National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Aarthi Vijayaragavan
Student ID: X23438533

School of Computing
National College of Ireland

Supervisor: Dr. Abid Yaqoob

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Aarthi Vijayaragavan
Student ID:	X23438533
Programme:	MSc Data Analytics
Year:	2025-2026
Module:	MSc Research Project
Supervisor:	Dr. Abid Yaqoob
Submission Due Date:	28/01/2026
Project Title:	A Deep Learning-Powered Tool for Early Detection of Student Depression with SHAP Interpretability
Word Count:	1737
Page Count:	13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Aarthi Vijayaragavan
Date:	28/01/2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Aarthi Vijayaragavan
X23438533

1 Introduction

This Configuration Manual is a straightforward and very useful reference for the installation, configuration, and operation of the SHAP-Selected Multilayer Perceptron (MLP) Depression Risk Prediction System. User-friendliness is the main characteristic of this guide that its target group: researchers, academic staff, or technical evaluators will benefit the most as the environment is prepared, the application is launched, and its outputs are understood. It details all the required system specifications, software, and installation steps followed by the operation process. Moreover, it offers guidance for troubleshooting, enabling the users to not only deploy but also to interact with the Streamlit-based prediction system, even if they do not have previous experience in machine learning or development of applications.

2 System Configuration

The MLP Depression Prediction System's hardware and software specifications for installation and operation are covered in this section. Additionally, it gives a full list of Python libraries required for running the model, SHAP interpretability, and deployment using Streamlit.

2.1 Hardware Requirements

The system specifications utilized for the development, testing, and evaluation of the MLP Depression Risk Prediction System comprise the subsequent hardware setup:

- Processor: Intel® Core™ i7-8550U CPU @ 1.80GHz (up to 1.99GHz)
- Installed RAM: 16.0 GB (15.9 GB usable)
- System Type: 64-bit operating system, x64-based processor

This hardware setup is more than enough for training the machine-learning models, performing SHAP explainability computations, and running the Streamlit application with no performance drawbacks.

2.2 Operating System

The system was implemented on the following operating environment:

- Edition: Windows 11 Home
- Version: 24H2
- OS Build: 26100.4652

Windows 11 provides full compatibility with Python environments, SHAP libraries, and Streamlit deployment.

3 Software Requirements

In this part, the entire list of software components needed for the installation, execution, and deployment of the SHAP-Selected MLP Depression Risk Prediction System is presented. The setting was created to provide a reproducible, easy-to-setup, and compatible environment with contemporary machine-learning workflows.

3.1 Software Requirements

Software	Version	Purpose
Python	3.12.3	Required for running all model scripts and the Streamlit app
Jupyter Notebook	Latest (Anaconda / pip)	Used for coding, model training, and analysis
Streamlit	Latest stable	Used for deploying the interactive prediction application

3.2 Required Python Libraries

The following Python libraries (Figure 1) must be installed within the project virtual environment:

```
[ ]: # Core libraries
import os
import re
import time
import math
import joblib
import numpy as np
import pandas as pd
# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.graphics.mosaicplot import mosaic
# Utilities
from itertools import islice
import shap
# Scikit-learn: preprocessing, pipelines, and model selection
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
# Scikit-learn: models
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
# Scikit-learn: metrics
from sklearn.metrics import (accuracy_score,f1_score,classification_report,roc_auc_score,confusion_matrix,roc_curve,precision_recall_curve,auc)
# TensorFlow / Keras
from tensorflow.keras.models import Sequential, load_model, save_model
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

Figure 1: Libraries used in this project

4 Dataset and Project Files

The dataset utilized in the project, its original source, and the entire collection of implementation files that are necessary for the reproduction of the MLP Depression Prediction System are described in this section. Furthermore, the whole project is presented in a Jupyter Notebook file to guarantee easy execution and total reproduction of results.

4.1 Dataset Description

- Dataset Name: Student Depression Dataset
- Source: Kaggle (Figure 2)
- Dataset Link: https://www.kaggle.com/datasets/hopesb/student-depression-dataset
- Total Records: 27,920 anonymised student entries
- Feature Types:
    1. Demographic attributes (Age, Gender, City)
    2. Academic indicators (CGPA, study hours, academic pressure)
    3. Lifestyle factors (sleep duration, diet habits, work hours)
    4. Psychological indicators (stress level, suicidal thoughts, coping behaviour)

This dataset was selected due to its relevance, accessibility, and suitability for structured machine-learning analysis in mental-health prediction research (Hossain & Talukder 2020), (Rahman et al. 2023), (Gupta & Vishwakarma 2023).

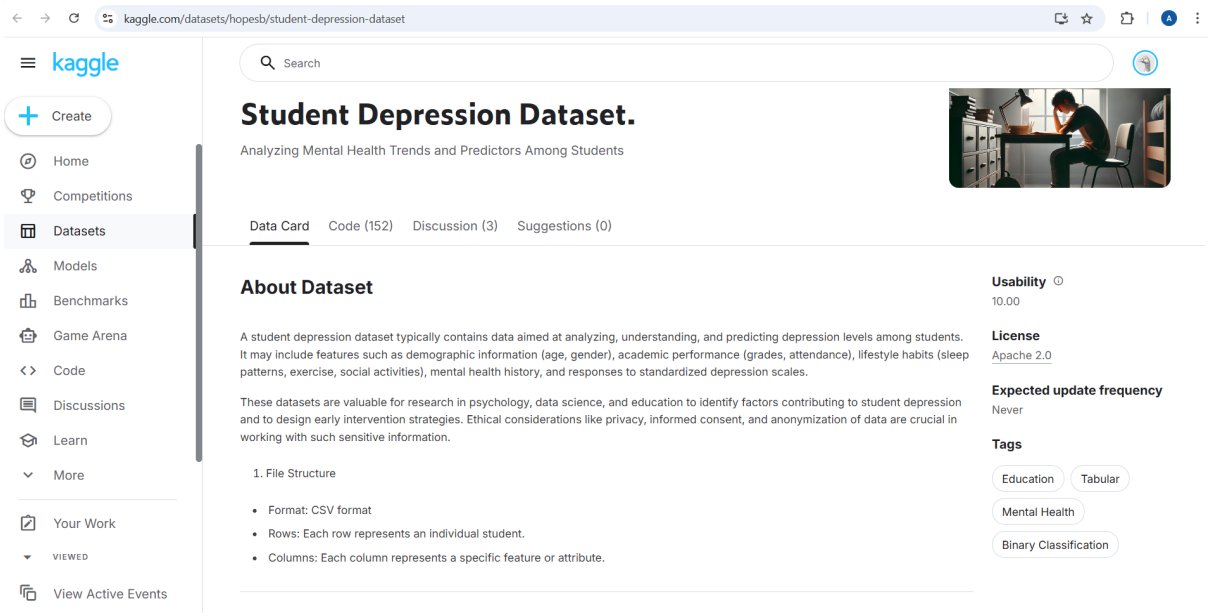


Figure 2: Data Source

4.2 Project Implementation Files

The entire implementation of the system that predicts the risk of depression was delivered in a simple manner through a small set of essential project files. All the components to reproduce the model training, SHAP analysis, and Streamlit deployment are in these files. Files Included

- Main Jupyter Notebook (.ipynb)

  It has the complete workflow, with data preprocessing, baseline model training, MLP developing, SHAP analyzing, evaluation and visualizations.

- SHAP-Selected MLP Model (mlp_shap_selected.h5)

  The last trained model that was used for the prediction in real-time by the Streamlit application.

- Streamlit Application Script (depressionapp.py)

  It puts into practice the user interface, the prediction logic, and the SHAP-based explanation module.

- SHAP Background Data (shap_background.csv)

  A small representative dataset was deployed for generating SHAP explanations during the deployment.

With these files, it is easy to run the training environment and the deployed application without the need for any additional configuration.

5 Workflow to Replicate the Experiment

This section outlines the complete workflow required to reproduce the depression-risk prediction system. Each step can be executed in sequence within the provided Jupyter Notebook, ensuring full reproducibility of the experiment.

5.1 Setting Up Your Environment

- It is very important that all necessary software and Python libraries are installed.
- Now, using the environment that you have set up, open Jupyter Notebook.
- Change your active directory to the directory where project files (.ipynb, model file, SHAP background file, and Streamlit script) are located (Peng 2011).

5.2 Exploratory Data Analysis (EDA)

The workflow begins with an initial examination of the dataset to understand its structure and characteristics.Included tasks:

- Inspect dataset size and variable types (Figure 3)
- Visualise distributions (histograms, boxplots)

- Identify missing values
- Examine correlations between academic, lifestyle, and mental-health factors (Rahman et al. 2023).
- Assess class imbalance in the target variable

```
[4]: # Basic info
print("Shape of dataset:", df.shape)
print("\nData types and missing values:")
print(df.info())
print("\nSummary statistics:")
print(df.describe())
```

```
Shape of dataset: (27901, 17)

Data types and missing values:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27901 entries, 0 to 27900
Data columns (total 17 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Gender                                 27901 non-null  object
 1   Age                                    27901 non-null  float64
 2   City                                   27901 non-null  object
 3   Profession                             27901 non-null  object
 4   Academic Pressure                      27901 non-null  float64
 5   Work Pressure                          27901 non-null  float64
 6   CGPA                                   27901 non-null  float64
 7   Study Satisfaction                     27901 non-null  float64
 8   Job Satisfaction                       27901 non-null  float64
 9   Sleep Duration                         27901 non-null  object
 10  Dietary Habits                         27901 non-null  object
 11  Degree                                 27901 non-null  object
 12  Have you ever had suicidal thoughts ?  27901 non-null  object
 13  Work/Study Hours                       27901 non-null  float64
 14  Financial Stress                       27898 non-null  float64
 15  Family History of Mental Illness       27901 non-null  object
 16  Depression                             27901 non-null  int64
dtypes: float64(8), int64(1), object(8)
memory usage: 3.6+ MB
```

Figure 3: Exploratory Data Analysis code

5.3 Data Preprocessing

The dataset undergoes a systematic preprocessing pipeline to be ready for model training. The entire preprocessing routine is already executed in the Jupyter Notebook that comes with the project. Processes performed

1. Handling missing values
   - For numerical features, mean imputation was used for replacement.
   - For categorical features, mode imputation was used for replacement.
2. Encoding of categorical data

   One-Hot encoding is done for all the categorical variables converting them to numerical form (Figure 4).

```python
# Drop duplicate column entries
numerical_cols = list(set(numerical_cols))

#  Clean categorical values
def group_rare_categories(df, col, threshold=50):
    value_counts = df[col].value_counts()
    rare_values = value_counts[value_counts < threshold].index
    df[col] = df[col].apply(lambda x: 'Other' if x in rare_values else x)
    return df

for col in categorical_cols:
    X = group_rare_categories(X, col)

#  Define pipelines
numeric_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', MinMaxScaler())
])

categorical_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(drop='first', sparse_output=False, handle_unknown='ignore'))
])

# Combine pipelines
preprocessor = ColumnTransformer([
    ('num', numeric_pipeline, numerical_cols),
    ('cat', categorical_pipeline, categorical_cols)
])

#  Apply preprocessing
X_processed = preprocessor.fit_transform(X)

# Inspect output
print("Processed shape:", X_processed.shape)
print("Feature names:", preprocessor.get_feature_names_out())
```

Figure 4: Data pipelines

3. Feature scaling
   - Min-Max Normalization is used for the numerical attributes scaling.
   - This ensures that all the values are in the range of 0–1 making the neural-network training stable.
4. Feature engineering
   - Development of an interaction feature: Work Pressure × Study Satisfaction
   - Combining of rare categories into an 'Other' class for noise reduction
5. Train-Test split
   - Dataset is divided into 80% for training and 20% for testing (Figure 5)
   - Stratified splitting is done to keep the original class distribution

```python
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_processed, y, test_size=0.2, random_state=42, stratify=y
)

print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
```

Figure 5: Data Splitting 80:20

6. Class balancing

   SMOTE is applied in case of baseline models to overcome imbalance when necessary

6 Baseline Model Training

Before the implementation of the deep learning approach, a performance benchmark is set up by training four conventional machine-learning models. The complete training/testing process and the corresponding code are provided in the project notebook (Figure 6). Models trained

- Logistic Regression (Baba & Bunji n.d.)(Alishiri et al. 2008)
- Decision Tree (Baba & Bunji n.d.)
- Naïve Bayes (Cruz et al. 2023)
- Random Forest (Bhatt & Bhalla 2024)

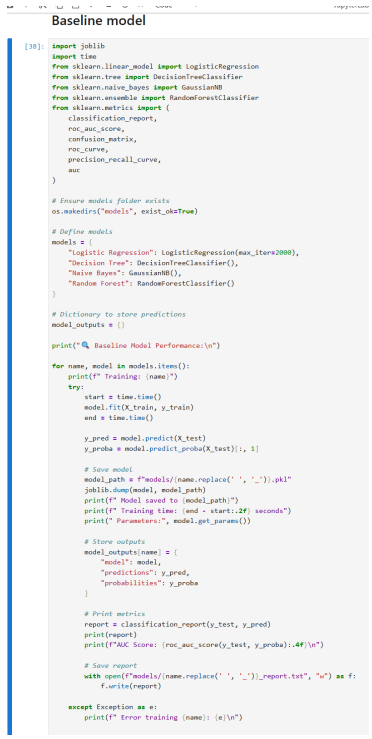


Figure 6: Baseline models code

Outputs Generated

- Classification metrics: Accuracy, Precision, Recall, F1-Score, AUC-ROC
- Confusion matrices for analyzing the patterns of errors
- ROC and Precision-Recall curves for comparing visually the performance

The baseline results become the starting point for quantifying the gains brought by the MLP and SHAP-selected MLP models.

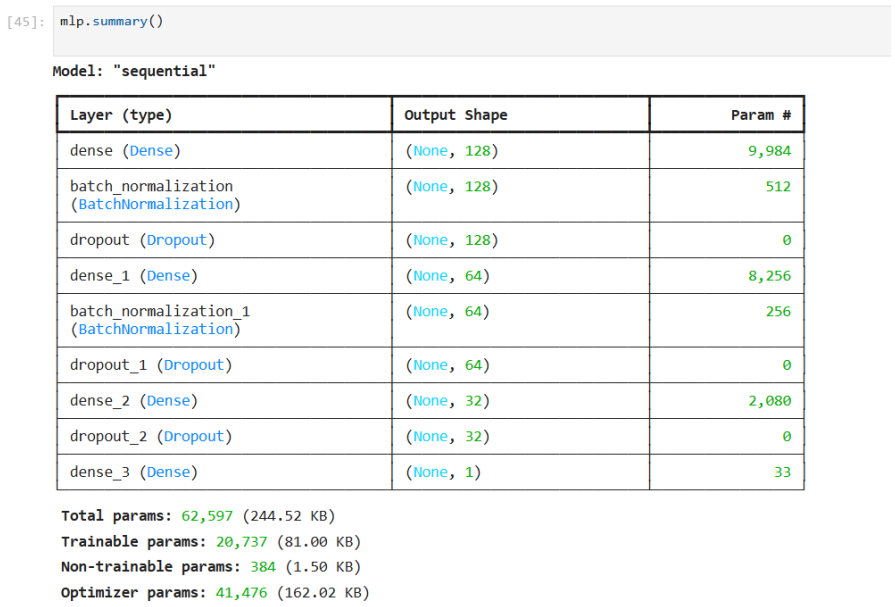
7 MLP Model Development

This phase consists of creating and training two instances of the Multilayer Perceptron (MLP) to determine if the feature dimensionality and SHAP-based feature selection have an effect.

1. Full-Feature MLP (77 Features)

   First, the complete set of 77 preprocessed features is used to train the model (Figure 7). This version of MLP gives the performance of a deep learning architecture without any feature reduction as a baseline. It will show:

   - How good an MLP can be in capturing complex non-linear relationships,
   - The effect of high-dimensional input on training time and stability,
   - If model behavior is affected by unnecessary or noisy features.



```
[45]: mlp.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	9,984
batch_normalization (BatchNormalization)	(None, 128)	512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
dropout_2 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 1)	33

Total params: 62,597 (244.52 KB)
Trainable params: 20,737 (81.00 KB)
Non-trainable params: 384 (1.50 KB)
Optimizer params: 41,476 (162.02 KB)

Figure 7: Full Feature MLP

2. SHAP-Selected MLP (10 Features)

   The second model only applies the 10 features that were found to be most influential through SHAP global importance analysis (Figure 8). This model aims to be the following:

   - Reduce noise and overfitting,
   - Improve training efficiency,
   - Provide clearer interpretability,
   - Keep (or improve) predictive quality with fewer inputs.

Figure 8: SHAP-Selected MLP

8 SHAP Interpretability Analysis

In this project, SHAP is applied to interpret the predictions made by the MLP model. It is great to see that this step is going to be of great help to everybody in knowing the features that contribute to the increase or decrease of a student's depression-risk score.

8.1 Outputs Produced by SHAP

The generated outputs from the notebook are:

- Global SHAP Summary Plot – depicts the most significant features in general.
- Mean SHAP Importance Plot – classifies features based on average effect.
- Local SHAP Waterfall Plots – gives reasons for particular predictions.

8.2 How SHAP Works in This Project

The SHAP explainer takes the trained MLP model, which then gives:

- Global explanations – Suicide thoughts, financial stress, academic pressure, sleeping, etc. are mentioned as the most significant factors.
- Local explanations – An illustration of the contribution of each feature to a user's prediction of being either high or low at risk.

This renders the model's decisions clear and comprehensible to users without any technical background.

9 Model Export

Following the training and evaluation stages, the last SHAP-chosen MLP model is exported for production. Generated files:

1. mlp_shap_selected.h5

   The remaining model trained in its final step, which serves the Streamlit-based application.

2. shap_background.csv

   A sample dataset for SHAP background that is to be used up to the same point during predictions to ensure consistency of explanations.

Both the files mentioned above need to be placed together in the same folder with the Streamlit app script.

10 Deployment Using Streamlit

10.1 Before running

Ensure the following files are in the same folder:

- depressionapp.py
- mlp_shap_selected.h5
- shap_background.csv

Update file paths in depressionapp.py if needed.

10.2 Running the application

1. Open Windows Terminal (Figure 9)
2. Navigate to project directory:

   cd path/to/your/project

3. Start Streamlit:

   streamlit run depressionapp.py

The app launches automatically in a browser window (e.g., http://localhost:8501) (Nath et al. 2025).

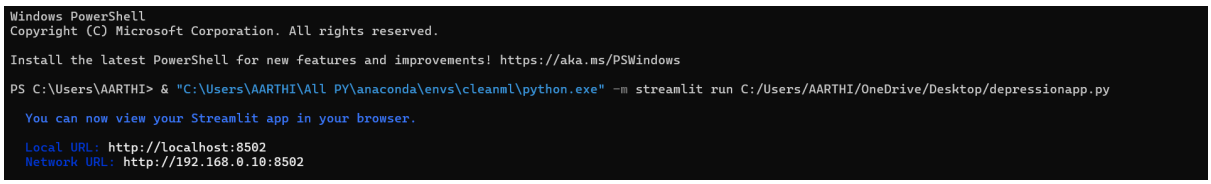
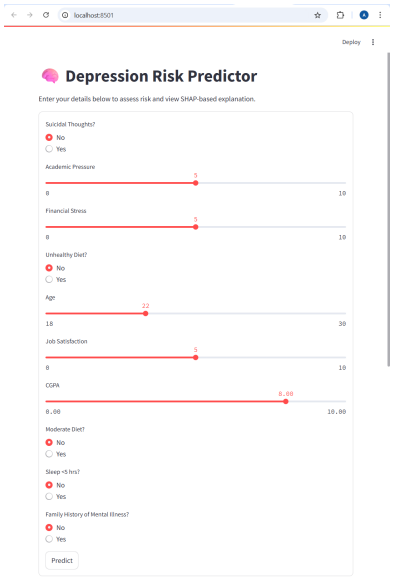


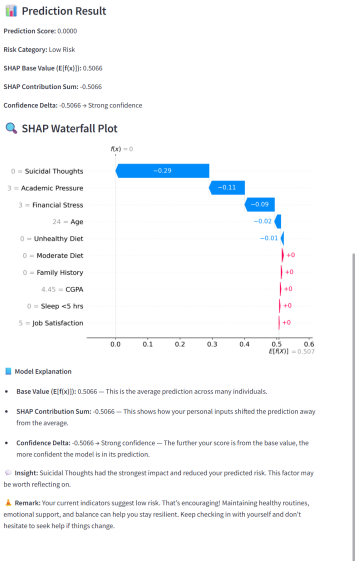
Figure 9: Windows Terminal

10.3 App Features

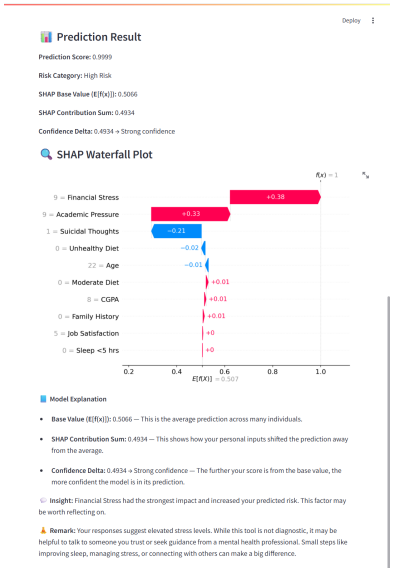
- Interactive user input form
- Instant depression-risk prediction
- Local SHAP explanation (waterfall plot)
- Global SHAP importance insight



(a) User Input Interface



(b) Low-Risk Prediction Output



(c) High-Risk Prediction Output

Figure 10: Streamlit-based deployment of the depression risk prediction system with SHAP explanations

11 Troubleshooting Guide

Issue	Possible Cause	Solution
Streamlit fails to launch	Missing or outdated libraries	Reinstall packages using `pip install -r requirements.txt` or install missing modules manually.
Model not loading	Incorrect model file path	Update the model path inside `depressionapp.py` (e.g., `mlp_shap_selected.h5`).
SHAP plot not displaying	Missing background dataset	Ensure `shap_background.csv` is in the same folder as the Streamlit app.
Notebook errors	Wrong Python version or missing dependencies	Use Python **3.12+**, reinstall dependencies, and restart the kernel.
Streamlit cannot find images or plots	Incorrect folder structure	Move all required `.png` SHAP images into the working directory.
App running locally but not opening	Port or firewall issues	Open `http://localhost:8501` manually or run with: `streamlit run app.py --server.port 8502`.

Table 1: Troubleshooting common issues in the Streamlit-based depression risk prediction system

12 Conclusion

This Configuration Manual provides details on the entire procedure to install, configure, and the SHAP-Selected MLP Depression Risk Prediction System. The users can securely repeat every single result and move the Streamlit-based prediction interface with complete SHAP interpretability as they proceed the steps provided which go from conditioning the environment and preparing the dataset to executing the model and deploying the application. The system is built around the concepts of access, transparency, and reproducibility thus researchers, teachers, and doctors can safely consider its application in the research or mental health screening workflows.

13 Validation & Reproducibility

The project guarantees total reproducibility that allows the system to work consistently across different realms and users. Users can perform the following via the rerun of the supplied Jupyter Notebook:

- Reproduce the entire experiment workflow starting from preprocessing to evaluation
- Retrain all the baseline models and MLP variants

- Regenerate metrics like accuracy, AUC, and also confusion matrices, ROC curves, and PR curves
- Create new SHAP visualizations (summary plots, bar charts, and waterfall explanations)
- Compare model performance across different runs or datasets
- Confirm that the outputs of the notebook are consistent with those of the Streamlit application

This reproducibility feature provides a guarantee that the outcomes are transparent, verifiable, and usable for further research,model auditing, or real-world deployment testing.

References

Alishiri, G. H. et al. (2008), 'Logistic regression models for predicting physical and mental health-related quality of life in rheumatoid arthritis patients', *Modern Rheumatology* **18**, 601–608.

Baba, A. & Bunji, K. (n.d.), Prediction of mental health problem using machine learning approaches. Unpublished manuscript.

Bhatt, S. & Bhalla, S. (2024), 'Analyzing mental health problems by linear regression and random forest model', *International Research Journal of Modern Engineering and Technology Science (IRJMETS)* **6**(7), 1760–1764.

Cruz, F. T., Coaquira Flores, E. E. & Condori Quispe, S. J. (2023), Prediction of depression level in university students through a naive bayes based machine learning model, *in* 'Proceedings of the Conference on Engineering and Informatics'.

Gupta, A. & Vishwakarma, B. (2023), 'Predicting student depression using machine learning', *International Journal of Innovative Science and Research Technology (IJISRT)* **8**(1), 876–882.

Hossain, M. T. & Talukder, M. A. R. (2020), 'Prediction of mental health problem using annual student health survey: Machine learning approach'. Unpublished.

Nath, M. D., Ahamed, M. K. U., Ahmed, O., Ahmed, T., Roy, S. & Uddin, M. N. (2025), 'Smart web interface for student mental health prediction using machine learning with blockchain technology', *Neuroscience Informatics* **5**, 100236.

Peng, R. D. (2011), 'Reproducible research in computational science', *Science* **334**(6060), 1226–1227.

Rahman, H. A. et al. (2023), 'Machine learning-based prediction of mental well-being using health behavior data from university students', *Bioengineering* **10**(5), 575.