National College of Ireland

# Configuration Manual

MSc Research Project
Data Analytics


Vishwadeep Sharma
Student ID: x22125256

School of Computing
National College of Ireland


Supervisor: Vladimir Milosavljevic

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Vishwadeep Sharma |
| **Student ID:** | x22125256 |
| **Programme:** | Data Analytics |
| **Year:** | 2023 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Vladimir Milosavljevic |
| **Submission Due Date:** | 14/12/2023 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | 524 |
| **Page Count:** | 5 |

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:** | Vishwadeep Sharma |
| **Date:** | 14th December 2023 |

**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

Vishwadeep Sharma

x22125256

## 1 Introduction

This guide is designed to give you a complete walkthrough on how to install and set up the software applications Python 3 and Jupyter Notebook. It covers everything from system requirements to installation, configuration, and troubleshooting. You can follow this guide step by step, and it also offers useful tips to ensure that your installation and configuration process goes smoothly.

## 2 Hardware Details

Hardware is the physical components of a computer system, such as the central processing unit (CPU), memory, storage devices, network cards, motherboards, and other components. Below are the details used in this project in figure 1. Local environment has been used in the initial phase of the project to sort the data and its visualization if any.

| | |
|---|---|
| Device name | Vishwa |
| Processor | AMD Ryzen 7 5700U with Radeon Graphics 1.80 GHz |
| Installed RAM | 16.0 GB (15.4 GB usable) |
| Device ID | 34954B61-C565-4F71-9DB6-634846B8EACE |
| Product ID | 00342-42621-89976-AAOEM |
| System type | 64-bit operating system, x64-based processor |
| Pen and touch | No pen or touch input is available for this display |

Figure 1: Hardware Details

# 3 Software Details

## 3.1 Anaconda

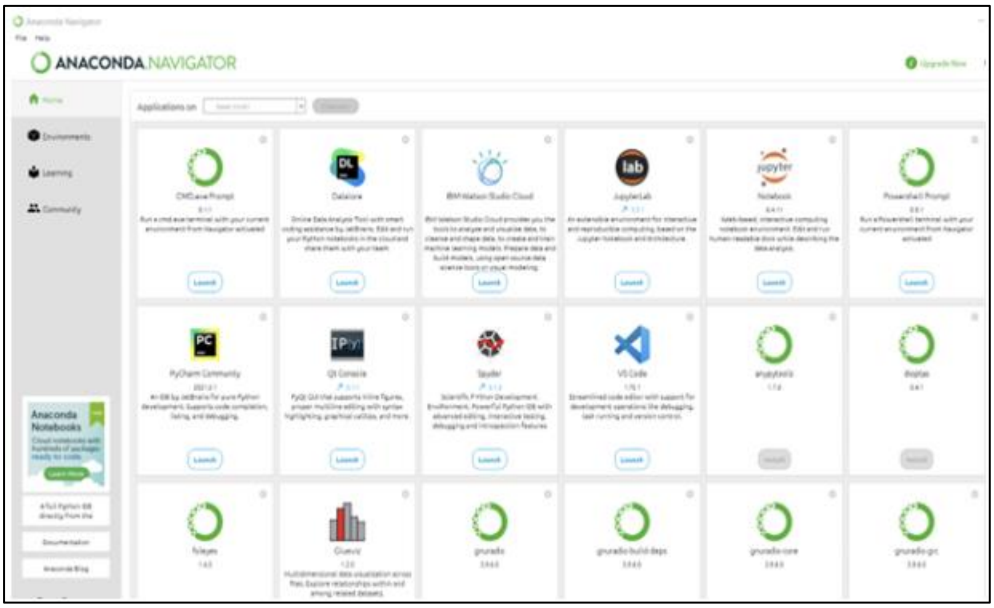


Figure 2: Anaconda Navigator Window

Anaconda is a package that combines the Python and R programming languages for scientific computing. Its purpose is to simplify the management and deployment of packages. Figure 2 shows the Anaconda Navigator window. Python was chosen as the programming language because of its extensive range of libraries, which are useful for analyzing data.

## 3.2 Jupyter Notebook


Figure 3 Jupyter Notebook

It serves as a handy tool for deploying machine learning models as it functions as an integrated development editor (IDE). This is made possible by allowing interactive code execution through a browser-based interface. You can find the launch icon for the Jupyter notebook in the Anaconda Navigator, as shown in Figure 3.

# 4 Python Libraries

```python
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import nltk
import emoji
import tensorflow as tf
import matplotlib.pyplot as plt
from nltk.tokenize import TweetTokenizer
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, LSTM, GRU, Dense
```

Figure 4 Utilized Libraries

In Figure 4, you can see the essential libraries imported using the "import" keyword. Additional libraries are imported as needed at later stages.

# 5 Dataset

```python
data = pd.read_csv(r'Random Tweets from Pakistan- Cleaned- Anonymous.csv', encoding_errors = 'ignore')
```

Figure 5 Data Insertion

The dataset comprising 202,202 rows and 7 columns was loaded from the CSV file titled 'Random Tweets from Pakistan- Cleaned- Anonymous.csv' as shown in Figure 5.

## 5.1 Dataset Pre-Processing

```
# Removing Urdu language
reg = re.compile(r'[\u0600-\u06ff]+', re.UNICODE)
data = data.apply(lambda x: re.sub(reg, "", x))

#Removing Spaces
data = data.apply(lambda x: re.sub(r'[  ]+', " ", x))

#Converting to lowercase letters
data = data.apply(lambda x: x.strip().lower())

#Removing https:
data = data.apply(lambda x: re.sub(r'https?:\/\/.*[\r\n]*', '', x))

#Removing symbols
data = data.apply(lambda x:re.sub(r'@.+?\s', '', x))
data = data.apply(lambda x: re.sub(r'#', '', x))
data = data.apply(lambda x: re.sub(r'rt : ', '', x))
data = data.apply(lambda x: re.sub(r'\n', ' ', x))
```

Figure 6 Pre-Processing Steps

As shown in Figure 6, preprocessing has been performed on the dataset such as removing tweets in Urdu language, removing spaces, converting all to lowercase letters and removing unwanted symbols and stopwords.

## 5.2 Tokenization

```
#Tokenizing the tweets
tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True,
                           reduce_len=True)
data = data.apply(tokenizer.tokenize)
```

Figure 7 Tokenization

Figure 7 depicts the tokenization of the cleaned dataset so that each tweet is converted into a series of tokens which will be later converted into sequences.

## 5.3 Sensitive Words Detection

```
# Function to check for sensitive words in different classes
def detect_sensitive_words(tokens):
    religious_words = ['god', 'allah', 'jesus', 'church', 'temple', 'mosque', 'religion', 'faith', 'prayer',
                       'sacred', 'prophet', 'holy', 'divine', 'scripture', 'worship']
    sexual_words = ['sex', 'intimate', 'desire', 'sexual', 'romance', 'erotic', 'passion', 'arousal', 'seduction',
                    'sensual', 'explicit', 'fantasy', 'climax']
    political_words = ['politics', 'government', 'president', 'election', 'vote', 'democracy', 'legislation',
                       'policy', 'corruption', 'revolution', 'activism', 'campaign', 'diplomacy']
    personal_info_words = ['ssn', 'address', 'phone', 'email', 'birthdate', 'credit', 'social',
                           'passport', 'bank', 'PIN']

    sensitive_words = {
        'religious': any(word in tokens for word in religious_words),
        'sexual': any(word in tokens for word in sexual_words),
        'political': any(word in tokens for word in political_words),
        'personal_info': any(word in tokens for word in personal_info_words),
    }

    return sensitive_words
```

Figure 8 Detect Sensitive Words

A function is created to detect sensitive words which are further classified into 4 categories such as religious, sexual, political and personal information.

# 6 Modelling

```
# Define the model
embedding_dim = 50
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=embedding_dim, input_length=max_length))
model.add(LSTM(units=50, return_sequences=True))
model.add(GRU(units=50))
model.add(Dense(units=4, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

num_classes = 4
y = np.random.randint(2, size=(len(padded_sequences), num_classes))

# Ensure X and y have the same number of samples
min_samples = min(len(padded_sequences), len(y))
X = padded_sequences[:min_samples]
y = y[:min_samples]

from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

Figure 9 Stacked Model

A stacked model is created by using Long short term memory and Gated recurrent units with an embedding layer on top which converts sequences into dense vectors of fixed size. Following that an LSTM layer is added to capture long term dependencies in sequences. GRU layer captures the short term dependencies and allows the model to learn all aspects of sequential patterns. Finally, the dense layer represents the output classes and uses the sigmoid activation function. In this project, I have prepared six sets of hyperparameters for this stacked model which will be compared with an individual LSTM and GRU.