370 Assignment 3: Test Plan

Diwash Bhattarai

# A. DATASET DESCRIPTION

The primary dataset used for training, validating, and testing was compiled to capture various aspects of a student's life that could potentially correlate with mental well-being and depression.

**Source:** The dataset was a CSV file named “Student Depression Dataset”, sourced from Kaggle

**url:** https://www.kaggle.com/datasets/hopesb/student-depression-dataset

**INITIAL SIZE:** 27901 Rows with 18 attributes each

**SIZE AFTER PREPROCESSING:** 27851 Rows with 91 attributes each

**TARGET VARIABLE:** 'Depression', a binary indicator

**KEY FEATURES:** The dataset originally comprised features categorized as follows:

Demographics:

* id: Unique identifier (dropped during preprocessing).
* Gender: Male, Female.
* Age: Numerical age of the student.
* City: The city where the student resides/studies.

Academics:

* Profession: Student
* Degree: The student's current or highest completed academic degree.
* CGPA: Cumulative Grade Point Average.
* Academic Pressure: Self-reported level (e.g., 1-5).
* Study Satisfaction: Self-reported level (e.g., 1-5).
* Work/Study Hours: Average hours spent per day.

Well-being & Lifestyle:

* Work Pressure: (This column was found to have constant values and was dropped).
* Job Satisfaction: (This column was found to have constant values and was dropped).
* Sleep Duration: Categorical (e.g., "Less than 5 hours", "5-6 hours").
* Dietary Habits: Categorical (e.g., "Healthy", "Moderate", "Unhealthy").

Mental Health Indicators:

* Have you ever had suicidal thoughts?: Yes/No.
* Family History of Mental Illness: Yes/No.

Socioeconomic Factors:

* Financial Stress: Self-reported level (e.g., 1-5).

**DATA QUALITY AND PREPROCESSING:**

Initial Cleaning:

* Dropped uninformative columns
* Handled junk values identified in the city column
* Outlier Treatment

Feature Engineering:

* Suicidal\_Thoughts, Family History of Mental Illness: Converted from Yes/No to binary (1/0).
* Sleep\_Ordinal: Created from Sleep Duration categorical values.
* Total\_Stress: Derived by summing Academic Pressure and Financial Stress.
* Region: The City column was mapped to broader geographical regions
* The Degree column was generalized into Degree\_Level, Degree\_Field, Degree\_Type.

Encoding:

* Categorical features were one-hot encoded

# B. Unit + Component Testing

## *UT = Unit Testing*

*CT = Component Testing*

## Test Case ID: UT\_001

***Class::Function:***

DecisionTree::calculate\_gini\_impurity()

***Input****:*

* labels = {0, 0, 1, 1}
* indices = {0, 1, 2, 3}

***Scenario****:* Calculate Gini impurity for a perfectly balanced two-class node.

***Expected Outcome:*** Gini Impurity = 1.0 - ( (2/4)^2 + (2/4)^2 ) = 1.0 - (0.25 + 0.25) = 0.5

***Assertion Method:***

Check if the returned double is approximately equal to 0.5 (within a small epsilon).

***Status****: PASS*

## Test Case ID: UT\_002

***Class::Function***: DecisionTree::calculate\_gini\_impurity()

***Input:***

* labels = {0, 0, 0, 0}
* indices = {0, 1, 2, 3}

***Scenario:*** Calculate Gini impurity for a pure node (all class 0).

***Expected Outcome:*** Gini Impurity = 1.0 - ( (4/4)^2 + (0/4)^2 ) = 1.0 - 1.0 = 0.0

***Assertion Method:*** Check if the returned double is approximately equal to 0.0.

***Status****: PASS*

## Test Case ID: UT\_003

***Class::Function:*** DecisionTree::majority\_class()

***Input:***

* labels = {0, 1, 1, 0, 1}
* indices = {0, 1, 2, 3, 4}

***Scenario:*** Find majority class in a mixed set.

***Expected Outcome:*** 1

***Assertion Method:*** Check if the returned int is 1.

***Status****: PASS*

## Test Case ID: UT\_004

***Class::Function:*** DecisionTree::find\_best\_split()

***Input:***

* features = {{1.0, 10.0}, {2.0, 20.0}, {3.0, 10.0}, {4.0, 20.0}}
* labels = {0, 1, 0, 1}
* current\_indices = {0, 1, 2, 3}
* feature\_subset\_indices = {0, 1} (consider both features)
* params.min\_samples\_leaf = 1

***Scenario:*** Simple 2-feature dataset where splitting on feature 1 at value 15.0 (or feature 0 at 2.5) should yield a good Gini gain.

***Expected Outcome:***

SplitInfo.gini\_gain > 0. SplitInfo.feature\_index and SplitInfo.split\_value should correspond to the optimal split (e.g., feature 1, value around 15.0, or feature 0, value around 2.5). Left/Right indices should be correctly partitioned.

***Assertion Method:***

Verify best\_split.gini\_gain > 0. Manually calculate or inspect the expected optimal split and compare feature\_index, split\_value. Check sizes of left\_indices and right\_indices.

***Status****: PASS*

## Test Case ID: CT\_001

***Class::Function:*** RandomForest::train() and RandomForest::predict\_class()

***Input:***

* Use the same simple linearly separable dataset as UT\_DT\_006.
* rf\_params.num\_trees = 5
* rf\_params.tree\_params.max\_depth = 2, rf\_params.tree\_params.min\_samples\_leaf = 1
* rf\_params.bootstrap\_sample\_ratio = 1.0

***Scenario***: Train a small random forest and predict on training data.

***Expected Outcome:***

Predictions on training data should be highly accurate (likely 100% for this simple case, but bagging might introduce slight variations if a tree gets a very skewed bootstrap sample). Each tree should be trained.

***Assertion Method:***

Call train. Verify model.trees.size() is 5. For each sample in input features, call predict\_class and compare to the true label. Aim for high accuracy.

***Status****: PASS*

## Test Case ID: UT\_002

***Class::Function:*** RandomForest::predict\_probability\_class1()

***Input:***

* Train with UT\_RF\_001.
* sample = {1,1} (belongs to class 0)
* sample = {2,2} (belongs to class 1)

***Scenario:*** Check probability output.

***Expected Outcome:***

For {1,1}, probability of class 1 should be low (e.g., <= 0.2, ideally 0.0 if all trees agree). For {2,2}, probability of class 1 should be high (e.g., >= 0.8, ideally 1.0).

***Assertion Method:***

Call predict\_probability\_class1 and check if the returned double is within the expected range.

***Status****: PASS*

# C. System-wise Testing (Functional Requirements)

These tests evaluate the end-to-end functionality based on the CLI interaction.

## Test Case ID: SYS\_001

***Input:***

User inputs via CLI leading to a known "High Risk" prediction (e.g., low sleep, high academic pressure, high financial stress, suicidal thoughts=YES, low CGPA, etc. - define a specific set of values).

* Age: 20
* Academic Pressure: 5
* CGPA: 5.5
* Study Satisfaction: 1
* Suicidal Thoughts: 1 (Yes)
* Work/Study Hours: 10
* Financial Stress: 5
* Family History: 1 (Yes)
* Sleep Ordinal: 0 (<5 hours)
* Gender: 1 (Male)
* Degree Type: 0 (Science/Tech)

***Scenario:*** User enters data indicating high likelihood of depression. This is the first run (no history).

***Expected Outcome:***

* CLI displays "No past prediction history found."
* CLI displays "Predicted Risk: HIGH (Likely Depressed)".
* CLI displays a probability of High Risk > 0.5
* CLI displays recommendations appropriate for high risk, including urgent advice for suicidal thoughts, and tips related to high academic/financial stress, low sleep.
* risk\_history.txt is created/updated with one entry (timestamp, probability, "HIGH").

***Assertion Method:*** Observe CLI output. Inspect risk\_history.txt content.

***Status:*** PASS

## Test Case ID: SYS\_002

***Input:***

Precondition: risk\_history.txt contains the entry from SYS\_FN\_001.

User inputs via CLI leading to a known "Low Risk" prediction (e.g., good sleep, low academic/financial stress, no suicidal thoughts, high CGPA, etc.).

* Age: 22
* Academic Pressure: 1
* CGPA: 9.0
* Study Satisfaction: 5
* Suicidal Thoughts: 0 (No)
* Work/Study Hours: 6
* Financial Stress: 1
* Family History: 0 (No)
* Sleep Ordinal: 2 (7-8 hours)
* Gender: 0 (Female)
* Degree Type: 1 (Non-Science/Arts)

***Scenario:*** User enters data indicating low likelihood of depression. History exists.

***Expected Outcome:***

* CLI displays the previous "HIGH" risk entry from history.
* CLI displays "Predicted Risk: LOW (Likely Not Depressed)".
* CLI displays a probability of High Risk < 0.5 (e.g., < 0.3 based on input).
* CLI displays recommendations appropriate for low risk (e.g., maintain habits).
* risk\_history.txt is updated with a new second entry (timestamp, probability, "LOW").

***Assertion Method:*** Observe CLI output. Inspect risk\_history.txt content (should now have two entries).

***Status:*** Pass

Test Case ID: SYS\_003

***Input:***

Test with values at the boundaries and within typical ranges for each simplified CLI input field (e.g., Age=17, Age=40; Academic Pressure=1, Academic Pressure=3, Academic Pressure=5; CGPA=0, CGPA=6.9, CGPA=7.1, CGPA=10).

***Scenario:*** Test various combinations of inputs to see how predictions and recommendations change.

***Expected Outcome:***

* System handles all valid inputs within defined ranges without crashing.
* Predictions and recommendations appear logical and consistent with the input changes (e.g., increasing multiple stressors should generally increase risk probability).

***Assertion Method:*** Observe CLI output for sensible predictions and recommendations. Ensure no crashes or exceptions for valid inputs.

***Status:*** Pass

# D. Acceptance Testing for Non-Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | System Requirement (Verifiable) | Verification Criteria & Method | Status |
| SR4 | The machine learning modelmust achieve at least 80% accuracy in predicting depression risk when evaluated on a held-out, unseen test dataset. | Criteria: Test set accuracy ≥ 80.0%.  Method: After training the model on the training split, run the C++ program's evaluation part on the remaining 20% test split. | PASS |
| SR5 | The system, when running via the CLI, must process user inputs for a single prediction and display the risk level and initial recommendations within 5 seconds of the user submitting the final input. | Criteria: Average response time ≤ 5.0 seconds.  Method: Perform 10 interactive prediction sessions via the CLI. For each session, use a stopwatch to measure the time from hitting "Enter" after the last required input field to the moment the prediction and first line of recommendations appear. Calculate the average time. | PASS |
| SR6 | For the current CLI single-user version, no personal identifiable information (like name) is explicitly requested or stored beyond the session, other than the anonymized risk history. The risk history file (risk\_history.txt) will contain only timestamps, probabilities, and risk levels, without direct user identifiers. | Criteria: risk\_history.txt contains only timestamp, probability, and risk category string. No other user-identifying data is persistently stored by the application.   Method:  1. Run several prediction sessions.   2. Manually inspect the contents of risk\_history.txt.   3. Review the C++ source code to ensure no other files are written with user-specific session data (beyond the feature values passed to the model in memory). | PASS |