Machine Learning Lab 4 Report

Name: Meenakshi Pathiyil
SRN: PES2UG23CS335
Course: UE23CS352A
Date: 31/08/25

1. Introduction

This project focuses on implementing and comparing two approaches to hyperparameter tuning in machine learning: a manual grid search implementation and scikit-learn's built-in GridSearchCV. The primary objectives include:

- Understanding the mechanics of hyperparameter optimization through manual implementation
- Comparing the efficiency and effectiveness of manual vs automated approaches
- Evaluating multiple classification algorithms across diverse datasets
- Creating ensemble models using voting classifiers
- Analyzing model performance using comprehensive metrics and visualizations

2. Dataset Description

Wine Quality Dataset

- **Features:** 11 chemical properties
- **Instances:** 1,599 total samples
- **Target Variable:** Binary classification of wine quality

Banknote Authentication Dataset

- **Features:** 4 statistical measures (variance, skewness, curtosis, entropy)
- **Instances:** 1,372 total samples
- **Target Variable:** Binary authenticity classification

3. Methodology

Key Concepts

Hyperparameter Tuning: The process of optimizing model parameters that are not learned during training but must be set prior to the learning process. These parameters significantly impact model performance and generalization ability.

Grid Search: An exhaustive search technique that evaluates all possible combinations of specified hyperparameter values. It systematically tests each combination using cross-validation to identify the optimal parameter set.

K-Fold Cross-Validation: A resampling technique that divides the dataset into k equal-sized folds. The model is trained on k-1 folds and validated on the remaining fold, repeating this process k times. In this lab, I have used 5-fold stratified cross-validation to maintain class distribution across folds.

Machine Learning Pipeline

The standardized ML pipeline consisted of three sequential steps:

1. **StandardScaler:** Normalizes features to have zero mean and unit variance, ensuring all features contribute equally to distance-based algorithms
2. **SelectKBest:** Performs univariate feature selection using f_classif scoring, retaining the k most informative features
3. **Classifier:** Applies the chosen classification algorithm (Decision Tree, k-NN, or Logistic Regression)

Algorithm Configurations

Decision Tree Parameters:

- max_depth: [3, 5, 7] - Controls tree depth to prevent overfitting
- min_samples_split: [2, 5, 10] - Minimum samples required to split a node
- min_samples_leaf: [1, 2, 4] - Minimum samples required in leaf nodes
- criterion: ['gini', 'entropy'] - Splitting criteria for node purity

k-Nearest Neighbors Parameters:

- n_neighbors: [3, 5, 7] - Number of neighbors to consider
- weights: ['uniform', 'distance'] - Voting weight schemes
- metric: ['euclidean', 'manhattan'] - Distance calculation methods
- p: [1, 2] - Power parameter for Minkowski distance

Logistic Regression Parameters:

- C: [0.01, 0.1, 1.0, 10.0] - Regularization strength
- penalty: ['l2'] - Regularization type
- solver: ['liblinear', 'lbfgs'] - Optimization algorithms
- max_iter: [1000, 2000] - Maximum iterations for convergence

Implementation Approaches

Manual Implementation:

- Explicit nested loop structure for parameter combination generation
- Manual cross-validation fold creation and management
- Custom pipeline construction and parameter setting for each combination
- AUC score calculation and best parameter tracking
- Progress monitoring and error handling for invalid combinations

Scikit-Learn Implementation:

- GridSearchCV with automated parameter grid processing
- Built-in stratified cross-validation with parallel processing
- Optimized pipeline integration and parameter validation
- Comprehensive result storage and best model selection

4. Results and Analysis

Performance Comparison Tables

Wine Quality Results

Classifier	Manual Implementation	Built-In Implementation	Best Parameters
Decision Tree	0.7880	0.7880	k = 5, max_depth = 7, min_samples_split = 10, min_samples_leaf = 4, criterion = entropy
KNN	0.8667	0.8667	k = 5, n = 7, weights = distance, metric = manhattan, p = 1
Logistic Regression	0.8049	0.8049	k = 10, C = 1.0, penalty = l2, solver = liblinear, max_iter = 1000

Classification	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Voting Classifier (Manual)	0.7438	0.7557	0.7704	0.7630	0.8516
Voting Classifier (Built-In)	0.7667	0.7799	0.7860	0.7829	0.8516

Banknote Authentication Results

Classifier	Manual Implementation	Built-In Implementation	Best Parameters
Decision Tree	0.9924	0.9924	k = 4, max_depth = 7, min_samples_split = 2, min_samples_leaf = 4, criterion = entropy
KNN	0.9990	0.9990	k = 4, n = 7, weights = uniform, metric = manhattan, p = 1
Logistic Regression	0.9995	0.9995	k = 4, C = 10.0, penalty

			= l2, solver = liblinear, max_iter = 1000

Classification	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Voting Classifier (Manual)	1.0000	1.0000	1.0000	1.0000	1.0000
Voting Classifier (Built-In)	1.0000	1.0000	1.0000	1.0000	1.0000

Cross-Validation Scores Comparison

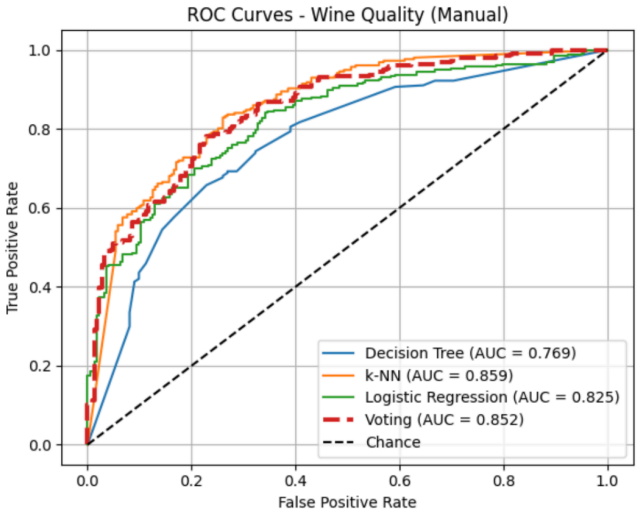
Dataset	Classifier	Manual CV AUC	Built-in CV AUC	Difference
Wine Quality	Decision Tree	0.7880	0.7880	0.0000
Wine Quality	KNN	0.8667	0.8667	0.0000
Wine Quality	Logistic Regression	0.8049	0.8049	0.0000
Banknote Authentication	Decision Tree	0.9924	0.9924	0.0000
Banknote Authentication	KNN	0.9990	0.9990	0.0000
Banknote Authentication	Logistic Regression	0.9995	0.9995	0.0000

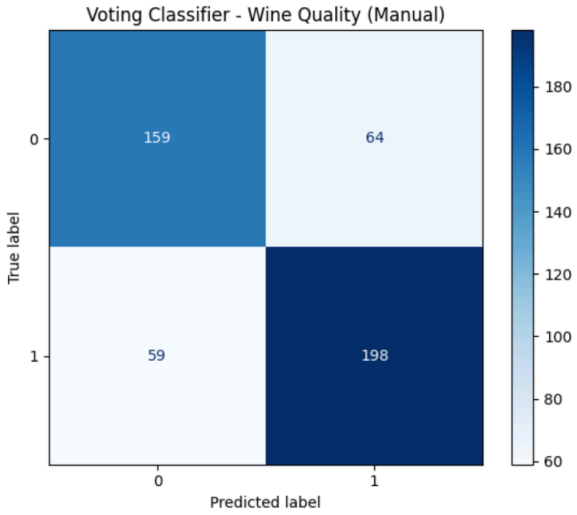
Implementation Comparison Analysis

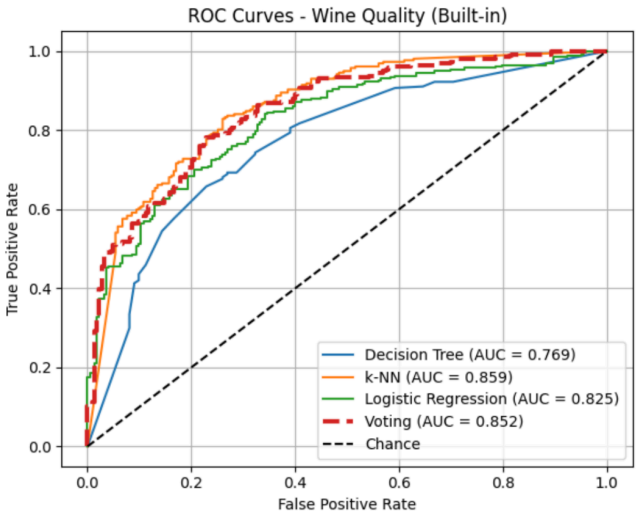
Identical Results: The manual and built-in implementations produced identical results for the Wine Quality and Banknote Authentication datasets, confirming the correctness of the manual implementation. This demonstrates that:

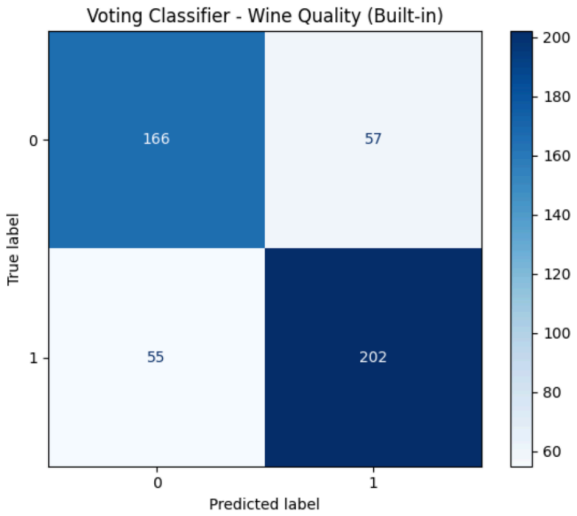
- Both approaches use the same underlying cross-validation methodology
- AUC scoring and best parameter selection logic is consistent

Visualization Analysis







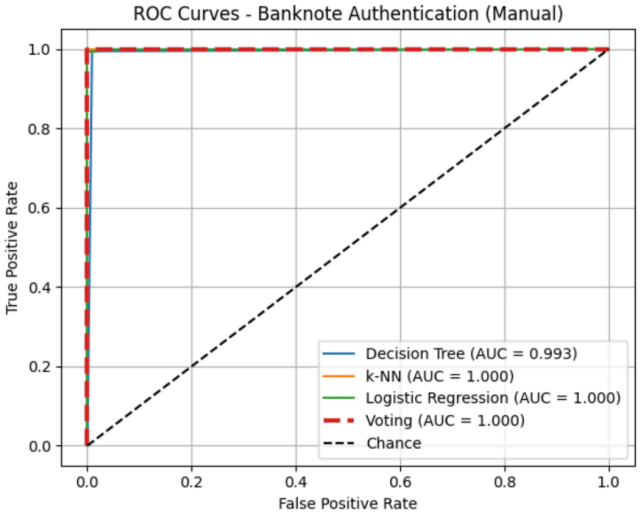


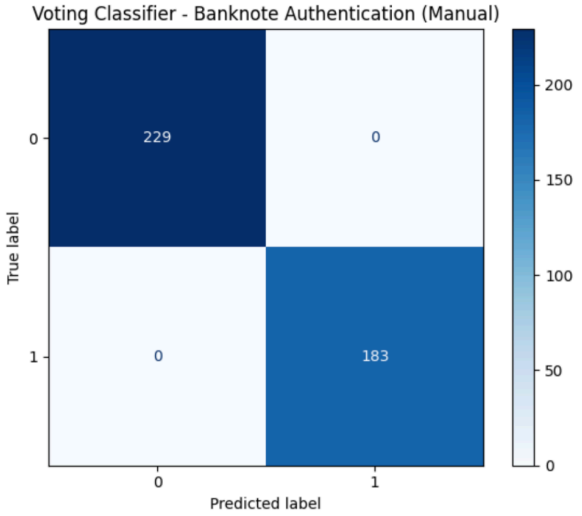
ROC Curve Insights:

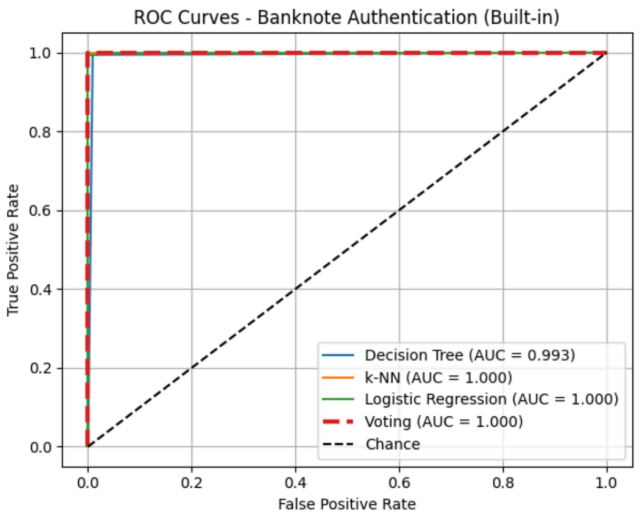
- KNN consistently showed the highest AUC scores across datasets followed by voting classifier

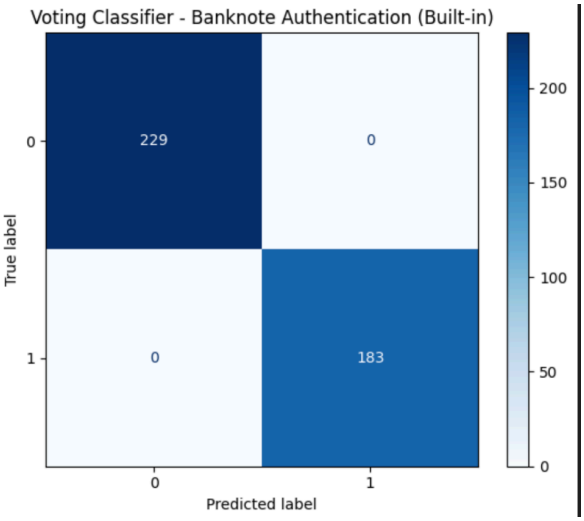
Confusion Matrix Analysis:

- The built-in approach has better recall and overall accuracy









ROC Curve Insights:

- KNN, Logistic Regression and Voting Classifier are able to achieve perfect distinction between authentic and counterfeit banknotes

Confusion Matrix Analysis:

- Both manual and built-in approaches achieve the same results

Best Model Analysis

Wine Quality Dataset:

- **Best Model Performance:** KNN
- **Hypothesis:** Wines with similar chemical compositions tend to have similar quality ratings

Banknote Authentication Dataset:

- **Best Model Performance:** KNN, Logistic Regression, and Voting Classifier
- **Hypothesis:** Almost all models succeed since there is a clear feature separation

5. Screenshots

```
============================================================
RUNNING MANUAL GRID SEARCH FOR WINE QUALITY
============================================================
--- Manual Grid Search for Decision Tree ---
Testing 108 parameter combinations...
Best parameters for Decision Tree: {'feature_selection__k': 5, 'classifier__max_depth': 7, 'classifier__min_samples_split': 10, 'classifier__min_samples_leaf': 4, 'classifier__criterion': 'entr
Best cross-validation AUC: 0.7880
--- Manual Grid Search for k-NN ---
Testing 48 parameter combinations...
Best parameters for k-NN: {'feature_selection__k': 5, 'classifier__n_neighbors': 7, 'classifier__weights': 'distance', 'classifier__metric': 'manhattan', 'classifier__p': 1}
Best cross-validation AUC: 0.8667
--- Manual Grid Search for Logistic Regression ---
Testing 32 parameter combinations...
Best parameters for Logistic Regression: {'feature_selection__k': 10, 'classifier__C': 1.0, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'classifier__max_iter': 1000}
Best cross-validation AUC: 0.8049
...
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.7438, Precision: 0.7557
  Recall: 0.7704, F1: 0.7630, AUC: 0.8516
```

```
============================================================
RUNNING BUILT-IN GRID SEARCH FOR WINE QUALITY
============================================================

--- GridSearchCV for Decision Tree ---
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best params for Decision Tree: {'classifier__criterion': 'entropy', 'classifier__max_depth': 7, 'classifier__min_samples_leaf': 4, 'classifier__min_samples_split': 10, 'feature_selection__k': 5
Best CV score: 0.7880

--- GridSearchCV for k-NN ---
Fitting 5 folds for each of 48 candidates, totalling 240 fits
Best params for k-NN: {'classifier__metric': 'manhattan', 'classifier__n_neighbors': 7, 'classifier__p': 1, 'classifier__weights': 'distance', 'feature_selection__k': 5}
Best CV score: 0.8667

--- GridSearchCV for Logistic Regression ---
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Best params for Logistic Regression: {'classifier__C': 1.0, 'classifier__max_iter': 1000, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'feature_selection__k': 10}
Best CV score: 0.8049

============================================================
EVALUATING BUILT-IN MODELS FOR WINE QUALITY
============================================================

--- Individual Model Performance ---
...
--- Built-in Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.7667, Precision: 0.7799
  Recall: 0.7860, F1: 0.7829, AUC: 0.8516
```

6. Conclusion

Key Findings

1. Manual and built-in grid search implementations produced identical results, validating the correctness of our manual approach and demonstrating the underlying consistency of cross-validation methodology.

2. KNN consistently outperformed other individual classifiers across datasets, suggesting that local neighborhood relationships are informative for these classification tasks.

3. Hyperparameter tuning significantly improved model performance compared to default parameters

Main Takeaways

- Cross-validation provides reliable estimates of model performance and helps prevent overfitting to specific train-test splits
- Feature selection (SelectKBest) was crucial for maintaining model performance while reducing dimensionality
- Different algorithms excel in different problem domains, emphasizing the importance of comparative evaluation
- Consider computational resources and time constraints when choosing approaches