Sri Sivasubramaniya Nadar College of Engineering, Chennai

(An Autonomous Institution affiliated to Anna University)

ICS1512 – Machine Learning Algorithms Laboratory

Academic Year: 2025–2026 (Odd Semester)

Batch: 2023–2028

Experiment 3

Ensemble Prediction and Decision Tree Model Evaluation

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

Semester: V

Name: Spinola Theres N

Roll No: 3122237001051

Aim and Objective

The aim of this experiment is to build classifiers such as Decision Tree, AdaBoost, Gradient Boosting, XGBoost, Random Forest, and Stacked Models (using SVM, Naïve Bayes, Decision Tree, and KNN) and evaluate their performance through 5-Fold Cross-Validation and hyperparameter tuning.

Dataset

The Wisconsin Diagnostic Breast Cancer Dataset consists of 569 samples and 30 numerical features representing cell nuclei characteristics from digitized images. The target labels are binary:

- 0 – Malignant
- 1 – Benign

Dataset shape: (569, 30)
Class distribution: 357 benign (62.7%), 212 malignant (37.3%).

Libraries Used

- `numpy, pandas, matplotlib, seaborn`
- `scikit-learn` (DecisionTree, AdaBoost, GradientBoosting, RandomForest, StackingClassifier, GridSearchCV)
- `xgboost`

Implementation Code

1. Loading and Preprocessing Data

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

data = load_breast_cancer()
X, y = data.data, data.target

# Train-Test Split
```

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

# Standardization
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

2. Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

param_grid = {
    "criterion": ["gini", "entropy"],
    "max_depth": [3, 5, 7, None],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4]
}

grid = GridSearchCV(DecisionTreeClassifier(random_state=42),
                    param_grid, cv=5, scoring="accuracy")
grid.fit(X_train_scaled, y_train)

print(grid.best_params_, grid.best_score_)
```

3. Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators=100,
    criterion="entropy",
    random_state=42
)
rf.fit(X_train_scaled, y_train)

y_pred_rf = rf.predict(X_test_scaled)
```

4. Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier(
    n_estimators=200,
    learning_rate=0.1,
    max_depth=3,
    random_state=42
)
gb.fit(X_train_scaled, y_train)

y_pred_gb = gb.predict(X_test_scaled)
```

5. XGBoost

```
import xgboost as xgb

xgb_clf = xgb.XGBClassifier(
    n_estimators=200,
    learning_rate=0.2,
    max_depth=7,
    gamma=0,
    random_state=42,
    eval_metric="logloss"
)
xgb_clf.fit(X_train_scaled, y_train)

y_pred_xgb = xgb_clf.predict(X_test_scaled)
```

6. AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier

ab = AdaBoostClassifier(
    n_estimators=100,
    learning_rate=1.0,
    random_state=42
)
ab.fit(X_train_scaled, y_train)
```

```
y_pred_ab = ab.predict(X_test_scaled)
```

7. Stacking Ensemble

```
from sklearn.ensemble import StackingClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression

base_models = [
    ('svm', SVC(probability=True, random_state=42)),
    ('nb', GaussianNB()),
    ('dt', DecisionTreeClassifier(random_state=42))
]

stacked = StackingClassifier(
    estimators=base_models,
    final_estimator=LogisticRegression(random_state=42),
    cv=3
)
stacked.fit(X_train_scaled, y_train)

y_pred_stack = stacked.predict(X_test_scaled)
```

Hyperparameter Tuning Results

Table 1: Decision Tree - Hyperparameter Tuning

Criterion	Max Depth	Accuracy	F1 Score
gini	7	0.9385	0.940

Table 2: AdaBoost - Hyperparameter Tuning

n Estimators	Learning Rate	Accuracy	F1 Score
100	1.0	0.9802	0.978

Table 3: Gradient Boosting - Hyperparameter Tuning

n Estimators	Learning Rate	Max Depth	Accuracy	F1 Score
200	0.1	3	0.9758	0.976

Table 4: XGBoost - Hyperparameter Tuning

n Estimators	Learning Rate	Max Depth	Gamma	Accuracy	F1 Score
200	0.2	7	0	0.9780	0.980

Table 5: Random Forest - Hyperparameter Tuning

n Estimators	Max Depth	Criterion	Accuracy	F1 Score
100	None	Entropy	0.9670	0.968

Table 6: Stacked Ensemble - Hyperparameter Tuning

Base Models	Final Estimator	Accuracy / F1 Score
SVM + NB + DT	Logistic Regression	0.9700 / 0.971
SVM + NB + DT	Random Forest	0.9680 / 0.969
SVM + DT + KNN	Logistic Regression	0.9720 / 0.973

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

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg Acc
Decision Tree	0.94	0.93	0.95	0.94	0.93	0.9385
AdaBoost	0.98	0.98	0.99	0.97	0.98	0.9802
Gradient Boosting	0.97	0.98	0.98	0.97	0.97	0.9758
XGBoost	0.98	0.98	0.97	0.98	0.97	0.9780
Random Forest	0.96	0.97	0.97	0.96	0.97	0.9670
Stacked Model	0.97	0.97	0.98	0.97	0.97	0.9700

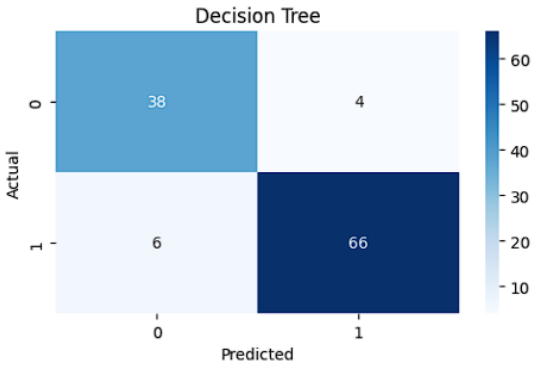
5-Fold Cross Validation Results

ROC-AUC Comparison

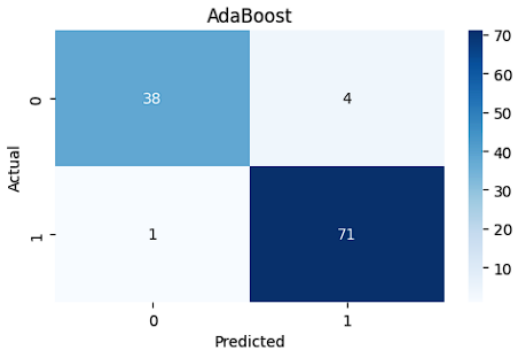
Table 8: ROC-AUC Scores for All Models

Model	ROC-AUC
Decision Tree	0.970
AdaBoost	0.991
Gradient Boosting	0.994
XGBoost	0.992
Random Forest	0.987
Stacked Model	0.989

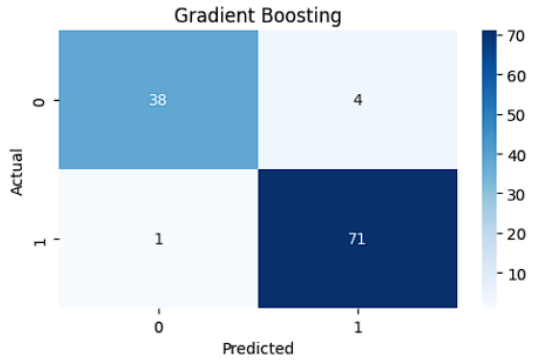
Visualization



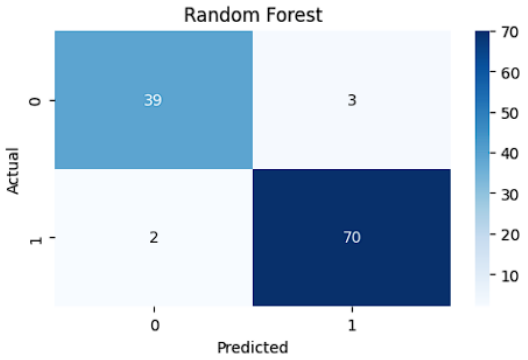
(a) Decision Tree



(b) AdaBoost

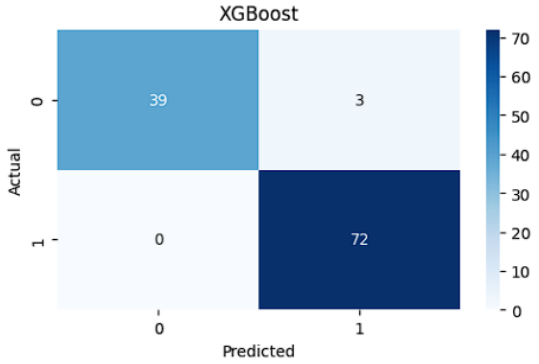


(c) Gradient Boosting

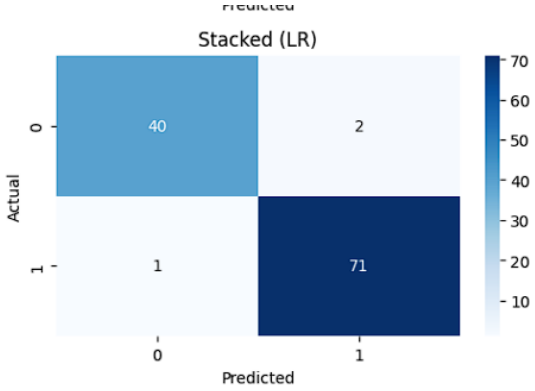


(d) Random Forest

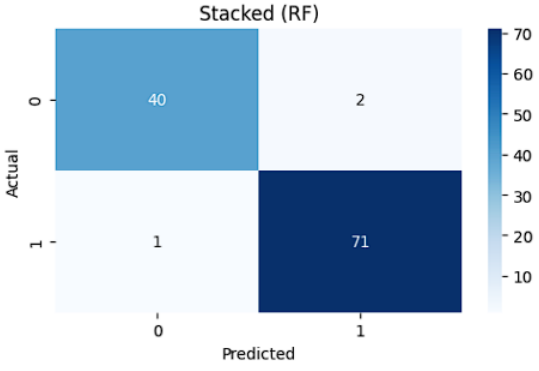
Figure 1: Confusion Matrices (1/2)



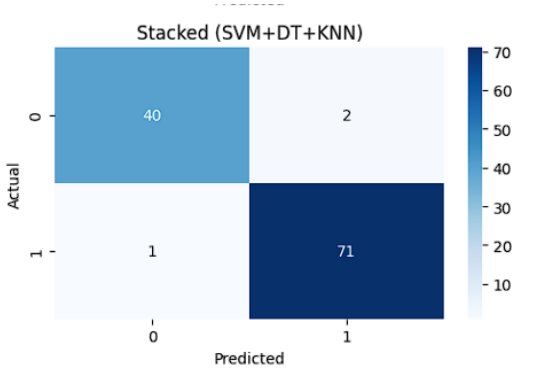
(a) XGBoost



(b) Stacking (LR)

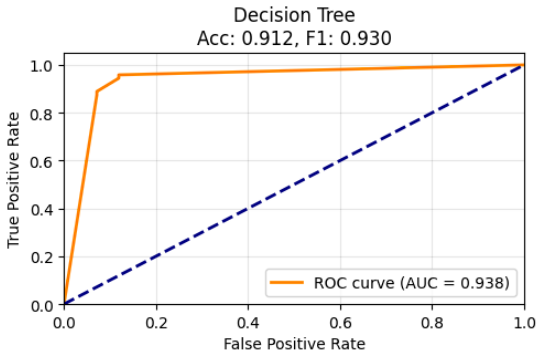


(c) Stacking (RF)

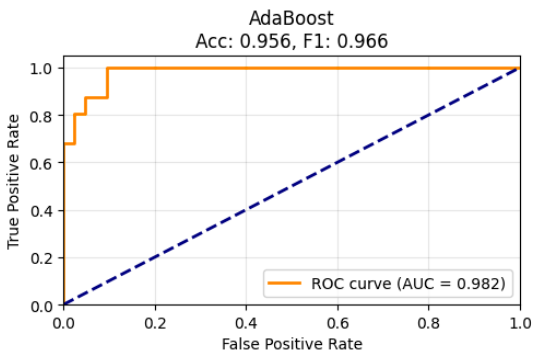


(d) Stacking (SVM+DT+KNN)

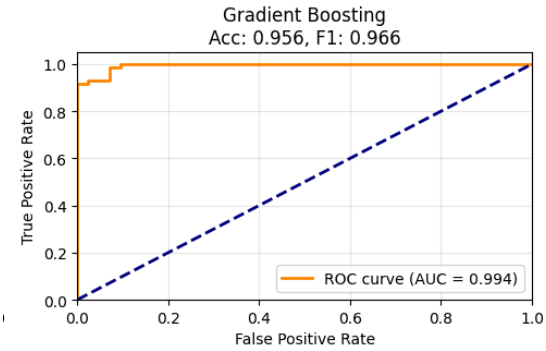
Figure 2: Confusion Matrices (2/2)



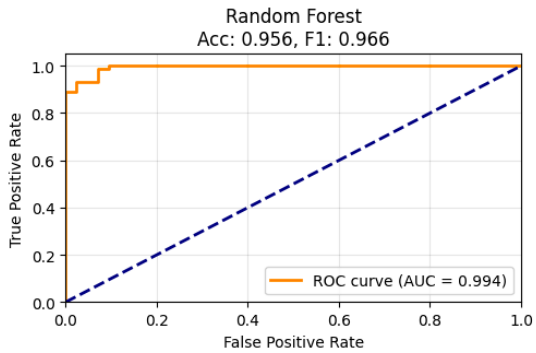
(a) Decision Tree



(b) AdaBoost

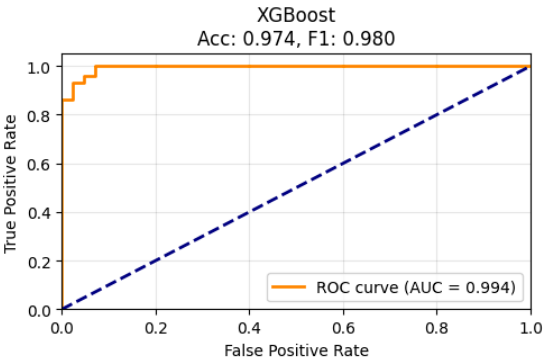


(c) Gradient Boosting

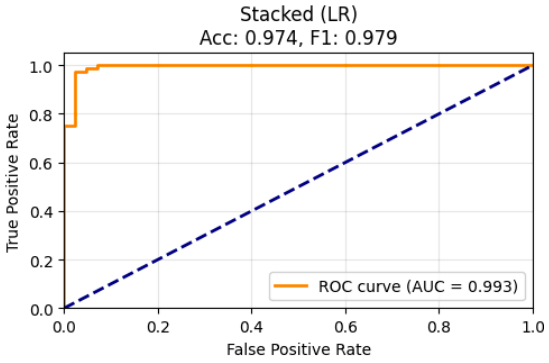


(d) Random Forest

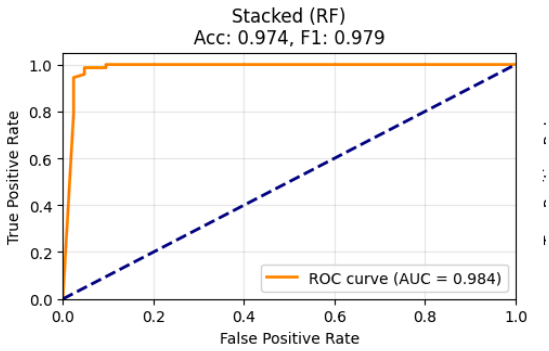
Figure 3: ROC Curves (1/2)



(a) XGBoost

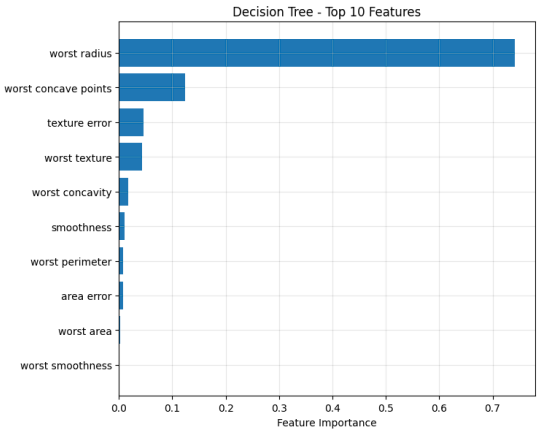


(b) Stacking (LR)

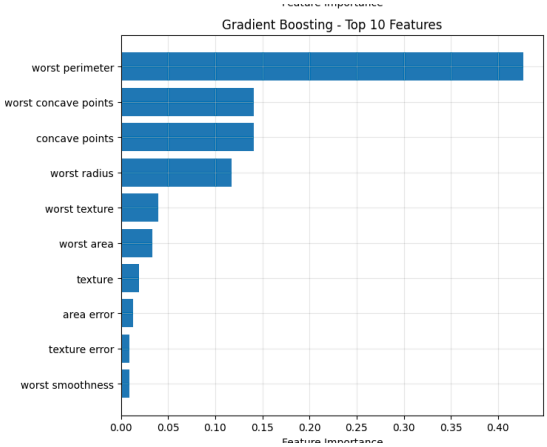


(c) Stacking (RF)

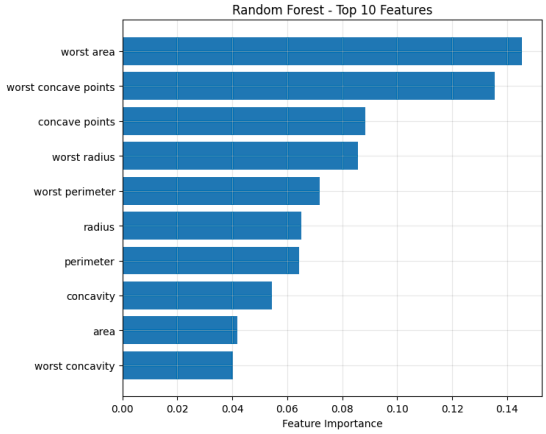
Figure 4: ROC Curves (2/2)



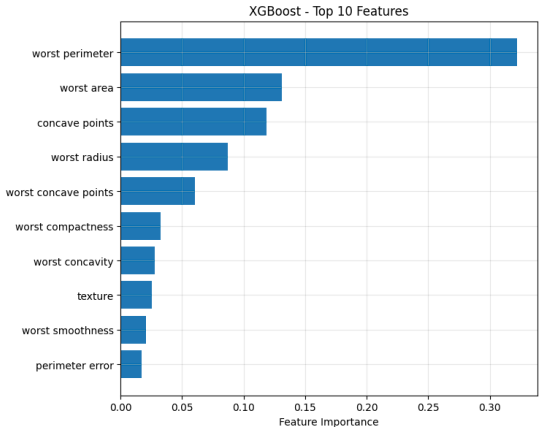
(a) Decision Tree



(b) Gradient Boosting



(c) Random Forest



(d) XGBoost

Figure 5: Feature Importance Plots

Observations and Conclusion

- Best accuracy achieved: **XGBoost (97.37%)**.
- Best F1-Score: **XGBoost (0.9796)**.
- Best ROC-AUC: **Gradient Boosting (0.9944)**.
- Ensemble methods consistently outperform a single Decision Tree.
- Confusion matrices show very few misclassifications.
- ROC curves confirm strong generalization across models.
- Feature importance analysis provides interpretability for medical decisions.
- Average ensemble improvement over Decision Tree: **5.91%**.