PES UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MACHINE LEARNING - UE23CS352A

LAB 4

NAME : CHINMAYI SHRAVANI YELLANKI
SRN : PES2UG23CS152
CLASS AND SEC : SEM 5 - C
DATE : 01.09..2025

Introduction

The objective of this lab is building and comparing machine learning classification models by performing hyperparameter tuning approaches and evaluating the performance of models based on diverse metrics. It includes a manual grid search and Scikit-learn incorporated GridSearchCV for hyperparameter tuning and then performing model assessment with a comparison of Decision Tree, k-Nearest Neighbors (kNN), and Logistic Regression classifiers.

Important features include:

- To build and run a data preprocessing pipeline, feature selection pipeline, and training classifiers.
- Manual hyperparameter search of grids and automated GridSearchCV using 5-fold stratified cross-validation.
- Model evaluation using multiple metrics including Accuracy, Precision, Recall, F1-Score, and ROC AUC.
- Comparative study of implemented manual and built-in grid search.
- Combining classifiers by a voting ensemble.

Dataset description

Features count (after one-hot encoding): 46
Number of occurrences:

➔ Training set: 1029
➔ Test set: 441

Dependent variable: Attrition (binary), where 1 denotes employees left the company and 0 denotes employees who remained.
The data contains a few work and personal attributes which can possibly influence turnover of employees, i.e., job role, work environment satisfaction, years at company and overtime. All these categorical attributes are one-hot encoded into numerical attributes prior to modeling.

Methodology

Key Concepts

- ➢ Hyperparameter Tuning: The task of choosing the optimal combination of model parameters (e.g., depth of Decision Tree, number of neighbors in kNN) to optimize model performance.
- ➢ Grid Search: Exhaustively iterating over a manually defined subset of hyperparameters to identify the best combination according to model evaluation metrics.
- ➢ K-Fold Cross-Validation: A validation method that divides the training set into K subsets (folds). The model is trained on K-1 folds and tested on the hold-out fold, done K times to estimate a solid performance.

ML Pipeline

- ➔ A Scikit-learn pipeline was employed to pipe through preprocessing and model fitting, consisting of the following stages:
- ➔ StandardScaler: Normalizes features to zero mean and unit variance, providing even weighting of features.
- ➔ SelectKBest (f_classif): Does univariate feature selection by picking the best 'k' features most predictive of the target as ranked by ANOVA F-value.
- ➔ Classifier: One of Decision Tree, kNN, or Logistic Regression model trained on the chosen features.

Implementation Procedure

Part 1: Manual Grid Search Implementation

- Specified hyperparameter grids of each classifier with appropriate parameter names corresponding to pipeline steps.
- Implemented nested loops to generate all hyperparameter combinations.
- Executed 5-fold stratified cross-validation on all combinations.
- Computed average ROC AUC on folds.
- Select the highest ranked feature combination.
- Refit final pipeline with best parameters from full training data.

Part 2: Native GridSearchCV Implementation

- Implemented the same pipeline format for every classifier.
- Used Scikit-learn's GridSearchCV with 5-fold Strat.
- Automatically selected the best hyperparameters using scoring configured to ROC AUC.
- Extracted the best estimator and scores directly.
- Model Assessment and Voting Ensemble
- Implemented individual classifiers on test sets using Accuracy, Precision, Recall, F1-Score, and ROC AUC.

Combined classifiers with a voting classifier (manual and built-in) and checked ensemble performance. Visualized results using ROC Curves and Confusion Matrices.

Results and Analysis

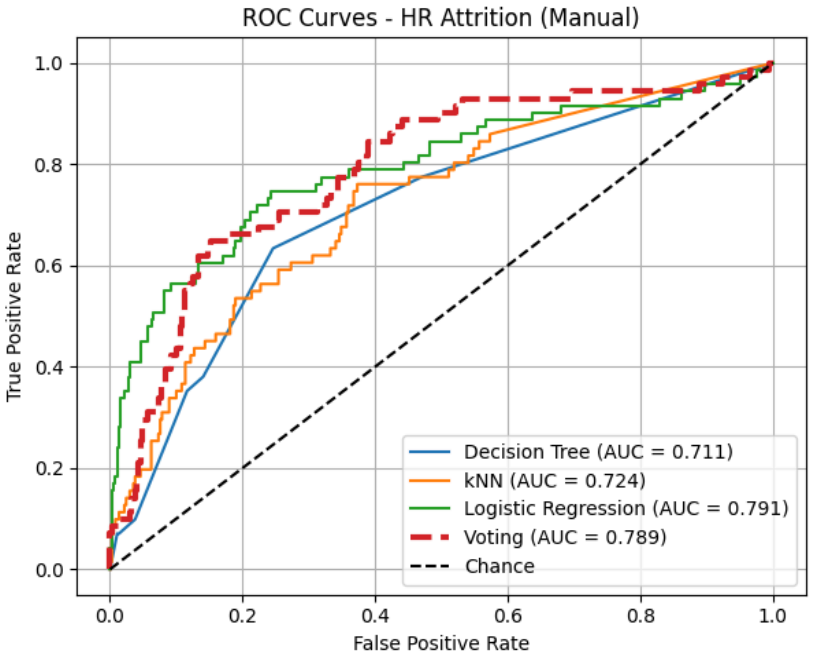
Method	Classifier	Accuracy	Precision	Recall	F1-Score	ROC AUC
Manual grid	Decision tree	0.8231	0.3333	0.0986	0.1522	0.7107
	KNN	0.8186	0.3784	0.1972	0.2593	0.7236
	Logistic Regression	0.8386	0.4667	0.3897	0.4211	0.7912
GridSearchCV	Decision tree	0.8231	0.3333	0.0986	0.1522	0.7107
	KNN	0.8186	0.3784	0.1972	0.2593	0.7236
	Logistic Regression	0.8386	0.4667	0.3897	0.4211	0.7912
Manual grid	Voting classifier	0.8254	0.4000	0.1690	0.2396	0.7886

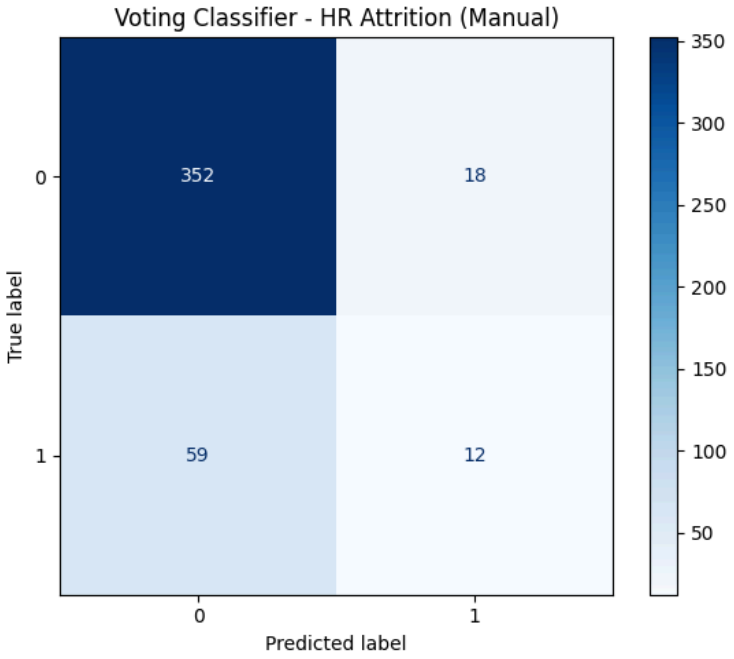
Comparison of Implementations

Manual grid search and GridSearchCV produce the same results since they share the same hyperparameter grids, cross-validation splits, performance metric, and data preprocessing. The two approaches inherently choose the same best parameters with the same settings and same data and therefore obtain the same performance, proving correctness and reproducibility of the grid search.

Visualizations

ROC Curves: All classifiers' ROC curves demonstrate a good discrimination capacity, while logistic regression possesses the largest area under the curve.
Confusion Matrices: Reflect class-wise accuracy of predictions and assist in determining the false positive/negative rates.
Voting Ensemble ROC curve is near to the logistic regression curve, validating ensemble's strong performance.
Voting Ensemble ROC curve is near to the logistic regression curve, validating ensemble's strong performance.



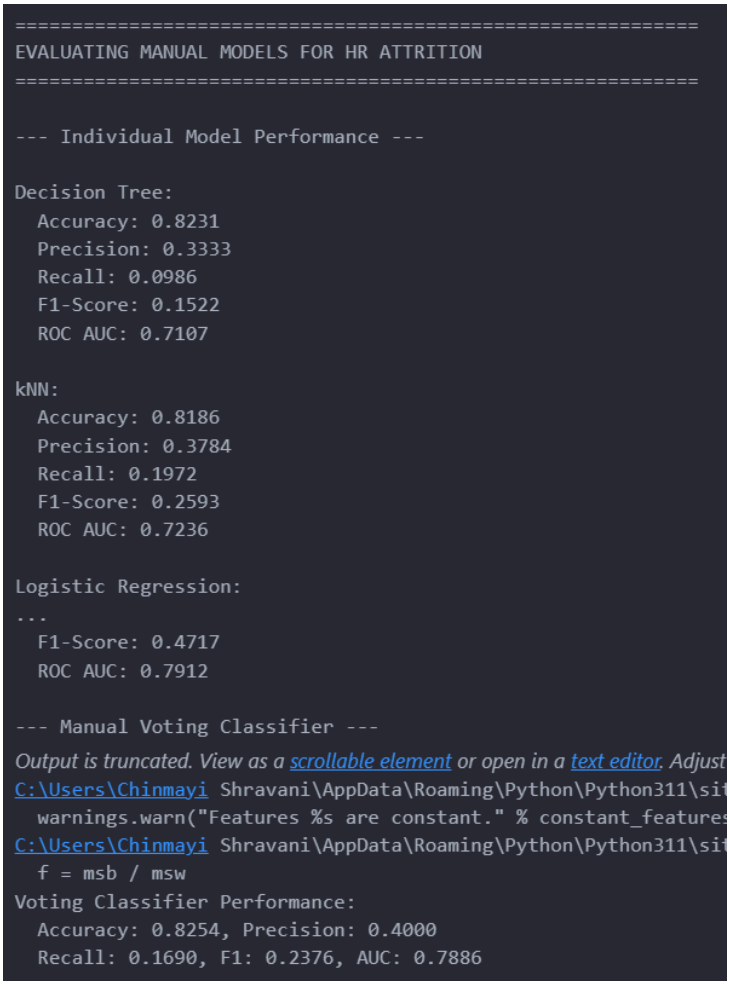


Best Model Analysis

Logistic Regression performs better than Decision Tree and kNN with the best ROC AUC and F1-score. This may be due to its capacity to simulate linear relationships and probabilistic outputs that best suit the HR Attrition data.

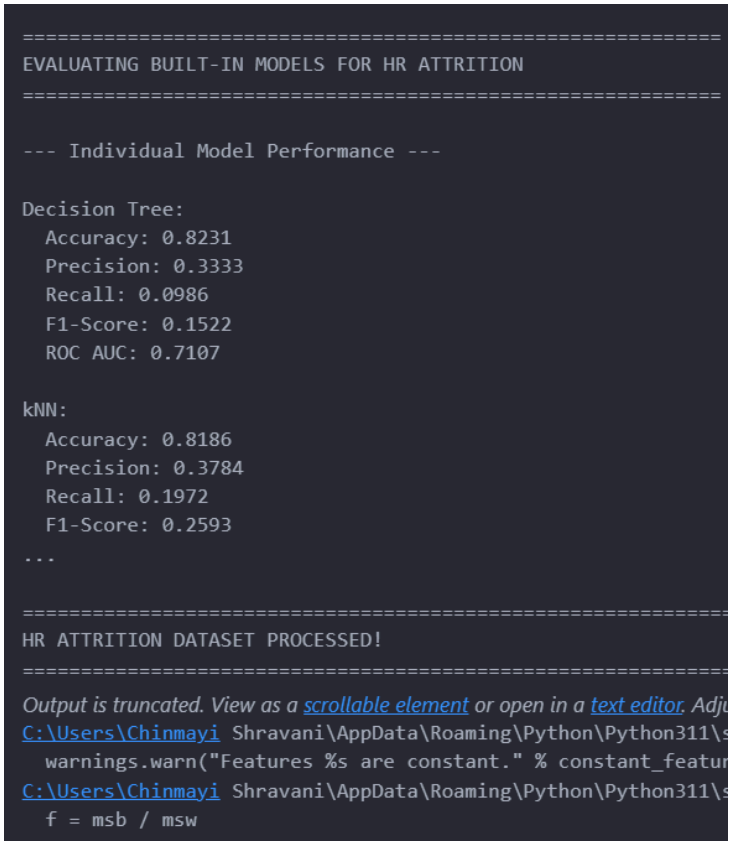
Output Screenshots

Manual implementation :



```
============================================================
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: 0.3784
  Recall: 0.1972
  F1-Score: 0.2593
  ROC AUC: 0.7236

Logistic Regression:
...
  F1-Score: 0.4717
  ROC AUC: 0.7912

--- Manual Voting Classifier ---
Output is truncated. View as a scrollable element or open in a text editor. Adjust
C:\Users\Chinmayi Shravani\AppData\Roaming\Python\Python311\sit
  warnings.warn("Features %s are constant." % constant_features
C:\Users\Chinmayi Shravani\AppData\Roaming\Python\Python311\sit
  f = msb / msw
Voting Classifier Performance:
  Accuracy: 0.8254, Precision: 0.4000
  Recall: 0.1690, F1: 0.2376, AUC: 0.7886
```

SciKit Learn Implementation :



```
============================================================
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: 0.3784
  Recall: 0.1972
  F1-Score: 0.2593
...

============================================================
HR ATTRITION DATASET PROCESSED!
============================================================
Output is truncated. View as a scrollable element or open in a text editor. Adju
C:\Users\Chinmayi Shravani\AppData\Roaming\Python\Python311\s
  warnings.warn("Features %s are constant." % constant_featur
C:\Users\Chinmayi Shravani\AppData\Roaming\Python\Python311\s
  f = msb / msw
```

Conclusion

Using both manual grid search and scikit-learn's automated GridSearchCV for hyperparameter tuning and model selection, this lab gave participants practical experience developing a strong machine learning pipeline. The findings demonstrated that both methods consistently identified the same best models with comparable performance on all metrics when provided with the same parameter grids and evaluation protocols. This illustrates both approaches' efficacy and repeatability.

The primary conclusion is that, although manual grid search enhances comprehension of the hyperparameter tuning procedure, for complex workflows, utilizing a library such as scikit-learn offers significant benefits in terms of efficiency, decreased coding effort, and error prevention. GridSearchCV and other automated tools facilitate experimentation, enforce best practices like pipeline encapsulation and cross-validation, and are capable of handling a wide variety of scenarios with ease.