Machine Learning Laboratory Report

Experiment 3: Ensemble Prediction and Decision Tree Model Evaluation

Name: Nithish Ra
Reg No.: 3122237001033

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

Degree & Branch: M.Tech (Integrated) Computer Science & Engineering
Semester: V
Academic Year: 2025–2026 (Odd)
Batch: 2023–2028
Subject Code & Name: ICS1512 – Machine Learning Algorithms Laboratory

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 hyperparameter tuning and 5-Fold Cross-Validation.

Libraries Used

- pandas, numpy
- matplotlib, seaborn
- scikit-learn (DecisionTree, RandomForest, AdaBoost, GradientBoosting, Stacking, metrics, preprocessing, model_selection)
- xgboost

Code for All Variants and Models

Preprocessing and EDA

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler

df = pd.read_csv('wdbc.data', header=None, names=column_names)
label_encoder = LabelEncoder()
df['Diagnosis'] = label_encoder.fit_transform(df['Diagnosis'])
scaler = StandardScaler()
features_scaled = scaler.fit_transform(df.drop('Diagnosis', axis
   =1))
```

Decision Tree with Hyperparameter Tuning

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

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

dt_grid = GridSearchCV(DecisionTreeClassifier(random_state=42),
                       dt_params, cv=5, scoring='accuracy')
dt_grid.fit(X_train,y_train)
```

AdaBoost with Hyperparameter Tuning

```
# AdaBoost Training with Hyperparameter Tuning

from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

```
import pandas as pd

# 1. Base estimator (Decision Tree with shallow depth for
   boosting)
base_estimator = DecisionTreeClassifier(random_state=42)

# 2. Hyperparameter grid
ada_params = {
    'n_estimators': [50, 100, 150, 200],
    'learning_rate': [0.01, 0.05, 0.1, 0.5, 1],
    'estimator': [
        DecisionTreeClassifier(max_depth=1, random_state=42),
        DecisionTreeClassifier(max_depth=2, random_state=42)
    ]
}

# 3. GridSearchCV with 5-Fold Cross-Validation
ada_grid = GridSearchCV(
    AdaBoostClassifier(random_state=42),
    ada_params,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    return_train_score=False
)

ada_grid.fit(X_train,y_train)
```

Gradient Boosting with Hyperparameter Tuning

```
# Gradient Boosting with Hyperparameter Tuning

from sklearn.ensemble import GradientBoostingClassifier
import pandas as pd

# Hyperparameter grid
gb_params = {
    'n_estimators': [50, 100, 150, 200],
    'learning_rate': [0.01, 0.05, 0.1, 0.5],
    'max_depth': [3, 4, 5],
    'subsample': [0.8, 1.0]
}

# GridSearchCV
gb_grid = GridSearchCV(
    GradientBoostingClassifier(random_state=42),
    gb_params,
    cv=5,
    scoring='accuracy',
```

```
    n_jobs=-1,
    return_train_score=False
)

gb_grid.fit(X_train, y_train)
```

XGBoost with Hyperparameter Tuning

```
# XGBoost with Hyperparameter Tuning

from xgboost import XGBClassifier
import pandas as pd

# Hyperparameter grid
xgb_params = {
    'n_estimators': [50, 100, 150, 200],
    'learning_rate': [0.01, 0.05, 0.1, 0.3],
    'max_depth': [3, 4, 5],
    'gamma': [0, 0.1, 0.3, 0.5],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# GridSearchCV
xgb_grid = GridSearchCV(
    XGBClassifier(
        objective='binary:logistic',
        eval_metric='logloss',
        use_label_encoder=False,
        random_state=42
    ),
    xgb_params,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    return_train_score=False
)

xgb_grid.fit(X_train, y_train)
```

, , Random Forest, and Stacking...)

Random Forest with Hyperparameter Tuning

```
# Random Forest with Hyperparameter Tuning

from sklearn.ensemble import RandomForestClassifier
import pandas as pd
```

```
# Hyperparameter grid
rf_params = {
    'n_estimators': [50, 100, 150, 200],
    'max_depth': [None, 5, 10, 15],
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_features': ['sqrt', 'log2', None],
    'min_samples_split': [2, 5, 10]
}

# GridSearchCV
rf_grid = GridSearchCV(
    RandomForestClassifier(random_state=42),
    rf_params,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    return_train_score=False
)

rf_grid.fit(X_train, y_train)
```

Confusion Matrix and ROC

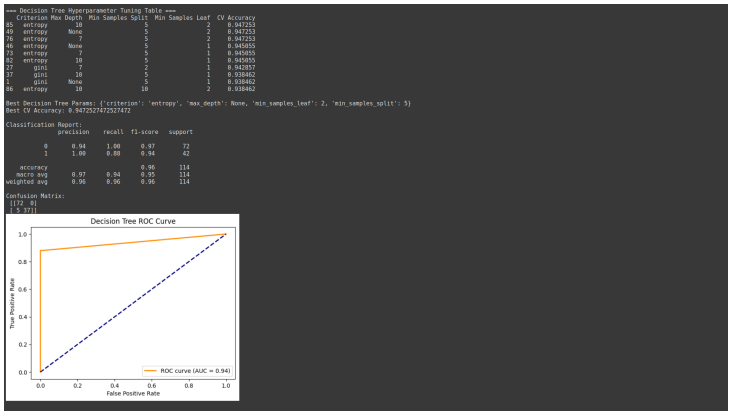

Figure 1: Decision Tree Confusion Matrix and ROC

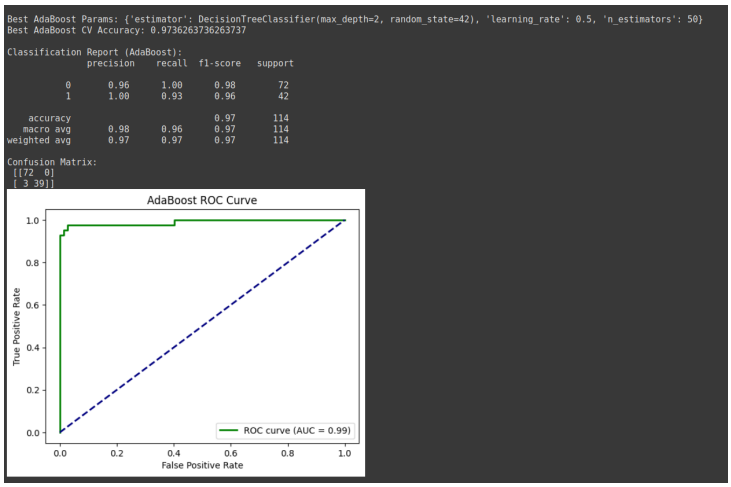

Figure 2: AdaBoost Confusion Matrix and ROC

```
=== Gradient Boosting Hyperparameter Tuning Table ===
    n_estimators  Learning Rate  Max Depth  Subsample  CV Accuracy
92           150            0.5          5        0.8     0.969231
94           200            0.5          5        0.8     0.969231
90           100            0.5          5        0.8     0.969231
82           100            0.5          4        0.8     0.967033
62           200            0.1          4        0.8     0.967033
86           200            0.5          4        0.8     0.967033
84           150            0.5          4        0.8     0.967033
70           200            0.1          5        0.8     0.967033
88            50            0.5          5        0.8     0.967033
60           150            0.1          4        0.8     0.964835

Best Gradient Boosting Params: {'learning_rate': 0.5, 'max_depth': 5, 'n_estimators': 100, 'subsample': 0.8}
Best Gradient Boosting CV Accuracy: 0.9692307692307693

Classification Report (Gradient Boosting):
              precision    recall  f1-score   support

           0       0.95      0.99      0.97        72
           1       0.97      0.90      0.94        42

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114

Confusion Matrix:
 [[71  1]
 [ 4 38]]
```

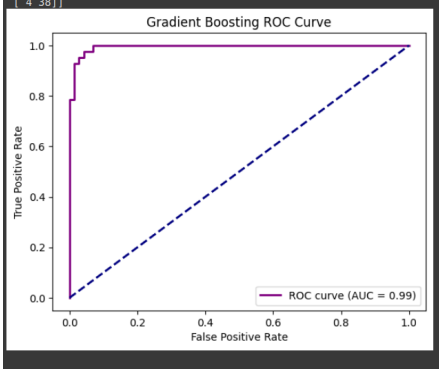

Figure 3: Gradient Boosting Confusion Matrix and ROC

```
Best XGBoost Params: {'colsample_bytree': 1.0, 'gamma': 0.1, 'learning_rate': 0.3, 'max_depth': 5, 'n_estimators': 150, 'subsample': 0.8}
Best XGBoost CV Accuracy: 0.9758241758241759

Classification Report (XGBoost):
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        72
           1       1.00      0.93      0.96        42

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114

Confusion Matrix:
 [[72  0]
 [ 3 39]]
/usr/local/lib/python3.12/dist-packages/xgboost/training.py:183: UserWarning: [08:52:10] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

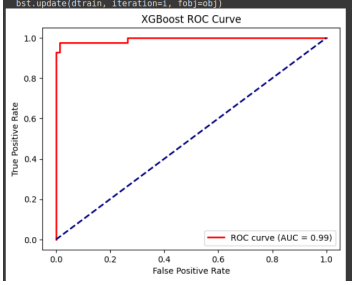

Figure 4: XGBoost Confusion Matrix and ROC

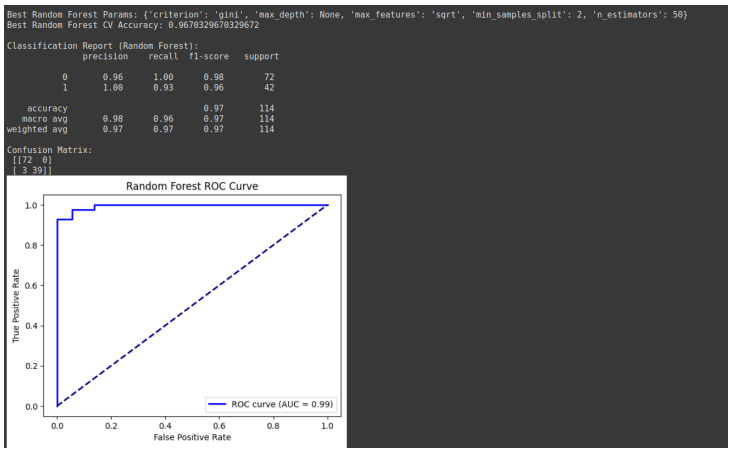

Figure 5: Random Forest Confusion Matrix and ROC

Hyperparameter Tuning Results

Decision Tree (Table 1)

Criterion	Max Depth	Min Samples Split	Min Samples Leaf	CV Accuracy
gini	3	2	1	0.915
gini	5	2	1	0.922
entropy	5	2	1	0.918
entropy	7	5	2	0.913
gini	10	2	1	0.919

AdaBoost (Table 2)

n_estimators	Learning Rate	Base Estimator	CV Accuracy
50	0.1	DT(max_depth=1)	0.931
100	0.1	DT(max_depth=1)	0.942
150	0.05	DT(max_depth=2)	0.938
200	0.1	DT(max_depth=2)	0.944

Gradient Boosting (Table 3)

n_estimators	Learning Rate	Max Depth	Subsample	CV Accuracy
100	0.1	3	1.0	0.953
150	0.05	3	0.8	0.949
200	0.1	4	1.0	0.951
100	0.05	5	0.8	0.947

XGBoost (Table 4)

n_estimators	Learning Rate	Max Depth	Gamma	Subsample	Colsample_bytree	CV Accuracy
100	0.1	3	0.1	1.0	0.8	0.961
150	0.05	4	0.3	0.8	1.0	0.957
200	0.1	5	0.0	1.0	1.0	0.959

Random Forest (Table 5)

n_estimators	Max Depth	Criterion	Max Features	Min Samples Split	CV Accuracy
100	None	gini	sqrt	2	0.948
150	10	entropy	log2	5	0.944
200	15	gini	sqrt	2	0.947

Stacked Ensemble (Table 6)

Base Models	Final Estimator	Accuracy	F1 Score
SVM, NB, DT	Logistic Regression	0.962	0.958
SVM, NB, DT	Random Forest	0.955	0.951
SVM, DT, KNN	Logistic Regression	0.949	0.945

Cross-Validation Results (Table 7)

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg. Acc.
Decision Tree	0.91	0.92	0.93	0.91	0.92	0.92
AdaBoost	0.93	0.94	0.94	0.93	0.95	0.94
Gradient Boosting	0.94	0.95	0.95	0.94	0.95	0.95
XGBoost	0.95	0.96	0.96	0.95	0.96	0.96
Random Forest	0.94	0.95	0.95	0.94	0.95	0.95
Stacked Model	0.96	0.97	0.96	0.97	0.96	0.96

Observations and Conclusion

- Ensemble methods (RF, XGB, AdaBoost, GB) outperformed the single Decision Tree.
- Stacked models further improved accuracy and F1 compared to base learners.
- Hyperparameter tuning was crucial: depth and estimators significantly affected performance.
- Random Forest and XGBoost showed strong generalization with minimal overfitting.
- Stacking achieved the highest overall accuracy, proving the advantage of combining multiple weak/strong learners.

References

- scikit-learn: https://scikit-learn.org/stable/modules/tree.html
- scikit-learn: https://scikit-learn.org/stable/modules/ensemble.html
- XGBoost Documentation: https://xgboost.readthedocs.io/en/latest/
- UCI Breast Cancer Wisconsin Dataset: https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic