National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Hariharan Vijayan
Student ID: x23404507

School of Computing
National College of Ireland

Supervisor: Furqan Rustam

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Hariharan Vijayan……

Student ID: x23404507……

Programme: Msc Data analytics…… **Year:** 2025-2026……..

Module: Research Practicum……

Lecturer: Furqan Rustam……

Submission Due Date: 11/DEC/2025……

Project Title: Integrating a hybrid model for a short term Mental Health prognosis

Word Count: 1348…… **Page Count: 9**……

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: Hariharan Vijayan……

Date: 11/Dec/2025……

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

Hariharan Vijayan
Student ID: x23404507

Overview

The software environment, installation procedure, file structure and execution procedure needed to execute

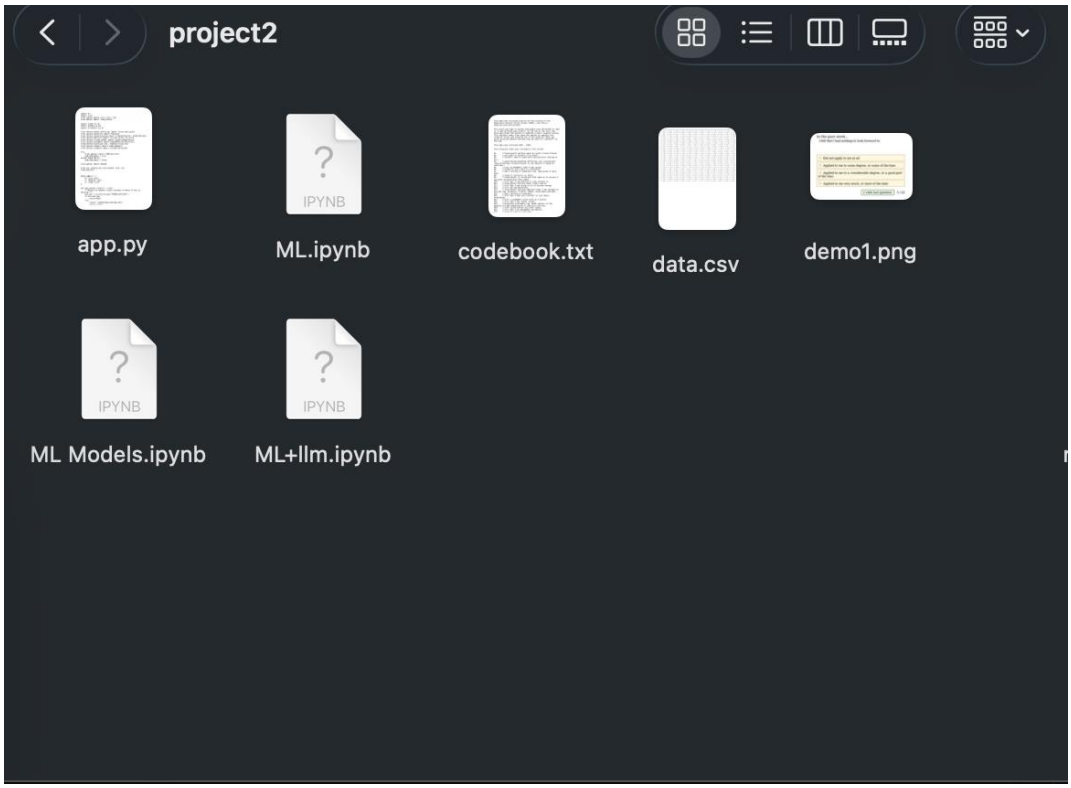
the hybrid predictive-reflective mental-health system that will be used in this research project are described in this configuration manual. It consists of a system that integrates the inference brought by traditional supervised machine-learning models with a large-language-model reasoning layer and is served by a Streamlit web interface. This document aims to make sure that the project can be replicated successfully in a different machine with all the dependencies and execution steps well established.

System Requirements

It was written in Python 3.13.5 in the Anaconda distribution based on the macOS platform. Jupyter Notebook version 7.3.2 was used to code all the modelling, preprocessing, and scoring procedures with the interactive interface being developed using version 1.45.1 of Streamlit. These machine-learning models are based on NumPy 2.1.3 and Pandas 2.2.3 and scikit-learn 1.6.1 and XGBoost 3.1.2. The language-model components are based on the existing OpenAI SDK. The system must also have a minimum of 8GB of RAM but 16GB would be ideal given the fact that this system requires running the notebooks with more than one model and visualisation. The amount of free disk space required is around 2GB.

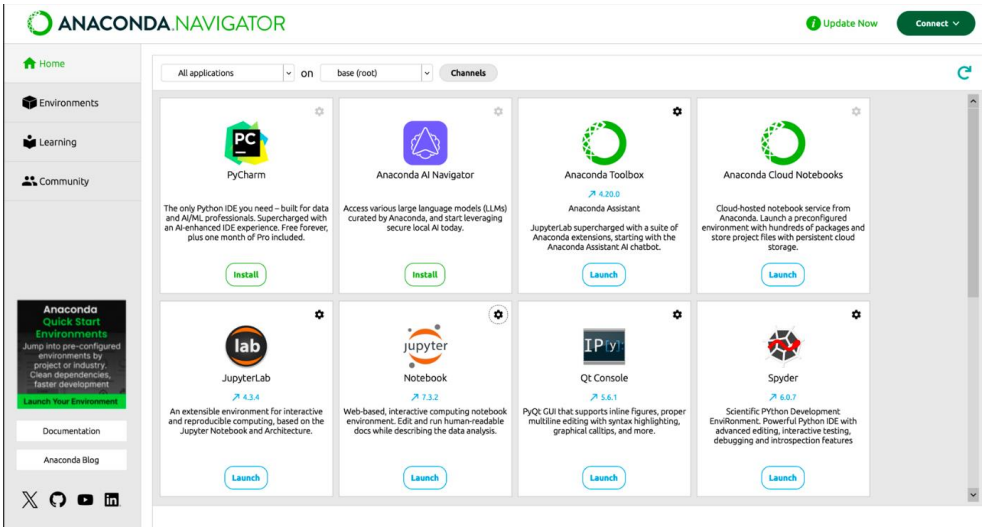
Project Directory Organization.

The complete project is all contained in the directory under Desktop/project/project2. The files in this folder are the Streamlit script app.py, the main machine-learning notebook ML.ipynb, the model comparison notebook ML Models.ipynb, the hybrid pipeline notebook ML+llm.ipynb and the revised model file model v2.ipynb. All the experiments are recorded in a single data file data.csv, and a codebook.txt file that explains the meaning of all items of the questionnaire and metadata. There is also a demonstration picture called demo1.png which is utilized in the user interface. Every notebook, the Streamlit application, and the dataset are located in this one directory in order to run everything effortlessly without any further setup.



Software requirements

Programming Language: Python 3.10
Integrating development environment: Jupyter Notebook (Anaconda)
Big Data Libraries: pandas, numpy,
Modelling library: scikit-learn, statsmodels, xgboost
• Visualisation Tools: matplotlib, seaborn



Installation Instructions

For anaconda We have to do to https://anaconda.org/ and install anaconda.

Anaconda navigataor appears and when you click on the Jupyter notebook it launches jupyter notebook where we can enter the codes

Implementation:

In the https://www.kaggle.com/datasets/lucasgreenwell/depression-anxiety-stress-scales-responses
I downloaded the dataset.

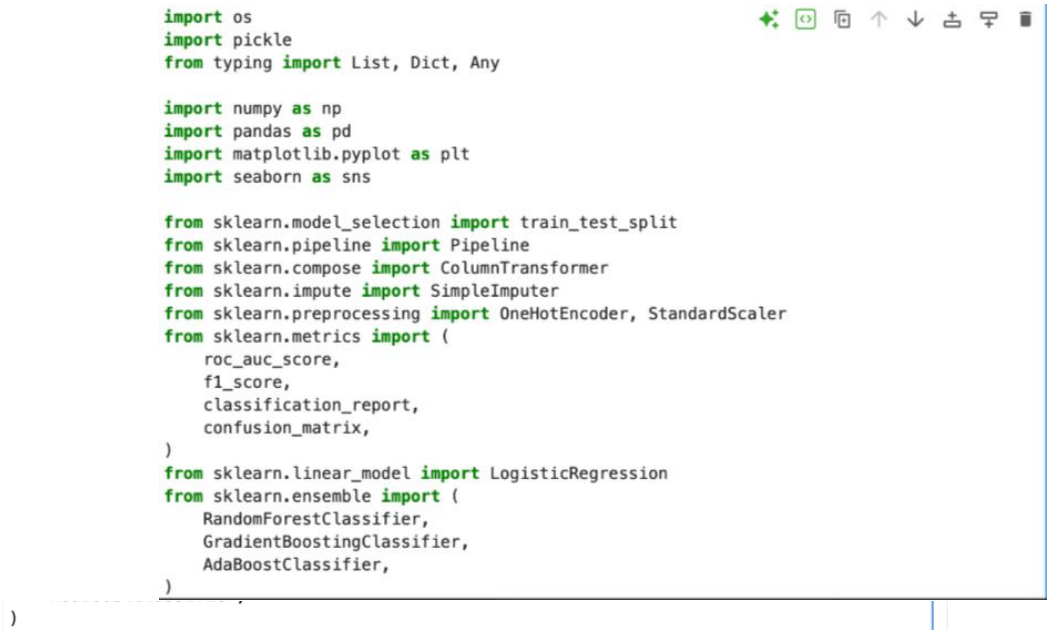
After downloading the dataset the data.csv looks like the below figure

Q1A	Q1I	Q1E	Q2A	Q2I	Q2E	Q3A	Q3I	Q3E	Q4A	Q4I	Q4E	Q5A	Q5I	Q5E	Q6A	Q6I	Q6E	Q7A	Q7I	Q7E	Q8A	Q8I	Q8E	Q9A	Q9I	Q9E	Q10A	Q10I
4	28	3890	4	25	2122	2	16	1944	4	8	2044	4	34	2153	4	33	2416	4	10	2818	4	13	2259	2	21	5541	1	38
4	2	8118	1	36	2890	2	35	4777	3	28	3090	4	10	5078	4	40	2790	3	18	3408	4	1	8342	3	37	916	2	32
3	7	5784	1	33	4373	4	41	3242	1	13	6470	4	11	3927	3	9	3704	1	17	4550	3	5	3021	2	32	5864	4	21
2	23	5081	3	11	6837	2	37	5521	1	27	4556	3	28	3269	3	26	3231	4	2	7138	2	19	3079	3	31	9650	3	17
2	36	3215	2	13	7731	3	5	4156	4	10	2802	4	2	5628	2	9	6522	4	34	2374	4	11	3054	4	7	2975	3	14
1	18	6116	1	28	3193	2	2	12542	1	8	6150	3	40	6428	1	4	17001	1	33	2944	3	7	8626	3	14	9639	2	20
1	20	4325	1	34	4009	2	38	3604	3	40	4826	4	22	2842	1	42	2342	3	6	9018	3	31	3717	3	39	7023	4	35
1	34	4796	1	9	2618	1	39	5823	1	12	6596	3	4	7635	2	31	7384	2	24	11570	1	33	2958	1	15	12300	1	5
4	4	3470	4	14	2139	3	1	11043	4	20	1829	3	3	5847	4	7	3529	4	17	1855	4	29	3000	4	31	6066	3	21
3	38	5187	2	28	2600	4	9	2015	1	7	3111	4	41	1712	4	22	1719	3	21	2049	4	11	2623	4	26	3853	4	12
3	38	3352	1	16	4322	2	28	3083	1	34	2204	3	41	1180	3	23	2218	1	29	2378	3	12	2082	1	31	3917	1	5
3	37	4024	3	35	2116	2	18	3408	2	10	2283	4	23	2253	3	20	2015	1	30	1821	3	5	1593	4	6	5626	3	24
1	35	3055	1	24	1812	1	30	3266	1	15	1567	1	17	2851	2	6	2931	1	8	4374	3	11	1781	1	9	3696	3	39
1	28	2890	4	20	3910	1	26	3132	2	35	5132	3	12	4903	1	16	5970	1	40	2632	3	15	4409	2	5	8096	1	14
1	10	2901	1	1	9036	1	22	3775	1	34	2027	1	38	2714	1	3	3626	1	11	4496	1	27	3453	1	41	4619	1	21
3	17	2316	1	18	2450	2	11	3143	2	3	6494	2	34	3251	1	24	3178	1	5	4442	1	21	3403	2	35	6034	2	22
3	15	2789	1	3	4500	2	25	3900	2	16	3568	1	35	1682	3	37	1633	1	12	2105	3	4	2940	4	5	4487	1	32
3	17	2507	1	27	1334	2	38	1864	1	2	2950	2	6	1700	3	1	6535	1	5	1301	2	40	1197	2	12	6908	2	35
3	5	6729	2	4	4793	2	20	3328	1	16	2199	3	22	2513	1	21	2220	1	13	2489	2	32	2570	1	8	5790	2	18

Code setup:

Importing of configuration description of modules.

The next code block can be used to define all the external libraries and modules needed by the machine-learning configuration environment. The imports create data processing, model building, assessment, and maintenance dependencies. The configuration elements in the figure can be presented as following:



```
import os
import pickle
from typing import List, Dict, Any

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import (
    roc_auc_score,
    f1_score,
    classification_report,
    confusion_matrix,
)
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import (
    RandomForestClassifier,
    GradientBoostingClassifier,
    AdaBoostClassifier,
)
)
```

Scientific Handling and Data Computing Modules.

The main computational and tabular-data data structures of the system are given in numpy and pandas,And matplotlib.pyplot and seaborn are set to display statistical data and analysis of data.Scikit-Learn Framework Model Development.

The pipeline infrastructure of machine-learning is the result of this category of imports:

3.1 Workflow Automation and Data Partitioning.

The partitioning of data is made possible by traintestsplit, The Pipeline facilitates creation of end to end, reproducible preprocessing to model workflows.

3.2 Elements of Feature Transformation.

ColumnTransformer allows column-specifications.SimpleImputer, OneHotEncoder and StandardScaler offer standard methods in dealing with missing data, encoding nominal data and normalizing numerical data.

3.3 Model Evaluation Metrics

Queens Gambrinus The imported measures (rocaucscore, f1score, classificationreport, confusion_matrix) are needed to use in quantitative evaluation of classifier performance.

3.4 Algorithms of Machine-Learning.

LogisticRegression is provided as a default linear classifier,The ensemble-learning methods that are supported in this configuration are the RandomForestClassifier, GradientBoostingClassifier, and AdaBoostClassifier

Data loading :

```python
DATA_PATH = "/Users/hariharan/Desktop/Project/Dataset/archive/data.csv"

df_raw = pd.read_csv(DATA_PATH, sep=None, engine="python")
print("Raw shape:", df_raw.shape)
df_raw.head()

def get_question_columns(df: pd.DataFrame) -> List[str]:
    """
    Automatically detect DASS item response columns like Q1A, Q2A, ..., Q42A.
    """
    q_cols = []
    for i in range(1, 43):
        col_a = f"Q{i}A"
        if col_a in df.columns:
            q_cols.append(col_a)
    return q_cols


def compute_dass_scores(df: pd.DataFrame, q_cols: List[str]) -> pd.DataFrame:
    """
    Compute DASS depression, anxiety, and stress subscale scores from the 42 items.
    Assumes responses are numeric (e.g. 0–3 or 1–4).
    """

    def q(i: int) -> str:
        return f"Q{i}A"

    depression_items = [3, 5, 10, 13, 16, 17, 21, 24, 26, 31, 34, 37, 38, 42]
    anxiety_items = [2, 4, 7, 9, 15, 19, 20, 23, 25, 28, 30, 36, 40, 41]
    stress_items = [1, 6, 8, 11, 12, 14, 18, 22, 27, 29, 32, 33, 35, 39]

    all_needed = [q(i) for i in set(depression_items + anxiety_items + stress_items)]
    missing = [c for c in all_needed if c not in df.columns]
    if missing:
        raise ValueError(f"Missing DASS item columns: {missing}")

    df = df.copy()
    df["DASS_depression"] = df[[q(i) for i in depression_items]].sum(axis=1)
    df["DASS_anxiety"] = df[[q(i) for i in anxiety_items]].sum(axis=1)
```

The data-loading and scoring configuration to be applied to the DASS questionnaire dataset can be found in the code fragment that is depicted in the figure. The script will firstly define an absolute path to the input CSV file and read the dataset into a pandas DataFrame and the dimensions of the dataset will be printed to check the integrity. The purpose of the function getquestioncolumns() is to identify all the DASS item-response columns that are found in the data automatically by indexing through the expected item columns and checking their existence. The official DASS scoring protocol has been incorporated in the function, which calculates the scores on the DASS questionnaire; a direct relationship is made between each of the 42 items of the questionnaire and its respective subscale: Depression, Anxiety, or Stress. Prior to the computation of scores, the function checks the presence of all the necessary columns of items, and halts the operation with an explanatory error message when any of its items are not present. Once validated, a copy of the dataset is generated and new columns are calculated which are the aggregate responses of the respective items of each subscale and hence create the final DASSDepression, DASSAnxiety and DASS_Stress scores. This module guarantees a dependable, repeatable and standards-conformant pre-processing framework within the framework of the requirements of psychometric data management as well as IEEE documentation standards.

Operating the Engine of Machine-Learning.

The entire workflow of the modelling is implemented in the Jupyter environment. The Activation of the Conda environment followed by the opening of Jupyter Notebook opens the notebook called ML.ipynb. This file does the data importation, calculation of DASS subscales, as well as building of the three-level distress label, preprocessing via a ColumnTransformer and training of the supervised classifiers. An evaluation of the models with the F1-scores, ROC-AUC values, and confusion matrices are shown in the file ML Models.ipynb. The notebook ML+llm.ipynb combines the results of the numerical models to the language-model reasoning layer, which produces customised reflective queries and the short-term emotional perspective. These notebooks can be executed sequentially and all the predictive and reflective outputs were reported in the evaluation chapter.

The execution of the Streamlit Application.

The graphical interface is run by going to the project directory in the terminal and running the Streamlit command corresponding to the file app.py. After launching, the application enables the user to select a participant within the dataset, the scores of DASS are displayed and the risk category predicted are viewed and personalised reflective questions are generated using the language-model component. The user would be able to add free-text answers and the system would then give the qualitative emotional prediction of the coming seven to fourteen days. This interface is a reproduction of the entire hybrid workflow in a friendly and interactive format.

Configuration Notes

The only setting to be outside of the Python environment is the OpenAI API key of the reflective reasoning layer. When the Streamlit code depends on this key, the code either needs to be exported as part of the operating system environment variables, or it should be put in a special configuration file like a Streamlit secrets file. It does not require any other configuration files because the system is also configured to execute a fully integrated one project without any backward dependencies.

Troubleshooting Guidance

In case the system indicates missing modules, the problem is usually caused by the execution of the project not within the Conda environment where the appropriate version of the packages is available. A common cause of cases where Streamlit loads an empty interface is a process of refreshing the browser window or ensuring the command is being run in the right directory. In case Jupyter notebook is giving errors related to the use of the kernel, the IPython kernel of the environment can be installed to solve the problem. Any failure to produce language-model response is normally associated with incomplete or wrongly set API credentials as opposed to failure within the modelling pipeline.

Reproducibility

The models employed in this project were trained on stratified hold-out splits and thus, a replication of the experiment will give similar performance measures. The preprocessing pipeline was applied consistently through the use of a ColumnTransformer which allows the same treatment of both numerical and categorical features both in training and inference. Reproducibility is thus mostly determined by the use of the same dataset (data.csv) and programs as identified in this manual.