Programming Environment

Programming Language:

C++ (Standard: C++17):

The core application, including data loading, preprocessing, the Random Forest machine learning model (decision trees, ensemble logic), prediction, recommendation engine, and the command-line interface (CLI), is implemented in C++. C++17 was chosen for its modern features, performance capabilities, and control over memory management, suitable for building a computational application from fundamental components.

External Software Packages/Libraries and Their Roles:

Standard C++ Library: Extensively used for core functionalities:

iostream: For console input and output (CLI interactions).

vector: For dynamic arrays to store datasets, features, labels, and collections of objects string,

sstream: For string manipulation and parsing CSV data.

fstream: For file input/output (loading the dataset, reading/writing history).

algorithm: For functions like std::sort, std::shuffle, std::min, std::max, std::iota, std::transform.

random: For random number generation

map: For counting class occurrences

cmath: For mathematical functions like std::sqrt.

limits: For std::numeric_limits

iomanip: For output formatting

chrono, ctime: For generating timestamps for the prediction history.

(No External Machine Learning Libraries for Core RF): Notably, for the Random Forest algorithm itself (decision tree construction, splitting logic, ensemble management), no external pre-built machine learning libraries were used in this C++ phase. The decision tree and random forest logic were implemented from scratch as per the project's C++ focus.

User's Guide

This guide demonstrates how a user interacts with the Command Line Interface (CLI) of the Behavioral Optimization & Mental Wellness System to achieve their requirements.

Pre-requisite:

The C++ application (depression_classification) has been compiled and is ready to run in a terminal/console environment. **The cleaned_student_data.csv file must be in the same directory as the executable, or its path correctly specified in the C++ code.**

Execution Steps:

Step 1: Compile the main.cpp file to get the C++ application

Navigate to you project dir (can be done by running the following command)

```
git clone https://github.com/DiwBhat/Depression-Classification.git
```

Then cd into the cpp_port derictory

```
cd cpp_port
```

Run the following command to get the application

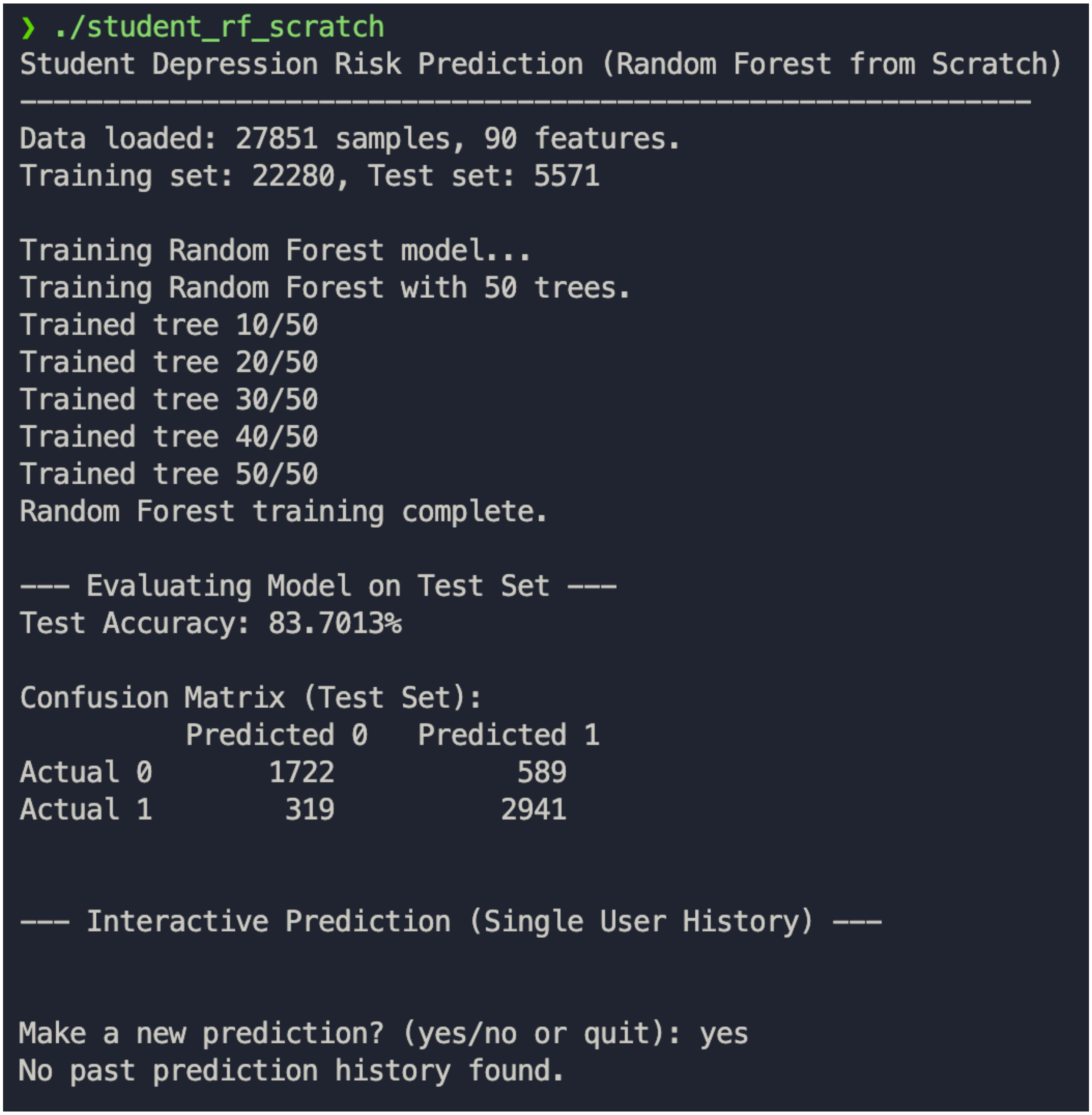
```
g++ -std=c++17 -Wall -Wextra -O2 -o depression_classification main.cpp
```

Step 2: Run the C++ application

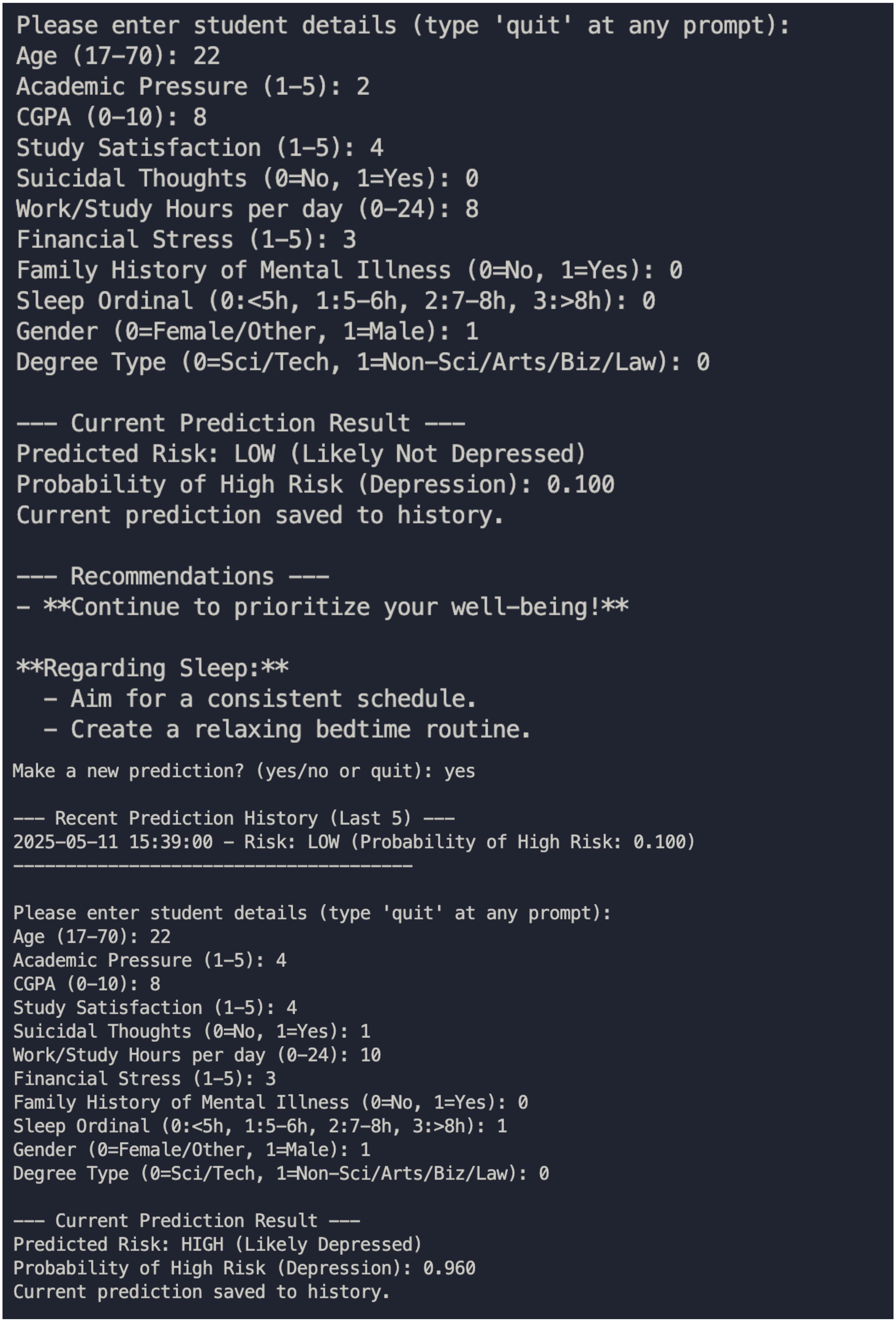
Run the following command to run the application

```
./depression_classification
```

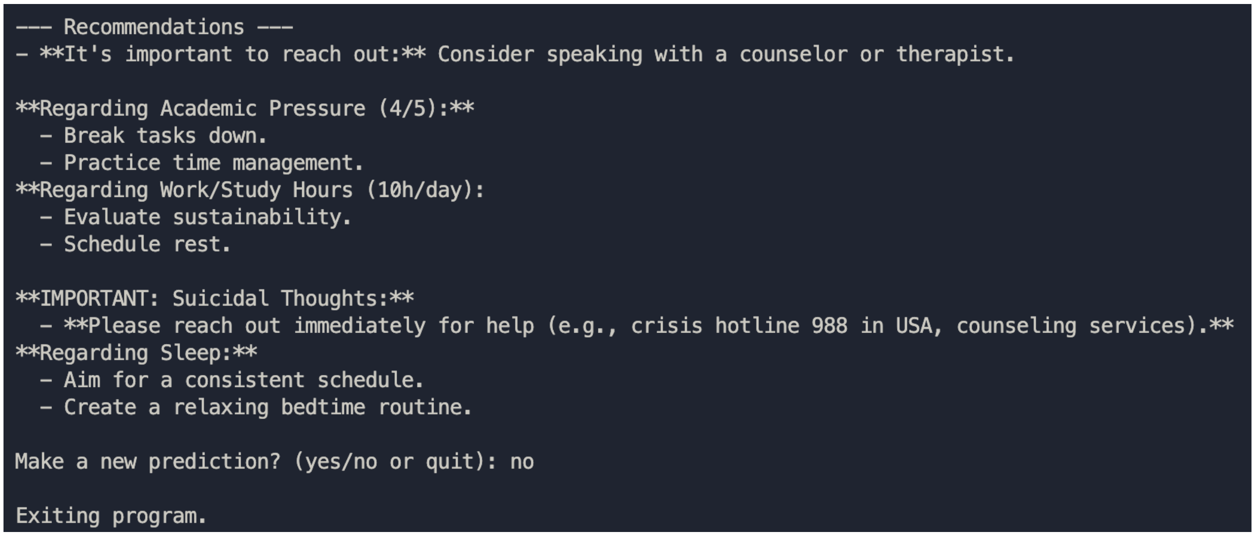
Screenshot of a sample workflow



```
❯ ./student_rf_scratch
Student Depression Risk Prediction (Random Forest from Scratch)
-------------------------------------------------------------
Data loaded: 27851 samples, 90 features.
Training set: 22280, Test set: 5571

Training Random Forest model...
Training Random Forest with 50 trees.
Trained tree 10/50
Trained tree 20/50
Trained tree 30/50
Trained tree 40/50
Trained tree 50/50
Random Forest training complete.

--- Evaluating Model on Test Set ---
Test Accuracy: 83.7013%

Confusion Matrix (Test Set):
          Predicted 0   Predicted 1
Actual 0       1722           589
Actual 1        319          2941


--- Interactive Prediction (Single User History) ---


Make a new prediction? (yes/no or quit): yes
No past prediction history found.
```



```
Please enter student details (type 'quit' at any prompt):
Age (17-70): 22
Academic Pressure (1-5): 2
CGPA (0-10): 8
Study Satisfaction (1-5): 4
Suicidal Thoughts (0=No, 1=Yes): 0
Work/Study Hours per day (0-24): 8
Financial Stress (1-5): 3
Family History of Mental Illness (0=No, 1=Yes): 0
Sleep Ordinal (0:<5h, 1:5-6h, 2:7-8h, 3:>8h): 0
Gender (0=Female/Other, 1=Male): 1
Degree Type (0=Sci/Tech, 1=Non-Sci/Arts/Biz/Law): 0

--- Current Prediction Result ---
Predicted Risk: LOW (Likely Not Depressed)
Probability of High Risk (Depression): 0.100
Current prediction saved to history.

--- Recommendations ---
- **Continue to prioritize your well-being!**

**Regarding Sleep:**
  - Aim for a consistent schedule.
  - Create a relaxing bedtime routine.
Make a new prediction? (yes/no or quit): yes

--- Recent Prediction History (Last 5) ---
2025-05-11 15:39:00 - Risk: LOW (Probability of High Risk: 0.100)
-------------------------------------

Please enter student details (type 'quit' at any prompt):
Age (17-70): 22
Academic Pressure (1-5): 4
CGPA (0-10): 8
Study Satisfaction (1-5): 4
Suicidal Thoughts (0=No, 1=Yes): 1
Work/Study Hours per day (0-24): 10
Financial Stress (1-5): 3
Family History of Mental Illness (0=No, 1=Yes): 0
Sleep Ordinal (0:<5h, 1:5-6h, 2:7-8h, 3:>8h): 1
Gender (0=Female/Other, 1=Male): 1
Degree Type (0=Sci/Tech, 1=Non-Sci/Arts/Biz/Law): 0

--- Current Prediction Result ---
Predicted Risk: HIGH (Likely Depressed)
Probability of High Risk (Depression): 0.960
Current prediction saved to history.
```



```
--- Recommendations ---
- **It's important to reach out:** Consider speaking with a counselor or therapist.

**Regarding Academic Pressure (4/5):**
  - Break tasks down.
  - Practice time management.
**Regarding Work/Study Hours (10h/day):
  - Evaluate sustainability.
  - Schedule rest.

**IMPORTANT: Suicidal Thoughts:**
  - **Please reach out immediately for help (e.g., crisis hotline 988 in USA, counseling services).**
**Regarding Sleep:**
  - Aim for a consistent schedule.
  - Create a relaxing bedtime routine.

Make a new prediction? (yes/no or quit): no

Exiting program.
```

Summary of Work Planned for Phase I

Complete Work (Phase I):

- Data Loading & Basic Preprocessing: C++ module to load the data.
- Decision Tree Implementation: Core logic for a decision tree classifier built from scratch, including Gini impurity calculation, best split finding, and recursive tree construction with parameters for max depth and min samples per leaf.
- Random Forest Implementation: Ensemble logic built from scratch, managing multiple decision trees, and feature subsampling.
- Model Training: Functionality to train the Random Forest model.
- Prediction Logic: Class prediction based on majority voting from trees in the forest.
- Probability estimation for the positive class based on the proportion of tree votes.
- Basic Evaluation: Calculation and display of test set accuracy and a confusion matrix.
- Simplified CLI: A command-line interface for user interaction.
- Prompts for simplified, more intuitive user inputs.
- Displays prediction results (risk category and probability).
- Rule-Based Recommendation Engine: Generates basic recommendations based on the predicted risk level and specific user input values
- Single-User History Tracking: Saves prediction timestamps, probabilities, and risk levels to a local text file (risk_history.txt) and displays recent history to the user.

Incomplete Work for Future Improvement (Beyond Phase I Scope):

- Advanced Hyperparameter Tuning in C++: While the Random Forest has tunable parameters, a systematic hyperparameter optimization framework (like GridSearchCV or Bayesian Optimization) was not implemented in C++. Parameters are currently set manually.
- Robust Input Validation in CLI: The CLI input validation is basic. More comprehensive checks for data types, ranges, and formats could be added.
- Advanced Feature Importance Calculation: A from-scratch implementation for calculating feature importance specific to this Random Forest was not developed.
- Cross-Validation within C++ Training: The current C++ training trains on the whole training split. Implementing k-fold cross-validation within the C++ training loop for more robust model parameter assessment was not part of this phase.
- More Sophisticated Recommendation Engine: The rule-based engine is simple; it could be expanded with more nuanced rules or draw from a larger knowledge base.
- Error Handling and Logging: While some basic error handling exists, a more comprehensive error handling and logging mechanism could be implemented.
- No GUI: The interface is CLI-only.

Planning for Phase II

Phase II could focus on enhancements and addressing some of the incomplete areas:

- GUI Development
- Enhanced Recommendation System
- Model Iteration and Improvement using Advanced Hyperparameter Tuning
- Feature Engineering & Selection
- Feedback Mechanism

Location for GitHub repository for code sharing

The repo can be accessed via the following url

https://github.com/DiwBhat/Depression-Classification/cpp_port