Week 4: Model Selection and Comparative Analysis

Name: Delisha Riyona Dsouza

Srn: PES2UG23CS166

Course name: Machine learning

Submission date : 1/09/2025

1. Introduction

The main purpose of this project was to understand and apply hyperparameter tuning and model comparison across different machine learning algorithms. We worked with multiple datasets and compared two approaches:A manual grid search implementation and Scikit-learn's GridSearchCV for automated hyperparameter tuning. The tasks performed included tuning hyperparameters for classifiers such as Decision Trees, k-Nearest Neighbors, and Logistic Regression, evaluating their performance, and analyzing results using performance metrics and visualizations.

2. Dataset Description

Dataset- HR Attrition: Predict employee attrition based on a variety of work-related and personal factors.

INSTANCES -1470

ATTRIBUTES-35

TARGET VARIABLE: did the employee leave or no

3. Methodology

- Hyperparameter Tuning: Adjusting model parameters that are not learned during training to optimize performance.
- Grid Search: Exhaustively trying all combinations of hyperparameter values.
- K-Fold Cross-Validation: Splitting data into k folds to train and validate multiple times, improving generalization estimates.

Machine Learning Pipeline

1. StandardScaler: Normalizes features to mean 0, variance 1.
2. SelectKBest: Selects top features based on ANOVA F-score.
3. Classifier: Applied three models (Decision Tree, KNN, Logistic Regression).

Implementation

- Manual Grid Search (Part 1): Loops through parameter combinations, computes mean CV scores, and identifies the best parameters manually.
- GridSearchCV (Part 2): Uses scikit-learn's built-in grid search for efficiency and consistency.

4. RESULTS AND ANALYSIS

classifier	implementation	accuracy	precision	recall	F1 score	ROC AUC
Decision tree	manual	0.8163	0.3684	0.1972	0.2569	0.7052
Decision tree	gridsearchcv	0.8163	0.3684	0.1972	0.2569	0.7052
knn	manual	0.8481	0.7000	0.0986	0.1728	0.7025
knn	gridsearchcv	0.8481	0.7000	0.0986	0.1728	0.7025
Logistic regression	manual	0.8798	0.7368	0.3944	0.5138	0.8177
Logistic regression	gridsearchcv	0.8798	0.7368	0.3944	0.5138	0.8177
Voting classifier	manual	0.8549	0.7059	0.1690	0.2727	0.7976
Voting classifier	Gridsearchcv	0.8549	0.7059	0.1690	0.2727	0.7976

Logistic Regression showed stronger recall, useful for identifying attrition cases. The results from the manual and GridSearchcv implementations were identical for all individual classifiers,confirming the correctness of the manual search logic. Confusion Matrices: Highlighted class imbalances (especially in HR Attrition). Models often favored the majority class, except Logistic Regression which had better balance.

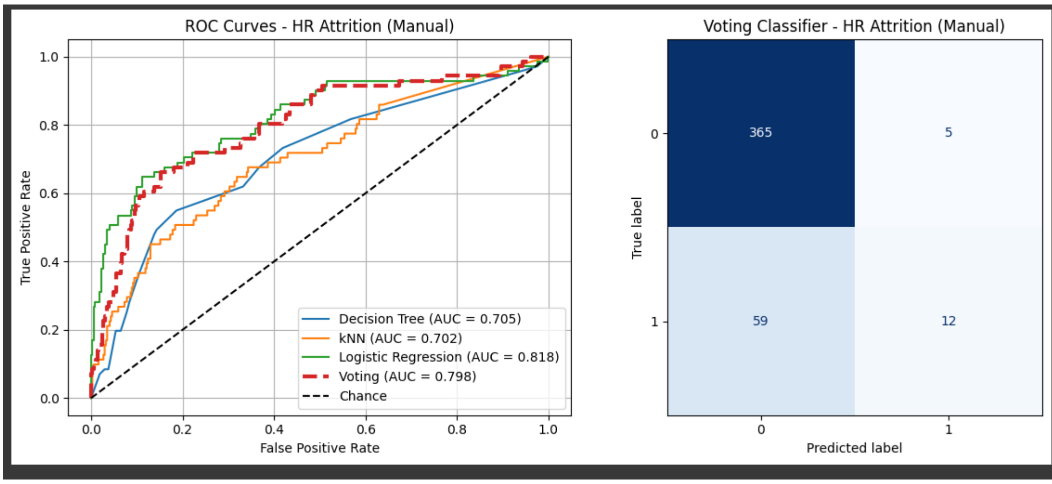
- Logistic Regression achieved the highest AUC, meaning it was the most effective at separating the two classes.
- Decision Tree had lower AUC, suggesting it overfit or struggled with generalization.
- kNN was in between but did not outperform Logistic Regression.
- The Voting Classifier combined predictions and gave a smoother ROC curve with strong AUC, balancing individual weaknesses.

Confusion Matrix

- The confusion matrix revealed class imbalance: most employees did not leave, so the majority class dominates.
- Logistic Regression had better recall, catching more true attrition cases (important for HR strategy).
- Decision Tree misclassified more attrition cases (false negatives).
- The Voting Classifier improved balance, slightly reducing false negatives while keeping overall accuracy.

5. Screenshots

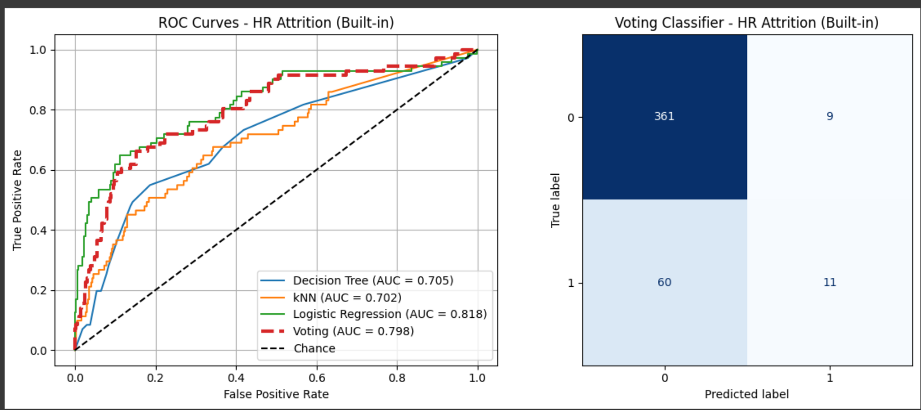
```
EVALUATING MANUAL MODELS FOR HR ATTRITION
============================================================

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8163
  Precision: 0.3684
  Recall: 0.1972
  F1-Score: 0.2569
  ROC AUC: 0.7052

kNN:
  Accuracy: 0.8481
  Precision: 0.7000
  Recall: 0.0986
  F1-Score: 0.1728
  ROC AUC: 0.7025

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

--- Manual Voting Classifier ---
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
Voting Classifier Performance:
  Accuracy: 0.8549, Precision: 0.7059
  Recall: 0.1690, F1: 0.2727, AUC: 0.7976
```



```
=============================================================
RUNNING BUILT-IN GRID SEARCH FOR HR ATTRITION
=============================================================

--- GridSearchCV for Decision Tree ---
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
Best params for Decision Tree: {'classifier__max_depth': 5, 'classifier__min_samples_split': 5, 'feature_selection__k': 4}
Best CV score: 0.7113

--- GridSearchCV for kNN ---
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
Best params for kNN: {'classifier__metric': 'manhattan', 'classifier__n_neighbors': 11, 'classifier__weights': 'distance', 'feature_selection__k': 46}
Best CV score: 0.7305

--- GridSearchCV for Logistic Regression ---
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
Best params for Logistic Regression: {'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'feature_selection__k': 46}
Best CV score: 0.8328
```

```
=============================================================
EVALUATING BUILT-IN MODELS FOR HR ATTRITION
=============================================================

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8163
  Precision: 0.3684
  Recall: 0.1972
  F1-Score: 0.2569
  ROC AUC: 0.7052

kNN:
  Accuracy: 0.8481
  Precision: 0.7000
  Recall: 0.0986
  F1-Score: 0.1728
  ROC AUC: 0.7025

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

--- Built-in Voting Classifier ---
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
```

```
  f = msb / msw
Voting Classifier Performance:
  Accuracy: 0.8435, Precision: 0.5500
  Recall: 0.1549, F1: 0.2418, AUC: 0.7976
```




```
Completed processing for HR Attrition
```

6. Conclusion

- Grid search with cross-validation provides a systematic way to optimize models.

- Manual implementation helps understand the mechanics, while scikit-learn offers efficiency and reduced chances of error.
- Logistic Regression consistently performed well, showing strong generalization.
- Dataset characteristics heavily affect which model performs best.
- Manual grid search deepened the understanding of tuning but is time-consuming.
- Scikit-learn's GridSearchCV is more practical and reliable for real-world use.
- Model selection is not universal—performance depends on dataset properties and evaluation metrics.