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).

. Instructions and Tasks Your main task is to complete the TODO sections within the
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 (thepipeline with the best-found parameters), the best parameters, and the best cross-validation score.

Key Concepts

- **Hyperparameter Tuning: The process of searching for the best combination of parameters (hyperparameters) that optimize model performance.**
- **Grid Search: A systematic way to explore hyperparameter combinations by evaluating all specified settings.**
- **K-Fold Cross-Validation: Splitting data into k subsets (folds) to repeatedly train and validate the model, yielding robust performance estimates.**

1. **StandardScaler: Standardizes features to have zero mean and unit variance.**
2. **SelectKBest: Selects the top `k` features based on statistical tests (ANOVA F-value `f_classif`), where `k` is a hyperparameter to be tuned.**
3. **Classifier: The final step, which can be a Decision Tree, k-Nearest Neighbors (kNN), or Logistic Regression model.**
4. **Logistic Regression model.**

Manual Grid Search Implementation

- **Define hyperparameter grids for each classifier (e.g., max_depth for Decision Tree, number of neighbors for kNN, regularization strength for Logistic Regression).**
- **Implement nested loops to generate all hyperparameter combinations.**
- **For each combination, perform 5-fold stratified cross-validation:**
    - **For each fold, train the pipeline on the training split and evaluate on the validation split.**
    - **Collect the ROC AUC scores and average across folds.**
- **Select the hyperparameter combination with the highest mean ROC AUC.**
- **Fit the final pipeline with the best parameters on the entire training dataset.**

Built-in GridSearchCV Implementation

- **Use scikit-learn's `GridSearchCV` to automate hyperparameter tuning:**
  - **Create the same 3-step pipeline.**
  - **Set up the grid search with the parameter grid, ROC AUC as scoring, and 5-fold stratified cross-validation.**
  - **Fit `GridSearchCV` on the training data to find the best parameters and model.**
- **Extract best estimator, best parameters, and best cross-validation score.**

Model Evaluation and Ensemble Methods

- **Evaluate individual tuned models on the test set using metrics: accuracy, precision, recall, F1-score, and ROC AUC.**
- **Implement a voting classifier ensemble:**
  - **For the manual implementation, aggregate predictions and average predicted probabilities across models to make majority vote decisions.**
  - **For built-in, use scikit-learn's VotingClassifier with soft voting.**
- **Visualize results with ROC curves and confusion matrices to compare models and ensemble performance.**

This methodology provides hands-on experience in model selection, hyperparameter tuning, and ensemble learning while reinforcing important ML concepts like pipelines, cross-validation, and model evaluation metrics. It is applied to at least two datasets as given in the lab.