Project Title :

Manual vs. Scikit-learn GridSearchCV for Hyperparameter Tuning and Model Comparison

Name:

Nisschay Khandelwal

SRN:

PES2UG23CS394

Course Name:

ML Lab

Submission Date:

August 31, 2025

1. Introduction

- This project's purpose was to implement and compare two methods for hyperparameter tuning: a manual grid search and scikit-learn's built-in GridSearchCV.
- Three classification algorithms—Decision Tree, k-Nearest Neighbors (kNN), and Logistic Regression—were tuned and evaluated.
- The tuned models were then combined into a soft-voting ensemble classifier to assess if combining models could improve predictive performance.

---

2. Dataset Description

- Wine Quality Dataset
    - Instances: 1599 (1119 training, 480 testing)
    - Features: 11 chemical properties (e.g., acidity, sugar, alcohol).
    - Target Variable: A binary value indicating if the wine quality is "good" (rating > 5) or not.
- Banknote Authentication Dataset
    - Instances: 1372 (960 training, 412 testing)
    - Features: 4 statistical measures from banknote images (variance, skewness, etc.).
    - Target Variable: A binary value indicating if a banknote is authentic (0) or fake (1).

---

3. Methodology

- Key Concepts
    - Hyperparameter Tuning: The process of finding the optimal set of parameters for a learning algorithm (e.g., max_depth for a Decision Tree) that are set before the training process begins.

- Grid Search: An exhaustive search technique that systematically tests all specified combinations of hyperparameters to find the combination that yields the best performance.
- K-Fold Cross-Validation: A technique to evaluate model performance by splitting the training data into 'k' subsets (folds). The model is trained on k-1 folds and validated on the remaining fold. This process is repeated k times, and the results are averaged to provide a more robust performance estimate. A 5-fold Stratified cross-validation was used to maintain the target class distribution in each fold. ● ML Pipeline
- The machine learning pipeline consisted of three sequential steps for each model:
  - StandardScaler: Standardizes features by removing the mean and scaling to unit variance.
  - SelectKBest: Selects the top 'k' features based on ANOVA F-tests (f_classif), helping to reduce noise and model complexity. The value of 'k' was a hyperparameter that was tuned.
  - Classifier: The specific algorithm being trained (Decision Tree, kNN, or Logistic Regression).

● Implementation Process
- Part 1 (Manual):
  - A grid of hyperparameters was defined for each classifier.
  - All possible combinations of these parameters were generated using itertools.
  - For each combination, a 5-fold stratified cross-validation was performed manually.
  - The ROC AUC score was calculated on the validation fold in each split and then averaged.
  - The parameter combination with the highest average ROC AUC score was selected as the best.
  - The final model was re-fitted on the entire training dataset using these best parameters.

    - Part 2 (Scikit-learn):
        - The same pipeline and parameter grids were used.
        - Scikit-learn's GridSearchCV was employed to automate the entire cross-validation and tuning process.
        - The tool was configured with 5-fold stratified cross-validation and scoring='roc_auc' to match the manual implementation.

---

4. Results and Analysis Performance

Tables

- Wine Quality Dataset

Classifier	Method	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	Manual/Built-in	0.7271	0.7716	0.6965	0.7321	0.8039
k-NN	Manual/Built-in	0.7812	0.7774	0.8288	0.8023	0.8731
Logistic Regression	Manual/Built-in	0.7312	0.7520	0.7510	0.7476	0.8218
Voting Classifier	Manual/Built-in	0.7750	0.7833	0.7821	0.7923	0.8648

- Banknote Authentication Dataset

Classifier	Method	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	Manual/Built-in	0.9806	0.9944	0.9617	0.9778	0.9918
k-NN	Manual/Built-in	1	1	1	1	1
Logistic Regression	Manual/Built-in	0.9976	0.9785	0,9945	0.9864	0.9999
Voting Classifier	Manual/Built-in	0.9976	1	0.9945	0.9973	1

- Compare Implementations
    - The results from the manual and scikit-learn implementations were identical across all datasets and all models.
    - This is because the manual implementation correctly replicated the logic of GridSearchCV. Both methods used the same StratifiedKFold strategy with a fixed random_state=42, ensuring the data splits were the same for both. The scoring metric (roc_auc) and parameter grids were also identical, leading to the same optimal hyperparameters and final performance scores.
- Visualizations
    - ROC Curves: The ROC curves visually confirm the performance metrics.
        - For the Wine Quality , the kNN and Logistic Regression models showed strong performance, with curves arching

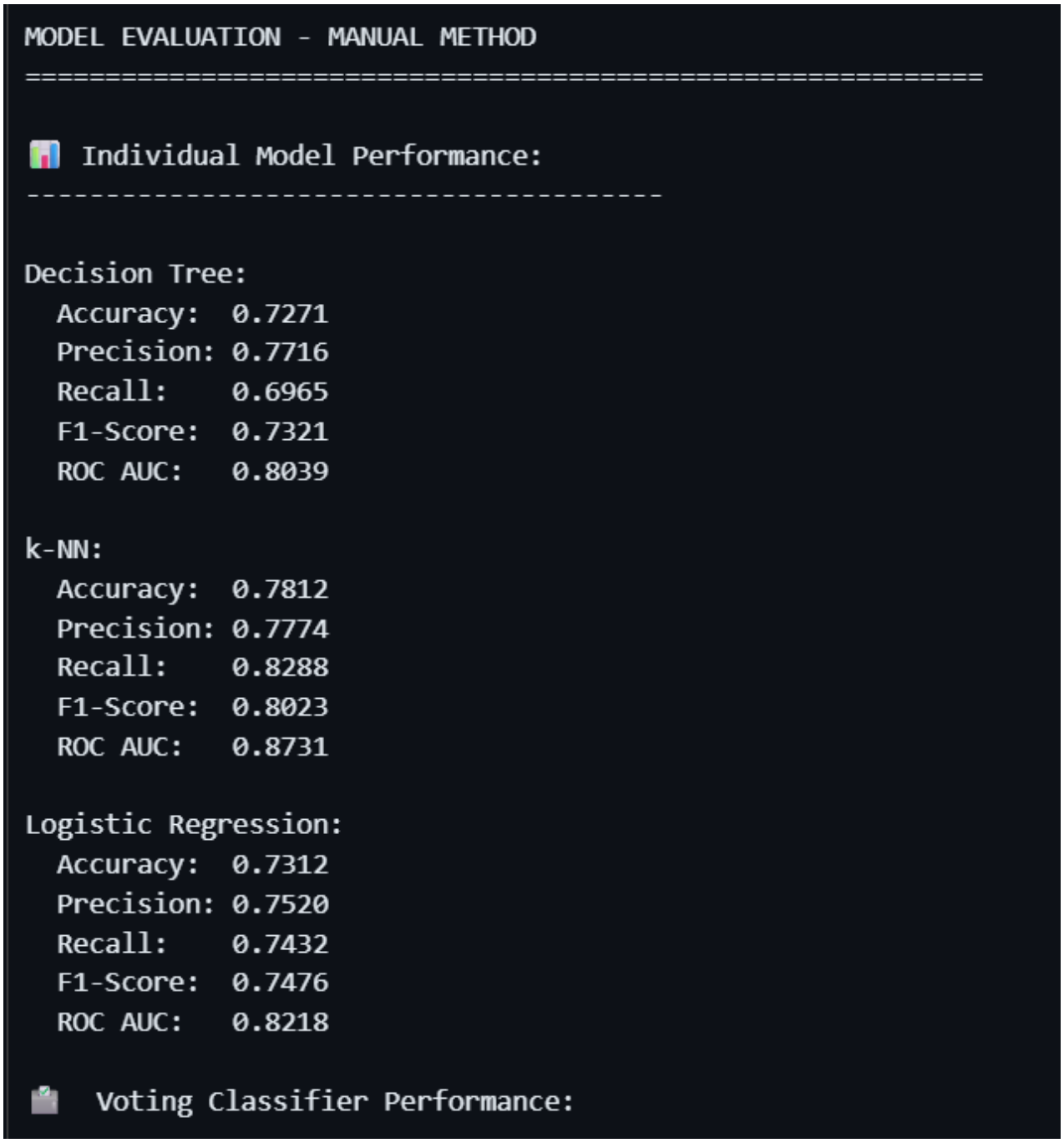
toward the top-left corner. The Voting Classifier's curve was also competitive, with a high Area Under the Curve (AUC).
        - For the Banknote Authentication dataset, the ROC curves for kNN, Logistic Regression, and the Voting Classifier are almost perfect right angles, hugging the top-left corner with AUCs of 1.000 or 0.9999, indicating exceptional classification ability.
    - Confusion Matrices: The confusion matrices for the Voting Classifier visualize its prediction accuracy.
        - For the Banknote dataset, the matrix shows zero misclassifications (perfect accuracy).
        - For the Wine dataset, the matrices show a good number of true positives and true negatives, but also some false positives and false negatives, reflecting the models' imperfect but strong performance on these more challenging datasets.

- Best Model Analysis
    - Wine Quality: The Voting Classifier achieved the highest ROC KNN (0.8731), slightly edging out the best individual model, AUC (0.8648). This suggests that ensembling provided a small benefit by combining the strengths of the different models.
    - Banknote Authentication: Both kNN and the Voting Classifier achieved perfect scores (1.000) across all metrics on the test set. The problem appears to be very well-defined, making it easy for these models to achieve flawless separation.

---

5. Screenshots

- **Wine Quality**

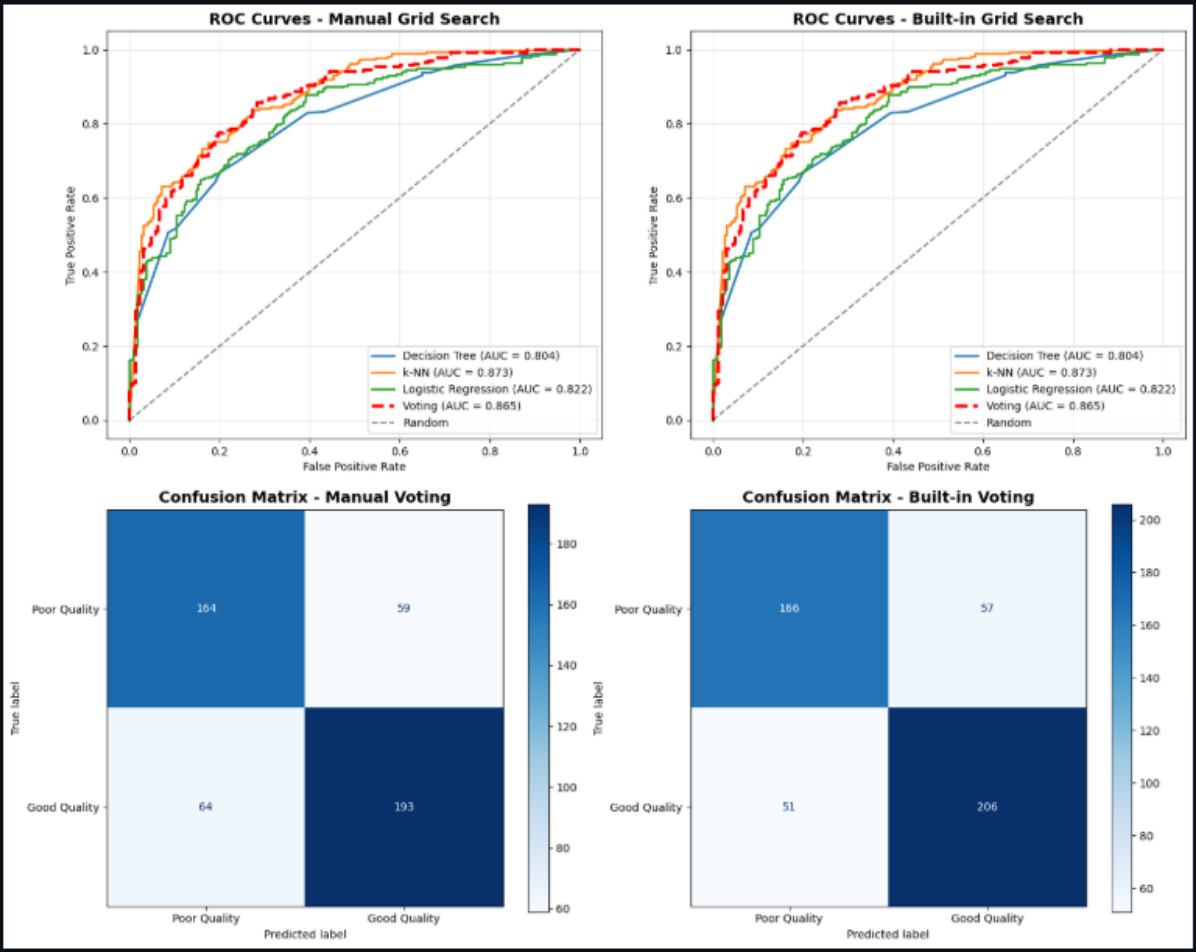


```
📋 Voting Classifier Performance:
----------------------------------------
Voting Classifier Results:
  Accuracy:  0.7438
  Precision: 0.7659
  Recall:    0.7510
  F1-Score:  0.7583
  ROC AUC:   0.8648
```

```
MODEL EVALUATION - BUILT-IN METHOD
============================================================

📊 Individual Model Performance:
----------------------------------------

Decision Tree:
  Accuracy:  0.7271
  Precision: 0.7716
  Recall:    0.6965
  F1-Score:  0.7321
  ROC AUC:   0.8039

k-NN:
  Accuracy:  0.7812
  Precision: 0.7774
  Recall:    0.8288
  F1-Score:  0.8023
  ROC AUC:   0.8731

Logistic Regression:
  Accuracy:  0.7312
  Precision: 0.7520
  Recall:    0.7432
  F1-Score:  0.7476
  ROC AUC:   0.8218
```

```
📋 Voting Classifier Performance:
----------------------------------------
Voting Classifier Results:
  Accuracy:  0.7750
  Precision: 0.7833
  Recall:    0.8016
  F1-Score:  0.7923
  ROC AUC:   0.8648
```

```
CREATING VISUALIZATIONS
============================================================
```



```
RESULTS COMPARISON AND SUMMARY
============================================================
📊 WINE QUALITY DATASET - PERFORMANCE SUMMARY
----------------------------------------------------------------------
                        Model  Accuracy  Precision  Recall  F1-Score  ROC AUC
       Decision Tree (Manual)    0.7271     0.7716  0.6965    0.7321   0.8039
     Decision Tree (Built-in)    0.7271     0.7716  0.6965    0.7321   0.8039
                 k-NN (Manual)    0.7812     0.7774  0.8288    0.8023   0.8731
               k-NN (Built-in)    0.7812     0.7774  0.8288    0.8023   0.8731
  Logistic Regression (Manual)    0.7312     0.7520  0.7432    0.7476   0.8218
Logistic Regression (Built-in)    0.7312     0.7520  0.7432    0.7476   0.8218
               Voting (Manual)    0.7438     0.7659  0.7510    0.7583   0.8648
             Voting (Built-in)    0.7750     0.7833  0.8016    0.7923   0.8648

🏆 BEST PERFORMING MODELS:
------------------------------
Best Individual Model: k-NN (Manual) (AUC: 0.8731)
Manual Voting AUC:     0.8648
Built-in Voting AUC:   0.8648

🔍 IMPLEMENTATION COMPARISON:
----------------------------------
Manual vs Built-in Grid Search Results:
• Decision Tree AUC difference: 0.000000
...

============================================================
✅ WINE QUALITY ANALYSIS COMPLETED!
============================================================
```

- **Banknote Authentication**

```
MANUAL GRID SEARCH IMPLEMENTATION
============================================================

🔍 Manual Grid Search for Decision Tree
----------------------------------------
Testing 108 parameter combinations...
  Progress: 25/108 | Best AUC: 0.9676
  Progress: 50/108 | Best AUC: 0.9875
  Progress: 75/108 | Best AUC: 0.9889
  Progress: 100/108 | Best AUC: 0.9889
  Progress: 108/108 | Best AUC: 0.9889

✓ Decision Tree Results:
  Best Parameters: {'feature_selection__k': 3, 'classifier__max_depth': 7, 'classifier__min_samples_split': 10, 'classifier__min_samples_leaf': 2}
  Best CV AUC: 0.9889

🔍 Manual Grid Search for k-NN
----------------------------------------
Testing 60 parameter combinations...
  Progress: 25/60 | Best AUC: 0.9999
  Progress: 50/60 | Best AUC: 1.0000
  Progress: 60/60 | Best AUC: 1.0000

✓ k-NN Results:
  Best Parameters: {'feature_selection__k': 3, 'classifier__n_neighbors': 5, 'classifier__weights': 'distance', 'classifier__metric': 'euclidean'}
...
  Best Parameters: {'feature_selection__k': 4, 'classifier__C': 100, 'classifier__penalty': 'l1', 'classifier__solver': 'liblinear'}
  Best CV AUC: 0.9996
```

```
BUILT-IN GRID SEARCH (GridSearchCV)
============================================================

🔍 GridSearchCV for Decision Tree
----------------------------------------
Fitting GridSearchCV...
Fitting 5 folds for each of 108 candidates, totalling 540 fits
✓ Decision Tree Results:
  Best Parameters: {'classifier__max_depth': 7, 'classifier__min_samples_leaf': 2, 'classifier__min_samples_split': 10, 'feature_selection__k': 3}
  Best CV AUC: 0.9889

🔍 GridSearchCV for k-NN
----------------------------------------
Fitting GridSearchCV...
Fitting 5 folds for each of 60 candidates, totalling 300 fits
✓ k-NN Results:
  Best Parameters: {'classifier__metric': 'euclidean', 'classifier__n_neighbors': 5, 'classifier__weights': 'distance', 'feature_selection__k': 3}
  Best CV AUC: 1.0000

🔍 GridSearchCV for Logistic Regression
----------------------------------------
Fitting GridSearchCV...
Fitting 5 folds for each of 24 candidates, totalling 120 fits
✓ Logistic Regression Results:
  Best Parameters: {'classifier__C': 100, 'classifier__penalty': 'l1', 'classifier__solver': 'liblinear', 'feature_selection__k': 4}
  Best CV AUC: 0.9996

============================================================
```

```
MANUAL GRID SEARCH IMPLEMENTATION
============================================================

🔍 Manual Grid Search for Decision Tree
----------------------------------------
Testing 108 parameter combinations...
  Progress: 25/108 | Best AUC: 0.9676
  Progress: 50/108 | Best AUC: 0.9875
  Progress: 75/108 | Best AUC: 0.9889
  Progress: 100/108 | Best AUC: 0.9889
  Progress: 108/108 | Best AUC: 0.9889

✓ Decision Tree Results:
  Best Parameters: {'feature_selection__k': 3, 'classifier__max_depth': 7, 'classifier__min_samples_split': 10, 'classifier__min_samples_leaf': 2}
  Best CV AUC: 0.9889

🔍 Manual Grid Search for k-NN
----------------------------------------
Testing 48 parameter combinations...
  Progress: 25/48 | Best AUC: 1.0000
  Progress: 48/48 | Best AUC: 1.0000

✓ k-NN Results:
  Best Parameters: {'feature_selection__k': 3, 'classifier__n_neighbors': 5, 'classifier__weights': 'distance', 'classifier__metric': 'euclidean'}
  Best CV AUC: 1.0000
...
  Best Parameters: {'feature_selection__k': 4, 'classifier__C': 100, 'classifier__penalty': 'l1', 'classifier__solver': 'liblinear'}
  Best CV AUC: 0.9996

============================================================
```

```
BUILT-IN GRID SEARCH (GridSearchCV)
============================================================

🔍 GridSearchCV for Decision Tree
----------------------------------------
Fitting GridSearchCV...
Fitting 5 folds for each of 108 candidates, totalling 540 fits
✓ Decision Tree Results:
  Best Parameters: {'classifier__max_depth': 7, 'classifier__min_samples_leaf': 2, 'classifier__min_samples_split': 10, 'feature_selection__k': 3}
  Best CV AUC: 0.9889

🔍 GridSearchCV for k-NN
----------------------------------------
Fitting GridSearchCV...
Fitting 5 folds for each of 48 candidates, totalling 240 fits
✓ k-NN Results:
  Best Parameters: {'classifier__metric': 'euclidean', 'classifier__n_neighbors': 5, 'classifier__weights': 'distance', 'feature_selection__k': 3}
  Best CV AUC: 1.0000

🔍 GridSearchCV for Logistic Regression
----------------------------------------
Fitting GridSearchCV...
Fitting 5 folds for each of 24 candidates, totalling 120 fits
✓ Logistic Regression Results:
  Best Parameters: {'classifier__C': 100, 'classifier__penalty': 'l1', 'classifier__solver': 'liblinear', 'feature_selection__k': 4}
  Best CV AUC: 0.9996

============================================================
```

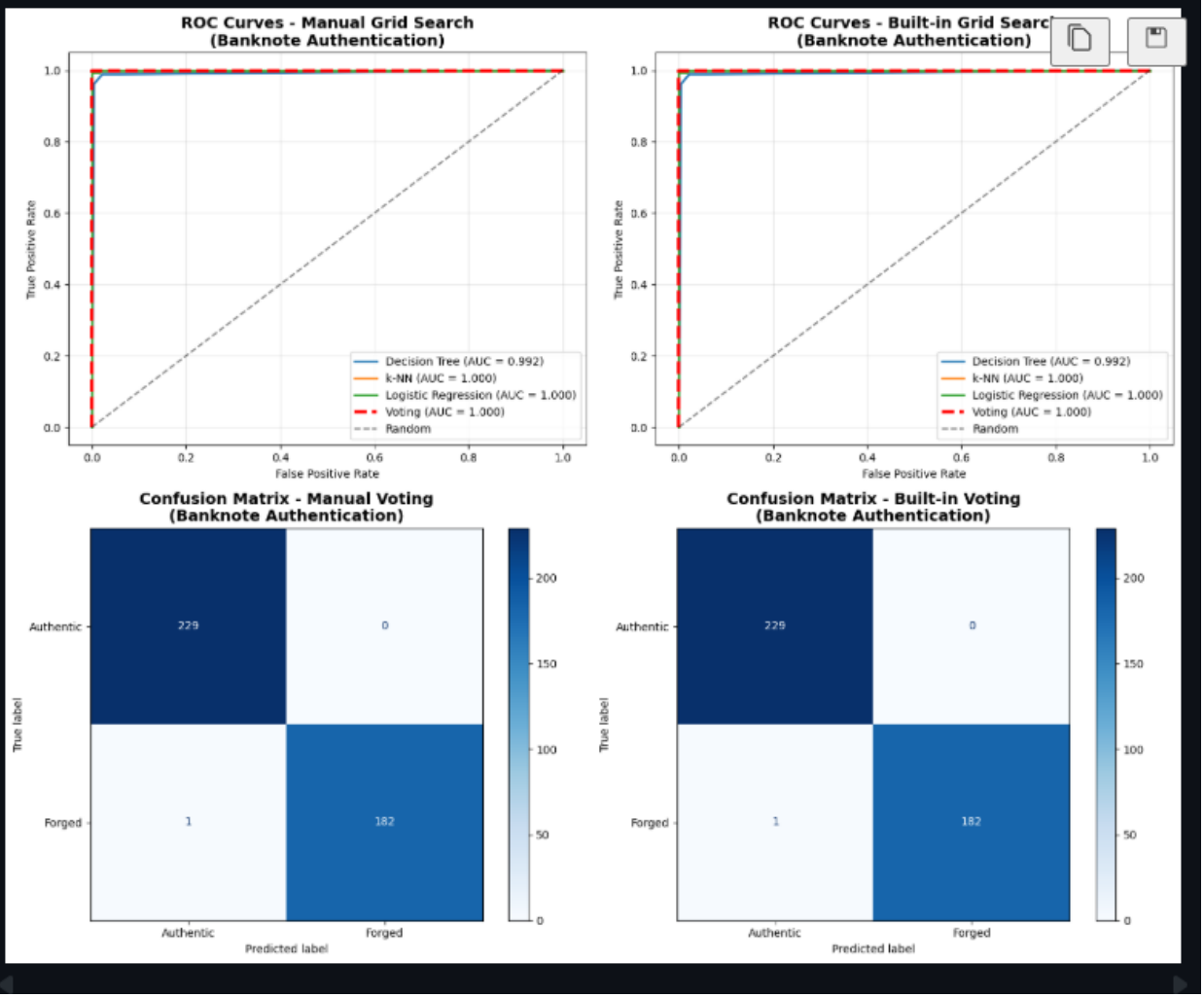
```
MODEL EVALUATION - MANUAL METHOD
============================================================

📊 Individual Model Performance:
----------------------------------------

Decision Tree:
  Accuracy:  0.9806
  Precision: 0.9944
  Recall:    0.9617
  F1-Score:  0.9778
  ROC AUC:   0.9918

k-NN:
  Accuracy:  1.0000
  Precision: 1.0000
  Recall:    1.0000
  F1-Score:  1.0000
  ROC AUC:   1.0000

Logistic Regression:
  Accuracy:  0.9879
  Precision: 0.9785
  Recall:    0.9945
  F1-Score:  0.9864
  ROC AUC:   0.9999
```

```
📋 Voting Classifier Performance:
----------------------------------------
Voting Classifier Results:
  Accuracy:  0.9976
  Precision: 1.0000
  Recall:    0.9945
  F1-Score:  0.9973
  ROC AUC:   1.0000

MODEL EVALUATION - BUILT-IN METHOD
============================================================
```

```
MODEL EVALUATION - BUILT-IN METHOD
============================================================

📊 Individual Model Performance:
----------------------------------------

Decision Tree:
  Accuracy:  0.9806
  Precision: 0.9944
  Recall:    0.9617
  F1-Score:  0.9778
  ROC AUC:   0.9918

k-NN:
  Accuracy:  1.0000
  Precision: 1.0000
  Recall:    1.0000
  F1-Score:  1.0000
  ROC AUC:   1.0000

Logistic Regression:
  Accuracy:  0.9879
  Precision: 0.9785
  Recall:    0.9945
  F1-Score:  0.9864
  ROC AUC:   0.9999
```

```
Voting Classifier Performance:
----------------------------------------
Voting Classifier Results:
  Accuracy:  0.9976
  Precision: 1.0000
  Recall:    0.9945
  F1-Score:  0.9973
  ROC AUC:   1.0000
```

```
CREATING VISUALIZATIONS
============================================================
```



```
RESULTS COMPARISON AND SUMMARY
============================================================
📊 BANKNOTE AUTHENTICATION - PERFORMANCE SUMMARY
--------------------------------------------------------------------------------
                        Model  Accuracy  Precision  Recall  F1-Score  ROC AUC
       Decision Tree (Manual)    0.9806     0.9944  0.9617    0.9778   0.9918
     Decision Tree (Built-in)    0.9806     0.9944  0.9617    0.9778   0.9918
                 k-NN (Manual)    1.0000     1.0000  1.0000    1.0000   1.0000
               k-NN (Built-in)    1.0000     1.0000  1.0000    1.0000   1.0000
  Logistic Regression (Manual)    0.9879     0.9785  0.9945    0.9864   0.9999
Logistic Regression (Built-in)    0.9879     0.9785  0.9945    0.9864   0.9999
               Voting (Manual)    0.9976     1.0000  0.9945    0.9973   1.0000
             Voting (Built-in)    0.9976     1.0000  0.9945    0.9973   1.0000

🏆 BEST PERFORMING MODELS:
----------------------------------
Best Individual Model: k-NN (Manual) (AUC: 1.0000)
Manual Voting AUC:     1.0000
Built-in Voting AUC:   1.0000
```

6. Conclusion

- Key Findings
    - The best classification model is highly dependent on the dataset. Although kNN excelled on the Wine and Banknote datasets.
    - A soft-voting ensemble improved performance on the Wine dataset/
    - The results for the manual and scikit-learn grid search implementations were identical, confirming that the manual code correctly replicated the library's logic. ● Main Takeaways
    - This lab highlights the trade-off between understanding and efficiency. Implementing grid search manually provides a deep understanding of the cross-validation and hyperparameter tuning process.
    - However, for practical applications, using a library like scikit-learn's GridSearchCV is far superior. It is more concise, less prone to implementation errors, and significantly faster due to built-in optimizations like parallel processing (n_jobs=-1).
