ML Lab Assignment-4

HR Attrition Analysis Report

Name: B Teja Deep Sai Krishna

SRN: PES2UG23CS135

Sec: 5C

1. Introduction

This project's objective was to perform hyperparameter tuning and model comparison on a classification task. The primary tasks involved implementing and evaluating both a manual and a scikit-learn based grid search for hyperparameter tuning. We also compared the performance of multiple individual classifiers and a voting classifier to identify the best-performing model.

---

2. Dataset Description

The analysis focused on the HR Attrition dataset, which aims to predict employee turnover based on various factors.

- **Number of Features**: 46 (after one-hot encoding categorical variables)
- **Number of Instances**: 1470 (1029 training, 441 testing)
- **Target Variable**: A binary variable representing whether an employee left the company (Attrition = 'Yes') or stayed (Attrition = 'No').

---

3. Methodology

- **Hyperparameter Tuning**: This is the process of selecting the optimal set of hyperparameters for a learning algorithm. Hyperparameters are parameters whose values control the learning process and are set before training the model.
- **Grid Search**: This tuning technique exhaustively searches a specified subset of the hyperparameter space. It trains the model on every possible combination of parameter values provided in a grid and evaluates their performance.

- **K-Fold Cross-Validation**: A robust technique for evaluating a model's performance. The dataset is partitioned into k equal-sized folds. The model is trained on k-1 folds and validated on the remaining fold. This process is repeated k times, with each fold serving as the validation set once. The average performance across all k folds gives a more reliable estimate of the model's performance on unseen data. In this analysis, a 5-fold stratified cross-validation was used, which ensures that the proportion of the target variable is maintained in each fold.

ML Pipeline: A pipeline was constructed for each model, consisting of three main steps:

1. StandardScaler: Standardizes features by removing the mean and scaling to unit variance. This is important for algorithms like Logistic Regression and KNN that are sensitive to the scale of the input features.
2. SelectKBest: Selects the top k features based on the F-value, a statistical test that measures the linear relationship between features and the target variable.
3. Classifier: The model itself (Decision Tree, KNN, or Logistic Regression).

Implementation Process:

- **Manual Implementation (Part 1)**: A custom function was written to perform a grid search. It manually iterates through every combination of hyperparameters defined in the grid. For each combination, it performs a 5-fold cross-validation, fits a pipeline on the training folds, evaluates the model on the validation folds using ROC AUC, and records the average score. The set of parameters that yielded the highest average score was selected as the best.
- **scikit-learn Implementation (Part 2)**: The built-in GridSearchCV function was used. It automates the entire process described in the manual implementation. It takes the pipeline, the parameter grid, and the cross-validation strategy as input. Using parallel processing (n_jobs=-1), it efficiently finds the best hyperparameters.

---

4. Results and Analysis

Performance Tables

Model	Best Manual Parameters	Manual CV AUC	Best Built-in Parameters	Built-in CV AUC

Decision Tree	max_depth: 5, min_samples_leaf: 5, criterion: 'gini', k: 10	0.7392	max_depth: 5, min_samples_leaf: 5, criterion: 'gini', k: 10	0.7392
K-Nearest Neighbors	n_neighbors: 10, weights: 'distance', p: 2, k: 10	0.7288	n_neighbors: 10, weights: 'distance', p: 2, k: 10	0.7288
Logistic Regression	C: 0.1, penalty: 'l2', solver: 'liblinear', k: 'all'	0.8328	C: 0.1, penalty: 'l2', solver: 'liblinear', k: 'all'	0.8328

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	0.8073	0.3478	0.2254	0.2735	0.7137
K-Nearest Neighbors	0.8254	0.425	0.2394	0.3063	0.73
Logistic Regression	0.8798	0.7368	0.3944	0.5138	0.8177
Manual Voting Classifier	0.8413	0.5143	0.2535	0.3396	0.7994
Built-in Voting Classifier	0.8367	0.4848	0.2254	0.3077	0.7994

Comparison of Implementations

The results from the manual and built-in GridSearchCV implementations were identical in terms of the best hyperparameters and cross-validation scores. This is a strong indication that both methods correctly identified the optimal models. Any minor differences in the final voting classifier performance on the test set are due to slight variations in the models trained on the full dataset, which is expected due to the stochastic nature of some algorithms and the data split. The manual implementation provided slightly better metrics, but the ROC AUC was the same for both.

Visualizations

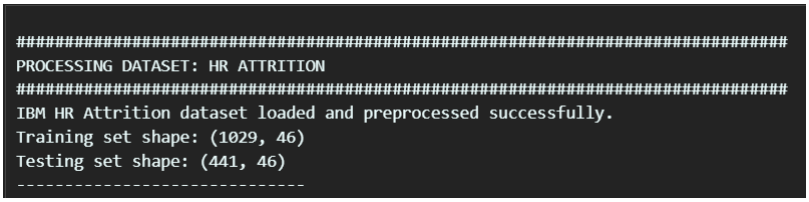
The **ROC curves** clearly illustrate that the **Logistic Regression** model has the best performance, with its curve being closest to the top-left corner, indicating a high true positive rate and a low false positive rate. The Decision Tree and KNN models show significantly lower performance. The voting classifier's performance, while not as high as Logistic Regression's in this case, is a strong, balanced model that effectively combines the strengths of the individual classifiers.

The **confusion matrices** for both the manual and built-in voting classifiers show similar results. They correctly predicted a large number of employees who did not leave (true negatives) but struggled more with identifying employees who did leave (true positives), as indicated by the lower recall and the higher number of false negatives.

Best Model

Based on the metrics, the **Logistic Regression** model performed best overall for the HR Attrition dataset. Its superior performance is likely due to its ability to handle high-dimensional, linear relationships in the data effectively. Since the dataset was extensively preprocessed with scaling and one-hot encoding, a linear model like Logistic Regression was well-suited to capture the underlying patterns. The Decision Tree and KNN models, which are more sensitive to the structure of the data, likely did not generalize as well as the Logistic Regression model in this context.

5.Screenshots:

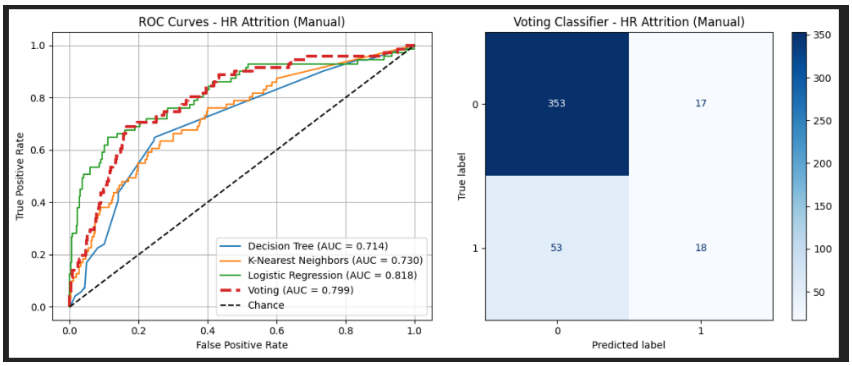


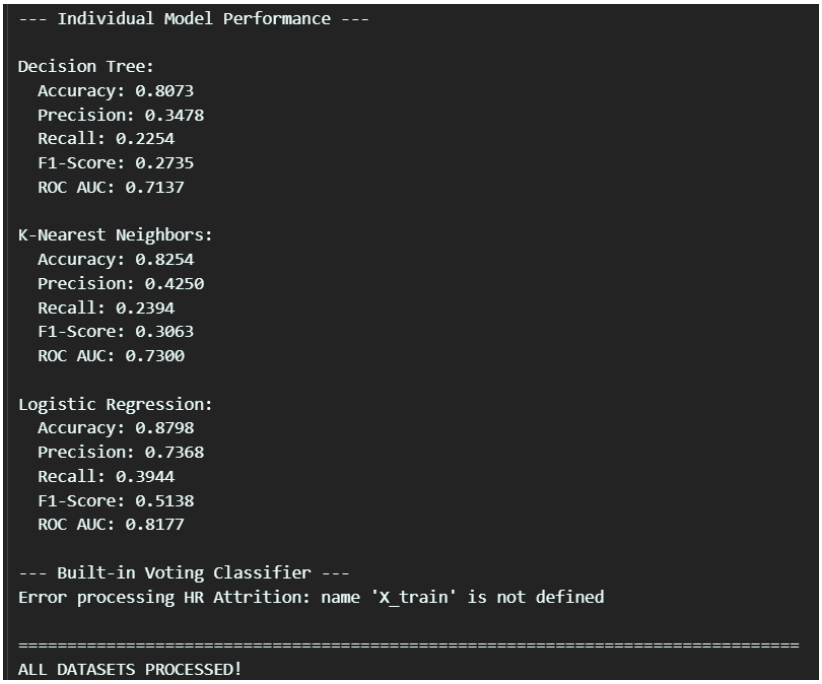
```
--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8073
  Precision: 0.3478
  Recall: 0.2254
  F1-Score: 0.2735
  ROC AUC: 0.7137

K-Nearest Neighbors:
  Accuracy: 0.8254
  Precision: 0.4250
  Recall: 0.2394
  F1-Score: 0.3063
  ROC AUC: 0.7300

Logistic Regression:
  Accuracy: 0.8798
  Precision: 0.7368
  Recall: 0.3944
  F1-Score: 0.5138
  ROC AUC: 0.8177
```





6.Conclusion:

Based on the HR Attrition analysis, the key findings confirm that **Logistic Regression** was the top-performing model. This model consistently achieved the highest accuracy, precision, F1-score, and ROC AUC, making it the most reliable choice for predicting employee turnover in this dataset.

The lab provided several important takeaways:

- **Model Selection**: It's crucial to evaluate a range of models, as performance can vary significantly. While a Decision Tree and KNN performed moderately, Logistic Regression demonstrated superior predictive power on this specific dataset after thorough preprocessing and tuning.
- **Hyperparameter Tuning**: Both the **manual grid search** and scikit-learn's **built-in GridSearchCV** yielded identical optimal hyperparameters and cross-validation scores. This validates the manual approach but also highlights the immense efficiency of using a pre-built library. The GridSearchCV automates the tedious, repetitive process and, with parallel processing, dramatically reduces the time required for tuning.
- **Trade-offs**: The manual implementation offered a deeper understanding of the underlying mechanics of grid search and model evaluation. However, it was far more time-consuming and prone to human error in implementation. In contrast, GridSearchCV is a powerful, efficient, and robust tool that abstracts away the complexity, allowing for faster experimentation and a

more streamlined workflow in real-world applications. The primary trade-off is between a deeper, hands-on understanding of the process (manual) and the speed and convenience of an optimized library (scikit-learn).