UE23CS352A: MACHINE LEARNING

Week 4: Model Selection and Comparative Analysis

1. Lab Overview & Objectives

Welcome to the lab on building a complete machine learning pipeline. The goal of this assignment is to give you hands-on experience with model selection and evaluation by implementing hyperparameter tuning and ensemble methods—two critical techniques in applied machine learning.

You will be working with the provided Jupyter Notebook: `Week4_Lab_Boilerplate.ipynb`. This project is divided into two main parts:

- **Part 1: Manual Implementation.** You will build a Grid Search from the ground up using basic loops to understand its inner workings.
- **Part 2: Scikit-learn Implementation.** You will then use scikit-learn's highly optimized, built-in `GridSearchCV` to perform the same task, demonstrating the efficiency of modern ML libraries.

Learning Goals

- Understand and implement a pipeline for scaling, feature selection, and model training.
- Grasp the fundamentals of hyperparameter tuning using **Grid Search**.
- Implement **k-fold cross-validation** to get robust performance estimates.
- Effectively use scikit-learn's `Pipeline` and `GridSearchCV` tools.
- Evaluate and compare the performance of different classification models using various metrics.
- Analyze and interpret model results to draw meaningful conclusions.

2. Datasets

The boilerplate notebook includes functions to load and preprocess four different datasets.

You are required to choose at least two of these datasets to run your complete pipeline on.

- **Wine Quality:** Predict whether a red wine is of 'good quality' based on its chemical properties.
- **HR Attrition:** Predict employee attrition based on a variety of work-related and personal factors.
- **Banknote Authentication:** Distinguish between genuine and forged banknotes based on image features.
- **QSAR Biodegradation:** Predict whether a chemical is readily biodegradable based on its quantitative structure-activity relationship (QSAR) properties.

3. Key Concepts and The ML Pipeline

Classifiers

You will be tuning and comparing three fundamental classification algorithms:

1. **Decision Tree:** A model that uses a tree-like structure of decisions to classify data.
2. **k-Nearest Neighbors (kNN):** A non-parametric algorithm that classifies a data point based on the majority class of its 'k' closest neighbors.
3. **Logistic Regression:** A linear model that predicts the probability of a binary outcome, used for classification tasks.

The Scikit-Learn Pipeline

Your notebook uses a scikit-learn **`Pipeline`** to chain together the data processing and modeling steps. This is crucial for preventing data leakage and streamlining your workflow. The pipeline for each classifier consists of three stages:

`StandardScaler` -> `SelectKBest` -> `Classifier`

1. **`StandardScaler`**: Standardizes features to have a mean of 0 and a standard deviation of 1.
2. **`SelectKBest`**: Selects the top 'k' features using a statistical test (`f_classif`). The number of features, `k`, is a key hyperparameter you will tune.
3. **`Classifier`**: The final modeling step (Decision Tree, kNN, or Logistic Regression).

4. Instructions and Tasks

Your main task is to complete the `TODO` sections within the `Wee4_Lab_Boilerplate.ipynb` notebook for at least two datasets.

Part 1: Manual Grid Search Implementation (`run_manual_grid_search` function)

In this section, you will complete the code to perform a grid search from scratch.

1. **Define Parameter Grids:** In the "Models and Parameter Grids" cell, define the hyperparameter grids (`param_grid_dt`, `param_grid_knn`, `param_grid_lr`) you want to test. Remember that parameter names must match the pipeline step names (e.g., `'classifier__max_depth'`).
2. **Implement the Search Loop:** Inside the `run_manual_grid_search` function, you need to:
    - Generate all possible combinations of hyperparameters from your defined grid.
    - For each combination, perform **5-fold stratified cross-validation**.
    - Inside each fold, build the pipeline, fit it on the training part of the fold, and evaluate it on the validation part.
    - Calculate the **average ROC AUC score** across all 5 folds for that hyperparameter combination.
    - Keep track of the combination that yields the highest average score.
3. **Fit the Best Model:** After the search is complete, the function will automatically refit a new pipeline using the best-found parameters on the *entire* training dataset.

Part 2: Built-in `GridSearchCV` Implementation (`run_builtin_grid_search` function)

Here, you will leverage scikit-learn's powerful `GridSearchCV` to automate the process.

1. **Set up `GridSearchCV`:** Inside the `run_builtin_grid_search` function, for each classifier, you need to:
    - Create the same three-step `Pipeline` as in the manual part.
    - Instantiate `GridSearchCV` with the pipeline, the parameter grid, `scoring='roc_auc'`, and a 5-fold `StratifiedKFold` cross-validator.
    - Fit the `GridSearchCV` object on the training data.
2. **Extract Results:** The function will automatically extract the best estimator (the

pipeline with the best-found parameters), the best parameters, and the best cross-validation score.

5. Expected Deliverables

You are required to submit two items:

1. Completed Jupyter Notebooks (`.ipynb` files)

- Submit a **separate, completed notebook for each dataset** you chose (minimum of two).
- The notebooks must be fully executed, with all `TODO` sections completed and all output cells (printouts, plots) visible.
- Ensure your code is clean, well-commented, and runs without errors from top to bottom.

2. Comprehensive Lab Report (`.pdf` file)

- A formal report that details your work and findings, following the structure outlined below.

6. Report Structure and Guidelines

Your report is a critical part of this lab and should demonstrate your understanding of the concepts.

Title Page

- Project Title, Your Name, Student ID, Course Name, Submission Date.

1. Introduction

- A brief overview of the project's purpose and the tasks performed (hyperparameter tuning, model comparison).

2. Dataset Description

- For each dataset you chose, provide a brief description: number of features, number

of instances, and what the target variable represents.

3. Methodology

- Explain the key concepts: Hyperparameter Tuning, Grid Search, K-Fold Cross-Validation.
- Describe the ML pipeline used (`StandardScaler`, `SelectKBest`, Classifier).
- Describe the process you followed for both the manual implementation (Part 1) and the scikit-learn implementation (Part 2).

4. Results and Analysis

- For each dataset, present your findings clearly. **Using tables is highly recommended.**
- **Performance Tables:** Use tables to display the key performance metrics (Accuracy, Precision, Recall, F1-Score, and ROC AUC) for the best version of each classifier from both Part 1 and Part 2.
- **Compare Implementations:** Compare the results from your manual implementation with the scikit-learn implementation. Are they identical? Discuss any potential reasons for minor differences.
- **Visualizations:** Include the final **ROC Curve plots** and **Confusion Matrix plots** generated in your notebook. Analyze these plots to compare model performance.
- **Best Model:** Analyze which model performed best overall for each dataset and offer a hypothesis as to why.

5. Screenshots

Attach screenshots of the output of your code.

6. Conclusion

- Summarize your key findings.
- Discuss the main takeaways from this lab. What did you learn about model selection and the trade-offs between manual implementation and using a library like scikit-learn?

Good luck!