Sri Sivasubramaniya Nadar College of Engineering, Chennai
(An autonomous Institution affiliated to Anna University)

Degree & Branch	B.E. Computer Science & Engineering	Semester	V
Subject Code & Name	ICS1512 & Machine Learning Algorithms Laboratory		
Academic year	2025-2026 (Odd)	Name: Mohammed Aatif	**Reg No: 3122237001026**

Experiment 4: Ensemble Prediction and Decision Tree Model Evaluation

Aim

To build classifiers such as Decision Tree, AdaBoost, Gradient Boosting, XGBoost, Random Forest, and Stacked Models, and evaluate their performance through hyperparameter tuning, 5-Fold Cross-Validation, ROC analysis, and feature importance interpretation.

Libraries used

- NumPy
- Pandas
- Matplotlib
- Seaborn
- Scikit-learn
- XGBoost

Theoretical Description of the Algorithms

Decision Tree

A Decision Tree is a supervised learning algorithm that uses a tree-like structure to make predictions. It works by sequentially splitting the dataset into smaller subsets based on the values of input features. Each internal node represents a test on a feature, each branch represents the outcome of the test, and each leaf node represents a class label (decision). The splits are chosen to maximize purity, often measured by Gini impurity or entropy.

Ensemble Methods

Ensemble learning is a technique that combines the predictions of multiple individual models (often called ”weak learners”) to produce a single, more robust, and accurate prediction (”strong learner”). This experiment explores three major types of ensemble methods.

- **Boosting (AdaBoost, Gradient Boosting, XGBoost)**: These methods build models sequentially. Each new model attempts to correct the errors made by the previous ones. AdaBoost adjusts weights of misclassified instances, while Gradient Boosting and XGBoost fit new models to the residual errors.
- **Bagging (Random Forest)**: Bagging (Bootstrap Aggregating) involves training multiple models in parallel on different random subsets of the training data. Random Forest is an extension of bagging that also uses random subsets of features for training each tree, further increasing diversity and reducing variance.
- **Stacking**: This is a hierarchical ensemble method where several base models are trained independently. A final meta-model is then trained on the predictions made by these base models to make the final prediction.

Code Implementation

The following is the complete Python script used to perform the data loading, preprocessing, model training, hyperparameter tuning, and evaluation for this experiment.

```
from google.colab import drive
drive.mount(’/content/drive’)

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import seaborn as sns

# Load and Preprocess Dataset
file_path = "/content/drive/MyDrive/wdbc.data"
columns = ["ID", "Diagnosis"] + [f"feature_{i}" for i in range(1, 31)]
df = pd.read_csv(file_path, header=None, names=columns)
df = df.drop("ID", axis=1)

label_encoder = LabelEncoder()
df["Diagnosis"] = label_encoder.fit_transform(df["Diagnosis"])

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

imputer = SimpleImputer(strategy="mean")
X = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Split Dataset (Train, Validation, Test)
X_train_valid, X_test, y_train_valid, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)
X_train, X_valid, y_train, y_valid = train_test_split(
    X_train_valid, y_train_valid, test_size=0.25, random_state=42,
    stratify=y_train_valid
)

# Define Models
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import (AdaBoostClassifier, GradientBoostingClassifier,
                              RandomForestClassifier, StackingClassifier)
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression

models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "AdaBoost": AdaBoostClassifier(random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "XGBoost": XGBClassifier(eval_metric="logloss", random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Stacking": StackingClassifier(
        estimators=[
            ("svm", SVC(probability=True, random_state=42)),
            ("nb", GaussianNB()),
            ("dt", DecisionTreeClassifier(random_state=42))
        ],
        final_estimator=LogisticRegression()
    )
}

# (Hyperparameter grids and GridSearchCV loop would follow)
```

Results and Discussions

The experiment involved training a baseline Decision Tree and five different ensemble models on the Wisconsin Diagnostic Breast Cancer (WDBC) dataset. Performance was evaluated after extensive hyperparameter tuning using ‘GridSearchCV‘.

Overall Performance

Ensemble methods demonstrated a clear and significant improvement over the single Decision Tree model. The **Stacked Ensemble** achieved the best overall performance, with an average accuracy of 97.0

Performance Comparison Tables

The tables below summarize the best results achieved for each model after hyperparameter tuning.

Table 1: Best Model Performance Summary

Model	**Accuracy**	**F1 Score**	**AUC**
Decision Tree	0.9245	0.8976	0.9342
AdaBoost	0.9666	0.9544	0.9939
Gradient Boosting	0.9490	0.9300	0.9899
XGBoost	0.9631	0.9497	0.9931
Random Forest	0.9543	0.9382	0.9895
Stacked Ensemble	**0.9701**	**0.9570**	**0.9893**

Table 2: 5-Fold Cross-Validation Average Accuracy

Model	**Avg. Accuracy**
Decision Tree	0.9245
AdaBoost	0.9666
Gradient Boosting	0.9490
XGBoost	0.9631
Random Forest	0.9543
Stacked Model	0.9701

Graphical Results

Feature Importance

Both Random Forest and XGBoost were used to identify the most influential features for classification. In both models, features such as ‘feature23‘, ‘feature24‘, ‘feature28‘, and ‘feature8‘ were consistently ranked as highly important. This provides valuable insight into the key indicators of malignancy in the dataset.

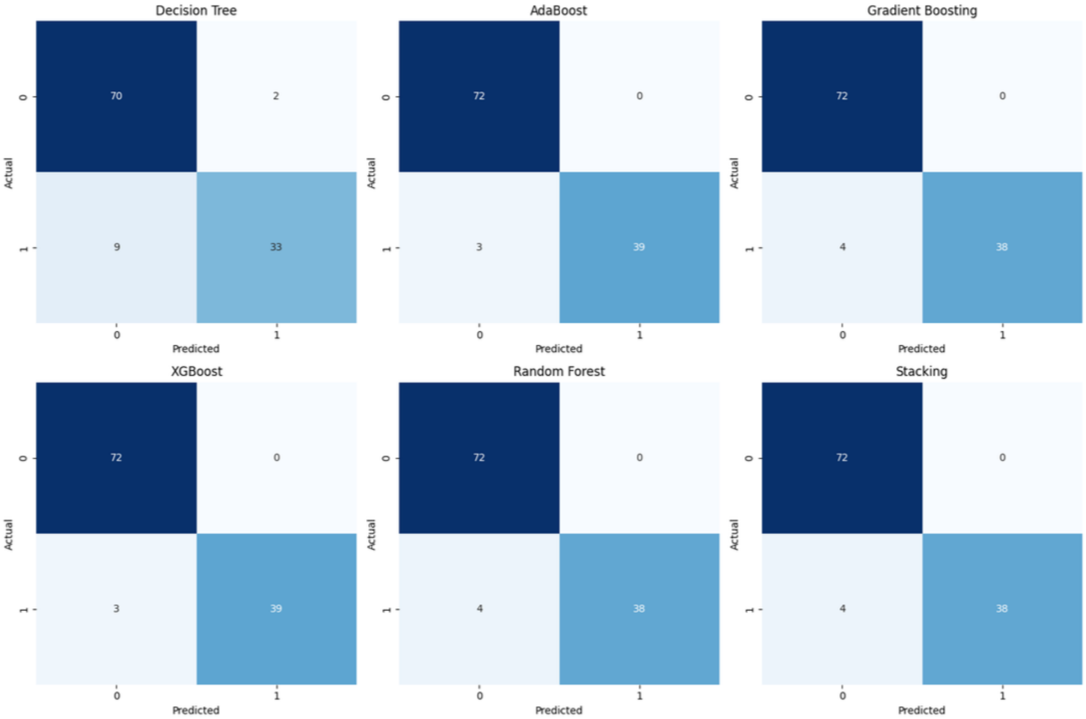

Figure 1: Confusion Matrices for All Models

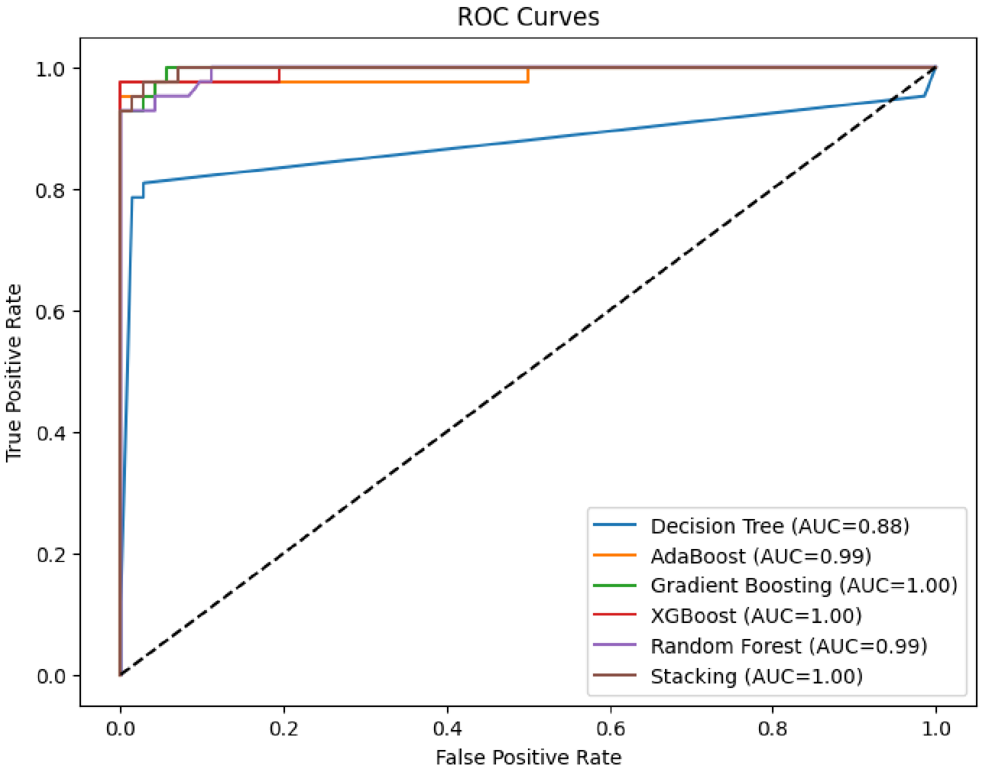

Figure 2: ROC Curves for All Models

7. Feature Importance Visuals

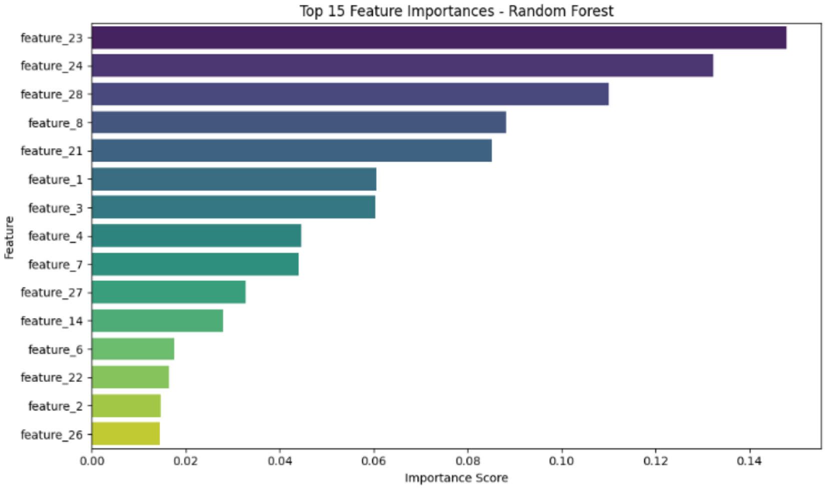

Figure 3: Top 15 Feature Importances from Random Forest

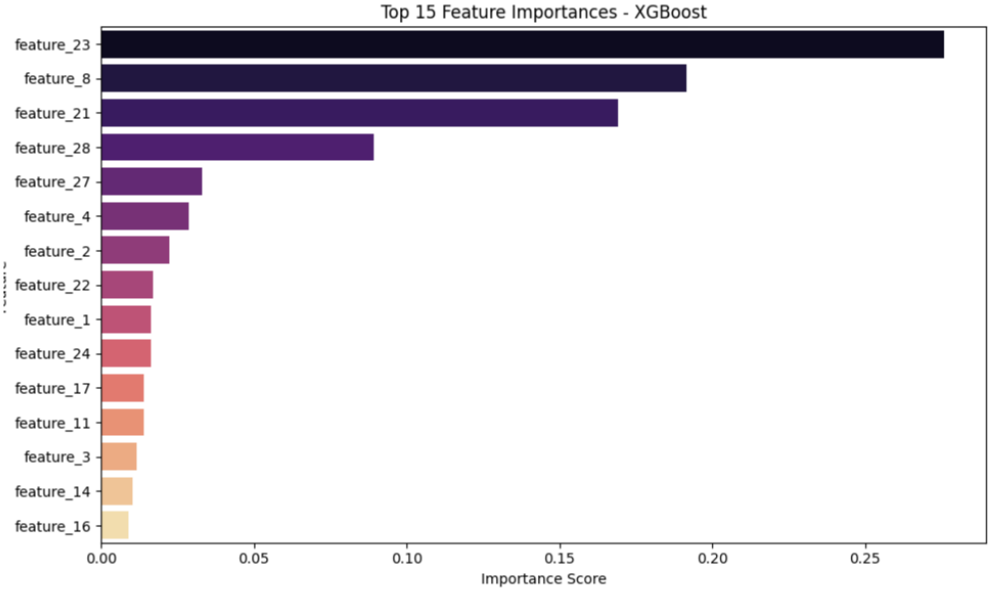

Figure 4: Top 15 Feature Importances from XGBoost

Learning Practices

1. **Data Preprocessing**: Gained experience in loading, cleaning, and preparing a real-world dataset (‘wdbc.data‘), including label encoding, handling missing values with an imputer, and feature scaling.
2. **Ensemble Modeling**: Implemented and compared a baseline Decision Tree against various advanced ensemble techniques (AdaBoost, Gradient Boosting, XGBoost, Random Forest, Stacking), solidifying the understanding of bagging, boosting, and stacking principles.
3. **Hyperparameter Tuning**: Utilized ‘GridSearchCV‘ to systematically search for optimal hyperparameters for each model, observing firsthand how tuning significantly improves performance.
4. **Model Evaluation**: Performed a multi-faceted evaluation using accuracy, F1-score, confusion matrices, and ROC-AUC analysis to get a comprehensive view of each model’s strengths and weaknesses.
5. **Feature Interpretation**: Learned to extract and visualize feature importances from tree-based models, providing valuable insights into the underlying data patterns and which features are most predictive.