ML Lab 4

Name: Dinakar Emmanuel

SRN: PES2UG23CS178

Course: Machine Learning

Submission date: 01/09/2025

Introduction

This lab involves implementing a manual grid search function and comparing its performance with scikit-learn's built-in GridSearchCV for hyperparameter tuning along with comparing the performance of 3 different classification algorithms (Decision Tree, k-NN, Logistic Regression) and combining them using voting classifiers.

Model performance was visualised using ROC curves and confusion matrices for each model, for both manual and built-in grid search. The dataset used was the “HR Attrition”, used to predict employee turnover.

Dataset Description

Dataset: WA_Fn-UseC_-HR-Employee-Attrition.csv
Features: 35
Instances: 1470
Target variable: the 2nd column ‘Attrition’ has values ‘Yes’ and ‘No’ that indicate the turnover of an employee based on 34 factors.

Methodology

1. Hyperparamter tuning: Hyperparameters are settings that control the *training process* of a model and are not values learned from the data. These could be parameters like learning rate, regularisation in logistic regression, depth of a decision tree, k in k-NN, etc. Tuning refers to finding the best combination of hyperparameter values to maximize the performance of the model. Wrong hyperparameters can cause underfitting (too simple) or overfitting (too complex). Good tuning is important for better accuracy, generalization, and efficiency of the model.

2. Grid search: This is a brute-force method for hyperparameter tuning. We define a grid that contains a list of possible values for each hyperparameter, and the algorithm tries all possible combinations to find the best one. Take k-NN for example,
    a. Number of neighbours = [7, 9, 11, 13]
    b. Weights = ['uniform', 'distance']

    Grid search will test all 4 x 2 = 8 combinations. This process is simple and guaranteed to test all combinations but it can be very slow, especially if the grid is too large.
3. K-fold Cross-Validation: It is a resampling technique to evaluate a model's performance more reliably.
    a. The dataset is split into k equally sized folds.
    b. The model trains on k − 1 folds and tests on the remaining fold.
    c. This is repeated k times, each fold serving as the test set once.
    d. The final score is the average of all test performances.

    This avoids having to rely on a single train-test split and gives a better estimate of how the model generalizes.

    Our goal is to find the best hyperparameters (tuning), which is done using the grid search method and K-Fold Cross-Validation is the evaluation technique used to ensure the chosen hyperparameters perform well across different splits of the data.

    ML pipeline:

    1. StandardScaler: This step normalises the features so that they have a mean = 0 and a standard deviation = 1. Many algorithms (e.g., Logistic Regression, k-NN, Neural Networks) perform poorly if features are on very different scales. StandardScaler brings all features to the same scale, making optimization stable and distance-based models fair.
    2. SelectKBest: This is a feature selection method that picks the top K features according to a scoring function. Some common scoring functions: f_classif, mutual_info_classif, chi2, etc. The scoring function used is the f_classif function that uses the ANOVA F-value. It is used to
        a. Remove noisy or irrelevant features
        b. Reduce dimensionality
        c. Improve accuracy and training speed
    3. Classifier: this is the actual model that performs classification after preprocessing and feature selection. The classifiers used: Logistic Regression, Decision Tree and k-NN.

    Grid Search:

    1. Manual implementation: This approach (defined in run_manual_grid_search() function) implements a manual grid search

to find the best hyperparameters for a set of classifiers using cross-validation. The flow of the function:

   a. It iterates through all combinations of specified hyperparameters.
   b. Performs a 5-Fold Cross-Validation for each combination.
   c. The ML pipeline described above is integrated in that order.
   d. Trains the models on different folds of the training data.
   e. Evaluates their performance using AUC of the ROC curve, and selects the combination that yields the best average score.
   f. It trains the selected best model on the entire training dataset.

2. Scikit-Learn implementation: This approach (defined in run_builtin_grid_search() function) uses scikit-learn's built-in 'GridSearchCV' module to perform hyperparameter tuning for a set of classifiers. This method is convenient and reduces lines of code.
   a. It takes the training data and a list of classifiers with their respective parameter grids.
   b. Sets up a pipeline including scaling and feature selection, and then runs GridSearchCV with stratified 5-fold cross-validation to find the best hyperparameters based on AUC scoring.
   c. The function returns a dictionary containing the best estimator, best cross-validation score, and best parameters for each classifier.
   d. Additionally, parallel processing is achieved by setting the n_jobs parameter to -1, which uses all the available cores.

Results and Analysis

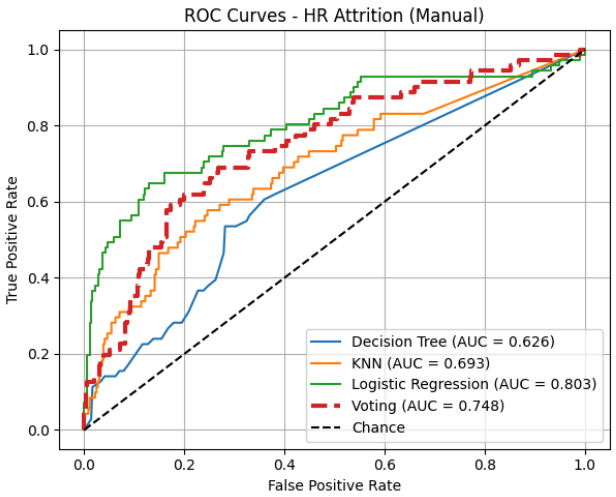
Model	Manual / scikit-learn	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree		0.7937	0.2727	0.1690	0.2087	0.6260
		0.7937	0.2727	0.1690	0.2087	0.6260
k-NN		0.8413	0.5556	0.0704	0.1250	0.6933
		0.8413	0.5556	0.0704	0.1250	0.6933
Logistic Regression		0.8776	0.7297	0.3803	0.5000	0.8033
		0.8776	0.7297	0.3803	0.5000	0.8033
Voting Classifier		0.8481	0.6429	0.1268	0.2118	0.7478
		0.8435	0.5625	0.1268	0.2069	0.7478

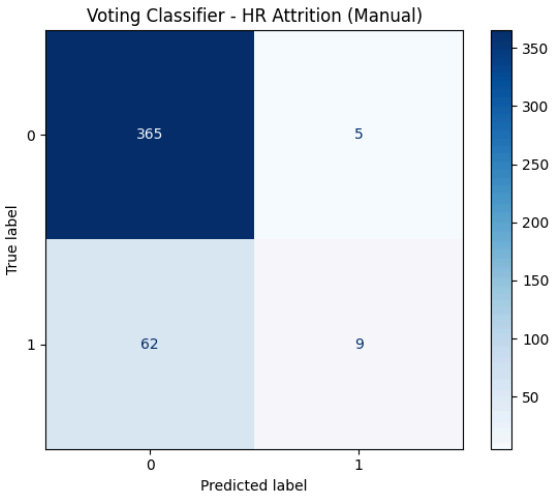
The Logistic Regression model had the best AUC score

Model		Accuracy	Precision	Recall	F1-Score	ROC AUC
Logistic Regression	Manual	0.8776	0.7297	0.3803	0.5000	0.8033
	scikit-learn	0.8776	0.7297	0.3803	0.5000	0.8033

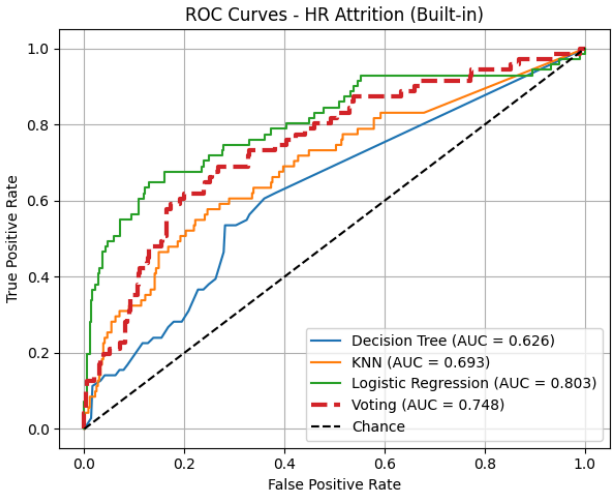
The models in both implementations produce identical results and minor differences in the scores of the voting classifier. Since the results are consistent, the manual implementation is correct.

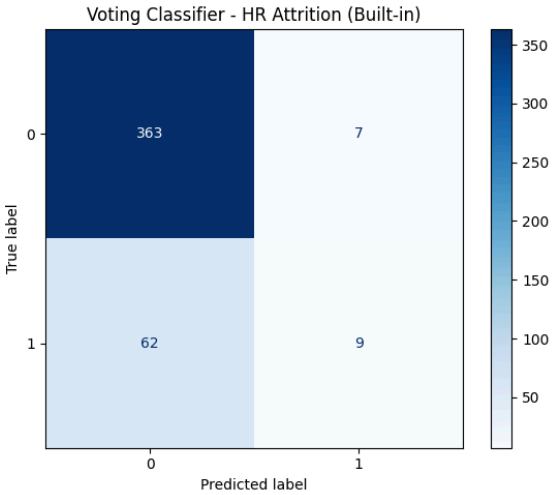
ROC and Confusion Matrix:
Manual implementation





Scikit-Learn implementation:

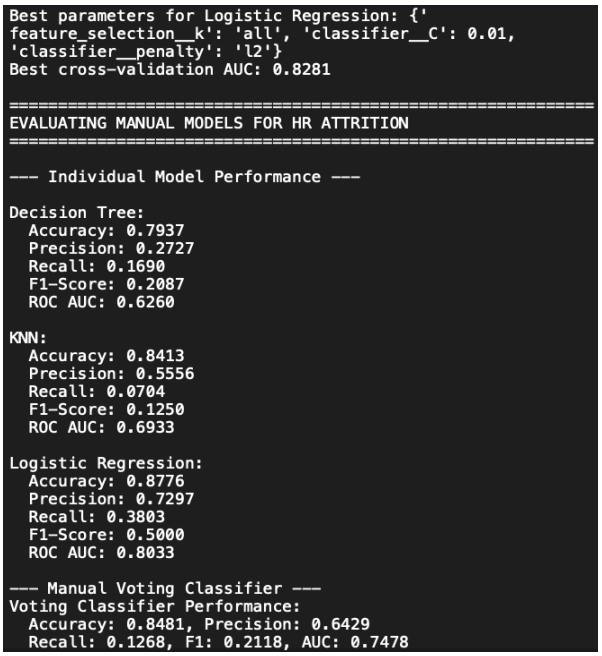


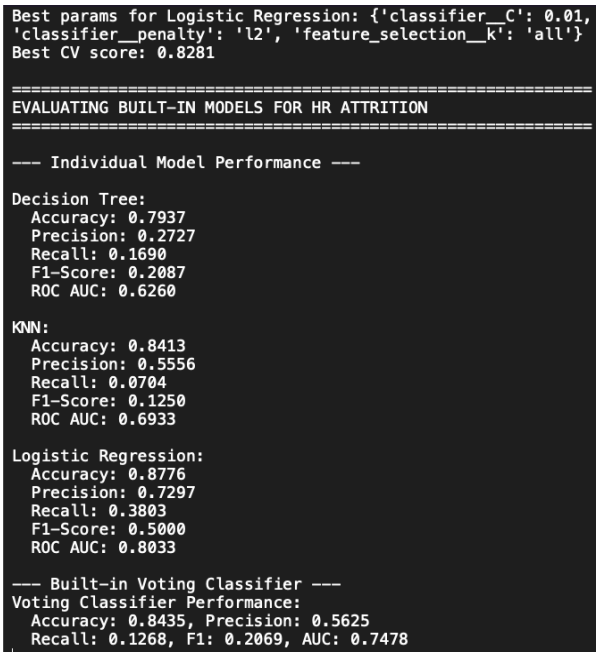


The 3 models in both of the approaches show nearly identical ROC curves. The Logistic Regression model is the most reliable model for this HR attrition dataset due to having the highest AUC score of 0.803, while decision tree is the least reliable with the lowest AUC score of 0.626

Logistic regression performed the best due to the probable existence of linear relationships between many of the features and the target. Since logistic regression assumes linear relationship, it is a good fit for this problem. Logistic regression is also less prone to overfitting and class imbalance compared to k-NN and decision tree models.

Output Screenshots





Conclusion

From the results, we can say that Logistic Regression is the model that is best suited for the HR Attrition problem. Both the manual and built-in functions determined that logistic regression with the hyperparameters set to select all features, with regularisation 0.01 and penalty as l2 would give the best results. It is very reliable and robust compared to decision trees and k-NN models. We can also see that the manual and scikit-learn implementations produce nearly identical results, proving that the manual implementation is accurate.

Manual and Built-in trade-offs:

1. Scikit-learn provides a standard implementation which has less customisability compared to the manual function.
2. Manual implementation is more prone to errors requiring more debugging.
3. Scikit-learn function is more efficient and optimised and can scale well with larger datasets.
4. Manually implementing such functions can be time-consuming in real world projects.