UE23CS352A : Machine Learning

Week 4: Model Selection and Comparative Analysis

Name:P Padmesh	SRN: PES2UG24CS817
Class:5 C	Submission date: 01/09/2025

Part 1: Import Libraries and Setup

We start by bringing in all the libraries we need like pandas, numpy, sklearn etc. Without these nothing will run later so its kinda like setting up the toolbox first.

```
3.2 HR Attrition Dataset

[7] def load_hr_data():
        """
        Load and preprocess the HR Attrition dataset from uploaded file.
        """
        import pandas as pd
        from sklearn.model_selection import train_test_split

        # Read from local file (make sure name matches your uploaded file)
        data = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")

        # Encode target: Attrition (Yes=1, No=0)
        data['Attrition'] = data['Attrition'].map({'Yes': 1, 'No': 0})

        # Drop identifier column if present
        if 'EmployeeNumber' in data.columns:
            data = data.drop(['EmployeeNumber'], axis=1)

        # Convert categoricals to dummy variables
        data = pd.get_dummies(data, drop_first=True)

        # Split features and labels
        X = data.drop('Attrition', axis=1)
        y = data['Attrition']
```

```
        # Drop identifier column if present
        if 'EmployeeNumber' in data.columns:
            data = data.drop(['EmployeeNumber'], axis=1)

        # Convert categoricals to dummy variables
        data = pd.get_dummies(data, drop_first=True)

        # Split features and labels
        X = data.drop('Attrition', axis=1)
        y = data['Attrition']

        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.3, random_state=42, stratify=y
        )

        print("HR Attrition dataset loaded successfully.")
        print(f"Train shape: {X_train.shape}, Test shape: {X_test.shape}")
        return X_train, X_test, y_train, y_test, "HR Attrition"

    # ✅ Call function
    X_train, X_test, y_train, y_test, dataset_name = load_hr_data()

HR Attrition dataset loaded successfully.
Train shape: (1029, 46), Test shape: (441, 46)
```

2.Models and Parameter Grids

Here we set up the 3 models we're using (Decision Tree, kNN, Logistic Regression) and also their parameter grids. The grids basically tell the code what values to try when tuning the models.

3.1 Dataset Loading Functions

We write small functions to load and prep the datasets. This makes it easier since we just call the function instead of repeating code.

3.2 HR Attrition Dataset

This dataset is about predicting if an employee will leave the company. We clean it by dropping ID, encoding categories etc. Its like a real world HR problem.

Part 4: Manual Grid Search Implementation

Manual grid search means we loop through different parameters ourselves using cross-validation. It's slower and more code but we can see exactly how the model performs at each step.

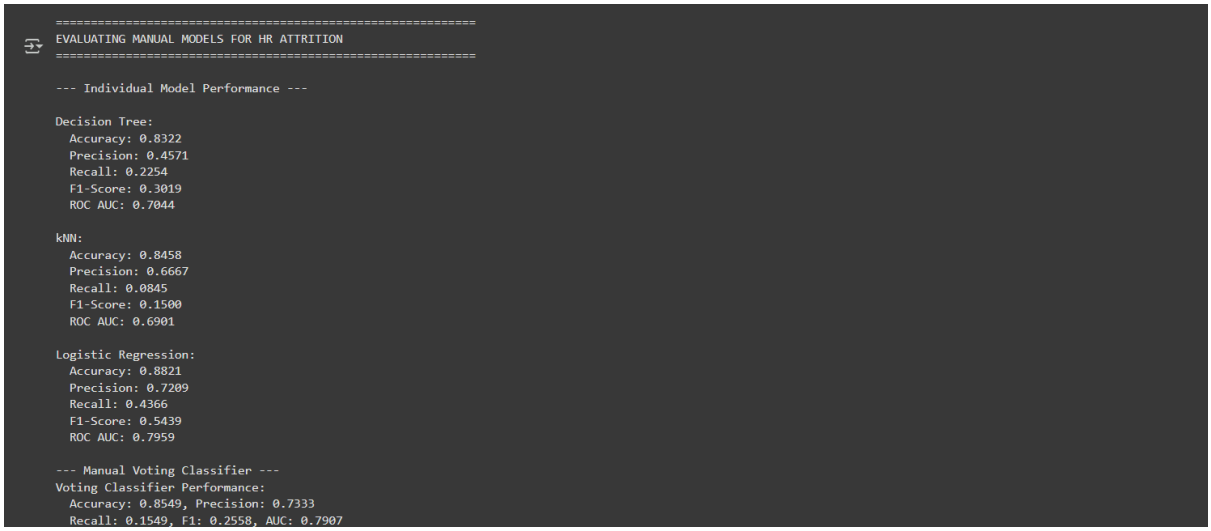
Part 5: Built-in Grid Search Implementation

Here we use sklearn's GridSearchCV which does all the work for us. It's way faster and cleaner, and even uses multiple cores.

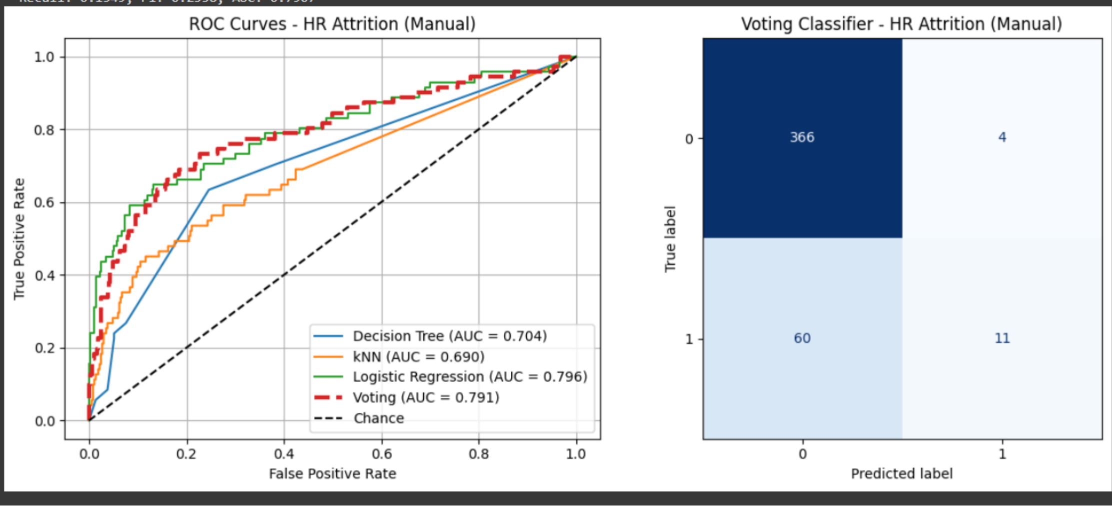
Part 6: Model Evaluation and Voting Classifiers

We check how each model did with metrics like accuracy, recall, f1 etc. Then we also try combining the models with a voting classifier so they vote on predictions.

Output:



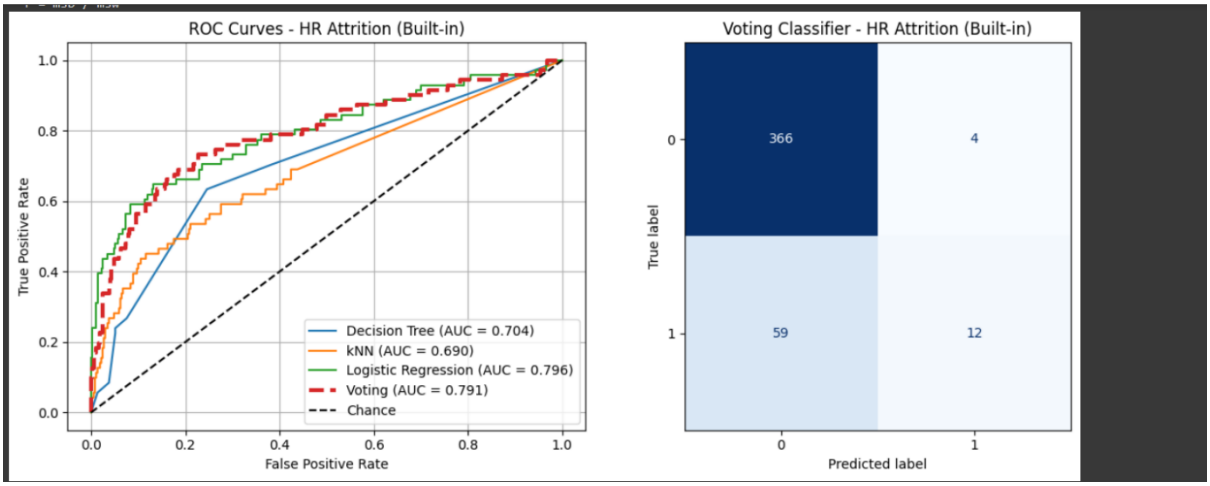
```
--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8549, Precision: 0.7333
  Recall: 0.1549, F1: 0.2558, AUC: 0.7907
```



This graph shows how the models perform on the HR attrition dataset. The ROC curves tell us Logistic Regression is the strongest single model (AUC ~0.80), while the Voting Classifier combines models to get close performance (AUC ~0.79). The confusion matrix reveals the classifier is good at

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8322
  Precision: 0.4571
  Recall: 0.2254
  F1-Score: 0.3019
  ROC AUC: 0.7044

kNN:
  Accuracy: 0.8458
  Precision: 0.6667
  Recall: 0.0845
  F1-Score: 0.1500
  ROC AUC: 0.6901

Logistic Regression:
  Accuracy: 0.8821
  Precision: 0.7209
  Recall: 0.4366
  F1-Score: 0.5439
  ROC AUC: 0.7959
```

```
--- Built-in Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8571, Precision: 0.7500
  Recall: 0.1690, F1: 0.2759, AUC: 0.7907
/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
/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
/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
```




```
(array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
        0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        0]),
```

```
array([0.24719043, 0.04501292, 0.14798925, 0.10614924, 0.11864059,
       0.05842378, 0.02738958, 0.02659698, 0.17467678, 0.02440618,
       0.07159677, 0.15370555, 0.31507412, 0.16065317, 0.07370092,
       0.05880117, 0.02485104, 0.41367155, 0.03272388, 0.50991316,
       0.08672687, 0.0250815 , 0.36582459, 0.02428831, 0.02570761,
       0.20376388, 0.05929396, 0.11772373, 0.17943917, 0.27971698,
       0.07677307, 0.24062739, 0.04327277, 0.06582524, 0.05908924,
       0.02424686, 0.47105488, 0.03516762, 0.08059623, 0.52113801,
       0.05740129, 0.04536978, 0.02855296, 0.02450425, 0.02582812,
       0.08134818, 0.12078123, 0.70473836, 0.30329722, 0.16152346,
       0.0256519 , 0.27743496, 0.02649998, 0.03684827, 0.07101898,
       0.09890786, 0.0387991 , 0.03301212, 0.12413572, 0.09336045,
       0.05998099, 0.02940669, 0.03280872, 0.09617796, 0.61975358,
       0.0260083 , 0.18673303, 0.11568525, 0.11101054, 0.06733297,
       0.34351333, 0.19143092, 0.07812586, 0.12108115, 0.0641284 ,
       0.03109987, 0.06983883, 0.12963013, 0.19273875, 0.4399874 ,
       0.0288226 , 0.10240067, 0.10508062, 0.08333038, 0.09675811,
       0.11360358, 0.11676035, 0.21068057, 0.03758885, 0.0633692 ,
       0.21937433, 0.06750513, 0.0768603 , 0.07493412, 0.08752105,
       0.0347226 , 0.17129496, 0.07248055, 0.05019144, 0.07489291,
       0.08260124, 0.18061262, 0.05042073, 0.03051942, 0.03589259,
       0.04677202, 0.11708916, 0.02581203, 0.20510301, 0.03690284,
       0.11105395, 0.02949333, 0.04197482, 0.02539146, 0.48909926,
       0.03675354, 0.06369568, 0.13641012, 0.14372088, 0.03032993,
       0.11380875, 0.38782482, 0.15956608, 0.02940797, 0.47058954,
       0.13585744, 0.02480464, 0.0593647 , 0.03975861, 0.25016445,
       0.15360978, 0.03040089, 0.09084721, 0.02551036, 0.0255571 ,
       0.09347961, 0.21052848, 0.13253801, 0.07400586, 0.55545581,
       0.22703587, 0.26949024, 0.03739708, 0.09493663, 0.1206587 ,
       0.08323083, 0.07984754, 0.13137426, 0.02991616, 0.03247536,
       0.11717335, 0.30171885, 0.0454897 , 0.02436212, 0.02462382,
       0.19403237, 0.06063969, 0.59472141, 0.31011779, 0.02444332,
       0.13943473, 0.33539821, 0.08597733, 0.40025619, 0.10208523,
       0.14019103, 0.13938174, 0.02452723, 0.13249483, 0.10866289,
       0.13187423, 0.04456307, 0.02453507, 0.23731324, 0.03681619,
       0.24797968, 0.03811917, 0.12051158, 0.02593903, 0.10753125,
       0.1366097 , 0.10689968, 0.02776826, 0.07574065, 0.09130981,
```

```
0.22703587, 0.26949024, 0.03739708, 0.09493663, 0.1206587 ,
0.08323083, 0.07984754, 0.13137426, 0.02991616, 0.03247536,
0.11717335, 0.30171885, 0.0454897 , 0.02436212, 0.02462382,
0.19403237, 0.06063969, 0.59472141, 0.31011779, 0.02444332,
0.13943473, 0.33539821, 0.08597733, 0.40025619, 0.10208523,
0.14019103, 0.13938174, 0.02452723, 0.13249483, 0.10866289,
0.13187423, 0.04456307, 0.02453507, 0.23731324, 0.03681619,
0.24797968, 0.03811917, 0.12051158, 0.02593903, 0.10753125,
0.1366097 , 0.10689968, 0.02776826, 0.07574065, 0.09130981,
0.02453415, 0.17690559, 0.13576385, 0.28095825, 0.04768855,
0.25944131, 0.09558287, 0.02597809, 0.11971399, 0.11383067,
0.08162383, 0.14369482, 0.02763365, 0.02724739, 0.17281265,
0.03491112, 0.03525821, 0.04559182, 0.02766978, 0.55573758,
0.07509319, 0.37581633, 0.08701802, 0.33387226, 0.93966973,
0.0290276 , 0.02807804, 0.36969365, 0.05871142, 0.02715387,
0.11834635, 0.3199349 , 0.28483101, 0.09063767, 0.30621713,
0.28598352, 0.06332846, 0.06949796, 0.08079953, 0.20163619,
0.02518305, 0.26092788, 0.08059158, 0.14735195, 0.59095925,
0.31853394, 0.10478363, 0.26711895, 0.09395474, 0.26228531,
0.26919425, 0.02888285, 0.11502954, 0.43196793, 0.02460447,
0.11020441, 0.50582761, 0.07508106, 0.14511017, 0.13889302,
0.14577015, 0.27923481, 0.0321916 , 0.10886786, 0.05698198,
0.06843044, 0.1492221 , 0.02416482, 0.07078506, 0.06746735,
0.03379556, 0.02497603, 0.12783644, 0.02790435, 0.3150842 ,
0.06440313, 0.02454438, 0.15583941, 0.06300708, 0.17660332,
0.12734222, 0.0619201 , 0.46225949, 0.11520111, 0.0759374 ,
0.07209999, 0.33964876, 0.07353484, 0.05717243, 0.23273158,
0.11299499, 0.07601601, 0.1160358 , 0.0541071 , 0.02458596,
0.07268296, 0.13187784, 0.17893701, 0.20520179, 0.03512333,
0.74159836, 0.18090833, 0.04635691, 0.15807619, 0.0732929 ,
0.07053892, 0.06865863, 0.09624116, 0.04780494, 0.07329423,
0.02449043, 0.02858545, 0.23710997, 0.03163212, 0.14839864,
0.25549243, 0.12868481, 0.22167457, 0.79259302, 0.29091866,
0.03035405, 0.02508707, 0.32945152, 0.07361286, 0.04821735,
0.07610505, 0.04455471, 0.40772668, 0.8241956 , 0.26368895,
0.1530216 , 0.2560751 , 0.02548355, 0.15844732, 0.14261247,
0.06432695, 0.09805341, 0.06744745, 0.0350373 , 0.13018419,
```

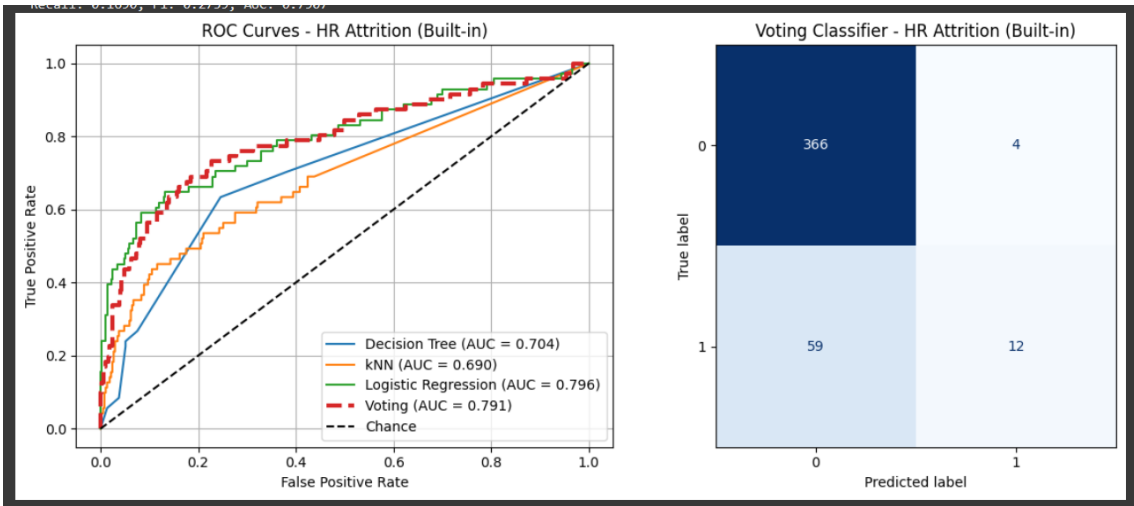
```
 0.02950485, 0.14290373, 0.03235395, 0.30265409, 0.30725511,
 0.07795075, 0.06317791, 0.04074665, 0.02975587, 0.36047876,
 0.0585749 , 0.38506301, 0.09129069, 0.07532046, 0.03110205,
 0.02573385, 0.02456372, 0.11037979, 0.19238073, 0.03583696,
 0.10635016, 0.30148818, 0.02485632, 0.0802188 , 0.2533831 ,
 0.03094066, 0.11153493, 0.23570408, 0.024552  , 0.1535727 ,
 0.03543203, 0.02615592, 0.467838  , 0.25607899, 0.11099547,
 0.12892759, 0.29002561, 0.09136157, 0.04099182, 0.07805069,
 0.04158992, 0.02573416, 0.07719405, 0.02561919, 0.10150968,
 0.05877968, 0.02470329, 0.03389634, 0.06070297, 0.0343217 ,
 0.31048583, 0.0575928 , 0.06556159, 0.03066661, 0.56408719,
 0.47376616, 0.06916904, 0.38403713, 0.25377245, 0.02497518,
 0.07125037, 0.24003104, 0.1446494 , 0.15767297, 0.03175628,
 0.15127149, 0.08124955, 0.50419099, 0.0699444 , 0.03051739,
 0.10413204, 0.24162741, 0.14700385, 0.02761649, 0.02511218,
 0.26682114, 0.1353717 , 0.32071907, 0.06694986, 0.196604  ,
 0.16291006, 0.03148632, 0.05806613, 0.02562438, 0.08323688,
 0.12211139, 0.09396649, 0.08510662, 0.13109742, 0.07009575,
 0.18110362, 0.03285248, 0.06962638, 0.0396069 , 0.23958875,
 0.43679123, 0.11247032, 0.45190025, 0.04085926, 0.45713875,
 0.3130946 , 0.35503809, 0.25382152, 0.13470515, 0.07611432,
 0.14701583, 0.18103372, 0.04373769, 0.11394966, 0.11558542,
 0.13976353, 0.50376546, 0.08121544, 0.40232052, 0.15292157,
 0.07660314]))
```

Part 7: Complete Pipeline Function

This part connects everything into one big function that runs the whole experiment for a dataset. Makes life easy since we don't have to run each step manually.

output:

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8322
  Precision: 0.4571
  Recall: 0.2254
  F1-Score: 0.3019
  ROC AUC: 0.7044

kNN:
  Accuracy: 0.8458
  Precision: 0.6667
  Recall: 0.0845
  F1-Score: 0.1500
  ROC AUC: 0.6901

Logistic Regression:
  Accuracy: 0.8821
  Precision: 0.7209
  Recall: 0.4366
  F1-Score: 0.5439
  ROC AUC: 0.7959

--- Built-in Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8571, Precision: 0.7500
  Recall: 0.1690, F1: 0.2759, AUC: 0.7907
```

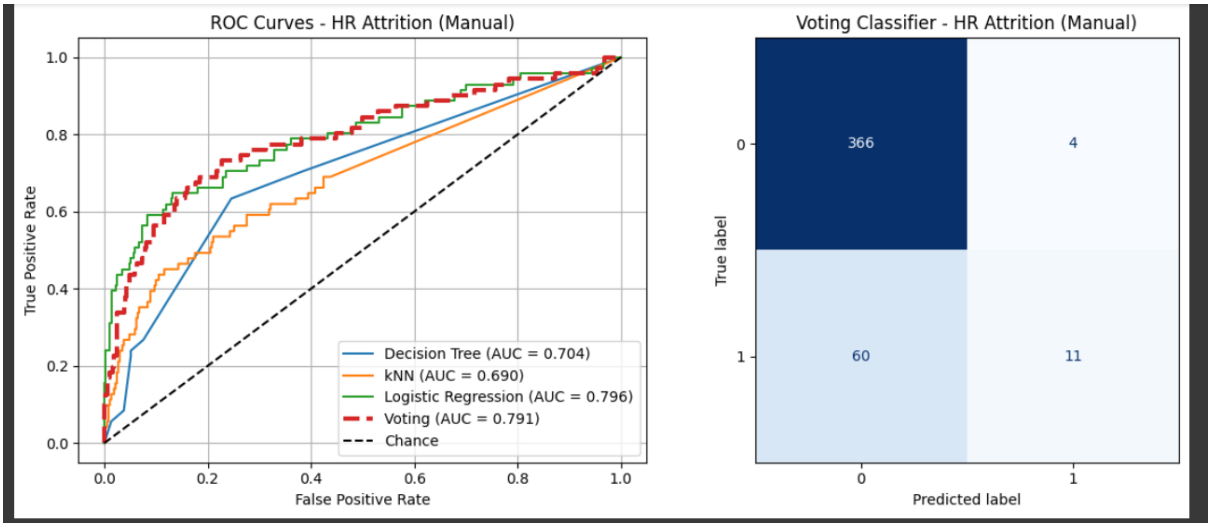



```
Completed processing for HR Attrition
================================================================================
```

Part 8: Execute the Complete Lab

Finally we run the pipeline. In my case I only ran it for the HR dataset since that's what the lab asked. This gives the final results and plots.

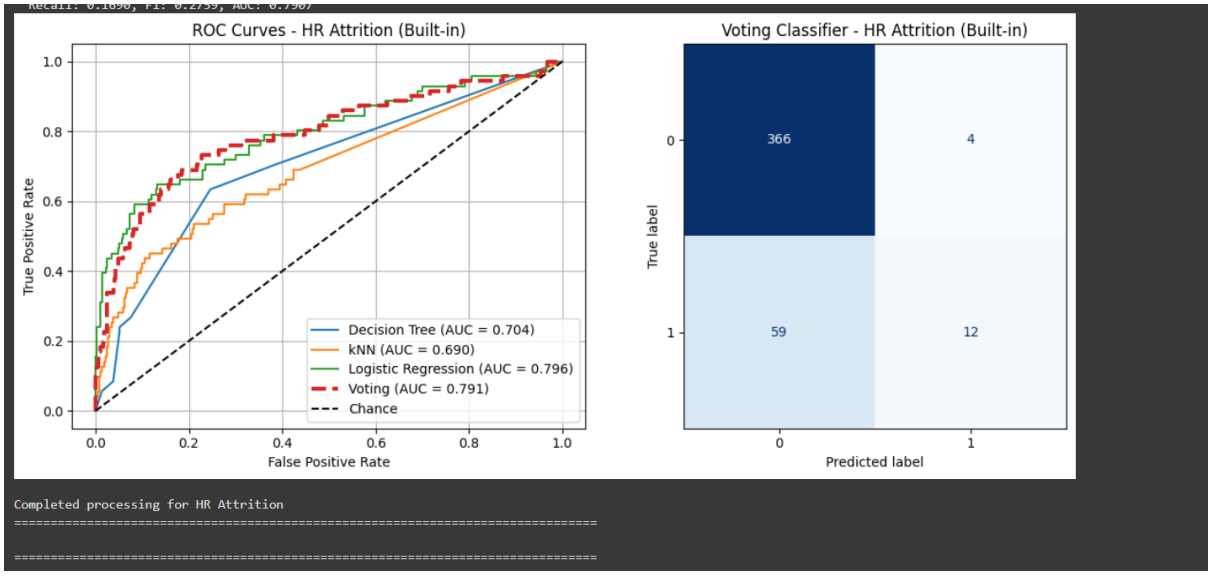
```
============================================================
EVALUATING MANUAL MODELS FOR HR ATTRITION
============================================================

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8322
  Precision: 0.4571
  Recall: 0.2254
  F1-Score: 0.3019
  ROC AUC: 0.7044

kNN:
  Accuracy: 0.8458
  Precision: 0.6667
  Recall: 0.0845
  F1-Score: 0.1500
  ROC AUC: 0.6901

Logistic Regression:
  Accuracy: 0.8821
  Precision: 0.7209
  Recall: 0.4366
  F1-Score: 0.5439
  ROC AUC: 0.7959

--- Manual Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8549, Precision: 0.7333
  Recall: 0.1549, F1: 0.2558, AUC: 0.7907
```



The ROC curve on the left shows that Logistic Regression (AUC = 0.796) performs best at distinguishing employees who leave versus stay, closely followed by the Voting classifier (AUC = 0.791). The confusion matrix on the right indicates that the Voting classifier predicts most non-attrition cases correctly (366 out of 370) but struggles with attrition cases, correctly identifying only 12 out of 71. Overall, the models are good at identifying non-attrition but less accurate at predicting attrition.

```
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': 3, 'classifier__min_samples_split': 2, 'feature_selection__k': 10}
Best CV score: 0.7082

--- 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': 7, 'classifier__weights': 'distance', 'feature_selection__k': 30}
Best CV score: 0.7103

--- GridSearchCV for Logistic Regression ---
/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
/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
/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
/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 Logistic Regression: {'classifier__C': 1, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'feature_selection__k': 30}
```

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

--- Individual Model Performance ---

Decision Tree:
  Accuracy: 0.8322
  Precision: 0.4571
  Recall: 0.2254
  F1-Score: 0.3019
  ROC AUC: 0.7044

kNN:
  Accuracy: 0.8458
  Precision: 0.6667
  Recall: 0.0845
  F1-Score: 0.1500
  ROC AUC: 0.6901

Logistic Regression:
  Accuracy: 0.8821
  Precision: 0.7209
  Recall: 0.4366
  F1-Score: 0.5439
  ROC AUC: 0.7959

--- Built-in Voting Classifier ---
Voting Classifier Performance:
  Accuracy: 0.8571, Precision: 0.7500
  Recall: 0.1690, F1: 0.2759, AUC: 0.7907
```



Conclusion:

Among the three models tested on the HR Attrition dataset, Logistic Regression performed the best overall. It achieved the highest accuracy (88.2%) and also the strongest ROC-AUC score (0.796), making it the most reliable for distinguishing between employees likely to leave or stay. The Decision Tree and kNN models showed decent accuracy but struggled with recall and overall discrimination ability, as reflected in their lower ROC-AUC values. The ensemble Voting Classifier gave balanced results but still did not surpass Logistic Regression. This shows that for this dataset, a simpler linear model outperformed more complex approaches.

Key findings:

- Logistic Regression was the best-performing model with 88.2% accuracy and the highest ROC-AUC (0.796).
- The Decision Tree and kNN models achieved reasonable accuracy but had poor recall and weaker ROC-AUC scores, limiting their effectiveness.
- The ensemble Voting Classifier provided balanced results but did not outperform Logistic Regression.
- This suggests that, for the HR Attrition dataset, a **simpler linear model** generalized better than more complex algorithms.