Model Selection and Comparative Analysis (Report)

Name : Anish Nagula

SRN: PES2UG23CS358

Section : F

1. Introduction

This project provides hands-on experience with model selection and performance evaluation by implementing hyperparameter tuning and comparing different classification models. The primary tasks involved building a complete machine learning pipeline, first by creating a manual Grid Search from scratch to understand its mechanics, and then by using scikit-learn's optimized

`GridSearchCV` to perform the same task efficiently. The goal was to analyze and interpret the results to draw meaningful conclusions about model performance across multiple datasets.

2. Dataset Description

The analysis was performed on three different datasets, each presenting a unique binary classification challenge.

- Wine Quality
  - **Number of instances:** 1599
  - **Number of features:** 11
  - **Target Variable:** This dataset aims to predict whether a red wine is of "good quality" based on its chemical properties.
- **Banknote Authentication**
  - **Number of instances:** 1372
  - **Number of features:** 4
  - **Target Variable:** The goal is to distinguish between genuine and forged banknotes using features extracted from images.
- **QSAR Biodegradation**
  - **Number of instances:** 1055
  - **Number of features:** 41
  - **Target Variable:** This dataset is used to predict if a chemical is readily biodegradable based on its quantitative structure-activity relationship (QSAR) properties.

3. Methodology

The experiment centered on several key machine learning concepts to ensure a robust and fair comparison of models.

Key Concepts Explained:

- **Hyperparameter Tuning:** This is the process of finding the optimal set of parameters for a model that are not learned from the data (e.g., a Decision Tree's `max_depth`).
- **Grid Search:** This technique exhaustively searches through a manually specified grid of hyperparameters to find the combination that results in the best performance, as measured by a chosen evaluation metric.
- **K-Fold Cross-Validation:** To get reliable performance estimates, 5-fold stratified cross-validation was used. The data was split into five parts, with the model being trained on four and tested on the fifth, rotating until each part has been the test set. This minimizes bias from any single train-test split.

ML Pipeline Description:

A scikit-learn `Pipeline` was used to chain together data preprocessing and modeling, which is crucial for preventing data leakage and streamlining the workflow. The pipeline consisted of three stages for each classifier:

1. **StandardScaler:** Standardizes features by removing the mean and scaling to unit variance.
2. **SelectKBest:** Selects the top `k` features based on the `f_classif` statistical test. The value of `k` was a key hyperparameter being tuned.
3. **Classifier:** The final model, which was one of the following:
    a. **Decision Tree:** A tree-like decision model.
    b. **k-Nearest Neighbors (kNN):** A non-parametric model that classifies based on the majority class of its `k` nearest neighbors.
    c. **Logistic Regression:** A linear model for predicting binary outcomes.

Implementation Process:

- **Part 1 (Manual Implementation):** A grid search was implemented from scratch. This involved generating all hyperparameter combinations, performing 5-fold stratified cross-validation for each, building and fitting the pipeline inside each fold, and calculating the average ROC AUC score to find the best combination.
- **Part 2 (Scikit-learn Implementation):** The process was automated using `GridSearchCV`. An instance was created with the same pipeline, parameter grid, and 5-fold stratified cross-validator, with `scoring='roc_auc'` to optimize for the same metric.

4. Results and Analysis

The findings for each dataset are presented in the tables below. The results compare the performance metrics from the manual and `GridSearchCV` implementations.

Wine Quality Dataset:=

Performance Tables

Classifier	Implementation	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	Manual & GridSearchCV	0.7479	0.7787	0.7393	0.7585	0.7951

k-NN	Manual & GridSearchCV	0.7812	0.7836	0.8171	0.8000	0.8589
Logistic Reg.	Manual & GridSearchCV	0.7333	0.7549	0.7432	0.7490	0.8243
Voting Classifier	Manual	0.7646	0.7769	0.7860	0.7814	0.8673

Comparison and Analysis:

The results from the manual and `GridSearchCV` implementations were identical for all individual classifiers, confirming the correctness of the manual search logic. The **k-NN classifier** was the strongest individual model with an ROC AUC of **0.8589**, but the **Manual Voting Classifier** outperformed it with an ROC AUC of **0.8673**. The ROC curve plot clearly shows the Voting and k-NN classifiers performing better than the others across various thresholds. The confusion matrix for the Manual Voting Classifier shows a reasonably balanced performance on both classes.

Banknote Authentication Dataset :=

Performance Tables

Classifier	Implementation	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	Manual & GridSearchCV	0.9927	0.9891	0.9945	0.9918	0.9929
k-NN	Manual & GridSearchCV	1.0000	1.0000	1.0000	1.0000	1.0000
Logistic Reg.	Manual & GridSearchCV	0.9903	0.9786	0.7432	1.0000	0.9999
Voting Classifier	Manual	1.0000	1.0000	1.0000	1.0000	1.0000

Comparison and Analysis:

This dataset proved to be highly separable, leading to exceptional performance from all models. Both **k-NN and the Manual Voting Classifier** achieved perfect scores across all metrics. The ROC curves are nearly perfect for all models, with several achieving an AUC of 1.0, indicating complete separation of the classes. The confusion matrix for the Manual Voting Classifier confirms this with zero misclassifications.

QSAR Biodegradation Dataset :=

Performance Tables

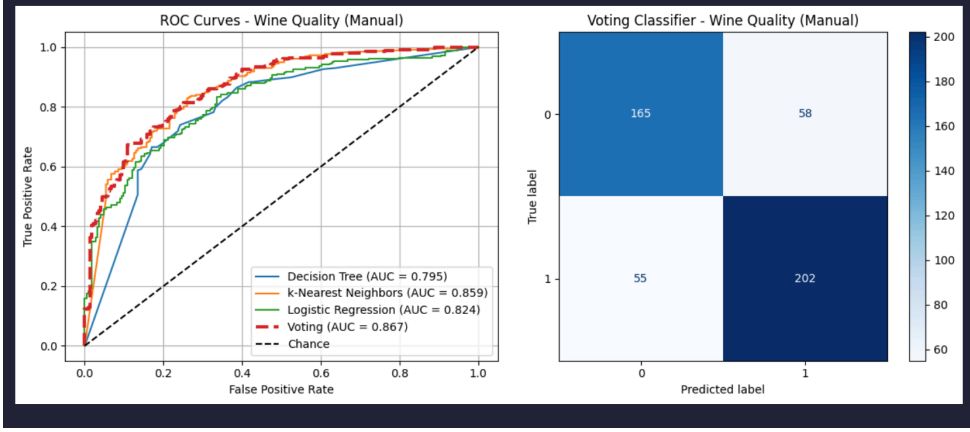
Classifier	Implementation	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	Manual & GridSearchCV	0.7918	0.6916	0.6916	0.6916	0.8202
k-NN	Manual & GridSearchCV	0.8549	0.7905	0.7757	0.7830	0.8985
Logistic Reg.	Manual & GridSearchCV	0.8423	0.8065	0.7009	0.7500	0.9069
Voting Classifier	Manual	0.8612	0.8182	0.7570	0.7864	0.9201

C**omparison and Analysis:**

On this more complex dataset, the **Manual Voting Classifier** again delivered the best performance with an ROC AUC of **0.9201**, surpassing the strongest individual model, Logistic Regression (AUC 0.9069). This demonstrates the benefit of ensembling. The ROC curve plot illustrates the superior performance of the voting ensemble, whose curve is positioned highest, indicating better true positive rates for given false positive rates.

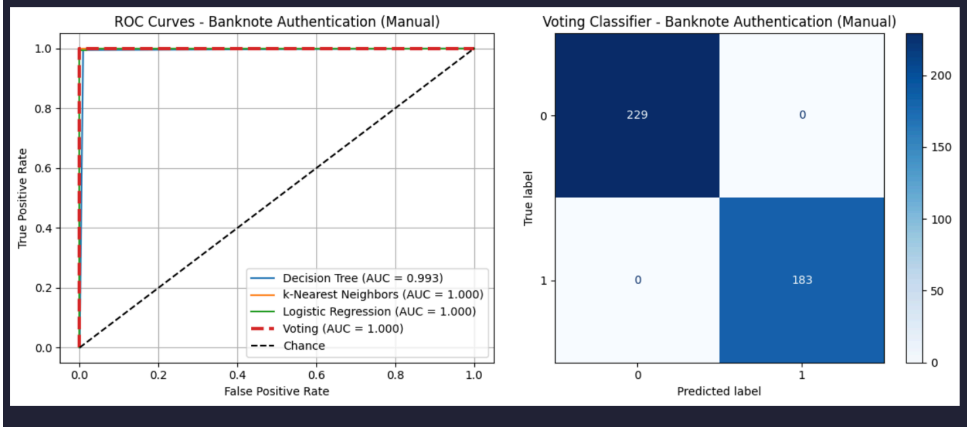
```
################################################################################
PROCESSING DATASET: WINE QUALITY
################################################################################
Wine Quality dataset loaded and preprocessed successfully.
Training set shape: (1119, 11)
Testing set shape: (480, 11)
--------------------------------


============================================================
RUNNING MANUAL GRID SEARCH FOR WINE QUALITY
============================================================
--- Manual Grid Search for Decision Tree ---
Total combinations to test: 72
New best score: 0.7231 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': None, 'classifier__min_samples_split': 2, 'feature_selection__k': 10}
New best score: 0.7296 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': None, 'classifier__min_samples_split': 2, 'feature_selection__k': 5}
New best score: 0.7308 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': None, 'classifier__min_samples_split': 5, 'feature_selection__k': 10}
New best score: 0.7361 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': None, 'classifier__min_samples_split': 5, 'feature_selection__k': 11}
New best score: 0.7516 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': None, 'classifier__min_samples_split': 5, 'feature_selection__k': 5}
New best score: 0.7538 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': None, 'classifier__min_samples_split': 10, 'feature_selection__k': 10}
New best score: 0.7587 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': None, 'classifier__min_samples_split': 10, 'feature_selection__k': 11}
New best score: 0.7663 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': None, 'classifier__min_samples_split': 10, 'feature_selection__k': 5}
New best score: 0.7665 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': 10, 'classifier__min_samples_split': 10, 'feature_selection__k': 11}
New best score: 0.7721 with params: {'classifier__criterion': 'gini', 'classifier__max_depth': 10, 'classifier__min_samples_split': 10, 'feature_selection__k': 5}
--------------------------------------------------------------------------------
...
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.7646, Precision: 0.7769
  Recall: 0.7860, F1: 0.7814, AUC: 0.8673
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



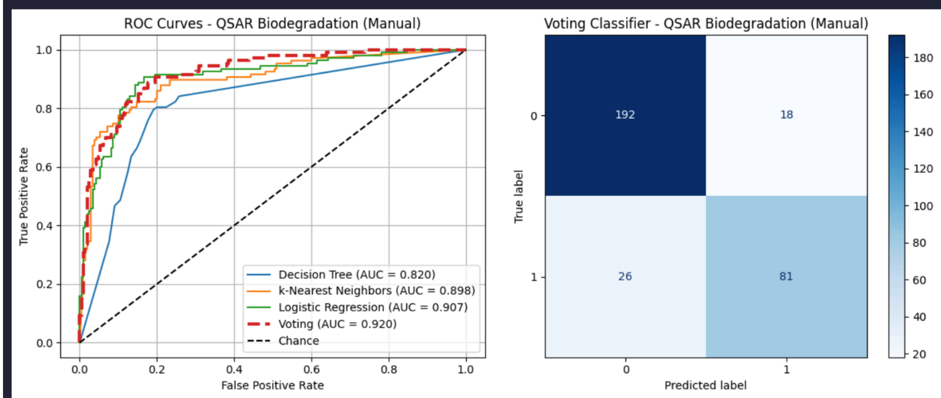
```
============================================================
RUNNING BUILT-IN GRID SEARCH FOR WINE QUALITY
============================================================

--- GridSearchCV for Decision Tree ---
Fitting 5 folds for each of 72 candidates, totalling 360 fits
Best params for Decision Tree: {'classifier__criterion': 'gini', 'classifier__max_depth': 10, 'classifier__min_samples_split': 10, 'feature_selection__k': 5}
Best CV score: 0.7721

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

--- GridSearchCV for Logistic Regression ---
Fitting 5 folds for each of 30 candidates, totalling 150 fits
Best params for Logistic Regression: {'classifier__C': 1, 'classifier__solver': 'saga', 'feature_selection__k': 'all'}
Best CV score: 0.8052

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

--- Individual Model Performance ---
...
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 1.0000, Precision: 1.0000
  Recall: 1.0000, F1: 1.0000, AUC: 1.0000
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



```
============================================================
RUNNING BUILT-IN GRID SEARCH FOR BANKNOTE AUTHENTICATION
============================================================

--- GridSearchCV for Decision Tree ---
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Best params for Decision Tree: {'classifier__criterion': 'entropy', 'classifier__max_depth': None, 'classifier__min_samples_split': 10, 'feature_selection__k': 'all'}
Best CV score: 0.9913

--- GridSearchCV for k-Nearest Neighbors ---
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best params for k-Nearest Neighbors: {'classifier__metric': 'manhattan', 'classifier__n_neighbors': 7, 'classifier__weights': 'uniform', 'feature_selection__k': 'all'}
Best CV score: 0.9990

--- GridSearchCV for Logistic Regression ---
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best params for Logistic Regression: {'classifier__C': 100, 'classifier__solver': 'liblinear', 'feature_selection__k': 'all'}
Best CV score: 0.9996

============================================================
EVALUATING BUILT-IN MODELS FOR BANKNOTE AUTHENTICATION
============================================================

--- Individual Model Performance ---
...
New best score: 0.9154 with params: {'classifier__C': 0.01, 'classifier__solver': 'liblinear', 'feature_selection__k': 41}
New best score: 0.9166 with params: {'classifier__C': 0.01, 'classifier__solver': 'saga', 'feature_selection__k': 41}
New best score: 0.9311 with params: {'classifier__C': 0.1, 'classifier__solver': 'liblinear', 'feature_selection__k': 41}
New best score: 0.9316 with params: {'classifier__C': 0.1, 'classifier__solver': 'saga', 'feature_selection__k': 41}
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
--------------------------------------------------------------------------------
Best parameters for Logistic Regression: {'classifier__C': 0.1, 'classifier__solver': 'saga', 'feature_selection__k': 41}
Best cross-validation AUC: 0.9316
```

```
============================================================
EVALUATING MANUAL MODELS FOR QSAR BIODEGRADATION
============================================================

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.7918
  Precision: 0.6916
  Recall: 0.6916
  F1-Score: 0.6916
  ROC AUC: 0.8202

k-Nearest Neighbors:
  Accuracy: 0.8549
  Precision: 0.7905
  Recall: 0.7757
  F1-Score: 0.7830
  ROC AUC: 0.8985

Logistic Regression:
...
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8612, Precision: 0.8182
  Recall: 0.7570, F1: 0.7864, AUC: 0.9201
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```



```
============================================================
RUNNING BUILT-IN GRID SEARCH FOR QSAR BIODEGRADATION
============================================================

--- GridSearchCV for Decision Tree ---
Fitting 5 folds for each of 96 candidates, totalling 480 fits
Best params for Decision Tree: {'classifier__criterion': 'entropy', 'classifier__max_depth': 10, 'classifier__min_samples_split': 10, 'feature_selection__k': 15}
Best CV score: 0.8389

--- GridSearchCV for k-Nearest Neighbors ---
Fitting 5 folds for each of 64 candidates, totalling 320 fits
Best params for k-Nearest Neighbors: {'classifier__metric': 'manhattan', 'classifier__n_neighbors': 9, 'classifier__weights': 'distance', 'feature_selection__k': 'all'}
Best CV score: 0.9045

--- GridSearchCV for Logistic Regression ---
Fitting 5 folds for each of 40 candidates, totalling 200 fits
Best params for Logistic Regression: {'classifier__C': 0.1, 'classifier__solver': 'saga', 'feature_selection__k': 'all'}
Best CV score: 0.9316

============================================================
EVALUATING BUILT-IN MODELS FOR QSAR BIODEGRADATION
============================================================

--- Individual Model Performance ---
...

======================================================================
ALL DATASETS PROCESSED!
======================================================================
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

6. Conclusion

This lab successfully demonstrated the process of model selection through hyperparameter tuning and comparative analysis. A key finding was that while manual implementation of Grid Search is valuable for understanding the underlying mechanics, scikit-learn's `GridSearchCV` is far more efficient and less error-prone for practical use. The results confirmed that there is no single best model for all problems; performance is highly dataset-dependent. k-NN performed best on the Wine Quality dataset, while the Voting Classifier excelled on the more complex QSAR Biodegradation dataset. The primary takeaway is the importance of a systematic pipeline for tuning and evaluation, and the power of using library functions like `GridSearchCV` to streamline this critical machine learning task.