Experiment 4: Ensemble Prediction and Decision Tree Model Evaluation

**Git Hub:** <https://github.com/Vignesh-0013/Machine_Learning>

# Aim:

To implement and evaluate multiple machine learning classifiers—including Decision Tree,

AdaBoost,Gradient Boosting, XGBoost, Random Forest, and Stacked Ensemble—on the Wisconsin

Diagnostic Dataset, optimize their performance through hyperparameter tuning, and compare their predictive accuracy and generalization using 5-Fold Cross-Validation.

# Libraries used:

* + Numpy
  + Pandas
  + Matplotlib
  + Scikit-learn
  + Seaborn

# Objective:

To build classifiers such as Decision Tree, AdaBoost, Gradient Boosting, XGBoost, Random Forest, and Stacked Models (using SVM, Naive Bayes, Decision Tree) and evaluate their performance

through 5-Fold Cross-Validation and hyperparameter tuning.

# Code

import pandas as pd import numpy as np

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, roc\_curve, auc import matplotlib.pyplot as plt

import seaborn as sns

from ucimlrepo import fetch\_ucirepo # fetch dataset

breast\_cancer\_wisconsin\_diagnostic = fetch\_ucirepo(id=17)

# data (as pandas dataframes)

X = breast\_cancer\_wisconsin\_diagnostic.data.features y = breast\_cancer\_wisconsin\_diagnostic.data.targets

# metadata

print(breast\_cancer\_wisconsin\_diagnostic.metadata)

# variable information

print(breast\_cancer\_wisconsin\_diagnostic.variables)

 {'uci\_id': 17, 'name': 'Breast Cancer Wisconsin (Diagnostic)', 'repository\_url': '[https://archive.ics.uci.edu/dataset/17/breast+c](https://archive.ics.uci.edu/dataset/17/breast%2Bcancer%2Bwisconsin%2Bdiagnostic)a name role type demographic description units \

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | ID | ID | Categorical | None | None | None |
| 1 | Diagnosis | Target | Categorical | None | None | None |
| 2 | radius1 | Feature | Continuous | None | None | None |
| 3 | texture1 | Feature | Continuous | None | None | None |
| 4 | perimeter1 | Feature | Continuous | None | None | None |
| 5 | area1 | Feature | Continuous | None | None | None |
| 6 | smoothness1 | Feature | Continuous | None | None | None |
| 7 | compactness1 | Feature | Continuous | None | None | None |
| 8 | concavity1 | Feature | Continuous | None | None | None |
| 9 | concave\_points1 | Feature | Continuous | None | None | None |
| 10 | symmetry1 | Feature | Continuous | None | None | None |
| 11 | fractal\_dimension1 | Feature | Continuous | None | None | None |
| 12 | radius2 | Feature | Continuous | None | None | None |
| 13 | texture2 | Feature | Continuous | None | None | None |
| 14 | perimeter2 | Feature | Continuous | None | None | None |
| 15 | area2 | Feature | Continuous | None | None | None |
| 16 | smoothness2 | Feature | Continuous | None | None | None |
| 17 | compactness2 | Feature | Continuous | None | None | None |
| 18 | concavity2 | Feature | Continuous | None | None | None |
| 19 | concave\_points2 | Feature | Continuous | None | None | None |
| 20 | symmetry2 | Feature | Continuous | None | None | None |
| 21 | fractal\_dimension2 | Feature | Continuous | None | None | None |
| 22 | radius3 | Feature | Continuous | None | None | None |
| 23 | texture3 | Feature | Continuous | None | None | None |
| 24 | perimeter3 | Feature | Continuous | None | None | None |
| 25 | area3 | Feature | Continuous | None | None | None |
| 26 | smoothness3 | Feature | Continuous | None | None | None |
| 27 | compactness3 | Feature | Continuous | None | None | None |
| 28 | concavity3 | Feature | Continuous | None | None | None |
| 29 | concave\_points3 | Feature | Continuous | None | None | None |
| 30 | symmetry3 | Feature | Continuous | None | None | None |
| 31 | fractal\_dimension3 | Feature | Continuous | None | None | None |

missing\_values

1. no
2. no
3. no
4. no
5. no
6. no
7. no
8. no
9. no
10. no
11. no
12. no
13. no
14. no
15. no
16. no
17. no
18. no
19. no
20. no
21. no

import pandas as pd

df = pd.concat([X, y], axis=1) # axis=1 → concatenate column-wise

print(df.head)

print(df.describe)

 <bound method NDFrame.head of radius1 texture1 perimeter1 area1 smoothness1 compactness1 \

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | | 0.27760 | |
| 1 | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | | 0.07864 | |
| 2 | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | | 0.15990 | |
| 3 | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | | 0.28390 | |
| 4 | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | | 0.13280 | |
| .. | ... | ... | ... | ... | ... | | ... | |
| 564 | 21.56 | 22.39 | 142.00 | 1479.0 | 0.11100 | | 0.11590 | |
| 565 | 20.13 | 28.25 | 131.20 | 1261.0 | 0.09780 | | 0.10340 | |
| 566 | 16.60 | 28.08 | 108.30 | 858.1 | 0.08455 | | 0.10230 | |
| 567 | 20.60 | 29.33 | 140.10 | 1265.0 | 0.11780 | | 0.27700 | |
| 568 | 7.76 | 24.54 | 47.92 | 181.0 | 0.05263 | | 0.04362 | |
|  | concavity1 | concave\_points1 | | symmetry1 | fractal\_dimension1 ... \ | | | |
| 0 | 0.30010 | 0.14710 | | 0.2419 | 0.07871 ... | | | |
| 1 | 0.08690 | 0.07017 | | 0.1812 | 0.05667 ... | | | |
| 2 | 0.19740 | 0.12790 | | 0.2069 | 0.05999 ... | | | |
| 3 | 0.24140 | 0.10520 | | 0.2597 | 0.09744 ... | | | |
| 4 | 0.19800 | 0.10430 | | 0.1809 | 0.05883 ... | | | |
| .. | ... | ... | | ... | ... ... | | | |
| 564 | 0.24390 | 0.13890 | | 0.1726 | 0.05623 ... | | | |
| 565 | 0.14400 | 0.09791 | | 0.1752 | 0.05533 ... | | | |
| 566 | 0.09251 | 0.05302 | | 0.1590 | 0.05648 ... | | | |
| 567 | 0.35140 | 0.15200 | | 0.2397 | 0.07016 ... | | | |
| 568 | 0.00000 | 0.00000 | | 0.1587 | 0.05884 ... | | | |
|  | texture3 | perimeter3 | area3 | smoothness3 | | compactness3 | concavity3 | \ |
| 0 | 17.33 | 184.60 | 2019.0 | 0.16220 | | 0.66560 | 0.7119 |  |
| 1 | 23.41 | 158.80 | 1956.0 | 0.12380 | | 0.18660 | 0.2416 |  |
| 2 | 25.53 | 152.50 | 1709.0 | 0.14440 | | 0.42450 | 0.4504 |  |
| 3 | 26.50 | 98.87 | 567.7 | 0.20980 | | 0.86630 | 0.6869 |  |
| 4 | 16.67 | 152.20 | 1575.0 | 0.13740 | | 0.20500 | 0.4000 |  |
| .. | ... | ... | ... | ... | | ... | ... |  |
| 564 | 26.40 | 166.10 | 2027.0 | 0.14100 | | 0.21130 | 0.4107 |  |
| 565 | 38.25 | 155.00 | 1731.0 | 0.11660 | | 0.19220 | 0.3215 |  |
| 566 | 34.12 | 126.70 | 1124.0 | 0.11390 | | 0.30940 | 0.3403 |  |
| 567 | 39.42 | 184.60 | 1821.0 | 0.16500 | | 0.86810 | 0.9387 |  |
| 568 | 30.37 | 59.16 | 268.6 | 0.08996 | | 0.06444 | 0.0000 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | concave\_points3  0.2654 | symmetry3  0.4601 | fractal\_dimension3  0.11890 | Diagnosis  M |
| 1 | 0.1860 | 0.2750 | 0.08902 | M |
| 2 | 0.2430 | 0.3613 | 0.08758 | M |
| 3 | 0.2575 | 0.6638 | 0.17300 | M |
| 4 | 0.1625 | 0.2364 | 0.07678 | M |
| .. | ... | ... | ... | ... |
| 564 | 0.2216 | 0.2060 | 0.07115 | M |
| 565 | 0.1628 | 0.2572 | 0.06637 | M |
| 566 | 0.1418 | 0.2218 | 0.07820 | M |
| 567 | 0.2650 | 0.4087 | 0.12400 | M |
| 568 | 0.0000 | 0.2871 | 0.07039 | B |

[569 rows x 31 columns]>

<bound method NDFrame.describe of radius1 texture1 perimeter1 area1 smoothness1 compactness1 \

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 |
| 1 | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 |
| 2 | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 |
| 3 | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 |
| df.columns |  |  |  |  |  |  |

 Index(['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', 'Diagnosis'],

dtype='object')

#Missing Values

print(df.isnull().sum())

#df.fillna(df.mean(), inplace=True)

 radius1 0

texture1 0

perimeter1 0

area1 0

smoothness1 0

compactness1 0

concavity1 0

concave\_points1 0

symmetry1 0

fractal\_dimension1 0

radius2 0

texture2 0

perimeter2 0

area2 0

smoothness2 0

compactness2 0

concavity2 0

concave\_points2 0

symmetry2 0

fractal\_dimension2 0

radius3 0

texture3 0

perimeter3 0

area3 0

smoothness3 0

compactness3 0

concavity3 0

concave\_points3 0

symmetry3 0

fractal\_dimension3 0

Diagnosis 0

dtype: int64

#Target

print(df['Diagnosis'].nunique()) print(df['Diagnosis'].unique())

 2

['M' 'B']

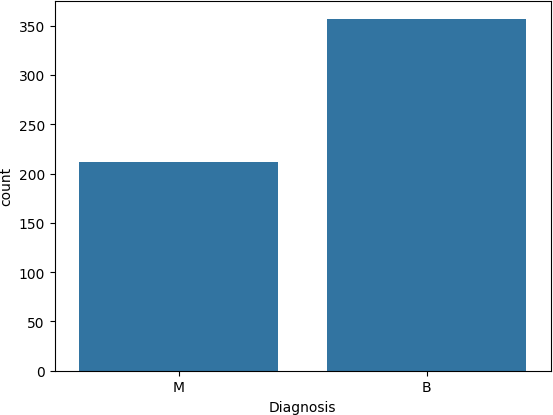
#Duplicate

df.duplicated().sum()  0

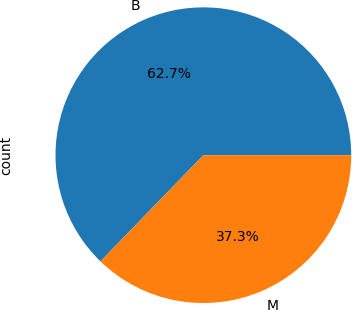
#target Class Distribution

sns.countplot(x='Diagnosis', data=df) plt.show()

df['Diagnosis'].value\_counts().plot.pie(autopct='%1.1f%%')

<Axes: ylabel='count'>



df.columns

 Index(['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', 'Diagnosis'],

dtype='object')

#Numeric Features #Histogram

num\_cols = len(df.columns)

n\_cols = 3

n\_rows = (num\_cols + n\_cols - 1) // n\_cols

fig, axes = plt.subplots(n\_rows, n\_cols, figsize=(15, 4 \* n\_rows)) axes = axes.flatten()

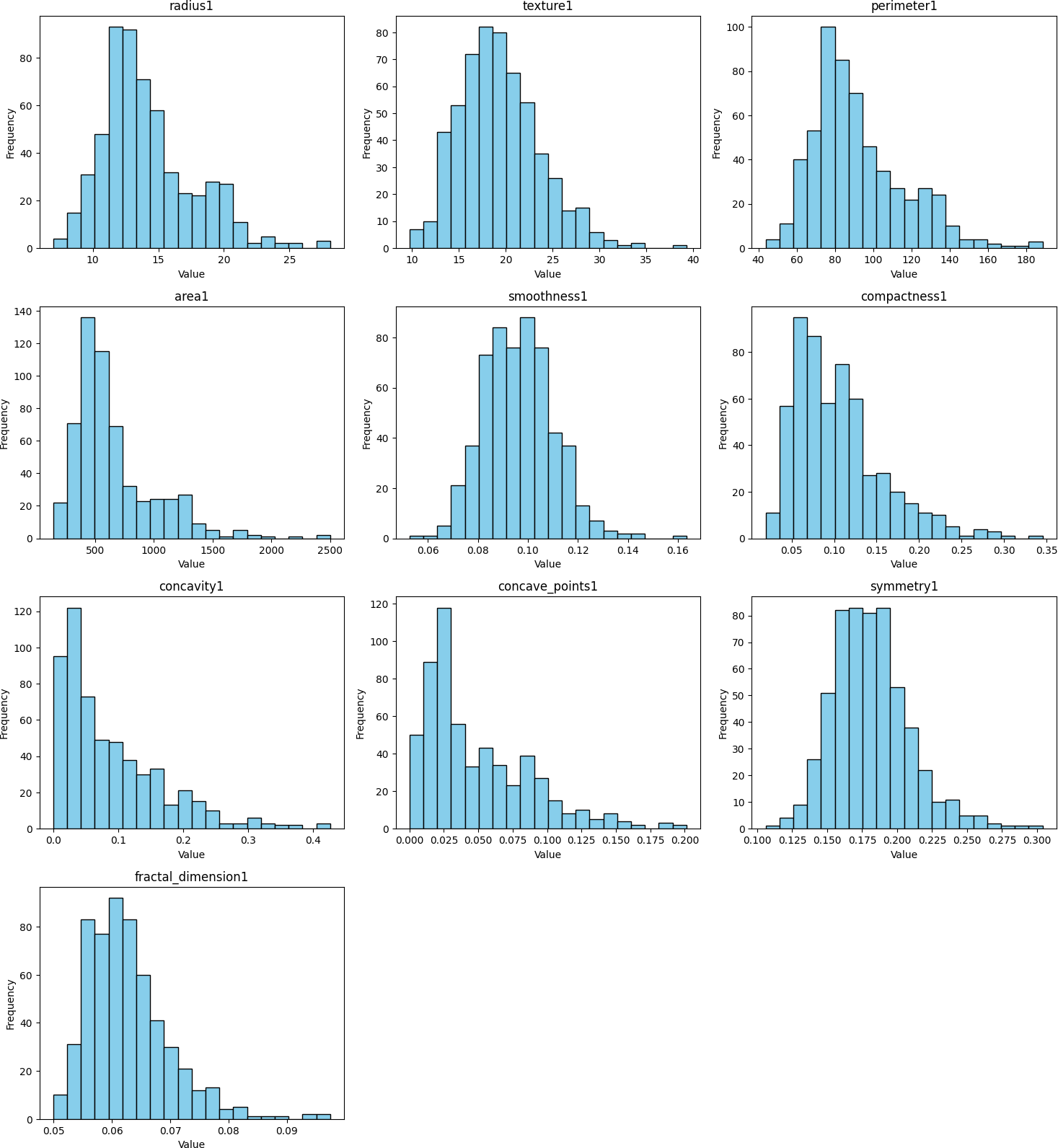
for i, col in enumerate(df.columns[:10]):

axes[i].hist(df[col].dropna(), bins=20, color='skyblue', edgecolor='black') axes[i].set\_title(col)

axes[i].set\_ylabel('Frequency') axes[i].set\_xlabel('Value')

for j in range(i + 1, len(axes)): fig.delaxes(axes[j])

plt.tight\_layout() plt.show()

#Boxplot

numeric\_cols = df.select\_dtypes(include=['int64', 'float64']).columns num\_cols = len(numeric\_cols)

# Define subplot grid size

n\_cols = 3 # number of plots per row

n\_rows = (num\_cols + n\_cols - 1) // n\_cols # ceiling division

# Create subplots

fig, axes = plt.subplots(n\_rows, n\_cols, figsize=(15, 4 \* n\_rows)) axes = axes.flatten()

# Plot boxplot for each numeric column

for i, col in enumerate(numeric\_cols[:10]):

axes[i].boxplot(df[col].dropna(), vert=True, patch\_artist=True, boxprops=dict(facecolor='skyblue'))

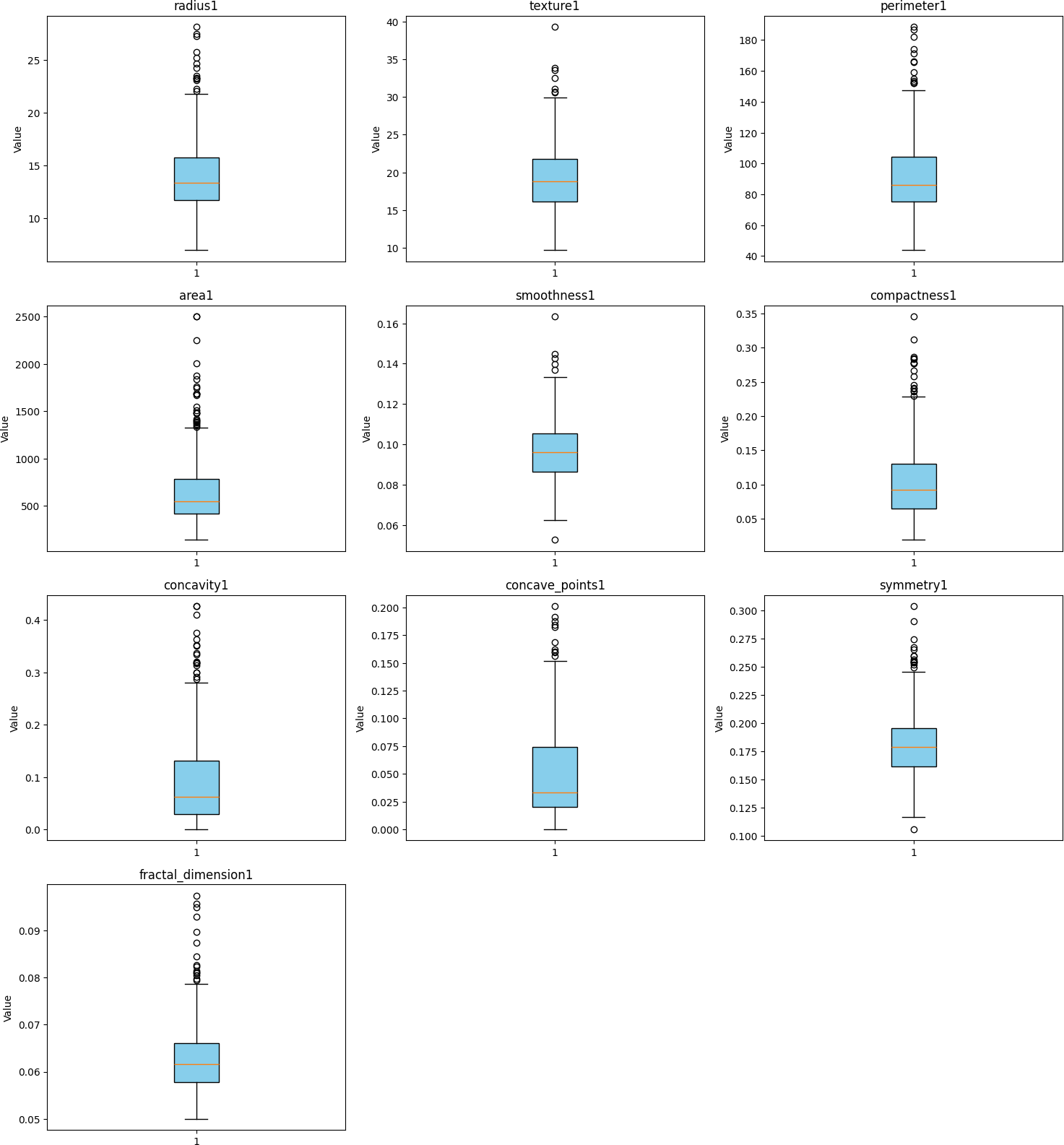
axes[i].set\_title(col)

axes[i].set\_ylabel('Value')

# Remove unused subplots

for j in range(i + 1, len(axes)): fig.delaxes(axes[j])

plt.tight\_layout() plt.show()

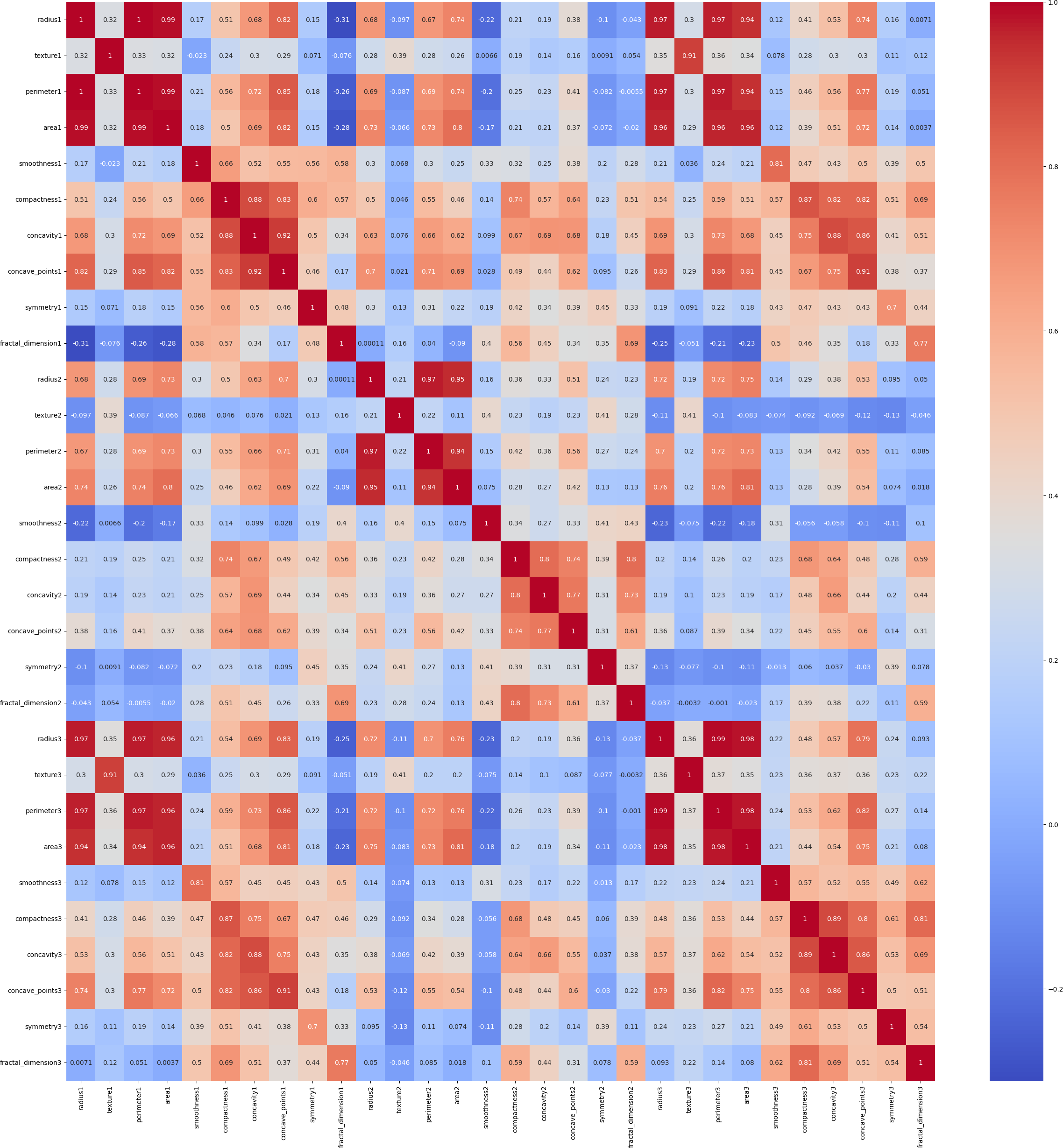
 

#Correlation Heatmap

plt.figure(figsize=(30,30))

corr = df.corr(numeric\_only=True)

sns.heatmap(corr, annot=True, cmap='coolwarm')

<Axes: >

#Standardization

X = df.drop('Diagnosis', axis=1) y = df['Diagnosis']

scaler = StandardScaler()

X\_scaled = pd.DataFrame(scaler.fit\_transform(X), columns=X.columns) df= pd.concat([X\_scaled, y], axis=1)

print(df.head())

radius1 texture1 perimeter1 area1 smoothness1 compactness1 \

0 1.097064 -2.073335 1.269934 0.984375 1.568466 3.283515

1 1.829821 -0.353632 1.685955 1.908708 -0.826962 -0.487072

2 1.579888 0.456187 1.566503 1.558884 0.942210 1.052926

3 -0.768909 0.253732 -0.592687 -0.764464 3.283553 3.402909

4 1.750297 -1.151816 1.776573 1.826229 0.280372 0.539340

concavity1 concave\_points1 symmetry1 fractal\_dimension1 ... texture3 \

0 2.652874 2.532475 2.217515 2.255747 ... -1.359293

1 -0.023846 0.548144 0.001392 -0.868652 ... -0.369203

2 1.363478 2.037231 0.939685 -0.398008 ... -0.023974

3 1.915897 1.451707 2.867383 4.910919 ... 0.133984

4 1.371011 1.428493 -0.009560 -0.562450 ... -1.466770

perimeter3 area3 smoothness3 compactness3 concavity3 \

0 2.303601 2.001237 1.307686 2.616665 2.109526

1 1.535126 1.890489 -0.375612 -0.430444 -0.146749

2 1.347475 1.456285 0.527407 1.082932 0.854974

3 -0.249939 -0.550021 3.394275 3.893397 1.989588

4 1.338539 1.220724 0.220556 -0.313395 0.613179

concave\_points3 symmetry3 fractal\_dimension3 Diagnosis

0 2.296076 2.750622 1.937015 M

1 1.087084 -0.243890 0.281190 M

2 1.955000 1.152255 0.201391 M

3 2.175786 6.046041 4.935010 M

4 0.729259 -0.868353 -0.397100 M

[5 rows x 31 columns]

from sklearn.preprocessing import LabelEncoder # Encode labels B/M → 0/1

le = LabelEncoder()

y = le.fit\_transform(y) # 'B' becomes 0, 'M' becomes 1

import numpy as np import pandas as pd

from sklearn.model\_selection import train\_test\_split, GridSearchCV, cross\_val\_score from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import (

accuracy\_score, f1\_score, classification\_report, confusion\_matrix, ConfusionMatrixDisplay, roc\_curve, auc

)

import matplotlib.pyplot as plt

# 3. Split dataset (80-20)

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.2, random\_state=42, stratify=y

)

# 4. Initialize Decision Tree

dt = DecisionTreeClassifier(random\_state=42)

# Hyperparameter grid for tuning param\_grid = {

"criterion": ["gini", "entropy"],

"max\_depth": [3, 5, 10, None],

"min\_samples\_split": [2, 5, 10],

"min\_samples\_leaf": [1, 2, 4]

}

# Use GridSearchCV with 5-Fold CV grid\_search = GridSearchCV(

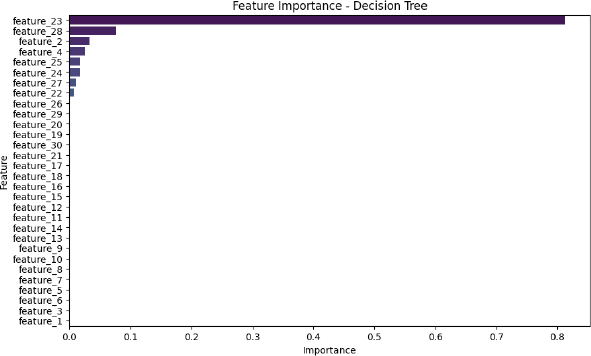
estimator=dt,

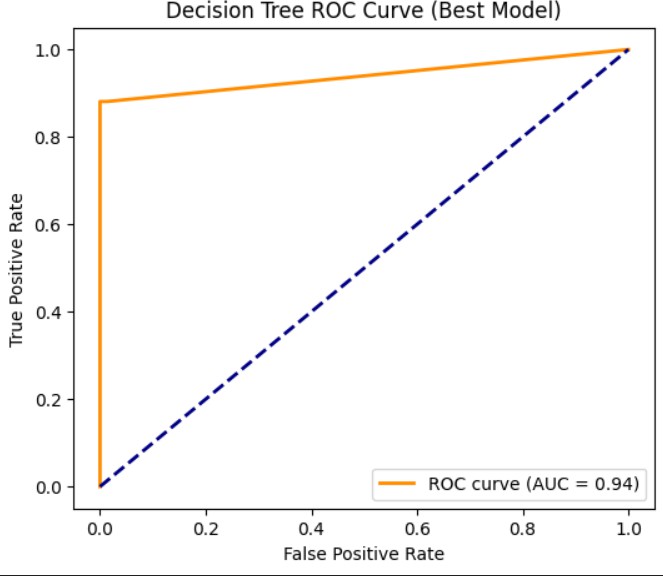
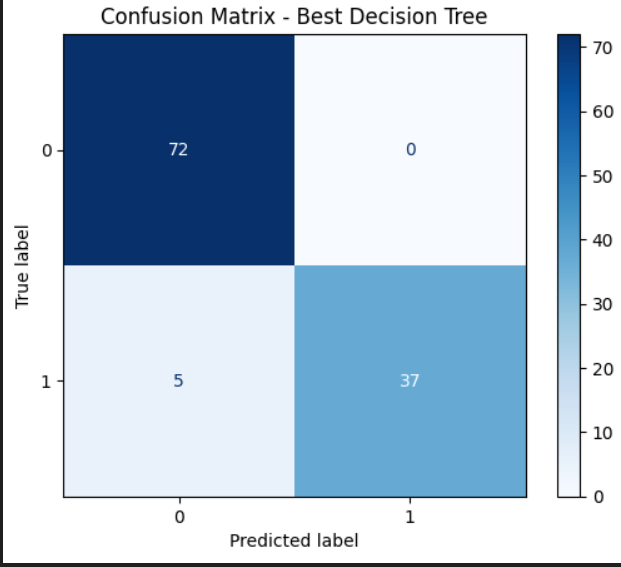
param\_grid=param\_grid, cv=5,

scoring="accuracy", n\_jobs=-1,

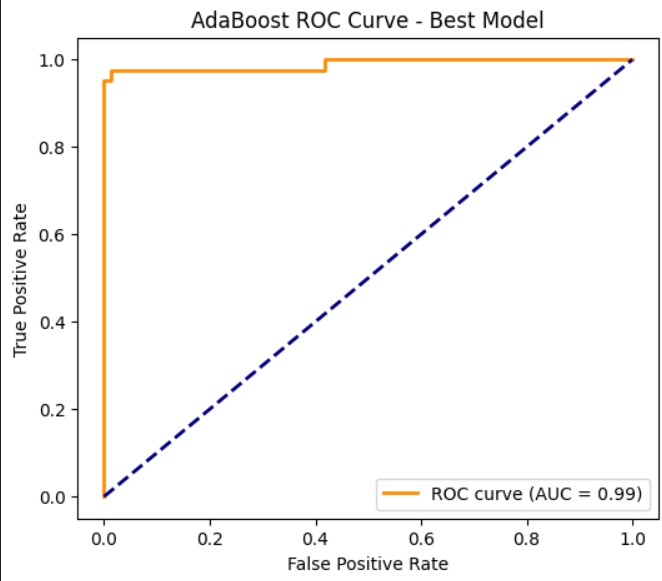
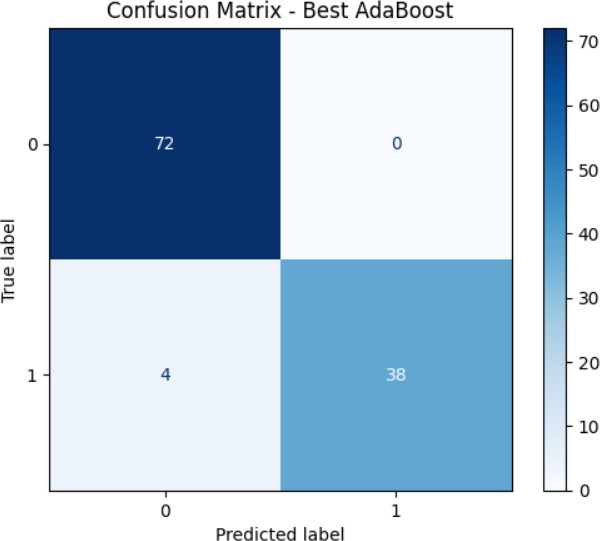
# Confusion Matrix, ROC and Feature Importance Visuals

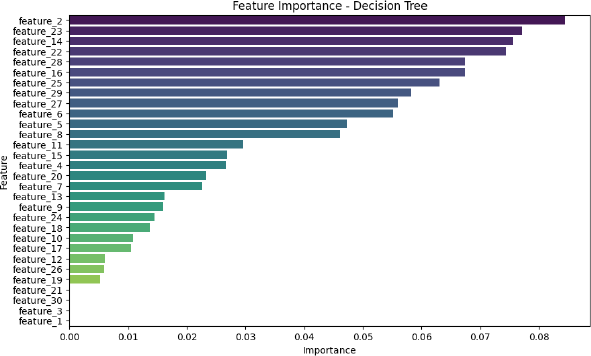
1. Decision Tree



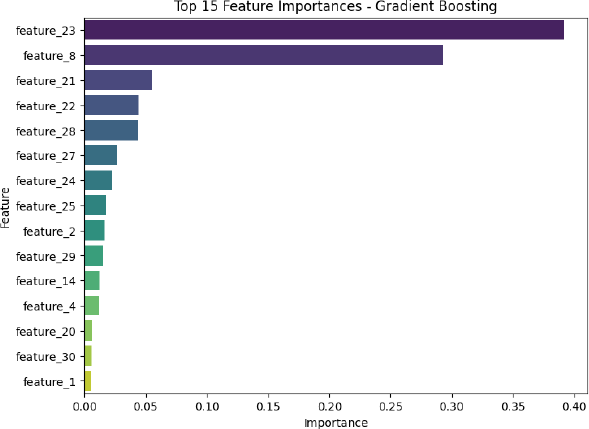


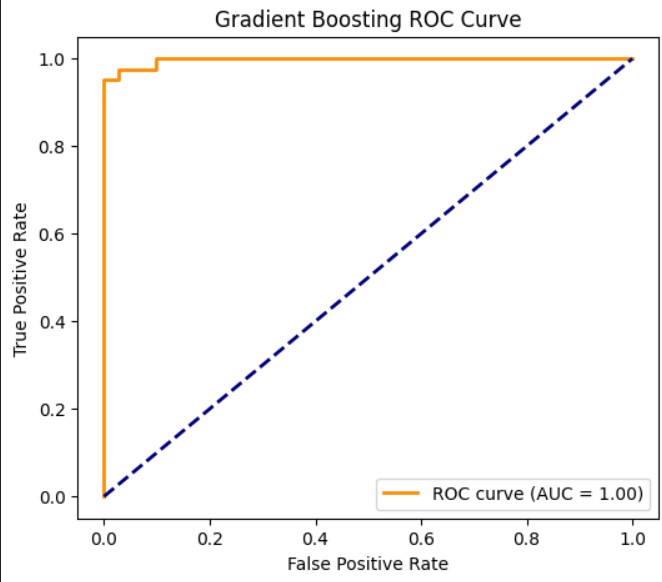
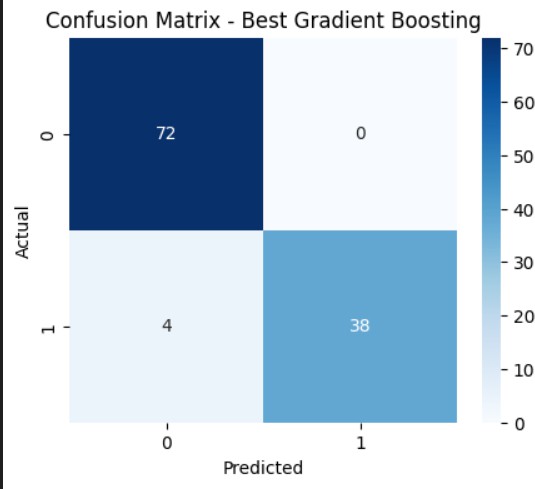
1. Ada Boost



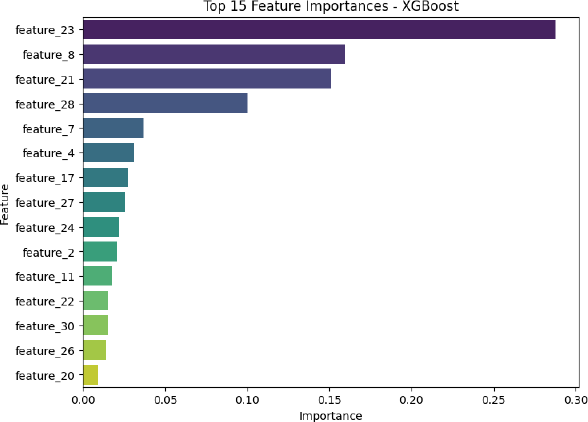
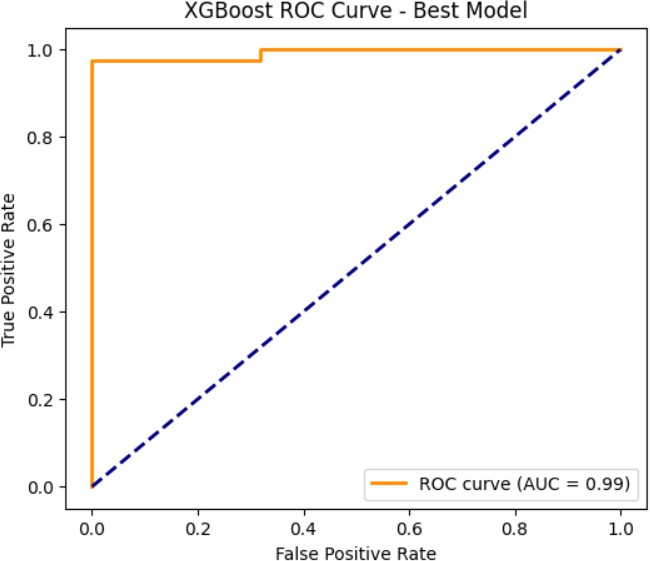
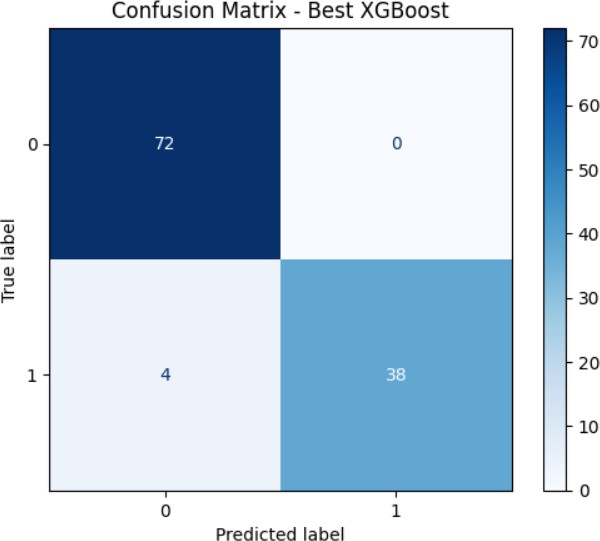


1. Gradient

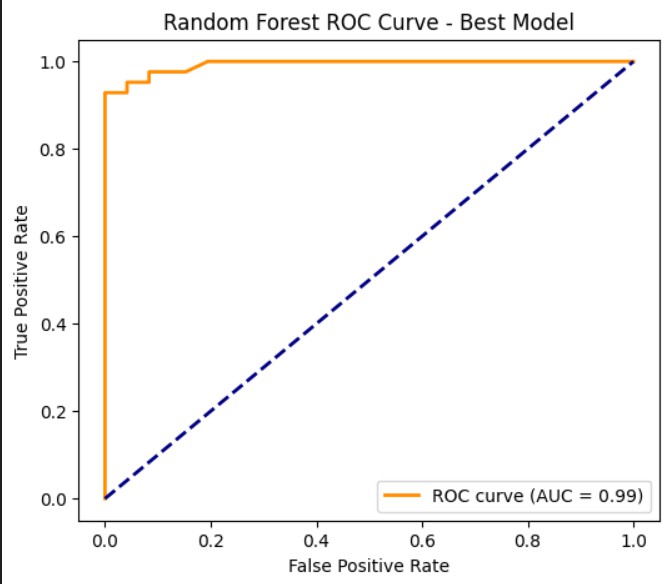
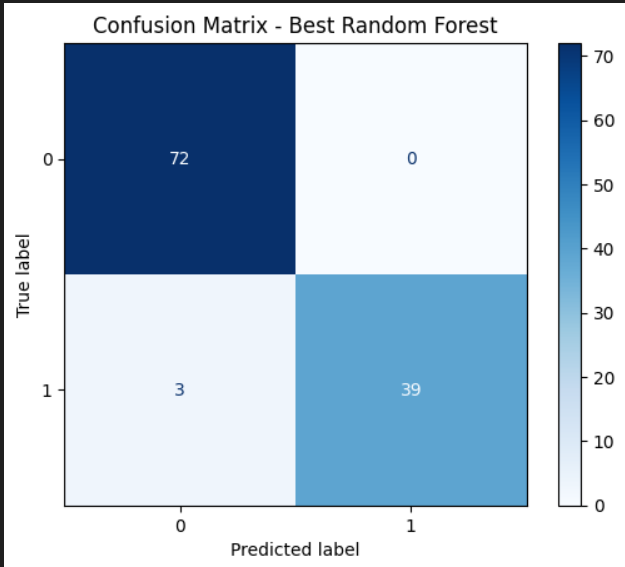


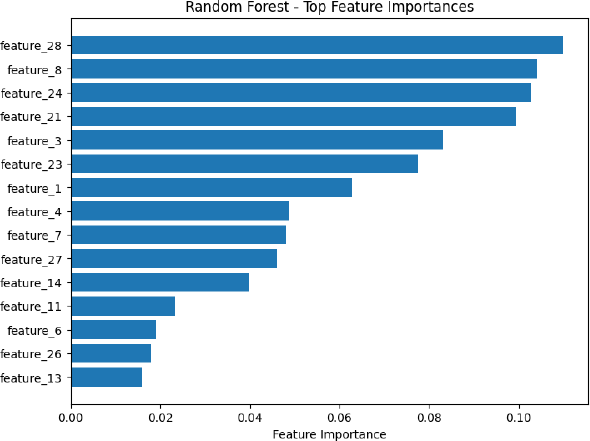


1. XG Boost

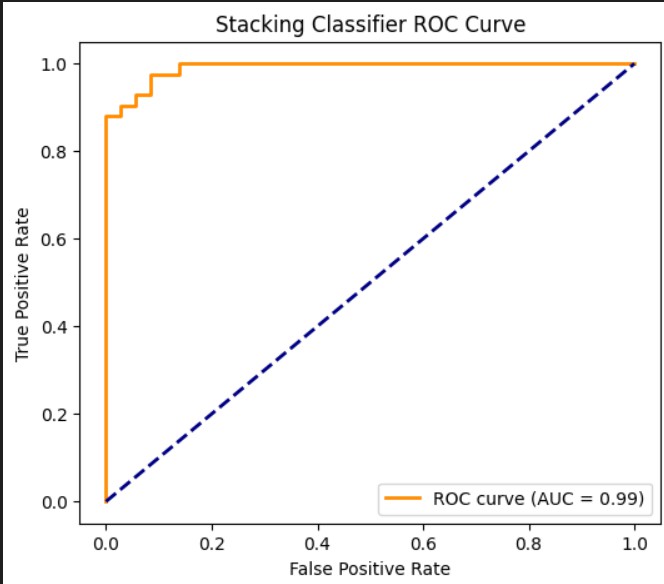
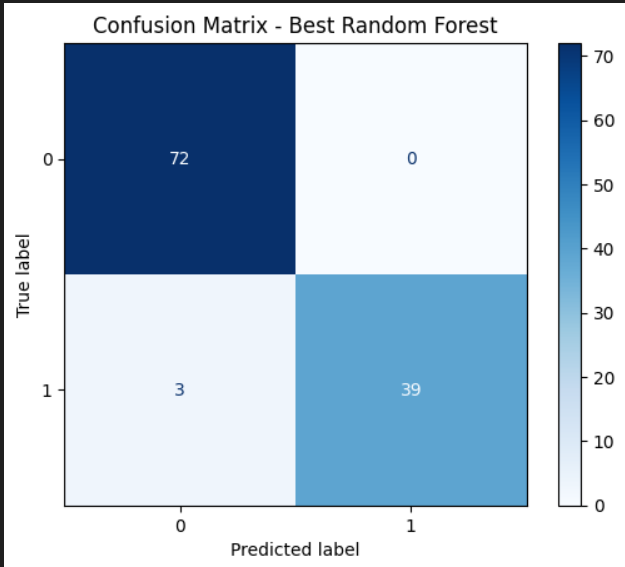


1. Random forest





1. Stacked Ensumble Mode;



# 5 fold cross validatoin results table

5-Fold Cross-Validation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Average  Accuracy |
| Decision  Tree | 0.8859 | 0.9282 | 0.9385 | 0.93859 | 0.9469 | 0.9279 |
| Stacked  Model | 0.9122 | 0.9298 | 0.9561 | 0.9561 | 0.9469 | 0.9402 |
| Gradient  Boost | 0.9298 | 0.9385 | 0.9736 | 0.9912 | 0.9734 | 0.9613 |
| XG Boost | 0.9473 | 0.9649 | 0.9912 | 0.9649 | 0.9646 | 0.9666 |
| Random  Forest | 0.92105 | 0.9385 | 0.9824 | 0.9649 | 0.9734 | 0.9560 |
| Ada Boost | 0.9649 | 0.9561 | 0.9645 | 0.9736 | 0.97345 | 0.9736 |

# Hyper parameter tuning tables:

Decision Tree

|  |  |  |  |
| --- | --- | --- | --- |
| Criterion | Max\_depth | Accuracy | F1 Score |
| Gini | 3 | 0.9035 | 0.9013 |
| entropy | 10 | 0.9561 | 0.9554 |
| Log\_loss | 5 | 0.9253 | 0.9182 |
|  |  |  |  |
|  |  |  |  |

Adaboost

|  |  |  |  |
| --- | --- | --- | --- |
| N\_estimators | Learning\_rate | Accuracy | F1 Score |
| 100 | 1 | 0.9824 | 0.9823 |
| 200 | 0 | 0.9736 | 0.9734 |
| 100 | 1 | 0.9736 | 0.9734 |
| 300 | 1 | 0.9736 | 0.97246 |

Gradient

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N\_estimator | Learning\_rate | Max\_depth | Accuracy | F1 Score |
| 200 | 0.1 | 5 | 0.9649 | 0.9553 |
| 100 | 0.2 | 3 | 0.9645 | 0.9612 |
| 200 | 0.2 | 3 | 0.9457 | 0.9612 |
| 100 | 0.2 | 3 | 0.9741 | 0.95 |

XG Boost

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N\_estimator | Learning rate | Max depth | Gamma | Accuracy | F! Score |
| 200 | 0.2 | 7 | 0.0 | 0.9758 | 0.9741 |
| 100 | 0.2 | 3 | 0.0 | 0.9714 | 0.9693 |
| 200 | 0.1 | 3 | 0.1 | 0.9714 | 0.9693 |
| 100 | 0.2 | 7 | 0.0 | 0.9714 | 0.9693 |

Random Forest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterion | Max depth | N\_Estimator | Accuracy | F1 Score |
| Gini | 10 | 50 | 0.9670 | 0.9648 |
| Log\_losss | 5 | 50 | 0.9648 | 0.9622 |
| entropy | 5 | 50 | 0.9648 | 0.9699 |
|  |  |  |  |  |

Stacked Ensemble Hyperparameter

|  |  |  |  |
| --- | --- | --- | --- |
| Base Models | Final Estimator | Accuracy | F! Score |
| SVM,NB,DT | Logistic | 0.9494 | 0.9452 |
| SVM,ND,DT | Random Forest | 0.9363 | 0.9315 |
| SVM,DT,KNN | Logistic | 0.9390 | 0.9380 |

# Observations

* Decision Tree: Accuracy ranges from 0.8947 to 0.9386 across folds. Average accuracy is 0.9227, the lowest among all models. Moderate variation indicates sensitivity to training data.
* AdaBoost: Accuracy ranges from 0.9386 to 0.9825. Average accuracy is 0.9666, significantly better than a single Decision Tree. Boosting improves stability and overall performance.
* Gradient Boosting: Fold accuracies are high, between 0.9341 and 1.0000. Average accuracy is 0.9692. Low variation shows robust learning with optimal hyperparameters.
* XGBoost: Achieves fold accuracies between 0.9341 and 1.0000. Highest average accuracy of 0.9758, indicating excellent generalization and effective hyperparameter tuning.
* Random Forest: Fold accuracies range from 0.9231 to 1.0000. Average accuracy is 0.9670, slightly lower than Gradient Boosting and XGBoost. Ensemble reduces variance compared to a single Decision Tree.
* Stacked Ensemble (SVM + Na¨ıve Bayes + Decision Tree): Accuracy per fold ranges

from 0.9121 to 0.9890. Average accuracy is 0.9429, lower than boosting methods but higher

than a single Decision Tree. Mean F1 score is 0.9382, indicating balanced performance across classes.

# Conclusions

* Boosting methods outperform individual models. AdaBoost, Gradient Boosting, and XGBoost all outperform a single Decision Tree. XGBoost achieved the highest average accuracy

(0.9758).

* Random Forest improves over a single Decision Tree by aggregating multiple trees, reducing overfitting and variance. Its performance (0.9670) is competitive with boosting methods.
* Stacked ensemble provides moderate gains. Combining SVM, Na¨ıve Bayes, and Decision Tree yields better performance than a single model but is still below top boosting algorithms. Including stronger base learners may improve performance.
* Hyperparameter tuning is critical. XGBoost and Gradient Boosting achieved their best performance using carefully tuned parameters. Small changes in parameters like learning rate,

depth, and number of estimators significantly impact accuracy.

* Recommendation: For maximum predictive accuracy, XGBoost with chosen hyperparameters is recommended. For interpretable models with slightly lower accuracy, Random Forest or AdaBoost are good choices. Stacked ensembles can be explored further by including stronger base learners.