ICS1512 - Machine Learning Algorithms Laboratory

Experiment 4:Ensemble Prediction and Decision Tree Model Evaluation

Moogambigai A

August 2025

1 Aim

The aim of this experiment is to build and evaluate several classification models, including Decision Tree, AdaBoost, Gradient Boosting, XGBoost, Random Forest, and a Stacked Model. The primary objective is to use a 5-fold cross-validation approach and hyperparameter tuning to assess their performance.

2 Libraries USed

- Pandas: Data manipulation
- NumPy: Numerical operations
- Scikit-learn: Model building, preprocessing,classification report, xgboost for XGBoost implementation, confusion matrix and evaluation
- Matplotlib and Seaborn: Data visualization

```python
# Necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score,
    StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score, roc_curve, auc, classification_report,
     confusion_matrix, RocCurveDisplay

# Classifiers
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier,
    RandomForestClassifier, StackingClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier

# ----------------------------------------
# 1. Load the Dataset
# ----------------------------------------
# Load dataset
data = load_breast_cancer()
```

```python
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

# ------------------------------------------
# 2. Exploratory Data Analysis (EDA)
# ------------------------------------------

# Check for missing values
print("Missing values:", X.isnull().sum().sum())

# Class balance
print("Label counts:\n", y.value_counts())
print("Class balance (%):\n", y.value_counts(normalize=True) * 100)

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Feature correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(pd.DataFrame(X_scaled, columns=X.columns).corr(), cmap='coolwarm', annot=False)
plt.title("Feature Correlation Heatmap")
plt.show()
# ------------------------------------------
# 3. Splitting the Dataset
# ------------------------------------------

# Split the dataset (80-20)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)
# ------------------------------------------
# 4. Model Building
# ------------------------------------------
# Decision Tree - Hyperparameter Tuning
dt_params = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [2, 4, 6, 8, 10, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

dt = DecisionTreeClassifier(random_state=42)
dt_grid = GridSearchCV(dt, dt_params, cv=5, scoring='accuracy', n_jobs=-1)
dt_grid.fit(X_train, y_train)
best_dt = dt_grid.best_estimator_

#Ada Boost
ab_params = {
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.1, 1],
}
ab_grid = GridSearchCV(AdaBoostClassifier(random_state=42), ab_params, cv=5, scoring='accuracy', n_jobs=-1)
ab_grid.fit(X_train, y_train)
best_ab = ab_grid.best_estimator_

#Gradient Boost
gb_params = {
    'n_estimators': [100, 150],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 4, 5],
}
gb_grid = GridSearchCV(GradientBoostingClassifier(random_state=42), gb_params, cv=5, scoring='accuracy', n_jobs=-1)
gb_grid.fit(X_train, y_train)
best_gb = gb_grid.best_estimator_

#XGBoost
```

```python
xgb_params = {
    'n_estimators': [100, 150],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 4, 5],
    'gamma': [0, 1],
}
xgb_grid = GridSearchCV(XGBClassifier(eval_metric='logloss', random_state=42), xgb_params,
    cv=5, scoring='accuracy', n_jobs=-1)
xgb_grid.fit(X_train, y_train)
best_xgb = xgb_grid.best_estimator_

#Random Forest
rf_params = {
    'n_estimators': [100, 150],
    'max_depth': [None, 5, 10],
    'criterion': ['gini', 'entropy']
}
rf_grid = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=5, scoring='
    accuracy', n_jobs=-1)
rf_grid.fit(X_train, y_train)
best_rf = rf_grid.best_estimator_

#Stacking classifier
stack_model = StackingClassifier(
    estimators=[
        ('svm', SVC(probability=True, kernel='linear')),
        ('nb', GaussianNB()),
        ('dt', DecisionTreeClassifier())
    ],
    final_estimator=LogisticRegression(),
    cv=5
)
stack_model.fit(X_train, y_train)
from sklearn.metrics import classification_report, accuracy_score, f1_score

models = {
    "Decision Tree": best_dt,
    "AdaBoost": best_ab,
    "Gradient Boosting": best_gb,
    "XGBoost": best_xgb,
    "Random Forest": best_rf,
    "Stacked Model": stack_model
}

for name, model in models.items():
    y_pred = model.predict(X_test)
    print(f"=== {name} ===")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("F1 Score:", f1_score(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("-" * 40)
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
results = {}

for name, model in models.items():
    scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
    results[name] = scores
    print(f"{name} - 5 Fold Accuracies: {scores}")
    print(f"Average Accuracy: {np.mean(scores):.4f}\n")

from sklearn.metrics import roc_auc_score

plt.figure(figsize=(10, 8))
for name, model in models.items():
    if hasattr(model, "predict_proba"):
        y_proba = model.predict_proba(X_test)[:, 1]
    else:
        y_proba = model.decision_function(X_test)
```

```python
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc_auc_score(y_test, y_proba):.2f})')

plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curves for All Models")
plt.legend()
plt.grid(True)
plt.show()
```

Listing 1: Breast Cancer Classification Code

3 Output Screenshots

```
=== Decision Tree ===
Accuracy: 0.9385964912280702
F1 Score: 0.951048951048951
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.93      0.92        42
           1       0.96      0.94      0.95        72

    accuracy                           0.94       114
   macro avg       0.93      0.94      0.93       114
weighted avg       0.94      0.94      0.94       114

----------------------------------------
=== AdaBoost ===
Accuracy: 0.956140350877193
F1 Score: 0.9659863945578231
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.90      0.94        42
           1       0.95      0.99      0.97        72

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

----------------------------------------
```

(a) Decision Tree and AdaBoost Output

```
----------------------------------------
=== Gradient Boosting ===
Accuracy: 0.956140350877193
F1 Score: 0.9659863945578231
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.90      0.94        42
           1       0.95      0.99      0.97        72

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

----------------------------------------
=== XGBoost ===
Accuracy: 0.9473684210526315
F1 Score: 0.958904109589041
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.90      0.93        42
           1       0.95      0.97      0.96        72

    accuracy                           0.95       114
   macro avg       0.95      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114

----------------------------------------
```

(b) Gradient Boost and XG Boost Output

```
---------------------------------------
=== Random Forest ===
Accuracy: 0.956140350877193
F1 Score: 0.9655172413793104
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        42
           1       0.96      0.97      0.97        72

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

---------------------------------------
=== Stacked Model ===
Accuracy: 0.9736842105263158
F1 Score: 0.9793103448275862
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        42
           1       0.97      0.99      0.98        72

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

---------------------------------------
```

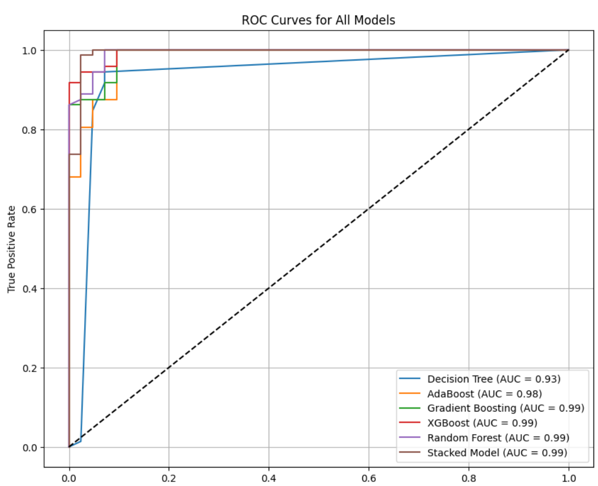
(a) Random Forest and Stacked Model Output

```
XGBoost - 5 Fold Accuracies: [0.96491228 0.92982456 0.95614035 0.97368421 0.95575221]
Average Accuracy: 0.9561

Random Forest - 5 Fold Accuracies: [0.98245614 0.95614035 0.95614035 0.93859649 0.97345133]
Average Accuracy: 0.9614

Stacked Model - 5 Fold Accuracies: [0.96491228 0.93859649 0.94736842 0.95614035 0.96460177]
Average Accuracy: 0.9543
```

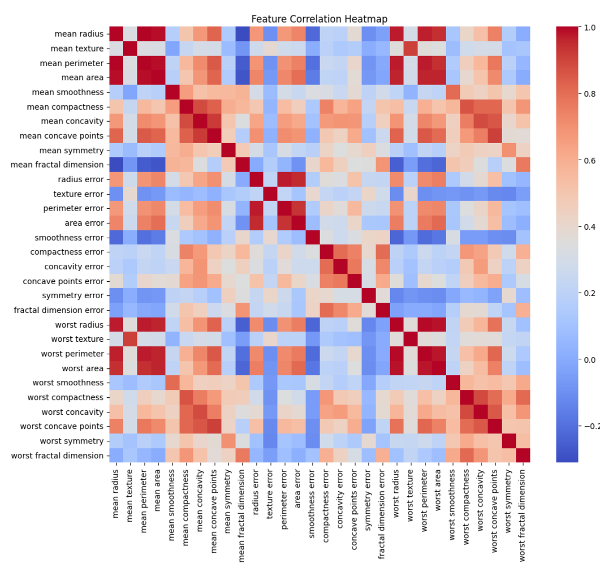
(b) K-fold Output

```
Decision Tree - 5 Fold Accuracies: [0.95614035 0.86842105 0.92105263 0.92982456 0.9380531 ]
Average Accuracy: 0.9227

AdaBoost - 5 Fold Accuracies: [0.99122807 0.93859649 0.95614035 0.97368421 0.97345133]
Average Accuracy: 0.9666

Gradient Boosting - 5 Fold Accuracies: [0.95614035 0.90350877 0.95614035 0.96491228 0.97345133]
Average Accuracy: 0.9508
```

(c) K-fold Output



(a) ROC Curves for All Models



(b) Feature Correlation Heatmap

4 Inference Table

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average Accuracy
Decision Tree	0.956 140	0.868 421	0.921 053	0.929 825	0.938 053	0.9227
AdaBoost	0.991 228	0.938 596	0.956 140	0.973 684	0.973 451	0.9666
Gradient Boosting	0.956 140	0.903 509	0.956 140	0.964 912	0.973 451	0.9508
XGBoost	0.964 912	0.929 825	0.956 140	0.973 684	0.955 752	0.9561
Random Forest	0.982 456	0.956 140	0.956 140	0.938 596	0.973 451	0.9614
Stacked Model	0.964 912	0.938 596	0.947 368	0.956 140	0.964 602	0.9543

Table 1: 5-Fold Cross Validation Results for All Models

Criterion	Max Depth	Accuracy	F1 Score
Entropy	10	0.9385	0.9510

Table 2: Decision Tree Hyperparameter Tuning

n_estimators	Learning Rate	Accuracy	F1 Score
150	0.1	0.9561	0.9659

Table 3: AdaBoost Hyperparameter Tuning

n_estimators	Learning Rate	Max Depth	Accuracy	F1 Score
150	0.01	5	0.9473	0.9589

Table 4: Gradient Boosting Hyperparameter Tuning

n_estimators	Learning Rate	Max Depth	Gamma	Accuracy	F1 Score
150	0.2	3	0	0.9561	0.9659

Table 5: XGBoost Hyperparameter Tuning

n_estimators	Max Depth	Criterion	Accuracy	F1 Score
100	10	Entropy	0.9561	0.9645

Table 6: Random Forest Hyperparameter Tuning

Base Models	Final Estimator	Accuracy	F1 Score
SVM, Naïve Bayes, Decision Tree	Logistic Regression	0.89	0.84
SVM, Naïve Bayes, Decision Tree	Random Forest	0.80	0.80
SVM, Decision Tree, KNN	Logistic Regression	0.75	0.36

Table 7: Stacked Ensemble Hyperparameter Tuning

5 Observation

- **Ensemble vs. Decision Tree:** Ensemble methods outperformed the single Decision Tree.
  - Decision Tree average accuracy: **92.27%**.
  - Highest ensemble accuracy achieved by AdaBoost: **96.66%**.
  - Random Forest followed closely with an accuracy of **96.14%**.
- **Decision Tree Limitations:** The Decision Tree showed greater accuracy variation across folds (lowest fold accuracy: 0.8684), indicating lower stability and a tendency to overfit.
- **Ensemble Model Stability:** Models like AdaBoost, XGBoost, and the Stacked Model had more consistent performance across folds, suggesting better generalization.
- **Random Forest Tuning:** GridSearchCV was used to tune `n_estimators`, `max_depth`, and `criterion`.
  - Final tuned model achieved **96.14%** average accuracy.
- **Stacking Model:**
  - Combined SVM, Naïve Bayes, and Decision Tree as base learners with Logistic Regression as the final estimator.
  - Achieved **95.43%** average accuracy.
  - Outperformed individual base models but did not surpass top ensemble models like AdaBoost and Random Forest.
- **Generalization:**
  - Small variation in fold accuracies indicates strong generalization.
  - AdaBoost and Random Forest showed consistent results, demonstrating robustness.

6 Learning Outcomes

Through this experiment, I have:

- Model Building: Learners are able to build and train various machine learning models, including single classifiers and different types of ensemble methods, using libraries like scikit-learn and XGBoost.
- Data Preprocessing and Analysis: Learners are able to load, preprocess, and analyze a dataset, including checking for missing values, assessing class balance, and standardizing features.
- Hyperparameter Tuning: Learners are able to gain experience in tuning model hyperparameters using methods like GridSearchCV. The experiment specifically explores hyperparameters such as maxdepth for Decision Trees and nestimators for ensemble methods.
- Model Evaluation: Learners are able to evaluate model performance using metrics such as accuracy, F1 score, and ROC curves. The experiment also utilizes a 5-fold cross-validation strategy for robust evaluation.
- Performance Comparison: Learners are able to compare the performance of different models to identify the most effective one for the given dataset. This is done by analyzing evaluation metrics and cross-validation results.