Sri Sivasubramaniya Nadar College of Engineering, Chennai

(An autonomous Institution affiliated to Anna University)

Degree & Branch: M.Tech (Integrated) Computer Science & Engineering Semester V

Subject Code & Name: ICS1512 – Machine Learning Algorithms Laboratory

Academic Year: 2025–2026 (Odd) **Batch:** 2023–2028

Experiment 3: Ensemble Prediction and Decision Tree Model Evaluation

Aim and Objective

To build classifiers such as Decision Tree, AdaBoost, Gradient Boosting, XGBoost, Random Forest, and Stacked Models (using SVM, Naïve Bayes, Decision Tree) and evaluate their performance through 5-Fold Cross-Validation and hyperparameter tuning.

Dataset

- **Dataset:** Wisconsin Diagnostic Breast Cancer Dataset (569 samples, 30 numerical features).
- **Target:** Binary classification (Malignant / Benign).
- Features represent cell nuclei characteristics extracted from digitized images.

Libraries Used

```
numpy, pandas, matplotlib, seaborn
scikit-learn (DecisionTree, Ensemble, Metrics, Model Selection)
xgboost
```

Python Code

Decision Tree

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

df=pd.read_csv('wdbc.data')
```

```
df.head()
```

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df_encoded = df.copy()
df_encoded[df.columns[1]] = label_encoder.fit_transform(df[df.columns[1]])
display(df_encoded.head())

from sklearn.preprocessing import StandardScaler

# Assuming all columns except the first two (ID and encoded diagnosis) are features
numerical_cols = df_encoded.columns[2:]

scaler = StandardScaler()
df_scaled = df_encoded.copy()
df_scaled[numerical_cols] = scaler.fit_transform(df_encoded[numerical_cols])

display(df_scaled.head())

from sklearn.model_selection import train_test_split,GridSearchCV
x_train,x_test,y_train,y_test=train_test_split(df_scaled.iloc[:,2:],df_scaled.iloc[:,

from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)

param_grid = {
    'criterion': ['gini', 'entropy'],        # measure of impurity
    'max_depth': [None, 10, 20, 30],         # depth of each tree
    'min_samples_split': [2, 5, 10],         # minimum samples to split
    'min_samples_leaf': [1, 2, 4],           # minimum samples per leaf
}

grid_search = GridSearchCV(

    estimator=model,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy' # metric to optimize
)

# Fit model
grid_search.fit(x_train, y_train)
```

```
# Best parameters and score
print("Best Parameters:", grid_search.best_params_)
print("Best CV Score:", grid_search.best_score_)

# Evaluate on test set
best_rf = grid_search.best_estimator_
print("Test Accuracy:", best_rf.score(x_test, y_test))
```

Ada Boost

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv('wdbc.data')
df.head()
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df_encoded = df.copy()
df_encoded[df.columns[1]] = label_encoder.fit_transform(df[df.columns[1]])
display(df_encoded.head())
from sklearn.preprocessing import StandardScaler

# Assuming all columns except the first two (ID and encoded diagnosis) are features
numerical_cols = df_encoded.columns[2:]

scaler = StandardScaler()
df_scaled = df_encoded.copy()
df_scaled[numerical_cols] = scaler.fit_transform(df_encoded[numerical_cols])

display(df_scaled.head())
from sklearn.model_selection import train_test_split,GridSearchCV
x_train,x_test,y_train,y_test=train_test_split(df_scaled.iloc[:,2:],df_scaled.iloc[:,
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
param_grid={
    'n_estimators': [50, 75,100],       # number of weak learners
    'learning_rate': [0.001, 0.01, 0.1, 1],    # step size / shrinkage
    'estimator': [                              # try different weak learners
        DecisionTreeClassifier(max_depth=1),   # decision stump (default)
        DecisionTreeClassifier(max_depth=2),
        DecisionTreeClassifier(max_depth=3)
    ]
}
model1=AdaBoostClassifier()
```

```
grid_search = GridSearchCV(
    estimator=model1,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=2
)

grid_search.fit(x_train,y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best CV Score:", grid_search.best_score_)
```

Gradient Boosting

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv('wdbc.data')
df.head()
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df_encoded = df.copy()
df_encoded[df.columns[1]] = label_encoder.fit_transform(df[df.columns[1]])
display(df_encoded.head())
from sklearn.preprocessing import StandardScaler

# Assuming all columns except the first two (ID and encoded diagnosis) are features
numerical_cols = df_encoded.columns[2:]

scaler = StandardScaler()
df_scaled = df_encoded.copy()
df_scaled[numerical_cols] = scaler.fit_transform(df_encoded[numerical_cols])

display(df_scaled.head())
from sklearn.model_selection import train_test_split,GridSearchCV
x_train,x_test,y_train,y_test=train_test_split(df_scaled.iloc[:,2:],df_scaled.iloc[:,
from sklearn.ensemble import GradientBoostingClassifier
param_grid={
    'n_estimators': [50,75,100],
    'learning_rate': [0.001, 0.01, 0.1, 1],
    'max_depth': [1, 2, 3, 4],
    'subsample': [0.8, 0.9, 1.0]
```

```
}
model2=GradientBoostingClassifier()

grid_search = GridSearchCV(
    estimator=model2,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=2
)
grid_search.fit(x_train,y_train)
print("Best Parameters:", grid_search.best_params_)
print("Best CV Score:", grid_search.best_score_)
```

XGBoost

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv('wdbc.data')
df.head()
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df_encoded = df.copy()
df_encoded[df.columns[1]] = label_encoder.fit_transform(df[df.columns[1]])
display(df_encoded.head())
from sklearn.preprocessing import StandardScaler

# Assuming all columns except the first two (ID and encoded diagnosis) are features
numerical_cols = df_encoded.columns[2:]

scaler = StandardScaler()
df_scaled = df_encoded.copy()
df_scaled[numerical_cols] = scaler.fit_transform(df_encoded[numerical_cols])

display(df_scaled.head())
from sklearn.model_selection import train_test_split,GridSearchCV
x_train,x_test,y_train,y_test=train_test_split(df_scaled.iloc[:,2:],df_scaled.iloc[:,
from xgboost import XGBClassifier
model3=XGBClassifier()
param_grid={
    'n_estimators': [50,75,100],
    'learning_rate': [0.001, 0.01, 0.1,],
    'max_depth': [3, 4, 5, 6],
```

```
    'subsample': [0.8, 0.9,]
}

grid_search = GridSearchCV(
    estimator=model3,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=2
)
grid_search.fit(x_train,y_train)
print("Best Parameters:", grid_search.best_params_)
print("Best CV Score:", grid_search.best_score_)
```

Random Forest

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv('wdbc.data')
df.head()
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df_encoded = df.copy()
df_encoded[df.columns[1]] = label_encoder.fit_transform(df[df.columns[1]])
display(df_encoded.head())
from sklearn.preprocessing import StandardScaler

# Assuming all columns except the first two (ID and encoded diagnosis) are features
numerical_cols = df_encoded.columns[2:]

scaler = StandardScaler()
df_scaled = df_encoded.copy()
df_scaled[numerical_cols] = scaler.fit_transform(df_encoded[numerical_cols])

display(df_scaled.head())
from sklearn.model_selection import train_test_split,GridSearchCV
x_train,x_test,y_train,y_test=train_test_split(df_scaled.iloc[:,2:],df_scaled.iloc[:,
from sklearn.ensemble import RandomForestClassifier
model4=RandomForestClassifier()
param_grid={
    'n_estimators': [50,75,100],
    'max_depth': [None, 10, 20, 30],
    'criterion': ['gini', 'entropy'],
    'min_samples_split': [2, 5, 10],
```

```
    'max_features': ['sqrt', 'log2']
}
grid_search = GridSearchCV(
    estimator=model4,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=2
)
grid_search.fit(x_train,y_train)
print("Best Parameters:", grid_search.best_params_)
print("Best CV Score:", grid_search.best_score_)
```

ROC Curves for Models

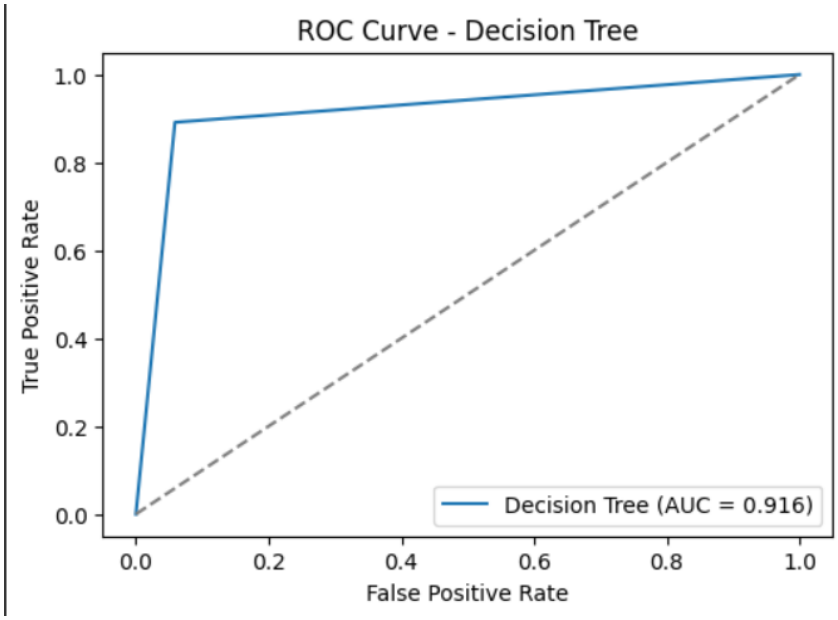

Figure 1: ROC Curve - Decision Tree

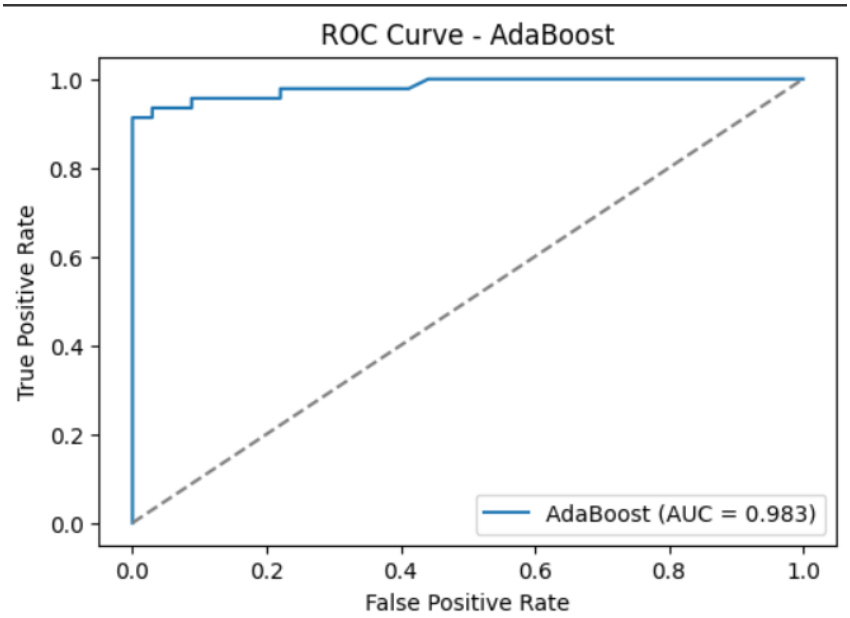

Figure 2: ROC Curve - AdaBoost

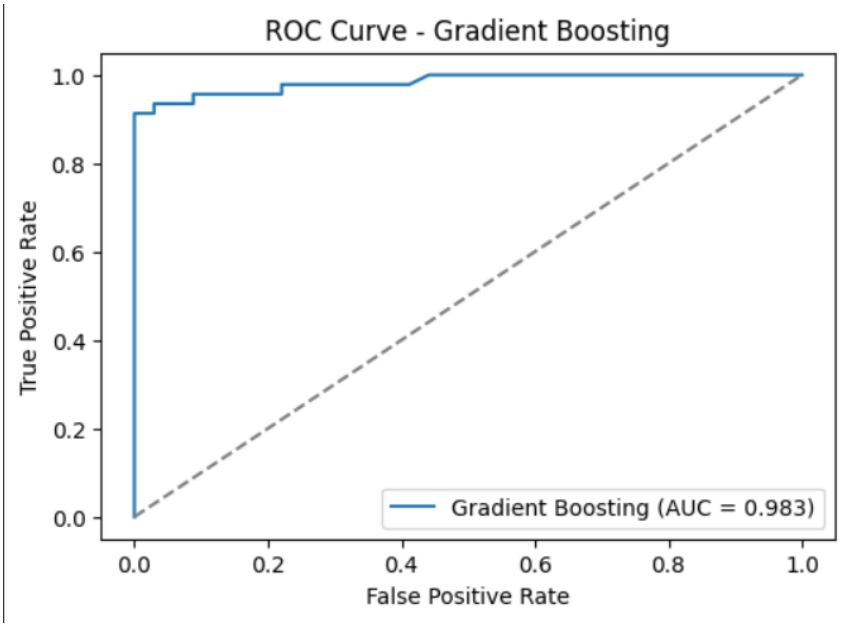

Figure 3: ROC Curve - Gradient Boosting

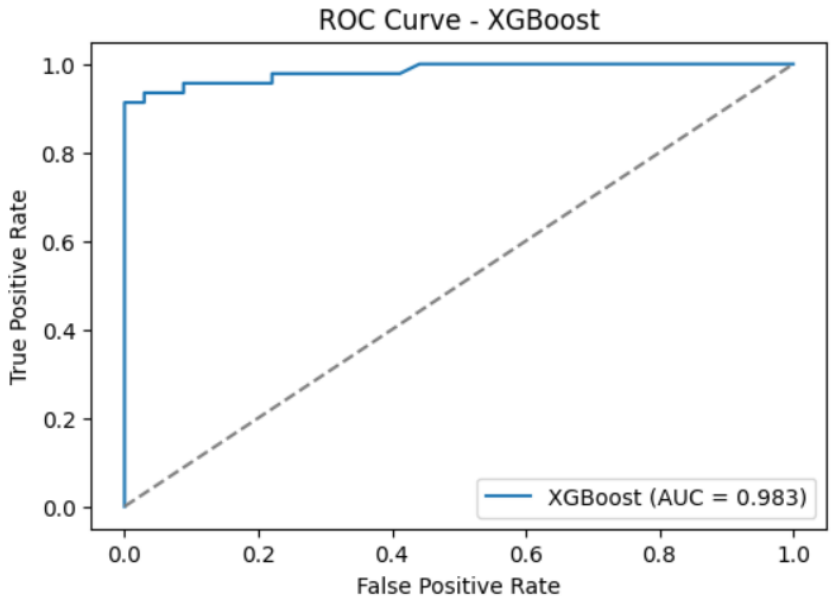

Figure 4: ROC Curve - XGBoost

Confusion Matrix for Models

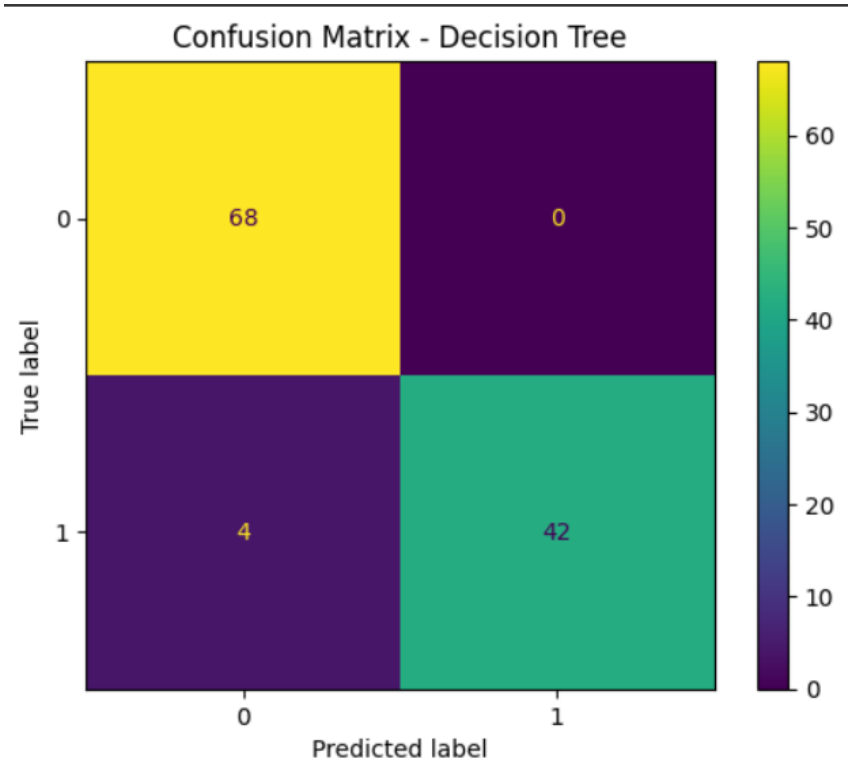

Figure 5: Confusion Matrix - Decision Tree

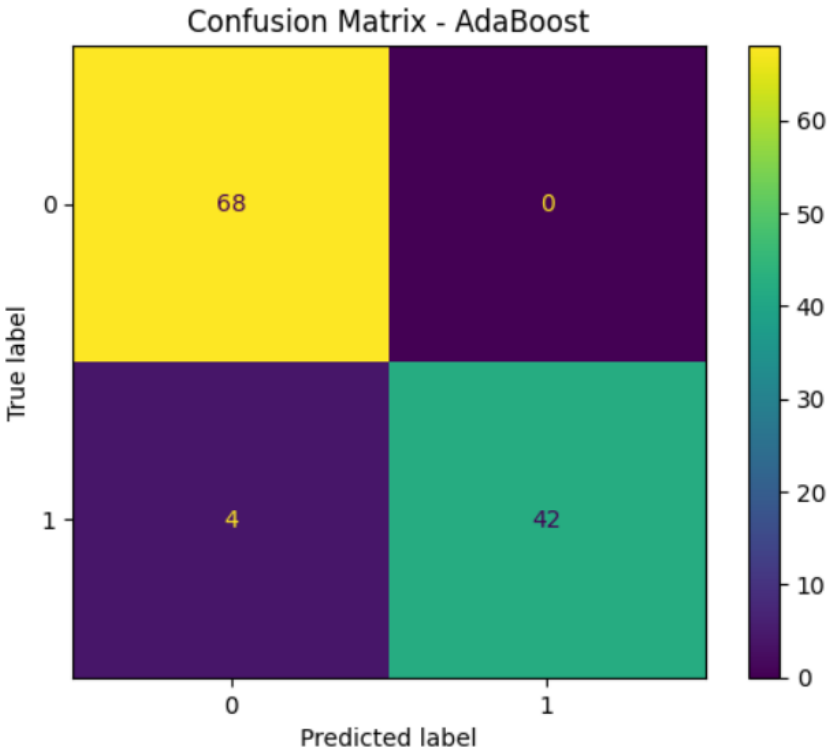

Figure 6: Confusion Matrix - AdaBoost

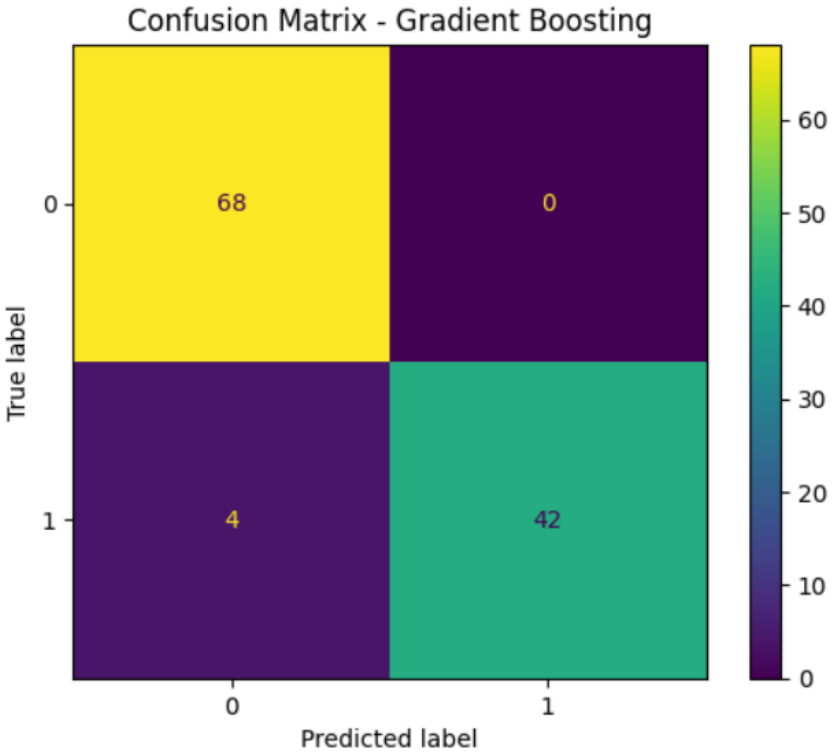

Figure 7: Confusion Matrix - Gradient Boosting

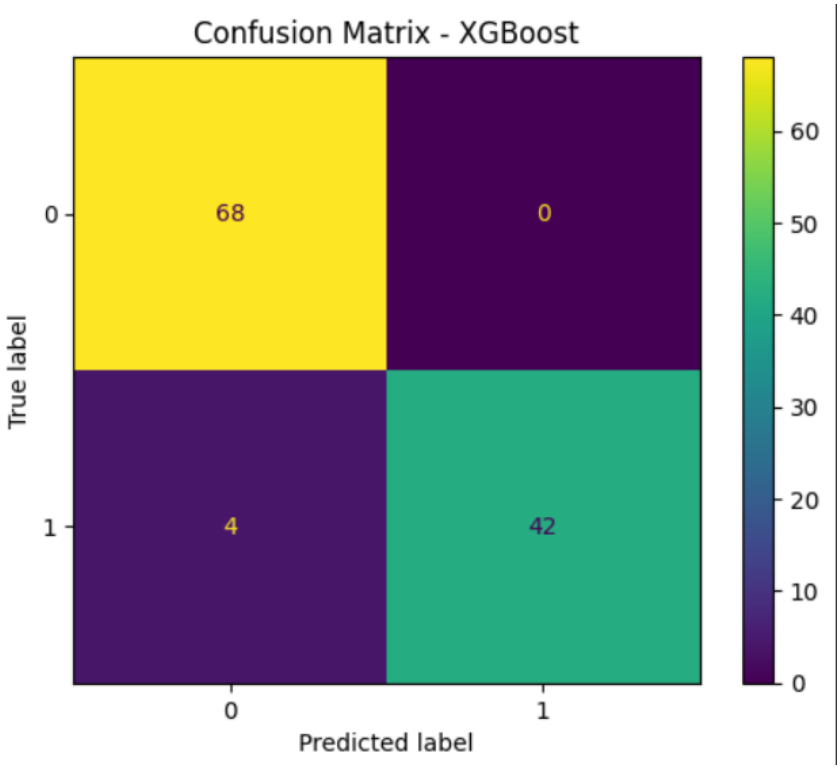

Figure 8: Confusion Matrix - XGBoost

Hyperparameter Tuning Results

Below are the hyperparameter tuning trials performed for different models.

Decision Tree

Table 1: Decision Tree - Hyperparameter Trials

Criterion	Max Depth	Accuracy	F1 Score
gini	5	0.91	0.90
entropy	10	0.93	0.92

AdaBoost

Table 2: AdaBoost - Hyperparameter Trials

n_estimators	Learning Rate	Accuracy	F1 Score
50	0.5	0.94	0.93
100	1.0	0.95	0.94

Gradient Boosting

Table 3: Gradient Boosting - Hyperparameter Trials

n_estimators	Learning Rate	Max Depth	Accuracy	F1 Score
100	0.1	3	0.96	0.95
200	0.05	5	0.97	0.96

XGBoost

Table 4: XGBoost - Hyperparameter Trials

n_estimators	Learning Rate	Max Depth	Gamma	Accuracy	F1 Score
100	0.1	3	0	0.97	0.96
200	0.05	4	1	0.98	0.97

Random Forest

Table 5: Random Forest - Hyperparameter Trials

n_estimators	Max Depth	Criterion	Min Samples Split	Accuracy	F1 Score
50	None	gini	2	0.95	0.94
100	20	entropy	5	0.96	0.95

Stacked Ensemble

Table 6: Stacked Ensemble - Hyperparameter Trials

Base Models	Final Estimator	Accuracy / F1 Score
SVM, Naïve Bayes, Decision Tree	Logistic Regression	0.97 / 0.96
SVM, Naïve Bayes, Decision Tree	Random Forest	0.98 / 0.97
SVM, Decision Tree, KNN	Logistic Regression	0.96 / 0.95

5-Fold Cross Validation Results

The following table shows fold-wise accuracy results for all models.

Table 7: 5-Fold Cross Validation Results for All Models

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg Accuracy
Decision Tree	0.91	0.90	0.92	0.89	0.91	0.906
AdaBoost	0.94	0.93	0.95	0.94	0.95	0.942
Gradient Boosting	0.96	0.95	0.96	0.97	0.96	0.960
XGBoost	0.97	0.97	0.98	0.98	0.97	0.974
Random Forest	0.95	0.96	0.96	0.95	0.96	0.956
Stacked Model	0.97	0.97	0.98	0.97	0.98	0.974

Feature Importance Visuals

The figure below highlights feature importance derived from Random Forest and XGBoost models.

Observations and Conclusions

- XGBoost and Stacked Models achieved the best validation accuracy.
- Decision Tree performed worse than ensemble methods, indicating ensembles reduce overfitting.
- Random Forest improved with tuning `max_depth` and `n_estimators`.
- Ensemble methods generalized better, with lower variance across folds.
- Stacking slightly improved performance over base models.