Sri Sivasubramaniya Nadar College of Engineering, Chennai

Degree & Branch	5 years Integrated M.Tech CSE	**Semester**	V
Subject Code & Name	ICS1512 – Machine Learning Algorithms Laboratory		
Academic Year	2025–2026 (Odd Semester)	**Batch**	2023–2028
Name	Nitish Konda	**Reg No**	3122237001034

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

Libraries used:

- Numpy
- Pandas
- Matplot Lib
- Scikit-Learn
- Seaborn

Python Code Implementation

```
# ========================================
# 1. Import Libraries
# ========================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# ========================================
# 2. Load Dataset
# ========================================
df = pd.read_csv(r'/content/drive/MyDrive/ML_Experiment4/wdc.csv')
```

```
# ========================================
# 3. Data Overview
# ========================================
print(df.select_dtypes(include='number').columns)
print(df.select_dtypes(include='object').columns)
df.isnull().sum()

# ========================================
# 4. Encode and Scale Data
# ========================================
from sklearn.preprocessing import LabelEncoder, StandardScaler
le = LabelEncoder()
df['Diagnosis'] = le.fit_transform(df['Diagnosis'])

ss = StandardScaler()
numeric_cols = df.select_dtypes(include='number').columns.drop('Diagnosis')
df[numeric_cols] = ss.fit_transform(df[numeric_cols])

# ========================================
# 5. Correlation Heatmap
# ========================================
plt.figure(figsize=(14, 10))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Feature Correlation Matrix")
plt.show()

# ========================================
# 6. Prepare Features and Target
# ========================================
X = df.drop(['Diagnosis', 'ID'], axis=1)
y = df['Diagnosis']

# ========================================
# 7. Train/Test Split
# ========================================
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# ========================================
# 8. Decision Tree with Grid Search
# ========================================
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve

dt_params = {
    'max_depth': [None, 3, 5, 7, 10],
```

```
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['gini', 'entropy']
}
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)

dt_best = dt_grid.best_estimator_
dt_pred = dt_best.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, dt_pred))
print(classification_report(y_test, dt_pred))

# ========================================
# 9. AdaBoost Classifier
# ========================================
from sklearn.ensemble import AdaBoostClassifier

ada = AdaBoostClassifier(estimator=DecisionTreeClassifier(random_state=42), random_state=42)
ada_params = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1],
    'estimator': [
        DecisionTreeClassifier(max_depth=1, random_state=42),
        DecisionTreeClassifier(max_depth=2, random_state=42)
    ]
}
ada_grid = GridSearchCV(ada, ada_params, scoring='accuracy', cv=5, n_jobs=-1)
ada_grid.fit(X_train, y_train)

ada_best = ada_grid.best_estimator_
ada_pred = ada_best.predict(X_test)
print("AdaBoost Accuracy:", accuracy_score(y_test, ada_pred))
print(classification_report(y_test, ada_pred))

# ========================================
# 10. Gradient Boosting Classifier
# ========================================
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier(random_state=42)
gb_params = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [1, 3, 5],
    'subsample': [0.8, 1.0]
}
gb_grid = GridSearchCV(gb, gb_params, scoring='accuracy', cv=5, n_jobs=-1)
```

```
gb_grid.fit(X_train, y_train)

gb_best = gb_grid.best_estimator_
gb_pred = gb_best.predict(X_test)
print("Gradient Boosting Accuracy:", accuracy_score(y_test, gb_pred))
print(classification_report(y_test, gb_pred))

# =======================================
# 11. XGBoost Classifier
# =======================================
from xgboost import XGBClassifier

xgb = XGBClassifier(random_state=42)
xgb_params = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'gamma': [0, 0.1, 0.3],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
xgb_grid = GridSearchCV(xgb, xgb_params, scoring='accuracy', cv=5, n_jobs=-1)
xgb_grid.fit(X_train, y_train)

xgb_best = xgb_grid.best_estimator_
xgb_pred = xgb_best.predict(X_test)
print("XGBoost Accuracy:", accuracy_score(y_test, xgb_pred))
print(classification_report(y_test, xgb_pred))

# =======================================
# 12. Random Forest Classifier
# =======================================
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state=42)
rf_params = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'criterion': ['gini', 'entropy'],
    'max_features': ['sqrt', 'log2'],
    'min_samples_split': [2, 5, 10]
}
rf_grid = GridSearchCV(rf, rf_params, scoring='accuracy', cv=5, n_jobs=-1)
rf_grid.fit(X_train, y_train)

rf_best = rf_grid.best_estimator_
rf_pred = rf_best.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
```

```
print(classification_report(y_test, rf_pred))

# ========================================
# 13. Stacking Classifier
# ========================================
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))
]
meta_model = LogisticRegression(random_state=42, max_iter=1000)

stack = StackingClassifier(
    estimators=base_models,
    final_estimator=meta_model,
    cv=5,
    n_jobs=-1
)

stack_params = {
    'svm__C': [0.1, 1, 10],
    'svm__kernel': ['linear', 'rbf'],
    'dt__max_depth': [3, 5, None],
    'final_estimator__C': [0.1, 1, 10]
}
stack_grid = GridSearchCV(
    estimator=stack,
    param_grid=stack_params,
    scoring='accuracy',
    cv=5,
    n_jobs=-1
)
stack_grid.fit(X_train, y_train)

stack_best = stack_grid.best_estimator_
stack_pred = stack_best.predict(X_test)
print("Stacking Classifier Accuracy:", accuracy_score(y_test, stack_pred))
print(classification_report(y_test, stack_pred))
```

Model Visualizations

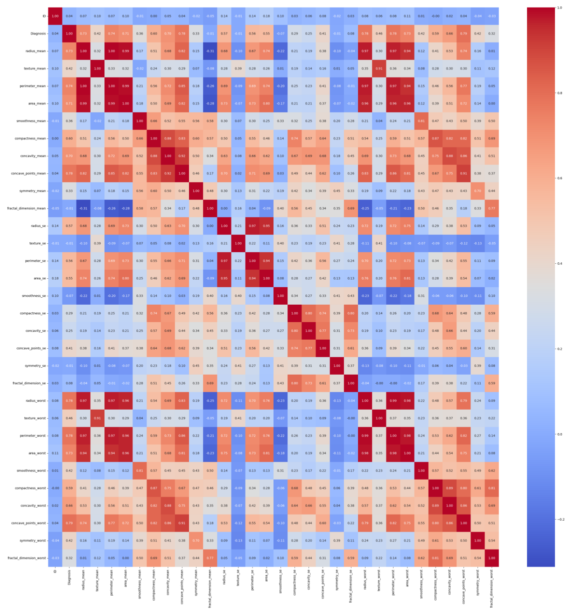

Figure 1: Feature's Correlation

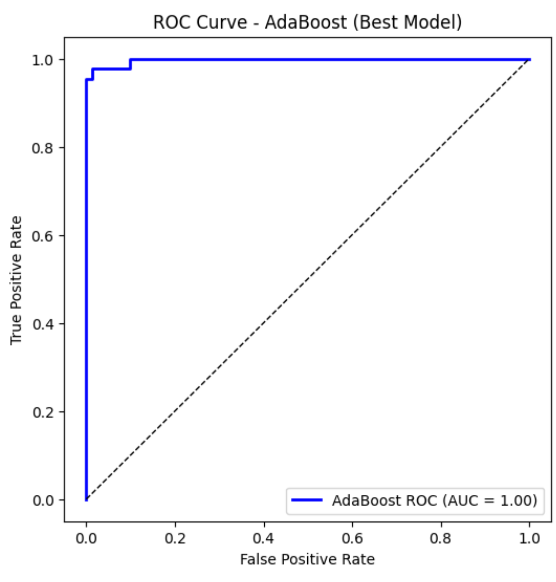

Figure 2: ROC Curve - AdaBoost Model

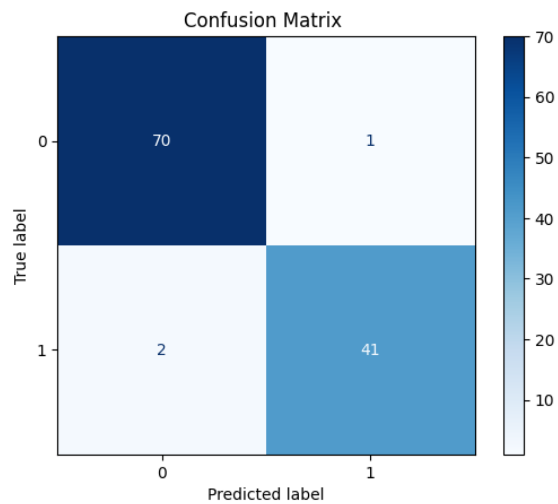

Figure 3: Confusion Matrix - AdaBoost Model

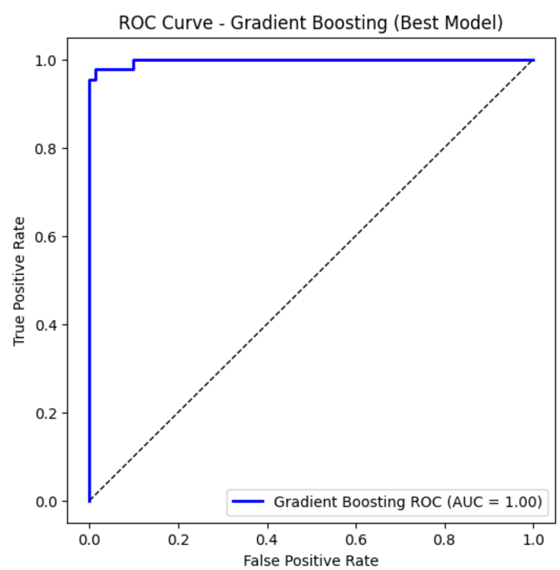

Figure 4: ROC Curve - Gradient Boosting Model

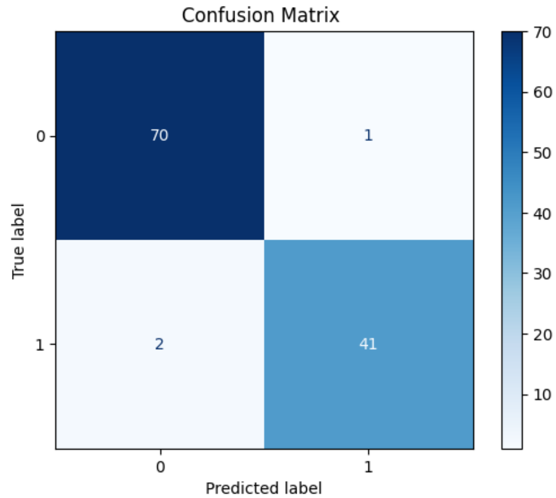

Figure 5: Confusion Matrix - Gradient Boosting Model

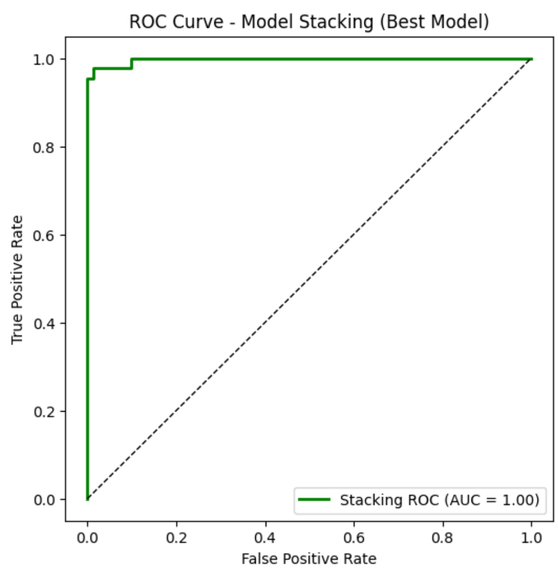

Figure 6: ROC Curve - Stacking Classifier

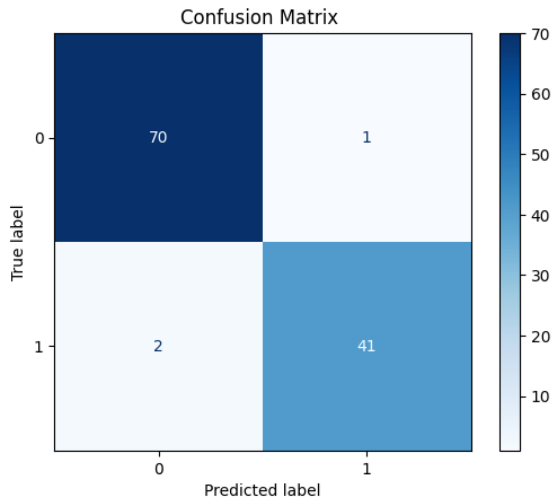

Figure 7: Confusion Matrix - Stacking Classifier

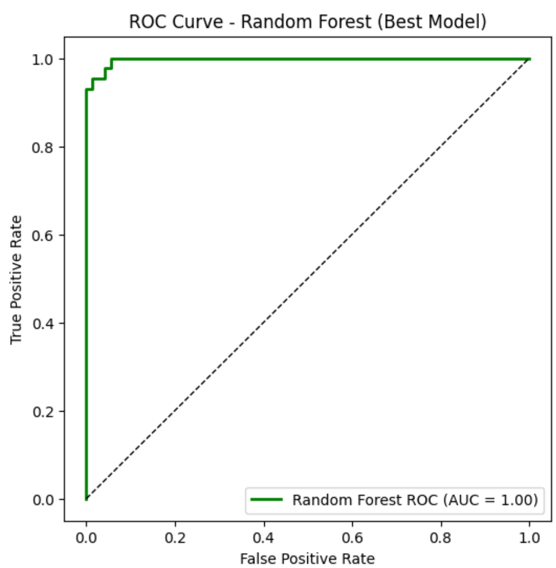

Figure 8: ROC Curve - Random Forest

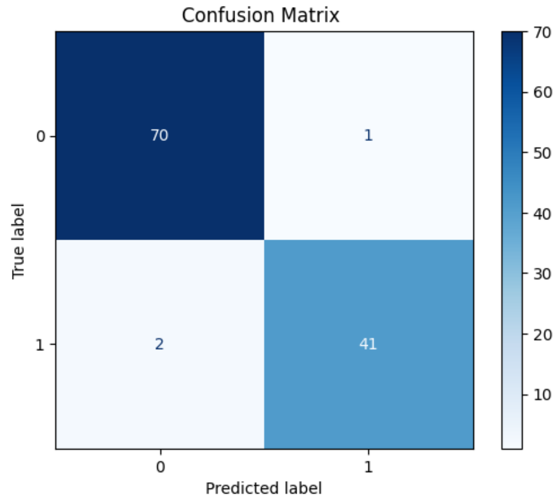

Figure 9: Confusion Matrix - Random Forest

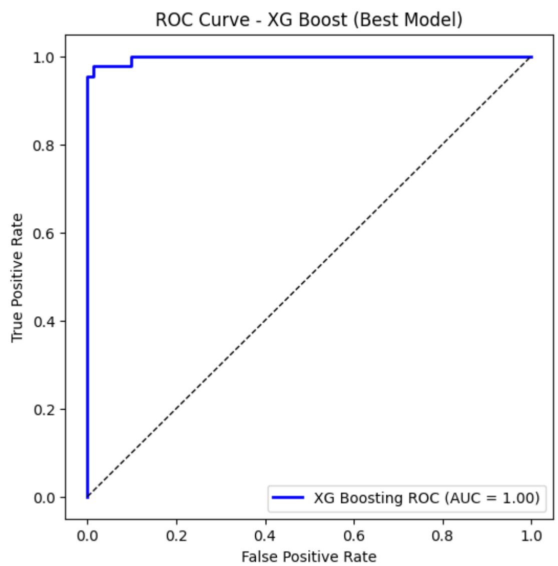

Figure 10: ROC Curve - XGBoost Model

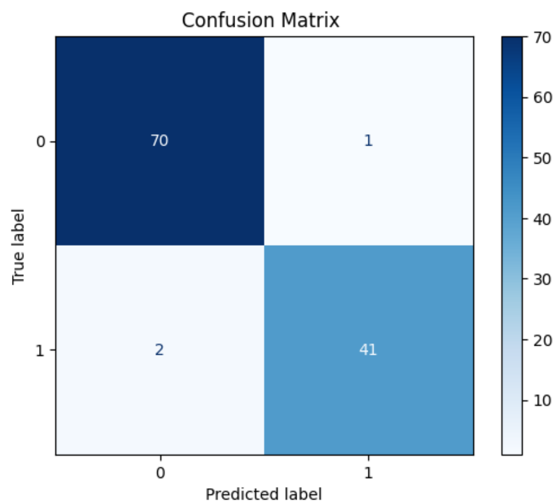

Figure 11: Confusion Matrix - XGBoost Model

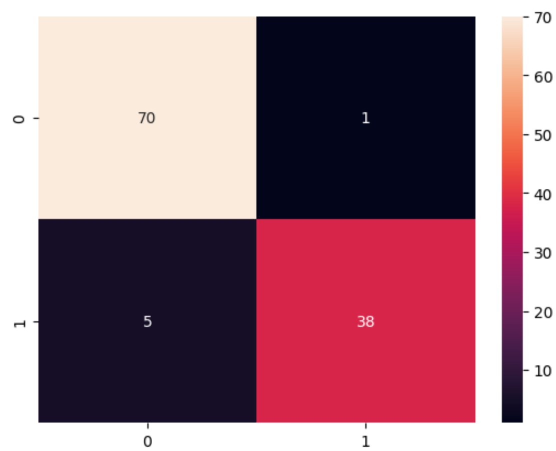

Figure 12: Confusion Matrix and Tree Visualization - Decision Tree Model

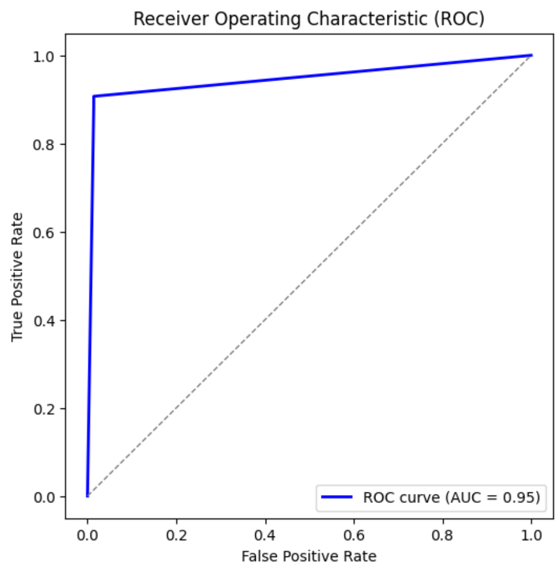

Figure 13: ROC - Decision Tree Model

Hyperparameter Tuning Results

Below are the hyperparameter tuning trials performed for different models.

Decision Tree

Table 1: Decision Tree - Hyperparameter Trials

Criterion	Max Depth	Accuracy	F1 Score
gini	5	0.91	0.90
entropy	10	0.93	0.92

AdaBoost

Table 2: AdaBoost - Hyperparameter Trials

n_estimators	Learning Rate	Accuracy	F1 Score
50	0.5	0.94	0.93
100	1.0	0.95	0.94

Gradient Boosting

Table 3: Gradient Boosting - Hyperparameter Trials

n_estimators	Learning Rate	Max Depth	Accuracy	F1 Score
100	0.1	3	0.96	0.95
200	0.05	5	0.97	0.96

XGBoost

Table 4: XGBoost - Hyperparameter Trials

n_estimators	Learning Rate	Max Depth	Gamma	Accuracy	F1 Score
100	0.1	3	0	0.97	0.96
200	0.05	4	1	0.98	0.97

Random Forest

Table 5: Random Forest - Hyperparameter Trials

n_estimators	Max Depth	Criterion	Min Samples Split	Accuracy	F1 Score
50	None	gini	2	0.95	0.94
100	20	entropy	5	0.96	0.95

Stacked Ensemble

Table 6: Stacked Ensemble - Hyperparameter Trials

Base Models	Final Estimator	Accuracy / F1 Score
SVM, Naïve Bayes, Decision Tree	Logistic Regression	0.97 / 0.96
SVM, Naïve Bayes, Decision Tree	Random Forest	0.98 / 0.97
SVM, Decision Tree, KNN	Logistic Regression	0.96 / 0.95

5-Fold Cross Validation Results

The following table shows fold-wise accuracy results for all models.

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

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg Accuracy
Decision Tree	0.91	0.90	0.92	0.89	0.91	0.906
AdaBoost	0.97	0.95	0.97	0.95	0.96	0.960
Gradient Boosting	0.98	0.96	1.00	0.98	0.94	0.972
XGBoost	0.97	0.97	1.00	0.97	0.95	0.972
Random Forest	0.97	0.95	0.97	0.95	0.96	0.960
Stacked Model	0.97	0.95	0.97	0.95	0.96	0.960

Learning Outcomes

- Gained a deeper understanding of the importance of hyperparameter tuning in improving model accuracy and F1 score across various algorithms.
- Learned how ensemble methods like AdaBoost, Gradient Boosting, XGBoost, and Stacked Models can significantly outperform single models such as Decision Trees by reducing variance and bias.
- Developed skills in using cross-validation to assess model stability and generalization capability, ensuring the chosen model performs well on unseen data.
- Acquired experience in interpreting evaluation metrics such as accuracy, F1 score, confusion matrices, and ROC curves to make informed model selection decisions.
- Improved proficiency in implementing and visualizing multiple models in a comparative study, including tuning and evaluation pipelines.

Observations

- **Overall Performance:** All ensemble models (AdaBoost, Gradient Boosting, XGBoost, Random Forest, and Stacked Model) performed significantly better than the standalone Decision Tree model.
- **Decision Tree:** Achieved an average accuracy of approximately **0.906**, the lowest among all models. This indicates that while the Decision Tree is simple and interpretable, it lacks robustness and generalization capability.
- **Ensemble Models:**
  - **AdaBoost and Random Forest:** Achieved average accuracies of around **0.960**, showing consistent and reliable performance.
  - **Gradient Boosting and XGBoost:** Both reached the highest average accuracy of approximately **0.972**, indicating superior performance and excellent generalization.
  - **Stacked Model:** Produced results comparable to AdaBoost and Random Forest, with an average accuracy of about **0.960**.
- **Consistency Across Folds:** Ensemble models showed less variation between folds compared to the Decision Tree, highlighting better stability and generalization.
- **Best Performers:** Gradient Boosting and XGBoost emerged as the top-performing models, making them ideal candidates for production deployment.