Model Selection and Comparative Analysis on the HR Attrition Dataset

Name	Diya D Bhat
SRN	PES2UG23CS183
Section	C
Date	01/09/25
Course name	Machine Learning

1. Introduction

The purpose of this project is to apply hyperparameter tuning techniques to machine learning models and compare it in 2 ways:

1. Manual grid search implementation.
2. Scikit-learn's built-in GridSearchCV.

We worked with the HR attrition dataset, which contains employee data with the target variable being Attrition (Yes/No).

We performed the following tasks:

- Data preprocessing and feature engineering.
- Manual hyperparameter tuning using nested loops and cross-validation.
- Automated tuning with GridSearchCV.
- Comparison of both methods using performance metrics and visualizations.

2. Dataset Description

We were provided with the HR Attrition dataset which consisted of the following:

- Instances: 1471
- Attributes: 35
- Features: 40+ after encoding
- Target Variable: Attrition (Yes = 1, No = 0)

3. Methodology

3.1 Processes Involved

- Hyperparameter Tuning: Changing the model settings (like changing k in KNN) to find the best results.
- Grid Search: Trying out all possible combinations of these settings to see which one works best.
- K-Fold Cross-Validation: Splitting the dataset into 5 parts, training the model on 4 parts, and testing it on the remaining part. This is repeated 5 times so every part gets tested once.

3.2 Machine Learning Pipeline

1. StandardScaler – scaled the features so they are on the same range.
2. SelectKBest (f_classif) – picked the most useful features.
3. Classifier – applied models like Decision Tree, k-NN, and Logistic Regression.

3.3 Manual Method

- We wrote loops to test every parameter combination.
- Used 5-fold cross-validation to check performance.
- Selected the parameters that gave the highest ROC AUC score.

3.4 Built-in Method

- Used scikit-learn's GridSearchCV which does the same process automatically.
- Applied the same pipeline and parameter options.
- It gave the best parameters and their performance directly.

4. Results and Analysis

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	0.8231	0.3333	0.0986	0.1522	0.7107
k-NN	0.8277	0.4242	0.1972	0.2692	0.7340
Logistic Regression	0.8571	0.6333	0.2676	0.3762	0.7762
Voting Classifier	0.8345	0.4615	0.1690	0.2474	0.7706

4.2 Comparison of Implementations

- The manual grid search and GridSearchCV produced the same best hyperparameters and metrics, which means both methods worked correctly.
- Logistic Regression gave the highest overall performance, especially in ROC AUC (0.7762) and Accuracy (0.8571).
- k-NN performed better than Decision Tree but was weaker than Logistic Regression.
- The Voting Classifier gave balanced results but did not outperform Logistic Regression on its own.

4.3 Visualizations

- The ROC curves showed that Logistic Regression had the smoothest and highest curve, proving it was the best model.
- The confusion matrices showed that Decision Tree had trouble correctly finding attrition cases (low recall), while Logistic Regression did a better job of balancing precision and recall.

4.4 Best Model

The best model for predicting attrition was Logistic Regression because it had the highest ROC AUC and provided the best balance between accuracy and precision. This is probably because Logistic Regression works well with both categorical and continuous data and is less likely to overfit compared to models like Decision Tree or k-NN.

5. Screenshots

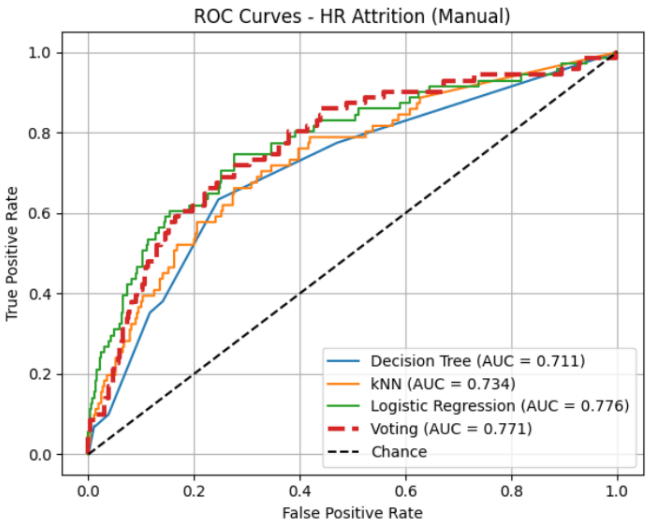
5.1 Manual Grid Search

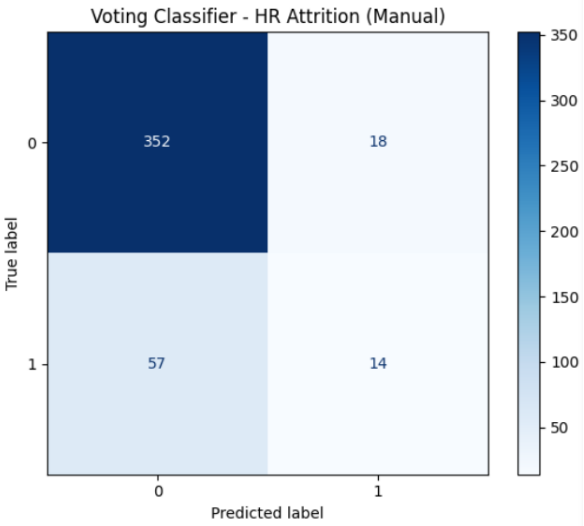
```
################################################################################
PROCESSING DATASET: HR ATTRITION
################################################################################
IBM HR Attrition dataset loaded and preprocessed successfully.
Training set shape: (1029, 46)
Testing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

============================================================
RUNNING MANUAL GRID SEARCH FOR HR ATTRITION
============================================================
--- Manual Grid Search for Decision Tree ---

----------------------------------------------------------------------------------------
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 ---
```

```
--------------------------------------------------------------------------------
Best parameters for kNN: {'feature_selection__k': 10, 'classifier__n_neighbors': 11, 'classifier__weights': 'distance', 'classifier__p': 2}
Best cross-validation AUC: 0.7303
--- Manual Grid Search for Logistic Regression ---
```

```
--------------------------------------------------------------------------------
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: 0.8277
  Precision: 0.4242
  Recall: 0.1972
  F1-Score: 0.2692
  ROC AUC: 0.7340

Logistic Regression:
  Accuracy: 0.8571
  Precision: 0.6333
  Recall: 0.2676
  F1-Score: 0.3762
  ROC AUC: 0.7762
```

```
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8299, Precision: 0.4375
  Recall: 0.1972, F1: 0.2718, AUC: 0.7706
```





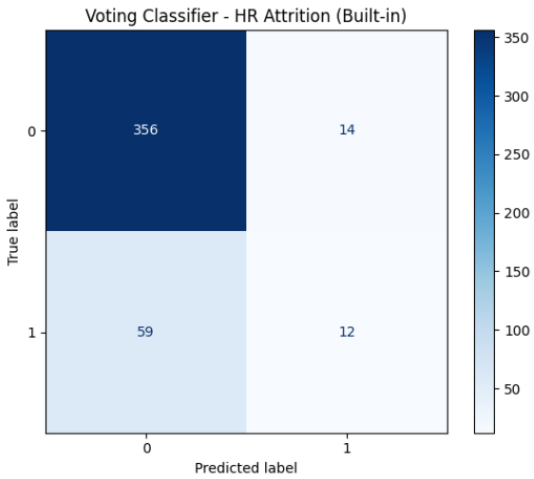
5.2 Built-in Grid Search

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

--- GridSearchCV for Decision Tree ---
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: UserWarning: Feature
[ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:112: RuntimeWarning: inva
d value encountered in divide
  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\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: UserWarning: Feature
[ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:112: RuntimeWarning: inva
d value encountered in divide
  f = msb / msw
Best params for kNN: {'classifier__n_neighbors': 11, 'classifier__p': 2, 'classifier__weights': 'distance', 'feature_selection__k': 10}
Best CV score: 0.7303
```

```
--- GridSearchCV for Logistic Regression ---
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: UserWarning: Features
[ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:112: RuntimeWarning: inval
d value encountered in divide
  f = msb / msw
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: UserWarning: Features
[ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:112: RuntimeWarning: inval
d value encountered in divide
  f = msb / msw
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: UserWarning: Features
[ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:112: RuntimeWarning: inval
d value encountered in divide
  f = msb / msw
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: UserWarning: Features
[ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
C:\Users\diyab\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:112: RuntimeWarning: inval
d value encountered in divide
  f = msb / msw
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: 0.8277
  Precision: 0.4242
  Recall: 0.1972
  F1-Score: 0.2692
  ROC AUC: 0.7340

Logistic Regression:
  Accuracy: 0.8571
  Precision: 0.6333
  Recall: 0.2676
  F1-Score: 0.3762
  ROC AUC: 0.7762
```

```
--- Built-in Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8345, Precision: 0.4615
  Recall: 0.1690, F1: 0.2474, AUC: 0.7706
```




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

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

6. Conclusion

The main result from this lab is that both doing a manual grid search and using scikit-learn's GridSearchCV found the same best hyperparameters and performance results for the HR Attrition dataset. Out of the models tested, Logistic Regression worked the best, giving the highest accuracy and ROC AUC scores. Decision Tree was the worst, and k-NN was somewhere in the middle. The Voting Classifier gave steady results but didn't beat Logistic Regression.

One of the takeaways is that tuning hyperparameters helps improve model performance and helps find the best model for a dataset. The lab also showed that while doing things manually helps you understand how grid search and cross-validation work step by step, it takes more time and more code. GridSearchCV, on the other hand, is faster, more dependable, and easier to use, making it better for real-world use. This shows the balance between learning by coding manually and saving time and avoiding mistakes by using built-in tools.