### **CS2\_Final project**

### **AI-Powered Movie Recommendation Classifier Using Binary Trees (C++)**

#### **Objective:**

In this project, you must implement a simple AI-powered movie recommendation system using binary trees. The system will predict whether a user would enjoy a particular movie based on its attributes (e.g., genre, rating, director popularity). The focus will be on applying key concepts such as **classes**, **recursion**, **binary trees**, **pointers**, **file I/O**, **STL containers**, and **exception handling**.

#### **Overview:**

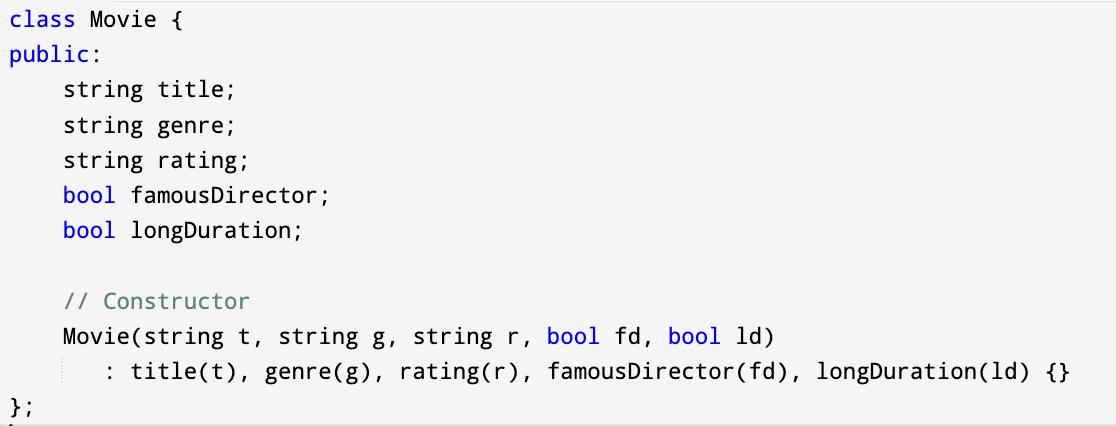
You have to:

1. Create a **binary decision tree** to make movie recommendations.
2. Implement **recursive algorithms** to traverse the tree.
3. Use **file I/O** to load a dataset of movies and their attributes.
4. Store movie data in **STL containers** (like vector or map).
5. Apply **exception handling** to handle invalid data inputs.
6. **Manually "train"** the decision tree based on predefined rules (instead of using machine learning).

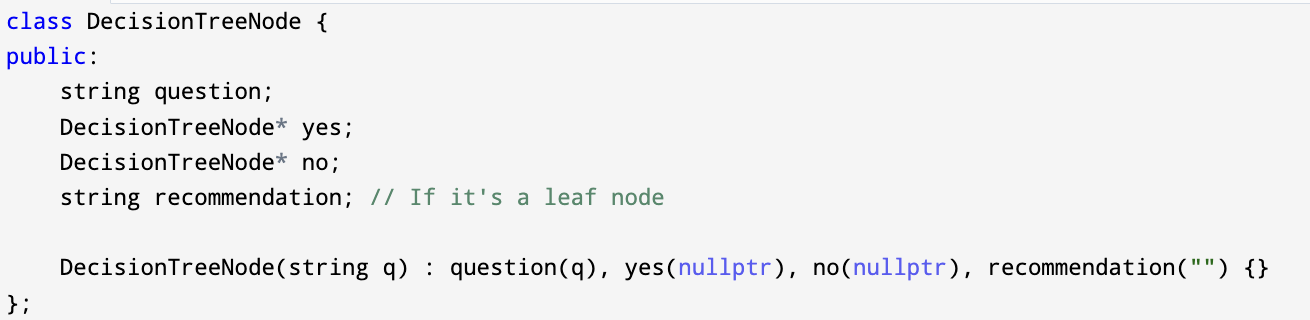
### **Detailed Project Breakdown**

#### **Part 1: Classes & Data Structures**

1. **Movie Class:**
   * Each movie has attributes such as **genre**, **age rating**, **director fame**, and **duration**.

Implement a Movie class with appropriate attributes.  
  


1. **DecisionTreeNode Class:**
   * Represents a node in the decision tree, with pointers to the **"yes"** and **"no"** child nodes.

Each node contains a question (e.g., "Is the genre Action?").  
  


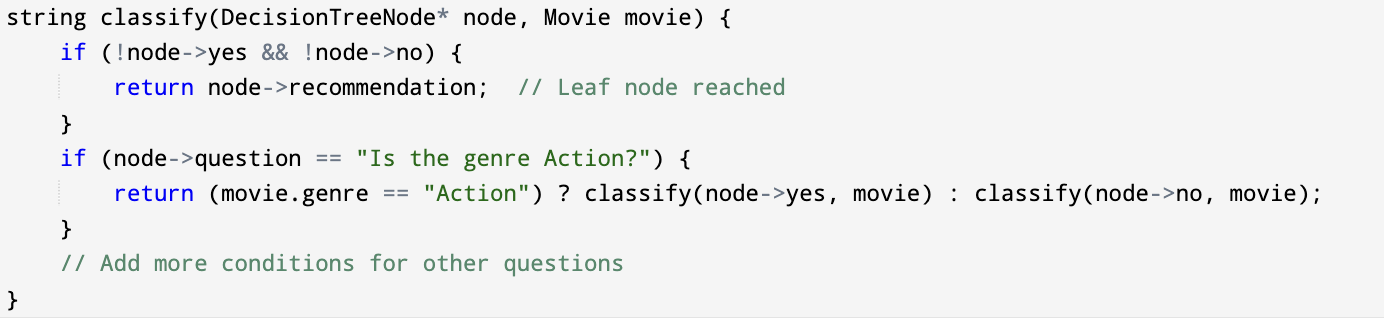
1. **DecisionTree Class:**
   * Manages the binary tree structure and provides methods for **classifying** movies and adding nodes.
   * The class will have methods like:
     + addNode(): Adds a node to the tree.
     + classify(): Recursively traverses the tree to make a recommendation.

#### **Part 2: Building the Decision Tree (Manual Training)**

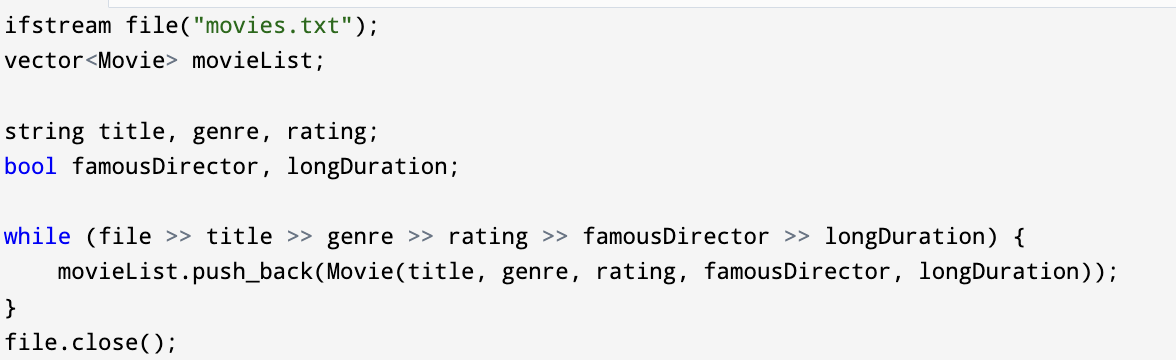
* Since we are not using ML algorithms, the decision tree will be **manually constructed** based on predefined logic. This is a simplification to focus on recursion and binary trees while still making the project interesting.
* Example decision tree logic:
  + If the **genre** is Action and the **rating** is PG-13 or lower, recommend the movie.
  + If the **director** is famous and the movie is **long**, recommend the movie.
  + If none of the conditions are satisfied, do not recommend the movie.

#### **Part 3: Recursion**

* The heart of the project is the **recursive traversal** of the binary tree.

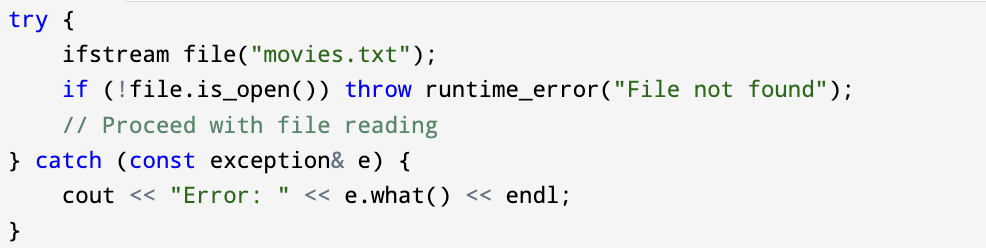
The classify() function will recursively traverse the tree based on movie attributes, until it reaches a **leaf node** (recommendation result).  
Example:  
  
****

#### **Part 4: File I/O**

The project will involve reading a **dataset of movies** from a file and storing the data in a **vector**.  
Example of file reading:  


#### **Part 5: Exception Handling**

Handle exceptions for invalid data inputs or file-related issues. For example, throw an error if the movie file is missing or incorrectly formatted:



#### **Part 6: STL Containers**

* Use **vectors** to store the movie data.
* Use **maps** for any additional attributes (e.g., a map to check if a director is famous).

### **Project Execution:**

1. **Movie Input File (movies.txt):**

Each line in the file represents a movie with attributes like this:  
  
"Inception Action PG-13 1 1"

* + where:
    - **Title**: Inception
    - **Genre**: Action
    - **Rating**: PG-13
    - **Famous Director**: Yes (1 for true, 0 for false)
    - **Long Duration**: Yes (1 for true, 0 for false)

1. **Classification Logic:**
   * Build the decision tree using simple if-else rules. The tree can be hardcoded for simplicity, but it should still use recursion to classify new movie data.
2. **Output:**
   * For each movie, print the prediction result (whether the user will like the movie or not).

#### **Data Source for Movies:**

* You can download small, sample movie datasets from public sources like:
  + **MovieLens:** A well-known dataset of movie preferences.
  + [**IMDB datasets**](https://www.imdb.com/interfaces/)**:** Provides access to movie data like genre, ratings, and more.
  + **Kaggle Datasets:** Offers various datasets related to movies.

You have multiple options for creating the dataset for this project. You can create your own dataset from public movie data sources, such as MovieLens, IMDB datasets, or other related datasets available on Kaggle as I listed above. I have also attached the Kaggle MovieLens dataset, which you can use after cleaning and keeping only the required attributes for the project. Additionally, to make things easier for you, I have attached a simpler dataset named 'movies\_sample.csv', which contains basic movie information that you can use directly for your implementation.

### **Project Workflow**

1. **Step 1:** Download and preprocess the movie dataset.
2. **Step 2:** Implement the Movie and DecisionTreeNode classes.
3. **Step 3:** Manually build the decision tree based on predefined logic.
4. **Step 4:** Implement recursive classification in the DecisionTree class.
5. **Step 5:** Read movies from a file and classify each using the decision tree.
6. **Step 6:** Handle exceptions for invalid inputs or missing data.
7. **Step 7:** Print out the results of each movie classification.

### **Grading Rubric (Total: 100 Points):**

* 25 Points: Correct implementation of classes (Movie, DecisionTreeNode, and DecisionTree).
* 25 Points: Recursive classification algorithm.
* 20 Points: File I/O for loading movies and printing results.
* 15 Points: Exception handling (e.g., invalid file, incorrect movie attributes).
* 15 Points: Use of STL containers (e.g., vector, map), and code style/documentation.

### **Training the Decision Tree:**

* In this project, instead of training using machine learning, **you manually "train"** the decision tree by creating **hardcoded decision rules** (e.g., based on the genre or rating). Each decision corresponds to a node in the tree, and the rules dictate how to build the tree.
* This approach simplifies the task while focusing on the core programming concepts like recursion, trees, and decision-making processes.

### **Output of the Decision Tree:**

The output of the project will display whether the user would enjoy a particular movie based on its attributes. For each movie, the system will traverse the decision tree and print the recommendation result, such as:

Example output in a C++ IDE:

| Movie: Inception - Recommended  Movie: Titanic - Not Recommended  Movie: Avatar - Recommended  Movie: The Godfather - Recommended  Movie: The Lion King - Not Recommended |
| --- |

This output will be shown in the console of the IDE, demonstrating the results of traversing the decision tree and providing movie recommendations based on the predefined logic.