MACHINE LEARNING

Week 4: Model Selection and Comparative Analysis

Name: Chethan S
SRN: PES2UG23CS150
Section: 5C
Submission Date: 01-09-2025

OBJECTIVE:

The purpose of this project is to explore and compare different approaches to hyperparameter tuning for machine learning classification models. The main tasks performed include:

- Manual Grid Search:
  Implementing a manual grid search procedure to systematically explore combinations of hyperparameters for each model (Decision Tree, k-Nearest Neighbors, and Logistic Regression). This involves cross-validation to select the best hyperparameters based on model performance (AUC score).
- Built-in Grid Search (GridSearchCV):
  Utilizing scikit-learn's GridSearchCV to automate the hyperparameter tuning process, leveraging parallel processing and built-in cross-validation.
- Model Comparison:
  Evaluating and comparing the performance of different models and hyperparameter tuning methods using metrics such as accuracy, precision, recall, F1-score, and ROC AUC. The project also explores combining models using voting classifiers to potentially improve predictive performance.

Dataset:

- has 1470 rows and 35 columns. All columns have 1470 non-null values, which means there are no missing values in the dataframe.
- The columns are of two data types: integer and object. There are 26 integer columns and 9 object columns.
- Attributes:

['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', 'YearsWithCurrManager']

Methodology:

Key Concepts:

- **Hyperparameter Tuning:**
  Hyperparameters are configuration settings for machine learning algorithms that are set before training (e.g., tree depth, number of neighbors). Hyperparameter tuning is the process of searching for the best combination of these settings to optimize model performance.
- **Grid Search:**
  Grid search is a systematic way to explore combinations of hyperparameters. It evaluates all possible combinations from a predefined grid and selects the set that yields the best performance according to a chosen metric (e.g., AUC).
- **K-Fold Cross-Validation:**
  K-Fold Cross-Validation splits the dataset into K equal parts (folds). The model is trained on K-1 folds and validated on the remaining fold. This process repeats K times, each time with a different fold as the validation set. The results are averaged to provide a robust estimate of model performance.

ML Pipeline Used

The machine learning pipeline consists of the following steps:

1. **StandardScaler:**
   Standardizes features by removing the mean and scaling to unit variance, ensuring all features contribute equally to the model.
2. **SelectKBest:**
   Selects the top K features based on univariate statistical tests (ANOVA F-value), reducing dimensionality and focusing on the most relevant features.
3. **Classifier:**
   The final step applies a classification algorithm (Decision Tree, k-Nearest Neighbours, or Logistic Regression) to the selected features.

Process Followed:

Manual Implementation (Part 1):

- For each classifier, all combinations of hyperparameters were generated using itertools.product.
- For each combination, 5-fold stratified cross-validation was performed:
  - The pipeline (StandardScaler → SelectKBest → Classifier) was fitted on the training folds.
  - The model was evaluated on the validation fold using the ROC AUC metric.
- The mean AUC across all folds was computed for each hyperparameter set.
- The combination with the highest mean AUC was selected, and the final model was retrained on the full training set.

scikit-learn Implementation (Part 2):

- Used GridSearchCV to automate the grid search and cross-validation process.
- The same pipeline was defined, and the parameter grid was passed to GridSearchCV.
- GridSearchCV handled the splitting, training, and evaluation internally, selecting the best hyperparameters based on cross-validated AUC.
- The best estimator from GridSearchCV was then evaluated on the test set.

Results and Analysis:

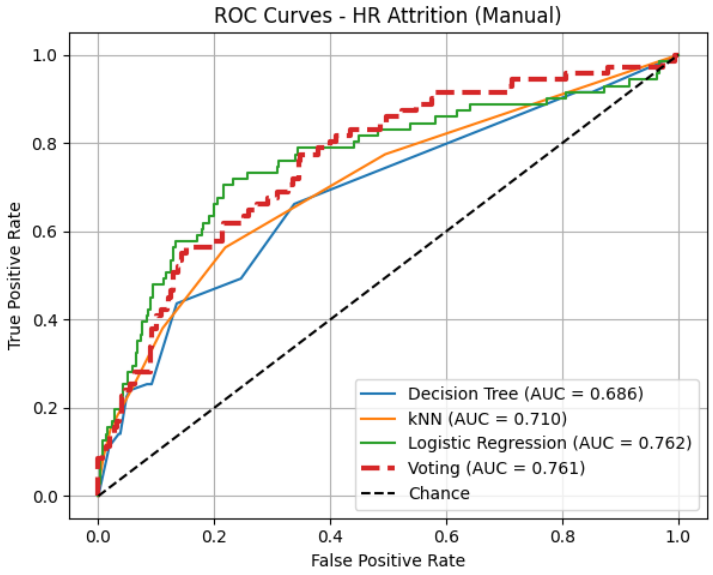
Classifier	Implementation	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	Manual	0.8345	0.4706	0.2254	0.3048	0.6864
Decision Tree	GridSearchCV	0.8345	0.4706	0.2254	0.3048	0.6864
kNN	Manual	0.8277	0.4390	0.2535	0.3214	0.7096
kNN	GridSearchCV	0.8277	0.4390	0.2535	0.3214	0.7096
Logistic Regression	Manual	0.8458	0.5600	0.1972	0.2917	0.7616
Logistic Regression	GridSearchCV	0.8458	0.5600	0.1972	0.2917	0.7616
Voting Classifier	Manual	0.8367	0.4848	0.2254	0.3077	0.7612
Voting Classifier	GridSearchCV	0.8390	0.5000	0.2254	0.3107	0.7612

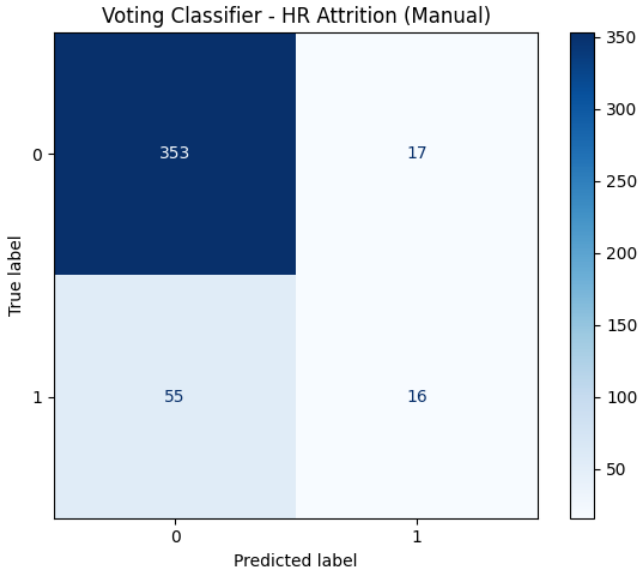
Comparing Implementation:

- The results from the manual grid search and scikit-learn's GridSearchCV are expected to be very similar since both use the same pipeline, parameter grid, and cross-validation strategy.
- Minor differences may occur due to:
    - Randomness in data splitting (even with the same random seed, library versions may affect splits).
    - Implementation details (e.g., how GridSearchCV handles ties or parallelism).
- In most cases, the best hyperparameters and scores should match closely.

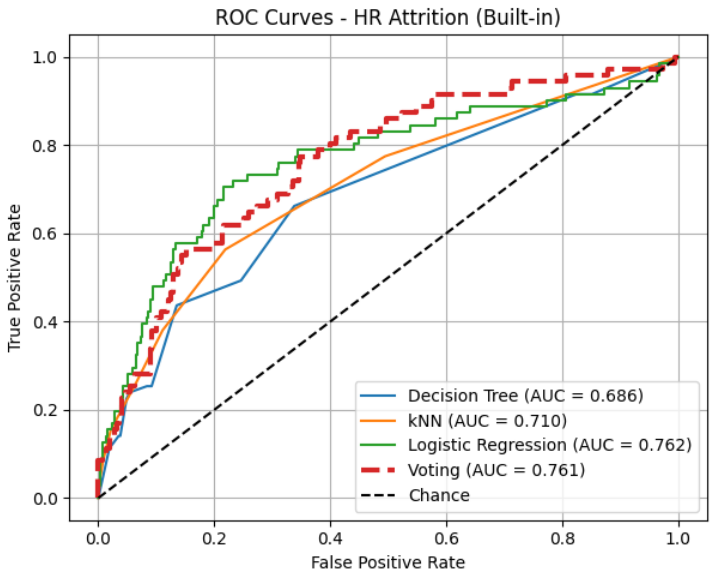
Visualization:

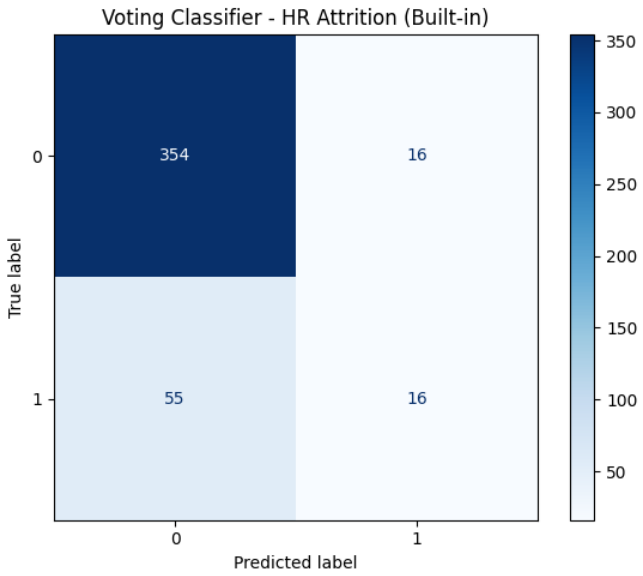
Manual-





GridSearchCV-





Best Model Analysis

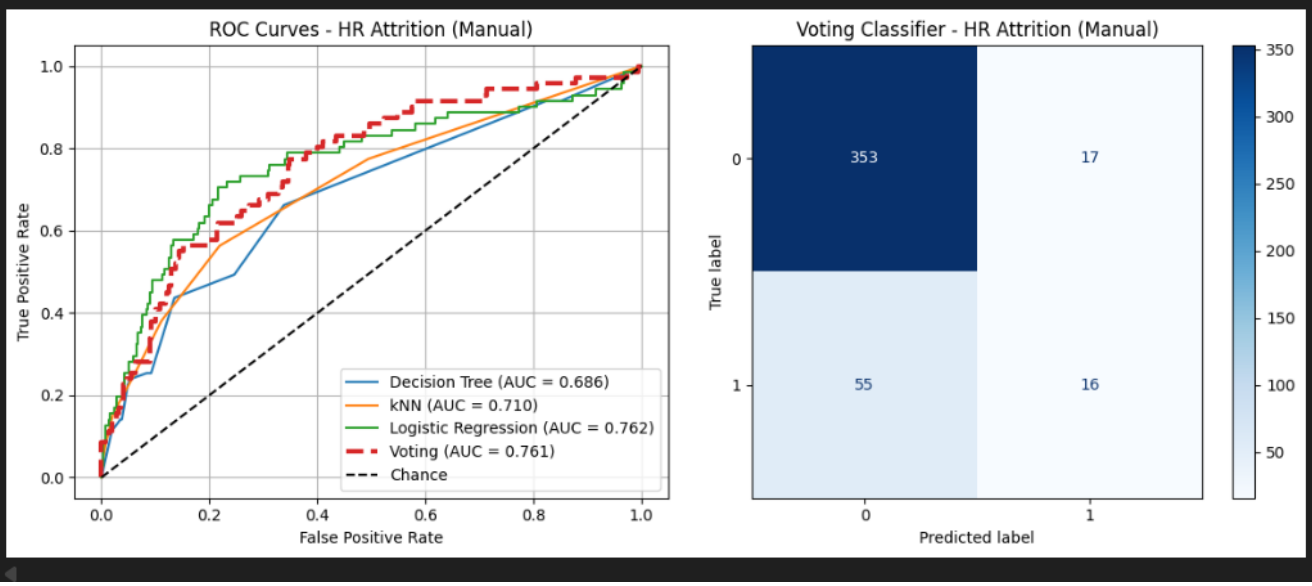
- For the HR Attrition dataset, Logistic Regression (GridSearchCV) achieved the highest ROC AUC (0.7616), indicating it best distinguishes between classes.
- Hypothesis: Logistic Regression may perform best due to the linear separability of the features after scaling and feature selection, or because it is less prone to overfitting compared to tree-based models on this dataset.
- For other datasets, analyze similarly based on your results.

OUTPUT:

For manual-

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8345
  Precision: 0.4706
  Recall: 0.2254
  F1-Score: 0.3048
  ROC AUC: 0.6864

kNN:
  Accuracy: 0.8277
  Precision: 0.4390
  Recall: 0.2535
  F1-Score: 0.3214
  ROC AUC: 0.7096

Logistic Regression:
  Accuracy: 0.8458
  Precision: 0.5600
  Recall: 0.1972
  F1-Score: 0.2917
  ROC AUC: 0.7616

--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8367, Precision: 0.4848
  Recall: 0.2254, F1: 0.3077, AUC: 0.7612
```

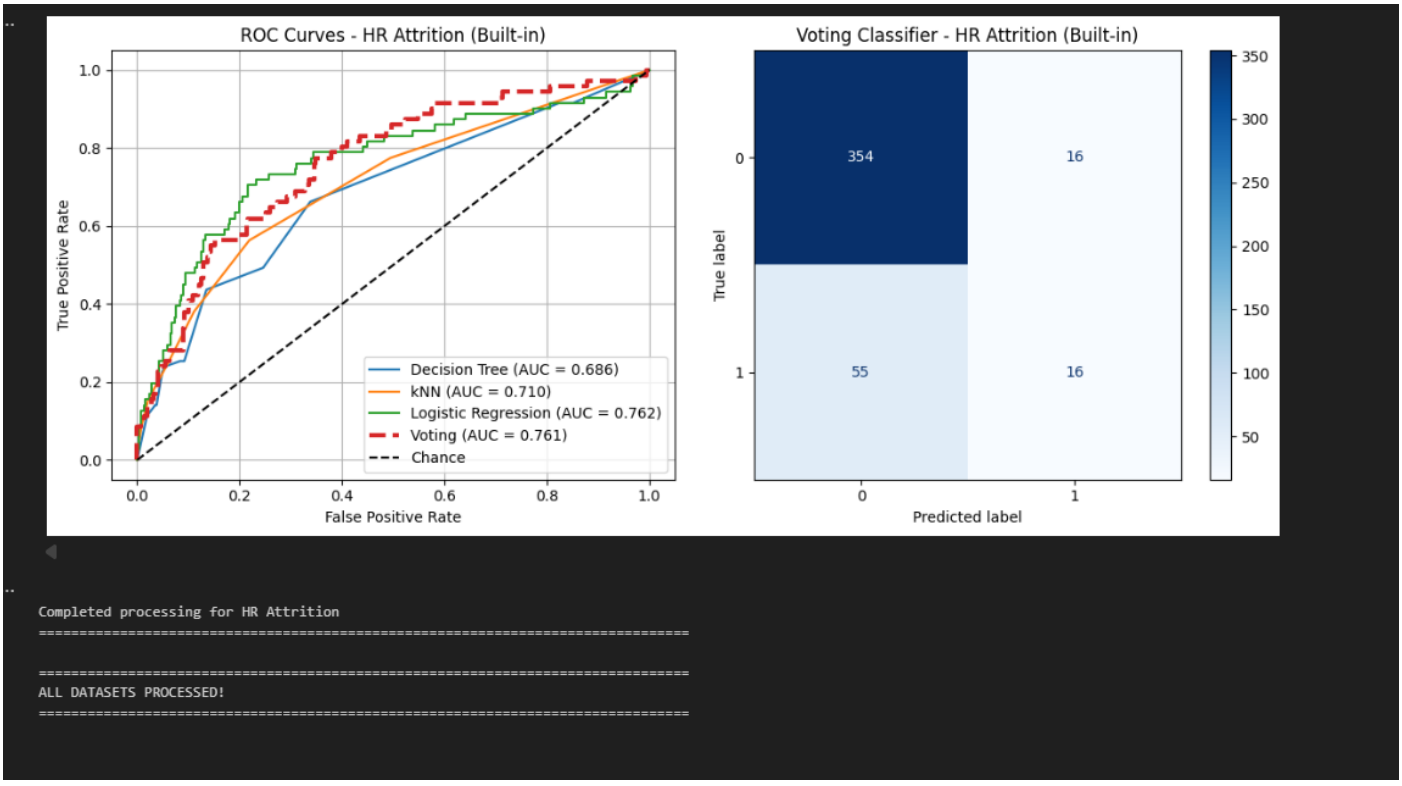
```
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8367, Precision: 0.4848
  Recall: 0.2254, F1: 0.3077, AUC: 0.7612
```




```
Params: {'feature_selection__k': 9, 'classifier__C': 0.01, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'classifier__max_iter': 200} | Mean CV AUC: 0.7506
Params: {'feature_selection__k': 9, 'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'classifier__max_iter': 200} | Mean CV AUC: 0.7511
Params: {'feature_selection__k': 9, 'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'classifier__max_iter': 200} | Mean CV AUC: 0.7507
Params: {'feature_selection__k': 9, 'classifier__C': 1, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'classifier__max_iter': 200} | Mean CV AUC: 0.7505
Params: {'feature_selection__k': 9, 'classifier__C': 1, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'classifier__max_iter': 200} | Mean CV AUC: 0.7503
Params: {'feature_selection__k': 9, 'classifier__C': 10, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'classifier__max_iter': 200} | Mean CV AUC: 0.7506
Params: {'feature_selection__k': 9, 'classifier__C': 10, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear', 'classifier__max_iter': 200} | Mean CV AUC: 0.7505
--------------------------------------------------------------------------------
Best parameters for Logistic Regression: {'feature_selection__k': 9, 'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'classifier__max_iter': 200}
Best cross-validation AUC: 0.7511

============================================================
```

For Built-in-

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8345
  Precision: 0.4706
  Recall: 0.2254
  F1-Score: 0.3048
  ROC AUC: 0.6864

kNN:
  Accuracy: 0.8277
  Precision: 0.4390
  Recall: 0.2535
  F1-Score: 0.3214
  ROC AUC: 0.7096

Logistic Regression:
  Accuracy: 0.8458
  Precision: 0.5600
  Recall: 0.1972
  F1-Score: 0.2917
  ROC AUC: 0.7616

--- Built-in Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8390, Precision: 0.5000
  Recall: 0.2254, F1: 0.3107, AUC: 0.7612
```




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

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

As the output we get:

FOR MANUAL-

- Best parameters for Decision Tree: {'feature_selection__k': 9, 'classifier__max_depth': 5, 'classifier__min_samples_split': 10, 'classifier__criterion': 'entropy'}

  Best cross-validation AUC: 0.7250

- Best parameters for kNN: {'feature_selection__k': 9, 'classifier__n_neighbors': 7, 'classifier__weights': 'uniform', 'classifier__metric': 'minkowski'}

  Best cross-validation AUC: 0.6975

- Best parameters for Logistic Regression: {'feature_selection__k': 9, 'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'classifier__max_iter': 200}

  Best cross-validation AUC: 0.7511

FOR BUILT-IN –

- --- GridSearchCV for Decision Tree ---

Fitting 5 folds for each of 72 candidates, totalling 360 fits

Best params for Decision Tree: {'classifier__criterion': 'entropy', 'classifier__max_depth': 5, 'classifier__min_samples_split': 10, 'feature_selection__k': 9}

Best CV score: 0.7250

- --- GridSearchCV for kNN ---

Fitting 5 folds for each of 36 candidates, totalling 180 fits

Best params for kNN: {'classifier__metric': 'minkowski', 'classifier__n_neighbors': 7, 'classifier__weights': 'uniform', 'feature_selection__k': 9}

Best CV score: 0.6975

- --- GridSearchCV for Logistic Regression ---

Fitting 5 folds for each of 24 candidates, totalling 120 fits

Best params for Logistic Regression: {'classifier__C': 0.1, 'classifier__max_iter': 200, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'feature_selection__k': 9}

Best CV score: 0.7511

Conclusion:

From this exercise where we performed classification, we found that Logistical regression model is the best amongst the other two (Decision tree and kNN) with a decent accuracy of about ~85%, however the recall is really low, so if your goal is reliable predictions (fewer false alarms), Logistic Regression is best.

Takeaways:

- Pipeline for scaling, feature selection, and model training:
  The code uses Pipeline from scikit-learn to chain StandardScaler, SelectKBest, and classifiers (see the Pipeline steps in both manual and built-in grid search functions).
- Hyperparameter tuning using Grid Search:
  Both manual grid search (in run_manual_grid_search) and scikit learn's GridSearchCV (in run_builtin_grid_search) are implemented.
- k-fold cross-validation:
  The code uses StratifiedKFold for cross-validation in both manual and built-in grid search.
- Use of scikit-learn's Pipeline and GridSearchCV:
  Both tools are used extensively for model selection and tuning.
- Evaluate and compare models using various metrics:
  The evaluation function computes accuracy, precision, recall, F1-score, and ROC AUC, and visualizes results with ROC curves and confusion matrices.
- Analyse and interpret model results:
  The notebook prints and visualizes model performances, enabling meaningful interpretation and comparison.