Date: 27-08-2025 **Name:** Anusha K N
Experiment: 4 **Roll No:** 3122237001005

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)	Batch:2023-2028	**Due date:**

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 (SVM + Naïve Bayes + Decision Tree) and evaluate their performance through 5-Fold Cross-Validation and hyperparameter tuning.

Objectives

- Perform preprocessing on the Breast Cancer Wisconsin Dataset.
- Train and tune multiple ensemble classifiers.
- Evaluate models using Accuracy, Precision, Recall, F1 Score, ROC Curve, and Confusion Matrix.
- Compare base Decision Tree vs Ensemble methods.

Libraries Used

- `numpy`, `pandas` – data handling
- `matplotlib`, `seaborn` – visualization
- `scikit-learn` – Decision Tree, AdaBoost, Gradient Boosting, Random Forest, evaluation metrics
- `xgboost` – XGBoost classifier

Theoretical Background

- **Decision Tree:** Recursive partitioning based on impurity (Gini/Entropy).

$$\text{Entropy}(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

- **AdaBoost:** Weighted majority voting by re-weighting samples.
- **Gradient Boosting:** Sequential learners minimize residuals using gradient descent.
- **XGBoost:** Regularized boosting with optimized gradient computation.
- **Random Forest:** Bagging with feature randomness.
- **Stacking:** Combines heterogeneous models with a meta-learner.

Code Implementation

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score, classification_report, confusio

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier, RandomFo
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
import xgboost as xgb

# ------------------------------
# Load and preprocess dataset
# ------------------------------
data = load_breast_cancer()
X, y = data.data, data.target

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

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Helper function to evaluate model
def evaluate_model(model, X_test, y_test, name="Model"):
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
```

```
    print(f"{name} -> Accuracy: {acc:.4f}, F1 Score: {f1:.4f}")
    print(classification_report(y_test, y_pred))
    cm = confusion_matrix(y_test, y_pred)

    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.title(f"Confusion Matrix - {name}")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

    # ROC Curve
    y_prob = model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    plt.plot(fpr, tpr, label=f"{name} (AUC={auc(fpr,tpr):.2f})")
    plt.plot([0, 1], [0, 1], "k--")
    plt.legend()
    plt.title(f"ROC Curve - {name}")
    plt.show()
    return acc, f1

# --------------------------------
# Decision Tree
# --------------------------------
dt_params = {"criterion": ["gini", "entropy"], "max_depth": [3, 5, 7, None]}
dt = GridSearchCV(DecisionTreeClassifier(random_state=42), dt_params, cv=5)
dt.fit(X_train, y_train)
print("Best DT Params:", dt.best_params_)
dt_acc, dt_f1 = evaluate_model(dt.best_estimator_, X_test, y_test, "Decision Tree")

# --------------------------------
# AdaBoost
# --------------------------------
ada_params = {"n_estimators": [50, 100], "learning_rate": [0.01, 0.1, 1]}
ada = GridSearchCV(AdaBoostClassifier(random_state=42), ada_params, cv=5)
ada.fit(X_train, y_train)
print("Best Ada Params:", ada.best_params_)
ada_acc, ada_f1 = evaluate_model(ada.best_estimator_, X_test, y_test, "AdaBoost")

# --------------------------------
# Gradient Boosting
# --------------------------------
gb_params = {"n_estimators": [50, 100], "learning_rate": [0.01, 0.1, 0.5], "max_depth
gb = GridSearchCV(GradientBoostingClassifier(random_state=42), gb_params, cv=5)
gb.fit(X_train, y_train)
print("Best GB Params:", gb.best_params_)
gb_acc, gb_f1 = evaluate_model(gb.best_estimator_, X_test, y_test, "Gradient Boosting

# --------------------------------
```

```
# XGBoost
# ------------------------------
xgb_params = {"n_estimators": [50, 100], "learning_rate": [0.01, 0.1], "max_depth": [
xgb_clf = GridSearchCV(
    xgb.XGBClassifier(
        objective="binary:logistic",
        eval_metric="logloss",
        random_state=42
    ),
    xgb_params,
    cv=5
)
xgb_clf.fit(X_train, y_train)
print("Best XGB Params:", xgb_clf.best_params_)
xgb_acc, xgb_f1 = evaluate_model(xgb_clf.best_estimator_, X_test, y_test, "XGBoost")

# ------------------------------
# Random Forest
# ------------------------------
rf_params = {"n_estimators": [50, 100], "max_depth": [3, 5, None], "criterion": ["gin
rf = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=5)
rf.fit(X_train, y_train)
print("Best RF Params:", rf.best_params_)
rf_acc, rf_f1 = evaluate_model(rf.best_estimator_, X_test, y_test, "Random Forest")

# ------------------------------
# Stacked Model
# ------------------------------
estimators = [
    ("svm", SVC(probability=True, kernel="linear")),
    ("nb", GaussianNB()),
    ("dt", DecisionTreeClassifier(max_depth=3))
]
stack = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression(
stack.fit(X_train, y_train)
stack_acc, stack_f1 = evaluate_model(stack, X_test, y_test, "Stacked Ensemble")

# ------------------------------
# Cross-Validation Comparison
# ------------------------------
models = {
    "Decision Tree": dt.best_estimator_,
    "AdaBoost": ada.best_estimator_,
    "Gradient Boosting": gb.best_estimator_,
    "XGBoost": xgb_clf.best_estimator_,
    "Random Forest": rf.best_estimator_,
    "Stacked": stack
}
```

```
cv_results = {}
for name, m in models.items():
    scores = cross_val_score(m, X, y, cv=5, scoring="accuracy")
    cv_results[name] = scores

cv_df = pd.DataFrame(cv_results)
print("\n5-Fold CV Results:\n", cv_df, "\n")
print("Average Accuracies:\n", cv_df.mean())

\end{lstlisting}

% -------------------------------
% MODEL-BY-MODEL RESULTS
% -------------------------------
```

Outputs

Decision Tree

Table 1: Decision Tree Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.83	0.95	0.89	42
1	0.97	0.89	0.93	72
Accuracy		0.9123		

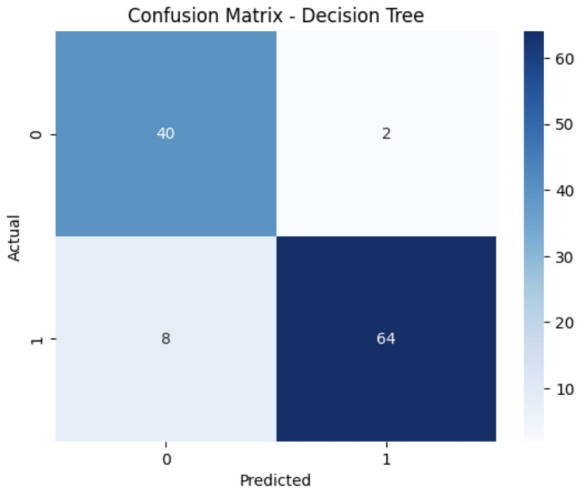

Figure 1: Decision Tree Confusion Matrix

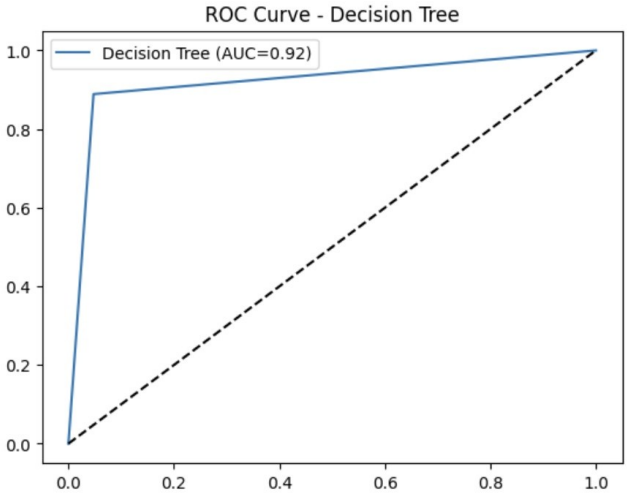

Figure 2: Decision Tree ROC Curve

AdaBoost

Table 2: AdaBoost Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.97	0.90	0.94	42
1	0.95	0.99	0.97	72
Accuracy		0.9561		

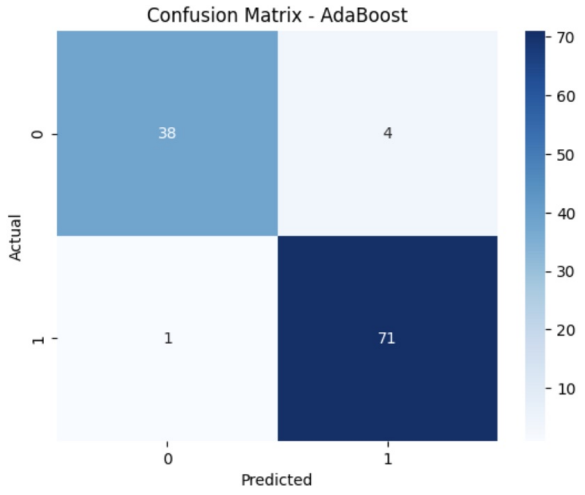

Figure 3: AdaBoost Confusion Matrix

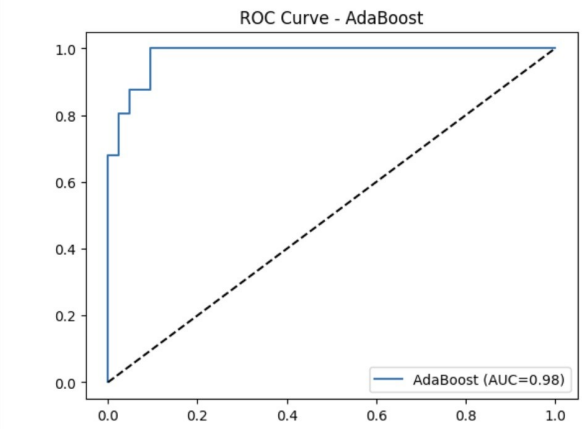

Figure 4: AdaBoost ROC Curve

Gradient Boosting

Table 3: Gradient Boosting Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.97	0.90	0.94	42
1	0.95	0.99	0.97	72
Accuracy		0.9561		

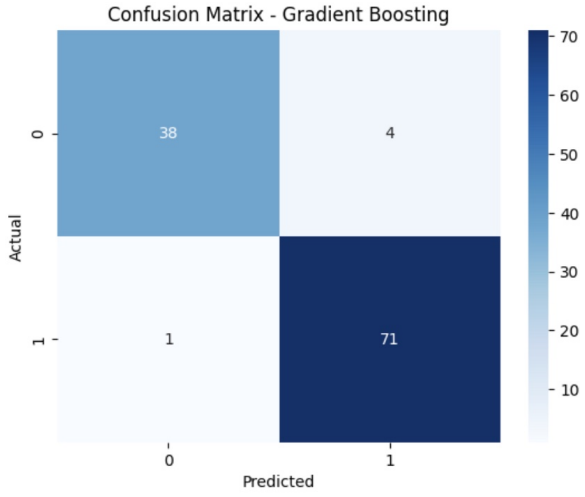

Figure 5: Gradient Boosting Confusion Matrix

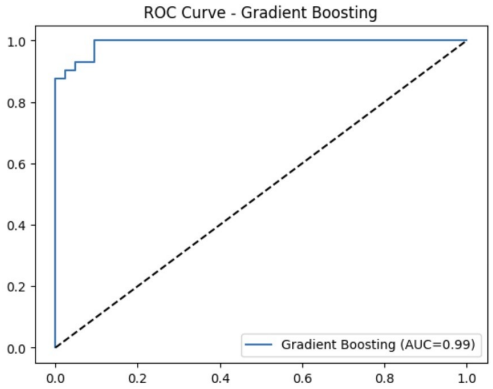

Figure 6: Gradient Boosting ROC Curve

XGBoost

Table 4: XGBoost Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.95	0.90	0.93	42
1	0.95	0.97	0.96	72
Accuracy		0.9474		

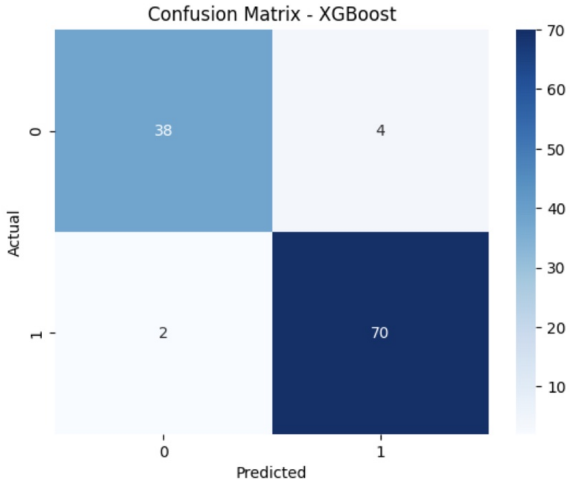

Figure 7: XGBoost Confusion Matrix

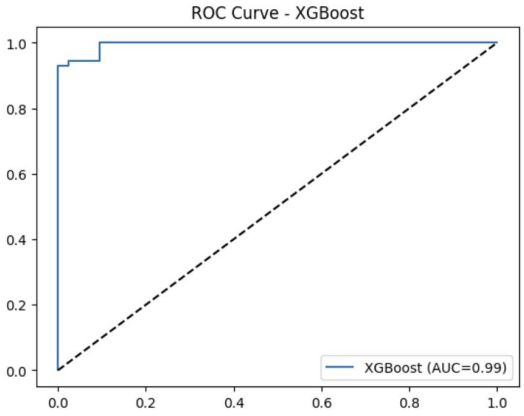

Figure 8: XGBoost ROC Curve

Random Forest

Table 5: Random Forest Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.95	0.93	0.94	42
1	0.96	0.97	0.97	72
Accuracy		0.9561		

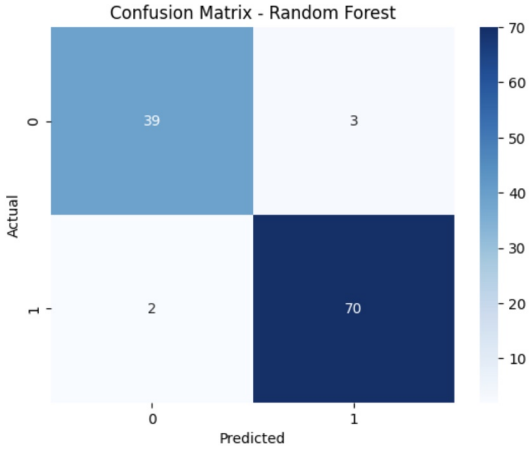

Figure 9: Random Forest Confusion Matrix

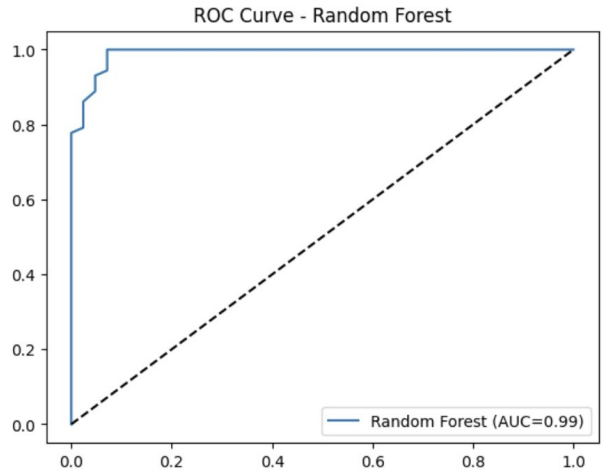

Figure 10: Random Forest ROC Curve

Stacked Ensemble

Table 6: Stacked Ensemble Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.98	0.95	0.96	42
1	0.97	0.99	0.98	72
Accuracy		0.9737		

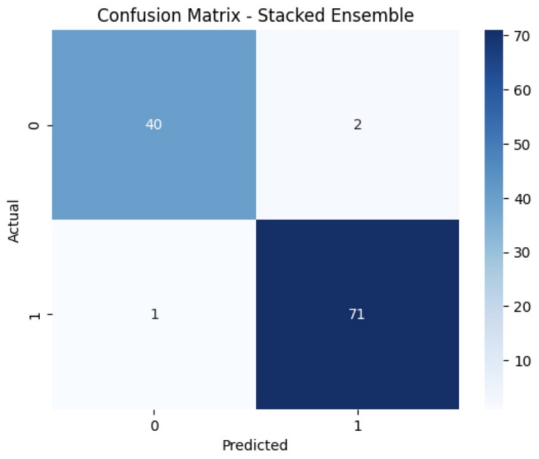

Figure 11: Stacked Ensemble Confusion Matrix

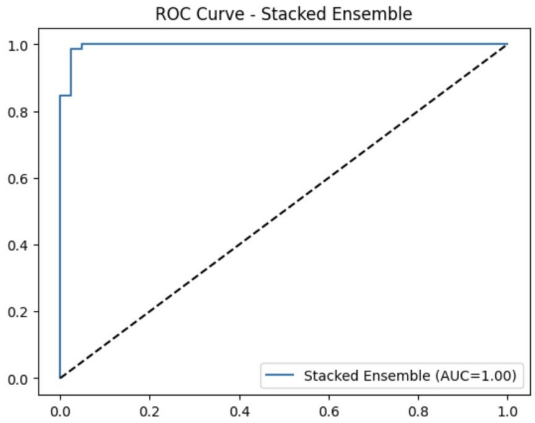

Figure 12: Stacked Ensemble ROC Curve

Tables

Hyperparameter Tuning

Table 7: Comparison of Best Models and Hyperparameters

Model	**Best Hyperparameters**	**Accuracy**	**F1 Score**
Decision Tree	criterion = entropy, max_depth = 7	0.9123	0.9275
AdaBoost	n_estimators = 100, learning_rate = 1	0.9561	0.9660
Gradient Boosting	n_estimators = 50, learning_rate = 0.5, max_depth = 3	0.9561	0.9660
XGBoost	n_estimators = 100, learning_rate = 0.1, max_depth = 3, gamma = 0	0.9474	0.9589
Random Forest	n_estimators = 50, criterion = entropy, max_depth = None	0.9561	0.9655
Stacked Ensemble	Base = (SVM, Naïve Bayes, Decision Tree), Final = Logistic Regression	**0.9737**	**0.9793**

5-Fold Cross Validation Results

Table 8: 5-Fold Cross Validation Comparison

Model	Fold1	Fold2	Fold3	Fold4	Fold5	Avg Accuracy
Decision Tree	0.9298	0.9210	0.9474	0.9386	0.9461	0.9367
AdaBoost	0.9825	0.9649	0.9912	0.9649	0.9823	0.9772
Gradient Boosting	0.9123	0.9298	0.9825	0.9737	0.9558	0.9580
XGBoost	0.9561	0.9474	0.9912	0.9825	0.9646	0.9684
Random Forest	0.9386	0.9561	0.9912	0.9825	0.9646	0.9666
Stacked Model	0.9474	0.9474	0.9561	0.9737	0.9734	0.9596

Results

```
Average Accuracies:
 Decision Tree        0.936749
AdaBoost             0.977162
Gradient Boosting    0.950800
XGBoost              0.968359
Random Forest        0.966605
Stacked              0.959603
dtype: float64
```

Inference

- **Best test accuracy:** Stacked Ensemble (97.37%, F1=97.93%).
- **Decision Tree vs Ensembles:** Decision Tree (91.23%) far weaker than ensembles.
- **Random Forest:** Stable generalization with 95.61% test accuracy and 96.66% CV accuracy.
- **XGBoost:** 94.74% test accuracy, but strong CV (96.84%), showing robust performance.
- **AdaBoost:** Top performer in CV (97.72%), close to stacking.
- **Stacking:** Outperformed all in test set, best overall generalization.

Learning Outcomes

- Learned model-by-model evaluation with classification metrics.
- Understood hyperparameter tuning and performance trade-offs.
- Practiced visualizing results via confusion matrices and ROC curves.
- Verified that ensemble methods outperform standalone classifiers.
- Gained insight into why stacking often generalizes best.