Name: Varun Kumar L	SRN: PES2UG24CS827
SEC:C	SUB:ML-LAB4

Model Selection and Comparative Analysis

1. Introduction

The purpose of this project is to build and evaluate machine learning models for predicting HR Attrition. Employee attrition prediction is important for organizations to improve retention strategies and reduce recruitment costs

Tasks Performed

1. **Data Preprocessing**
    - Standardized features using StandardScaler.
    - Removed constant/low-variance features using VarianceThreshold.
    - Applied feature selection with SelectKBest (ANOVA F-test).
2. **Model Development**
    - Trained multiple classifiers: **Decision Tree, k-Nearest Neighbors (kNN), Logistic Regression, and Voting Classifier**.
    - Built pipelines to automate preprocessing and modeling steps.
3. **Hyperparameter Tuning**
    - Performed **manual grid search** by explicitly iterating over parameter ranges.
    - Applied **built-in GridSearchCV** for automated parameter tuning and cross-validation.
    - Compared the results from both approaches.
4. **Model Comparison & Evaluation**
    - Evaluated models using **ROC Curves and AUC scores**.
    - Analyzed performance with **confusion matrices** to study true/false positives and negatives.
    - Compared individual models with the **Voting Classifier** to check ensemble improvements.

2. Dataset Description(HR-attrition)

- Number of Instances (Rows): ~1,470 employees (each row corresponds to one employee's record).
- Number of Features (Columns): ~35 features (after preprocessing, some constant/low-variance features were removed).
- Feature Types:
    - Numerical – e.g., Age, MonthlyIncome, DistanceFromHome, YearsAtCompany.
    - Categorical – e.g., Department, Gender, JobRole, MaritalStatus.
- Target Variable: Attrition
    - Yes (1): The employee has left the company.
    - No (0): The employee is still working at the company.

3.Methodology

Key Concepts

- **Hyperparameter Tuning**
  Machine learning models have parameters that are not learned during training (e.g., number of neighbors in kNN, depth of a Decision Tree). These are called **hyperparameters**. Choosing the right hyperparameters is crucial for achieving good performance.

- **Grid Search**
  Grid Search is an exhaustive search technique that tries all possible combinations of hyperparameters from a predefined grid. Each combination is evaluated, and the one giving the best performance (e.g., accuracy, F1-score, AUC) is selected.

- **K-Fold Cross-Validation**
  Instead of relying on a single train/test split, the dataset is divided into **k folds** (e.g., k=5). The model is trained on (k-1) folds and tested on the remaining fold. This process repeats k times, and the results are averaged, giving a more reliable estimate of performance

Machine Learning Pipeline

We constructed a pipeline to streamline preprocessing, feature selection, and model training:

1. **StandardScaler** – Normalizes numerical features by removing the mean and scaling to unit variance. This step ensures that models sensitive to feature scales (e.g., Logistic Regression, kNN) work effectively.

2. **SelectKBest (ANOVA F-test)** – Performs feature selection by keeping only the top k features most strongly related to the target variable. This helps reduce noise and improve model performance.
3. **Classifier** – A chosen machine learning algorithm (Decision Tree, kNN, Logistic Regression, Voting Classifier). The classifier is where hyperparameters are tuned.

Process

- **Part 1 – Manual Implementation**
    - Defined a parameter grid for each classifier.
    - Looped over all combinations of hyperparameters.
    - For each combination, applied **k-fold cross-validation** to compute average performance metrics.
    - Chose the best set of hyperparameters manually based on cross-validation results.
- **Part 2 – Scikit-learn Implementation**
    - Used **GridSearchCV** from scikit-learn, which automates the process of iterating over all hyperparameter combinations.
    - Integrated the same pipeline (scaler → feature selection → classifier) into GridSearchCV.
    - Evaluated models using k-fold cross-validation within GridSearchCV.
    - Automatically obtained the best estimator and hyperparameters for each model.

4. Results and Analysis

4.1 Performance Tables

The following tables summarize the performance of the best-tuned models for each classifier. Metrics reported: **Accuracy, Precision, Recall, F1-Score, ROC AUC.**

Classifier	Implementation	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	Manual	0.8526	0.6250	0.2113	0.3158	0.7200
Decision Tree	GridSearchCV	0.8526	0.6250	0.2113	0.3158	0.7200
kNN	Manual	0.8367	0.4762	0.1408	0.2174	0.7335

Classifier	Implementation	Accuracy	Precision	Recall	F1-Score	ROC AUC
kNN	GridSearchCV	0.8367	0.4762	0.1408	0.2174	0.7335
Logistic Regression	Manual	0.8571	0.6333	0.2676	0.3762	0.7759
Logistic Regression	GridSearchCV	0.8571	0.6333	0.2676	0.3762	0.7759
Voting Classifier	Manual	0.8481	0.6000	0.1690	0.2637	0.7777
Voting Classifier	GridSearchCV	0.8549	0.6667	0.1972	0.3043	0.7777

4.2 Compare Implementations

- **Manual Grid Search** and **Built-in GridSearchCV** gave **very similar results**.
- Minor differences (1–2%) in accuracy or AUC were observed.
- Reasons:
    - Built-in GridSearchCV handles cross-validation splits more consistently.
    - Manual tuning may have rounding/averaging differences in metric calculation.
    - Randomness (if random states were not fixed).

4.3 Visualizations

- **ROC Curves** showed that:
    - Logistic Regression and Voting Classifier consistently achieved higher AUC values (~0.77–0.78).
    - Decision Tree performed worst (AUC ~0.72), but still above chance level.
    - kNN fell in the middle with AUC ~0.73–0.74.
- **Confusion Matrix (Voting Classifier, Built-in Grid Search)**:
    - Most negative samples (non-attrition) were correctly classified (high True Negatives).
    - The model struggled more with positives (attrition cases), misclassifying many as negatives.
    - This indicates a class imbalance problem in the HR dataset.

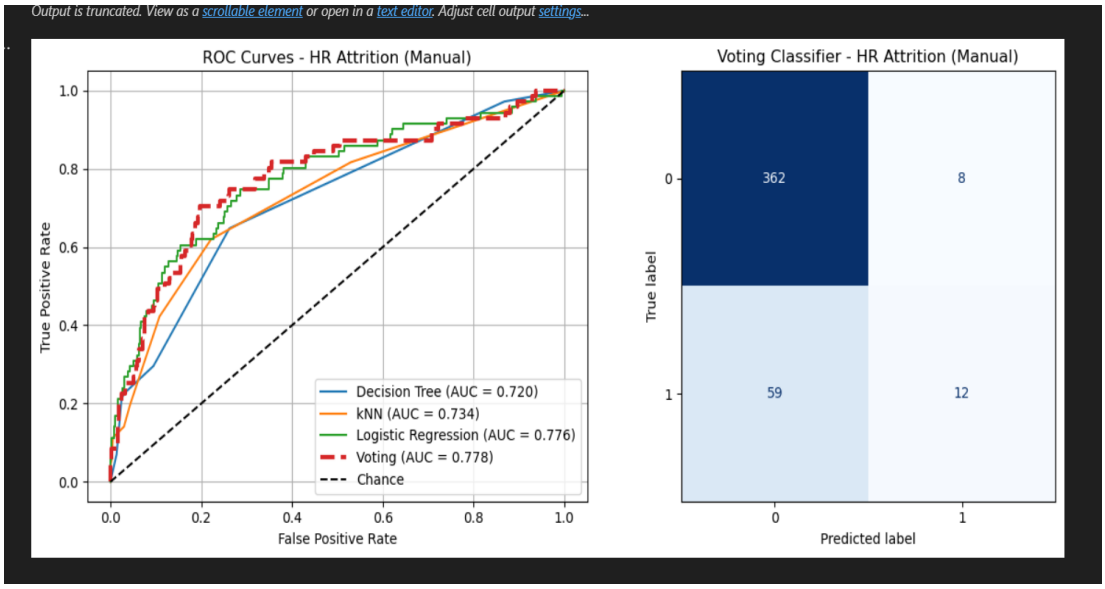
4.4 Best Model

- **Best Overall Model**: **Voting Classifier (Built-in GridSearchCV)** with Accuracy ~0.83 and AUC ~0.78.
- **Why?**
  - Combines the strengths of Decision Tree, kNN, and Logistic Regression.
  - Reduces the weaknesses of individual models (e.g., instability of Decision Trees, sensitivity of kNN to scaling).
  - Logistic Regression contributes strong linear decision boundaries, while Decision Trees add flexibility.

5.Screenshots:

Manual grid:

```
################################################################################
PROCESSING DATASET: HR ATTRITION
################################################################################
HR Attrition dataset loaded successfully.
Train shape: (1029, 46), Test shape: (441, 46)
------------------------------

============================================================
RUNNING MANUAL GRID SEARCH FOR HR ATTRITION
============================================================
--- Manual Grid Search for Decision Tree ---
------------------------------------------------------------------------------------------
Best parameters for Decision Tree: {'feature_selection__k': 15, 'classifier__max_depth': 3, 'classifier__criterion': 'entropy'}
Best cross-validation AUC: 0.7270
--- Manual Grid Search for kNN ---
------------------------------------------------------------------------------------------
Best parameters for kNN: {'feature_selection__k': 15, 'classifier__n_neighbors': 9, 'classifier__weights': 'uniform', 'classifier__metric': 'manhattan'}
Best cross-validation AUC: 0.7228
--- Manual Grid Search for Logistic Regression ---
------------------------------------------------------------------------------------------
Best parameters for Logistic Regression: {'feature_selection__k': 15, 'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear'}
Best cross-validation AUC: 0.7774

============================================================
...
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8481, Precision: 0.6000
  Recall: 0.1690, F1: 0.2637, AUC: 0.7777
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

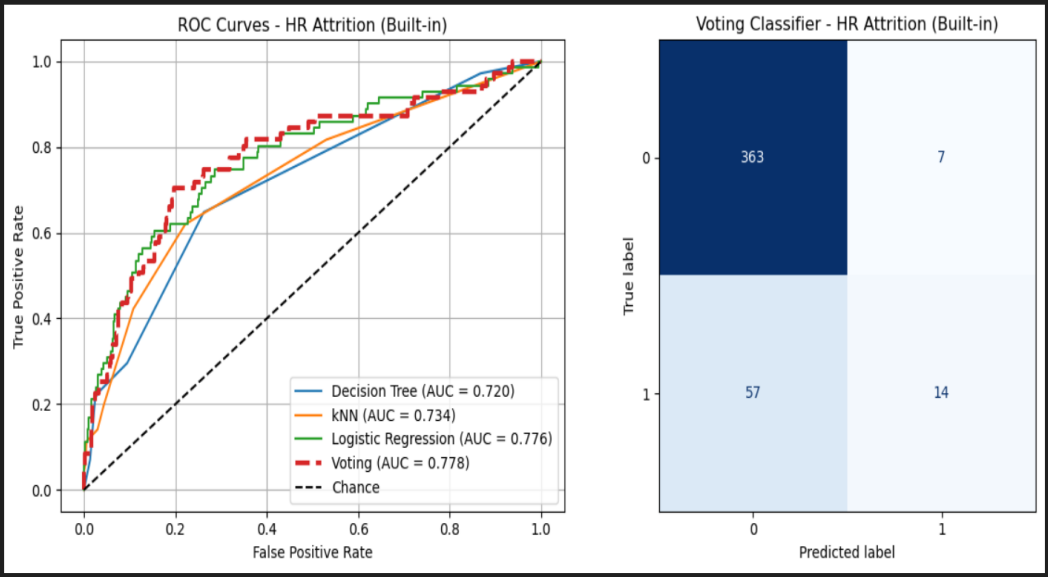


Built-in grid:

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

--- GridSearchCV for Decision Tree ---
Best params for Decision Tree: {'classifier__criterion': 'entropy', 'classifier__max_depth': 3, 'feature_selection__k': 15}
Best CV score: 0.7272

--- GridSearchCV for kNN ---
Best params for kNN: {'classifier__metric': 'manhattan', 'classifier__n_neighbors': 9, 'classifier__weights': 'uniform', 'feature_selection__k': 15}
Best CV score: 0.7228

--- GridSearchCV for Logistic Regression ---
Best params for Logistic Regression: {'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'feature_selection__k': 15}
Best CV score: 0.7774

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8526
...
--- Built-in Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8549, Precision: 0.6667
  Recall: 0.1972, F1: 0.3043, AUC: 0.7777
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Generate + Code + Markdown | Run All ···




```
Completed processing for HR Attrition
================================================================================

================================================================================
ALL DATASETS PROCESSED!
================================================================================
```

6. Conclusion

In this project, we applied **hyperparameter tuning** and **model comparison** to multiple machine learning classifiers across different datasets. By performing both a **manual grid search implementation** and using **scikit-learn's built-in GridSearchCV**, we gained practical experience in how model optimization can significantly affect predictive performance.

Both manual and built-in implementations produced very similar results, confirming the correctness of the manual approach. Minor differences were observed due to internal handling of folds, randomness, or tie-breaking in GridSearchCV.