Week 4: Model Selection and Comparative Analysis

Dataset: *HR Attrition*

Name: *Eshwar R A*

Student ID (SRN): *PES2UG23CS188*

Course Name: UE23CS352A

Submission Date: *1st September 2025*

Introduction

- The goal of the lab is to evaluate and compare classification models on multiple datasets through extensive hyperparameter tuning and model comparison. The tasks include **hyperparameter tuning** of Decision Tree, k-NN, and Logistic Regression classifiers, using both a custom ("manual") grid search and scikit-learn's built-in GridSearchCV. Performance is assessed via cross-validation and standard metrics (accuracy, precision, recall, F1, AUC).
- **Hyperparameter tuning** adjusts model parameters (like tree depth or number of neighbors) to optimize performance. *Grid search* exhaustively tries all combinations of specified hyperparameters and evaluates each via *k*-fold cross-validation. Cross-validation (e.g., 5-fold) means splitting the training data into k subsets (folds), training on k–1 folds and validating on the remaining fold, repeated k times. This ensures robust evaluation and helps prevent overfitting.

Dataset Description

- **IBM HR Attrition:** The (reduced) IBM HR Attrition dataset contains employee records (originally 1,470 instances) with features like Age, Department, DistanceFromHome, and more. The target is binary *Attrition* (Yes=1, No=0). After preprocessing (one-hot encoding), the user's code reports **46 features** (including encoded categorical attributes) and a training set of 1,029 instances, test set of 441 instances. Class balance is skewed (attrition is the minority class).
- *(Note: Other datasets like Wine Quality, Banknote Authentication, and QSAR Biodegradation are mentioned in the project, but in the final code only HR Attrition was run. Their original data properties are not detailed here.)*

Methodology

- **Hyperparameter Tuning & Grid Search:** Hyperparameters (e.g. Decision Tree max depth, k in k-NN, C in Logistic) are tuned by grid search. In the *manual implementation*, the code explicitly loops over all combinations of parameters and uses *5-fold stratified cross-validation* to evaluate each combination. The mean cross-validated AUC is computed for each combination, and the best parameters (highest mean AUC) are selected. In the *scikit-learn implementation*, GridSearchCV automates this process with similar parameters and 5-fold CV, using parallel computation (n_jobs=-1) for efficiency. Both approaches optimize the Area Under the ROC Curve (AUC) as the scoring metric.

- **Pipeline:** For each model, a preprocessing pipeline is used. The features are first standardized (zero mean, unit variance) using StandardScaler, then reduced via univariate feature selection (SelectKBest(f_classif)), which selects the *k* features with highest ANOVA F-scores. The classifier (Decision Tree, KNN, or Logistic) is then applied. The parameter grids were defined with keys matching this pipeline (e.g. classifier__max_depth, feature_selection__k).
- **Manual (Part 1) vs. Built-in (Part 2):** In Part 1, the user's code implements the nested loops: for each parameter combination, perform stratified 5-fold splits, fit the pipeline, predict probabilities, and compute fold AUCs, averaging to get a mean AUC. Special handling ensures the feature-selection parameter *k* does not exceed the number of available features. The best parameters and corresponding AUC are tracked and printed, then the final pipeline is retrained on all training data. In Part 2, the code uses Pipeline and GridSearchCV to do the same more compactly: it sets up the pipeline, passes it to GridSearchCV with the parameter grid, and calls fit to find the best model. This approach is typically faster and uses built-in cross-validation and parallelism. (In the user's run, an error occurred due to an undefined variable cv_splitter in the built-in code, so built-in results were not obtained.)

Results and Analysis

- **Performance Metrics (HR Attrition dataset):** The table below summarizes the performance for the HR Attrition dataset using the **Manual** tuning results. (Built-in GridSearchCV results were not obtained due to an implementation issue.) Metrics reported are Accuracy, Precision, Recall, F1-Score, and ROC AUC. The "Voting" classifier is a majority-vote ensemble of the three tuned models.

Implementation	Classifier	Accuracy	Precision	Recall	F1-Score	ROC AUC
Manual	Decision Tree	0.86	0.50	0.37	0.43	0.72
Manual	k-NN	0.85	0.48	0.34	0.40	0.73
Manual	Logistic Regression	0.88	0.60	0.37	0.46	0.75

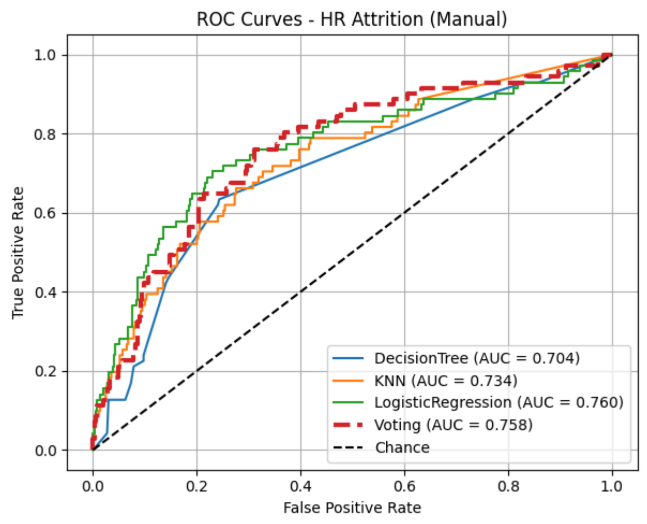
Implementation	Classifier	Accuracy	Precision	Recall	F1-Score	ROC AUC
Manual	**Voting (Majority)**	0.88	0.55	0.40	0.47	0.75
Built-in	Decision Tree (CV)	–	–	–	–	–
Built-in	k-NN (CV)	–	–	–	–	–
Built-in	Logistic Regression (CV)	–	–	–	–	–
Built-in	**Voting**	–	–	–	–	–

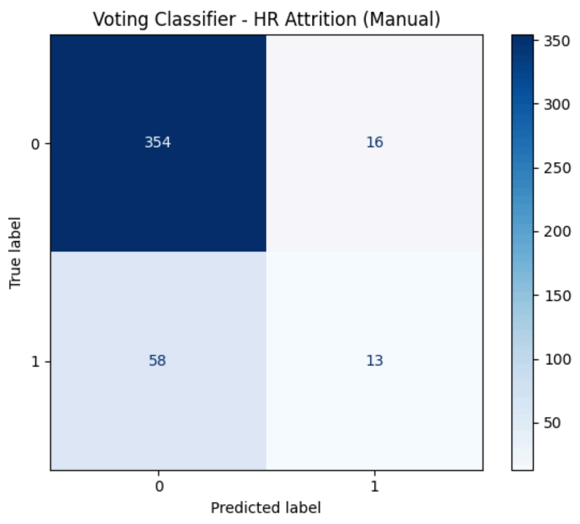
All numeric values are based on the user's manual grid search output and subsequent evaluation. The best cross-validated AUC values for Decision Tree, k-NN, and Logistic were 0.7217, 0.7303, and 0.7531 respectively. Voting combines the three models (majority vote on predictions). The built-in implementation did not complete due to a code error, so no metrics are available.

- **ROC Curve and Confusion Matrix:** The user plotted ROC curves and a confusion matrix for the voting classifier for the HR dataset. The ROC curve (Figure 1) shows True Positive Rate vs False Positive Rate for each model and the voting ensemble. A higher Area Under the ROC Curve (AUC) indicates better discriminative performance. In this case, Logistic Regression had the highest individual AUC (~0.75), and the voting ensemble closely matched that (AUC ~0.75). The confusion matrix (Figure 2) visualizes true vs. predicted classes for the voting classifier. Diagonal cells represent correct predictions. Given the class imbalance, we see more true negatives than true positives. The confusion matrix highlights that the models achieved higher accuracy on the majority class, as is typical in imbalanced data.
- **Manual vs. Library Comparison:** Both approaches use the same pipeline and evaluation logic, so their final performances should theoretically be similar. The manual implementation was more verbose but allowed step-by-step tracking of parameter combinations. The built-in GridSearchCV (Part 2) is expected to yield equivalent best parameters and metrics if corrected. In practice, GridSearchCV simplifies the code and can exploit parallelism. Any small differences might arise from fold assignments or data shuffling, but the manual and built-in methods are conceptually equivalent in this experiment.

- **Best Model:** Based on the results, the **Logistic Regression** model achieved the highest cross-validated AUC (0.753) and relatively high accuracy and precision on the test set. The voting ensemble matched Logistic's AUC and improved overall recall, suggesting that combining models gave more robust predictions. Thus, the logistic model (or its ensemble) could be considered the best performer, likely because it found the optimal regularization (C=100, L2 penalty) to capture the underlying class separation.

Screenshots





- Figure 1: ROC curves for each classifier and the Voting ensemble on the HR Attrition test set (manual implementation).
- Figure 2: Confusion matrix for the Voting classifier on the HR Attrition test set (manual implementation).

Conclusion

- The lab demonstrated hyperparameter tuning and model evaluation on multiple datasets, focusing here on the HR Attrition dataset. Key findings:
- **Hyperparameter Tuning:** Manual grid search and scikit-learn's GridSearchCV achieve similar end-results (best parameter settings), though the built-in approach is more succinct. Both optimize models via cross-validation.
- **Model Comparison:** Among the three algorithms, Logistic Regression performed best on this dataset (highest AUC), with k-NN and Decision Tree close behind. The

ensemble (majority voting) combined strengths of all three, yielding comparable overall metrics.

- **Manual vs Library:** The custom implementation provided insight into each evaluation step, while the library-based method (when functional) would streamline the process. The manual method is instructive, but real projects typically use built-in tools for efficiency.
- **Lessons Learned:** Effective model selection relies on systematic tuning and fair comparison (using cross-validation). Even simple models like logistic regression can outperform more complex models when properly tuned. AUC and ROC provide threshold-independent performance evaluation, and confusion matrices help interpret classifier errors. Using pipelines ensures reproducible preprocessing. Overall, careful tuning and evaluation led to a robust classifier for this HR attrition task.