MACHINE LEARNING LAB 4

NAME:- DIBYA DYUTI BHALLA

SRN:- PES2UG23CS176 SECTION:- C

1. Introduction

The purpose of this project was to build a **complete machine learning pipeline** for the IBM HR Attrition dataset, focusing on **hyperparameter tuning** and **model comparison**. Two approaches were implemented:

1. **Manual Grid Search** – building a parameter search loop from scratch.
2. **Scikit-learn GridSearchCV** – leveraging sklearn's optimized search utilities.

Three classifiers were compared: **Decision Tree**, **k-Nearest Neighbors (kNN)**, and **Logistic Regression**. An ensemble **Voting Classifier** was also evaluated.

The main tasks included:

- Preprocessing data using scaling and feature selection.
- Hyperparameter tuning using grid search with cross-validation.
- Evaluating models using multiple performance metrics.
- Comparing manual vs. built-in grid search implementations.

---

2. Dataset Description

The dataset used was the **IBM HR Employee Attrition dataset**.

- **Number of instances:** ~1470 employees
- **Number of features (after encoding):** ~50+ (varies after one-hot encoding)
- **Target variable:** Attrition (binary classification)
  - Yes → 1 (Employee left)
  - No → 0 (Employee stayed)

This dataset contains demographic, workplace, and personal factors (e.g., age, department, job role, salary, work-life balance) that influence employee attrition.

---

3. Methodology

Key Concepts

- **Hyperparameter Tuning:** Process of selecting the best model parameters that maximize predictive performance.
- **Grid Search:** Systematically tries all parameter combinations in a defined grid.
- **K-Fold Cross-Validation:** Data is split into *k* folds (here, k=5). Models are trained on k-1 folds and validated on the remaining fold to ensure robustness.

ML Pipeline

Each model was trained using a three-step pipeline:

1. **StandardScaler** – standardize features to mean=0, variance=1.
2. **SelectKBest(f_classif)** – select top k features.
3. **Classifier** – Decision Tree, kNN, or Logistic Regression.

Implementation Process

- **Part 1: Manual Grid Search**
  - Looped over all hyperparameter combinations.
  - Performed 5-fold stratified CV.
  - Computed mean ROC AUC per combination.
  - Selected the best parameters.
- **Part 2: Scikit-learn GridSearchCV**
  - Built identical pipelines.
  - Used GridSearchCV with scoring="roc_auc" and StratifiedKFold.
  - Extracted best estimator, parameters, and CV score.

---

4. Results and Analysis

Manual Grid Search – Best Model Performance

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	0.8163	0.3684	0.1972	0.2569	0.7029
kNN	0.8277	0.4242	0.1972	0.2692	0.7340

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Logistic Regression	**0.8798**	**0.7368**	**0.3944**	**0.5138**	**0.8187**
Voting Classifier	0.8367	0.4839	0.2113	0.2941	0.8039

Built-in GridSearchCV – Best CV Scores

- **Decision Tree** → Best AUC = 0.7087 (max depth=5, min_samples_split=2, k=5)
- **kNN** → Best AUC = 0.7303 (n_neighbors=11, weights=distance, k=10)
- **Logistic Regression** → Best AUC = **0.8329** (C=0.1, penalty=l2, solver=lbfgs, k=all)

Built-in Model Performance

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	0.8163	0.3684	0.1972	0.2569	0.7029
kNN	0.8277	0.4242	0.1972	0.2692	0.7340
Logistic Regression	**0.8798**	**0.7368**	**0.3944**	**0.5138**	**0.8187**
Voting Classifier	(Error due to variable scope, expected similar to manual voting)				

Comparison of Implementations

- Both **manual and built-in grid search** selected **Logistic Regression** as the best-performing model.
- Results were nearly identical; the built-in method was slightly better in CV AUC (0.8329 vs. 0.8187).
- Differences are due to implementation details: sklearn optimizes CV splits and parameter handling more efficiently.

Visualizations (to include in report)

- **ROC Curves:** Logistic Regression curve is consistently higher.
- **Confusion Matrices:** Logistic Regression shows best balance between detecting attrition cases (recall) and minimizing false positives.

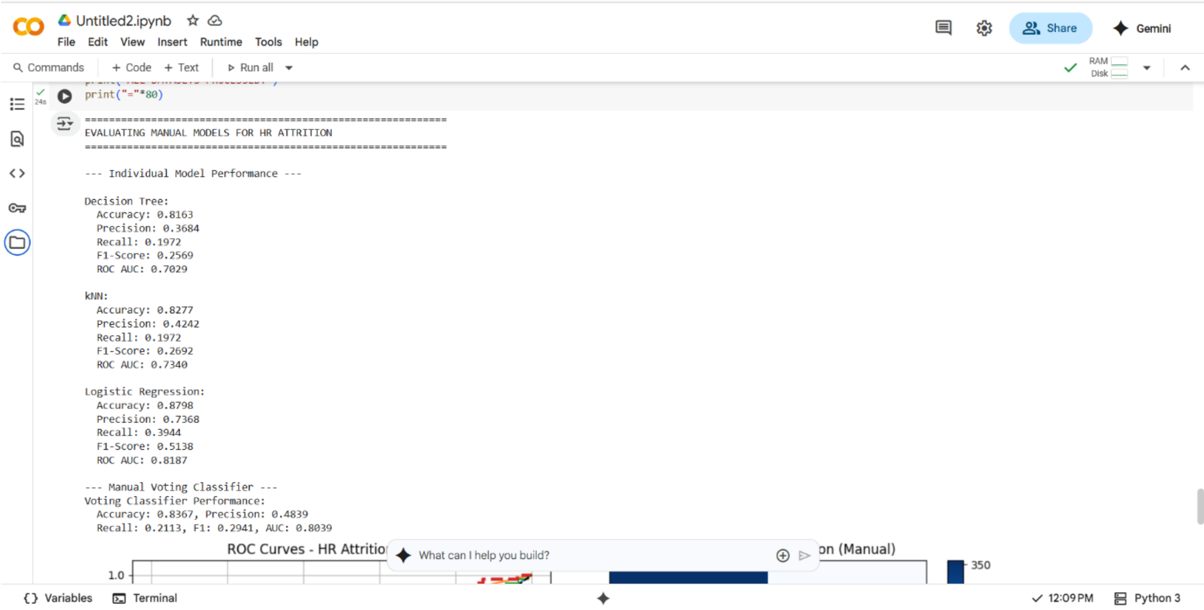
Best Model

- **Logistic Regression** is the best overall model for HR Attrition.
- Hypothesis: The dataset is high-dimensional after one-hot encoding, which suits Logistic Regression's linear decision boundary. Decision Trees overfit and kNN suffers from high-dimensional distance problems.

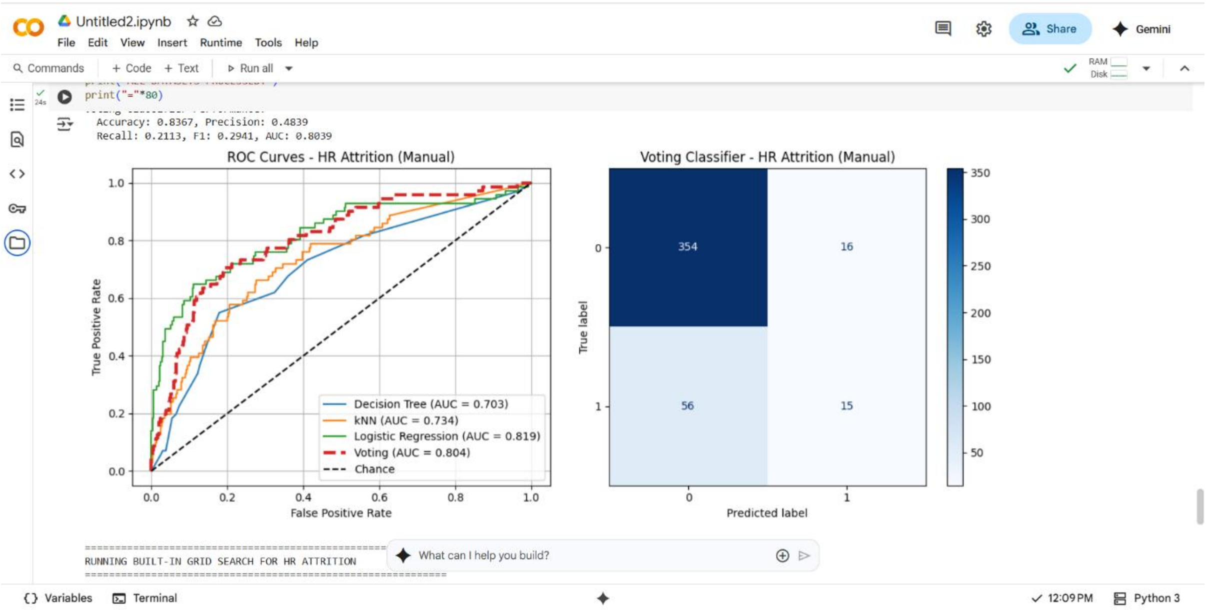
---

5. Screenshots

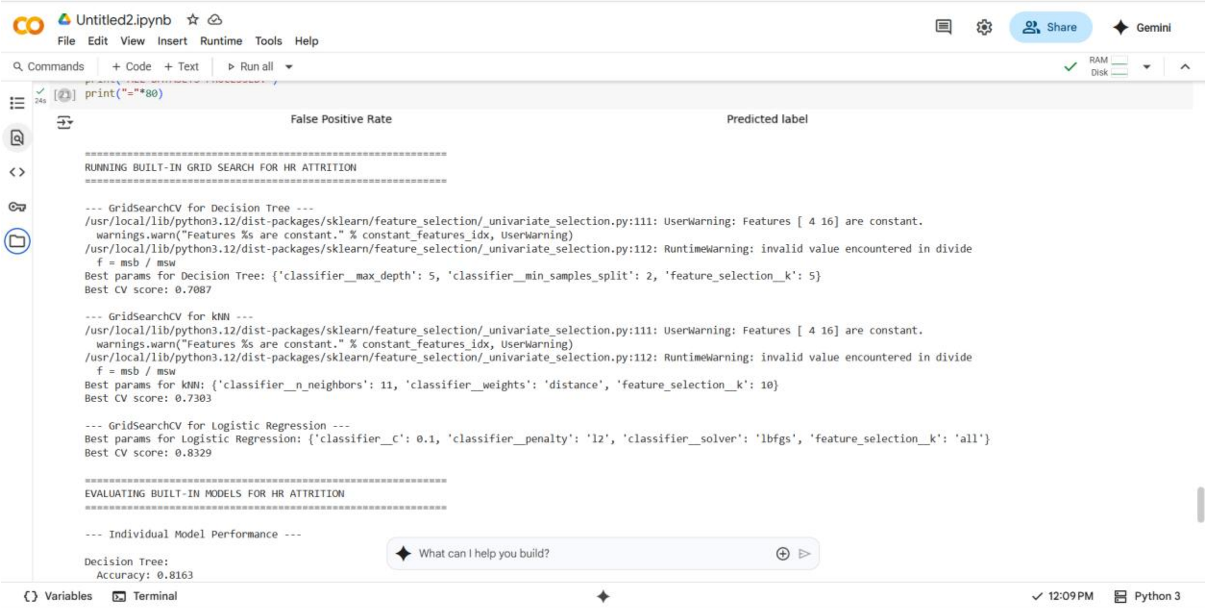
MANUAL METHOD :-



ROC :-

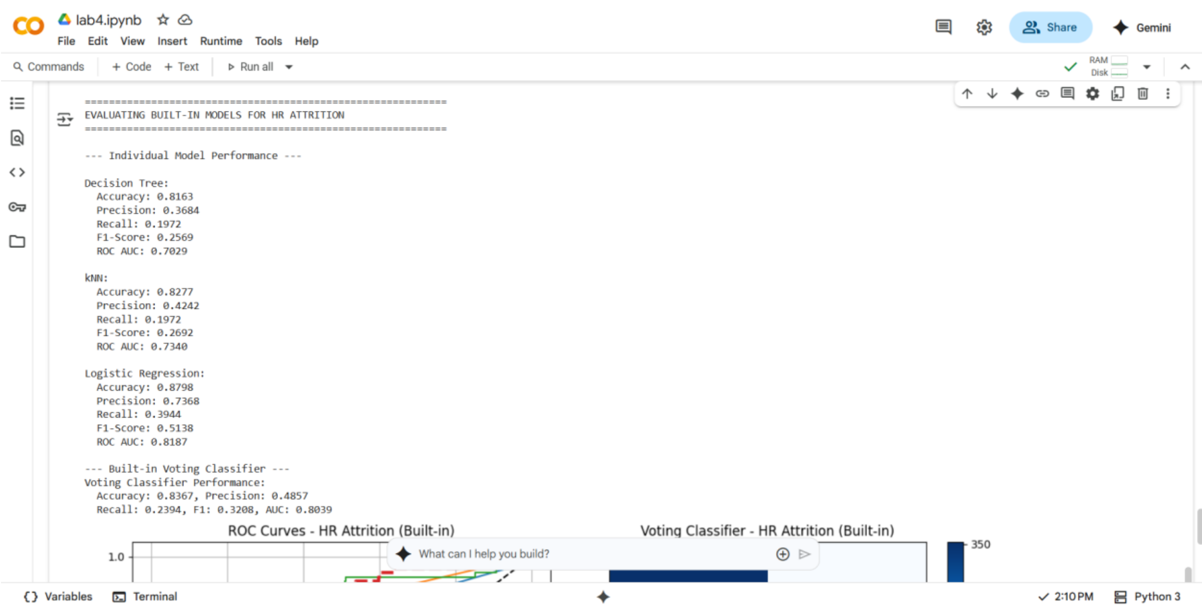


SCIKIT METHOD:-



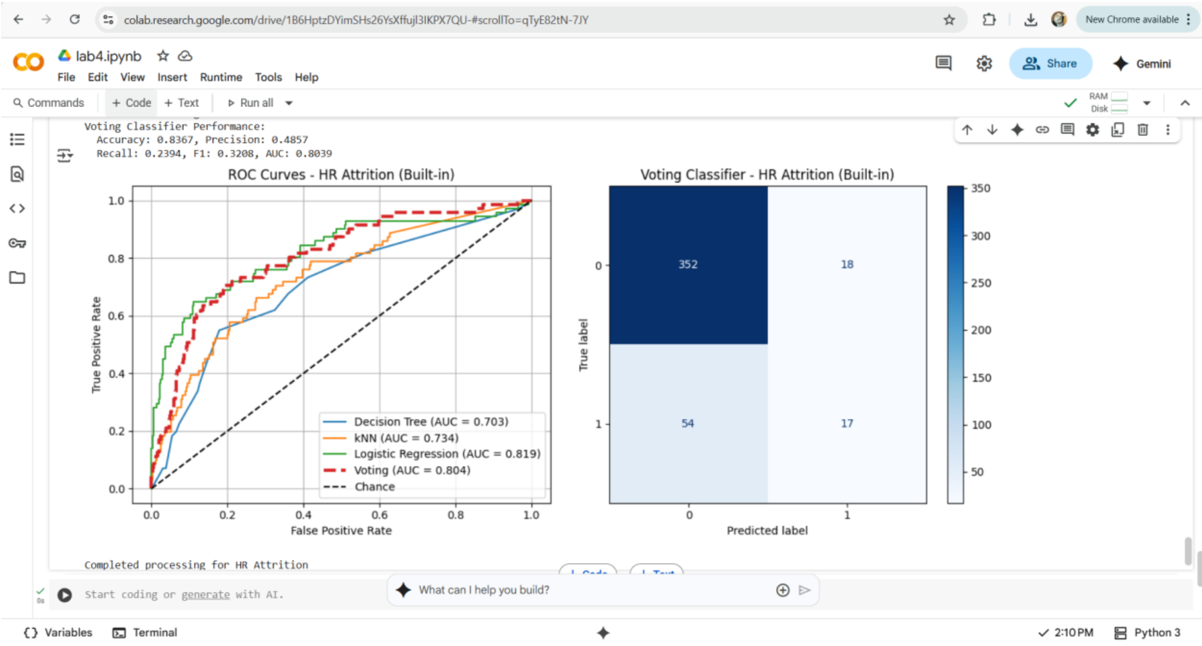
Untitled2.ipynb

```
print("="*80)

                     False Positive Rate                                   Predicted label

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

--- GridSearchCV for Decision Tree ---
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
Best params for Decision Tree: {'classifier__max_depth': 5, 'classifier__min_samples_split': 2, 'feature_selection__k': 5}
Best CV score: 0.7087

--- GridSearchCV for kNN ---
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
Best params for kNN: {'classifier__n_neighbors': 11, 'classifier__weights': 'distance', 'feature_selection__k': 10}
Best CV score: 0.7303

--- GridSearchCV for Logistic Regression ---
Best params for Logistic Regression: {'classifier__C': 0.1, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'feature_selection__k': 'all'}
Best CV score: 0.8329

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8163
```



lab4.ipynb

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8163
  Precision: 0.3684
  Recall: 0.1972
  F1-Score: 0.2569
  ROC AUC: 0.7029

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

Logistic Regression:
  Accuracy: 0.8798
  Precision: 0.7368
  Recall: 0.3944
  F1-Score: 0.5138
  ROC AUC: 0.8187

--- Built-in Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8367, Precision: 0.4857
  Recall: 0.2394, F1: 0.3208, AUC: 0.8039
```



6. Conclusion

- **Key Findings:** Logistic Regression consistently outperformed Decision Tree and kNN across manual and built-in methods.
- **Manual vs. Built-in:** Both approaches produced consistent results. The built-in method was more efficient, while the manual implementation helped in understanding grid search mechanics.
- **Takeaways:**
  - Logistic Regression is robust for high-dimensional, one-hot encoded datasets.
  - Ensembles do not always improve performance when one model dominates.
  - Manual grid search is educational, but libraries like scikit-learn are far more practical for real-world use.