Configuration Manual

MSc Research Project
Data Analytics

Tulasiram Dhullipalla
Student ID:X22196994

School of Computing
National College of Ireland

Supervisor: Ahmed Makki

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Tulasiram Dhullipalla

Student ID: X22196994

Programme: Data Analytics (MSCDAD_sep23) **Year:** 2023

Module: Research Project

Lecturer: Ahmed Makki

Submission Due Date: 12-08-2024

Project Title: Detection of suicidal content in the social media posts using advanced predictive classifiers
…………………………………………………………………………………………………….……

Word Count: **650** **Page Count: 10**

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: Tulasiram Dhullipalla……

Date: 12-08-2024……………………………………………………………………………

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
Detection of suicidal content in the social media posts using advanced predictive classifiers

Tulasiram Dhullipalla
X22196994

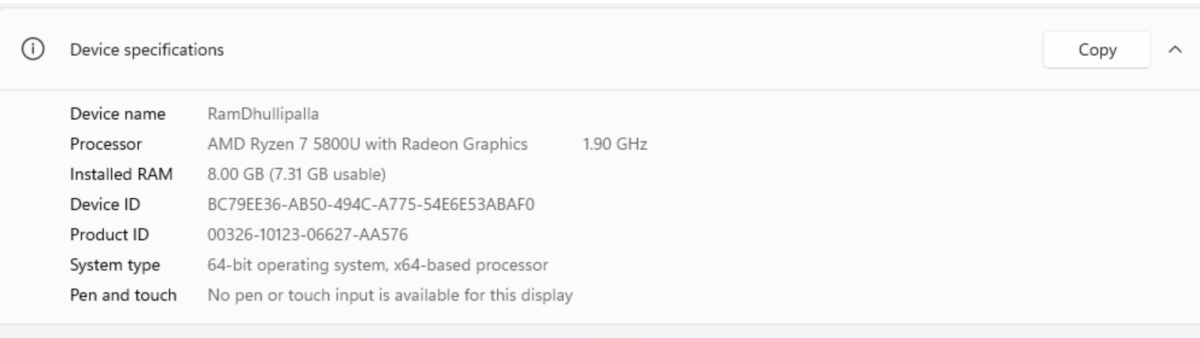
1 Overview

This is the configuration manual of the research project for the Detection of suicidal content in social media using advanced predictive classifier. This will provide detail instruction for setup the environment, software for code.

2 Hardware/Software Requirements

2.1 Hardware Requirements

The system hardware configuration on which the research models are developed and successfully executed. These configurations are given below.


Device specifications

Copy

Device name	RamDhullipalla
Processor	AMD Ryzen 7 5800U with Radeon Graphics 1.90 GHz
Installed RAM	8.00 GB (7.31 GB usable)
Device ID	BC79EE36-AB50-494C-A775-54E6E53ABAF0
Product ID	00326-10123-06627-AA576
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Fig1: Hardware configuration

- Operating System: Windows 11 Home, Version 23H2

- Processor: AMD Ryzen 7 5800U with Radeon Graphics, 1.90 GHz

- Storage: 500 GB SSD
- Installed RAM: 8.00 GB (7.31 GB usable)
- System Type: 64-bit operating system, x64-based processor

2.2 Software Requirement:

The software requirement for the research project to develop the machine learning and neural network algorithms are given below.

- Integrated Development Environment: Jupyter notebook
- Programming language: Python 3.7+
- Data storage: local server
- Other software: Excel /CSV, Text editor.

3 Environment setup:

Jupyter notebook: To setup jupyter notebook, Anaconda software must be downloaded first. After installing the anaconda, jupyter is the default software in anaconda.

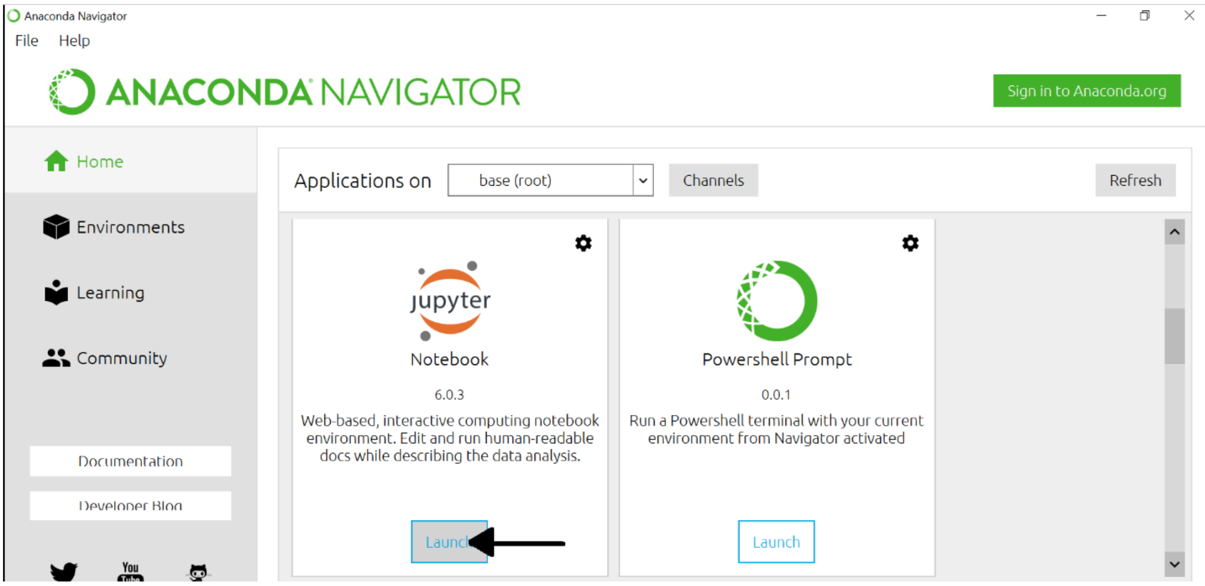


Fig2 : Anaconda application

4 Implementation

4.1 Data selection:

The dataset is taken from kaggle site which is popular for the data science and machine learning project dataset repository. Reddit dataset contains 700k rows and twitter dataset contains 200 k rows.

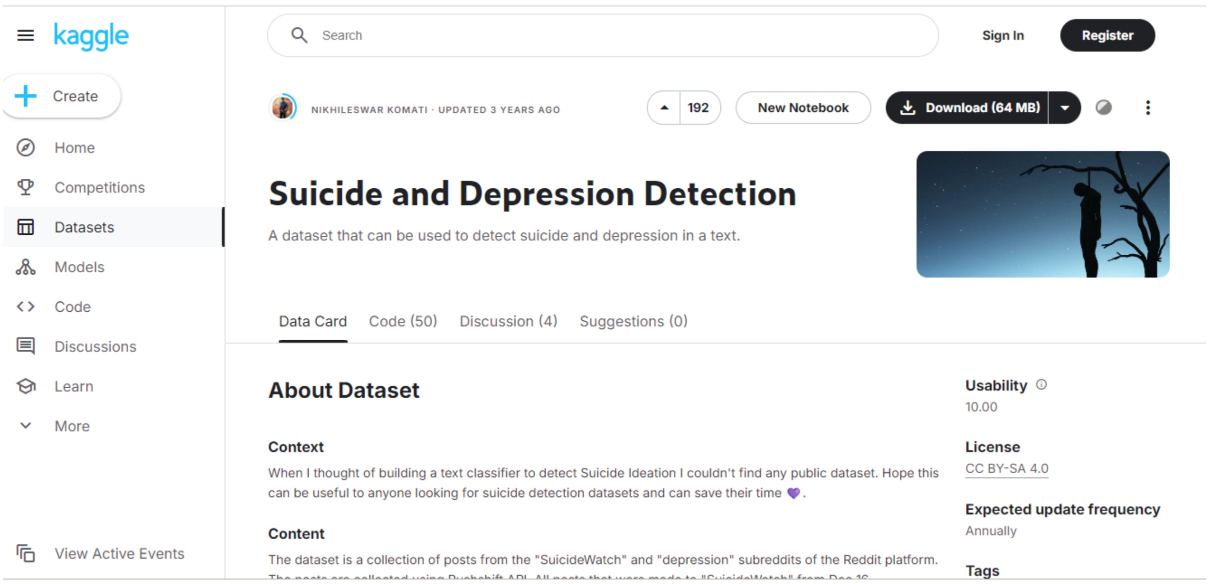


Fig3 : kaggle Application

4.2 Data transformation and Model Building

Importing required python library:

- re
- nltk
- tensorflow.keras
- pandas
- sklearn
- wordcloud
- matplotlib.pyplot

Analysis to understand the data:

1. word cloud is used to understand the most frequently used words (suicidal and non suicidal)

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Combine all tokenized words into a single string
all_tokens = ' '.join([word for tokens in df['tokens'] for word in tokens])

# Generate WordCloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(all_tokens)

# Display the WordCloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

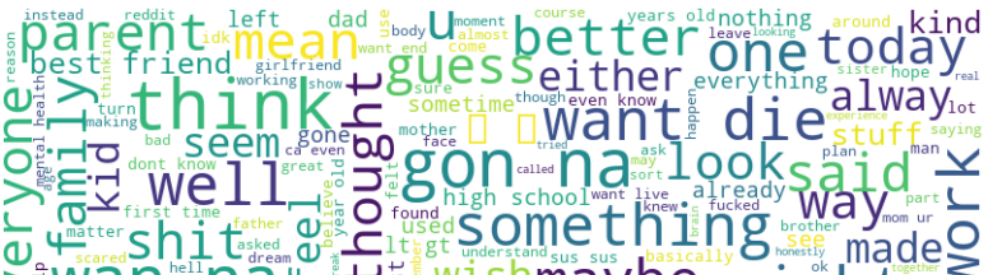


Fig4 : word cloud

2. To understand two and three combination , CountVectorizer is used to generate the n-gram . In this process, most combine words is generated

```python
from sklearn.feature_extraction.text import CountVectorizer

# Define a function to generate n-grams
def generate_ngrams(texts, n):
    vectorizer = CountVectorizer(ngram_range=(n, n))
    X = vectorizer.fit_transform(texts)
    ngrams = vectorizer.get_feature_names_out()
    counts = X.sum(axis=0).A1
    return dict(zip(ngrams, counts))

# Generate Bi-Grams
bi_grams = generate_ngrams(df['text'], 2)

# Generate Tri-Grams
tri_grams = generate_ngrams(df['text'], 3)

# Display some Bi-Grams and Tri-Grams
print("Bi-Grams:", dict(sorted(bi_grams.items(), key=lambda x: x[1], reverse=True)[:10]))
print("Tri-Grams:", dict(sorted(tri_grams.items(), key=lambda x: x[1], reverse=True)[:10]))
```

```
Bi-Grams: {'want to': 107589, 'to be': 70240, 'in the': 59049, 'to do': 52859, 'my life': 52468, 'filler filler': 50252, 'of m
y': 49337, 'in my': 45334, 'feel like': 44542, 'of the': 42780}
Tri-Grams: {'filler filler filler': 47465, 'don want to': 27207, 'don know what': 15686, 'to kill myself': 15152, 'just want t
o': 14844, 'what to do': 14586, 'know what to': 13124, 'want to be': 13110, 'want to die': 12760, 'fuck fuck fuck': 12184}
```

Fig5 : N-gram

Reddit Implementation:

1. reading the dataset from local storage and basic analysis shown given below

```python
import pandas as pd

# Load the dataset
file_path = r'C:\Users\dhull\OneDrive\Desktop\R_folder\Suicide_Detection.csv'
data = pd.read_csv(file_path)

#  the first few rows of the dataset
print(data.head())

# Check for missing values
print(data.isnull().sum())

# count
print(data['class'].value_counts())
```

```
   Unnamed: 0                                               text        class
0           2  Ex Wife Threatening SuicideRecently I left my ...      suicide
1           3  Am I weird I don't get affected by compliments...  non-suicide
2           4  Finally 2020 is almost over... So I can never ...  non-suicide
3           8          i need helpjust help me im crying so hard      suicide
4           9  I’m so lostHello, my name is Adam (16) and I’v...      suicide
Unnamed: 0    0
text          0
class         0
dtype: int64
class
suicide        116037
non-suicide    116037
```

Fig6 : Reddit basic analysis code

2. The data cleaning and preprocessing is done , it helps model to classify effectively. Those steps are removing null values, stop words (which mean removing the unimportant words) , removing special symbols and converting lower characters.

```
# Drop rows with missing values
data.dropna(inplace=True)

# Text preprocessing
import re
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    # Remove special characters and numbers
    text = re.sub(r'\W+', ' ', text)
    # Convert to lowercase
    text = text.lower()
    # Remove stopwords
    text = ' '.join([word for word in text.split() if word not in stop_words])
    return text

data['cleaned_text'] = data['text'].apply(preprocess_text)

[nltk_data] Downloading package stopwords to
```

Fig7: Stop words

3. The text data map to numerical values to make model effective in classifying the data. Tokenization is used to convert numerical values and pad_sequences for fix the length of the data .

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Tokenization
tokenizer = Tokenizer()
tokenizer.fit_on_texts(data['cleaned_text'])
sequences = tokenizer.texts_to_sequences(data['cleaned_text'])

# Padding sequences
max_length = 100  # Set based on data exploration
padded_sequences = pad_sequences(sequences, maxlen=max_length, padding='post')

# Train-test split
from sklearn.model_selection import train_test_split

X = padded_sequences
y = data['class'].apply(lambda x: 1 if x == 'suicide' else 0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig8: Tokenization words

4. Label encoder is used to convert target text (Suicidal or non-suicidal) to numerical values. It helps machine learning model to give accuract output.

```
# Drop rows with missing values
data.dropna(subset=['text', 'class'], inplace=True)

# Encode the target variable 'class'
label_encoder = LabelEncoder()
data['class'] = label_encoder.fit_transform(data['class'])
```

Fig9: label encoder

5. Similar to tonization , vectorization is used to convert text data into dense vector numerical form . it helps machine learning to identify the keyword related to suicidal.

```
# Initialize the TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)

# Fit and transform the text data
X_text = tfidf_vectorizer.fit_transform(data['text'])

# Target variable
y = data['class']
```

```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_text, y, test_size=0.3, random_state=42)
```

Fig10 : Vectorizer

6. After all the preprocess step, the data is ready for train the model. LSTM model is trained and test the data, given below.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

# Define the LSTM model
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=128, input_shape=(max_length,)))
model.add(LSTM(units=128))
model.add(Dense(1, activation='sigmoid'))

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

# Train the model
model.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.2)
```

```
C:\Users\dhull\anaconda3\Lib\site-packages\keras\src\layers\core\embedding.py:81: UserWarning: Do not pass an `input_shape`/`in
put_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the mod
el instead.
  super().__init__(**kwargs)
```

```
Epoch 1/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 1419s 305ms/step - accuracy: 0.8593 - loss: 0.3195 - val_accuracy: 0.9372 - val_loss: 0.1716
Epoch 2/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 1425s 307ms/step - accuracy: 0.9522 - loss: 0.1310 - val_accuracy: 0.9421 - val_loss: 0.1559
Epoch 3/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 1433s 309ms/step - accuracy: 0.9696 - loss: 0.0843 - val_accuracy: 0.9401 - val_loss: 0.1719
Epoch 4/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 1429s 308ms/step - accuracy: 0.9814 - loss: 0.0512 - val_accuracy: 0.9349 - val_loss: 0.2090
Epoch 5/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 83949s 18s/step - accuracy: 0.9881 - loss: 0.0317 - val_accuracy: 0.9323 - val_loss: 0.2535
```

Fig11 : LSTM model

7. The LSTM-CNN hybrid model is developed to classify the suicidal text in the post. it is given below.

```python
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dropout

#  CNN+LSTM model
cnn_lstm_model = Sequential()
cnn_lstm_model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=128, input_shape=(max_length,)))
cnn_lstm_model.add(Conv1D(filters=64, kernel_size=5, activation='relu'))
cnn_lstm_model.add(MaxPooling1D(pool_size=2))
cnn_lstm_model.add(LSTM(units=128))
cnn_lstm_model.add(Dense(1, activation='sigmoid'))

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

# Train the model
cnn_lstm_model.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.2)
```

```
Epoch 1/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 1303s 280ms/step - accuracy: 0.9029 - loss: 0.2480 - val_accuracy: 0.9385 - val_loss: 0.1698
Epoch 2/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 1290s 278ms/step - accuracy: 0.9565 - loss: 0.1236 - val_accuracy: 0.9398 - val_loss: 0.1611
Epoch 3/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 1390s 300ms/step - accuracy: 0.9716 - loss: 0.0815 - val_accuracy: 0.9362 - val_loss: 0.1815
Epoch 4/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 1338s 288ms/step - accuracy: 0.9837 - loss: 0.0495 - val_accuracy: 0.9284 - val_loss: 0.2296
Epoch 5/5
4642/4642 ━━━━━━━━━━━━━━━━━━━━ 40895s 9s/step - accuracy: 0.9907 - loss: 0.0284 - val_accuracy: 0.9274 - val_loss: 0.2547

<keras.src.callbacks.history.History at 0x1d1b4e9ff10>
```

Fig 12 : LSTM-CNN

8. Logistic regression is developed to classify the suicidal text in the post . it is given below.

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

# Initialize and train the Logistic Regression model
logistic_model = LogisticRegression(max_iter=1000, random_state=42)
logistic_model.fit(X_train, y_train)

# Make predictions
y_pred_logistic = logistic_model.predict(X_test)

# Evaluate the model
print("Logistic Regression Classification Report:")
print(classification_report(y_test, y_pred_logistic))
print("Logistic Regression Accuracy Score:", accuracy_score(y_test, y_pred_logistic))
```

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

           0       0.92      0.94      0.93     34824
           1       0.94      0.92      0.93     34799

    accuracy                           0.93     69623
   macro avg       0.93      0.93      0.93     69623
weighted avg       0.93      0.93      0.93     69623

Logistic Regression Accuracy Score: 0.931674877554831
```

Fig13 : Logistic regression

9. Svm is used to identify suicidal text , it is popular machine learning model for classification the data.

```
from sklearn.svm import SVC

# Initialize and train the SVM model
svm_model = SVC(kernel='linear', random_state=42, max_iter=1000)
svm_model.fit(X_train, y_train)

# Make predictions
y_pred_svm = svm_model.predict(X_test)

# Evaluate the model
print("SVM Classification Report:")
print(classification_report(y_test, y_pred_svm))
print("SVM Accuracy Score:", accuracy_score(y_test, y_pred_svm))
```

```
C:\Users\dhull\anaconda3\Lib\site-packages\sklearn\svm\_base.py:297: ConvergenceWarning: Solver terminated early (max_iter=100
0).  Consider pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(

SVM Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.71      0.74     34824
           1       0.73      0.79      0.76     34799

    accuracy                           0.75     69623
   macro avg       0.75      0.75      0.75     69623
weighted avg       0.75      0.75      0.75     69623

SVM Accuracy Score: 0.7527828447495799
```

Fig14 : SVM

Tweet Implementation:

1. Both Reddit dataset and twitter data follow the same preprocessing step, because both datasets have the same structure.

2. The LSTM-CNN hybrid model is developed to classify the suicidal text in the post. For the twitter LSTM, 10 epochs are used, whereas Reddit 5 epochs are used. It is given below.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

# Define LSTM model
model_lstm = Sequential([
    Embedding(input_dim=10000, output_dim=128, input_length=max_len),
    LSTM(64, return_sequences=False),
    Dense(1, activation='sigmoid')
])

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

# Train the model
history_lstm = model_lstm.fit(X_train_pad, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
C:\Users\dhull\anaconda3\Lib\site-packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument `input_length` is depre
cated. Just remove it.
  warnings.warn(
193/193 ━━━━━━━━━━━━━━━━━━━━ 11s 47ms/step - accuracy: 0.9631 - loss: 0.1920 - val_accuracy: 0.9857 - val_loss: 0.0621
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 9s 45ms/step - accuracy: 0.9823 - loss: 0.0692 - val_accuracy: 0.9896 - val_loss: 0.0516
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 9s 44ms/step - accuracy: 0.9869 - loss: 0.0429 - val_accuracy: 0.9909 - val_loss: 0.0528
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 9s 45ms/step - accuracy: 0.9910 - loss: 0.0296 - val_accuracy: 0.9903 - val_loss: 0.0515
Epoch 5/10
```

Fig15 : LSTM Twitter

3. The LSTM-CNN hybrid model is developed to classify the suicidal text in the post.

it is given below.

```
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten

# Define CNN-LSTM model
model_cnn_lstm = Sequential([
    Embedding(input_dim=10000, output_dim=128, input_length=max_len),
    Conv1D(filters=64, kernel_size=5, activation='relu'),
    MaxPooling1D(pool_size=4),
    LSTM(64, return_sequences=False),
    Dense(1, activation='sigmoid')
])

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

# Train the model
history_cnn_lstm = model_cnn_lstm.fit(X_train_pad, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 8s 27ms/step - accuracy: 0.9509 - loss: 0.1774 - val_accuracy: 0.9857 - val_loss: 0.0666
Epoch 2/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - accuracy: 0.9815 - loss: 0.0763 - val_accuracy: 0.9909 - val_loss: 0.0487
Epoch 3/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - accuracy: 0.9875 - loss: 0.0347 - val_accuracy: 0.9883 - val_loss: 0.0550
Epoch 4/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - accuracy: 0.9938 - loss: 0.0190 - val_accuracy: 0.9877 - val_loss: 0.0599
Epoch 5/10
193/193 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - accuracy: 0.9972 - loss: 0.0094 - val_accuracy: 0.9896 - val_loss: 0.0655
Epoch 6/10
```

Fig16: LSTM-CNN

4. The logistic regression is developed to classify the suicidal text in the post. it is given below.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

# Initialize and train Logistic Regression model
log_reg_model = LogisticRegression(max_iter=100, random_state=42)
log_reg_model.fit(X_train_tfidf, y_train)

# Predict on test data
y_pred_log_reg = log_reg_model.predict(X_test_tfidf)

# Evaluate the model
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_log_reg))
print(classification_report(y_test, y_pred_log_reg, target_names=label_encoder.classes_))
```

```
Logistic Regression Accuracy: 0.9792099792099792
              precision    recall  f1-score   support

 non-suicide       0.98      1.00      0.99      1880
     suicide       0.83      0.11      0.20        44

    accuracy                           0.98      1924
   macro avg       0.91      0.56      0.59      1924
weighted avg       0.98      0.98      0.97      1924
```

Fig 17 : logistic regression

5. Svm is used to identify suicidal text , it is popular machine learning model for classification the data.

```python
from sklearn.svm import SVC

# Initialize and train SVM model
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train_tfidf, y_train)

# Predict on test data
y_pred_svm = svm_model.predict(X_test_tfidf)

# Evaluate the model
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm, target_names=label_encoder.classes_))
```

```
SVM Accuracy: 0.9864864864864865
              precision    recall  f1-score   support

 non-suicide       0.99      1.00      0.99      1880
     suicide       0.95      0.43      0.59        44

    accuracy                           0.99      1924
   macro avg       0.97      0.72      0.79      1924
weighted avg       0.99      0.99      0.98      1924
```

Fig 18 : SVM