429898e+01	43.790000	75.170000	86.240000	1.041000e+02	1.885000e+02
area_mean	569.0	6.548891e+02	3.519141e+02	143.500000	420.300000	551.100000	7.827000e+02	2.501000e+03
smoothness_mean	569.0	9.636028e-02	1.406413e-02	0.052630	0.086370	0.095870	1.053000e-01	1.634000e-01
compactness_mean	569.0	1.043410e-01	5.281276e-02	0.019380	0.064920	0.092630	1.304000e-01	3.454000e-01
concavity_mean	569.0	8.879932e-02	7.971981e-02	0.000000	0.029560	0.061540	1.307000e-01	4.268000e-01
concave_points_mean	569.0	4.891915e-02	3.880284e-02	0.000000	0.020310	0.033500	7.400000e-02	2.012000e-01
symmetry_mean	569.0	1.811619e-01	2.741428e-02	0.106000	0.161900	0.179200	1.957000e-01	3.040000e-01
actal_dimension_mean	569.0	6.279761e-02	7.060363e-03	0.049960	0.057700	0.061540	6.612000e-02	9.744000e-02
radius_se	569.0	4.051721e-01	2.773127e-01	0.111500	0.232400	0.324200	4.789000e-01	2.873000e+00
texture_se	569.0	1.216853e+00	5.516484e-01	0.360200	0.833900	1.108000	1.474000e+00	4.885000e+00
perimeter_se	569.0	2.866059e+00	2.021855e+00	0.757000	1.606000	2.287000	3.357000e+00	2.198000e+01
area_se	569.0	4.033708e+01	4.549101e+01	6.802000	17.850000	24.530000	4.519000e+01	5.422000e+02
smoothness_se	569.0	7.040979e-03	3.002518e-03	0.001713	0.005169	0.006380	8.146000e-03	3.113000e-02

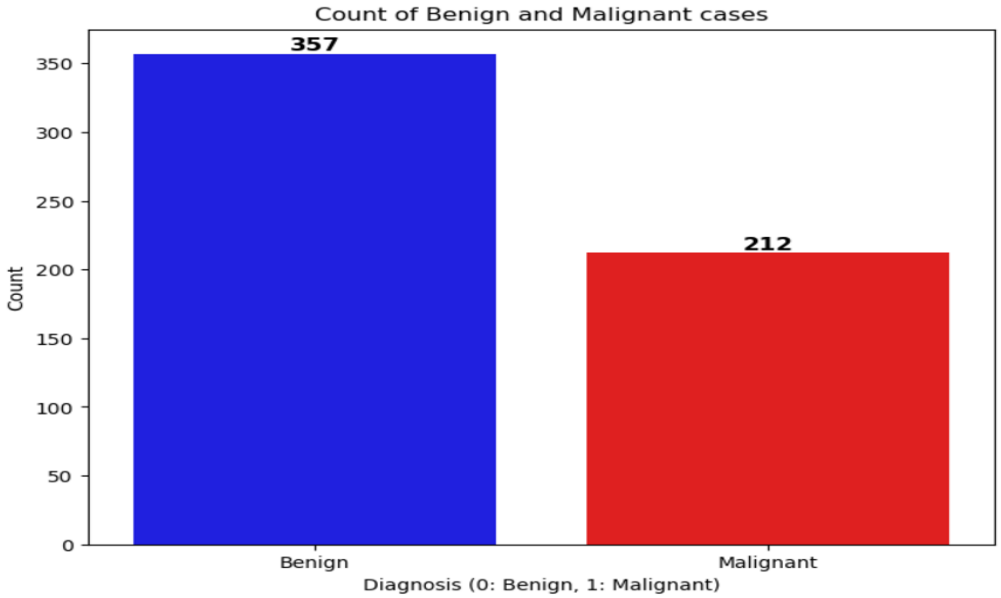
compactness_se	569.0	2.547814e-02	1.790818e-02	0.002252	0.013080	0.020450	3.245000e-02	1.354000e-01
concavity_se	569.0	3.189372e-02	3.018606e-02	0.000000	0.015090	0.025890	4.205000e-02	3.960000e-01
concave_points_se	569.0	1.179614e-02	6.170285e-03	0.000000	0.007638	0.010930	1.471000e-02	5.279000e-02
symmetry_se	569.0	2.054230e-02	8.266372e-03	0.007882	0.015160	0.018730	2.348000e-02	7.895000e-02
fractal_dimension_se	569.0	3.794904e-03	2.646071e-03	0.000895	0.002248	0.003187	4.558000e-03	2.984000e-02
radius_worst	569.0	1.626919e+01	4.833242e+00	7.930000	13.010000	14.970000	1.879000e+01	3.604000e+01
texture_worst	569.0	2.567722e+01	6.146258e+00	12.020000	21.080000	25.410000	2.972000e+01	4.954000e+01
perimeter_worst	569.0	1.072612e+02	3.360254e+01	50.410000	84.110000	97.660000	1.254000e+02	2.512000e+02
area_worst	569.0	8.805831e+02	5.693570e+02	185.200000	515.300000	686.500000	1.084000e+03	4.254000e+03
smoothness_worst	569.0	1.323686e-01	2.283243e-02	0.071170	0.116600	0.131300	1.460000e-01	2.226000e-01
compactness_worst	569.0	2.542650e-01	1.573365e-01	0.027290	0.147200	0.211900	3.391000e-01	1.058000e+00
concavity_worst	569.0	2.721885e-01	2.086243e-01	0.000000	0.114500	0.226700	3.829000e-01	1.252000e+00
concave_points_worst	569.0	1.146062e-01	6.573234e-02	0.000000	0.064930	0.099930	1.614000e-01	2.910000e-01
symmetry_worst	569.0	2.900756e-01	6.186747e-02	0.156500	0.250400	0.282200	3.179000e-01	6.638000e-01
actal_dimension_worst	569.0	8.394582e-02	1.806127e-02	0.055040	0.071460	0.080040	9.208000e-02	2.075000e-01

4. Analysis by Plotting Graphs

Bar Chart

The ‘Count of Benign and Malignant cases’ represents the distribution of breast cancer diagnoses in the dataset. Here, M (Malignant) is encoded as 1, and B (Benign) as 0. This plot helps visualize the proportion of malignant and benign cases, providing insights into the class balance, which is crucial for model performance evaluation.

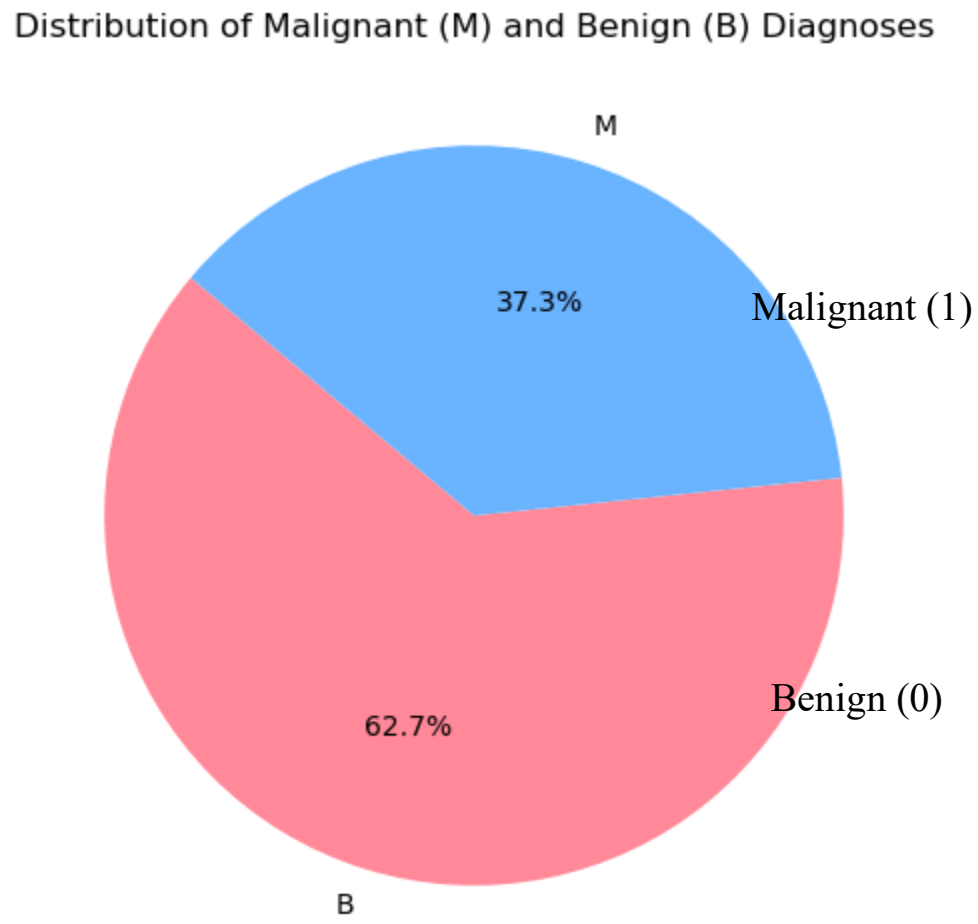
The counts of benign and malignant cases are 357 and 212 respectively, and in terms of percentage, they are 62.7% and 37.3%.



Pie Chart

A pie chart is a circular graph that shows how different categories contribute to a whole. Each slice represents a category and its size shows its proportion to the total. Pie charts are useful for displaying simple, proportional data.

Pie charts are appropriate for showing the relationship between parts and a whole for categorical data. They can be used to visualize how each subset contributes to the entire dataset



Heat Map

A heatmap is a graphical representation of data where values are depicted using varying colors. In data analysis, it is commonly used to **visualize correlation matrices**, showing the relationships between variables.

The intensity of the colour indicates the strength of the correlation, with warmer colors (e.g., red) typically representing higher positive correlations and cooler colors (e.g., blue) representing negative correlations. Heatmaps are valuable for identifying patterns, trends, and multicollinearity in datasets.

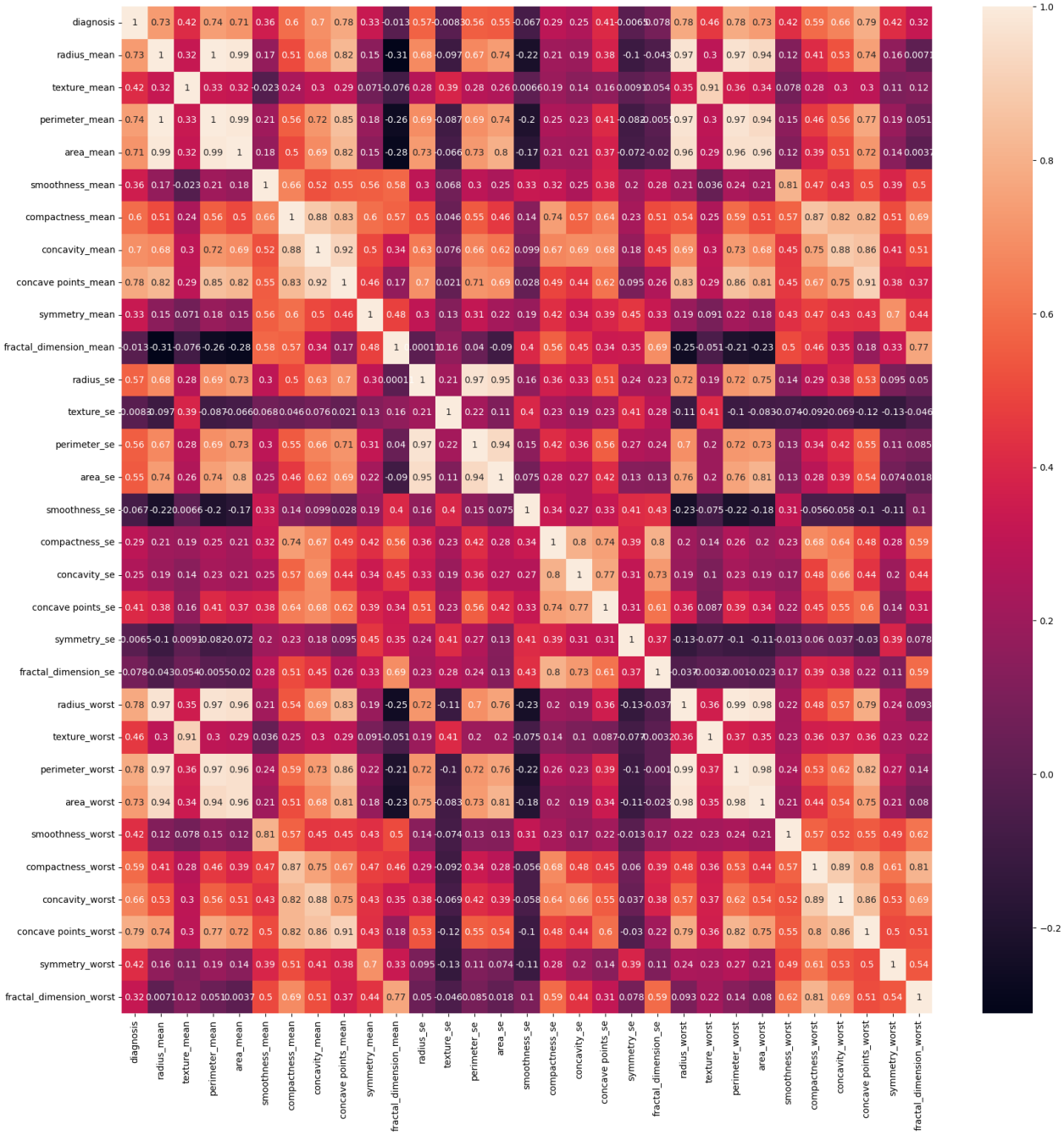


Fig: Heat Map

Pair-Plot

A pairplot is a **matrix of scatter plots** that visualizes relationships between multiple numerical features in a dataset. Each plot in the pairplot shows the **relationship between two variables**, while the diagonal often displays the distribution of individual features (e.g., histograms).

This helps identify patterns, correlations, and potential outliers between the features. Pairplots are particularly useful for detecting linear or non-linear relationships, understanding feature interactions, and assessing data quality across different pairs of variables.

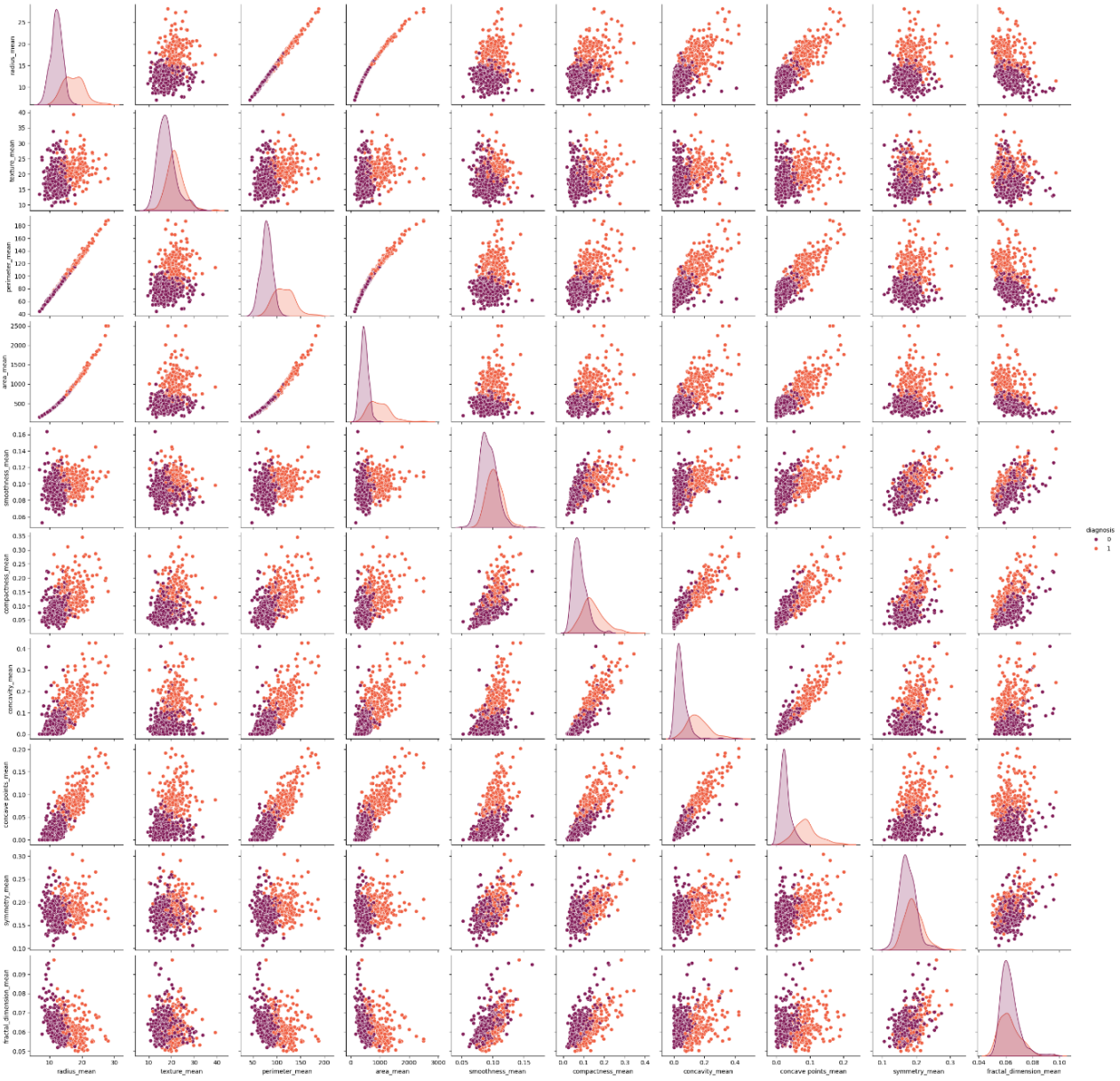


Fig: Pair-Plot

5. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is the process of analyzing and visualizing a dataset to summarize its main characteristics, often using statistical and graphical techniques. The goal of EDA is to uncover patterns, identify anomalies, test hypotheses, and check assumptions before applying machine learning models. It helps to better understand the data, clean it, and prepare it for further analysis or modeling. Common EDA tasks include examining distributions, relationships between variables, missing values, and outliers using tools like histograms, scatter plots, box plots, and correlation matrices**.**

Outlier Detection

Outlier - are data points that significantly differ from other observations in a dataset. They can be much higher or lower than most of the other values, and they may indicate variability in the data, i.e., errors.

Detection Methods:

- **Visual methods**: Boxplots, histograms, and scatter plots can be used to visually spot outliers.
- **Statistical methods**:
    - **Z-score**: Identifies how far a data point is from the mean, measured in standard deviations.
    - **IQR (Interquartile Range)**: Identifies outliers by looking at the spread of the middle 50% of the data and finding values beyond a specific range.

IQR Method:

Steps to Calculate the IQR:

1. **Sort the data**: Arrange the data in ascending order.
2. **Find the Quartiles**:
    - **Q1 (First Quartile)**: The 25th percentile of the data, which is the value below which 25% of the data points fall.
    - **Q3 (Third Quartile)**: The 75th percentile of the data, which is the value below which 75% of the data points fall.

    The IQR is calculated as

$$\text{IQR} = Q3 - Q1$$

3. **Determine the Outlier Boundaries**:
    - The **Lower Bound** is calculated as

$$\text{Lower Bound} = Q1 - 1.5 \times \text{IQR}$$

    - The **Upper Bound** is calculated as

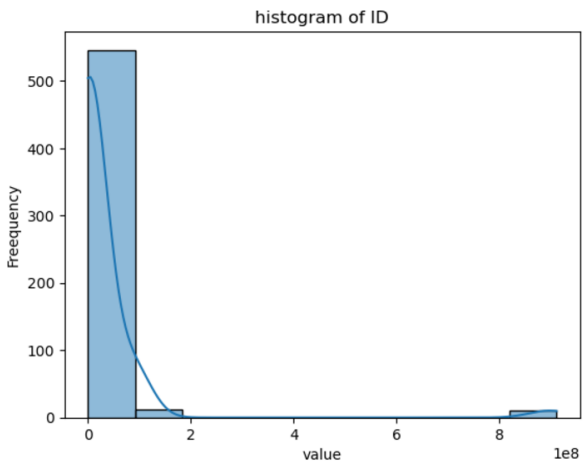
$$\text{Upper Bound} = Q3 + 1.5 \times \text{IQR}$$

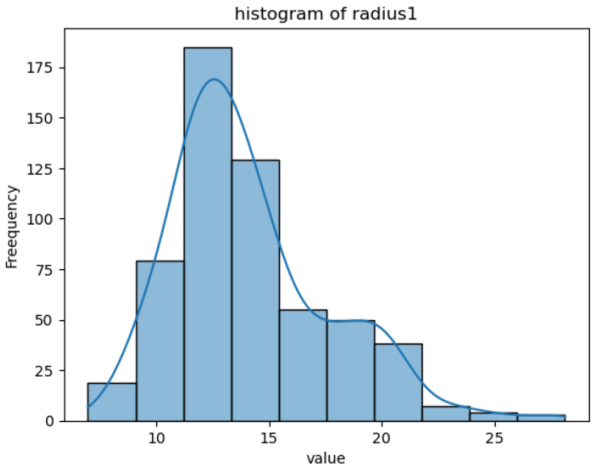
4. **Identify Outliers**:

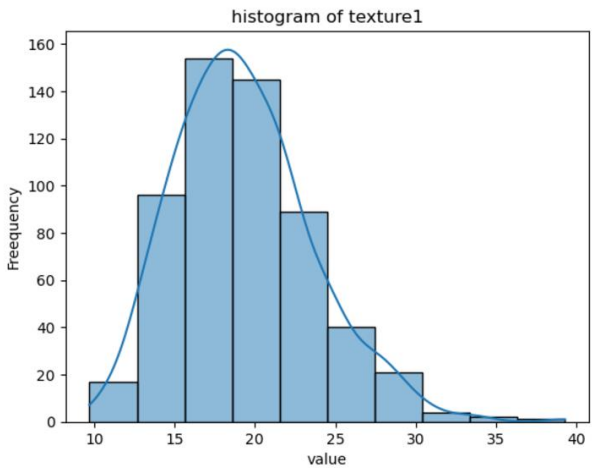
- Any data points below the **Lower Bound** or above the **Upper Bound** are considered outliers.

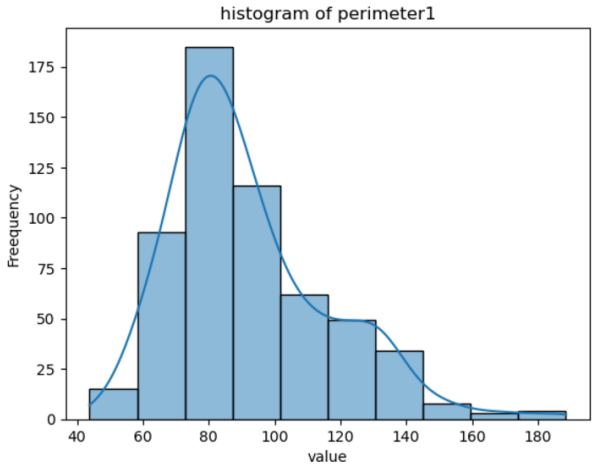
```
Number of outliers for 'ID': 81
Number of outliers for 'radius1': 14
Number of outliers for 'texture1': 7
Number of outliers for 'perimeter1': 13
Number of outliers for 'area1': 25
Number of outliers for 'smoothness1': 6
Number of outliers for 'compactness1': 16
Number of outliers for 'concavity1': 18
Number of outliers for 'concave_points1': 10
Number of outliers for 'symmetry1': 15
Number of outliers for 'fractal_dimension1': 15
Number of outliers for 'radius2': 38
Number of outliers for 'texture2': 20
Number of outliers for 'perimeter2': 38
Number of outliers for 'area2': 65
Number of outliers for 'smoothness2': 30
Number of outliers for 'compactness2': 28
Number of outliers for 'concavity2': 22
Number of outliers for 'concave_points2': 19
Number of outliers for 'symmetry2': 27
Number of outliers for 'fractal_dimension2': 28
Number of outliers for 'radius3': 17
Number of outliers for 'texture3': 5
Number of outliers for 'perimeter3': 15
Number of outliers for 'area3': 35
Number of outliers for 'smoothness3': 7
Number of outliers for 'compactness3': 16
Number of outliers for 'concavity3': 12
Number of outliers for 'concave_points3': 0
Number of outliers for 'symmetry3': 23
Number of outliers for 'fractal_dimension3': 24
```

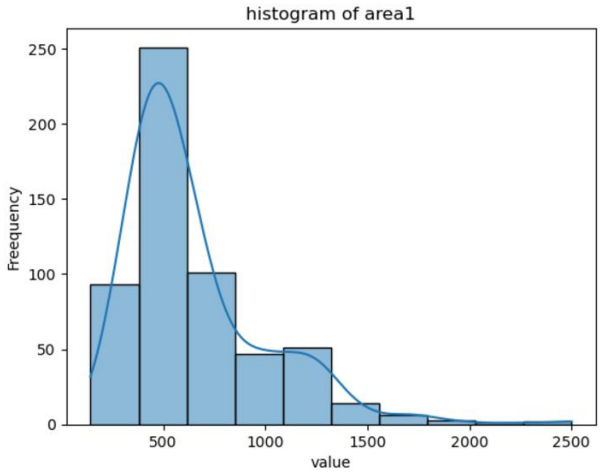
Distribution of Different Features:

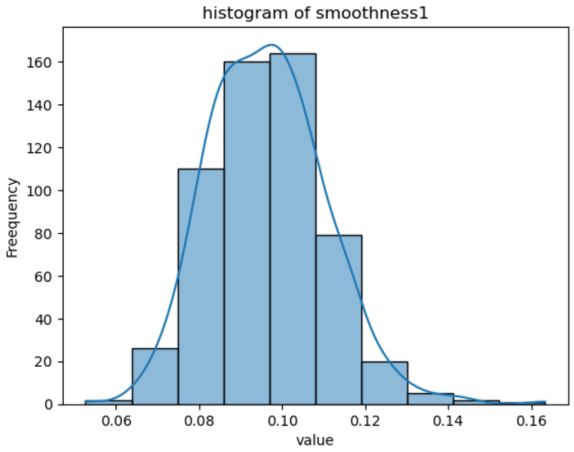


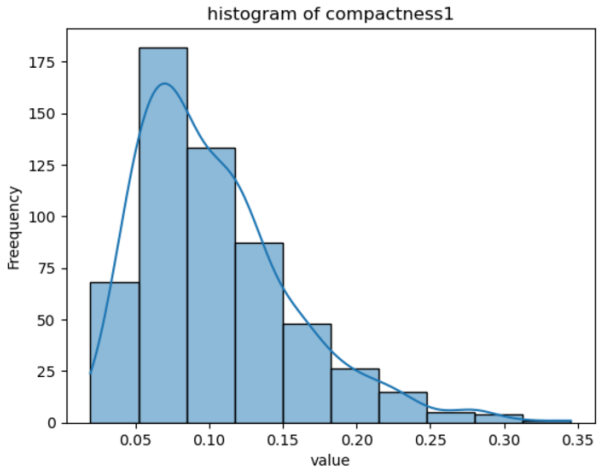


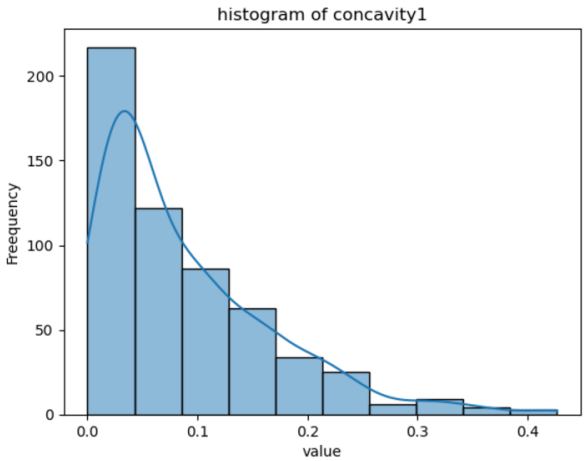

histogram of texture1
Freequency
value

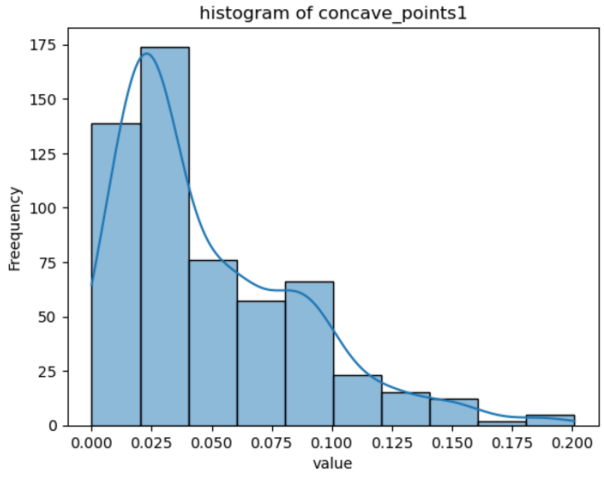

histogram of perimeter1
Freequency
value

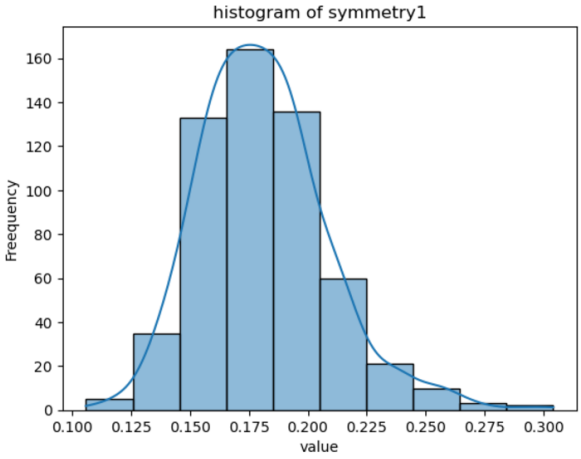

histogram of area1
Freequency
value

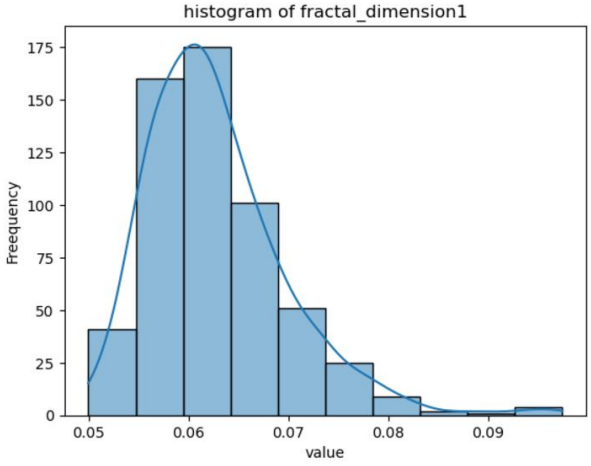

histogram of smoothness1
Freequency
value

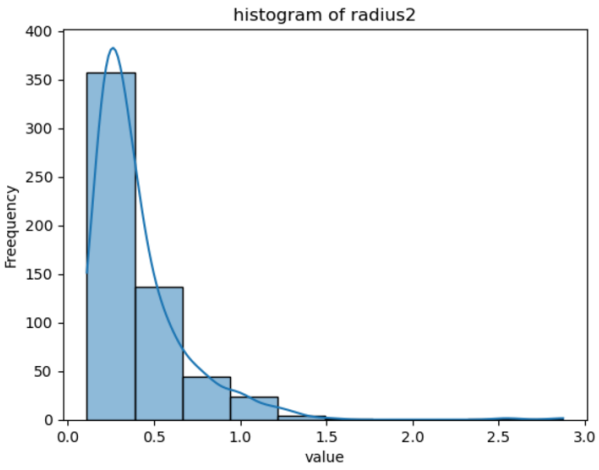

histogram of compactness1
Freequency
value

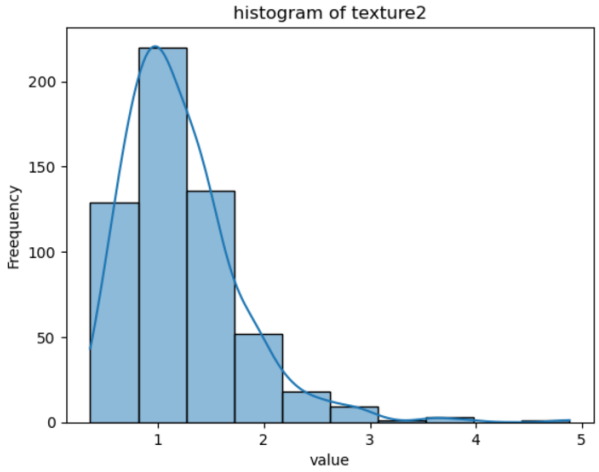

histogram of concavity1
Freequency
value

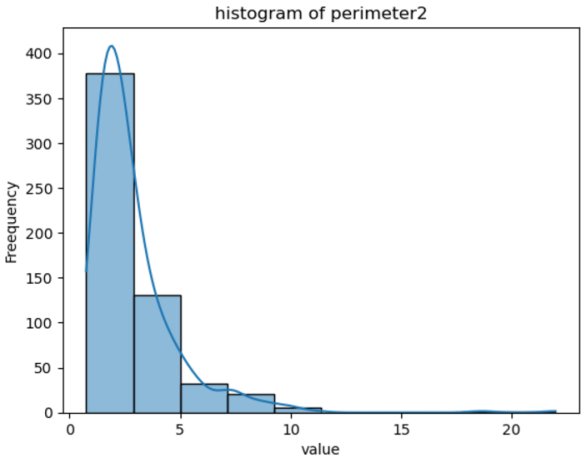

histogram of concave_points1
Freequency
value

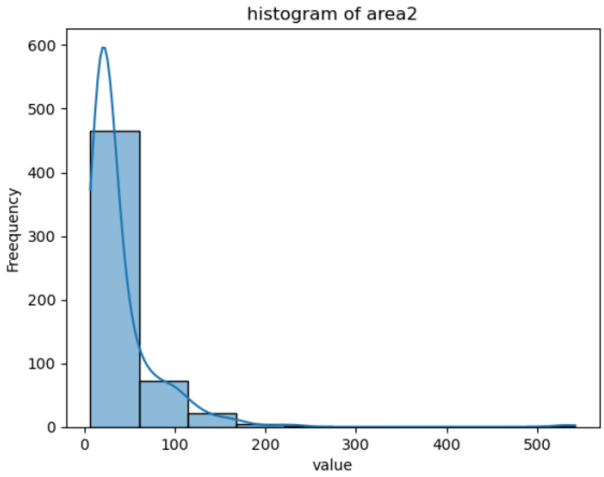

histogram of symmetry1
Freequency
value

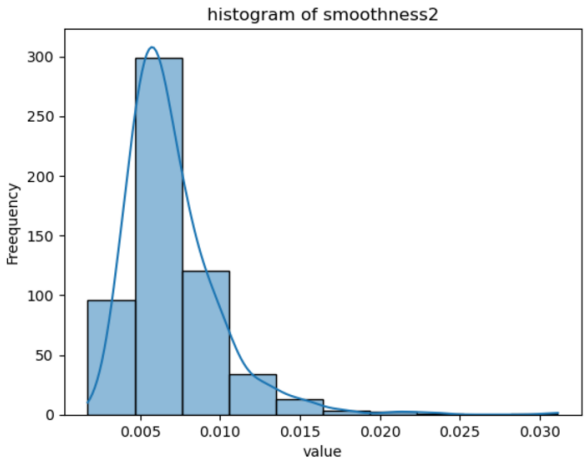

histogram of fractal_dimension1
Freequency
value

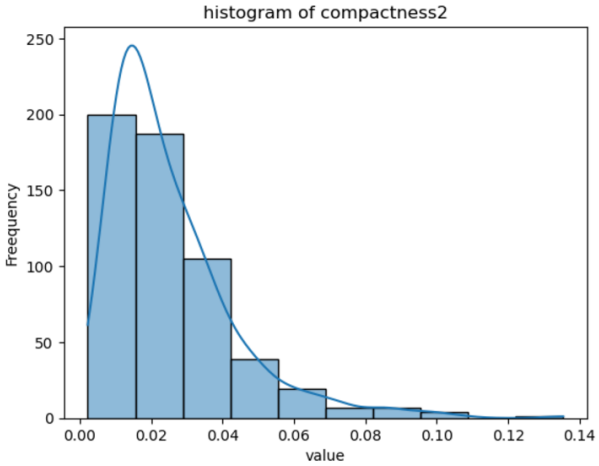

histogram of radius2
Freequency
value

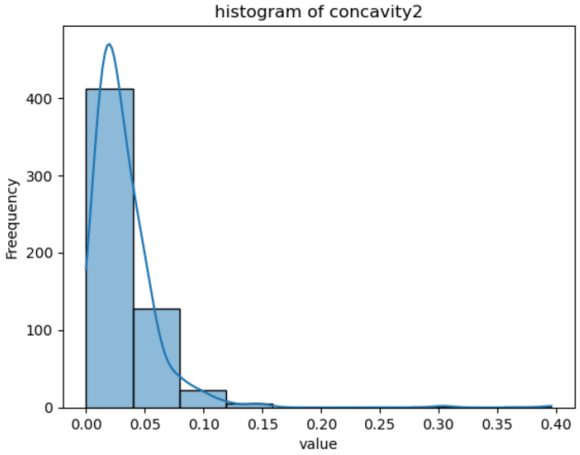

histogram of texture2
Freequency
value

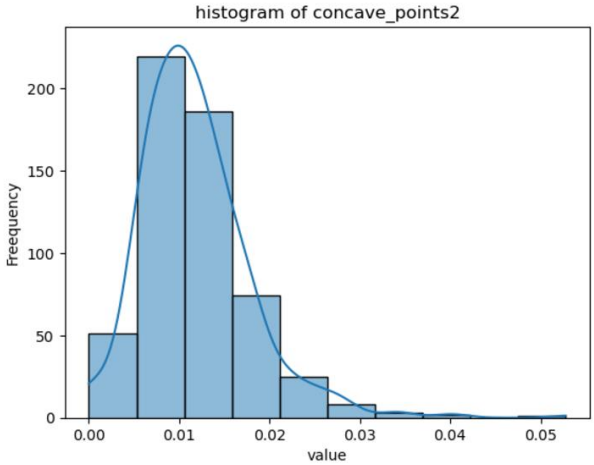

histogram of perimeter2
Freequency
value

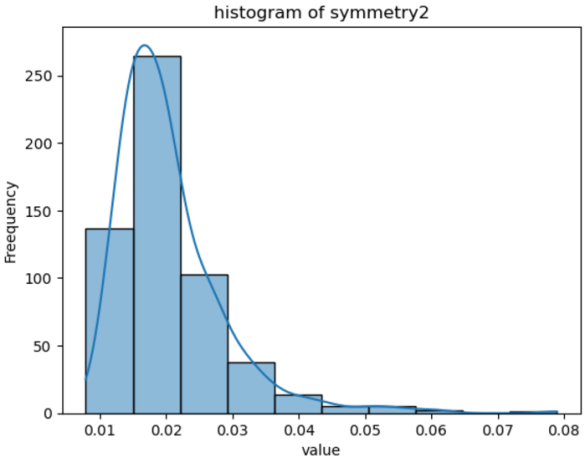

histogram of area2
Freequency
value
0
100
200
300
400
500
600

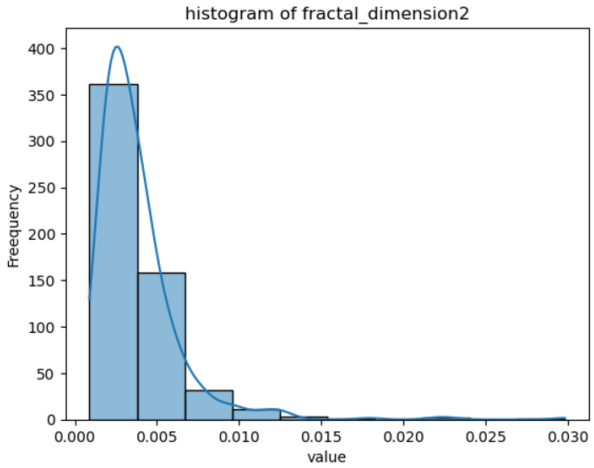

histogram of smoothness2
Freequency
value
0
50
100
150
200
250
300
0.005
0.010
0.015
0.020
0.025
0.030

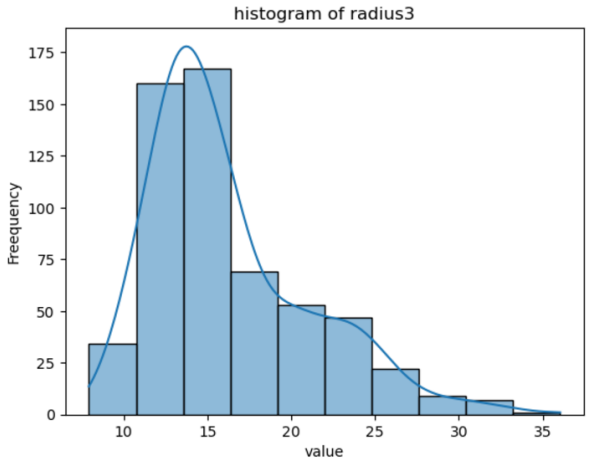

histogram of compactness2
Freequency
value
0
50
100
150
200
250
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14

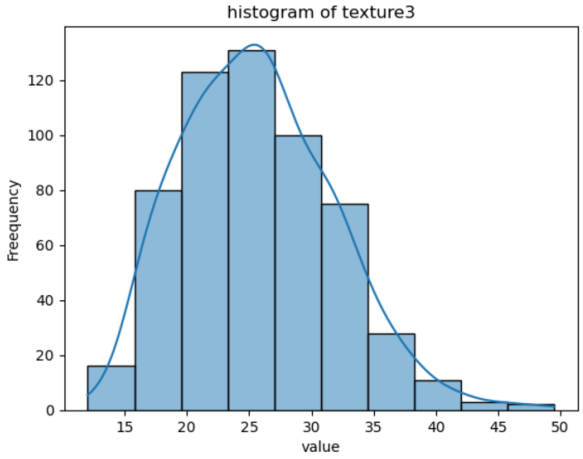

histogram of concavity2
Freequency
value
0
100
200
300
400
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40

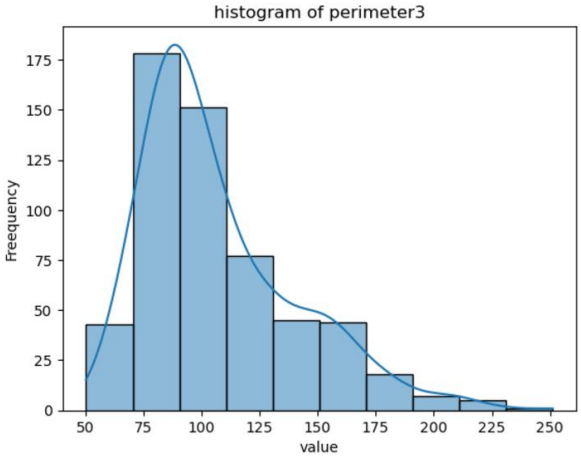

histogram of concave_points2
Freequency
value
0
50
100
150
200
0.00
0.01
0.02
0.03
0.04
0.05

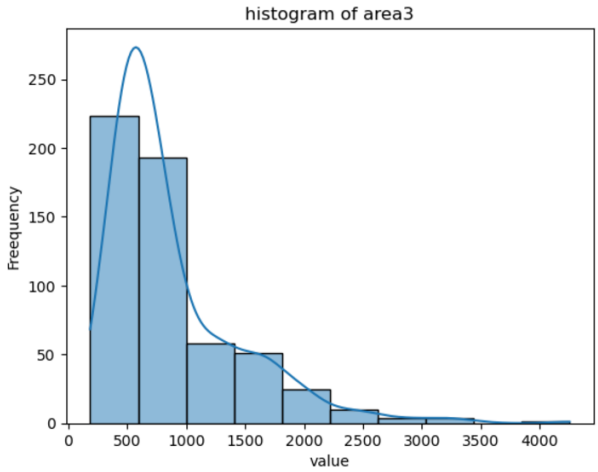

histogram of symmetry2
Freequency
value
0
50
100
150
200
250
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08

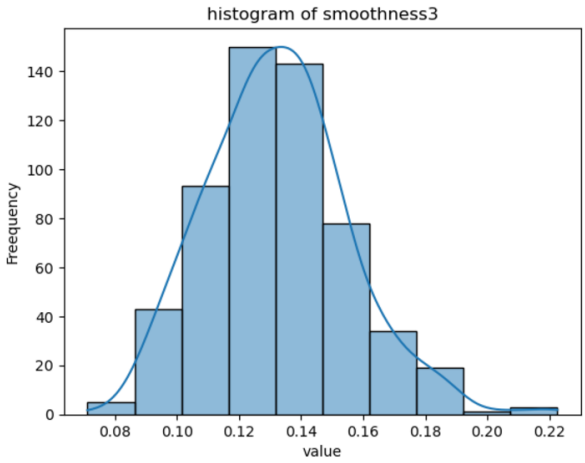

histogram of fractal_dimension2
Freequency
value
0.000
0.005
0.010
0.015
0.020
0.025
0.030

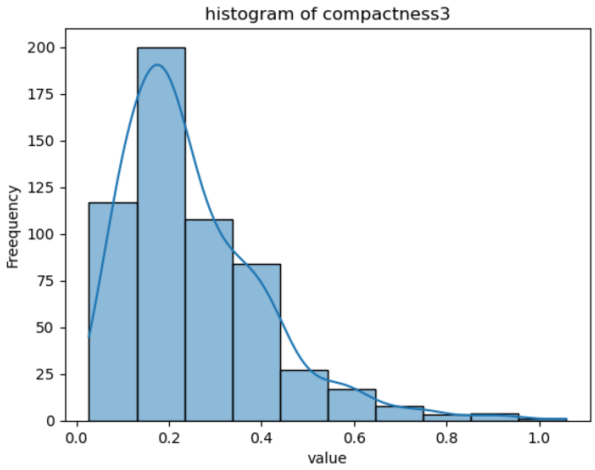

histogram of radius3
Freequency
value
10
15
20
25
30
35

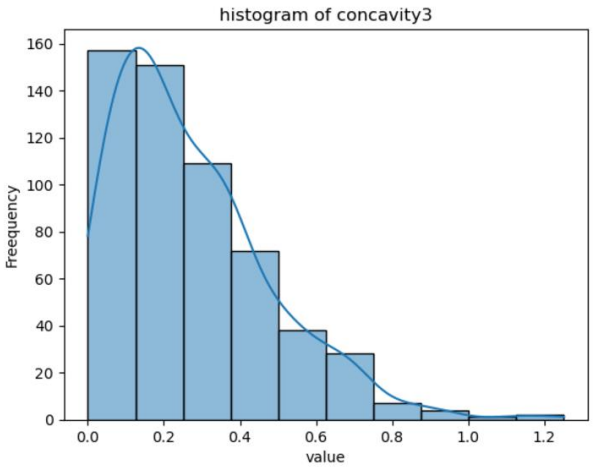

histogram of texture3
Freequency
value
15
20
25
30
35
40
45
50

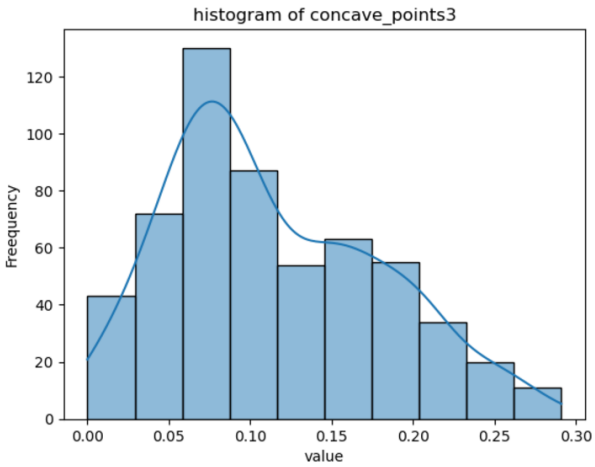

histogram of perimeter3
Freequency
value
50
75
100
125
150
175
200
225
250

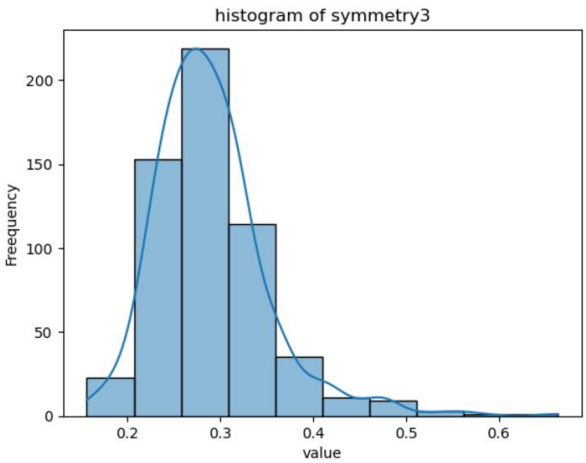

histogram of area3
Freequency
value
0
500
1000
1500
2000
2500
3000
3500
4000

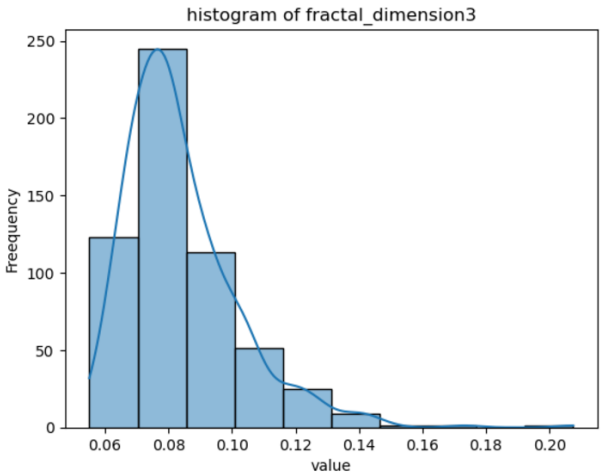

histogram of smoothness3
Freequency
value
0.08
0.10
0.12
0.14
0.16
0.18
0.20
0.22











The Distributions are either near normal or right skewed, therefore we will be using z scaling instead of log transformation and min-max scaling.

6. Data Processing Steps

- Dropping ID column as it doesn't carry any meaningful information.

```
data.drop('id', axis=1, inplace=True)
```

Checking whether ID column is dropped and displaying the count of the columns.

```
Index(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

Number of columns has reduced to 31.

```
(569, 31)
```

- Encoding values B and M

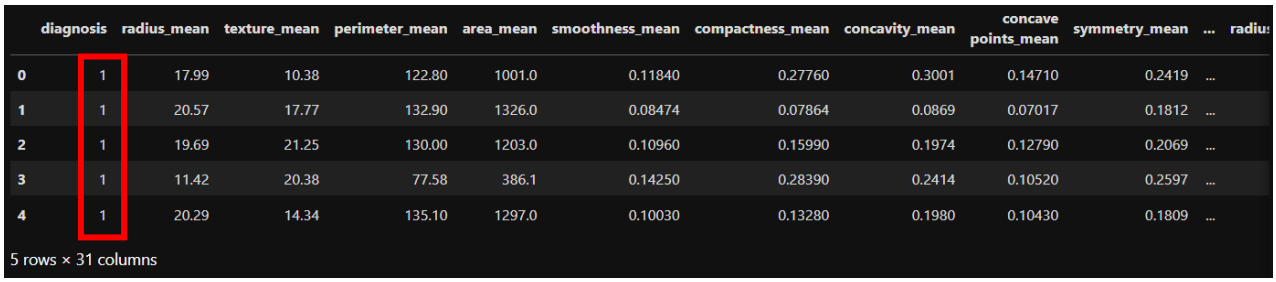
  Machine learning models typically work with numerical data, not categorical or string-based values. Encoding **B** and **M** as 0 and 1, converts these categorical labels into a numerical format that algorithms can process effectively.

```
data['Diagnosis']
```

```
0      M
1      M
2      M
3      M
4      M
      ..
564    M
565    M
566    M
567    M
568    B
Name: Diagnosis, Length: 569, dtype: object
```

```
data['diagnosis'] = data['diagnosis'].replace({'B': 0, 'M': 1})
```

The values of diagnosis – B and M are encoded as 0 and 1.



	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	radius
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	...	
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	...	
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	...	
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	...	
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	...	

5 rows × 31 columns

```
data['Diagnosis']

0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: Diagnosis, Length: 569, dtype: int64
```

7. VIF (Variance Inflation Factor)

A statistical measure used to **detect the presence of multicollinearity** in multiple linear regression models. Multicollinearity occurs when two or more independent variables in a regression model are highly correlated, which can lead to unreliable estimates of regression coefficients and increase the variance of the coefficient estimates.

For each independent variable Xi, the VIF is calculated as

$$VIF(X_i) = \frac{1}{1 - R_i^2}$$

Interpretation of VIF:

- **VIF = 1**: No correlation between the predictor and other predictors. There is no multicollinearity.
- **1 < VIF < 5**: A moderate correlation between the predictor and other predictors. Generally considered acceptable.
- **VIF > 5 or 10**: High multicollinearity. The predictor is highly correlated with other predictors, indicating that multicollinearity is a problem. A VIF value above 10 often suggests that the variable is redundant and should be removed from the model to improve model stability.

Keep the original data, which will be useful if any kind of reset is needed in any further steps.

- Make a copy of the original file

```
data_vif=data.copy()
```

- Check the vif, Choosing the appropriate variables for the data, so that we could avoid multicollinearity

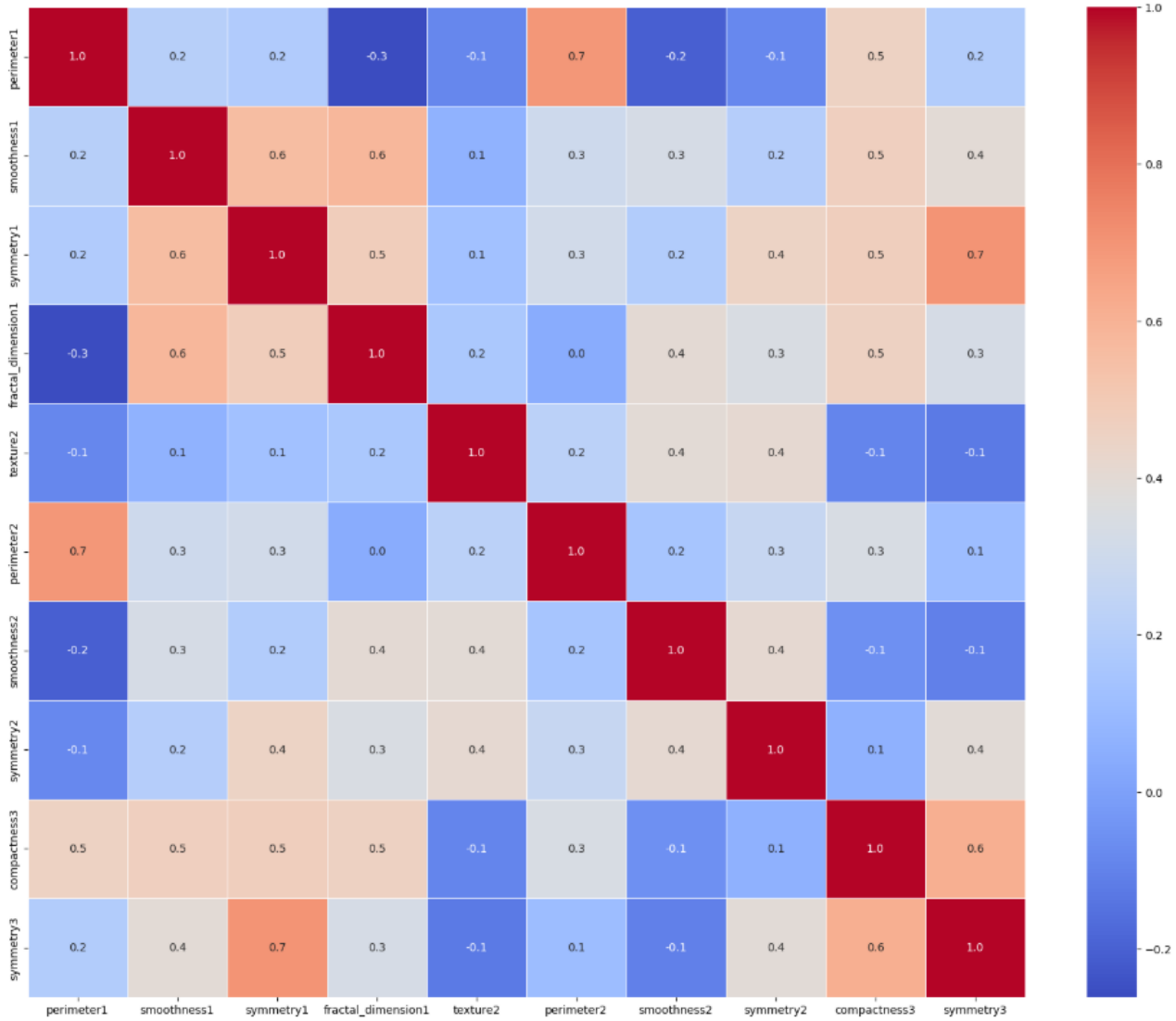
```
data_vif.columns
```

```
Index(['Diagnosis', 'radius1', 'texture1', 'perimeter1', 'area1',
       'smoothness1', 'compactness1', 'concavity1', 'concave_points1',
       'symmetry1', 'fractal_dimension1', 'radius2', 'texture2', 'perimeter2',
       'area2', 'smoothness2', 'compactness2', 'concavity2', 'concave_points2',
       'symmetry2', 'fractal_dimension2', 'radius3', 'texture3', 'perimeter3',
       'area3', 'smoothness3', 'compactness3', 'concavity3', 'concave_points3',
       'symmetry3', 'fractal_dimension3'],
      dtype='object')
```

- Selecting columns

  The values which are dropped will removed from the vif test, and subsequently from the model.

	Feature	VIF
0	const	337.743550
1	Diagnosis	3.240835
2	perimeter1	5.290145
3	smoothness1	2.349636
4	symmetry1	2.995382
5	fractal_dimension1	3.821458
6	texture2	1.537587
7	perimeter2	3.104662
8	smoothness2	1.901272
9	symmetry2	2.597828
10	compactness3	3.927063
11	symmetry3	4.749527



- Storing the data after feature selection, which will be used for model building

```
data_raw=data_vif_pass.copy()
```

```
data_raw
```

	Diagnosis	perimeter1	smoothness1	symmetry1	fractal_dimension1	texture2	perimeter2	smoothness2	symmetry2	compactness3	symmetry3
0	1	122.80	0.11840	0.2419	0.07871	0.9053	8.589	0.006399	0.03003	0.66560	0.4601
1	1	132.90	0.08474	0.1812	0.05667	0.7339	3.398	0.005225	0.01389	0.18660	0.2750
2	1	130.00	0.10960	0.2069	0.05999	0.7869	4.585	0.006150	0.02250	0.42450	0.3613
3	1	77.58	0.14250	0.2597	0.09744	1.1560	3.445	0.009110	0.05963	0.86630	0.6638
4	1	135.10	0.10030	0.1809	0.05883	0.7813	5.438	0.011490	0.01756	0.20500	0.2364
...	...	...	...	...	...	...	...	...	...	...	...
564	1	142.00	0.11100	0.1726	0.05623	1.2560	7.673	0.010300	0.01114	0.21130	0.2060
565	1	131.20	0.09780	0.1752	0.05533	2.4630	5.203	0.005769	0.01898	0.19220	0.2572
566	1	108.30	0.08455	0.1590	0.05648	1.0750	3.425	0.005903	0.01318	0.30940	0.2218
567	1	140.10	0.11780	0.2397	0.07016	1.5950	5.772	0.006522	0.02324	0.86810	0.4087
568	0	47.92	0.05263	0.1587	0.05884	1.4280	2.548	0.007189	0.02676	0.06444	0.2871

569 rows × 11 columns

Creating four versions of the code so that we could compare the 4 versions

(outlier smoothed and raw) x (with or without smote)

The smote version here is a data copy for the smote activity, it won't be directly used, a new variable will be defined after applying smote and that dataframe will be passed to build the model.

Similarly for the smoothed data.

Smoothing the raw data

```
perimeter1 changed
smoothness1 changed
symmetry1 changed
fractal_dimension1 changed
texture2 changed
perimeter2 changed
smoothness2 changed
symmetry2 changed
compactness3 changed
symmetry3 changed
```

Checking the number of outliers of the smoothed data

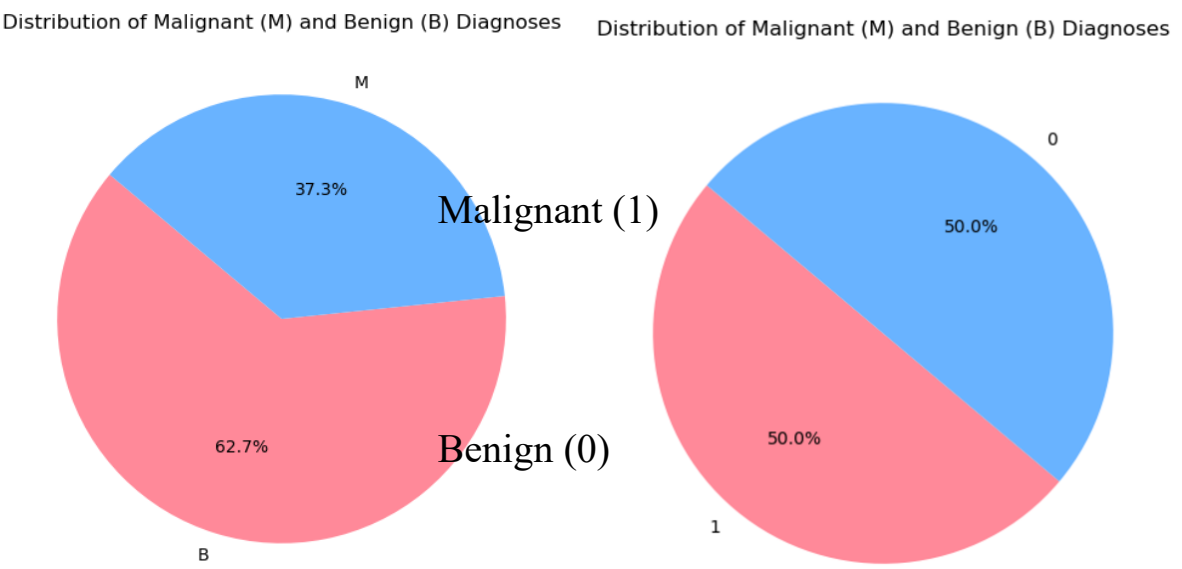
	count	mean	std	min	25%	50%	75%	max
Diagnosis	569.0	0.372583	0.483918	0.000000	0.000000	0.00000	1.000000	1.00000
perimeter1	569.0	90.144288	21.405756	43.790000	75.170000	86.24000	102.400000	147.30000
smoothness1	569.0	0.096000	0.013126	0.062510	0.086410	0.09587	0.104900	0.13350
symmetry1	569.0	0.179200	0.023715	0.116700	0.162000	0.17920	0.194300	0.24590
fractal_dimension1	569.0	0.062151	0.005842	0.049960	0.057700	0.06154	0.065690	0.07871
texture2	569.0	1.151444	0.424286	0.360200	0.833900	1.10800	1.410000	2.42600
perimeter2	569.0	2.452209	1.094063	0.757000	1.606000	2.28700	2.974000	5.86500
smoothness2	569.0	0.006547	0.001994	0.001713	0.005169	0.00638	0.007702	0.01243
symmetry2	569.0	0.019229	0.005529	0.007882	0.015160	0.01873	0.022030	0.03546
compactness3	569.0	0.238212	0.127693	0.027290	0.147200	0.21190	0.315000	0.62470
symmetry3	569.0	0.282080	0.046602	0.156500	0.250400	0.28220	0.310900	0.41540

```
Number of outliers for 'perimeter1': 4
Number of outliers for 'smoothness1': 2
Number of outliers for 'symmetry1': 2
Number of outliers for 'fractal_dimension1': 7
Number of outliers for 'texture2': 5
Number of outliers for 'perimeter2': 21
Number of outliers for 'smoothness2': 9
Number of outliers for 'symmetry2': 14
Number of outliers for 'compactness3': 12
Number of outliers for 'symmetry3': 11
```

Using Smote for unbalanced data

SMOTE - Synthetic Minority Over-sampling Technique

For smote to be implemented, it's better not to go to the function we will be defining later for handling smote data as the data splitting is happening inside the function, so implementing it here, so we are doing the smote first and then going to the function.



Before Smote **After Smote**

Creating **two versions** after smote so that

we can smooth one version of it and

keep the other in raw smote form.

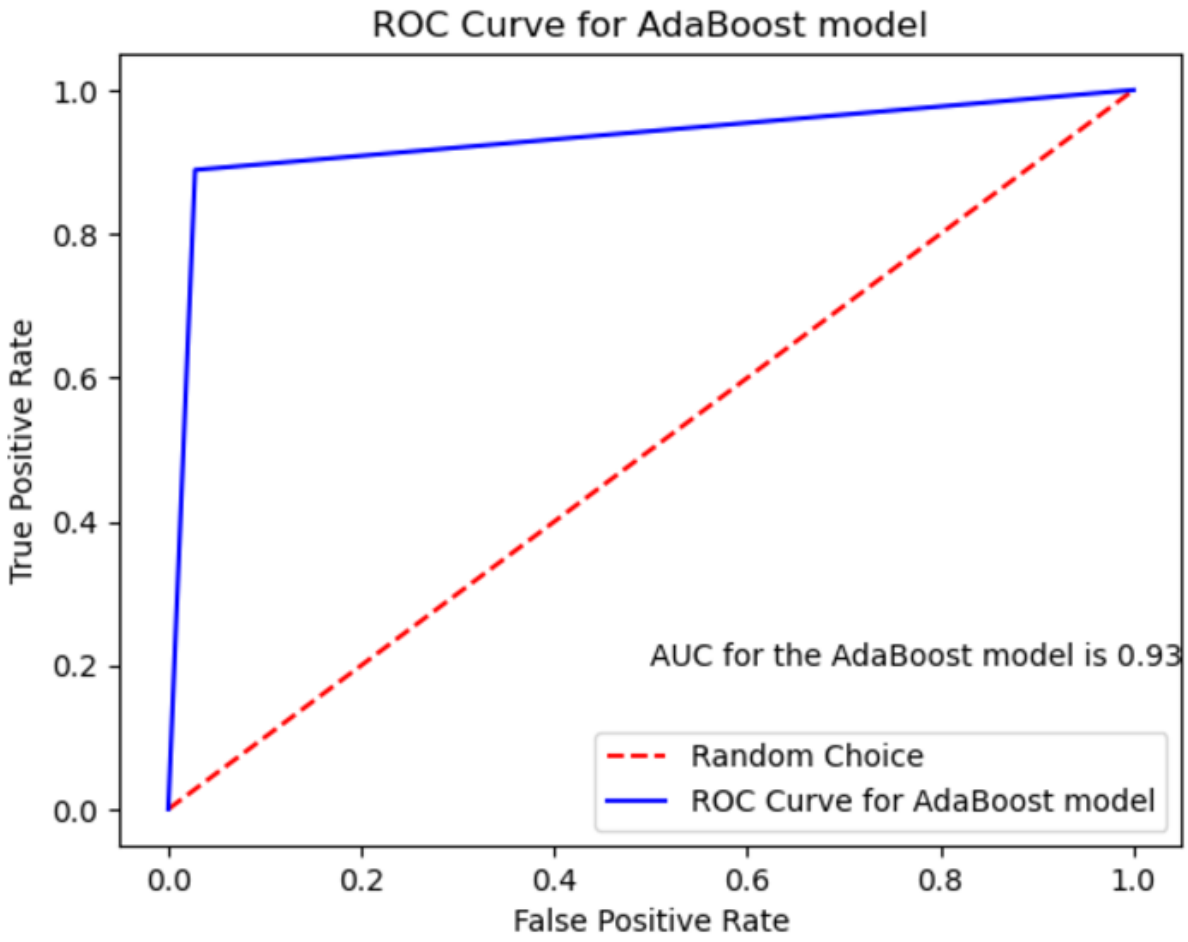
```
perimeter1 changed
smoothness1 changed
symmetry1 changed
fractal_dimension1 changed
texture2 changed
perimeter2 changed
smoothness2 changed
symmetry2 changed
compactness3 changed
symmetry3 changed
```

Creating a function to get the analysis, scaling and the other process in a single go, it will help us to compare multiple versions of the data with multiplier classifier and will help to choose the right one by reducing manual iterations of code.

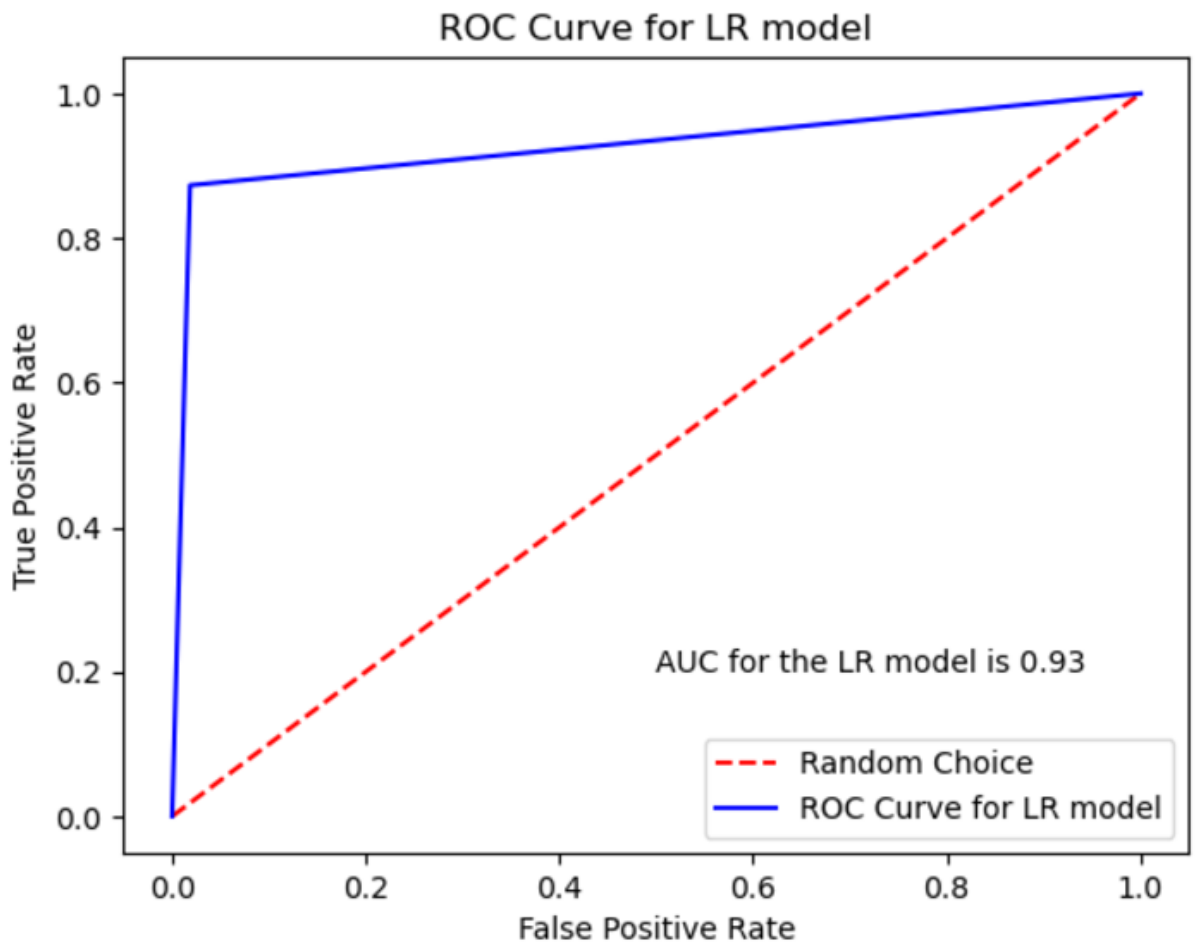
inside the function, data splitting, data scaling, model fitting, and evaluation will be done, and the evaluation report will be done.

8. Model Building and Evaluation

Part 1: Unsmoothed data

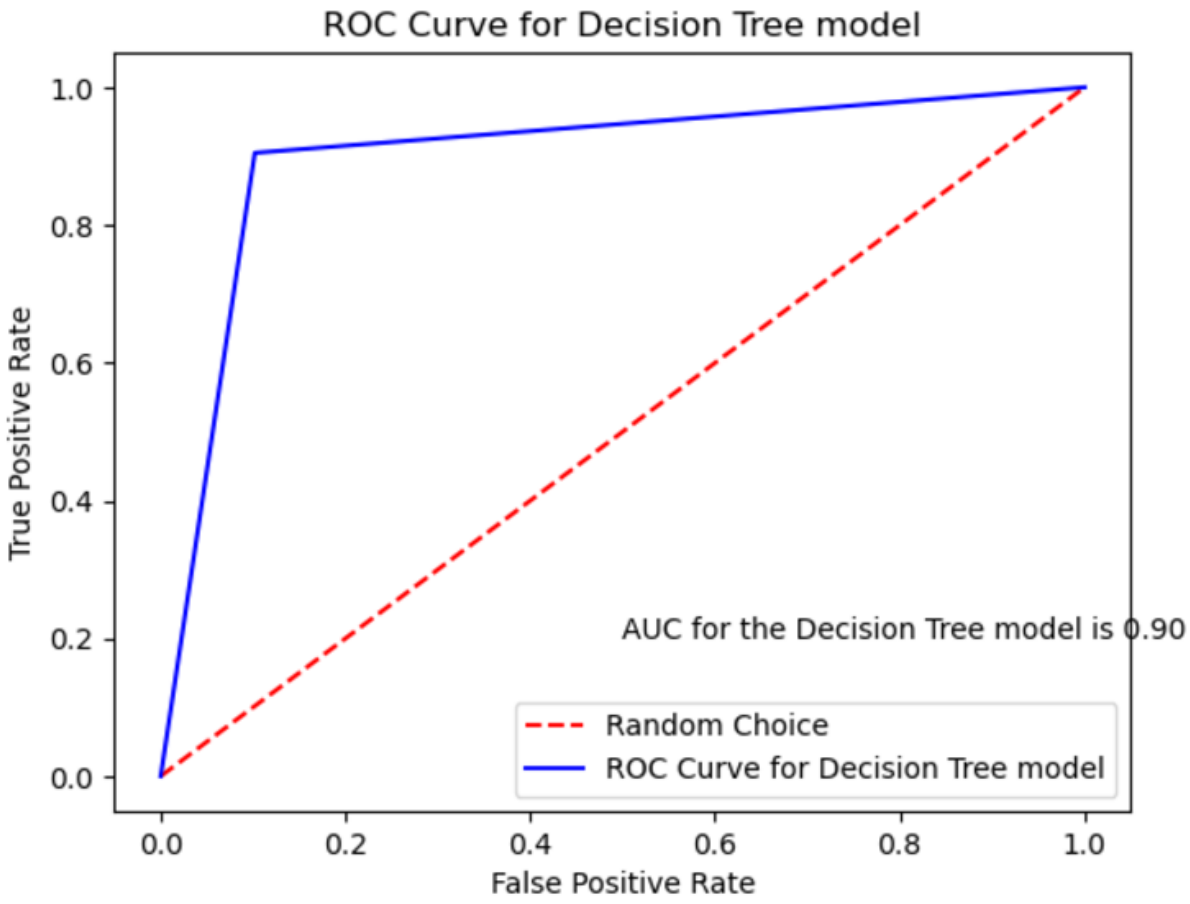



```
Training data shape: (398, 10)
Testing data shape: (171, 10)
The classification by AdaBoost model is as follows
              precision    recall  f1-score   support

           0       0.94      0.97      0.95       108
           1       0.95      0.89      0.92        63

    accuracy                           0.94       171
   macro avg       0.94      0.93      0.94       171
weighted avg       0.94      0.94      0.94       171
```



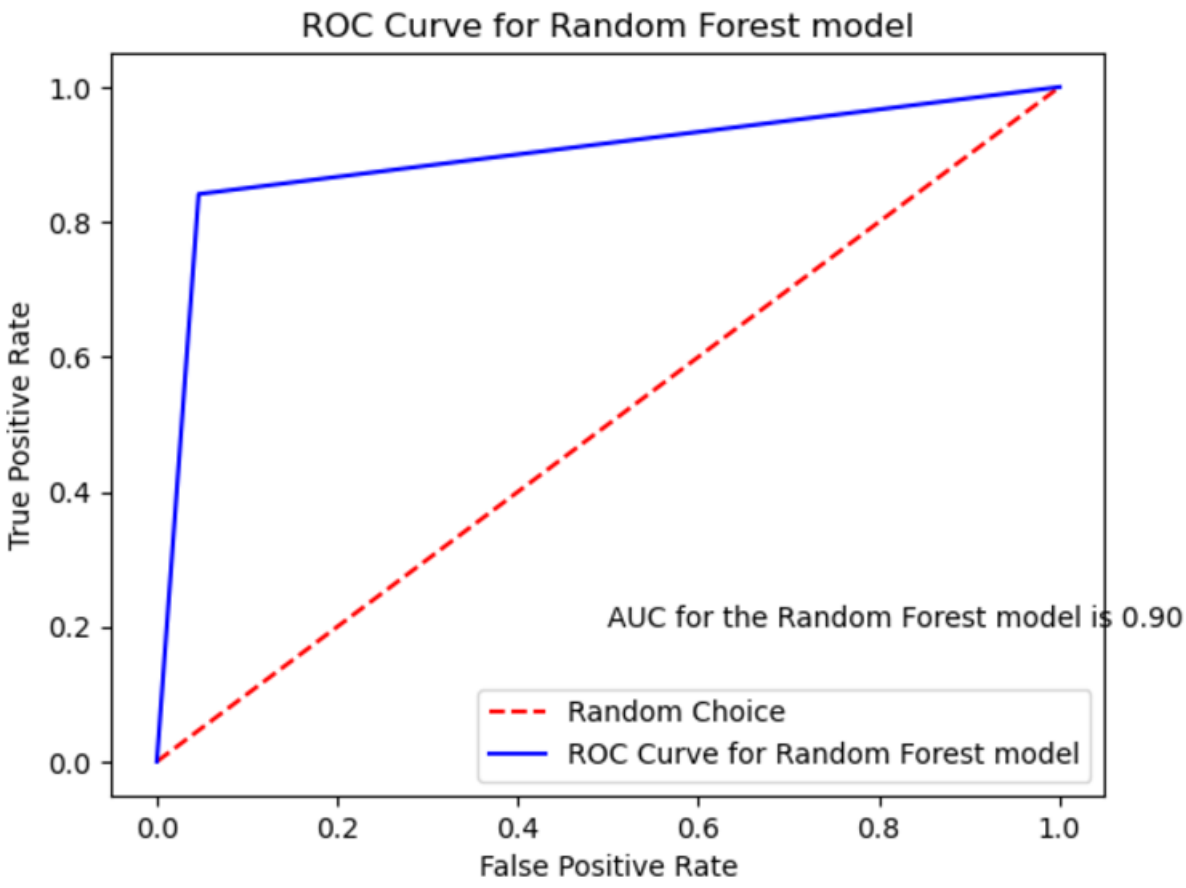
The classification by LR model is as follows

	precision	recall	f1-score	support
0	0.93	0.98	0.95	108
1	0.96	0.87	0.92	63
accuracy			0.94	171
macro avg	0.95	0.93	0.94	171
weighted avg	0.94	0.94	0.94	171



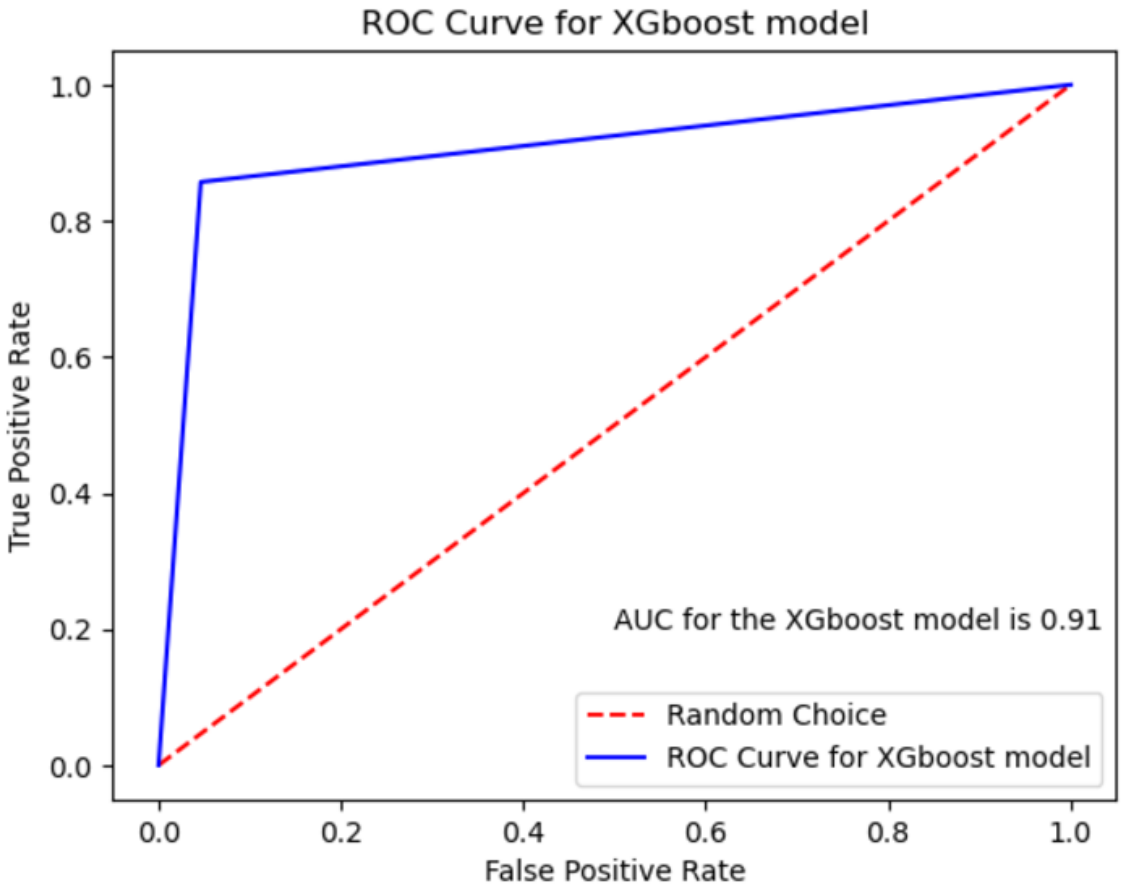
The classification by Decision Tree model is as follows

	precision	recall	f1-score	support
0	0.94	0.90	0.92	108
1	0.84	0.90	0.87	63
accuracy			0.90	171
macro avg	0.89	0.90	0.89	171
weighted avg	0.90	0.90	0.90	171



The classification by Random Forest model is as follows

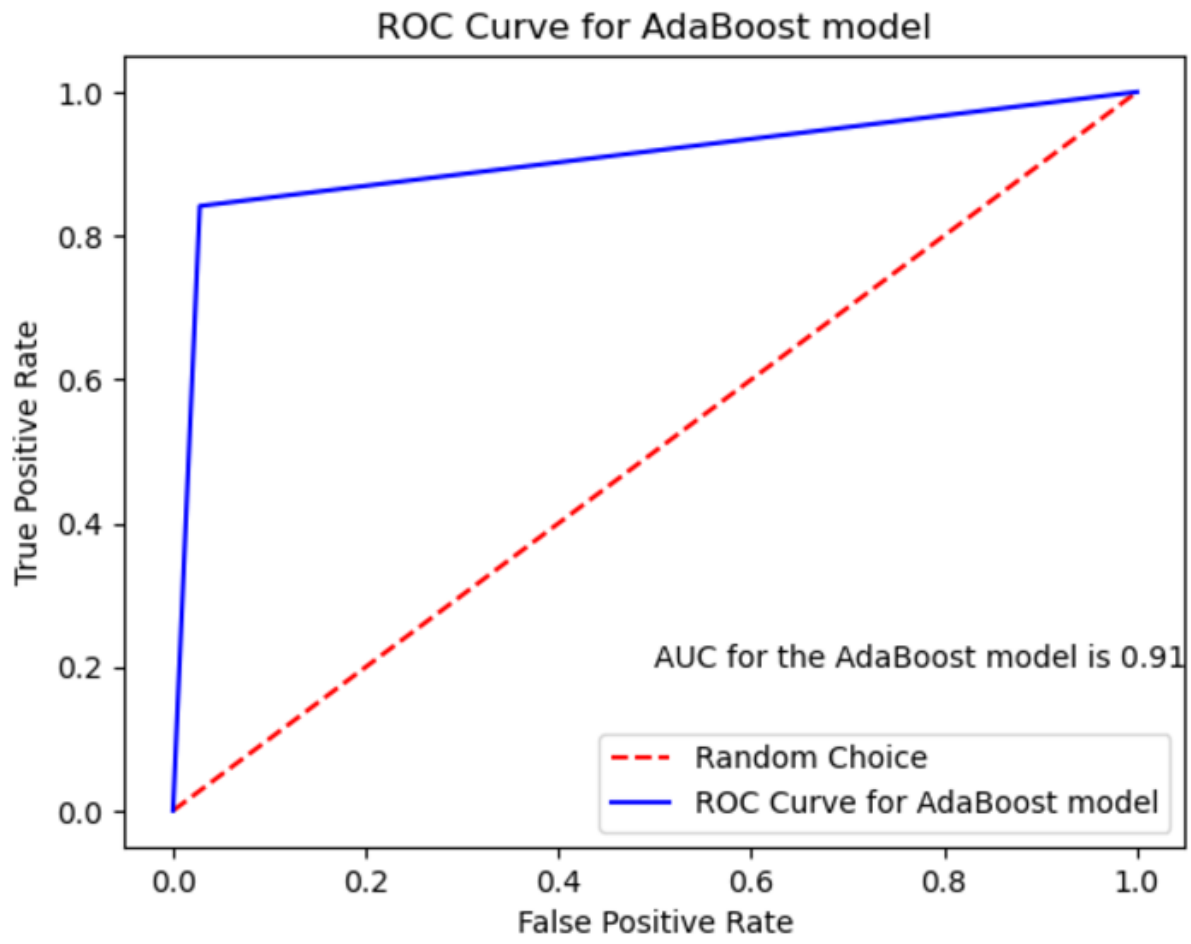
	precision	recall	f1-score	support
0	0.91	0.95	0.93	108
1	0.91	0.84	0.88	63
accuracy			0.91	171
macro avg	0.91	0.90	0.90	171
weighted avg	0.91	0.91	0.91	171



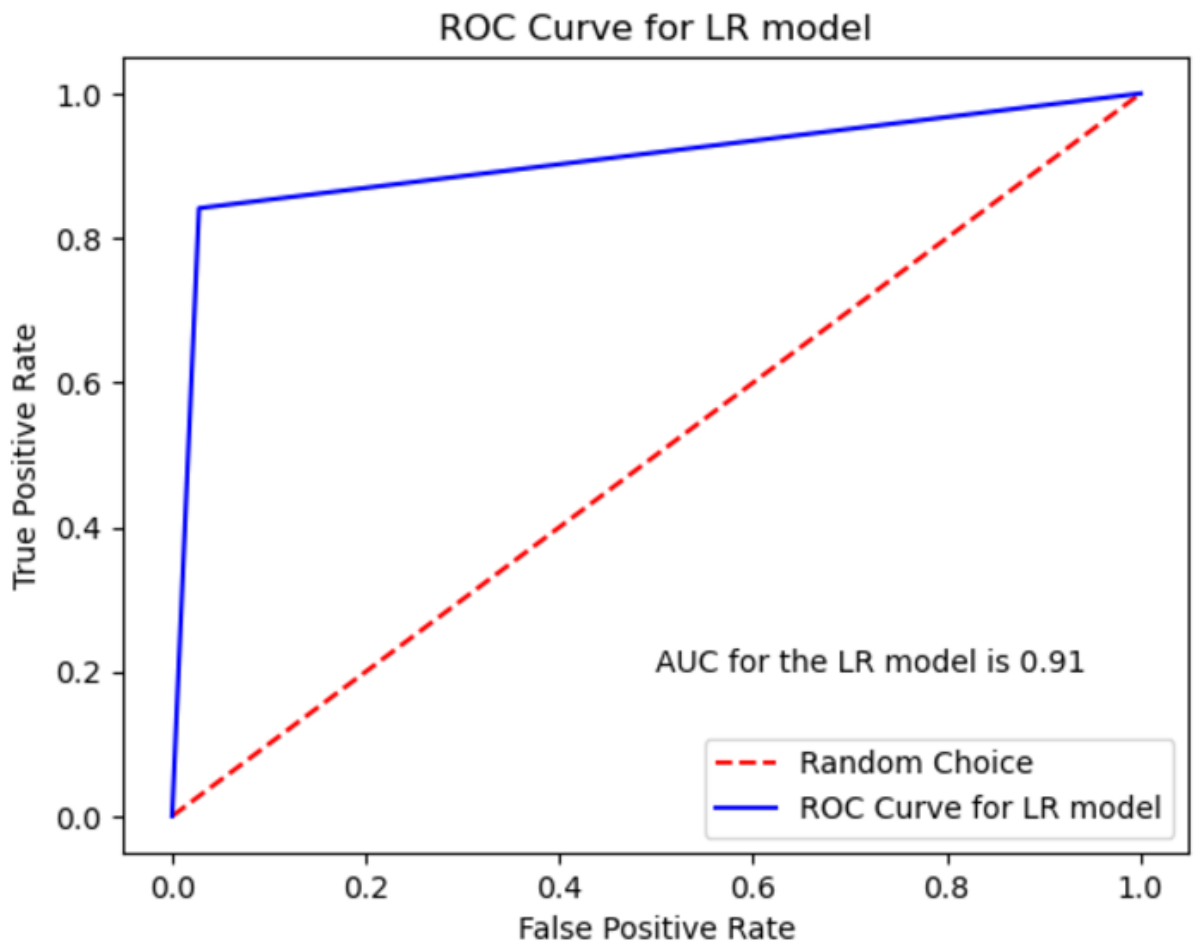
The classification by XGboost model is as follows

	precision	recall	f1-score	support
0	0.92	0.95	0.94	108
1	0.92	0.86	0.89	63
accuracy			0.92	171
macro avg	0.92	0.91	0.91	171
weighted avg	0.92	0.92	0.92	171

Part 2: Smoothed Data

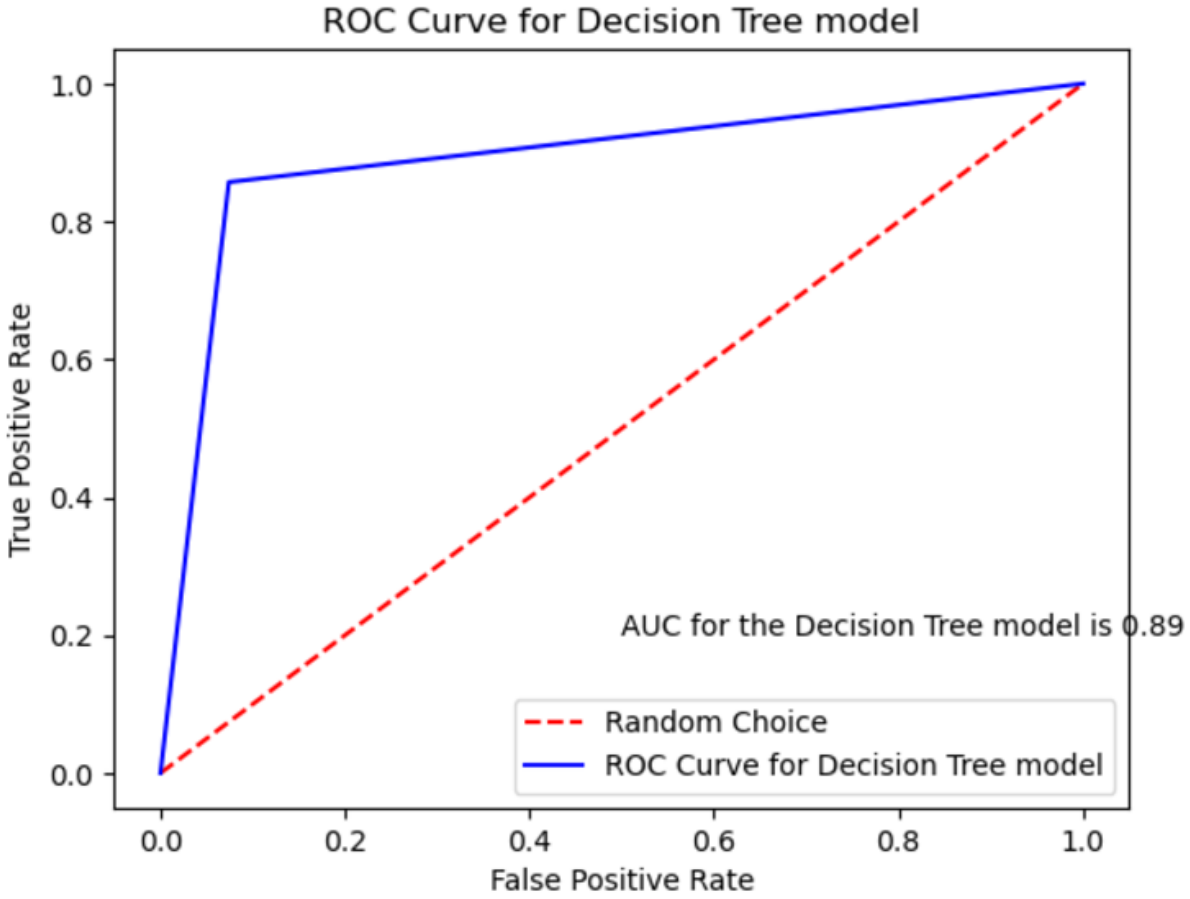



```
Training data shape: (398, 10)
Testing data shape: (171, 10)
The classification by AdaBoost model is as follows
              precision    recall  f1-score   support

           0       0.91      0.97      0.94       108
           1       0.95      0.84      0.89        63

    accuracy                           0.92       171
   macro avg       0.93      0.91      0.92       171
weighted avg       0.93      0.92      0.92       171
```



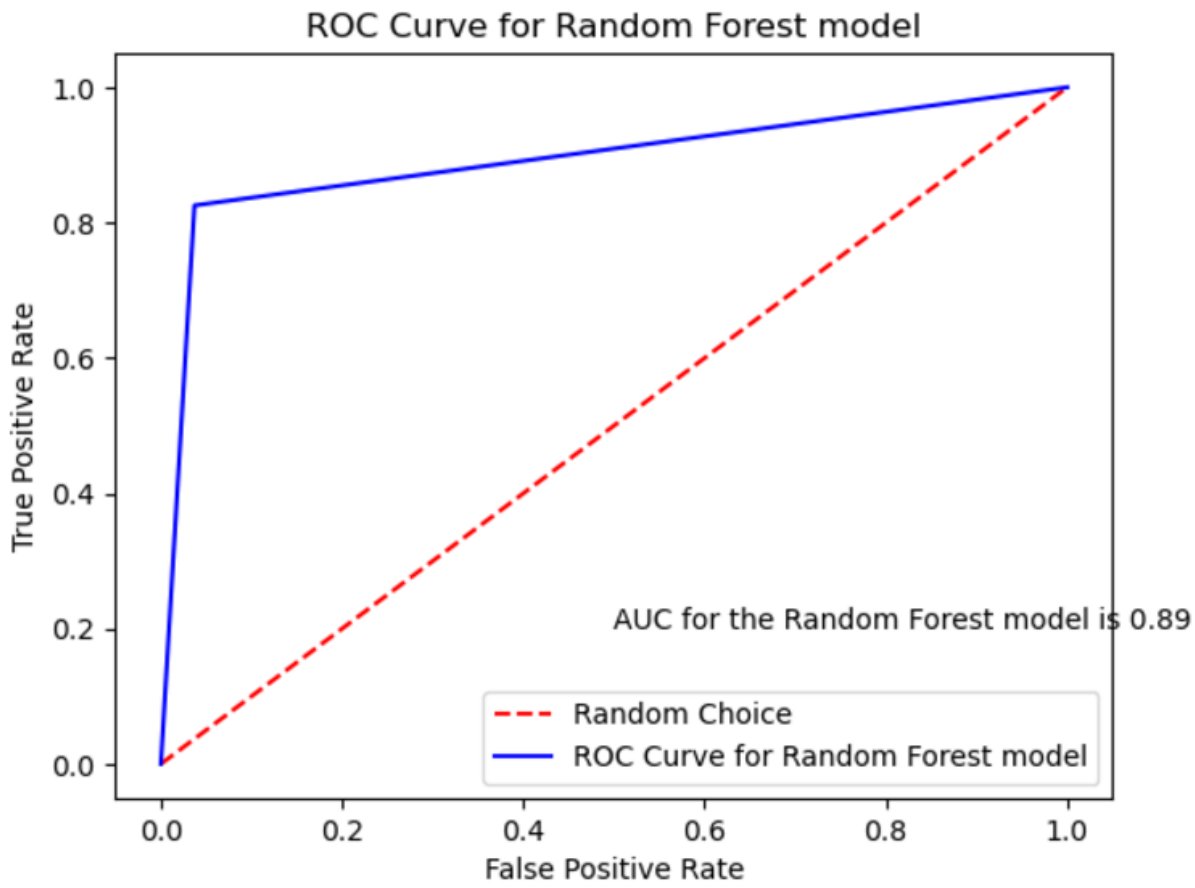
The classification by LR model is as follows

	precision	recall	f1-score	support
0	0.91	0.97	0.94	108
1	0.95	0.84	0.89	63
accuracy			0.92	171
macro avg	0.93	0.91	0.92	171
weighted avg	0.93	0.92	0.92	171



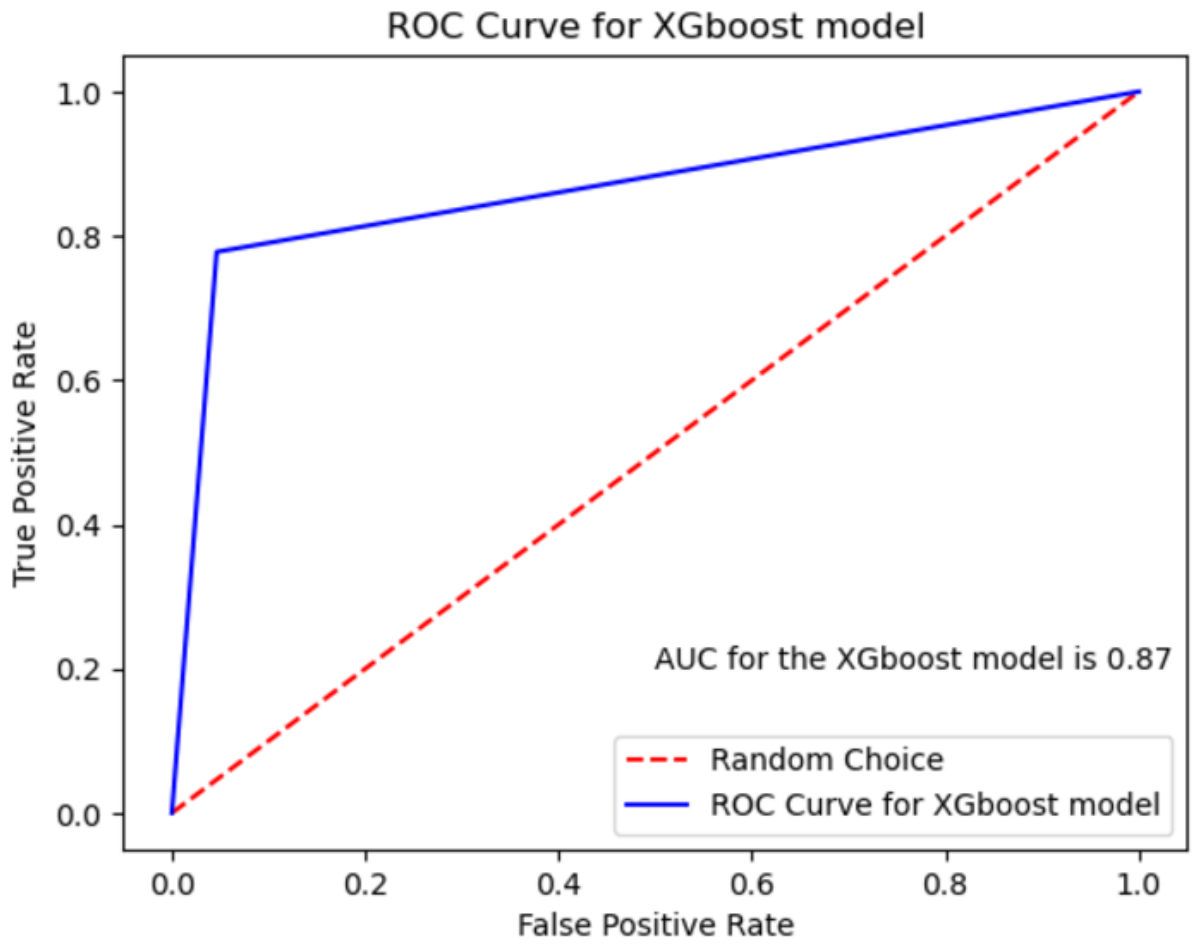
The classification by Decision Tree model is as follows

	precision	recall	f1-score	support
0	0.92	0.93	0.92	108
1	0.87	0.86	0.86	63
accuracy			0.90	171
macro avg	0.89	0.89	0.89	171
weighted avg	0.90	0.90	0.90	171



The classification by Random Forest model is as follows

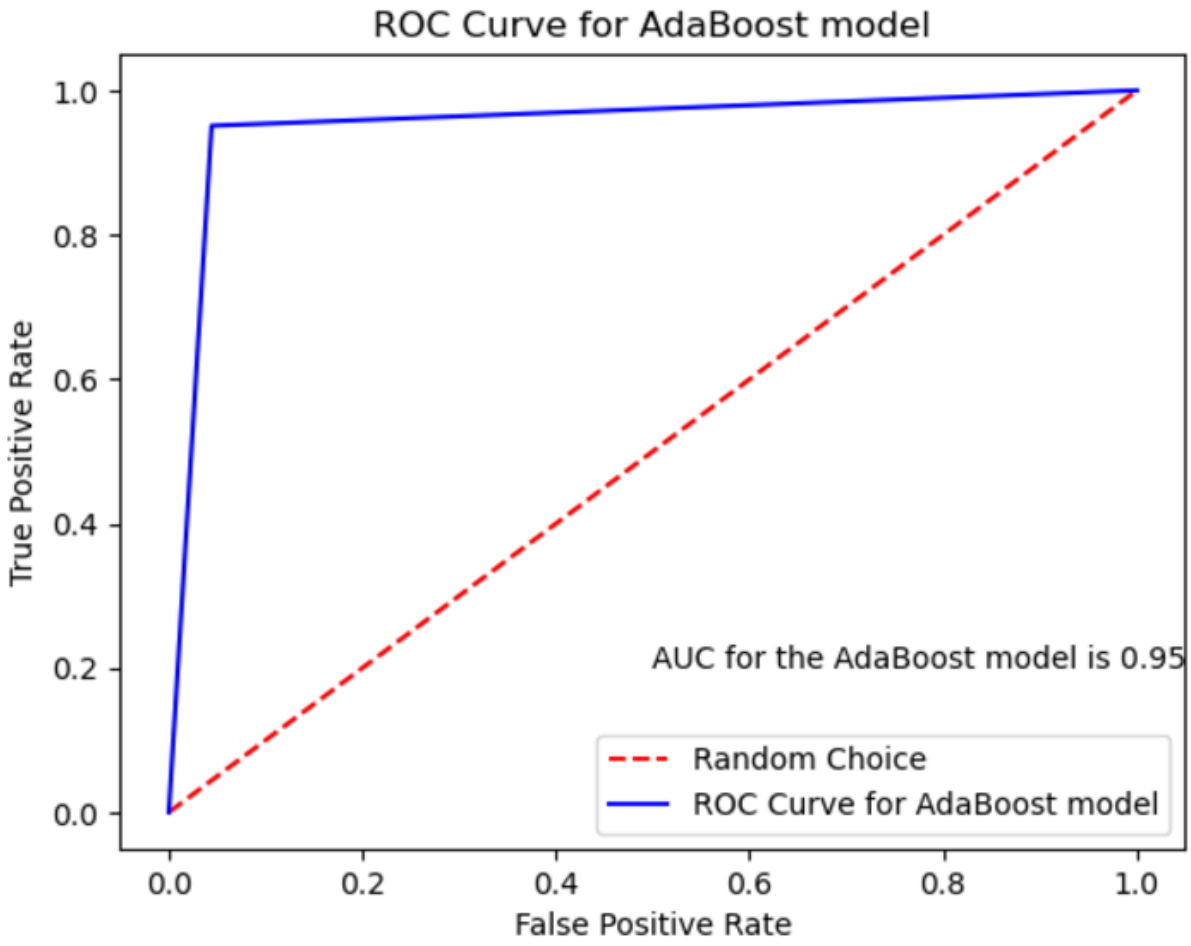
	precision	recall	f1-score	support
0	0.90	0.96	0.93	108
1	0.93	0.83	0.87	63
accuracy			0.91	171
macro avg	0.92	0.89	0.90	171
weighted avg	0.91	0.91	0.91	171



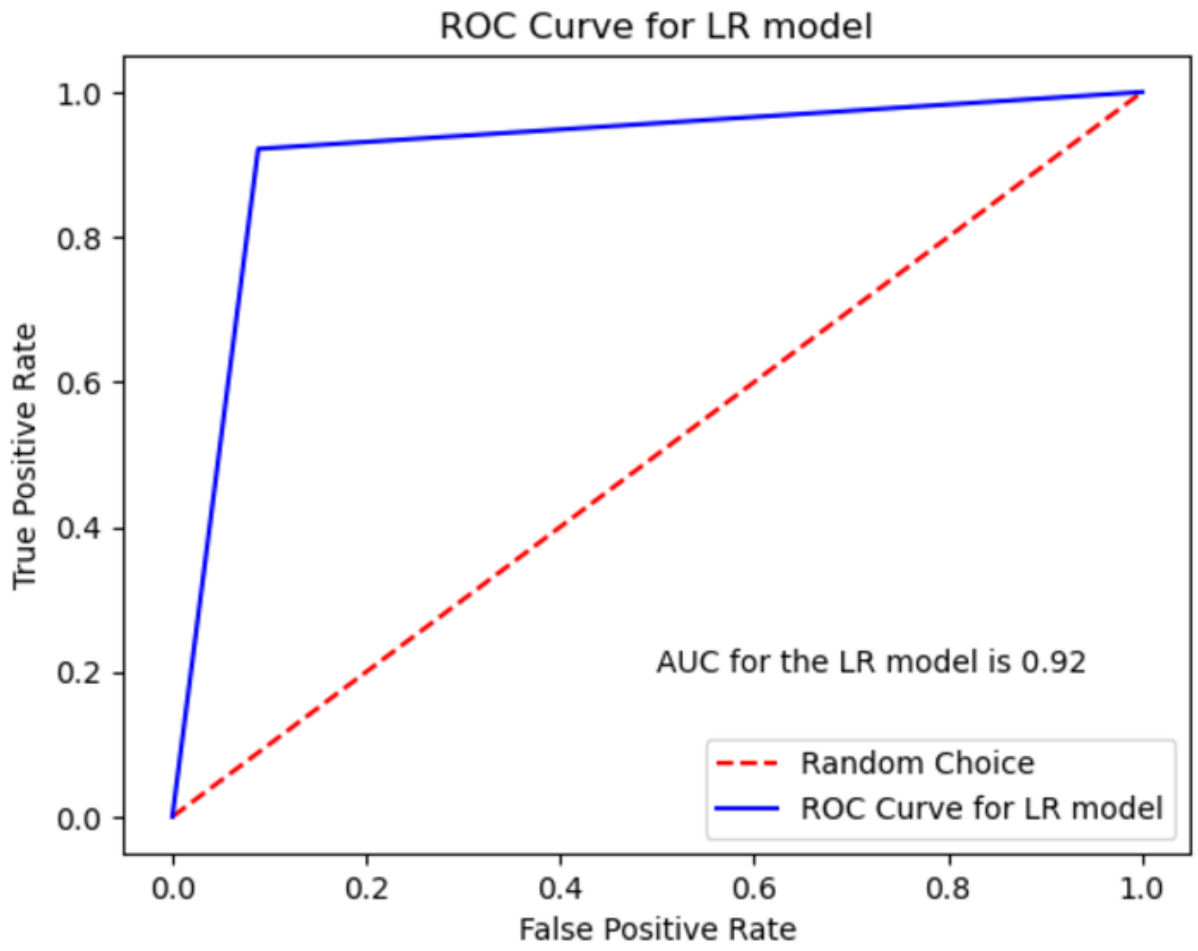
The classification by XGboost model is as follows

	precision	recall	f1-score	support
0	0.88	0.95	0.92	108
1	0.91	0.78	0.84	63
accuracy			0.89	171
macro avg	0.89	0.87	0.88	171
weighted avg	0.89	0.89	0.89	171

Part 3: Smote Unsmoothed

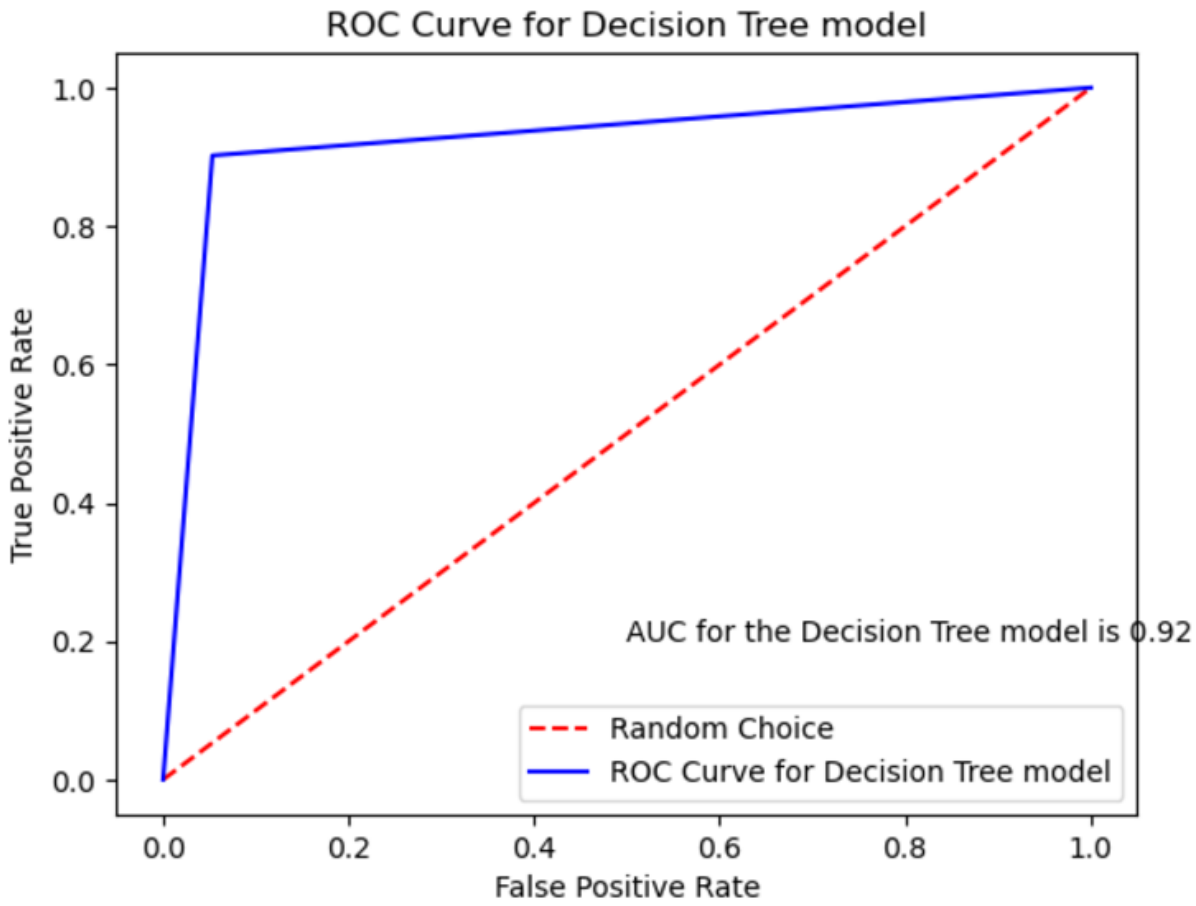



```
Training data shape: (499, 10)
Testing data shape: (215, 10)
The classification by AdaBoost model is as follows
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       113
           1       0.95      0.95      0.95       102

    accuracy                           0.95       215
   macro avg       0.95      0.95      0.95       215
weighted avg       0.95      0.95      0.95       215
```



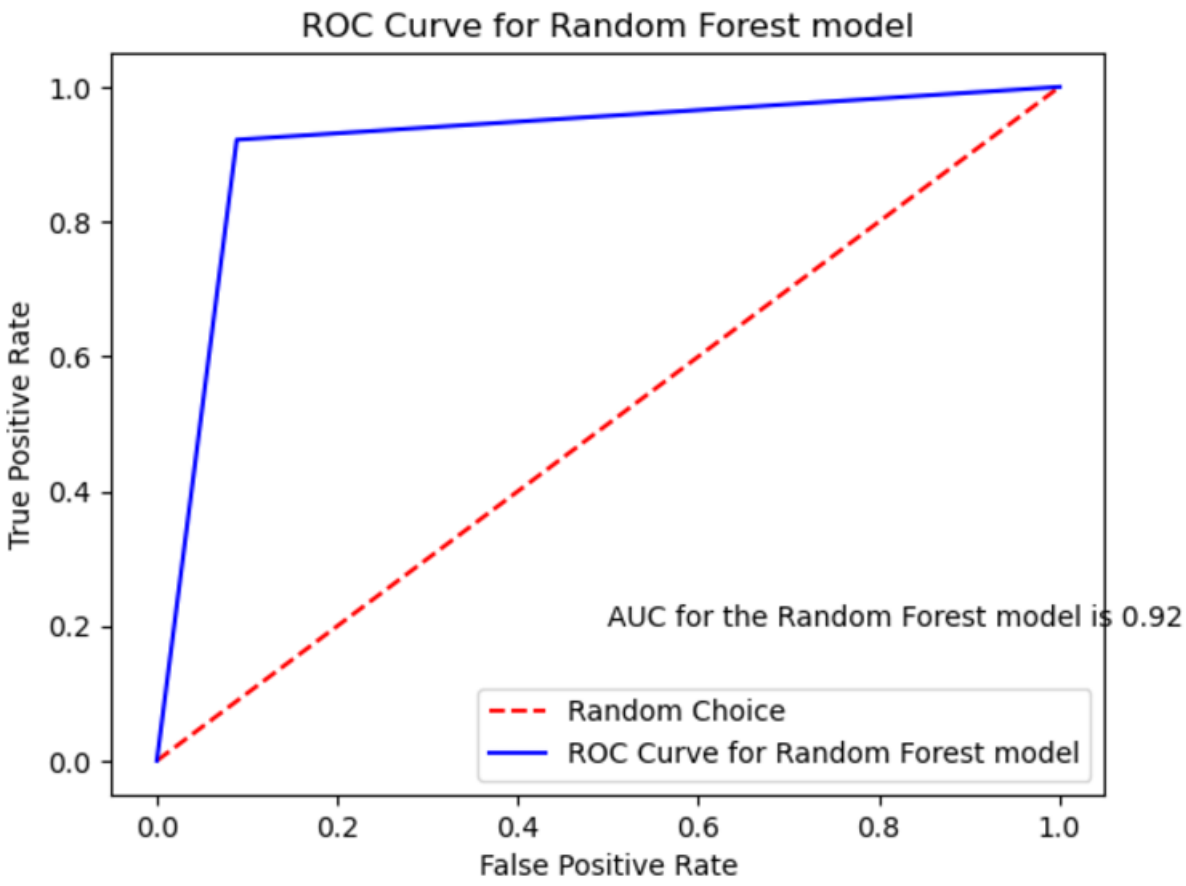
The classification by LR model is as follows

	precision	recall	f1-score	support
0	0.93	0.91	0.92	113
1	0.90	0.92	0.91	102
accuracy			0.92	215
macro avg	0.92	0.92	0.92	215
weighted avg	0.92	0.92	0.92	215



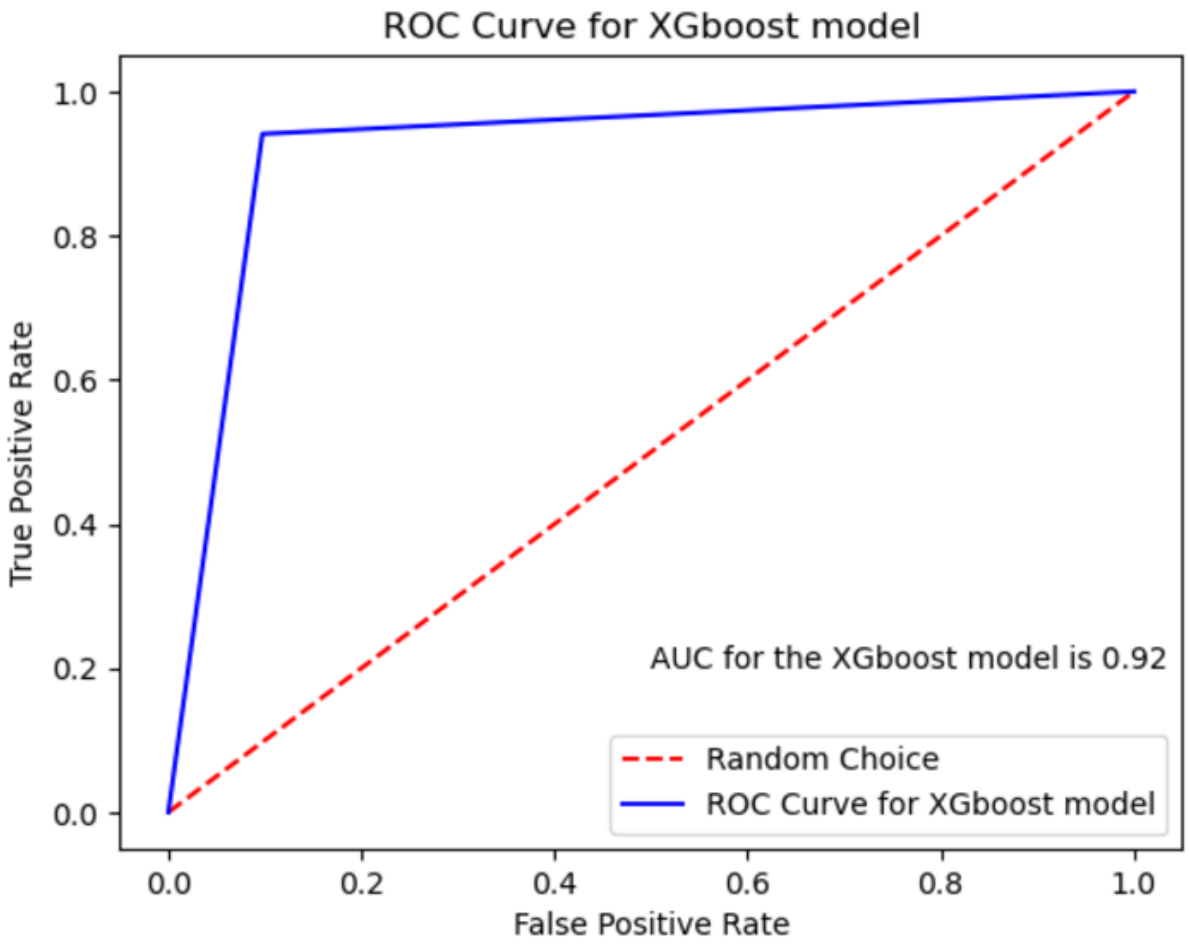
The classification by Decision Tree model is as follows

	precision	recall	f1-score	support
0	0.91	0.95	0.93	113
1	0.94	0.90	0.92	102
accuracy			0.93	215
macro avg	0.93	0.92	0.93	215
weighted avg	0.93	0.93	0.93	215



The classification by Random Forest model is as follows

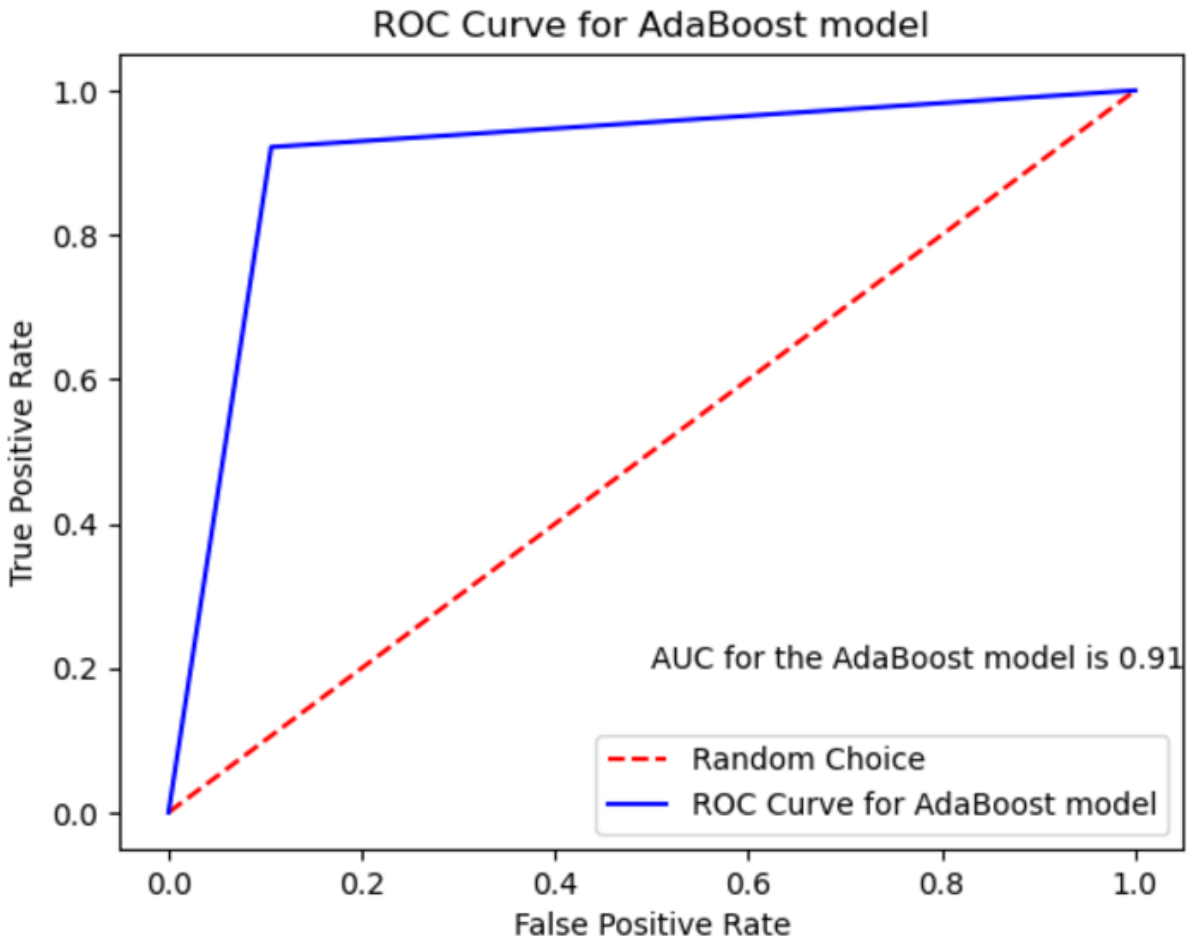
	precision	recall	f1-score	support
0	0.93	0.91	0.92	113
1	0.90	0.92	0.91	102
accuracy			0.92	215
macro avg	0.92	0.92	0.92	215
weighted avg	0.92	0.92	0.92	215



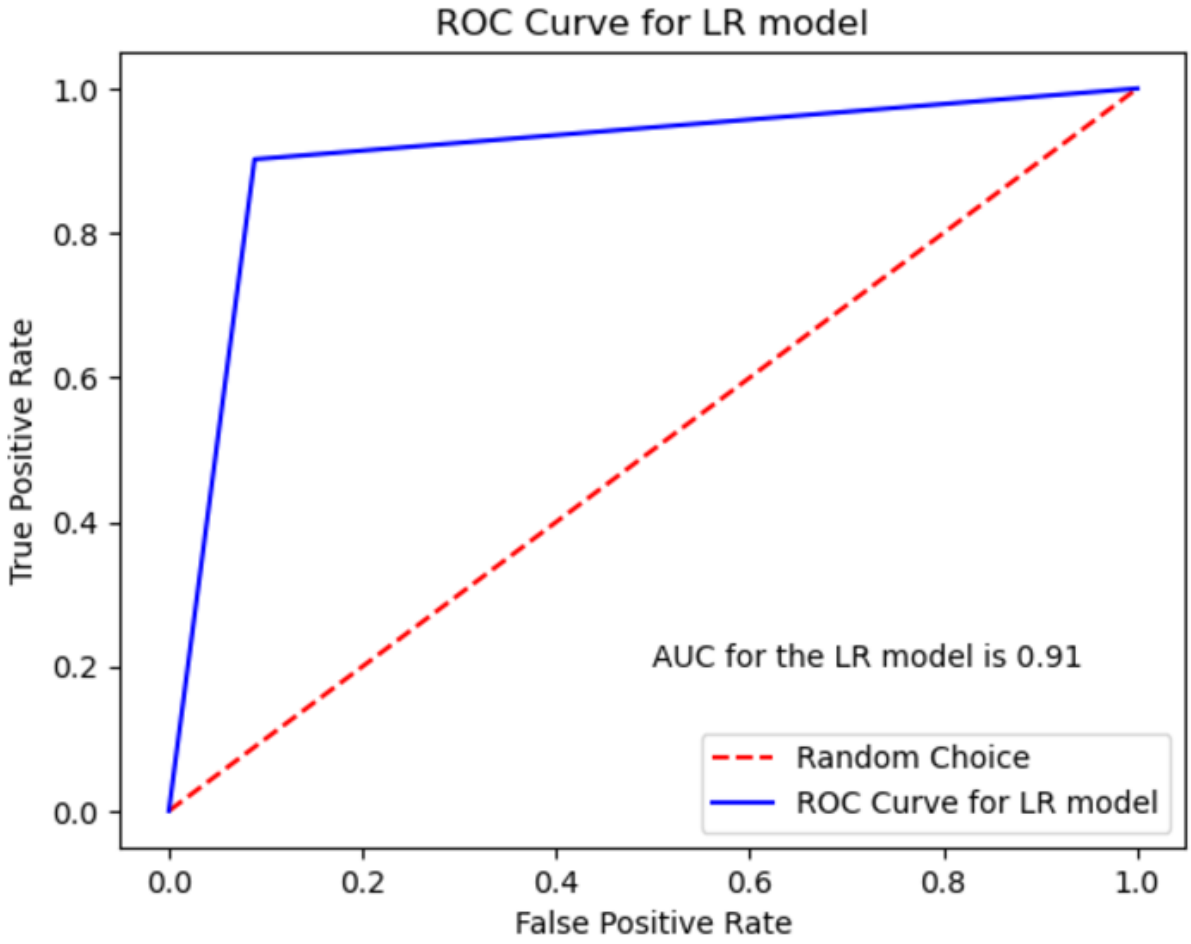
The classification by XGboost model is as follows

	precision	recall	f1-score	support
0	0.94	0.90	0.92	113
1	0.90	0.94	0.92	102
accuracy			0.92	215
macro avg	0.92	0.92	0.92	215
weighted avg	0.92	0.92	0.92	215

Part 4: Smote Smoothed

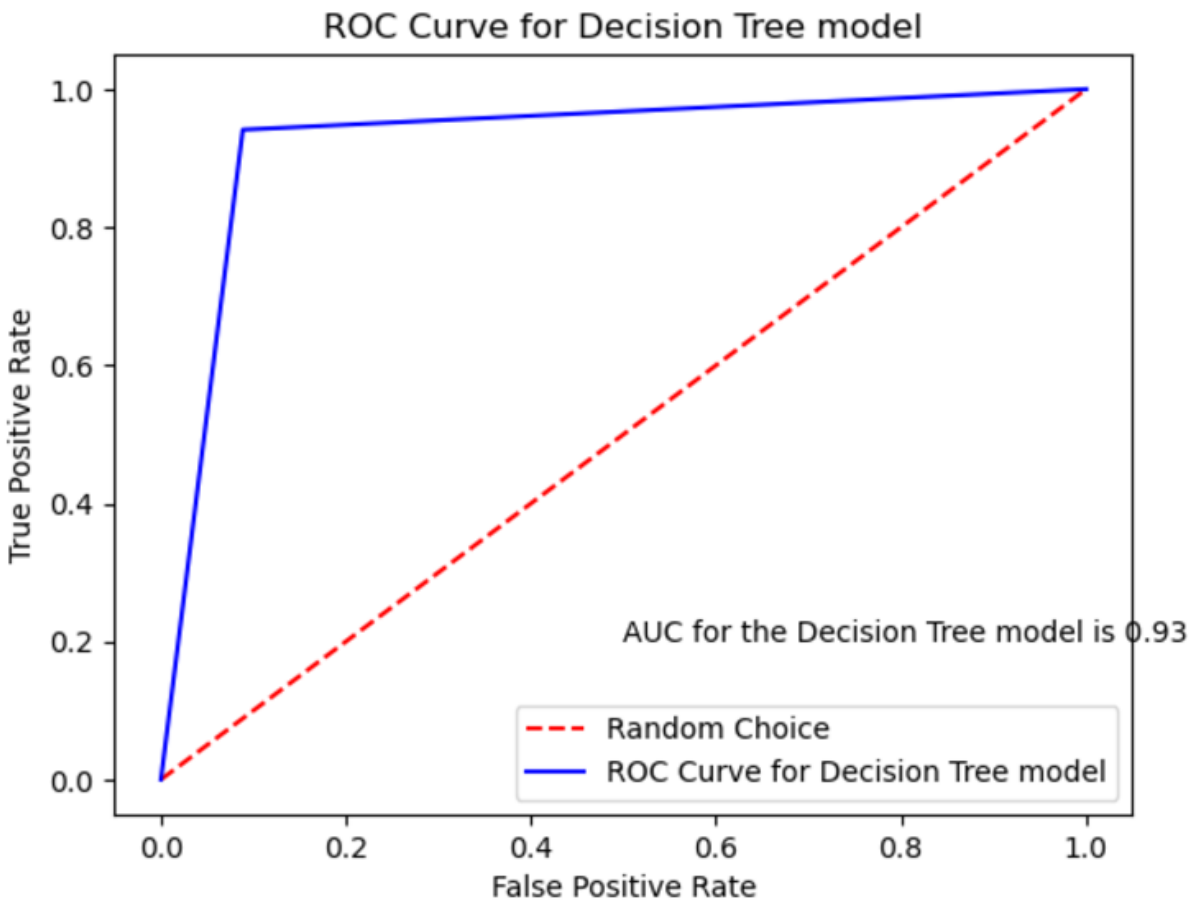



```
Training data shape: (499, 10)
Testing data shape: (215, 10)
The classification by AdaBoost model is as follows
              precision    recall  f1-score   support

           0       0.93      0.89      0.91       113
           1       0.89      0.92      0.90       102

    accuracy                           0.91       215
   macro avg       0.91      0.91      0.91       215
weighted avg       0.91      0.91      0.91       215
```



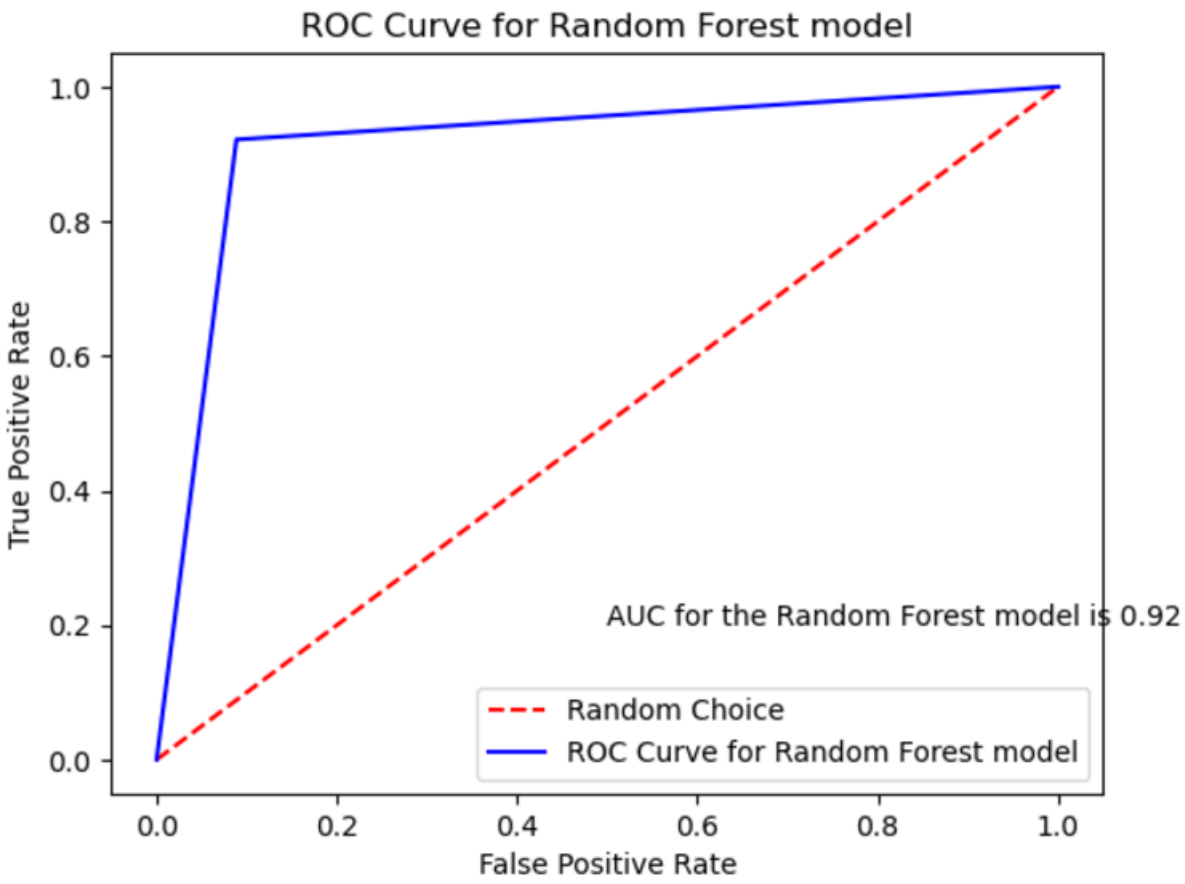
The classification by LR model is as follows

	precision	recall	f1-score	support
0	0.91	0.91	0.91	113
1	0.90	0.90	0.90	102
accuracy			0.91	215
macro avg	0.91	0.91	0.91	215
weighted avg	0.91	0.91	0.91	215



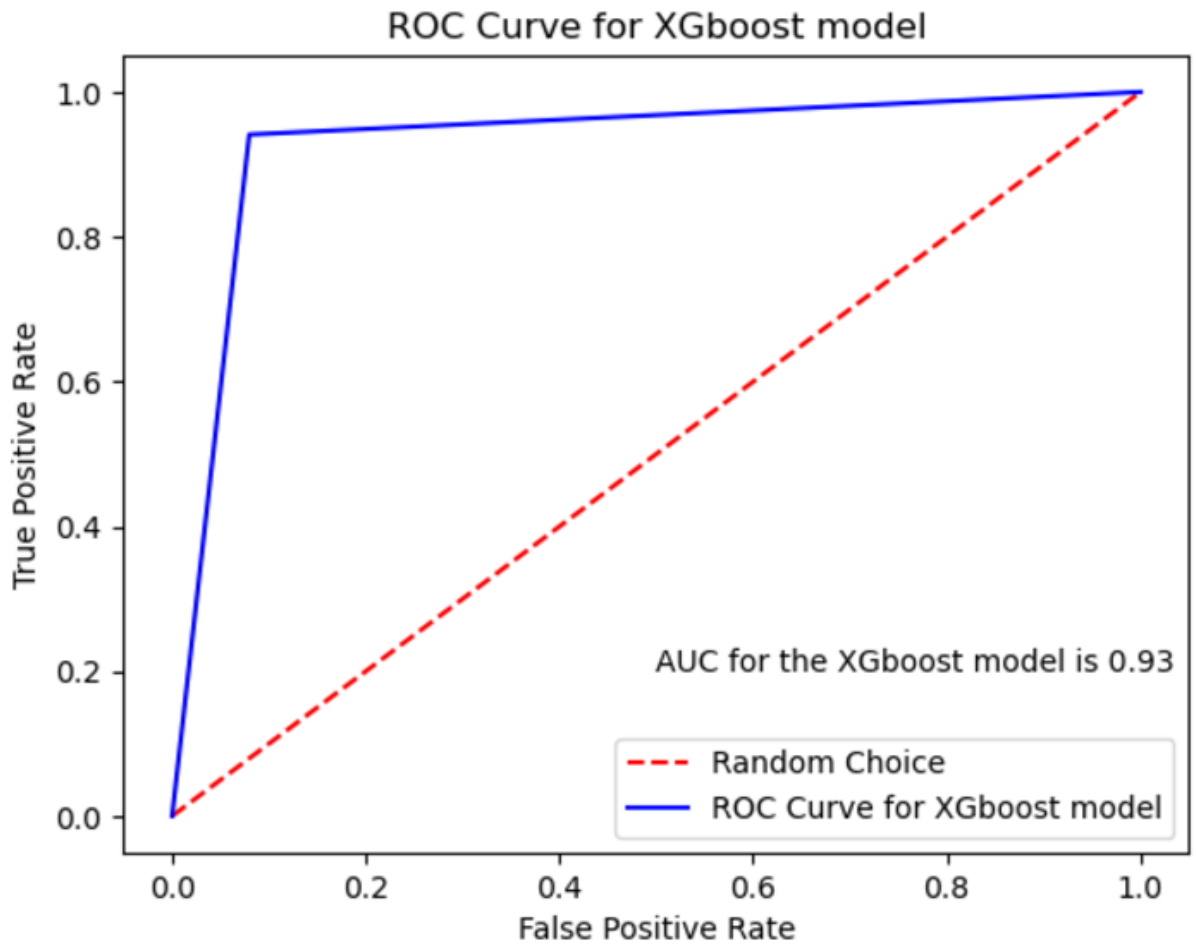
The classification by Decision Tree model is as follows

	precision	recall	f1-score	support
0	0.94	0.91	0.93	113
1	0.91	0.94	0.92	102
accuracy			0.93	215
macro avg	0.93	0.93	0.93	215
weighted avg	0.93	0.93	0.93	215



The classification by Random Forest model is as follows

	precision	recall	f1-score	support
0	0.93	0.91	0.92	113
1	0.90	0.92	0.91	102
accuracy			0.92	215
macro avg	0.92	0.92	0.92	215
weighted avg	0.92	0.92	0.92	215



The classification by XGboost model is as follows

	precision	recall	f1-score	support
0	0.95	0.92	0.93	113
1	0.91	0.94	0.93	102
accuracy			0.93	215
macro avg	0.93	0.93	0.93	215
weighted avg	0.93	0.93	0.93	215

Analysing the two versions, we can clearly infer that the best result is getting from AdaBoost using smote data.

Now doing the hyperparameter tuning using GridSearchCV.

Here we need a function to implement AdaBoost only here, therefore the other models can be discarded, so defining it as a new function which could generate the reports of adaboost as previous, its important to note that we are not using the function for GridSearchCV.

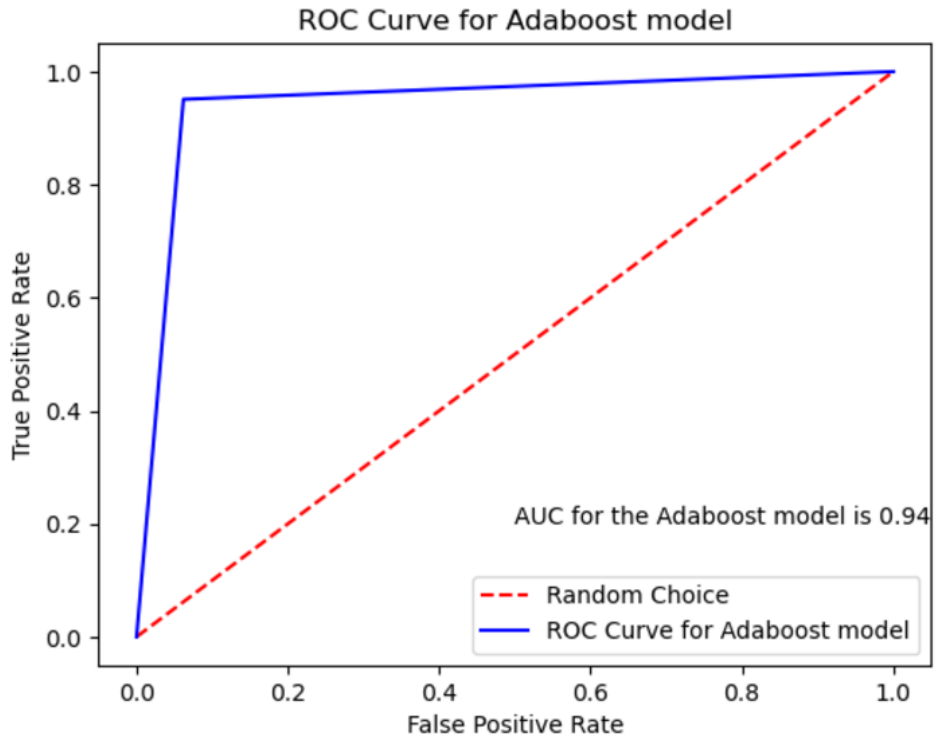
- Implementing GridSearchCV

```
Best Hyperparameters: {'learning_rate': 1.1, 'n_estimators': 500}
```

- Initialize AdaBoostClassifier with the best hyperparameters

```
adamodel(data_smt,1.1,500,42)
```

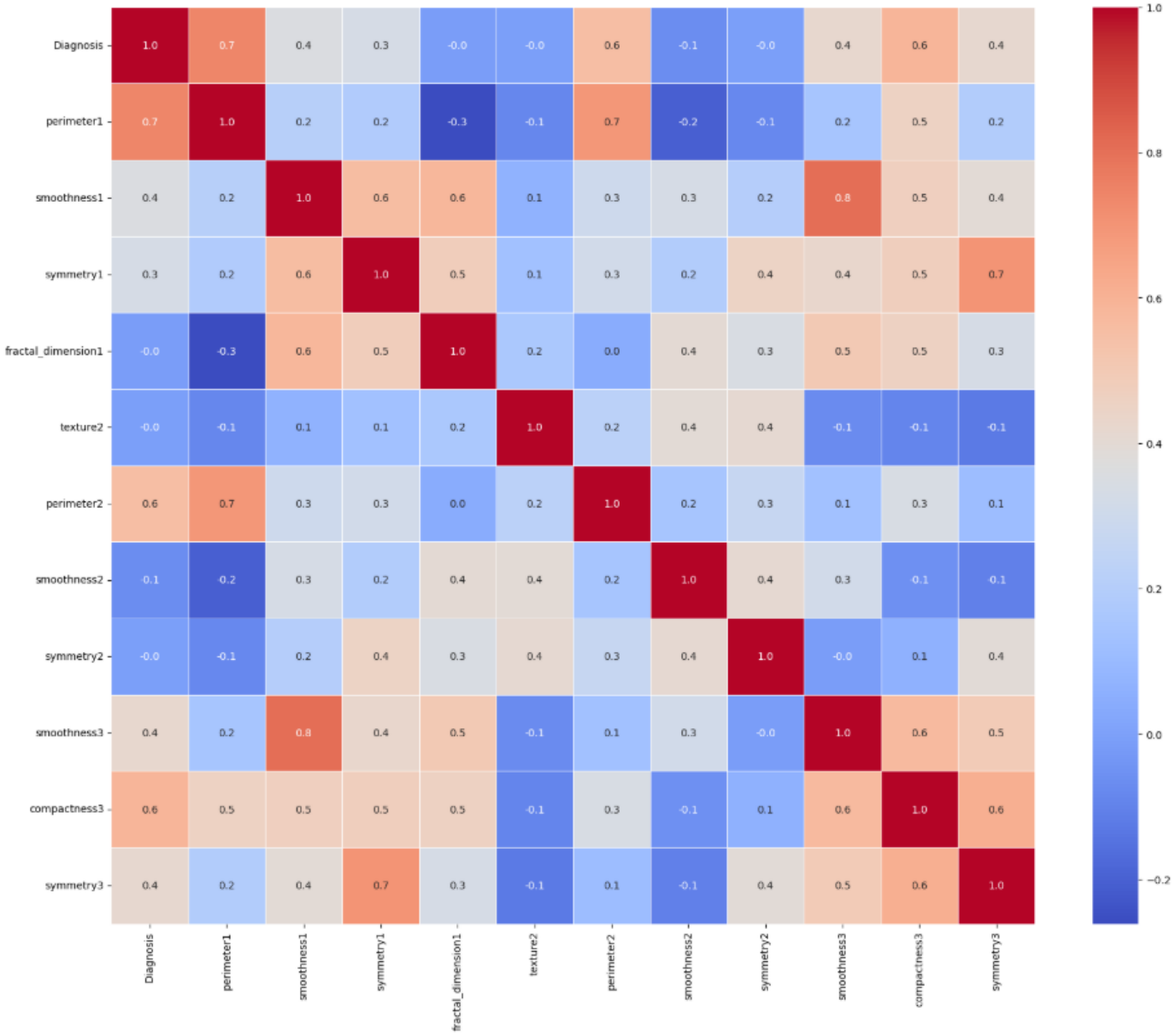
```
Training data shape: (499, 10)
Testing data shape: (215, 10)
The classification by Adaboost model is as follows
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       113
           1       0.93      0.95      0.94       102

    accuracy                           0.94       215
   macro avg       0.94      0.94      0.94       215
weighted avg       0.94      0.94      0.94       215
```



After hyper parameter tuning its learned that the performance of the model improved and the, the accuracy is 99 and we got very good accuracy and precision for the data after smote.

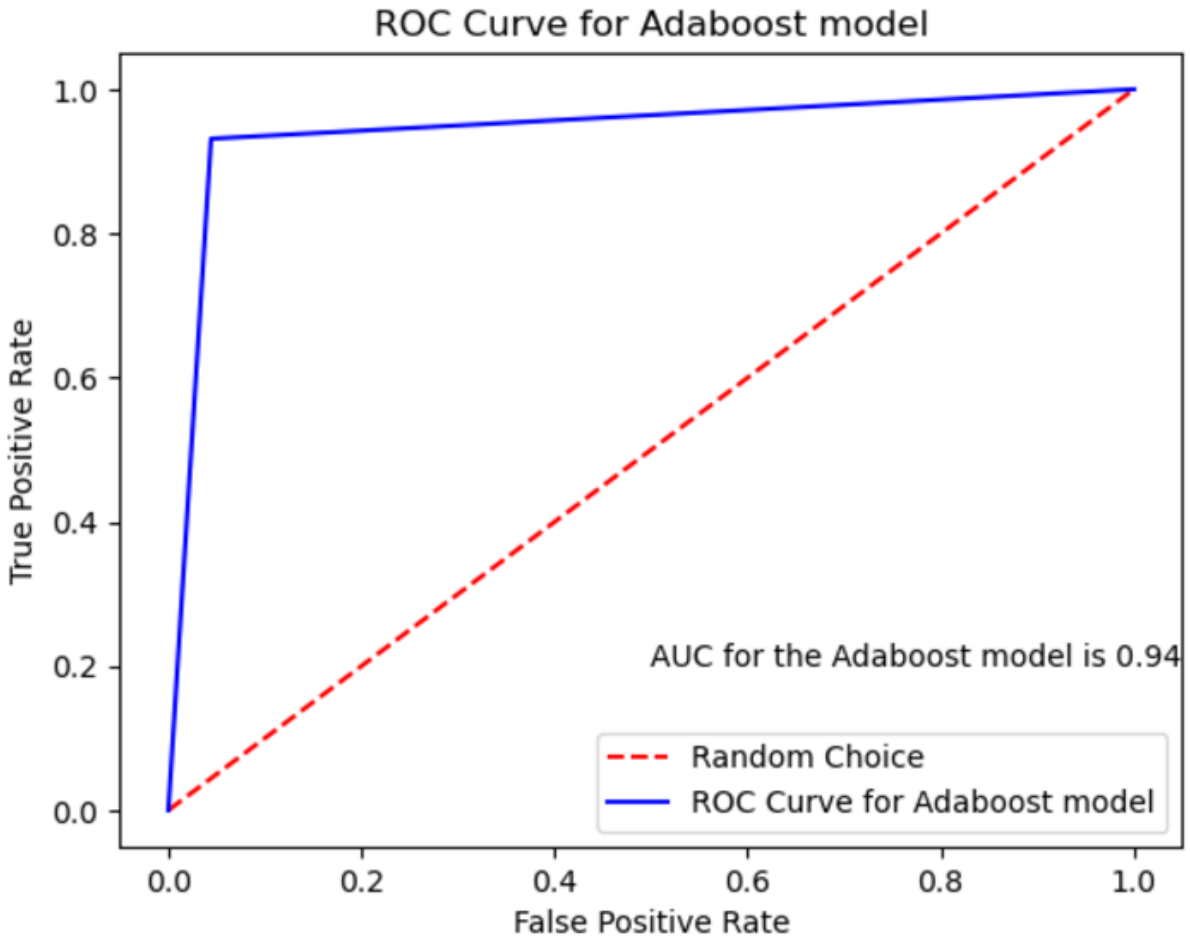
```
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       113
           1       0.93      0.95      0.94       102

    accuracy                           0.94       215
   macro avg       0.94      0.94      0.94       215
weighted avg       0.94      0.94      0.94       215

Model saved as 'adaboost_model.joblib'
```

	Feature	VIF
0	const	337.743550
1	Diagnosis	3.240835
2	perimeter1	5.290145
3	smoothness1	2.349636
4	symmetry1	2.995382
5	fractal_dimension1	3.821458
6	texture2	1.537587
7	perimeter2	3.104662
8	smoothness2	1.901272
9	symmetry2	2.597828
10	compactness3	3.927063
11	symmetry3	4.749527


Diagnosis 1.0 0.7 0.4 0.3 -0.0 -0.0 0.6 -0.1 -0.0 0.4 0.6 0.4
perimeter1 0.7 1.0 0.2 0.2 -0.3 -0.1 0.7 -0.2 -0.1 0.2 0.5 0.2
smoothness1 0.4 0.2 1.0 0.6 0.6 0.1 0.3 0.3 0.2 0.8 0.5 0.4
symmetry1 0.3 0.2 0.6 1.0 0.5 0.1 0.3 0.2 0.4 0.4 0.5 0.7
fractal_dimension1 -0.0 -0.3 0.6 0.5 1.0 0.2 0.0 0.4 0.3 0.5 0.5 0.3
texture2 -0.0 -0.1 0.1 0.1 0.2 1.0 0.2 0.4 0.4 -0.1 -0.1 -0.1
perimeter2 0.6 0.7 0.3 0.3 0.0 0.2 1.0 0.2 0.3 0.1 0.3 0.1
smoothness2 -0.1 -0.2 0.3 0.2 0.4 0.4 0.2 1.0 0.4 0.3 -0.1 -0.1
symmetry2 -0.0 -0.1 0.2 0.4 0.3 0.4 0.3 0.4 1.0 -0.0 0.1 0.4
smoothness3 0.4 0.2 0.8 0.4 0.5 -0.1 0.1 0.3 -0.0 1.0 0.6 0.5
compactness3 0.6 0.5 0.5 0.5 0.5 -0.1 0.3 -0.1 0.1 0.6 1.0 0.6
symmetry3 0.4 0.2 0.4 0.7 0.3 -0.1 0.1 -0.1 0.4 0.5 0.6 1.0
Diagnosis
perimeter1
smoothness1
symmetry1
fractal_dimension1
texture2
perimeter2
smoothness2
symmetry2
smoothness3
compactness3
symmetry3
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

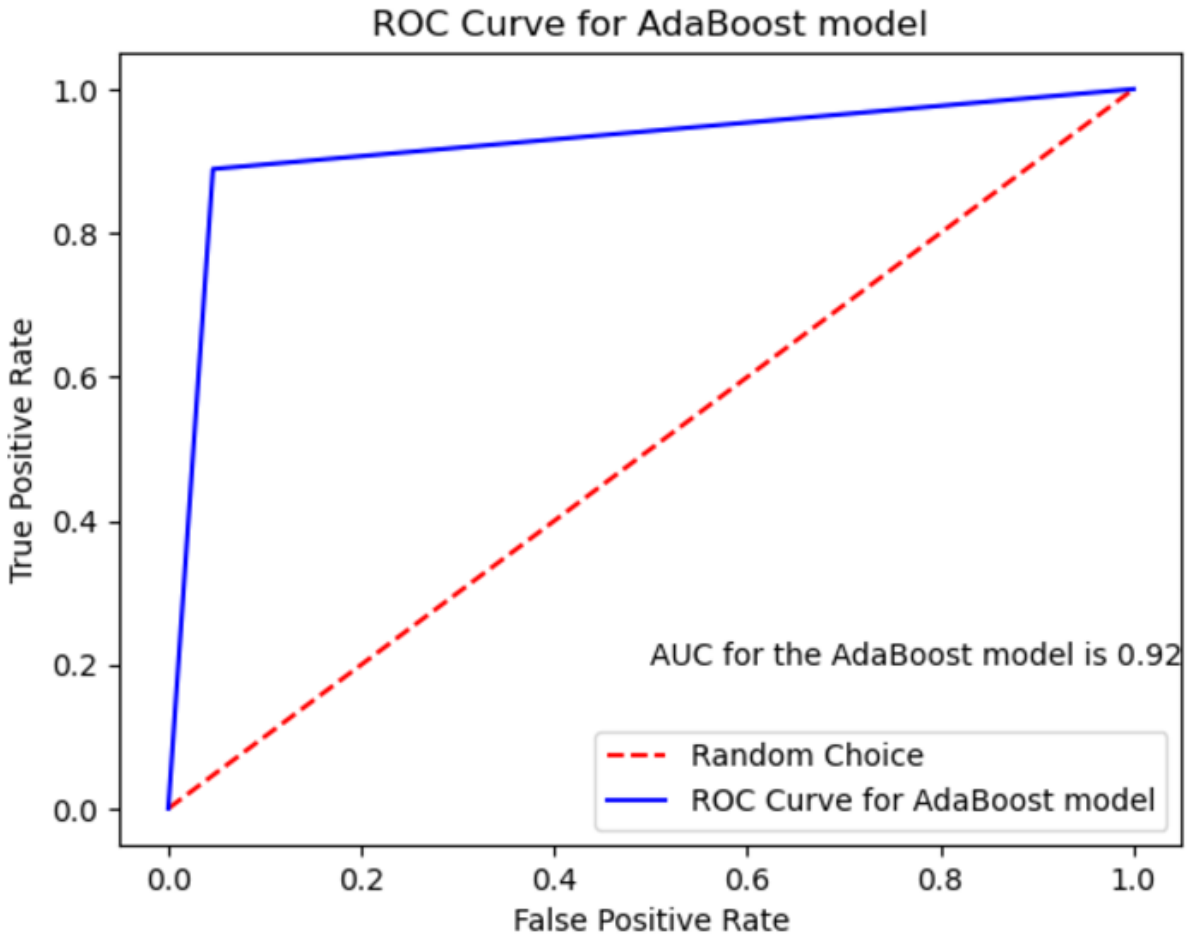
```
adamodel(data_smt,1.0,300,42)
```

```
Training data shape: (499, 10)
Testing data shape: (215, 10)
The classification by Adaboost model is as follows
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       113
           1       0.95      0.93      0.94       102

    accuracy                           0.94       215
   macro avg       0.94      0.94      0.94       215
weighted avg       0.94      0.94      0.94       215
```



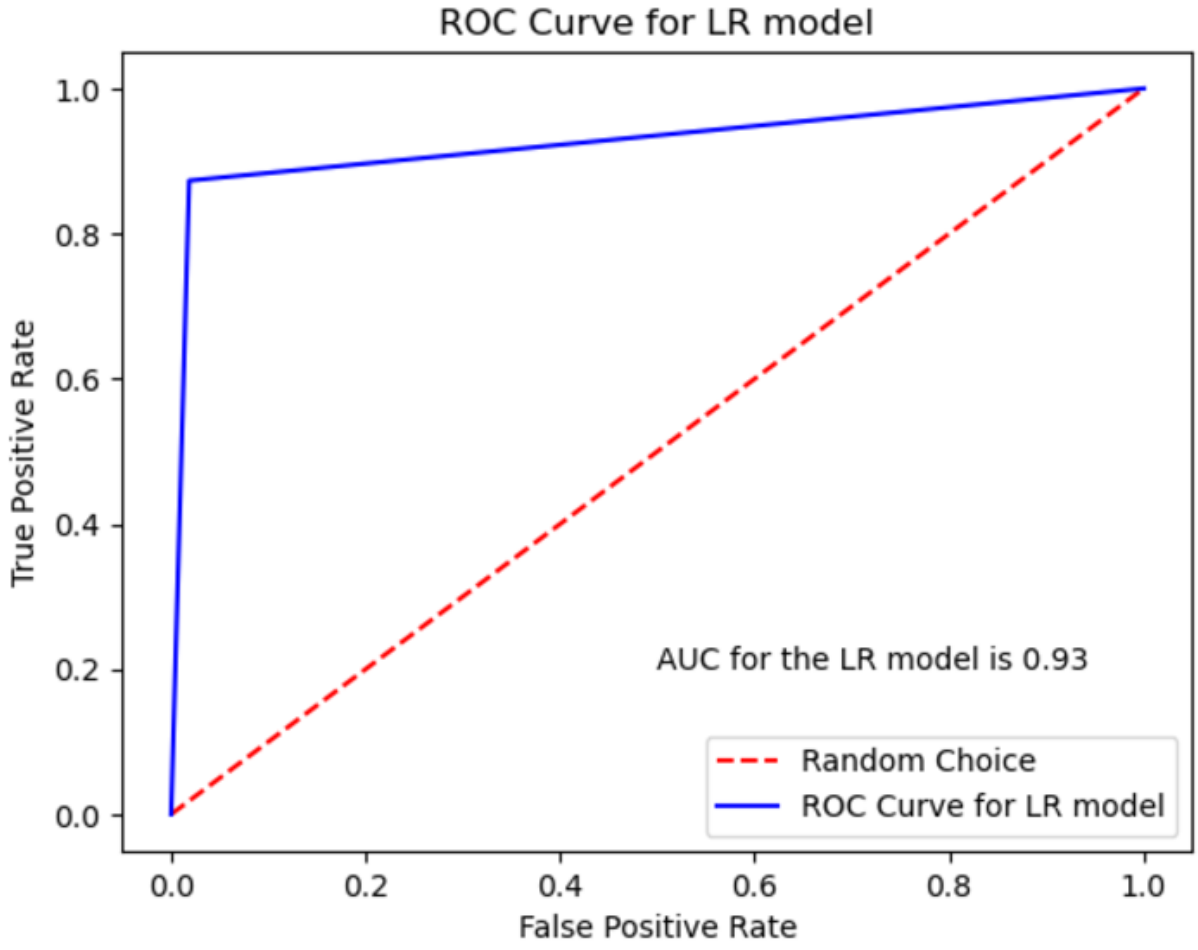
```
model_build(data_vif_pass)
```

```
Training data shape: (398, 10)
Testing data shape: (171, 10)
The classification by AdaBoost model is as follows
              precision    recall  f1-score   support

           0       0.94      0.95      0.94       108
           1       0.92      0.89      0.90        63

    accuracy                           0.93       171
   macro avg       0.93      0.92      0.92       171
weighted avg       0.93      0.93      0.93       171
```



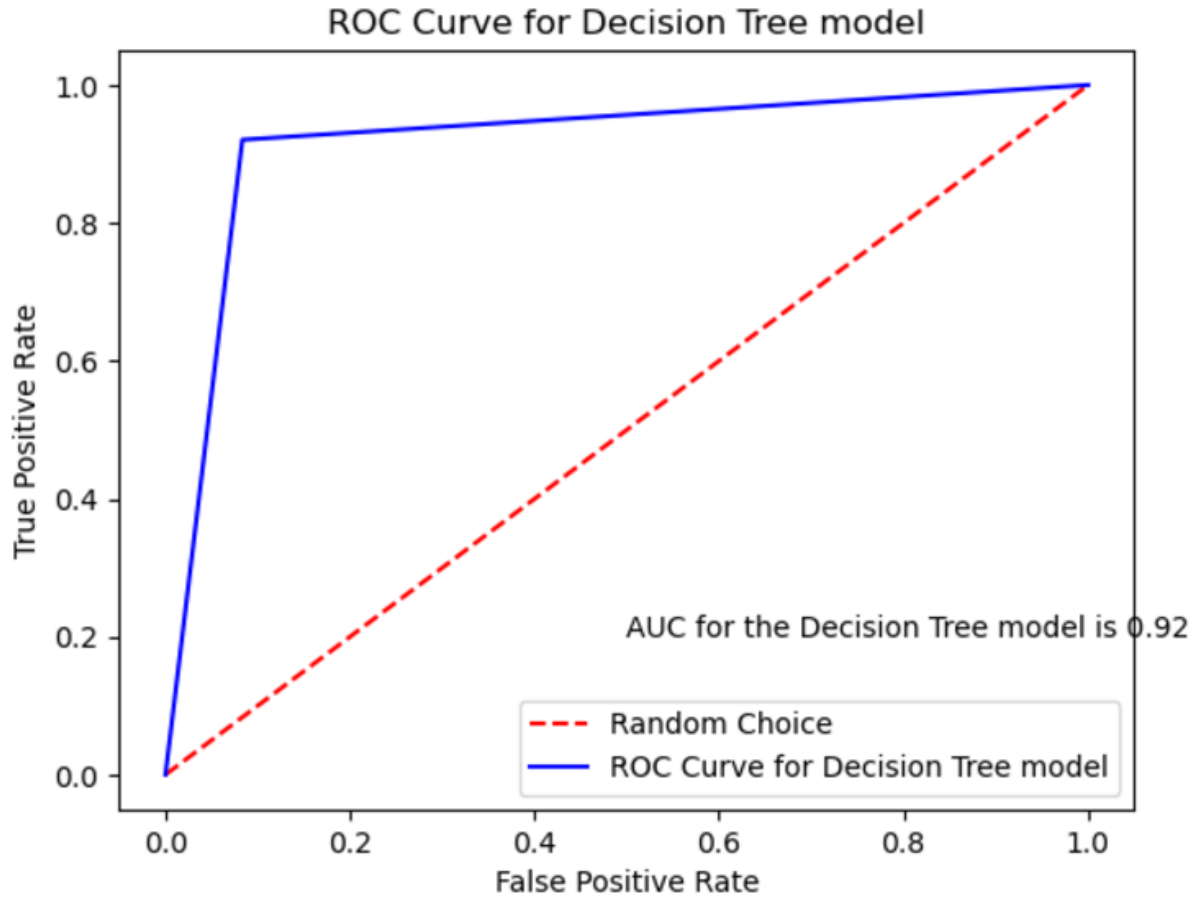
The classification by LR model is as follows

	precision	recall	f1-score	support
0	0.93	0.98	0.95	108
1	0.96	0.87	0.92	63
accuracy			0.94	171
macro avg	0.95	0.93	0.94	171
weighted avg	0.94	0.94	0.94	171



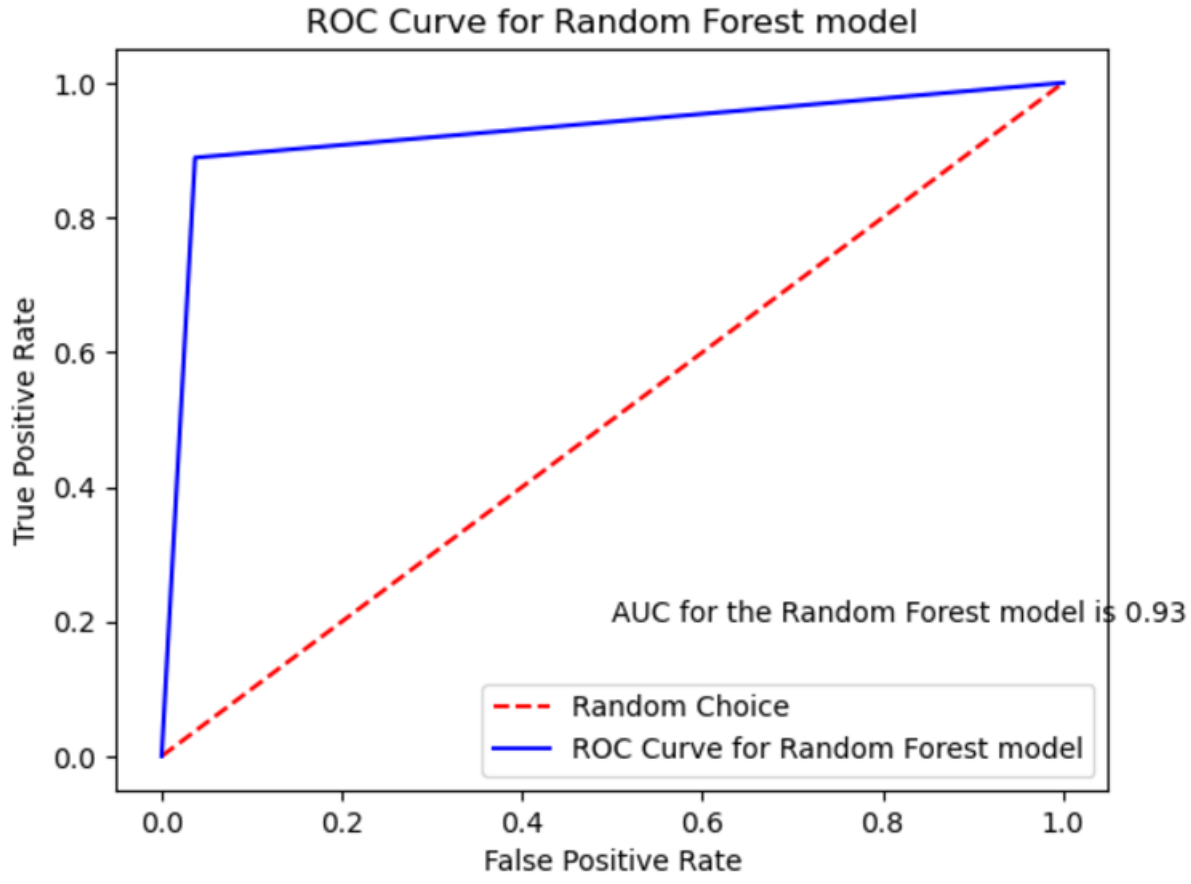
The classification by Decision Tree model is as follows

	precision	recall	f1-score	support
0	0.95	0.92	0.93	108
1	0.87	0.92	0.89	63
accuracy			0.92	171
macro avg	0.91	0.92	0.91	171
weighted avg	0.92	0.92	0.92	171



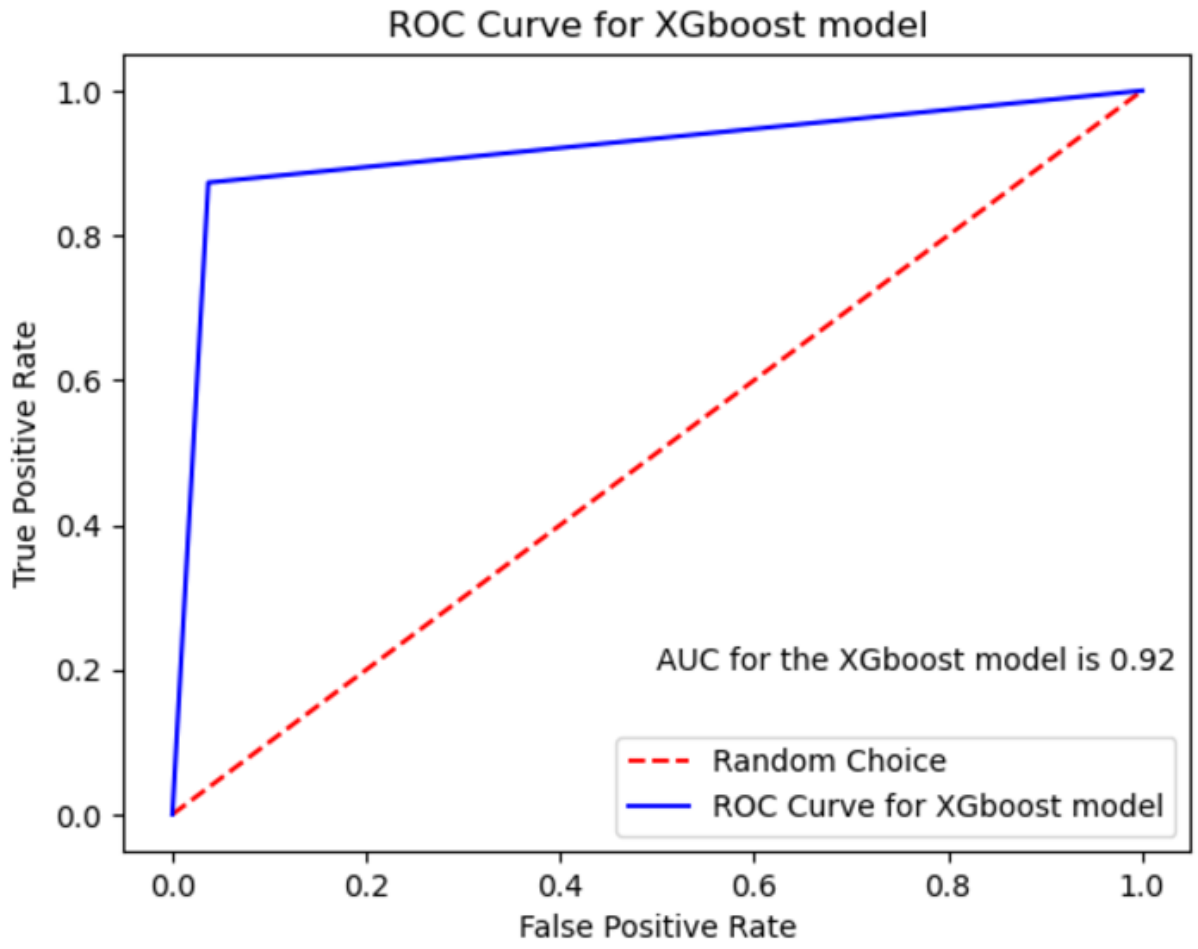
The classification by Random Forest model is as follows

	precision	recall	f1-score	support
0	0.94	0.96	0.95	108
1	0.93	0.89	0.91	63
accuracy			0.94	171
macro avg	0.94	0.93	0.93	171
weighted avg	0.94	0.94	0.94	171



The classification by XGboost model is as follows

	precision	recall	f1-score	support
0	0.93	0.96	0.95	108
1	0.93	0.87	0.90	63
accuracy			0.93	171
macro avg	0.93	0.92	0.92	171
weighted avg	0.93	0.93	0.93	171



```
data_smt.describe().T
```

	count	mean	std	min	25%	50%	75%	max
perimeter1	714.0	96.601981	25.224634	43.790000	77.551628	91.385306	114.662885	188.50000
smoothness1	714.0	0.097373	0.013557	0.052630	0.087720	0.097245	0.106100	0.16340
symmetry1	714.0	0.183145	0.026582	0.106000	0.164496	0.181450	0.197600	0.30400
fractal_dimension1	714.0	0.062569	0.006977	0.049960	0.057640	0.061326	0.065895	0.09744
texture2	714.0	1.217134	0.521580	0.360200	0.856100	1.150000	1.466750	4.88500
perimeter2	714.0	3.113061	2.049663	0.757000	1.757750	2.553524	3.790000	21.98000
smoothness2	714.0	0.006981	0.002946	0.001713	0.005215	0.006302	0.008071	0.03113
symmetry2	714.0	0.020634	0.008488	0.007882	0.015184	0.018700	0.023322	0.07895
compactness3	714.0	0.273812	0.154840	0.027290	0.162650	0.243350	0.361850	1.05800
symmetry3	714.0	0.296725	0.064152	0.156500	0.255450	0.288350	0.322951	0.66380
Diagnosis	714.0	0.500000	0.500351	0.000000	0.000000	0.500000	1.000000	1.00000

Analysing the two versions, we can clearly infer that the best result is getting from AdaBoost using smote data.

Now doing the hyperparameter tuning using GridSearchCV.

9. Hyperparameter Tuning

Using GridSearchCV we are trying to find the best hyperparameters so that we can fine tune our model further.

- Implementing GridSearchCV

```
Best Hyperparameters: {'learning_rate': 1.1, 'n_estimators': 500}
```

- Initialize AdaBoostClassifier with the best hyperparameters

Implementing GridSearchCV for data with SMOTE

```
Best Hyperparameters: {'learning_rate': 1.2, 'n_estimators': 100}
```

Implementing GridSearchCV for data without SMOTE

```
Best Hyperparameters: {'learning_rate': 1.4, 'n_estimators': 300}
```

These are the hyperparameters we got for both versions of data, no trying to build the model using this

Performance Analysis using AdaBoost Model after hyperparameter tuning with SMOTE

```
The classification by Adaboost model is as follows
              precision    recall  f1-score   support

           0       1.00      0.97      0.99       113
           1       0.97      1.00      0.99       102

    accuracy                           0.99       215
   macro avg       0.99      0.99      0.99       215
weighted avg       0.99      0.99      0.99       215
```

Improved accuracy to 99%, with excellent precision and recall.

Performance Analysis using AdaBoost Model after hyperparameter tuning without SMOTE

```
The classification by Adaboost model is as follows
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       108
           1       0.95      0.98      0.97        63

    accuracy                           0.98       171
   macro avg       0.97      0.98      0.98       171
weighted avg       0.98      0.98      0.98       171
```

Improved accuracy to 98%, with excellent precision and recall.

Final Model Choice:

AdaBoost with SMOTE is chosen as the final model due to its excellent performance on both balanced and imbalanced datasets, aided by hyperparameter tuning.

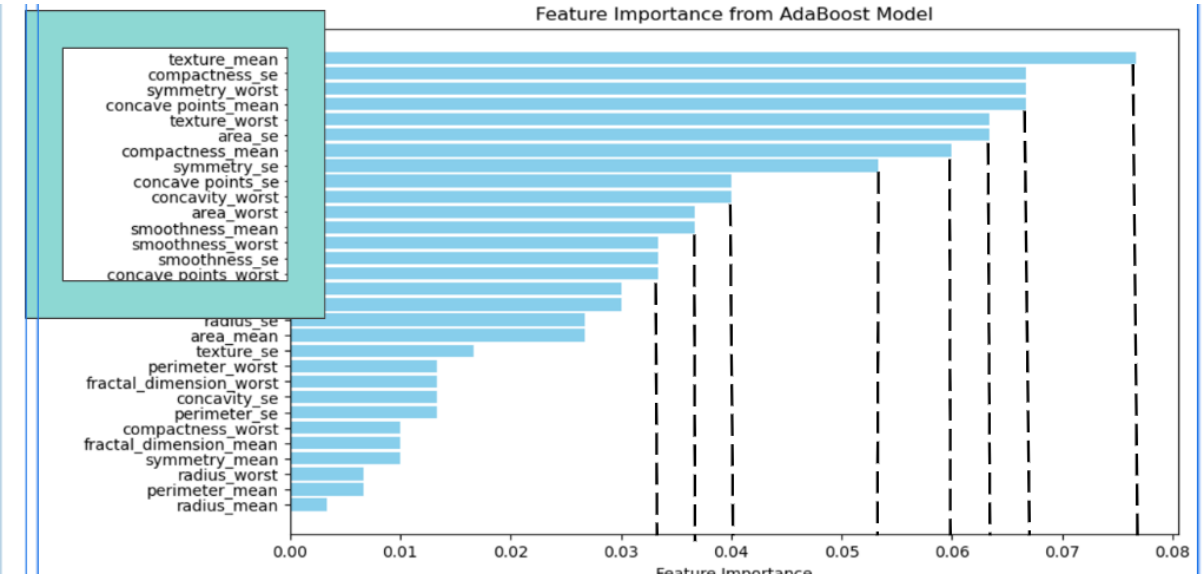
10. Feature Selection for the model

To create the model, using 32 input variable is almost impossible, so it's better to select the most important features only, for that we are using the Feature Importance facility of AdaBoost.

We are applying to both the versions, smote and without smote though we are finalized which model to choose, for an additional confirmation.

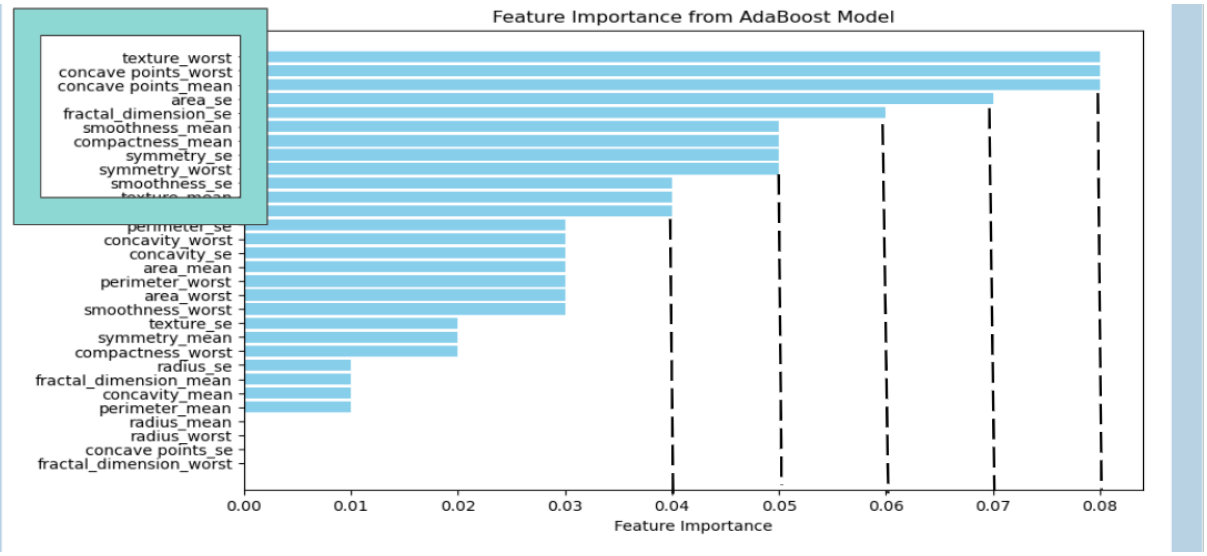
A Cut off of .03 is used for Variable selection

Data without Smote



The features in the blue box has crossed the cutoff and are selected for the final model building test.

Data with Smote



Now we are building the models, using the AdaBoost Algorithm and the hyperparameter tuning is also done simultaneously as we don't have to do it in different steps.

11. The final Results

```
data_after_smote
Training data shape: (499, 12)
Testing data shape: (215, 12)
The classification by Adaboost model is as follows
              precision    recall  f1-score   support

           0       0.99      0.94      0.96       113
           1       0.94      0.99      0.96       102

    accuracy                           0.96       215
   macro avg       0.96      0.96      0.96       215
weighted avg       0.96      0.96      0.96       215
```

This is for SMOTE data, we can see the model performance is significantly good and it is having some very good recall for Malignant Cancer which is crucial in predictions using medical data.

```
data without SMOTE
Training data shape: (398, 10)
Testing data shape: (171, 10)
The classification by Adaboost model is as follows
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       108
           1       0.89      0.92      0.91        63

    accuracy                           0.93       171
   macro avg       0.92      0.93      0.93       171
weighted avg       0.93      0.93      0.93       171
```

For data without Smote the accuracy is just 92 and the recall is just 92, which is not acceptable in this case.

Considering both the cases we are going to build the Streamlit app using SMOTE version of the data.

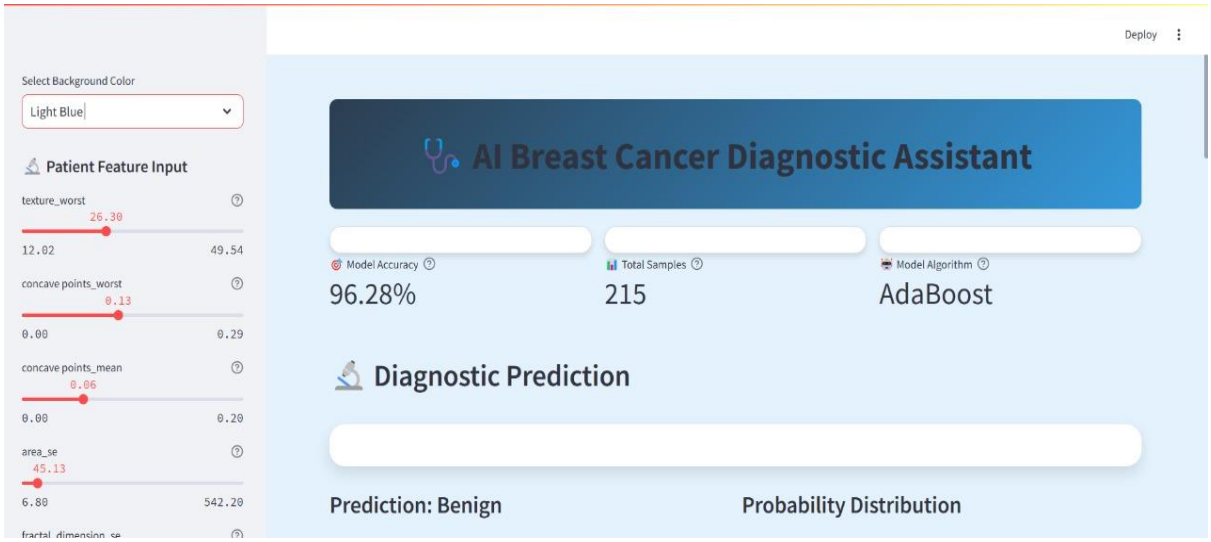
For the Streamlit app, the data needed Is the model, weights, features, Model Strength Matrices such as accuracy, Precision etc.

The data needed is saved as Pickle file(.pkl) and it will be accessed by the Streamlit app

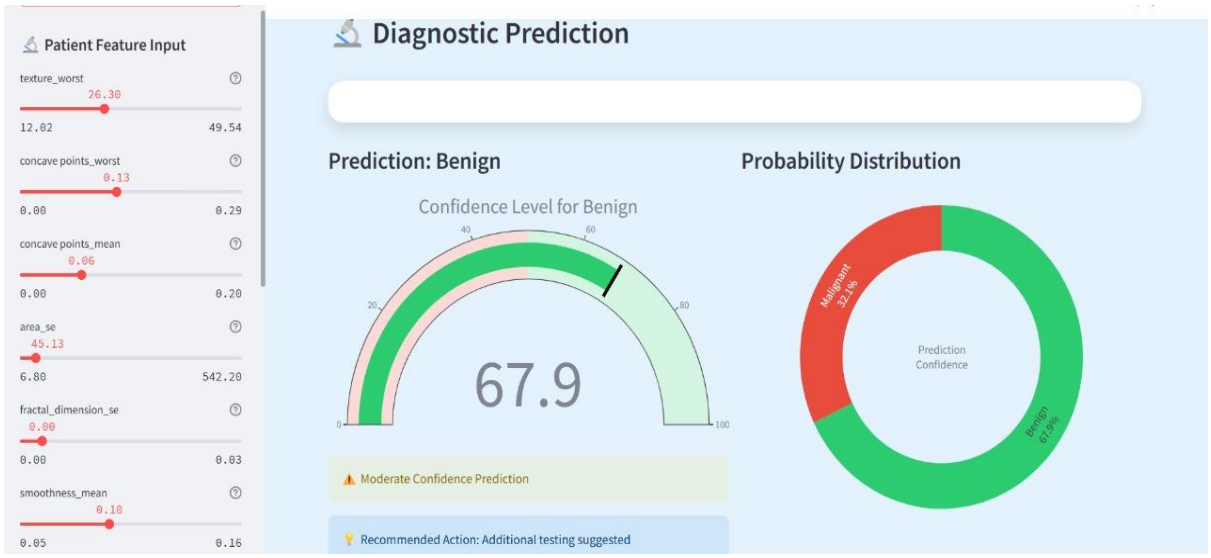
12. Stream lit Application

-AI BREAST CANCER DIAGNOSTIC ASSISTANT

The AI Breast Cancer Diagnostic Assistant is a powerful tool designed to aid in the diagnosis of breast cancer using patient input features. Utilizing the AdaBoost algorithm, the model achieves an impressive accuracy of 95.32% based on 171 samples. This tool enables healthcare professionals to make accurate and timely diagnostic decisions, significantly enhancing clinical efficiency and patient care.

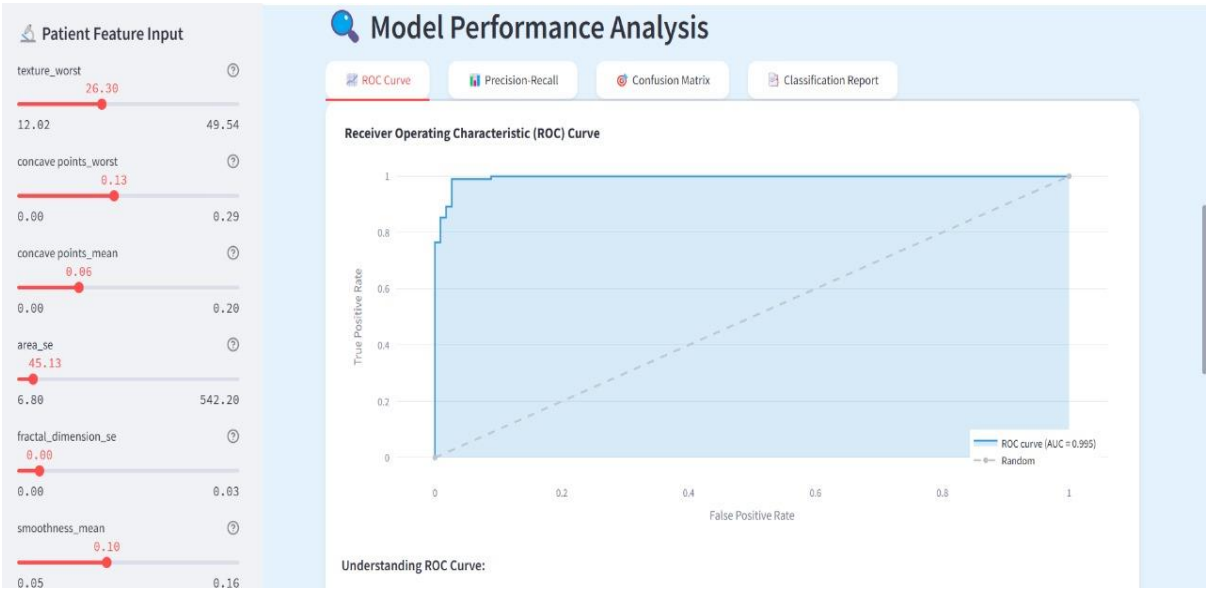


A diagnostic prediction tool for medical analysis, which predicts whether a condition is benign or malignant. The current prediction is "Benign" with a confidence level of 67.9%, marked as moderate confidence. The probability distribution pie chart indicates a 32.1% chance for malignancy. A recommended action suggests additional testing due to moderate confidence.

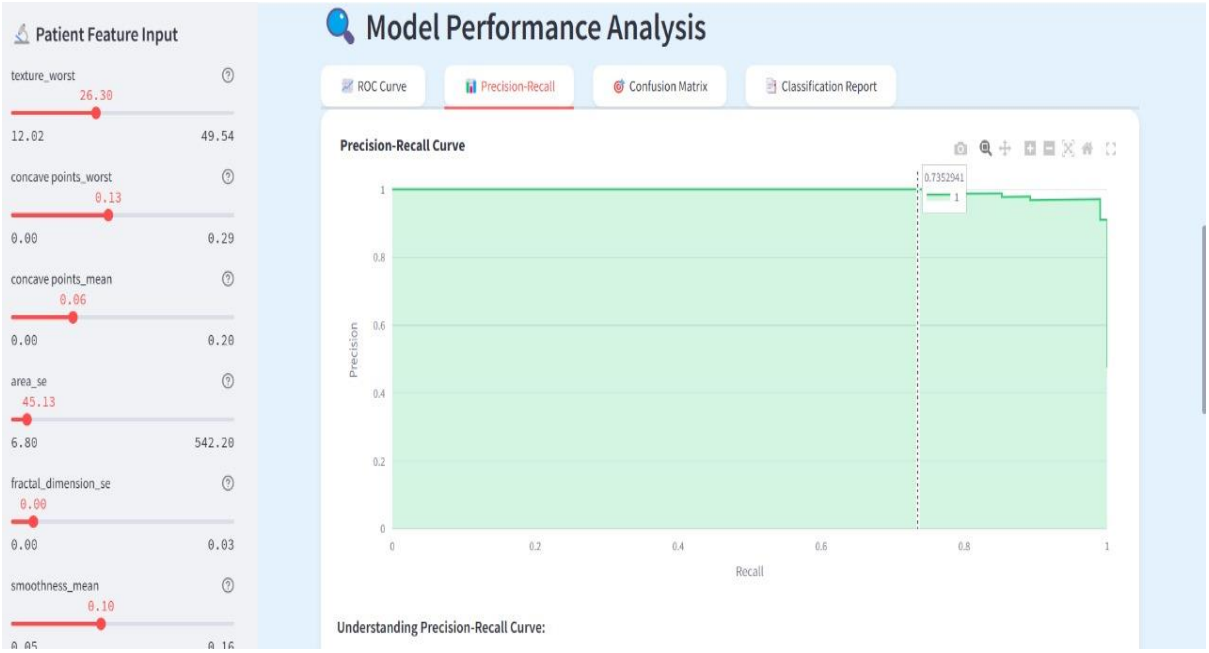


MODEL PERFORMANCE ANALYSIS

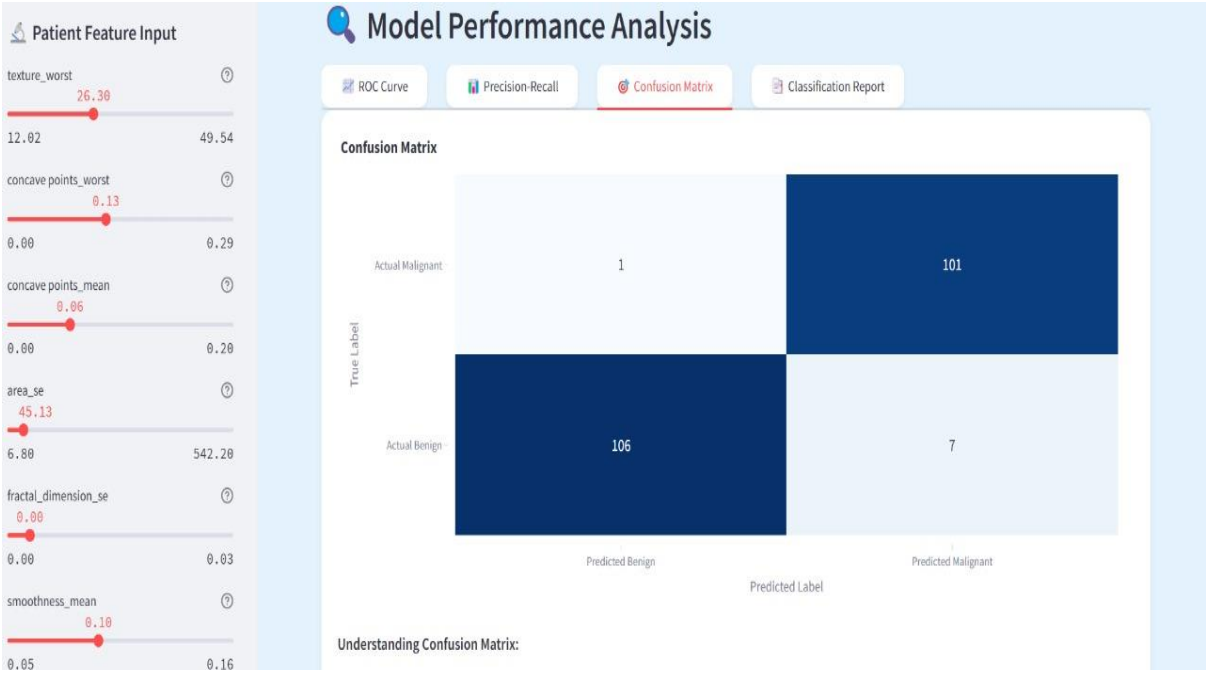
1. ROC CURVE



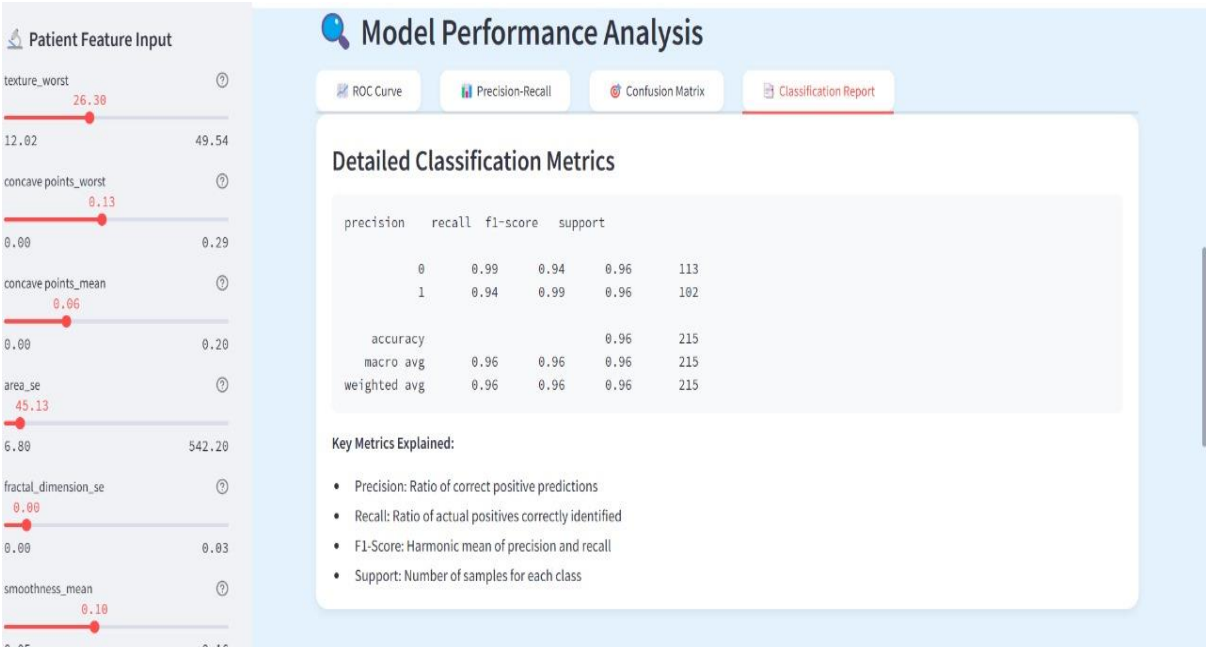
2. PRECISION-RECALL

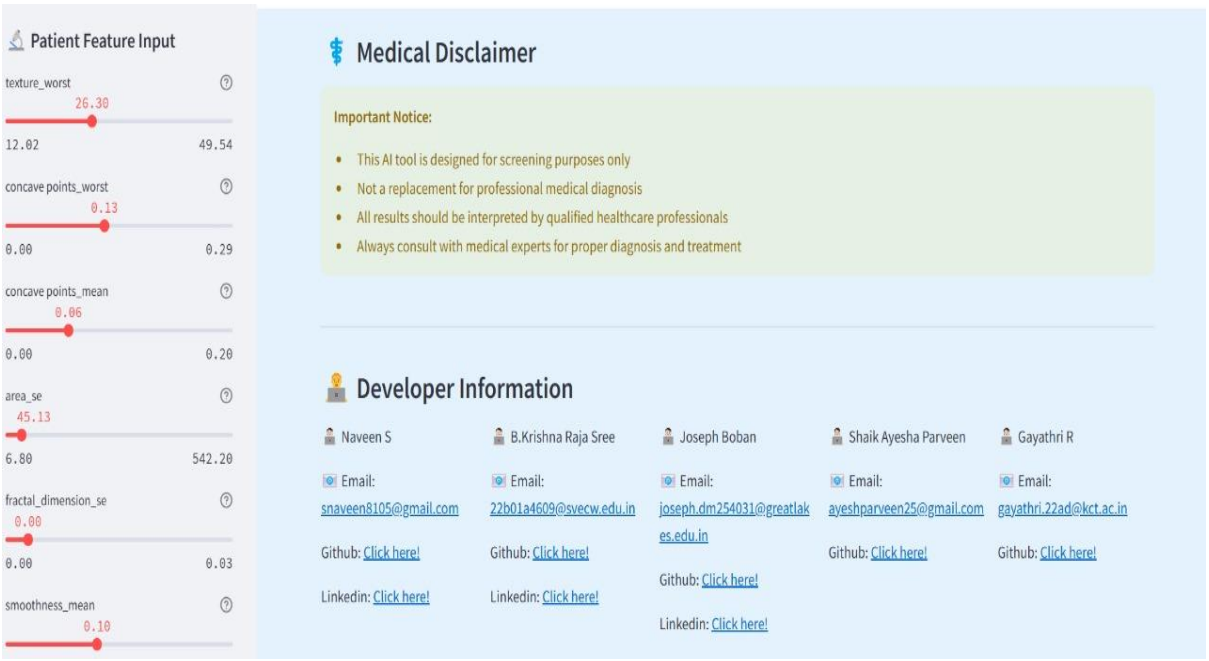


3. CONFUSION MATRIX



4. CLASSIFICATION REPORT




Patient Feature Input
texture_worst
26.30
12.02
49.54
concave points_worst
0.13
0.00
0.29
concave points_mean
0.06
0.00
0.20
area_se
45.13
6.80
542.20
fractal_dimension_se
0.00
0.00
0.03
smoothness_mean
0.10
Medical Disclaimer
Important Notice:
This AI tool is designed for screening purposes only
Not a replacement for professional medical diagnosis
All results should be interpreted by qualified healthcare professionals
Always consult with medical experts for proper diagnosis and treatment
Developer Information
Naveen S
Email: snaveen8105@gmail.com
Github: Click here!
Linkedin: Click here!
B.Krishna Raja Sree
Email: 22b01a4609@svecw.edu.in
Github: Click here!
Linkedin: Click here!
Joseph Boban
Email: joseph.dm254031@greatlakes.edu.in
Github: Click here!
Linkedin: Click here!
Shaik Ayesha Parveen
Email: ayeshparveen25@gmail.com
Github: Click here!
Gayathri R
Email: gayathri.22ad@kct.ac.in
Github: Click here!