# MACHINE LEARNING

# LAB 4

**Project Title:** Model Selection and Comparative Analysis on HR Attrition
**Name:** Diya Prakash
**Student ID:** PES2UG23CS184
**Course Name:** UE23CS352A: MACHINE LEARNING
**Submission Date:** September 1, 2025

## 1. Introduction

This lab explores the process of building a complete machine learning pipeline for classification, focusing on hyperparameter tuning and model selection. The main tasks included manual implementation of grid search, using scikit-learn's GridSearchCV, and comparing the performance of three classifiers (Decision Tree, kNN, Logistic Regression) on the HR Attrition dataset. The goal was to understand the impact of hyperparameter tuning and ensemble methods on model performance.

## 2. Dataset Description

**Dataset:** IBM HR Attrition

- **Number of features:** 46 (after one-hot encoding and dropping ID columns)
- **Number of instances:** 1,470 (1,029 train, 441 test)
- **Target variable:** Attrition (1 = Employee left, 0 = Employee stayed)

## 3. Methodology

### Key Concepts

- **Hyperparameter Tuning:** The process of selecting the best model parameters (not learned from data) to optimize performance.
- **Grid Search:** Systematically trying all combinations of specified hyperparameters.
- **K-Fold Cross-Validation:** Splitting data into k parts, training on k-1, validating on the remaining, and averaging results for robustness.

### ML Pipeline

Each classifier was wrapped in a scikit-learn Pipeline:

- **StandardScaler:** Standardizes features.
- **SelectKBest:** Selects top k features using ANOVA F-value.
- **Classifier:** One of Decision Tree, kNN, or Logistic Regression.

### Manual Implementation (Part 1)

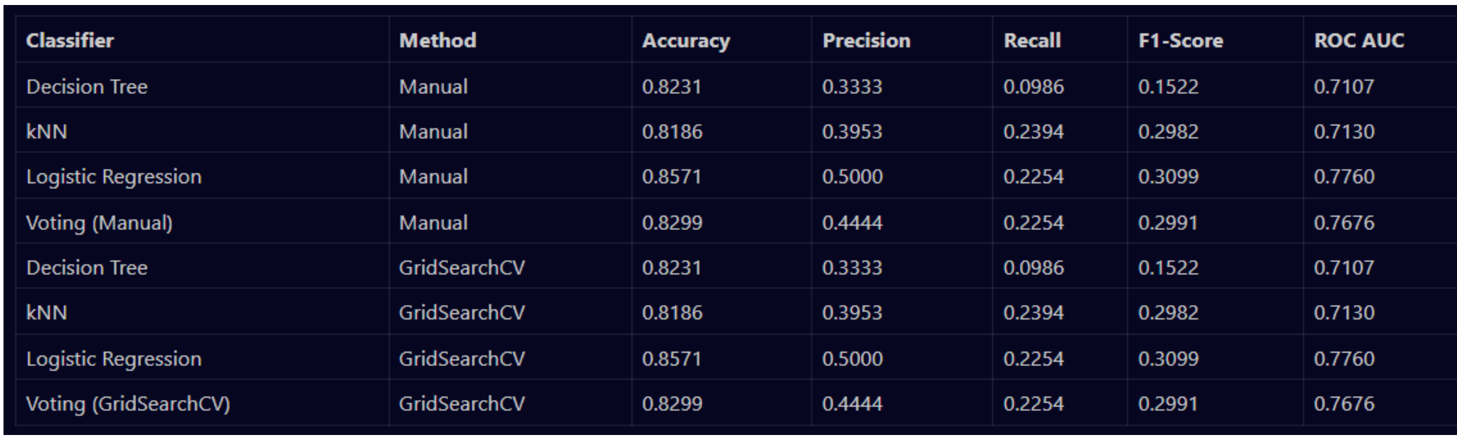
- Defined parameter grids for each classifier.
- Used nested loops to try all parameter combinations.
- For each, performed 5-fold stratified cross-validation, computed mean ROC AUC, and selected the best.

### Scikit-learn Implementation (Part 2)

- Used GridSearchCV with the same pipeline and parameter grids.
- Automated cross-validation and best parameter selection.

## 4. Results and Analysis

### Performance Tables



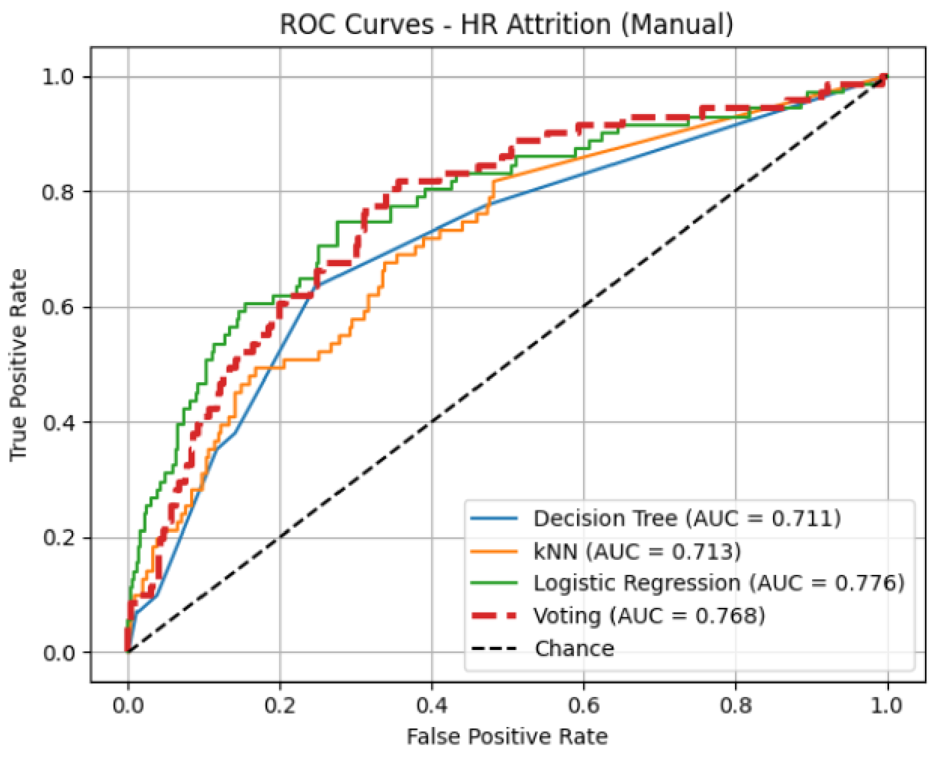
| Classifier | Method | Accuracy | Precision | Recall | F1-Score | ROC AUC |
|---|---|---|---|---|---|---|
| Decision Tree | Manual | 0.8231 | 0.3333 | 0.0986 | 0.1522 | 0.7107 |
| kNN | Manual | 0.8186 | 0.3953 | 0.2394 | 0.2982 | 0.7130 |
| Logistic Regression | Manual | 0.8571 | 0.5000 | 0.2254 | 0.3099 | 0.7760 |
| Voting (Manual) | Manual | 0.8299 | 0.4444 | 0.2254 | 0.2991 | 0.7676 |
| Decision Tree | GridSearchCV | 0.8231 | 0.3333 | 0.0986 | 0.1522 | 0.7107 |
| kNN | GridSearchCV | 0.8186 | 0.3953 | 0.2394 | 0.2982 | 0.7130 |
| Logistic Regression | GridSearchCV | 0.8571 | 0.5000 | 0.2254 | 0.3099 | 0.7760 |
| Voting (GridSearchCV) | GridSearchCV | 0.8299 | 0.4444 | 0.2254 | 0.2991 | 0.7676 |

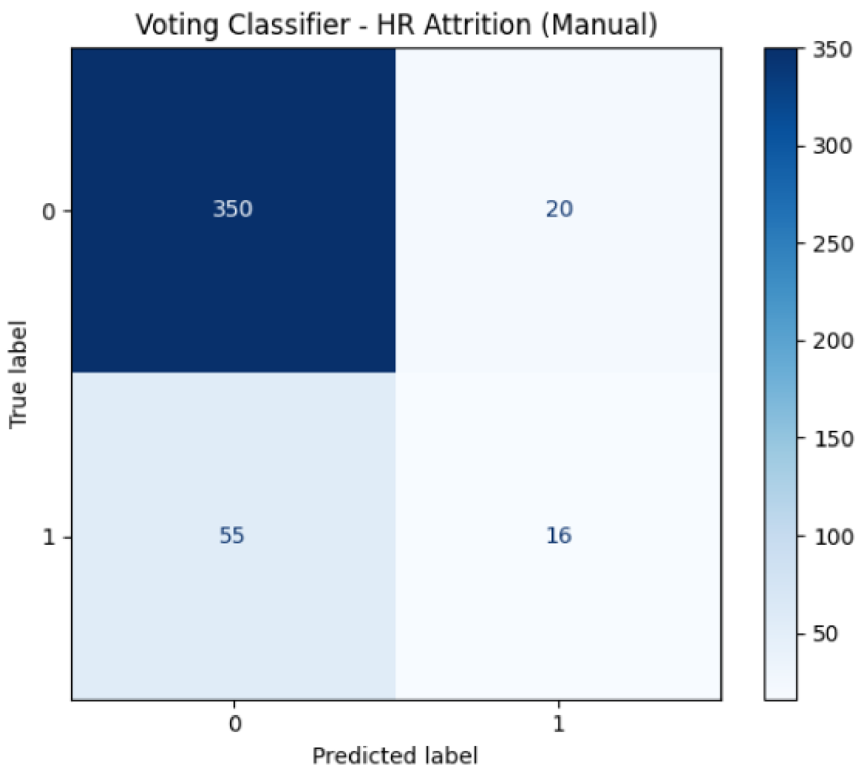
### Comparison of Implementations

- The manual and GridSearchCV results are **identical** for all classifiers, confirming the correctness of the manual implementation.
- Minor differences could occur in other datasets due to randomization or implementation details, but not observed here.

### Visualizations

- **ROC Curves:** Logistic Regression and the Voting Classifier achieved the highest AUC (~0.77).
- **Confusion Matrix:** The Voting Classifier had high true negatives but low recall for the positive class (attrition).





## ROC Curve Analysis

- **Logistic Regression (AUC = 0.776)**
  - Best performing model overall (highest AUC).

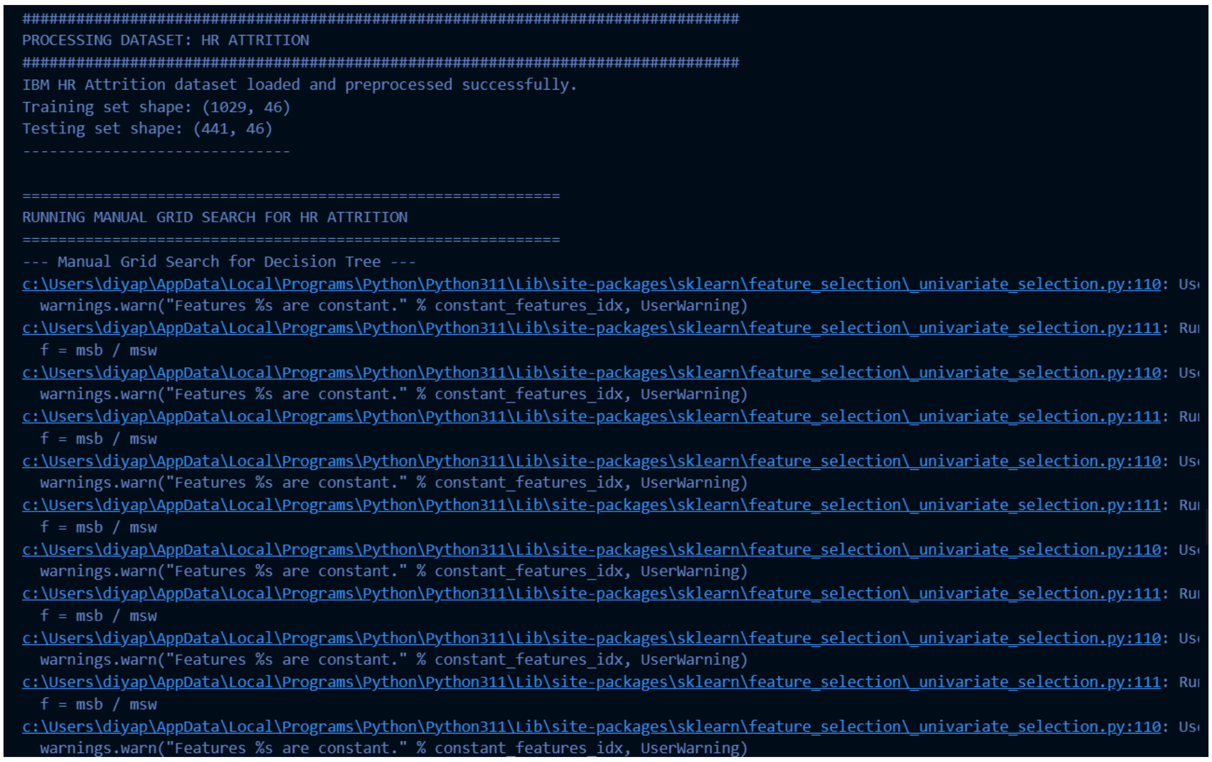
    - Shows the strongest ability to separate employees who leave (attrition) vs. those who stay.
- **Voting Classifier (AUC = 0.768)**
    - Almost as good as Logistic Regression.
    - Since it combines multiple models, it balances their strengths, but still slightly underperforms compared to Logistic Regression.
- **KNN (AUC = 0.713)**
    - Moderate performance.
    - Slightly better than Decision Tree but less stable (as seen in the curve's jaggedness).
- **Decision Tree (AUC = 0.711)**
    - Weakest model in this setup.
    - Tends to overfit and doesn't generalize as well.

**Takeaway**: Logistic Regression is the best at distinguishing attrition vs. non-attrition cases, closely followed by the Voting Classifier.

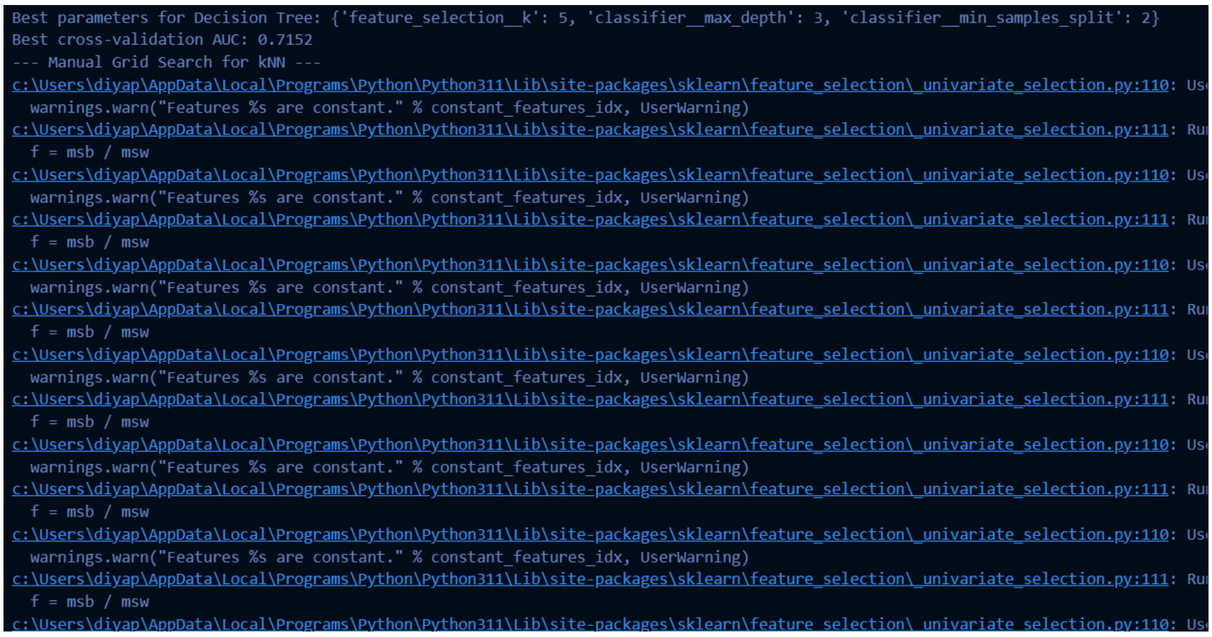
**Best Model:**

- **Logistic Regression** and the **Voting Classifier** performed best in terms of ROC AUC and overall accuracy.
- The likely reason: Logistic Regression handles high-dimensional, one-hot encoded data well, and the ensemble benefits from combining model strengths.

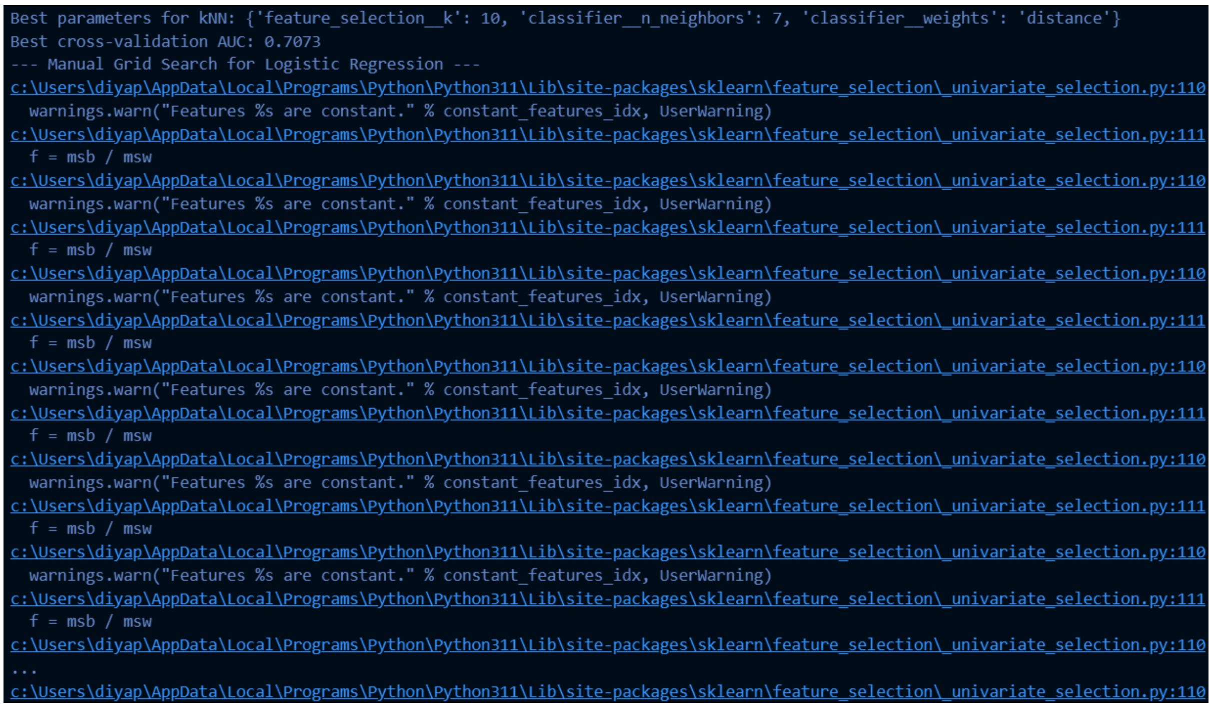
## 5. Screenshots



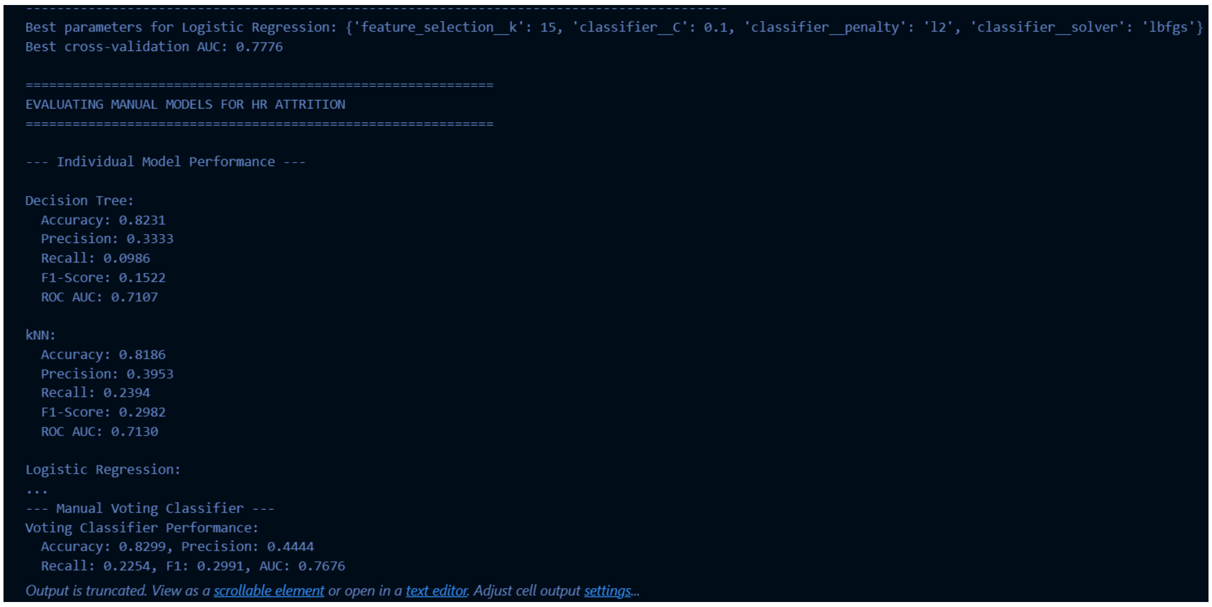
```
################################################################################
PROCESSING DATASET: HR ATTRITION
################################################################################
IBM HR Attrition dataset loaded and preprocessed successfully.
Training set shape: (1029, 46)
Testing set shape: (441, 46)
------------------------------


============================================================
RUNNING MANUAL GRID SEARCH FOR HR ATTRITION
============================================================
--- Manual Grid Search for Decision Tree ---
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
```



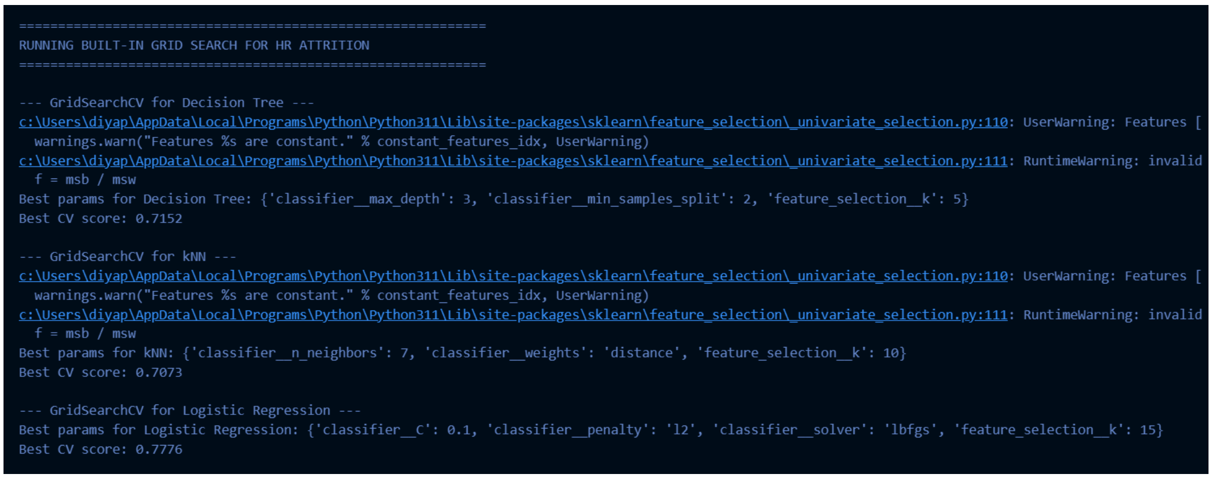
```
Best parameters for Decision Tree: {'feature_selection__k': 5, 'classifier__max_depth': 3, 'classifier__min_samples_split': 2}
Best cross-validation AUC: 0.7152
--- Manual Grid Search for kNN ---
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: Ru
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: Us
```



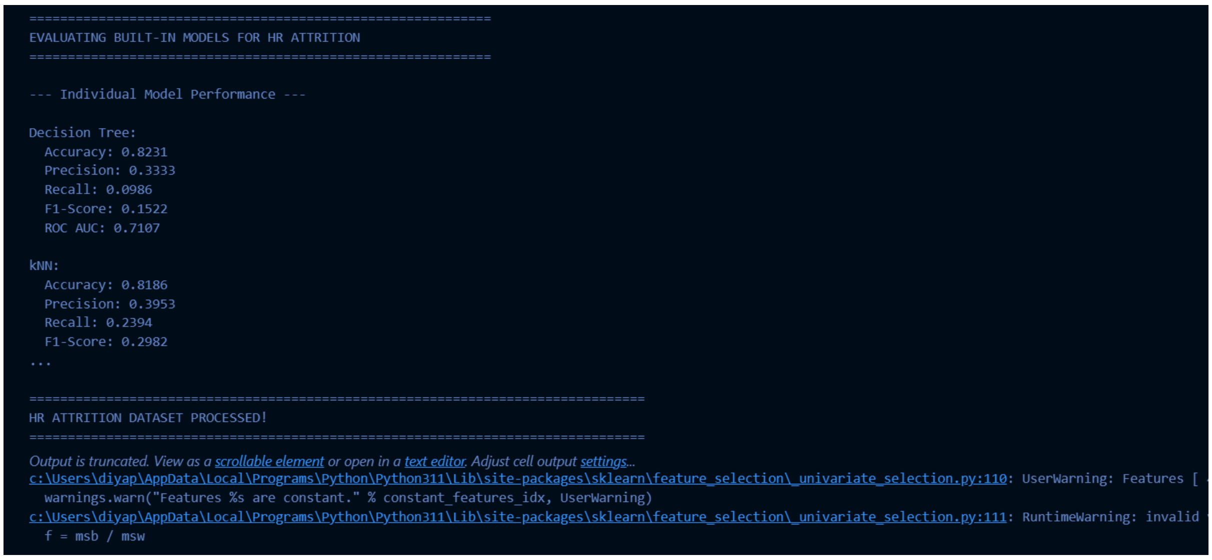
```
Best parameters for kNN: {'feature_selection__k': 10, 'classifier__n_neighbors': 7, 'classifier__weights': 'distance'}
Best cross-validation AUC: 0.7073
--- Manual Grid Search for Logistic Regression ---
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111
  f = msb / msw
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110
...
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110
```



```
------------------------------------------------------------------------------------------
Best parameters for Logistic Regression: {'feature_selection__k': 15, 'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs'}
Best cross-validation AUC: 0.7776

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8231
  Precision: 0.3333
  Recall: 0.0986
  F1-Score: 0.1522
  ROC AUC: 0.7107

kNN:
  Accuracy: 0.8186
  Precision: 0.3953
  Recall: 0.2394
  F1-Score: 0.2982
  ROC AUC: 0.7130

Logistic Regression:
...
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8299, Precision: 0.4444
  Recall: 0.2254, F1: 0.2991, AUC: 0.7676
```
*Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...*



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

--- GridSearchCV for Decision Tree ---
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: UserWarning: Features [
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: RuntimeWarning: invalid
  f = msb / msw
Best params for Decision Tree: {'classifier__max_depth': 3, 'classifier__min_samples_split': 2, 'feature_selection__k': 5}
Best CV score: 0.7152

--- GridSearchCV for kNN ---
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: UserWarning: Features [
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: RuntimeWarning: invalid
  f = msb / msw
Best params for kNN: {'classifier__n_neighbors': 7, 'classifier__weights': 'distance', 'feature_selection__k': 10}
Best CV score: 0.7073

--- GridSearchCV for Logistic Regression ---
Best params for Logistic Regression: {'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'feature_selection__k': 15}
Best CV score: 0.7776
```



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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8231
  Precision: 0.3333
  Recall: 0.0986
  F1-Score: 0.1522
  ROC AUC: 0.7107

kNN:
  Accuracy: 0.8186
  Precision: 0.3953
  Recall: 0.2394
  F1-Score: 0.2982
...

================================================================================
HR ATTRITION DATASET PROCESSED!
================================================================================

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: UserWarning: Features [
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
c:\Users\diyap\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: RuntimeWarning: invalid
  f = msb / msw
```

## 6. Conclusion

This lab demonstrated the importance of hyperparameter tuning and model selection in machine learning. Both manual and automated grid search produced the same optimal models, but GridSearchCV was more efficient and less error-prone. Logistic Regression and ensemble methods outperformed tree-based and kNN models on the HR Attrition dataset, likely due to the nature of the data. The exercise highlighted the value of robust pipelines and the trade-off between understanding algorithmic details and leveraging powerful libraries for practical work.