National University of Computer & Emerging Sciences, Karachi
Computer Science Department

Summer 2023, Lab Manual 07

Course Code: AI-2002	**Course: Artificial Intelligence Lab**
Instructor(s):	**Sania Urooj**

Contents:

1) Supervised Learning

Supervised learning and unsupervised learning are two common types of machine learning techniques. It learning involves training a machine learning model on a labeled dataset, where the input data is paired with corresponding output data. The goal of the model is to learn a mapping function that can accurately predict the output for new, unseen input data. Supervised learning is commonly used for tasks such as classification, where the goal is to assign input data to one of several categories, or regression, where the goal is to predict a continuous value.

Types of Supervised Learning Algorithms

1. Linear Regression

Linear regression is a statistical modeling technique used to establish the relationship between a dependent variable and one or more independent variables. The goal of linear regression is to find the linear relationship between the input variables (predictors or independent variables) and the output variable (response or dependent variable).

In simple linear regression, there is only one independent variable, and the linear relationship is defined by a straight line. In multiple linear regression, there are multiple independent variables, and the linear relationship is defined by a hyperplane.

The equation of a linear regression model is expressed as:

y = b0 + b1x1 + b2x2 + ... + bn*xn

where y is the dependent variable, x1, x2, ..., xn are the independent variables, b0 is the intercept or bias term, and b1, b2, ..., bn are the coefficients or weights of the independent variables. The goal of linear regression is to estimate the values of the coefficients that best fit the observed data, so that the model can be used to predict the value of the dependent variable for new values of the independent variables.

Key assumptions of effective linear regression

Assumptions to be considered for success with linear-regression analysis:

- **For each variable:** Consider the number of valid cases, mean and standard deviation.
- **For each model:** Consider regression coefficients, correlation matrix, part and partial correlations, multiple R, R2, adjusted R2, change in R2, standard error of the estimate, analysis-of-variance table, predicted values and residuals. Also, consider 95-percent-confidence intervals for each regression coefficient, variance-covariance matrix, variance inflation factor, tolerance, Durbin-Watson test, distance measures (Mahalanobis, Cook and leverage values), DfBeta, DfFit, prediction intervals and case-wise diagnostic information.
- **Plots:** Consider scatterplots, partial plots, histograms and normal probability plots.
- **Data:** Dependent and independent variables should be quantitative. Categorical variables, such as religion, major field of study or region of residence, need to be recoded to binary (dummy) variables or other types of contrast variables.
- **Other assumptions:** For each value of the independent variable, the distribution of the dependent variable must be normal. The variance of the distribution of the dependent variable should be constant for all values of the independent variable. The relationship between the dependent variable and each independent variable should be linear and all observations should be independent.

$$w = \frac{N\sum(xy) - \sum x \sum y}{N\sum(x^2) - (\sum x)^2}$$

$$b = \frac{\sum y - w\sum x}{N}$$

Least-Squares Regression

Evaluation Metrics for Linear Regression The strength of any linear regression model can be assessed using various evaluation metrics. These evaluation metrics usually provide a measure of how well the observed outputs are being generated by the model. The most used metrics are, 1. Coefficient of Determination or R-Squared (R2) 2. Root Mean Squared Error (RSME)

Root Mean Squared Error: The Root Mean Squared Error is the square root of the variance of the residuals. It specifies the absolute fit of the model to the data i.e. how close the observed data points are to the predicted values.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

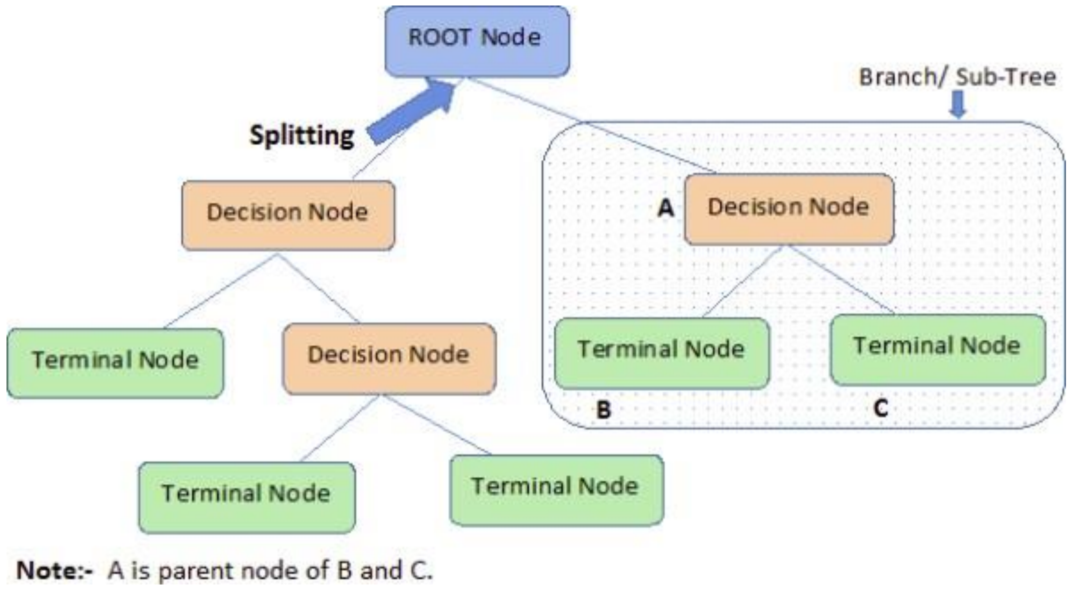
```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np


#load data

data = pd.read_csv('Salary_Data.csv')


data.head()

data.info()

plt.figure(figsize=(12,6))

sns.pairplot(data,x_vars=['YearsExperience'],y_vars=['Salary'],size=7,kind='scatter')

plt.xlabel('Years')

plt.ylabel('Salary')

plt.title('Salary Prediction')

plt.show()
```

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.7,random_state=100)

# Importing Linear Regression model from scikit learn

from sklearn.linear_model import LinearRegression

# Fitting the model

model = LinearRegression()

model.fit(X_train,y_train)


Predicting the Salary for the Test values

y_pred = model.predict(X_test)

# Importing metrics for the evaluation of the model

from sklearn.metrics import r2_score,mean_squared_error

# calculate Mean square error

mse = mean_squared_error(y_test,y_pred)
```

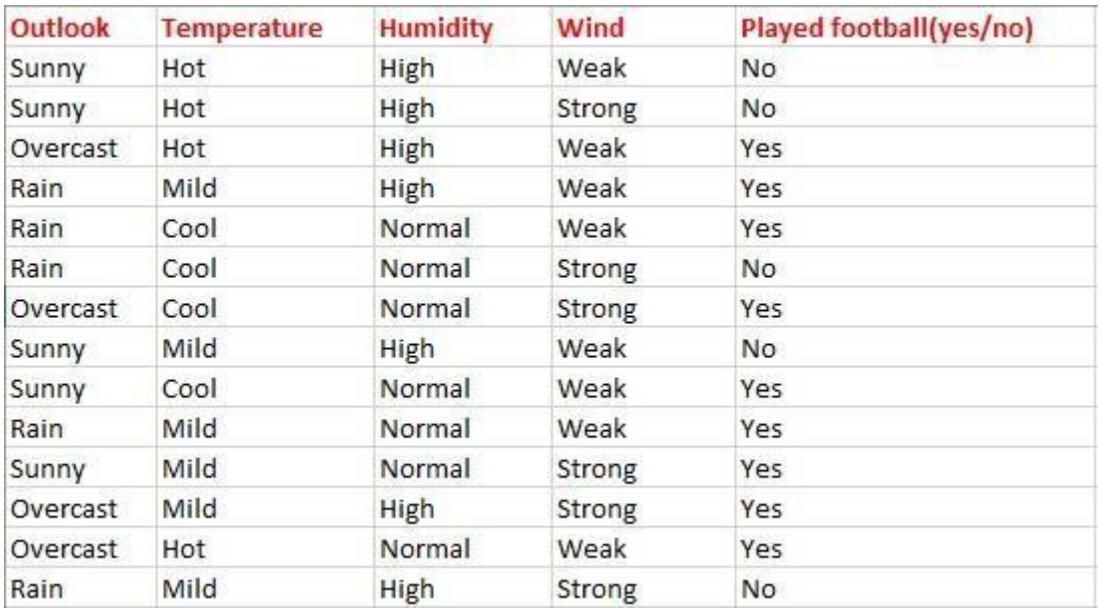
```
# Intecept and coeff of the line

print('Intercept of the model:',model.intercept_)

print('Coefficient of the line:',model.coef_)
```

1) **Decision Tree**

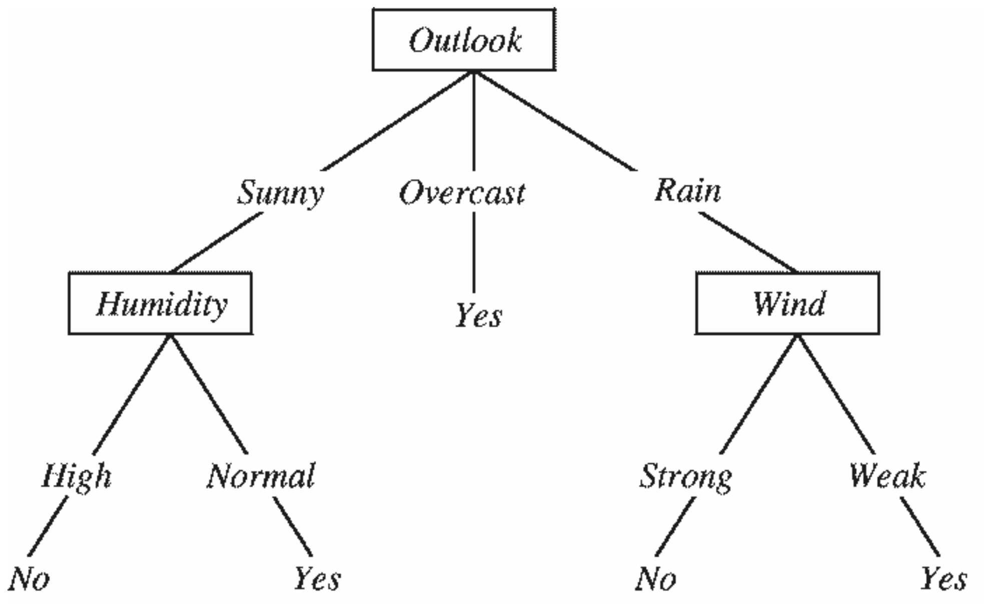
In general, Decision tree analysis is a predictive modeling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions trees are one of the most powerful algorithms that falls under the category of supervised algorithm. Below is an image explaining the basic structure of the decision tree. Every tree has a root node, where the inputs are passed through. This root node is further divided into sets of decision nodes where results and observations are conditionally based. The process of dividing a single node into multiple nodes is called splitting. If a node doesn't split into further nodes, then it's called a leaf node, or terminal node. A subsection of a decision tree is called a branch or sub-tree (e.g. in the box in the image below).



Consider whether a dataset based on which we will determine whether to play football or not.



Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No



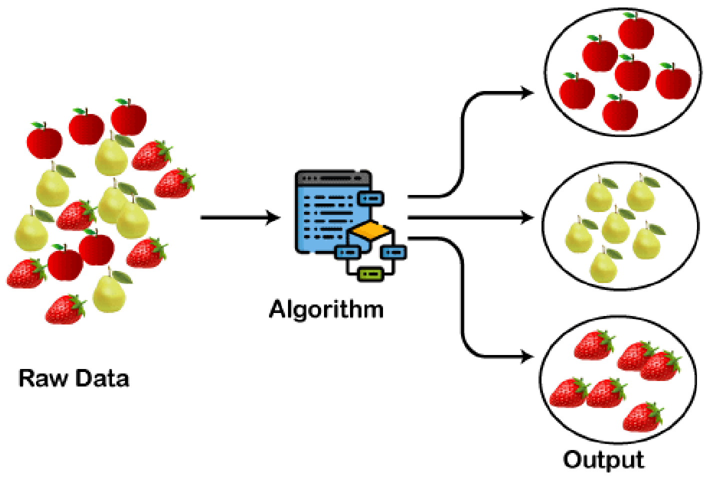
Key terminologies in Decision Tree:

- Information Gain
- Gini Index
- Entropy

```
from sklearn.datasets import load_iris

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score


# Load the dataset

iris = load_iris()


# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(

    iris.data, iris.target, test_size=0.2, random_state=42)


# Create a decision tree classifier

clf = DecisionTreeClassifier(random_state=42)

# Train the classifier on the training data

clf.fit(X_train, y_train)


# Use the trained classifier to predict the test data

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

# Generate a confusion matrix

cm = confusion_matrix(y_test, y_pred)
```

2) **Unsupervised Learning**

Unsupervised Learning is a machine learning technique in which the users do not need to supervise the model. Instead, it allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with the unlabeled data.



Clustering Algorithms

When choosing a clustering algorithm, you should consider whether the algorithm scales to your dataset. Datasets in machine learning can have millions of examples, but not all clustering algorithms scale efficiently. Many clustering algorithms work by computing the similarity between all pairs of examples.

Types of clustering

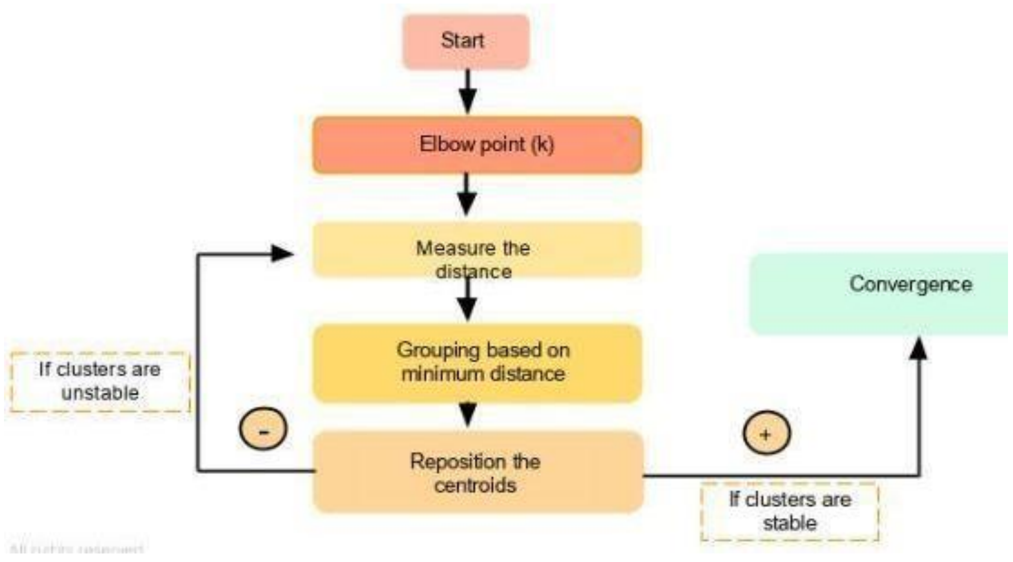
1. Partitioning Clustering
2. Density-Based Clustering
3. Distribution Model-Based Clustering
4. Hierarchical Clustering
5. Fuzzy Clustering

K-Means Clustering

K-Means clustering is an unsupervised learning algorithm. There is no labeled data for this clustering, unlike in supervised learning. K-Means performs the division of objects into clusters that share similarities and are dissimilar to the objects belonging to another cluster. The term ‘K’ is a number. You need to tell the system how many clusters you need to create. For example, K = 2 refers to two clusters. There is a way of finding out what is the best or optimum value of K for a given data.

How Does K-Means Clustering Work:

The flowchart below shows how k-means clustering works:



The goal of the K-Means algorithm is to find clusters in the given input data. There are a couple of ways to accomplish this. We can use the trial-and-error method by specifying the value of K (e.g., 3,4, 5). As we progress, we keep changing the value until we get the best clusters.

Another method is to use the Elbow technique to determine the value of K. Once we get the K's value, the system will assign that many centroids randomly and measure the distance of each of the data points from these centroids. Accordingly, it assigns those points to the corresponding centroid from which the distance is minimum. So each data point will be assigned to the centroid, which is closest to it. Thereby we have a K number of initial clusters. For the newly formed clusters, it calculates the new centroid position. The position of the centroid moves compared to the randomly allocated one.

Once again, the distance of each point is measured from this new centroid point. If required, the data points are relocated to the new centroids, and the mean position or the new centroid is calculated once again.

K-Means Clustering Algorithm

Let's say we have x1, x2, x3……… x(n) as our inputs, and we want to split this into K clusters.

The steps to form clusters are:

Step 1: Choose K random points as cluster centers called centroids.

Step 2: Assign each x(i) to the closest cluster by implementing Euclidean distance (i.e., calculating its distance to each centroid)

Step 3: Identify new centroids by taking the average of the assigned points.

Step 4: Keep repeating step 2 and step 3 until convergence is achieved Let's take a detailed look at it at each of these steps.

What are the Uses of Clustering?

Clustering has a myriad of uses in a variety of industries. Some common applications for clustering include the following:

- market segmentation
- social network analysis
- search result grouping
- medical imaging
- image segmentation
- anomaly detection

```
from sklearn.cluster import KMeans

import numpy as np

import matplotlib.pyplot as plt



# Generate some sample data

X = np.array([[1, 2], [1.5, 1.8], [5, 8], [8, 8], [1, 0.6], [9, 11]])

# Initialize k-means clustering algorithm with 2 clusters

kmeans = KMeans(n_clusters=2)

# Fit the data to the k-means model

kmeans.fit(X)

# Get the centroids and labels

centroids = kmeans.cluster_centers_

labels = kmeans.labels_

# Plot the results

colors = ['r', 'g', 'b', 'c', 'y', 'm']

for i in range(len(X)):

    plt.scatter(X[i][0], X[i][1], color=colors[labels[i]])



plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=200, linewidths=3, color='k')

plt.show()
```

3)<u>Evaluation Metrics</u>

1) Confusion Matrix:

Classification Models have multiple categorical outputs. Most error measures will calculate the total error in our model, but we cannot find individual instances of errors in our model. The model might misclassify some categories more than others, but we cannot see this using a standard accuracy measure.

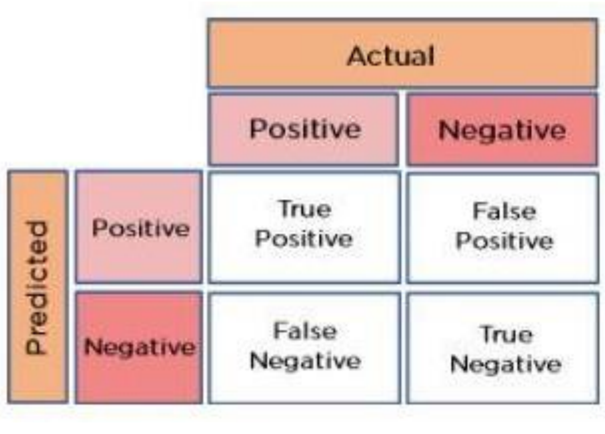
Furthermore, suppose there is a significant class imbalance in the given data. In that case, i.e., a class has more instances of data than the other classes, a model might predict the majority class for all cases and have a high accuracy score; when it is not predicting the minority classes. This is where confusion matrices are useful.

A confusion matrix presents a table layout of the different outcomes of the prediction and results of a classification problem and helps visualize its outcomes.

It plots a table of all the predicted and actual values of a classifier.

How to Create a 2x2 Confusion Matrix?

We can obtain four different combinations from the predicted and actual values of a classifier:



		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

- **True Positive**: The number of times our actual positive values are equal to the predicted positive. You predicted a positive value, and it is correct.
- **False Positive**: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.
- **True Negative:** The number of times our actual negative values are equal to predicted negative values. You predicted a negative value, and it is actually negative.

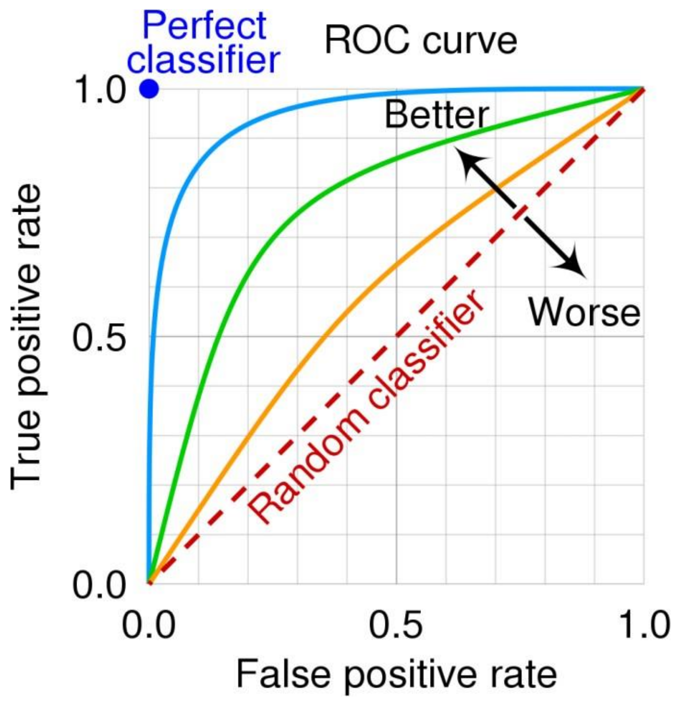
- **False Negative:** The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.

2) ROC

ROC stands for Receiver Operating Characteristic, which is a commonly used tool for evaluating the performance of binary classification algorithms. An ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) for a binary classifier at different classification thresholds. The TPR is the proportion of true positives (correctly predicted positive cases) out of all actual positives, while the FPR is the proportion of false positives (incorrectly predicted positive cases) out of all actual negatives.

An ideal classifier will have a TPR of 1 and an FPR of 0, which would be represented by a point in the top left corner of the plot. A completely random classifier would have a diagonal line from the bottom left to the top right corner of the plot, with an AUC of 0.5. A good classifier will have a curve that is closer to the top left corner, indicating a higher TPR and a lower FPR.

For example, if we had a binary classifier that was predicting whether an email was spam or not, and we plotted its ROC curve, it might look like a curve that rises steeply at the beginning (high TPR and low FPR), then levels off as the threshold increases, eventually approaching the diagonal line as the threshold approaches 1 (low TPR and high FPR). The area under the curve (AUC) would give us an overall measure of the classifier's performance, with higher values indicating better performance.



```
from sklearn.datasets import make_classification

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import roc_curve, roc_auc_score

from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt


# Generate sample data

X, y = make_classification(n_samples=1000, n_classes=2, random_state=42)


# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Train a Decision Tree Classifier model

dtc = DecisionTreeClassifier(random_state=42)

dtc.fit(X_train, y_train)


# Get predicted probabilities for the testing set

y_pred_prob = dtc.predict_proba(X_test)[:, 1]


# Calculate the false positive rate (fpr), true positive rate (tpr), and thresholds

fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)


# Calculate the AUC score

auc_score = roc_auc_score(y_test, y_pred_prob)


# Plot the ROC curve

plt.plot(fpr, tpr, label=f'AUC = {auc_score:.2f}')

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend()

plt.show()
```