T.C.
DOKUZ EYLUL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

2024 – 2025
SPRING SEMESTER

CME 3402
CONCEPTS OF PROGRAMMING LANGUAGES

ASSIGNMENT 1:
DECISION TREE CONSTRUCTION

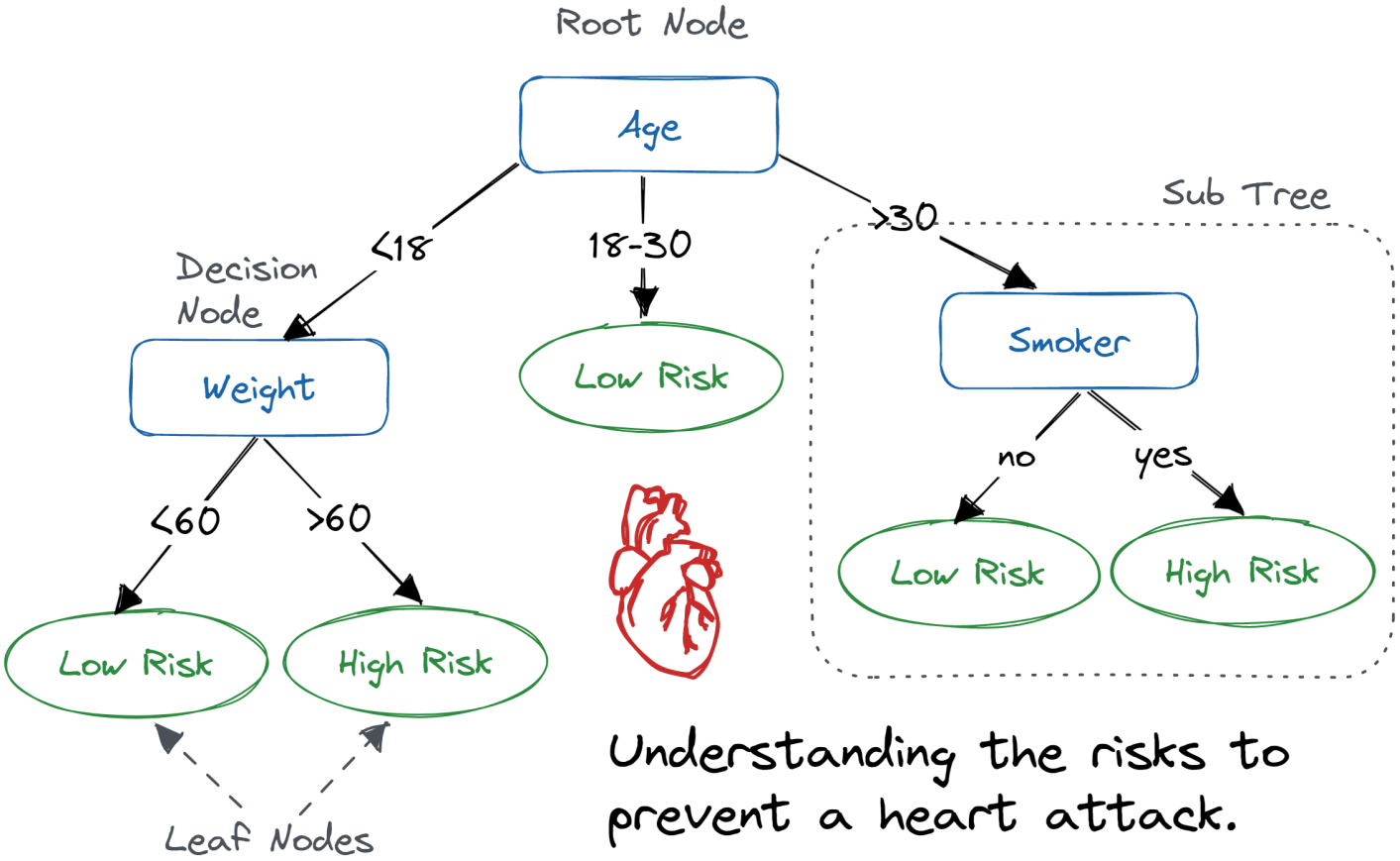
DUE DATE(S):
23:55 – 28.05.2025
23:55 – 04.06.2025
23:55 – 11.06.2025

Objective:
The purpose of this assignment is to help students understand how decision trees work in classification problems, using entropy and information gain to build the tree manually and implement it in code. The content of this assignment will be explained below with steps of algorithm shown and related examples.

A decision tree will be manually constructed step by step using the given dataset. At each step, the selected attribute and the reason for its selection (based on information gain calculations) must be clearly explained and justified.

Your assignment should work for csv files, text files that contain a table of data that is separated by commas, dots or other chosen characters. An example of such dataset and its processing will be given below.

What Are Decision Trees?:
Decision trees are intuitive models used in classification problems, where data is split into branches to reach a final decision. At each internal node, a specific attribute is used to divide the data, while the leaf nodes represent class labels. An example decision tree for heart attack risk is given below (image taken from website: https://www.datacamp.com/tutorial/decision-tree-classification-python).



Constructing a Decision Tree:
During the construction of a decision tree, information gain is calculated at each step to determine which attribute should be used for splitting. These calculations are based on the concept of entropy.

Implementation (Mandatory):
The decision tree algorithm must be implemented using 3 different languages from the given language groups below (The uploads will be done separately for 3 languages and required explanation will be given below):

First implementation should be done in Python.
Second implementation should be done in Java, C#, C or C++.
Third implementation should be done in Rust, Ruby or Go.

Student Expectations:
For this assignment, you are required to form groups of 2 or 3 students. Groups larger than 3 and individual student work will not be accepted. An upload will be open for students to form their groups in Sakai class webpage. You can see the requested format and an example upload given below.

Upload Format	**Example Upload**
<student_1_section>	1
<student_1_number>	2023510123
<student_1_name_and_surname>	Mehmet Yılmaz
<student_1_language>	Python
<student_2_section>	2
<student_2_number>	2024510124
<student_2_name_and_surname>	Fatih
<student_2_language>	Java
<student_1_section>	1
<student_1_number>	2025510125
<student_1_name_and_surname>	John Smith
<student_1_language>	Rust

Please order the students in student number, not in section, name or selected language. You must assign a selected language to each member of the group, who will also be responsible with that language upload to Sakai. If your group is made up of 2 students, Python should be common work of both students where second (Java, C#, C or C++) and third (Rust, Ruby or Go) implementation language should be assigned to students.

You are required to implement the decision tree algorithm from scratch using your own code, no external libraries or ready made functions should be used. You should use only the standard and main libraries or functions of the selected language.

You are also required to produce an output that visualizes the decision tree, text-based or terminal output is acceptable, however you can also produce an image as an output.

After providing an input dataset to your program, you should execute the decision tree making algorithm and show the calculation steps and results on console. After that showing and outputting decision tree that is created by this dataset.

After your program created the decision tree, it should be able to take inputs (supporting multiple inputs for different evaluations) from user for a prediction using the decision tree it created, repeating this more than once for different records.

You will be given three datasets as test cases for your assignment. These are breast_cancer.csv, contact_lenses.csv and weather.csv. Your assignment should be dataset agnostic and should work for other datasets as well. Your assignment should support and work for datasets that contain less than or equal to 10 columns of data. Support for larger than 10 columns of data is not necessary but encouraged.

For dataset processing, you should assume the last column is the output and the rest of columns are input for your decision tree. For weather.csv file, first 4 columns (outlook, temperature, humidity and windy) are inputs and last column (play) is the output decision of your decision tree algorithm.

For your evaluation, you will be asked to come to code control and show your work. During this code control, you maybe asked to explain and execute the code you have not personally written yourself because you are required to understand, explain and use all source codes your group has produced. This code control will be done in your lab sessions (possibly on your theoretical session hour as well) at the last 2 weeks of the semester.

Important Notes and Warnings:
The use of ready made libraries or functions for decision tree generation is strictly prohibited. (as an example; golearn for Go, ai4r for Ruby, smartcore for Rust and etc.).

Your code should work for all 3 given datasets and other possible datasets for your evaluation. Check your code for alternative datasets that you can find to make sure that it works for alternative datasets.

For the decision tree output (image or text) you are free to use main libraries of the selected language for visualization. However, try to make both text and image output, to make sure if one fails, other version is visible.

Your code must be clean, readable, and properly commented.

For every file or upload you are required to make, please make sure to order student numbers in ascending order.

Submission Format and Rules:
As it has been explained before, you are required to make 3 different uploads by different members of your group. For each upload, you are required to upload a single code file (of the language you have chosen) and a PDF file that contains a detailed report of your assignment that includes; step-by-step manual construction of the decision tree, including all information gain calculations, explanations and comments, sample output from the program (terminal screenshots of execution and etc.). The information related to all outputs are given and explained below:

First Upload Language: Python
First Upload Date: 23:55, 28.05.2025
First Upload Name Format:

GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_python.py
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_python_report.pdf

GROUP_01_2023510123_2024510124_2025510125_python.py
GROUP_01_2023510123_2024510124_2025510125_python_report.pdf

Second Upload Language: Java, C#, C or C++
Second Upload Date: 23:55, 04.06.2025
Second Upload Name Format:

GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_java.java
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_java_report.pdf
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_csharp.cs
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_csharp_report.pdf
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_c.c
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_c_report.pdf
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_cpp.cpp
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_cpp_report.pdf

GROUP_01_2023510123_2024510124_2025510125_java.java
GROUP_01_2023510123_2024510124_2025510125_java_report.pdf
GROUP_01_2023510123_2024510124_2025510125_csharp.cs
GROUP_01_2023510123_2024510124_2025510125_csharp_report.pdf
GROUP_01_2023510123_2024510124_2025510125_c.c
GROUP_01_2023510123_2024510124_2025510125_c_report.pdf
GROUP_01_2023510123_2024510124_2025510125_cpp.cpp
GROUP_01_2023510123_2024510124_2025510125_cpp_report.pdf

Second Upload Language: Rust, Ruby or Go
Second Upload Date: 23:55, 11.06.2025
Second Upload Name Format:

GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_rust.rs
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_rust_report.pdf
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_ruby.rb
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_ruby_report.pdf
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_go.go
GROUP_<group_number>_<student_number_1>_<student_number_2>_<student_number_3>_go_report.pdf

GROUP_01_2023510123_2024510124_2025510125_rust.rs
GROUP_01_2023510123_2024510124_2025510125_rust_report.pdf
GROUP_01_2023510123_2024510124_2025510125_ruby.rb
GROUP_01_2023510123_2024510124_2025510125_ruby_report.pdf
GROUP_01_2023510123_2024510124_2025510125_go.go
GROUP_01_2023510123_2024510124_2025510125_go_report.pdf

You are required to work in groups of 2 or 3 for this assignment, not working individually or on more than 3 person groups. You will be asked of your group structure during class or as a text assignment upload in Sakai. Do not forget to form your groups in requested time period.

Do not “zip” or “rar” requested files and upload them. It is not necessary and it makes it harder for us to evaluate your assignments. Please upload 3 files as they are without compressing them to a single file.

You are required to come to your lab session to show your code and execution. If you do not come to this code control, even if you made an upload, your assignment will be graded zero. The time table for code control will be done later and announced in Sakai later.

Your uploaded source codes will be checked for cheating and plagiarism. If cheating is detected, your entire assignment will be graded zero. If you or other students copy your code from an online source rather than writing it yourself, it will be considered as cheating as well.

Make sure that you upload your correct assignment to correct upload. If you accidentally upload another assignment (from another class for example) or to an incorrect upload (other section’s upload), it will be considered as not turned in and it will be graded as zero. Worst of all, you will only realize it after grades are published and it will too late to fix it.

If you have any questions or problems regarding this assignment, you can ask about it in our lab sessions. If you wish, you can also ask it in class forums or assignment page comments. If you send an email and if your question is answered, please share this information with other students to prevent asking of the same question again and again.

Your assignment will be open for one more day for late upload. This is done to allow students who may experience extreme problems (no Internet or electricity, computer crash or failure, etc.) and miss the deadline as a result. This extension will allow them to upload. If you are still unable to upload, send us an email informing your situation with your upload files in attachment and at the same time, try everything you can to upload your assignment.

Lastly, please do not forget to click “Submit” button after you upload your assignment files. If you do not, even though your files are uploaded to Sakai, you are labeled as “No Submission” and ignored when we try to download your assignments, making your uploaded files invisible to us, leading us to assume you did not make an assignment submission.

Example Assignment Operation:
Here you will see an example process of your assignment with dataset weather.csv. The weather.csv dataset contains 14 records with the following features, where the column names and possible values are given as below:

Outlook : Contains sunny, overcast or rain as values.
Temperature : Contains hot, mild or cool as values.
Humidity : Contains high or normal as values.
Wind : Contains TRUE or FALSE as values.
PlayTennis : Contains yes or no as values.

You can see the our weather.csv dataset below, with every row is a different record instance (just like in a database table) and every column is a different attribute. As you can see, the first row usually contains the names of the columns.

outlook,temperature,humidity,windy,play
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
rainy,mild,normal,FALSE,yes
sunny,mild,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes
rainy,mild,high,TRUE,no

Step 1: Entropy of the Entire Dataset
Entropy measures the impurity or uncertainty in a dataset. A high entropy means the classes are mixed; low entropy means they are more pure.

$$Entropy(S) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

S: The dataset
p_i: The probability of class i in the dataset
n: Total number of classes

Yes: 9, No: 5 (Total: 14)
Entropy(S) = -9/14 log2(9/14) - 5/14 log2(5/14) ≈ 0.940

Step 2: Information Gain for Each Feature
Information Gain measures the reduction in entropy achieved by splitting the dataset based on an attribute. The attribute with the highest information gain is chosen for the decision node.

$$InformationGain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v)$$

A: The attribute used to split the data
Values(A): All possible values of attribute A
S_v: Subset of S for which attribute A = v
|S_v| / |S|: Proportion of the subset in the full dataset

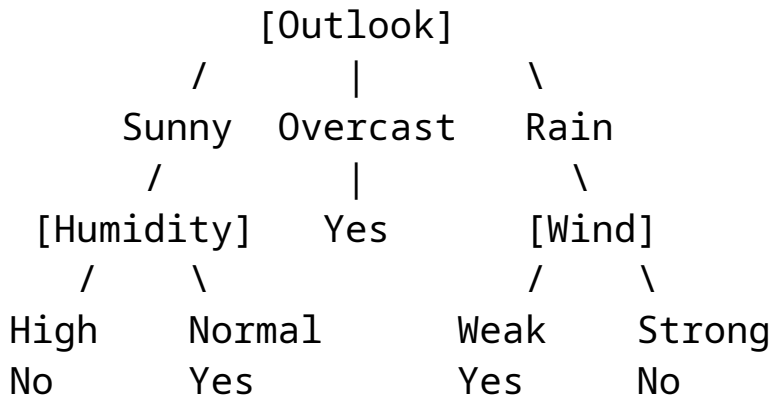
Outlook:
Sunny (5): 2 Yes, 3 No → Entropy ≈ 0.971
Overcast (4): 4 Yes → Entropy = 0
Rain (5): 3 Yes, 2 No → Entropy ≈ 0.971
Gain(S, Outlook) = 0.940 - (5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971) ≈ 0.246

Humidity: Gain ≈ 0.152
Temperature: Gain ≈ 0.029
Wind: Gain ≈ 0.048

Highest gain is for Outlook. Split based on Outlook.

Step 3: Building Subtrees
1. Overcast → All Yes → Leaf Node
2. Sunny Subset:
- Records: 5
- PlayTennis: 2 Yes, 3 No → Entropy ≈ 0.971
Check attributes within Sunny:
Humidity → Gain = 0.971 - (2/5 * 0 + 3/5 * 0) = 0.971 → Perfect split
Split Sunny on Humidity:
- High → No
- Normal → Yes
3. Rain Subset:
- Records: 5
- PlayTennis: 3 Yes, 2 No → Entropy ≈ 0.971
Check attributes within Rain:
Wind → Gain = 0.971 - (3/5 * 0 + 2/5 * 0) = 0.971 → Perfect split
Split Rain on Wind:
- Weak → Yes
- Strong → No

Final Decision Tree



If the user inputs:
Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong
→ The program should output: PlayTennis = No (based on the learned tree)