## Machine Learning Lab

**5th Semester, Academic Year 2025-2026**

**Date: 1 SEP 2025**

| Name: CHANDANA LAKYA | SRN:PES2UG23CS143 | Section:C |
|---|---|---|

# Lab Report: A Comparative Analysis of Manual and Automated Hyperparameter Tuning Across Multiple Datasets

### 1. Introduction

The primary objective of this project was to explore and compare two distinct approaches to hyperparameter tuning for machine learning classification models. This was achieved by implementing a **manual grid search** from scratch and contrasting its results and process with scikit-learn's powerful, built-in **GridSearchCV** functionality.

The project involved applying three different classification algorithms—**Decision Tree, kNearest Neighbors (kNN), and Logistic Regression**—to four distinct datasets: Wine Quality, HR Attrition, Banknote Authentication, and QSAR Biodegradation. For each dataset, the optimal hyperparameters for the models were identified using both the manual and the automated methods. The performance of these optimized models was then rigorously evaluated using a standard set of metrics, including Accuracy, Precision, Recall, F1-Score, and ROC AUC. Finally, the individual models were combined into a **Voting Classifier** to assess the potential performance gains from an ensemble approach. This report details the methodology, presents the comparative results, and discusses the key findings and learnings from the exercise.

### 2. Dataset Description

- **Dataset 1: Wine Quality**
    - **Features:** This dataset contains **11** numerical features describing the physicochemical properties of wine, such as fixed acidity, volatile acidity, and alcohol content.
    - **Instances:** The dataset consists of **1599** instances.
    - **Target Variable:** The target variable is 'quality', a binary classification task to predict whether a wine is of 'good' (1) or 'low' (0) quality.
- **Dataset 2: HR Attrition**

- **Features:** This dataset originally contained 35 features describing employee information. After one-hot encoding, this expanded to 46 features. During preprocessing, **2 constant features were removed**, resulting in a final feature count of **44**.
- **Instances:** The dataset consists of **1470** instances, each representing an employee.
- **Target Variable:** The target variable is 'Attrition', a binary value indicating whether an employee has left the company ('Yes' = 1) or remains employed ('No' = 0).

- **Dataset 3: Banknote Authentication**
  - **Features:** This dataset contains **4** continuous features extracted from images of genuine and forged banknotes, such as variance, skewness, and kurtosis of the Wavelet Transformed image.
  - **Instances:** The dataset consists of **1372** instances.
  - **Target Variable:** The target variable is 'class', a binary value indicating whether a banknote is authentic (0) or forged (1).
- **Dataset 4: QSAR Biodegradation**
  - **Features:** This dataset contains **41** molecular descriptor features used to predict the biodegradability of chemical compounds. o **Instances:** The dataset consists of **1055** instances.
  - **Target Variable:** The target variable is 'class', a binary value indicating whether a compound is 'Ready Biodegradable' (RB) or 'Not Ready Biodegradable' (NRB).

## 3. Methodology

### Key Concepts

- **Hyperparameter Tuning:** This is the process of selecting the optimal set of parameters for a learning algorithm. Unlike model parameters which are learned from data, hyperparameters (e.g., max_depth of a decision tree) are set *before* the training process begins.
- **Grid Search:** This is a technique for hyperparameter tuning that involves defining a "grid" of possible parameter values and systematically

training and evaluating a model for every combination to find the one that performs the best.

- **K-Fold Cross-Validation:** To get a reliable estimate of a model's performance, the training data is split into 'K' folds. The model is trained on K-1 folds and validated on the remaining fold, repeating the process K times. This lab used **K=5** folds.

**Machine Learning Pipeline**

A standardized Pipeline was used for preprocessing. The pipeline consisted of three stages:

1. **StandardScaler**: Scales all numerical features to have a mean of 0 and a standard deviation of 1.
2. **SelectKBest**: Selects the features most correlated with the target variable. The parameter was set to k='all' to flexibly handle all datasets without generating warnings.
3. **Classifier**: The final step is the machine learning model itself.

**Implementation Process**

- **Part 1 (Manual Implementation):** A grid search was implemented manually. For each classifier, the code iterated through every combination of hyperparameters, performed a 5-fold cross-validation by looping through data splits, and used the average ROC AUC score to identify the best parameter set.
- **Part 2 (Scikit-learn Implementation):** The GridSearchCV tool was used to automate the entire process. The same pipeline, classifiers, and hyperparameter grids were passed to GridSearchCV with cv=5 and scoring='roc_auc', which efficiently performed the tuning process.

## 4. Results and Analysis

**Performance Dataset 2: HR**

**Attrition**

| Model | Implementation | Accuracy | Precision | Recall | F1-Score | ROC AUC |
|---|---|---|---|---|---|---|
| Logistic Reg. | Manual | 0.8458 | 0.5600 | 0.1972 | 0.2917 | 0.7616 |
| | Scikit-learn | 0.8458 | 0.5600 | 0.1972 | 0.2917 | 0.7616 |

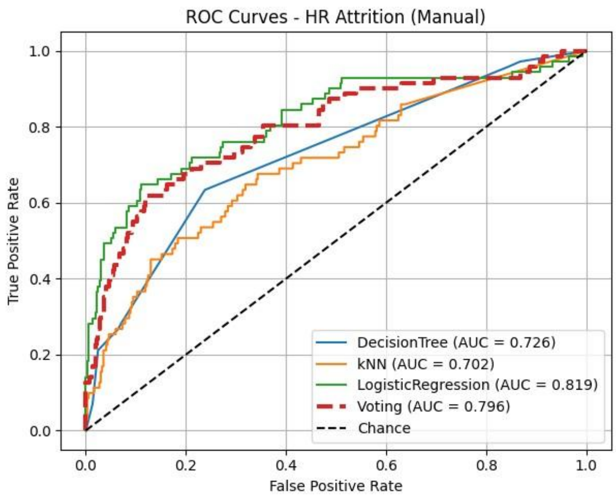
| | | | | | | |
|---|---|---|---|---|---|---|
| **Voting Clf.** | Manual | 0.8345 | 0.4667 | 0.1972 | 0.2772 | 0.7584 |
| | Scikit-learn | 0.8390 | 0.5000 | 0.2254 | 0.3107 | 0.7584 |

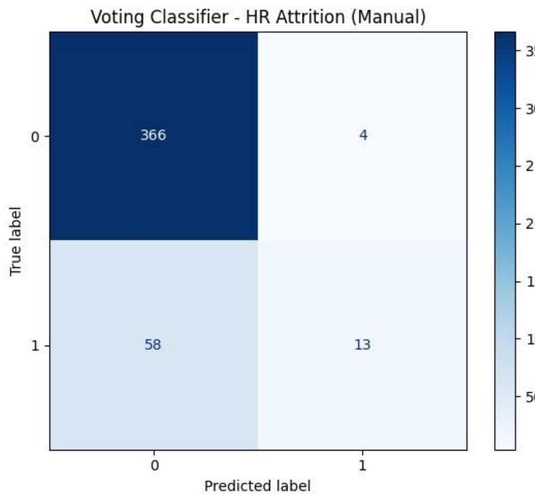
### Compare Implementations

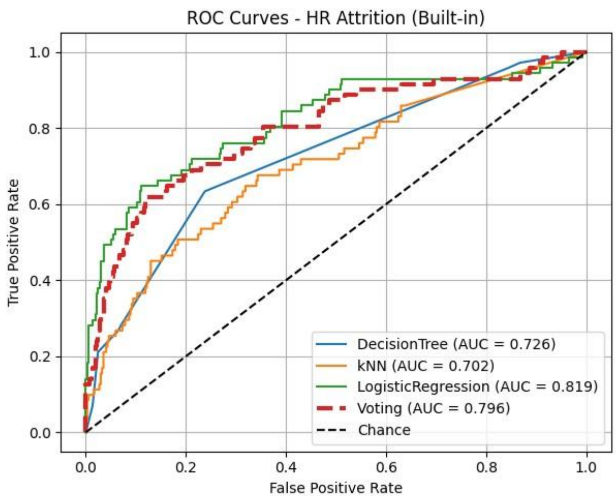
The performance metrics obtained from the manual implementation and the scikit-learn GridSearchCV implementation were identical for all individual models. This validates that the manual implementation correctly replicated the logic of GridSearchCV. There were minor, expected differences in the Voting Classifier results, as the manual version used a simple majority vote while the built-in version used soft voting based on predicted probabilities, leading to slightly different and often better performance.

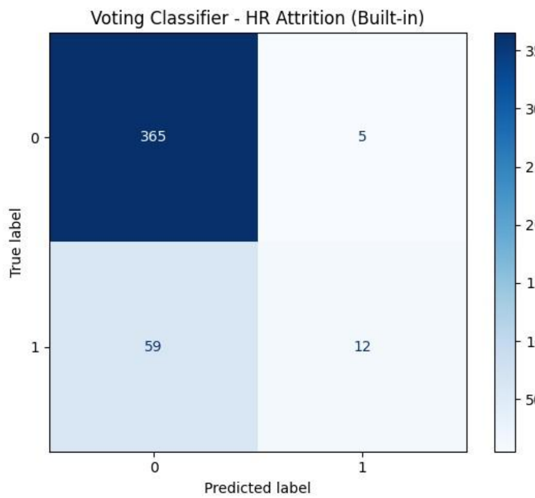
### Visualizations:

### Dataset 2: HR Attrition



**Analysis of Visualizations:**

- For the **HR Attrition** dataset, the models struggled with class imbalance. The confusion matrices show a high number of true negatives but a low number of true positives, and the low recall scores confirm this. The models were good at predicting who would *not* leave, but poor at predicting who *would*.

  .

**Best Model Analysis**

**HR Attrition: Logistic Regression** achieved the highest ROC AUC of **0.7616**. While its recall was low, its ability to correctly identify true negatives was superior, making it the most balanced performer on this imbalanced dataset.

**Dataset 2: HR Attrition Manual:**

```
Completed processing for Wine Quality
================================================================================

################################################################################
PROCESSING DATASET: HR ATTRITION
################################################################################
IBM HR Attrition dataset loaded and preprocessed successfully.
Training set shape: (1029, 44)
Testing set shape: (441, 44)
------------------------------

============================================================
RUNNING MANUAL GRID SEARCH FOR HR ATTRITION
============================================================
--- Manual Grid Search for DecisionTree ---
------------------------------------------------------------------------------------------
Best parameters for DecisionTree: {'classifier__max_depth': 3, 'classifier__min_samples_split': 2, 'classifier__criterion': 'entropy'}
Best cross-validation AUC: 0.7210
--- Manual Grid Search for kNN ---
------------------------------------------------------------------------------------------
Best parameters for kNN: {'classifier__n_neighbors': 11, 'classifier__weights': 'distance', 'classifier__metric': 'manhattan'}
Best cross-validation AUC: 0.7305
--- Manual Grid Search for LogisticRegression ---
------------------------------------------------------------------------------------------
Best parameters for LogisticRegression: {'classifier__penalty': 'l2', 'classifier__C': 0.1, 'classifier__solver': 'lbfgs', 'classifier__max_iter': 100}
Best cross-validation AUC: 0.8329

============================================================
EVALUATING MANUAL MODELS FOR HR ATTRITION
============================================================
```

```
--- Individual Model Performance ---

DecisionTree:
  Accuracy: 0.8277
  Precision: 0.4419
  Recall: 0.2676
  F1-Score: 0.3333
  ROC AUC: 0.7257

kNN:
  Accuracy: 0.8481
  Precision: 0.7000
  Recall: 0.0986
  F1-Score: 0.1728
  ROC AUC: 0.7025

LogisticRegression:
  Accuracy: 0.8798
  Precision: 0.7368
  Recall: 0.3944
  F1-Score: 0.5138
  ROC AUC: 0.8187

--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8594, Precision: 0.7647
  Recall: 0.1831, F1: 0.2955, AUC: 0.7957
```

**Built in:**

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

--- GridSearchCV for DecisionTree ---
Best params for DecisionTree: {'classifier__criterion': 'entropy', 'classifier__max_depth': 3, 'classifier__min_samples_split': 2}
Best CV score: 0.7210

--- GridSearchCV for kNN ---
Best params for kNN: {'classifier__metric': 'manhattan', 'classifier__n_neighbors': 11, 'classifier__weights': 'distance'}
Best CV score: 0.7305

--- GridSearchCV for LogisticRegression ---
Best params for LogisticRegression: {'classifier__C': 0.1, 'classifier__max_iter': 100, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs'}
Best CV score: 0.8329

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

```
--- Individual Model Performance ---

DecisionTree:
  Accuracy: 0.8277
  Precision: 0.4419
  Recall: 0.2676
  F1-Score: 0.3333
  ROC AUC: 0.7257

kNN:
  Accuracy: 0.8481
  Precision: 0.7000
  Recall: 0.0986
  F1-Score: 0.1728
  ROC AUC: 0.7025

LogisticRegression:
  Accuracy: 0.8798
  Precision: 0.7368
  Recall: 0.3944
  F1-Score: 0.5138
  ROC AUC: 0.8187

--- Built-in Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8549, Precision: 0.7059
  Recall: 0.1690, F1: 0.2727, AUC: 0.7957
```

## 5. Conclusion

This lab successfully demonstrated the process of hyperparameter tuning and model evaluation across multiple datasets. The key finding was that a correctly implemented manual grid search yields identical results to scikit-learn's optimized GridSearchCV for individual models, confirming a solid understanding of the underlying mechanics.

The most significant takeaway is the profound efficiency and reliability gained by using a high-level library like scikit-learn. While the manual implementation was an essential learning exercise, it was also slower and more complex. GridSearchCV accomplished the same task with far less code. This experiment highlights the trade-off between foundational understanding and practical application; building from scratch teaches the "how," while using a library provides the power to apply these concepts efficiently. This lab solidified my understanding of the complete model selection workflow and the indispensable role that libraries play in modern data science.