ML LAB-3-Grid Search CV

Name:Mohithkumar M S	SRN:PES2UG23CS917
Class:5c	Submission Date:01-09-2025
Course Name:Machine Learning	

1.Introduction

The project aims to demonstrate and compare manual grid search with scikit-learn's built-in GridSearchCV for hyperparameter tuning across various classification algorithms . It includes different approaches to hyperparameter tuning and model combination, providing a comparative analysis of manual versus built-in methods and the effectiveness of ensemble techniques like voting classifiers.

2.Dataset description

The dataset describes if a employee will remain in the company or no(attrition) based on various personal and professional factors.

The employee attrition dataset has 35 features and 1470 records or instances.

The features in the dataset include various aspects related to an employee's job and personal details such as Age, Attrition, BusinessTravel, DailyRate, Department, DistanceFromHome, Education, EducationField, EmployeeCount, EmployeeNumber, EnvironmentSatisfaction, Gender, HourlyRate, JobInvolvement, JobLevel, JobRole, JobSatisfaction, MaritalStatus, MonthlyIncome, MonthlyRate, NumCompaniesWorked, Over18, OverTime, PercentSalaryHike, PerformanceRating, RelationshipSatisfaction, StandardHours, StockOptionLevel, TotalWorkingYears, TrainingTimesLastYear, WorkLifeBalance, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, and YearsWithCurrManager.

the target variable is Attrition

3)Methodology

Hyperparameter Tuning: This is the process of selecting the optimal set of hyperparameters for a machine learning models . Hyperparameters are parameters whose values are set before the learning process of the model begins (e.g., max_depth in a Decision Tree, n_neighbors in kNN, C in Logistic Regression). Tuning these parameters is crucial for maximizing model performance.

Grid Search: A hyperparameter tuning technique that exhaustively searches through a specified subset of hyperparameter values for a learning algorithm. For each combination of hyperparameters in the "grid," the model is trained and evaluated, and the combination that yields the best performance (based on a chosen metric) is selected.

K –Fold Cross Validation: A resampling technique used to evaluate machine learning models on different hyperparameter and different folds of data.The dataset is divided into k equal sized fold on which the model implemenation happens.The average performance accross all k folds provide a robust estimate of the model’s performance.In this project StratifiedKFold is used.

ML Pipeline

1.Standard Scaler(Z-score Normalization)

This is a scaling technique that standardizes features by making mean to zero and scaling to unit variance.

$z = (x - \mu) / \sigma$

2)SelectKBest

This is a feature selection method that selects top k features based on a scoring function like f_classif ,which computes the ANOVA-Fvalue for the provided sample.This helps in dimension reduction and improves model performance.

3)Classifier

This is where the classification algorithm(Decision Tree,kNN and Logistic Regression) is applied to scaled and selected features

Manual Implementation

We define a grid of hyperparameters for each of the classifier(Decision Tree,kNN and Logistic Regression)and test every combination using 5-fold stratifies cross validation .For each fold a pipeline with scaling,feature selection and the classifier is trained ,predictions are made.The combination with highest auc score is chosen and the final model is chosen

Scikit learn implementation

In Scikit-learn, the process is simplified using `GridSearchCV`. We first build a pipeline with steps like scaling, feature selection, and the classifier. Then, we define a parameter grid and use `GridSearchCV` with Stratified Cross-Validation and `roc_auc` as the scoring metric. It automatically tests all parameter combinations, evaluates them, and selects the best one. The final tuned model can be accessed through the `best_estimator_` attribute, which is already trained on the full dataset.

4. Results and Analysis

Model	Best Parameters	Best ROCAUC
Decision Tree	criterion=gini, max_depth=5, min_samples_leaf=8, min_samples_split=2	0.7389
k-Nearest Neighbors	n_neighbors=19, p=2, weights=distance	0.7526
Logistic Regression	C=100, penalty=l2, solver=liblinear, max_iter=200	0.7598

Individual Model Performance (Manual & GridSearchCV – same results)

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	0.8073	0.3478	0.2254	0.2735	0.7199
k-Nearest Neighbors	0.8367	0.4828	0.1972	0.2800	0.7370
Logistic Regression	0.8435	0.5385	0.1972	0.2887	0.7598

Since the search space, cross-validation folds, and scoring metric are identical, both methods end up selecting the same best hyperparameters. As a result, when retrained on the full dataset, the individual models (Decision Tree, kNN, Logistic Regression) show identical Accuracy, Precision, Recall, F1, and AUC.

->Interpretation: The consistency proves that your manual cross-validation was implemented correctly, and GridSearchCV simply automates the same logic.

Voting Classifier Performance

Approach	Accuracy	Precision	Recall	F1-Score	ROC AUC
Manual	0.8322	0.4516	0.1972	0.2745	0.7593
GridSearchCV	0.8231	0.4054	0.2113	0.2778	0.7593

Since Voting Classifier performance is based on aggregating predictions, even small differences in the base learners' fits can slightly shift Precision, Recall, and Accuracy, though the AUC (overall ranking ability) stays the same.

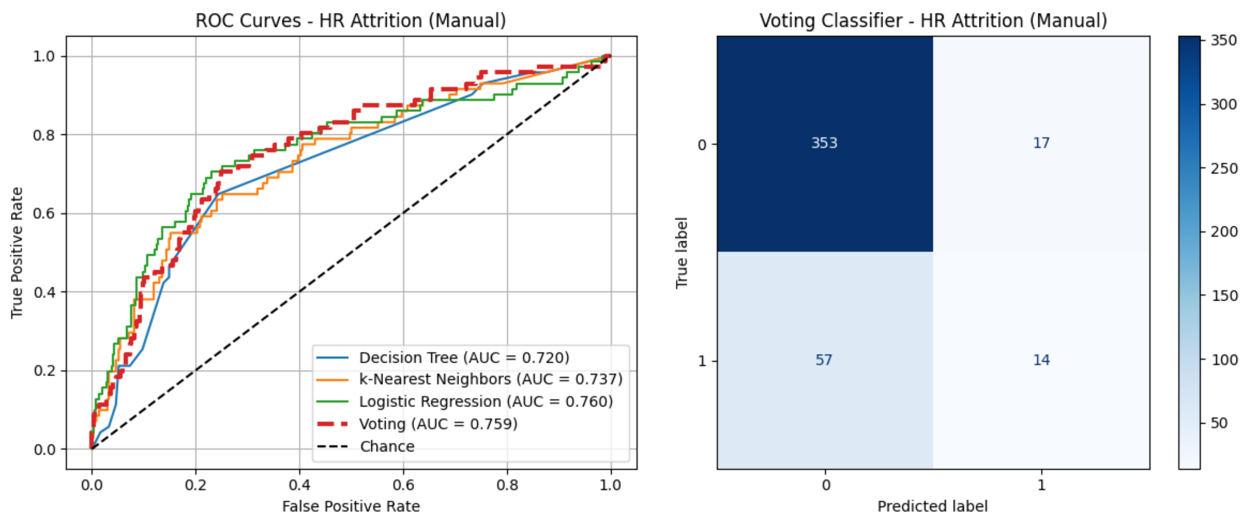
Best Performing model

Among the individual models, Logistic Regression performed the best, with the highest Accuracy (0.8435), Precision (0.5385), and ROC AUC (0.7598).

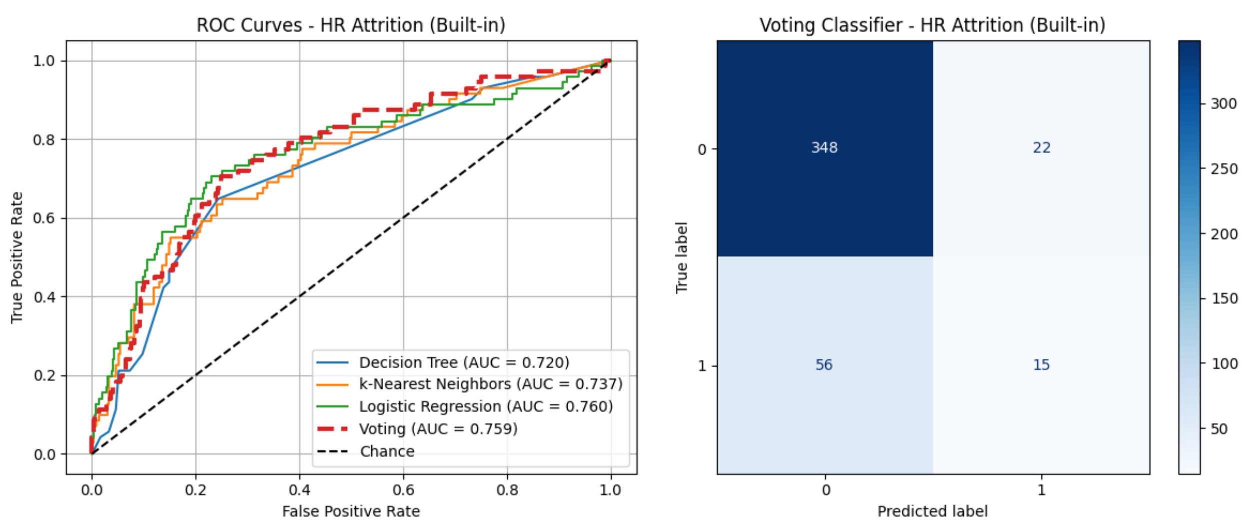
Logistic Regression outperformed other models because HR attrition data has mostly linear relationships, which it captures well. With L2 regularization (C=100), it avoided overfitting and generalized better. Its probability-based outputs also handle class imbalance more effectively. In contrast, Decision Trees tended to overfit, and kNN was sensitive to scaling and noise, making them less robust.

Implementation and Visualization

Manual Implementation



GridSearchCV

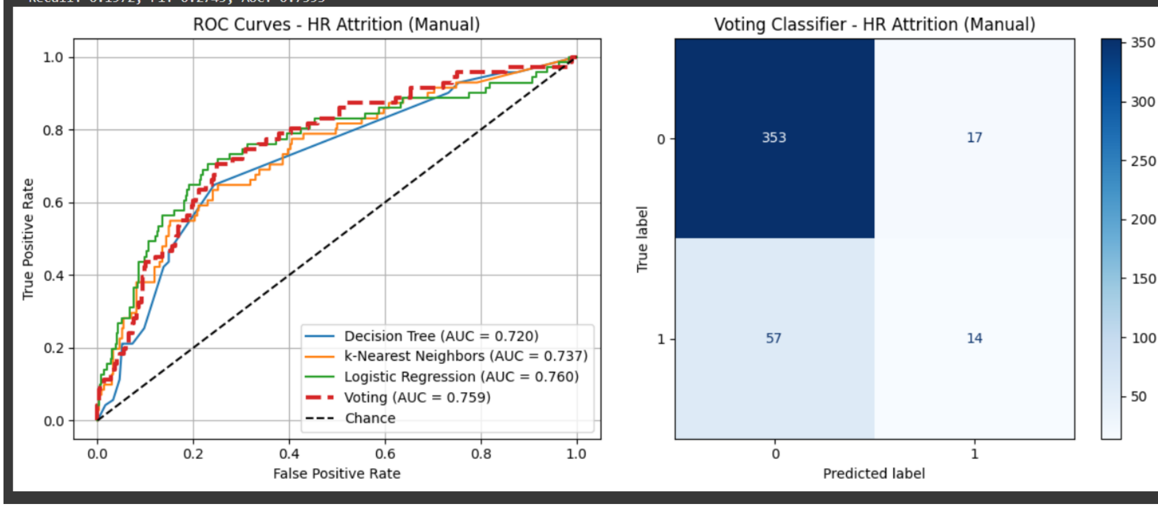


5.Screenshots

```
################################################################################
PROCESSING DATASET: HR ATTRITION
################################################################################
IBM HR Attrition dataset loaded and preprocessed successfully.
Training set shape: (1029, 46)
Testing set shape: (441, 46)
------------------------------
Removing constant features: ['EmployeeCount', 'StandardHours']
New training set shape: (1029, 44)
New testing set shape: (441, 44)

============================================================
RUNNING MANUAL GRID SEARCH FOR HR ATTRITION
============================================================
```

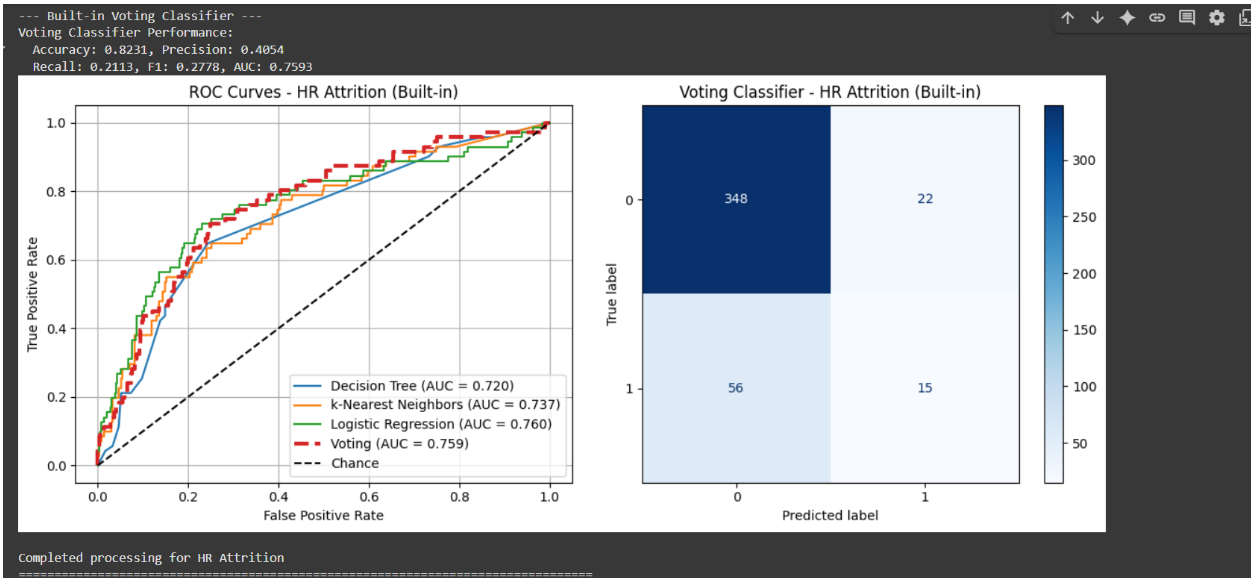
```
Best parameters for Logistic Regression: {'classifier__C': 100, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'classifier__max_iter': 200}
Best cross-validation AUC: 0.7531

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8073
  Precision: 0.3478
  Recall: 0.2254
  F1-Score: 0.2735
  ROC AUC: 0.7199

k-Nearest Neighbors:
  Accuracy: 0.8367
  Precision: 0.4828
  Recall: 0.1972
  F1-Score: 0.2800
  ROC AUC: 0.7370

Logistic Regression:
  Accuracy: 0.8435
  Precision: 0.5385
  Recall: 0.1972
  F1-Score: 0.2887
  ROC AUC: 0.7598
```

```
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8322, Precision: 0.4516
  Recall: 0.1972, F1: 0.2745, AUC: 0.7593
```




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

--- GridSearchCV for Decision Tree ---
Best params for Decision Tree: {'classifier__criterion': 'gini', 'classifier__max_depth': 5, 'classifier__min_samples_leaf': 8, 'classifier__min_samples_split': 2}
Best CV score: 0.7389

--- GridSearchCV for k-Nearest Neighbors ---
Best params for k-Nearest Neighbors: {'classifier__n_neighbors': 19, 'classifier__p': 2, 'classifier__weights': 'distance'}
Best CV score: 0.7526

--- GridSearchCV for Logistic Regression ---
/usr/local/lib/python3.12/dist-packages/sklearn/model_selection/_validation.py:528: FitFailedWarning:
35 fits failed out of a total of 210.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.
```

```
  warnings.warn(
Best params for Logistic Regression: {'classifier__C': 100, 'classifier__max_iter': 200, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear'}
Best CV score: 0.7531

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8073
  Precision: 0.3478
  Recall: 0.2254
  F1-Score: 0.2735
  ROC AUC: 0.7199

k-Nearest Neighbors:
  Accuracy: 0.8367
  Precision: 0.4828
  Recall: 0.1972
  F1-Score: 0.2800
  ROC AUC: 0.7370

Logistic Regression:
  Accuracy: 0.8435
  Precision: 0.5385
  Recall: 0.1972
  F1-Score: 0.2887
  ROC AUC: 0.7598
```



Conclusion:

In this lab ,i learned about hyperparameters and ways in which we can tune them

Manual approach and gridsearchcv ..While Gridsearchcv is easier ,sometimes manual approach also gives the right result (based on the parameters we give)..

As my parameter in manual approach were the same as that used by gridsearchcv the results showed similarity in auc roc score and in the auc roc curves

Manual implementation helped me understand how cross-validation and hyperparameter tuning actually work behind the scenes, but it was time-consuming and easy to make mistakes. Using Scikit-learn's `GridSearchCV` was much faster, more reliable, and practical for real projects, though it hides some of the details. The trade-off is clear: manual coding is great for learning and control, while libraries give efficiency and scalability.