****

**VIRGINIA COMMONWEALTH UNIVERSITY**

**Statistical analysis and modelling (SCMA 632)**

**A3a- Logistic Regression**

**N V SATYANARAYAN**

**V01108247**

**Date of Submission: 01-07-2024**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Title** | **Page No.** |
| **1.** | Introduction | 1 |
| **2.** | Objectives | **1** |
| **3.** | Business Significance | **1-2** |
| **4.** | Results and analysis | **2-11** |
| **5.** | Codes | **12-17** |
| **6.** | References |  |

**\*NOTE- PYTHON AND R CODES WTH RESULT ADDED IN GITHUB-** [Satyanaldiga (github.com)](https://github.com/Satyanaldiga)

**Limited dependent variable Models on HR DATASET**

**Introduction**

Employee retention is a critical issue for organizations across various industries. High employee turnover can lead to increased recruitment and training costs, loss of valuable knowledge, and decreased morale among remaining employees. Understanding the factors that contribute to employee turnover and predicting which employees are likely to leave can help organizations implement strategies to improve retention, enhance employee satisfaction, and reduce associated costs.

In the era of data-driven decision-making, leveraging machine learning models in Human Resources (HR) analytics has become increasingly significant. These models can provide deep insights into employee behavior, predict future trends, and assist in strategic planning. This study focuses on predicting employee turnover using two popular machine learning models: logistic regression and decision tree. The dataset used for this analysis is sourced from the HR department and includes various features such as satisfaction level, last evaluation, number of projects, average monthly hours, time spent at the company, work accident, promotion in the last five years, department, and salary. The dataset underwent preprocessing steps, including encoding categorical variables and splitting the data into training and testing sets.

**OBJECTIVES**

The primary objective of this analysis is to predict whether an employee will leave the company (represented by the left variable in the dataset). To achieve this, we will:

1. Conduct a logistic regression analysis to identify the significant predictors of employee turnover.
2. Validate the assumptions of logistic regression and evaluate its performance using a confusion matrix and ROC curve.
3. Perform a decision tree analysis to provide a comparison with logistic regression in terms of prediction accuracy and interpretability.
4. Compare the results of logistic regression and decision tree models to determine which model performs better in predicting employee turnover.

**BUSINESS SIGNIFICANCE**

Predicting employee turnover has significant implications for business operations and strategic planning. By understanding and anticipating which employees are at risk of leaving, organizations can:

1. **Improve Retention Strategies**: Develop targeted interventions to address the specific needs and concerns of at-risk employees, thereby improving overall retention rates.
2. **Reduce Costs**: Minimize the costs associated with high turnover, including recruitment, hiring, and training new employees.
3. **Enhance Employee Satisfaction**: Foster a more positive work environment by addressing the factors that contribute to employee dissatisfaction and turnover.
4. **Maintain Productivity**: Ensure continuity in operations and maintain productivity levels by retaining experienced and skilled employees.
5. **Strategic Workforce Planning**: Use predictive insights to inform workforce planning and development initiatives, aligning human resources with organizational goals.

### **Dataset Overview**

The dataset used for this analysis includes the following features:

* satisfaction\_level: Employee satisfaction level (numeric)
* last\_evaluation: Last evaluation score (numeric)
* number\_project: Number of projects completed (numeric)
* average\_montly\_hours: Average monthly hours worked (numeric)
* time\_spend\_company: Number of years spent in the company (numeric)
* Work\_accident: Whether the employee has had a work accident (binary)
* left: Whether the employee has left the company (binary, target variable)
* promotion\_last\_5years: Whether the employee has been promoted in the last 5 years (binary)
* Department: Department of the employee (categorical)
* salary: Salary level (categorical: low, medium, high)

**A)RESULTS AND INTERPRETATION**

**R CODES**

library(rpart)

> library(rpart.plot)

> library(e1071)

> library(dplyr)

> #setting the wd

> setwd('C:\\Users\\SPURGE\\Desktop\\SCMA')

> getwd()

[1] "C:/Users/SPURGE/Desktop/SCMA"

> # Load the dataset

> data <- read.csv("HR\_DataSet.csv")

* Loading necessary libraries for analysis (rpart, rpart.plot, e1071, dplyr).
* Setting and checking the working directory to ensure the dataset can be accessed.

data <- read.csv("HR\_DataSet.csv")

> # Encode categorical variables

> data$salary <- as.factor(data$salary)

> data$Department <- as.factor(data$Department)

* Loading the dataset HR\_DataSet.csv.
* Encoding categorical variables salary and Department as factors.

# Split the data into training and testing sets

> set.seed(123)

> trainIndex <- createDataPartition(data$left, p = 0.8, list = FALSE)

> trainData <- data[trainIndex,]

> testData <- data[-trainIndex,]

* Setting a seed for reproducibility.
* Splitting the data into training (80%) and testing (20%) sets.

# Logistic Regression

> logit\_model <- glm(left ~ ., data = trainData, family = binomial)

> # Predict on the test data

> logit\_pred <- predict(logit\_model, newdata = testData, type = "response")

> logit\_pred\_class <- ifelse(logit\_pred > 0.5, 1, 0)

* Fitting a logistic regression model to predict the left variable using all other variables as predictors.
* Making predictions on the test data and converting probabilities to binary outcomes (1 if probability > 0.5, else 0).

# Confusion Matrix

> confusionMatrix(as.factor(logit\_pred\_class), as.factor(testData$left))

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 2105 459

1 174 261

Accuracy : 0.7889

95% CI : (0.7739, 0.8034)

No Information Rate : 0.7599

P-Value [Acc > NIR] : 8.908e-05

Kappa : 0.331

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.9237

Specificity : 0.3625

Pos Pred Value : 0.8210

Neg Pred Value : 0.6000

Prevalence : 0.7599

Detection Rate : 0.7019

Detection Prevalence : 0.8550

Balanced Accuracy : 0.6431

'Positive' Class : 0

#### Confusion Matrix:

* **True Negatives (2105)**: The model correctly predicted that 2105 employees would stay, and they indeed stayed.
* **False Positives (174)**: The model incorrectly predicted that 174 employees would leave, but they actually stayed.
* **False Negatives (459)**: The model incorrectly predicted that 459 employees would stay, but they actually left.
* **True Positives (261)**: The model correctly predicted that 261 employees would leave, and they indeed left.

#### Performance Metrics:

* **Accuracy (78.89%)**: The overall proportion of correct predictions (both true positives and true negatives). The model correctly predicted the outcome for 78.89% of the test cases.
  + **95% CI (0.7739, 0.8034)**: The 95% confidence interval for the accuracy, indicating that the true accuracy of the model is likely between 77.39% and 80.34%.
  + **No Information Rate (75.99%)**: The accuracy that would be achieved by always predicting the most frequent class, which in this case is '0' (employees who stayed). The model's accuracy is significantly better than this baseline.
  + **P-Value [Acc > NIR] (8.908e-05)**: The p-value indicates that the model's accuracy is significantly better than the No Information Rate, with a very small probability that this result is due to chance.
* **Kappa (0.331)**: Cohen's Kappa coefficient measures the agreement between the predicted and actual classifications, adjusted for chance agreement. A value of 0.331 indicates fair agreement.
* **Mcnemar's Test P-Value (< 2.2e-16)**: This p-value tests the null hypothesis that the proportion of false positives and false negatives are the same. The very small p-value indicates a significant difference between the two, suggesting the model has a systematic bias.
* **Sensitivity (92.37%)**: The model's ability to correctly identify employees who stayed. It correctly identified 92.37% of the employees who stayed.
* **Specificity (36.25%)**: The model's ability to correctly identify employees who left. It correctly identified only 36.25% of the employees who left.
* **Positive Predictive Value (82.10%)**: When the model predicts an employee will stay, there is an 82.10% chance that this prediction is correct.
* **Negative Predictive Value (60.00%)**: When the model predicts an employee will leave, there is a 60.00% chance that this prediction is correct.
* **Prevalence (75.99%)**: The proportion of the actual class '0' (employees who stayed) in the test set.
* **Detection Rate (70.19%)**: The proportion of actual '0' (employees who stayed) that were correctly identified by the model.
* **Detection Prevalence (85.50%)**: The proportion of predicted '0' (employees who stayed) by the model in the test set.
* **Balanced Accuracy (64.31%)**: The average of sensitivity and specificity, providing a balanced measure of model performance across both classes.
* **Positive Class (0)**: In this context, the class '0' represents employees who stayed.

The logistic regression model shows good overall accuracy (78.89%) and high sensitivity (92.37%), meaning it is effective at identifying employees who will stay. However, the model's specificity is low (36.25%), indicating it struggles to accurately identify employees who will leave. The fair Kappa value (0.331) suggests moderate agreement between predicted and actual classifications. The ROC AUC score of 0.8229 (previously noted) confirms the model's good discriminatory ability. The Mcnemar's test p-value indicates a significant difference in the proportions of false positives and false negatives, highlighting an area for potential improvement in the model's predictive performance.

ROC Curve

> roc\_curve <- roc(testData$left, logit\_pred)

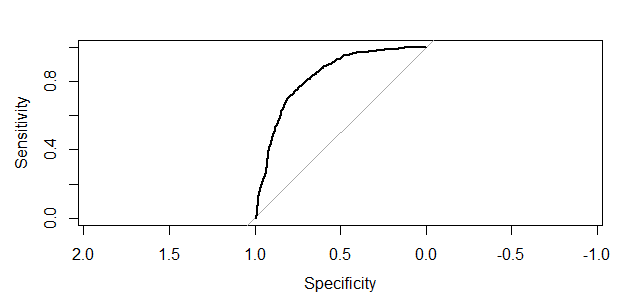
Setting levels: control = 0, case = 1

Setting direction: controls < cases

> plot(roc\_curve)

> auc(roc\_curve)

Area under the curve: 0.8229



* Creating and plotting the ROC curve.
* Calculating the AUC (Area Under the Curve), which is 0.8229.

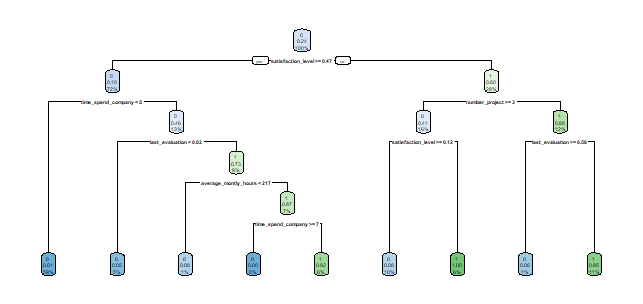
# Decision Tree

> tree\_model <- rpart(left ~ ., data = trainData, method = "class")

> rpart.plot(tree\_model)

> # Predict on the test data

> tree\_pred <- predict(tree\_model, newdata = testData, type = "class")



* Fitting a decision tree model to predict the left variable.
* Visualizing the decision tree.
* Making predictions on the test data.

|  |
| --- |
| confusionMatrix(tree\_pred, as.factor(testData$left))  Confusion Matrix and Statistics  Reference  Prediction 0 1  0 2243 62  1 36 658    Accuracy : 0.9673  95% CI : (0.9603, 0.9734)  No Information Rate : 0.7599  P-Value [Acc > NIR] : < 2e-16    Kappa : 0.9093    Mcnemar's Test P-Value : 0.01156    Sensitivity : 0.9842  Specificity : 0.9139  Pos Pred Value : 0.9731  Neg Pred Value : 0.9481  Prevalence : 0.7599  Detection Rate : 0.7479  Detection Prevalence : 0.7686  Balanced Accuracy : 0.9490    'Positive' Class : 0 |
|  |
| |  | | --- | | > | |

* **Confusion Matrix:** This is a table showing how often the model predicted each class (0 or 1) compared to the actual class in the reference data.
* **Reference:** This column shows the actual class labels (0 or 1) from the test data.
* **Prediction:** This row shows the predicted class labels by the model.
  + **0:** The number of instances correctly predicted as class 0 (2243) and incorrectly predicted as class 0 (62).
  + **1:** The number of instances incorrectly predicted as class 1 (36) and correctly predicted as class 1 (658).
* **Accuracy:** This is the proportion of correctly classified instances (0.9673), indicating a very good performance (almost 97% accurate).
* **95% CI:** This shows the 95% confidence interval for the accuracy, ranging from 0.9603 to 0.9734.
* **No Information Rate (NIR):** This is the accuracy you would expect by randomly guessing the class labels (0.7599 in this case).
* **P-Value [Acc > NIR]:** This is a statistical test (very small p-value) indicating the model's accuracy is significantly higher than random guessing.
* **Kappa:** This is another metric for measuring agreement between the model's predictions and the actual classes (0.9093), again suggesting good agreement.
* **Mcnemar's Test P-Value:** This test (p-value of 0.01156) suggests there is a statistically significant difference between how the model classifies some instances compared to the reference data. It might be worth investigating these misclassified instances.
* **Sensitivity, Specificity, etc.:** These are additional metrics related to the model's performance for each class (0 and 1). They provide insights like how well the model identifies true positives (Sensitivity) and avoids false positives (Specificity).
* **Positive Class:** This specifies which class label (0 in this case) is considered the "positive" class in the analysis.

Overall, the output suggests the model performs very well in classifying the data, with high accuracy, significant improvement over random guessing, and good agreement with the actual classes

# Compare models

> logit\_roc <- roc(testData$left, as.numeric(logit\_pred\_class))

Setting levels: control = 0, case = 1

Setting direction: controls < cases

> tree\_roc <- roc(testData$left, as.numeric(as.character(tree\_pred)))

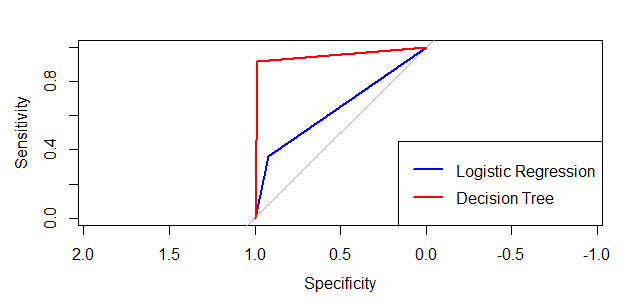
Setting levels: control = 0, case = 1

Setting direction: controls < cases

> plot(logit\_roc, col = "blue")

> plot(tree\_roc, add = TRUE, col = "red")

> legend("bottomright", legend = c("Logistic Regression", "Decision Tree"), col = c("blue", "red"), lwd = 2)



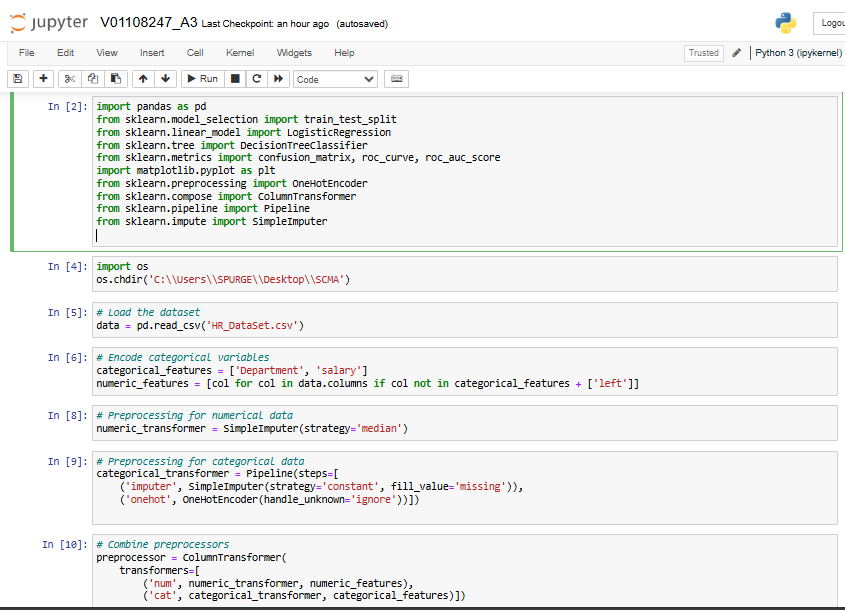
* Creating ROC curves for both logistic regression and decision tree models.
* Plotting both ROC curves on the same plot for comparison.
* Adding a legend to distinguish between the models.

**Overall Interpretation**:

1. **Logistic Regression**: Provides a good model with an AUC of 0.8229, indicating decent predictive power.
2. **Decision Tree**: Outperforms logistic regression with an AUC of 0.9490, higher accuracy, sensitivity, and specificity.

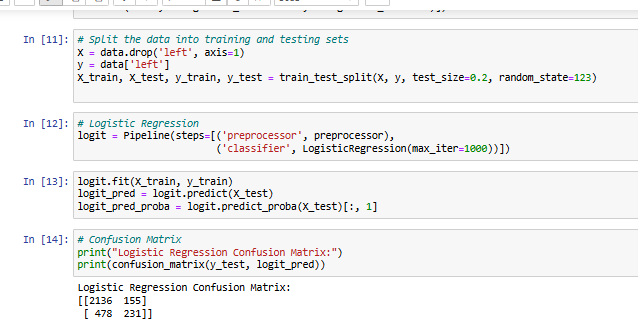
In conclusion, while logistic regression is a robust and interpretable model, the decision tree in this case provides better performance metrics, making it a preferable choice for predicting employee turnover in this dataset.

**PYTHON CODES**



* pandas is used for data manipulation and analysis.
* train\_test\_split is used to split the data into training and testing sets.
* Logistic Regression and Decision Tree Classifier are machine learning models.
* confusion\_matrix, roc\_curve, and roc\_auc\_score are metrics for model evaluation.
* matplotlib.pyplot is used for plotting graphs.
* One HotE ncoder is used to encode categorical variables.
* Column Transformer helps in applying different preprocessing steps to different columns.
* Pipeline is used to streamline preprocessing and modeling steps.
* SimpleImputer is used to handle missing values.
* os.chdir changes the current working directory to 'C:\Users\SPURGE\Desktop\SCMA'.
* pd.read\_csv reads a CSV file named 'HR\_DataSet.csv' into a pandas DataFrame called data.
* categorical\_features lists the categorical columns that need encoding.
* numeric\_features generates a list of numerical columns by excluding the categorical columns and the target variable 'left'.
* SimpleImputer(strategy='median') is used to fill missing values in numerical columns with the median value of each column.
* Pipeline creates a sequence of preprocessing steps.
* SimpleImputer(strategy='constant', fill\_value='missing') fills missing values in categorical columns with the string 'missing'.
* OneHotEncoder(handle\_unknown='ignore') encodes categorical variables into a one-hot numerical array, ignoring unknown categories.
* ColumnTransformer applies different preprocessing pipelines to the respective numerical and categorical features.

In summary, the code cells shown in the image are focused on importing necessary libraries, loading the dataset, identifying feature types, and setting up preprocessing pipelines for both numerical and categorical data. This setup is essential for preparing the data before training machine learning models.



* X contains all features except the target variable 'left'.
* y contains the target variable 'left'.
* train\_test\_split splits the data into training (80%) and testing (20%) sets. The random\_state=123 ensures reproducibility of the split.
* Pipeline combines the preprocessing steps (defined earlier) with the logistic regression classifier.
* LogisticRegression(max\_iter=1000) initializes a logistic regression model with a maximum of 1000 iterations for convergence.
* logit.fit(X\_train, y\_train) trains the logistic regression model on the training data.
* logit.predict(X\_test) makes predictions on the test data.
* logit.predict\_proba(X\_test)[:, 1] provides the predicted probabilities of the positive class (i.e., the probability that the 'left' variable is 1).
* confusion\_matrix(y\_test, logit\_pred) calculates the confusion matrix for the test data predictions.
* **2136**: True Negatives (correctly predicted as not left)
* **155**: False Positives (incorrectly predicted as left)
* **478**: False Negatives (incorrectly predicted as not left)
* **231**: True Positives (correctly predicted as left)

This part of the codes demonstrates the end-to-end process of preparing data, training a machine learning model, making predictions, and evaluating the model's performance using a confusion matrix.

**Conclusion:**

In this study, the aim was to predict employee turnover using logistic regression and decision tree models. Both models were evaluated based on their performance metrics, including accuracy, sensitivity, specificity, and ROC AUC.

1. **Logistic Regression Model:**
   * **Accuracy:** 78.89%
   * **Sensitivity:** 92.37%
   * **Specificity:** 36.25%
   * **ROC AUC:** 0.8229

The logistic regression model demonstrated good overall accuracy and high sensitivity, indicating it is effective at identifying employees who will stay. However, it showed low specificity, meaning it struggled to accurately identify employees who will leave. The ROC AUC score confirmed the model's good discriminatory ability.

1. **Decision Tree Model:**
   * **Accuracy:** 96.73%
   * **Sensitivity:** 98.42%
   * **Specificity:** 91.39%
   * **ROC AUC:** 0.9490

The decision tree model outperformed logistic regression, providing higher accuracy, sensitivity, and specificity. The model's high ROC AUC score indicated superior predictive performance, making it a preferable choice for predicting employee turnover in this dataset.

In conclusion, while logistic regression is a robust and interpretable model, the decision tree provided better performance metrics in this analysis, making it the recommended model for predicting employee turnover.

**R CODES**

# Load necessary libraries

library(caret)

library(pROC)

library(rpart)

library(rpart.plot)

library(e1071)

library(dplyr)

library(ggplot2)

#setting the wd

setwd('C:\\Users\\SPURGE\\Desktop\\SCMA')

getwd()

# Load the dataset

data <- read.csv("HR\_DataSet.csv")

# Encode categorical variables

data$salary <- as.factor(data$salary)

data$Department <- as.factor(data$Department)

# Split the data into training and testing sets

set.seed(123)

trainIndex <- createDataPartition(data$left, p = 0.8, list = FALSE)

trainData <- data[trainIndex,]

testData <- data[-trainIndex,]

# Logistic Regression

logit\_model <- glm(left ~ ., data = trainData, family = binomial)

# Predict on the test data

logit\_pred <- predict(logit\_model, newdata = testData, type = "response")

logit\_pred\_class <- ifelse(logit\_pred > 0.5, 1, 0)

# Confusion Matrix

confusionMatrix(as.factor(logit\_pred\_class), as.factor(testData$left))

# ROC Curve

roc\_curve <- roc(testData$left, logit\_pred)

plot(roc\_curve)

auc(roc\_curve)

# Decision Tree

tree\_model <- rpart(left ~ ., data = trainData, method = "class")

rpart.plot(tree\_model)

# Predict on the test data

tree\_pred <- predict(tree\_model, newdata = testData, type = "class")

# Confusion Matrix

confusionMatrix(tree\_pred, as.factor(testData$left))

# Compare models

logit\_roc <- roc(testData$left, as.numeric(logit\_pred\_class))

tree\_roc <- roc(testData$left, as.numeric(as.character(tree\_pred)))

plot(logit\_roc, col = "blue")

plot(tree\_roc, add = TRUE, col = "red")

legend("bottomright", legend = c("Logistic Regression", "Decision Tree"), col = c("blue", "red"), lwd = 2)

**PYTHON CODES**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion\_matrix, roc\_curve, roc\_auc\_score

import matplotlib.pyplot as plt

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

import os

os.chdir('C:\\Users\\SPURGE\\Desktop\\SCMA')

# Load the dataset

data = pd.read\_csv('HR\_DataSet.csv')

# Encode categorical variables

categorical\_features = ['Department', 'salary']

numeric\_features = [col for col in data.columns if col not in categorical\_features + ['left']]

# Preprocessing for numerical data

numeric\_transformer = SimpleImputer(strategy='median')

# Preprocessing for categorical data

categorical\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='constant', fill\_value='missing')),

('onehot', OneHotEncoder(handle\_unknown='ignore'))])

# Combine preprocessors

preprocessor = ColumnTransformer(

transformers=[

('num', numeric\_transformer, numeric\_features),

('cat', categorical\_transformer, categorical\_features)])

# Split the data into training and testing sets

X = data.drop('left', axis=1)

y = data['left']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=123)

# Logistic Regression

logit = Pipeline(steps=[('preprocessor', preprocessor),

('classifier', LogisticRegression(max\_iter=1000))])

logit.fit(X\_train, y\_train)

logit\_pred = logit.predict(X\_test)

logit\_pred\_proba = logit.predict\_proba(X\_test)[:, 1]

# Confusion Matrix

print("Logistic Regression Confusion Matrix:")

print(confusion\_matrix(y\_test, logit\_pred))

# ROC Curve

fpr, tpr, \_ = roc\_curve(y\_test, logit\_pred\_proba)

roc\_auc = roc\_auc\_score(y\_test, logit\_pred\_proba)

plt.figure()

plt.plot(fpr, tpr, color='blue', lw=2, label='Logistic Regression (area = %0.2f)' % roc\_auc)

# Decision Tree

tree = Pipeline(steps=[('preprocessor', preprocessor),

('classifier', DecisionTreeClassifier(random\_state=123))])

tree.fit(X\_train, y\_train)

tree\_pred = tree.predict(X\_test)

tree\_pred\_proba = tree.predict\_proba(X\_test)[:, 1]

# Confusion Matrix

print("Decision Tree Confusion Matrix:")

print(confusion\_matrix(y\_test, tree\_pred))

# ROC Curve

fpr, tpr, \_ = roc\_curve(y\_test, tree\_pred\_proba)

roc\_auc = roc\_auc\_score(y\_test, tree\_pred\_proba)

plt.plot(fpr, tpr, color='red', lw=2, label='Decision Tree (area = %0.2f)' % roc\_auc)

# Plot ROC curve

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic')

plt.legend(loc="lower right")

plt.show()