National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Mohammad Nihaal Naazim
Student ID: 23211717

School of Computing
National College of Ireland

Supervisor: Dr. Anu Sahani

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Mohammad Nihaal Naazim
Student ID:	23211717
Programme:	MSc in Data Analytics **Year:** 2024
Module:	Research Project
Lecturer:	Dr. Anu Sahani
Submission Due Date:	12th December 2024
Project Title:	A Deep Learning Framework for Stress and Depression Detection
Word Count:	**638** **Page Count: 8**

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: Mohammad Nihaal Naazim

Date: 11th December 2024

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

Mohammad Nihaal Naazim
Student ID: 23211717

1 Introduction

This configuration manual offers detailed information for deploying, operating, and analyzing the artifact. It fills the gap from the creation of the artifact to its application by providing precise instructions on how to set up hardware and software, obtain the data set, run a model, and analyze results. The manual intends to promote the project's reproducibility and the correct performance of the process. It applies different learning algorithms like Convolutional Neural Networks (CNNs) for feature extraction, and Recurrent Neural Networks (RNNs) for flow data handling within the analysis and classification of the dataset.

2 Hardware Requirements

To ensure optimal performance, the following hardware specifications are recommended:

Processor: Intel Core i5 (10th Generation) or AMD Ryzen 5, 3.0 GHz and higher.

RAM: 8 GB (minimum) / 16 GB (recommended for bigger datasets).

Storage: 500 GB SSD for the swift reading and writing; another HDD for duplication purposes.

Operating System: Windows 10 (64-bit) / Ubuntu 20.04.

Graphics Card: Nvidia GTX 1050 or higher is optional for faster training time of the model i.e GPU.

3 Software Requirements

Operating System: Works on any 64-bit Windows or Linux-based operating system.

Programming Environment:

Python Version: 3.9 or above.

Jupyter Notebook: Version 6.5 or higher, as part of the related Anaconda distribution.

Required Libraries:

The following Python libraries are used in the course of the project:

- Pandas: For manipulating and analyzing data as well as for executing different programs and processes related to data.
- Matplotlib: Specifically for creating static and dynamic visualizations.
- Seaborn: For complex statistical data analysis and representation in different formats.
- TensorFlow/Keras: For construction and developing CNN and RNN models.

To install these libraries:

```
pip install pandas matplotlib seaborn tensorflow
```

Ensure Jupyter Notebook is installed:

```
pip install notebook
```

4 About Datasets

The data used in this project is obtained from the links found in the Sources section of this document. Change the dataset location path to the location of the notebook file. After downloading check the format of the file is '.csv' and to preview the data entered in the file use 'df.head()'. Ensure that there are zeros with 'df.isnull().sum()' and apply imputation or deletion of zeros as required. To ensure compatibility with the analysis, it is crucial to refer to the 'df.info()' to check if all the columns' names and data types agree with the expected ones. As datasets may be dynamic and change in future, it's important that their structure and integrity can also be confirmed before data analysis.

5 Steps for Notebook Execution

Open the Notebook:

Open command prompt or terminal and type Jupyter Notebook and it will open along with the project folder.

Locate the notebook folder in browser.

Run Cells Sequentially:

Begin with the first cell and run them all at once through the use of Shift + Enter.

Review Outputs:

Analyze tabular data, numerical measures, and graphical displays for relevant information.

6 Model Preparation

We have deployed two models CNN and RNN on both the datasets.

6.1 Stress dataset

- **CNN Model**

```
# CNN model Building
model = Sequential()

# First Convolutional Block
model.add(Conv1D(filters=32, kernel_size=2, activation='relu', input_shape=(X_train.shape[1], 1)))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))

# Second Convolutional Block
model.add(Conv1D(filters=64, kernel_size=2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))

# Third Convolutional Block
model.add(Conv1D(filters=128, kernel_size=2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))

# Fully Connected Layers
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(np.unique(y)), activation='softmax'))  # Adjust output layer for multiclass classification


model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])


early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(X_train, y_train, epochs=11, batch_size=64, validation_split=0.2, callbacks=[early_stopping])


test_loss, test_accuracy = model.evaluate(X_test, y_test)
print("\nTest Accuracy:", test_accuracy)

y_pred = model.predict(X_test)
y_pred = np.argmax(y_pred, axis=1)


print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

- **RNN Model**

```
#  RNN model building
model = Sequential()


model.add(SimpleRNN(128, activation='relu', input_shape=(X_train.shape[1], 1), return_sequences=True))
model.add(BatchNormalization())
model.add(Dropout(0.3))


model.add(SimpleRNN(64, activation='relu', return_sequences=False))
model.add(BatchNormalization())
model.add(Dropout(0.3))


model.add(Dense(128, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(3, activation='softmax'))


model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])


early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

#  model training
history = model.fit(X_train, y_train, epochs=11, batch_size=64, validation_split=0.2, callbacks=[early_stopping])


test_loss, test_accuracy = model.evaluate(X_test, y_test)
print("\nTest Accuracy:", test_accuracy)


y_pred = model.predict(X_test)
y_pred = np.argmax(y_pred, axis=1)


print("\nClassification Report:")
print(classification_report(y_test, y_pred))

conf_matrix = confusion_matrix(y_test, y_pred)
```

6.2 Depression dataset

- **CNN Model**

```
#CNN model
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=2, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))  # Binary classification

model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

# model training
model.fit(X_train, y_train, epochs=4, batch_size=32, validation_data=(X_test, y_test), verbose=1)


y_pred = (model.predict(X_test) > 0.5).astype("int32")


print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)


disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["No Depression", "Depression"])
disp.plot(cmap='Blues')
```

- **RNN Model**

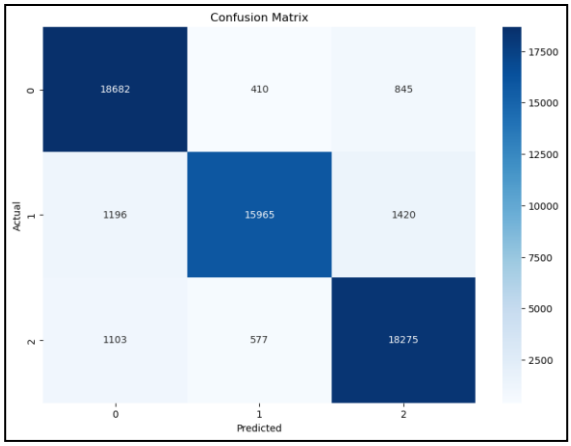
```
#RNN model
model = Sequential()
model.add(SimpleRNN(units=64, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))  # Binary classification

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# model training
model.fit(X_train, y_train, epochs=4, batch_size=32, validation_data=(X_test, y_test), verbose=1)


y_pred = (model.predict(X_test) > 0.5).astype("int32")


print(classification_report(y_test, y_pred))


cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["No Depression", "Depression"])
disp.plot(cmap='Blues')
```

7 Outputs

7.1 Stress dataset

- **CNN Model**

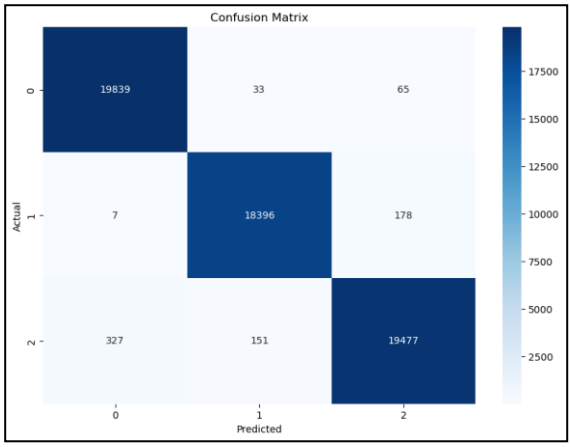
```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.94      0.91     19937
           1       0.94      0.86      0.90     18581
           2       0.89      0.92      0.90     19955

    accuracy                           0.91     58473
   macro avg       0.91      0.90      0.90     58473
weighted avg       0.91      0.91      0.90     58473
```

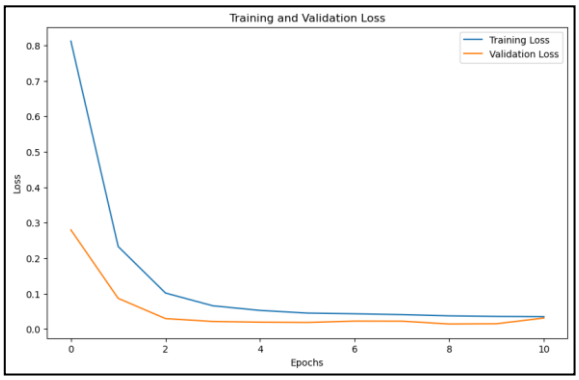


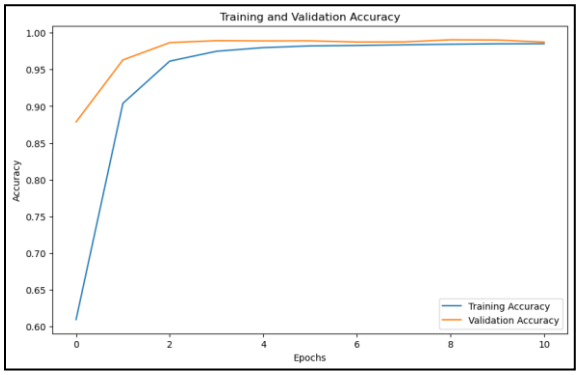
- **RNN Model**

Classification Report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	19937
1	0.99	0.99	0.99	18581
2	0.99	0.98	0.98	19955
accuracy			0.99	58473
macro avg	0.99	0.99	0.99	58473
weighted avg	0.99	0.99	0.99	58473





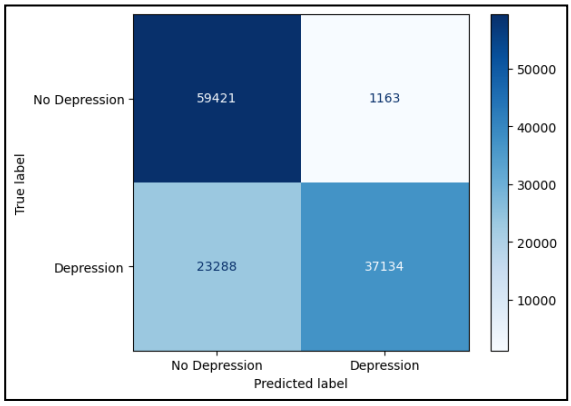


The charts show that the deep learning model is not overfitting, this is because there is a closer resemblance of the training and validation performance metrics. This means that, while the formation of the model is simple, it also generalizes well and does not overfit to the data that was used in its formation.

7.2 Depression dataset

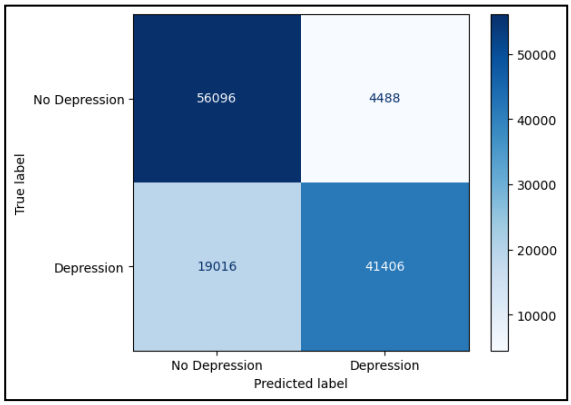
- **CNN Model**

	precision	recall	f1-score	support
0	0.72	0.98	0.83	60584
1	0.97	0.61	0.75	60422
accuracy			0.80	121006
macro avg	0.84	0.80	0.79	121006
weighted avg	0.84	0.80	0.79	121006



- **RNN Model**

	precision	recall	f1-score	support
0	0.75	0.93	0.83	60584
1	0.90	0.69	0.78	60422
accuracy			0.81	121006
macro avg	0.82	0.81	0.80	121006
weighted avg	0.82	0.81	0.80	121006



8 Troubleshooting

Library Not Found: Missing libraries need to be installed using the pip install and follow this syntax

<library_name> command.

File Path Errors: Make sure that the dataset file is located in right path.

Runtime Errors: Debug by commenting out a portion of code, identifying a particular cell which is causing issues, or use of the print statement.

Sources

Stress Dataset : https://www.kaggle.com/datasets/bhavikjikadara/mental-health-dataset

Depression Dataset : https://www.kaggle.com/datasets/anthonytherrien/depression-dataset?resource=download