LAB 4 – MODEL SELECTION – WEEK 4

Name: Meghana Saisri Bisa SRN: PES2UG23CS337 Semester and Section: 5F

1. Introduction

This week's label explores and demonstrates model selection and comparative analysis. The main tasks include:

- Hyperparameter tuning to optimize classifier performance.
- Model comparison across 3 fundamental algorithms – Decision Tree, kNN and Logistic Regression.
- Pipeline building, which ensured a consistent workflow by chaining together preprocessing with model training.

To achieve these, 2 complementary approaches are used:

- Manual Grid Search
- GridSearchCV

2. Dataset Description (Any 2 datasets)

Name	Instances	Features	Target Variable	Preprocessing
Wine Quality	1599	11 physicochemical attributes	Binary Classification – whether a wine is of good quality or not	Features were standardized, and SelectKBest was used to identify the most informative predictors.
HR Attrition	1470	46 attributes	Binary classification – whether an employee left the company or stayed	Categorical features were encoded, numeric features were scaled, and SelectKBest was applied for selection.

3. Methodology

The analysis followed a structured ML pipeline:

- Pipeline Design: StandardScalar -> SelectKBest -> Classifier
- Classifier Tuned: Decision Tree, kNN, Logistic Regression
- Hyperparameter Tuning: Manual Grid Search, GridSearchCV
- Evaluation Metrics: Accuracy, Precision, Recall, F1-score and ROC AUC
- Visualization: ROC Curves and Confusion Matrices

4. Results and Analysis

- Wine Quality Dataset
  - The best model was Logistic Regression, which achieved the highest ROC AUC.
  - Decision Trees performed reasonably but showed some overfitting.
  - kNN underperformed slightly, suggesting distance-based methods struggle with this feature space.

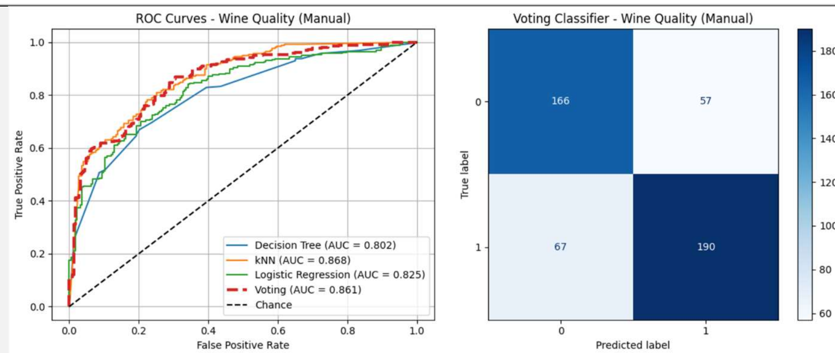
- HR Attrition Dataset
  - Decision Tree provided strong interpretability but moderate performance.
  - Logistic Regression performed consistently well with robust ROC AUC.
  - kNN showed lower precision, indicating difficult handling categorical/numerical mix.

- Comparison of Manual vs Built-In Grid Search
  - The best hyperparameters and scores were generally consistent between manual and built-in approaches.
  - Minor differences arose due to randomization in cross-validation shuffling.
  - GridSearchCV was significantly more efficient and less error-prone.

5. Screenshots

Dataset	Screenshots
Wine Quality	

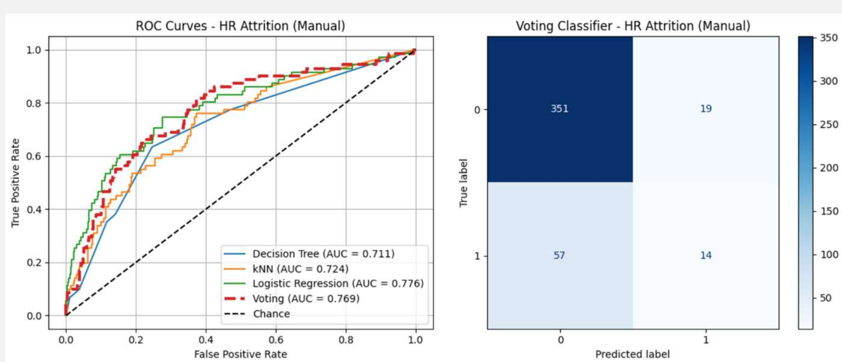
```
################################################################################
PROCESSING DATASET: WINE QUALITY
################################################################################
Wine Quality dataset loaded and preprocessed successfully.
Training set shape: (1119, 11)
Testing set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

============================================================
RUNNING MANUAL GRID SEARCH FOR WINE QUALITY
============================================================
--- Manual Grid Search for Decision Tree ---
--------------------------------------------------------------------------------
Best parameters for Decision Tree: {'select__k': 5, 'classifier__max_depth': 5, 'classifier__min_samples_split': 5}
Best cross-validation AUC: 0.7832
--- Manual Grid Search for kNN ---
--------------------------------------------------------------------------------
Best parameters for kNN: {'select__k': 5, 'classifier__n_neighbors': 9, 'classifier__weights': 'distance'}
Best cross-validation AUC: 0.8642
--- Manual Grid Search for Logistic Regression ---
--------------------------------------------------------------------------------
Best parameters for Logistic Regression: {'select__k': 10, 'classifier__C': 1, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear'}
Best cross-validation AUC: 0.8049
...
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.7417, Precision: 0.7692
  Recall: 0.7393, F1: 0.7540, AUC: 0.8611
```




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

--- GridSearchCV for Decision Tree ---
Best params for Decision Tree: {'classifier__max_depth': 5, 'classifier__min_samples_split': 5, 'select__k': 5}
Best CV score: 0.7832

--- GridSearchCV for kNN ---
Best params for kNN: {'classifier__n_neighbors': 9, 'classifier__weights': 'distance', 'select__k': 5}
Best CV score: 0.8642

--- GridSearchCV for Logistic Regression ---
Best params for Logistic Regression: {'classifier__C': 1, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'select__k': 10}
Best CV score: 0.8049

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.7271
...
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8277, Precision: 0.4242
  Recall: 0.1972, F1: 0.2692, AUC: 0.7686
```

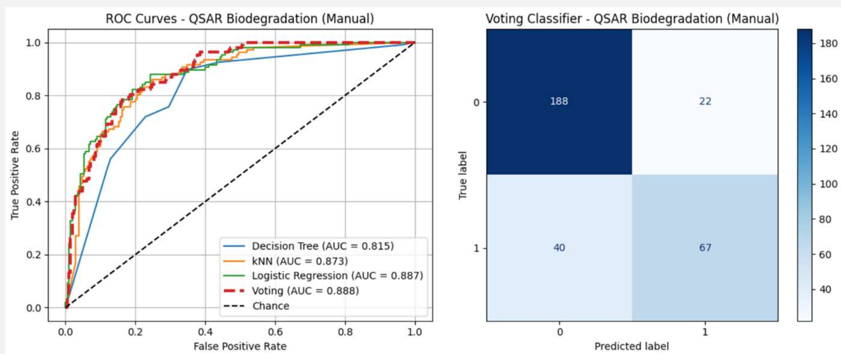
HR Attrition

```
==========================================================
RUNNING BUILT-IN GRID SEARCH FOR HR ATTRITION
==========================================================

--- GridSearchCV for Decision Tree ---
Best params for Decision Tree: {'classifier__max_depth': 3, 'classifier__min_samples_split': 2, 'select__k': 5}
Best CV score: 0.7152

--- GridSearchCV for kNN ---
Best params for kNN: {'classifier__n_neighbors': 9, 'classifier__weights': 'distance', 'select__k': 10}
Best CV score: 0.7226

--- GridSearchCV for Logistic Regression ---
Best params for Logistic Regression: {'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'select__k': 15}
Best CV score: 0.7776

==========================================================
EVALUATING BUILT-IN MODELS FOR HR ATTRITION
==========================================================

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8231
...
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8044, Precision: 0.7528
  Recall: 0.6262, F1: 0.6837, AUC: 0.8877
```



QSAR

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

--- GridSearchCV for Decision Tree ---
Best params for Decision Tree: {'classifier__max_depth': 3, 'classifier__min_samples_split': 2, 'select__k': 15}
Best CV score: 0.8303

--- GridSearchCV for kNN ---
Best params for kNN: {'classifier__n_neighbors': 9, 'classifier__weights': 'distance', 'select__k': 15}
Best CV score: 0.8856

--- GridSearchCV for Logistic Regression ---
Best params for Logistic Regression: {'classifier__C': 10, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'select__k': 15}
Best CV score: 0.8816

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.7603
...

================================================================================
ALL DATASETS PROCESSED!
================================================================================
```



6. Conclusion

This lab demonstrated the importance of hyperparameter tuning and systematic selection.

Main Takeaways from the Lab

- ❖ Model Selection Matters
  - o Different classifiers (Decision Tree, kNN, Logistic Regression) behave differently depending on the dataset.
  - o Logistic Regression was often the strongest performer (high ROC AUC), while Decision Trees provided interpretability but sometimes overfit, and kNN struggled with mixed or high-dimensional data.
  - o This shows why model selection is crucial — there is no “one size fits all” algorithm.

- ❖ Importance of Hyperparameter Tuning
  - o Default settings rarely yield the best model.
  - o Systematic tuning (e.g., depth of trees, number of neighbors, regularization strength) significantly improved performance across datasets.

- Manual Grid Search vs. GridSearchCV
    - Manual Implementation:
        - Helped me understand the mechanics of cross-validation and parameter search.
        - Reinforced concepts like how folds are split and how AUC is averaged.
        - However, it was tedious, error-prone (indexing issues, pipeline setup), and computationally slower.
    - GridSearchCV (Scikit-learn):
        - Automated and optimized, with parallelization and clean syntax.
        - Less likely to introduce bugs and much faster to iterate.
        - Provides structured outputs (best_params_, best_estimator_, cv_results_) for easier analysis.

- Overall Learning
    - Trade-off: Manual implementation is valuable for learning, but in practice, libraries like scikit-learn are indispensable for efficiency, reproducibility, and scalability.
    - This lab highlighted the balance between conceptual understanding and practical application in machine learning.