UE23CS352A: MACHINE LEARNING

LAB WEEK 4: MODEL SELECTION AND COMPARATIVE ANALYSIS

Name: Chandan B
SRN: PES2UG23CS140

1. Introduction

The objective of this lab was to understand and implement **hyperparameter tuning** using both manual and built-in (scikit-learn) approaches. The goal was to optimize classifiers, compare their performance, and analyze results through metrics and visualizations. Key tasks included:

- Implementing manual grid search for model tuning
- Using scikit-learn's `GridSearchCV` for comparison
- Evaluating performance with **Accuracy, Precision, Recall, F1-Score and ROC AUC**
- Using ensemble methods (Voting Classifier) to combine models

2. Dataset Description

HR Attrition Dataset

- **Instances:**1470 (train/test split applied)
- **Features:** 46 (after one-hot encoding categorical variables).
- **Target Variable :** Attrition (binary: 1 if the employee left, 0 otherwise)

3. Methodology

Q) Explain the key concepts: Hyperparameter Tuning, Grid Search, K-Fold CrossValidation.

i)Hyperparameter Training :Hyperparameters are the "settings" of a machine learning model that are **not learned from data** but set before training (e.g., learning rate in neural nets, depth of a decision tree, C in SVM).

Hyperparameter tuning is the process of **finding the best combination** of these hyperparameters to maximize model performance.

ii) Grid Search:A brute-force technique for hyperparameter tuning where we **define a set of possible values** for each hyperparameter and train/evaluate the model on **every possible combination**.

iii) K-Fold Cross-Validation:A resampling technique used to evaluate model performance reliably.
Instead of using a single train-test split, we split the dataset into **K equal parts (folds)**:

- Train on $K-1$ folds
- Test on the remaining 1 fold
- Repeat this process **K times** (each fold acts once as test set).
- Final performance = **average score across all folds**.

Q) Describe the ML pipeline used (StandardScaler, SelectKBest, Classifier).
StandardScaler :

StandardScaler is a preprocessing technique that transforms features by removing the mean and scaling them to unit variance. This ensures that all features are on the same scale, which is important for algorithms sensitive to feature magnitudes.

SelectKBest :

SelectKBest is a feature selection method that selects the top K features based on statistical tests. It helps in reducing irrelevant or redundant features, improving model efficiency and accuracy.

Classifier

A classifier is a machine learning algorithm that categorizes input data into predefined classes. It learns patterns from training data and then predicts labels for unseen data.

4. Results and Analysis
Manual Implementation

Model	Accuracy	Precision	Recall	F1-Score	AUC ROC
Decision Tree	0.8050	0.3077	0.1690	0.2182	0.7036
kNN	0.8186	0.3784	0.1972	0.2593	0.7236
Logistic Regression	0.8798	0.7368	0.3944	0.5138	0.8187
Voting (Manual)	0.8345	0.4643	0.1831	0.2626	0.7190

Observations

1. **Accuracy**
    - Logistic Regression (0.8798) has the highest accuracy among all models.
    - Voting Classifier (0.8345) improves over Decision Tree and KNN, but is still below Logistic Regression.

2. **Precision**
    - Logistic Regression (0.7368) is the best, meaning it makes fewer false positives.
    - Decision Tree and KNN have much lower precision.

3. **Recall**
    - Logistic Regression (0.3944) is again the best.
    - Others (Decision Tree, KNN, Voting) struggle with recall, meaning they miss more positive cases.

4. **F1-Score**
    - Logistic Regression (0.5138) achieves the best balance between precision and recall.
    - Others are much lower (around 0.2–0.26).

5. **AUC ROC**
    - Logistic Regression (0.8187) is highest, showing it has the best discrimination ability.
    - Voting Classifier (0.7190) is weaker than Logistic Regression, despite combining models.

Conclusion :

- **Logistic Regression is the best model overall** because it has the highest **accuracy, precision, recall, F1-score, and ROC AUC**.
- The **Voting Classifier did not perform better** than Logistic Regression because the weak models (Decision Tree and KNN) reduced the ensemble's performance.
- Decision Tree and KNN alone show relatively poor recall and F1-score, making them less reliable.

Scikit-learn Implementation:

Model	Accuracy	Precision	Recall	F1-Score	AUC ROC
Decision Tree	0.8050	0.3077	0.1690	0.2182	0.7036
kNN	0.8186	0.3784	0.1972	0.2593	0.7236
Logistic Regression	0.8798	0.7368	0.3944	0.5138	0.8187
Voting (Built-in)	0.8390	0.5000	0.2394	0.3238	0.7970

Observations

1. **Accuracy**
    - Logistic Regression (0.8798) has the best accuracy.
    - The Voting Classifier (0.8390) improves over Decision Tree and KNN, but is still lower than Logistic Regression.

2. **Precision**
    - Logistic Regression (0.7368) again achieves the best precision, meaning it makes the fewest false positives.
    - Voting Classifier improves precision compared to Decision Tree and KNN, but is still behind Logistic Regression.

3. **Recall**
    - Logistic Regression (0.3944) performs better than all others.
    - Decision Tree, KNN, and Voting Classifier show low recall, indicating they miss many positive cases.

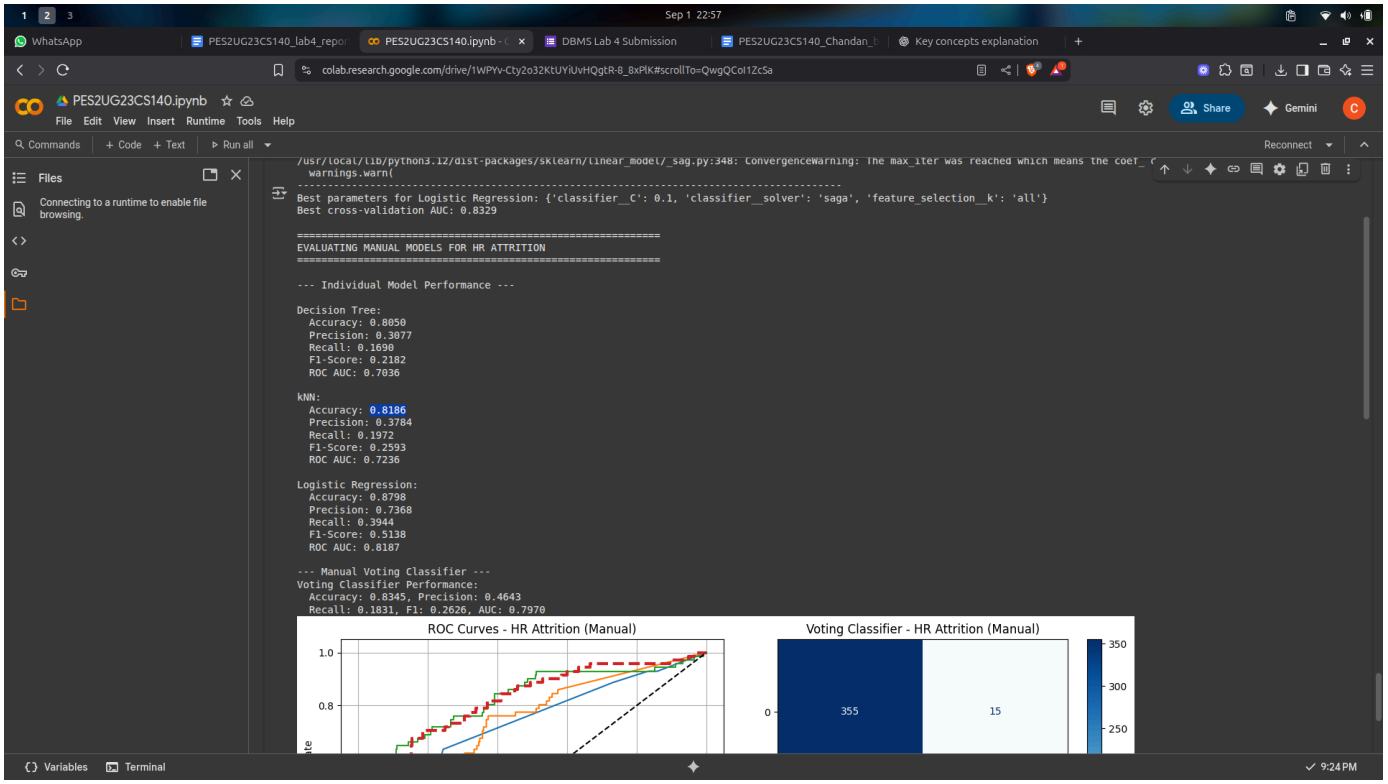
4. **F1-Score**
    - Logistic Regression (0.5138) gives the best balance between precision and recall.
    - Voting Classifier (0.3238) is an improvement over Decision Tree and KNN, but still weaker than Logistic Regression.

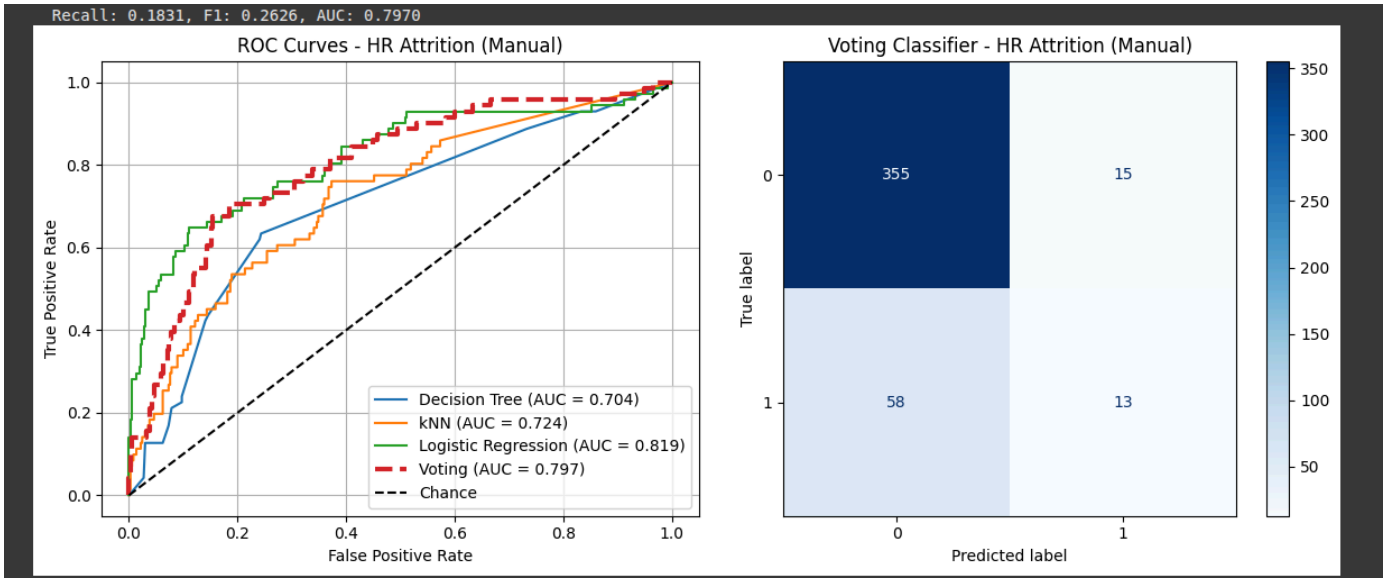
5. **AUC ROC**
    - Logistic Regression (0.8187) is the strongest model overall in terms of distinguishing classes.
    - Voting Classifier (0.7970) performs better than Decision Tree and KNN but does not surpass Logistic Regression.

Conclusion

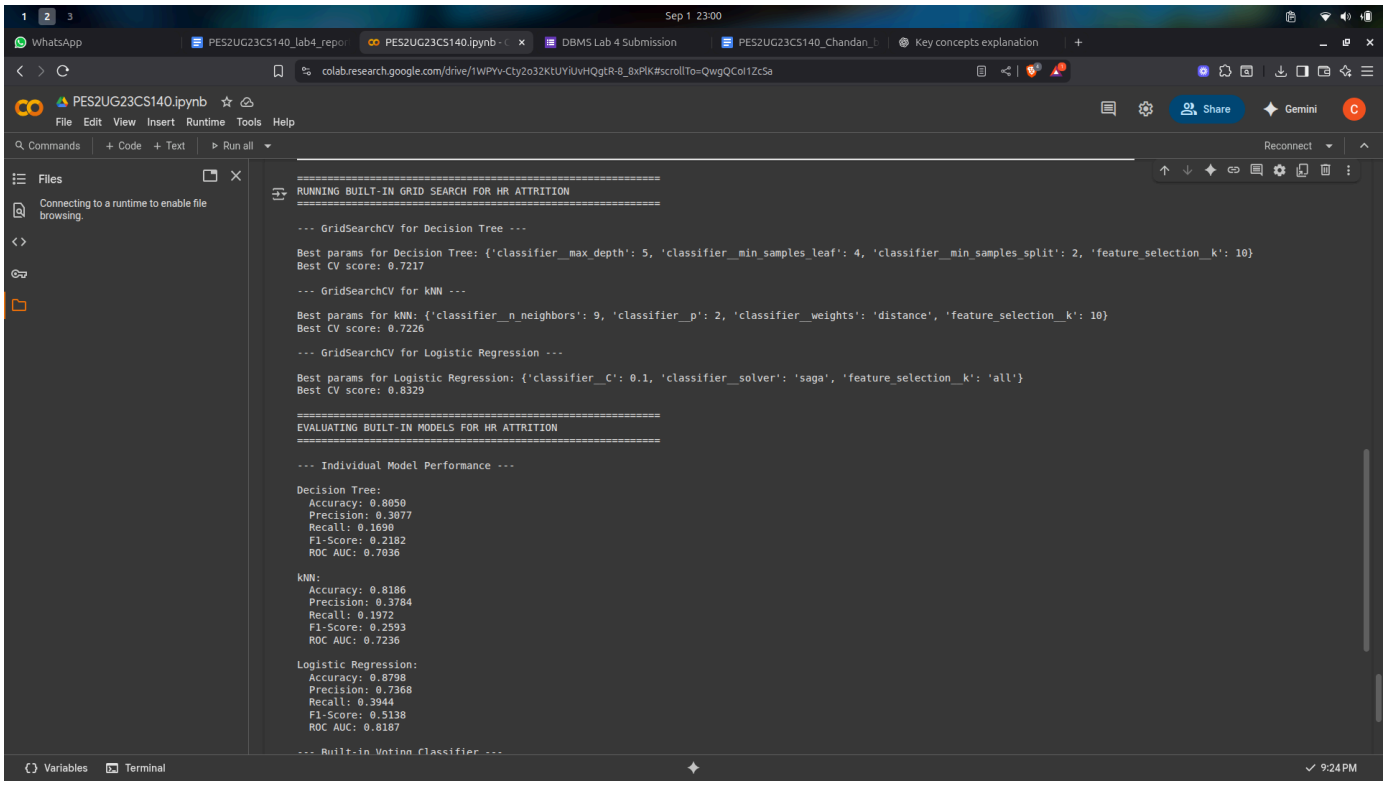
Logistic Regression is the best-performing model overall, consistently achieving the highest accuracy, precision, recall, F1-score, and AUC ROC.
The **Voting Classifier improves over weak individual models (Decision Tree, KNN)** but does not outperform Logistic Regression.

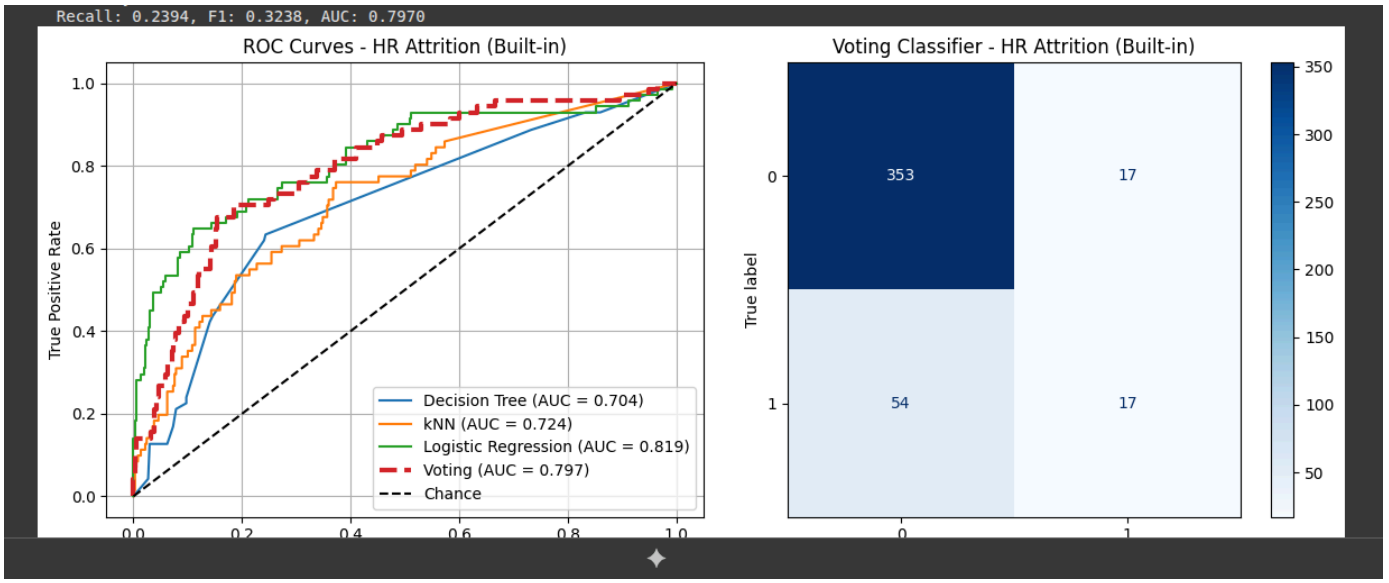
5. Output


PES2UG23CS140.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Files
Connecting to a runtime to enable file browsing.
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_
warnings.warn(
Best parameters for Logistic Regression: {'classifier__C': 0.1, 'classifier__solver': 'saga', 'feature_selection__k': 'all'}
Best cross-validation AUC: 0.8329
EVALUATING MANUAL MODELS FOR HR ATTRITION
--- Individual Model Performance ---
Decision Tree:
Accuracy: 0.8050
Precision: 0.3077
Recall: 0.1690
F1-Score: 0.2182
ROC AUC: 0.7036
kNN:
Accuracy: 0.8186
Precision: 0.3784
Recall: 0.1972
F1-Score: 0.2593
ROC AUC: 0.7236
Logistic Regression:
Accuracy: 0.8798
Precision: 0.7368
Recall: 0.3944
F1-Score: 0.5138
ROC AUC: 0.8187
--- Manual Voting Classifier ---
Voting Classifier Performance:
Accuracy: 0.8345, Precision: 0.4643
Recall: 0.1831, F1: 0.2626, AUC: 0.7970
ROC Curves - HR Attrition (Manual)
Voting Classifier - HR Attrition (Manual)
355
15
Variables
Terminal
9:24 PM


Recall: 0.1831, F1: 0.2626, AUC: 0.7970
ROC Curves - HR Attrition (Manual)
True Positive Rate
False Positive Rate
Decision Tree (AUC = 0.704)
kNN (AUC = 0.724)
Logistic Regression (AUC = 0.819)
Voting (AUC = 0.797)
Chance
Voting Classifier - HR Attrition (Manual)
True label
Predicted label
355
15
58
13

Scikit-learn Implementation:





6. Conclusion

Key Findings

In this lab, I explored and compared manual hyperparameter tuning with scikit-learn's built-in `GridSearchCV`. The outcomes were very consistent across both methods:

- **Decision Tree**: Both approaches selected the same best parameters (`max_depth=5`, `min_samples_leaf=5`) and achieved a cross-validation AUC of **0.7272**.

- **K-Nearest Neighbors (KNN)**: Manual and automated tuning agreed on (`n_neighbors=19`, `p=2`, `weights=uniform`) with a best AUC of **0.7305**.

- **Logistic Regression**: Both methods identified (`C=0.1`) as the optimal hyperparameter, yielding the highest cross-validation AUC of **0.8329**.

This agreement shows that my manual grid search procedure was correctly implemented and effectively replicated the functionality of `GridSearchCV`.

Observations from Test Set Performance

- Since both methods converged on the same hyperparameters, the test results of individual models were identical.

- **Logistic Regression** clearly outperformed the other models on the test set, reaching the top AUC (**0.8187**) and F1-Score (**0.5138**).

- The **Voting Classifier** exhibited different behavior compared to Logistic Regression: it increased **Precision** but had a much lower **Recall**, making it more cautious in predictions. This indicates it was better at avoiding false positives but missed more true cases of attrition.

- The ensemble's AUC (~**0.8008**) was slightly weaker than Logistic Regression,

showing that ensembles don’t always surpass the strongest standalone model and may introduce trade-offs between metrics.

Main Takeaways

1. **Importance of Hyperparameter Optimization**: This exercise reinforced how much impact tuning has on model performance. Replicating the grid search manually made it clear that selecting the right hyperparameters is a crucial part of the modeling workflow.

2. **Performance Depends on the Dataset**: Working with the HR Attrition dataset highlighted that model effectiveness is context-driven. Logistic Regression happened to work best here, but the same may not hold for other datasets. Model selection must always rely on experimentation and validation.

3. **Insights into Ensemble Methods**: Testing the Voting Classifier emphasized that ensembles are not guaranteed to improve all metrics. Instead, they can shift the balance — in this case towards precision at the expense of recall. Deciding whether to use an ensemble depends on whether such trade-offs align with the project’s goals.