National College of Ireland

Configuration Manual

MSc Research Project
MSc data Analytics

Varun Bhalerao
Student ID: 22206884

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Varun Atul Bhalerao		
Student ID:	22206884		
Programme:	Masters in Data Analytics	**Year:**	2024
Module:	MSc Research Project		
Supervisor:	Abubakr Siddig		
Submission Due Date:	12/08/2024		
Project Title:	Contextual Hate Speech Detection Leveraging RoBERTa: Overcoming Challenges in Online Communication		
Word Count:	847 **Page Count:** 11		

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:	Varun Atul Bhalerao
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

Varun Bhalerao
Student ID: x22206884

1 Introduction

This configuration document includes all the details relevant to the research study on hate speech detection using the RoBERTa model, implemented with a standard laptop. The manual is organized with step-by-step procedures for reproducing the thesis work and explaining all the artifacts related to the report. Code snippets are given that encompass the whole process from data collection and model development to experimentation and result evaluation.

2 Specifications

2.1 Hardware Specifications

This project was done on a standard laptop which is Acer nitro-5 2020 model. Given below are the hardware specifications for the device.

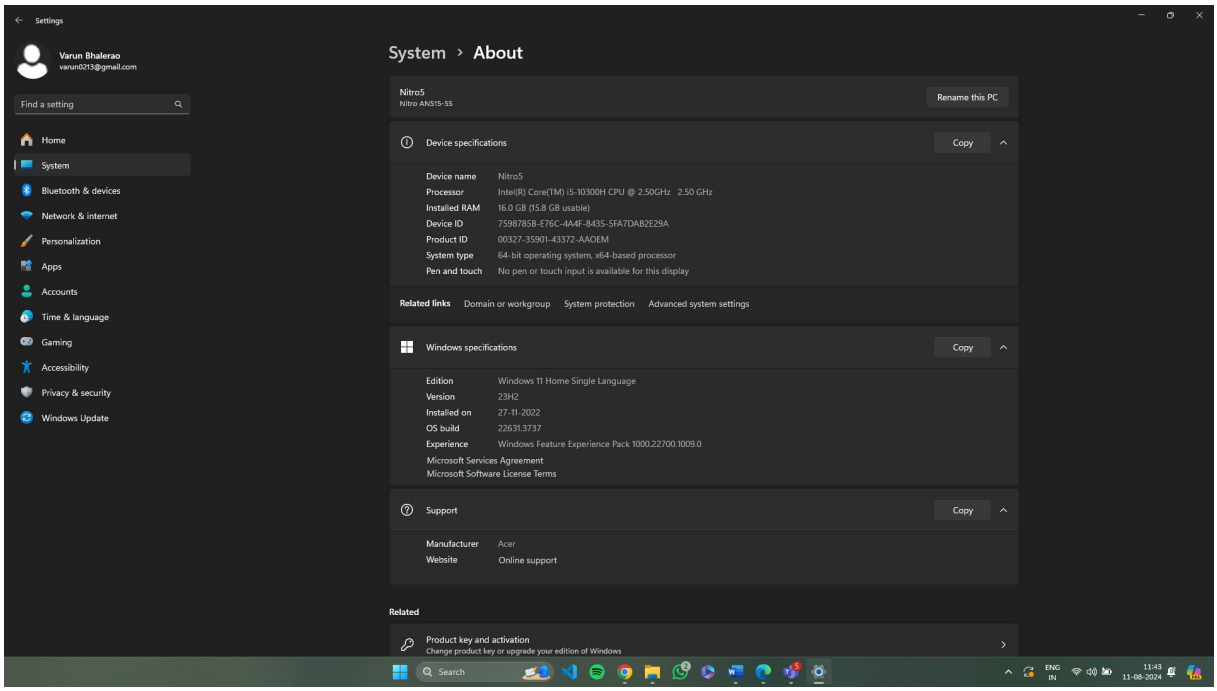

Figure 1: Device Specifications

2.2 Software specifications:

As this research was done on a deep learning task, the hardware and software specifications was not enough to process the data as well as run both the models. The model size being large, which is not compatible for a standard laptop (*FacebookAI/xlm-roberta-base · Hugging*

Face, no date) (*distilbert/distilbert-base-uncased · Hugging Face*, 2024). Hence, Google colab was used to process the models. The specifications for Google Colab are given below:

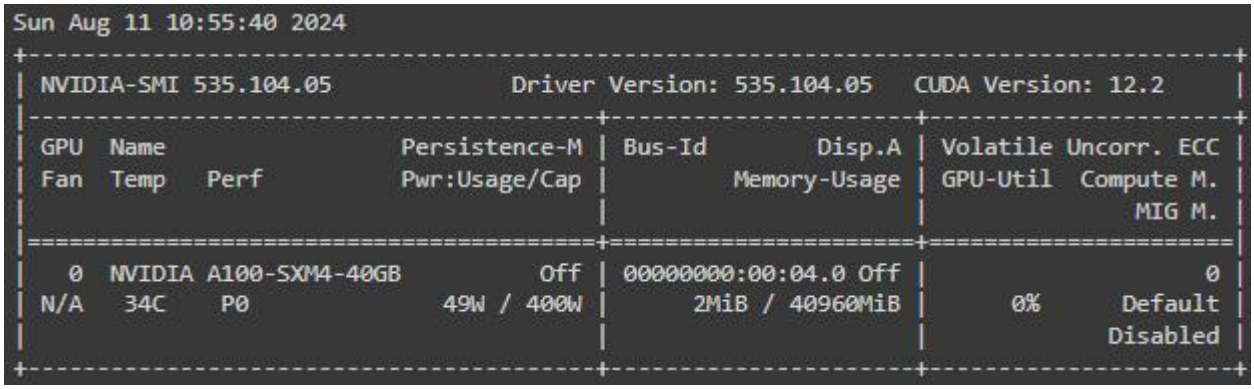

Figure 2: Google Colab Specifications

3 Data Collection

The dataset was taken from Kaggle. It was a hate speech and offensive language dataset (*Hate Speech and Offensive Language Dataset*, no date). The file was '.csv' file. The dataset has various tweets. It can be found on the following URL link given below: https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset

4 Data Preprocessing

4.1 Installing the required packages such as emoji, optuna, Textblob.

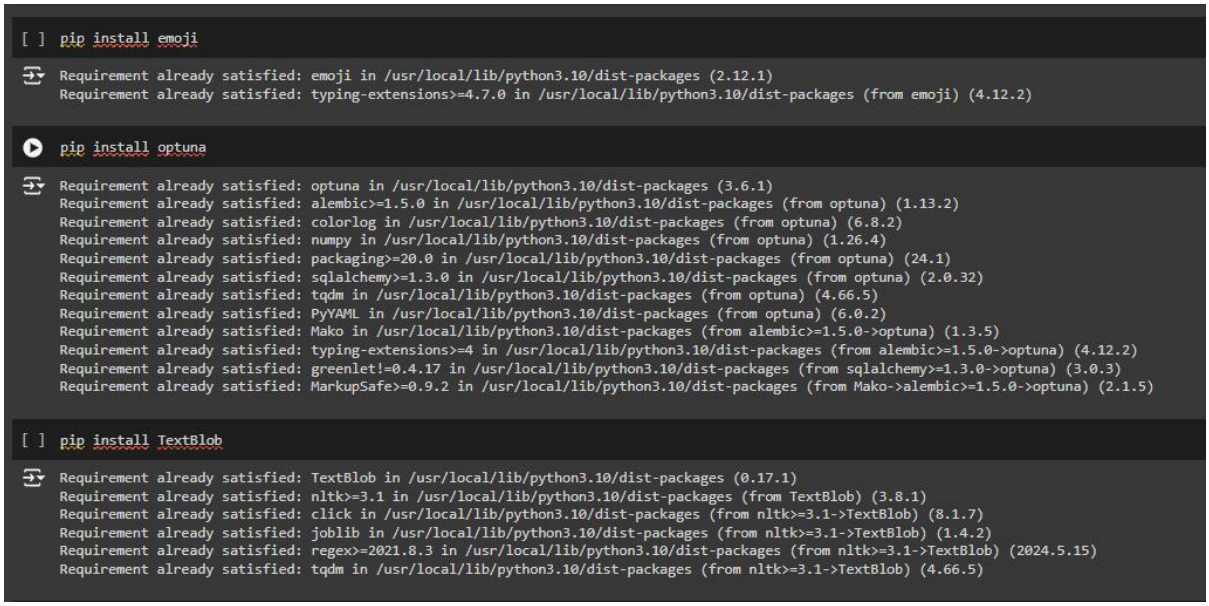

Figure 3: Installing Packages

4.2 Importing required libraries:

```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    import numpy as np
    import string
    from sklearn.preprocessing import StandardScaler
    from sklearn.model_selection import train_test_split
    import tensorflow as tf
    import optuna
    from transformers import RobertaTokenizer, TFRobertaForSequenceClassification
    from nltk.tokenize import word_tokenize
    from nltk.stem import SnowballStemmer
    from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
    from textblob import TextBlob
    import nltk
    import emoji
    import random
    import nltk
    nltk.download('wordnet')
    from nltk.corpus import wordnet
    from sklearn.preprocessing import LabelEncoder
    from transformers import RobertaTokenizerFast, TFRobertaForSequenceClassification, create_optimizer
    from sklearn.metrics import classification_report, accuracy_score
    from sklearn.metrics import f1_score
    from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

Figure 4: Importing Libraries

4.3 Downloading the tokenizers.

```
# Download the punkt tokenizer models
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

Figure 5: Downloading Tokenizers

4.4 Loading the dataset csv file using pandas.

```
[ ] # Load the dataset
    file_path = 'labeled_data.csv'
    data = pd.read_csv(file_path)
```

Figure 6: Loading the dataset

4.5 Printing the first 5 rows of the data set using ".head" command.

```
# Display the first 5 rows of the dataset
print("\nFirst 5 Rows of the Dataset:")
print(data.head())
```

```
First 5 Rows of the Dataset:
   Unnamed: 0  count  hate_speech  offensive_language  neither  class  \
0           0      3            0                   0        3      2
1           1      3            0                   3        0      1
2           2      3            0                   3        0      1
3           3      3            0                   2        1      1
4           4      6            0                   6        0      1

                                               tweet
0  !!! RT @mayasolovely: As a woman you shouldn't...
1  !!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2  !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3  !!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4  !!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
```

Figure 7: First 5 columns of the data set.

4.6 Checking the dataset to see if there are any null values by using the "data.isnull" command.

```
[ ] # Checking for missing values
    print("\nMissing Values:")
    print(data.isnull().sum())
```

```
Missing Values:
Unnamed: 0            0
count                 0
hate_speech           0
offensive_language    0
neither               0
class                 0
tweet                 0
dtype: int64
```

Figure 8: Missing Values

4.7 Printing the Class distribution

```
# Distribution of the 'class' column
plt.figure(figsize=(8, 6))
sns.countplot(x='class', data=data)
plt.title('Distribution of Classes')
plt.xlabel('Class')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1, 2], labels=['Hate Speech', 'Offensive Language', 'Neither'])
plt.show()
```

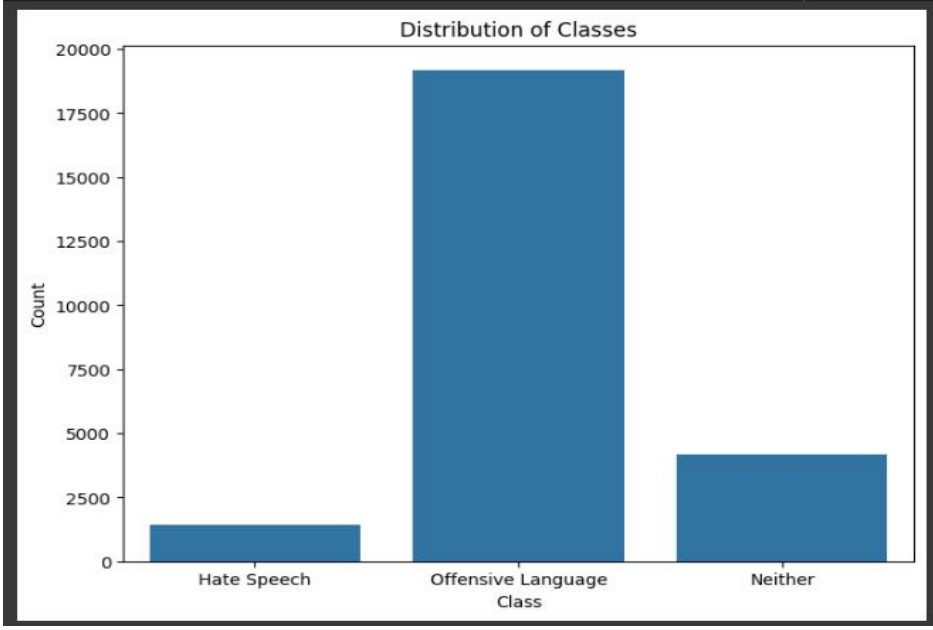

Figure 9: Class Distribution

4.8 Printing the average count of labels for the class.

```
# Average count of labels per class
class_counts = data.groupby('class')[['hate_speech', 'offensive_language', 'neither']].mean().reset_index()
class_counts_melted = class_counts.melt(id_vars='class', value_vars=['hate_speech', 'offensive_language', 'neither'],
                                        var_name='Label', value_name='Average Count')

plt.figure(figsize=(10, 6))
sns.barplot(x='class', y='Average Count', hue='Label', data=class_counts_melted)
plt.title('Average Count of Labels per Class')
plt.xlabel('Class')
plt.ylabel('Average Count')
plt.xticks(ticks=[0, 1, 2], labels=['Hate Speech', 'Offensive Language', 'Neither'])
plt.show()
```

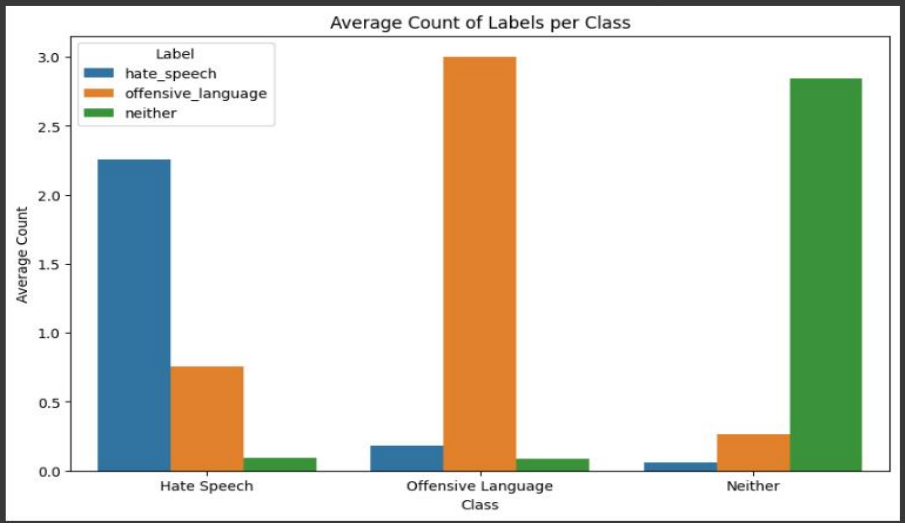

Figure 10: Average count of Labels

4.9 Performing text preprocessing for the dataset like feature scaling, Removing duplicates, creating new features for tweet length.

```
[ ] # Text Preprocessing for model training
    def preprocess_text(text):
        text = text.lower()  # Lowercasing
        text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation
        tokens = word_tokenize(text)  # Tokenization
        tokens = [word for word in tokens if word not in ENGLISH_STOP_WORDS]  # Remove stop words
        stemmer = SnowballStemmer('english')
        tokens = [stemmer.stem(word) for word in tokens]  # Stemming
        return ' '.join(tokens)

    data['tweet'] = data['tweet'].apply(preprocess_text)

[ ] # Feature Scaling
    scaler = StandardScaler()
    data[['count', 'hate_speech', 'offensive_language', 'neither']] = scaler.fit_transform(
        data[['count', 'hate_speech', 'offensive_language', 'neither']])

[ ] # Remove Duplicates
    data = data.drop_duplicates()

[ ] # Feature Engineering - Create new feature for tweet length
    data['tweet_length'] = data['tweet'].apply(lambda x: len(x.split()))
```

Figure 11: Text preprocessing

4.10 Performing sentiment analysis

```
[ ] # Perform Sentiment Analysis
    data['sentiment'] = data['tweet'].apply(lambda x: TextBlob(x).sentiment.polarity)

    # Sentiment per class
    plt.figure(figsize=(8, 6))
    sns.boxplot(x='class', y='sentiment', data=data)
    plt.title('Sentiment per Class')
    plt.xlabel('Class')
    plt.ylabel('Sentiment')
    plt.xticks(ticks=[0, 1, 2], labels=['Hate Speech', 'Offensive Language', 'Neither'])
    plt.show()
```

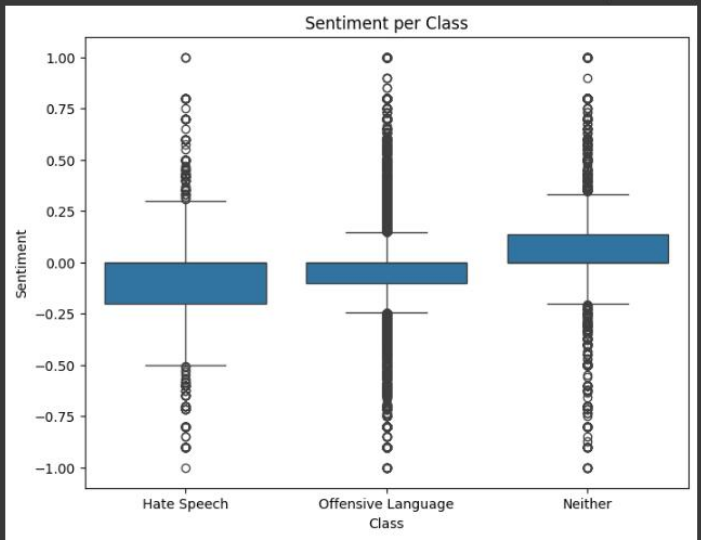

Figure 12: Sentiment analysis

4.11Saving the pre-processed file" as a ".csv".

```
[ ]  # Save the preprocessed data to a CSV file
     output_file_path = 'preprocessed_labeled_data.csv'
     data.to_csv(output_file_path, index=False)

     # Verify that the file was saved correctly
     print(f"Preprocessed data saved to {output_file_path}")

     Preprocessed data saved to preprocessed_labeled_data.csv
```

Figure 13: Saving file as .csv

Below is the before and after of the dataset after performing all the preprocessing techniques.

	count	hate_spee	offensive_l	neither	class	tweet
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!!
2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she start to cry? You be confused as shit
3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny
4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might be faker than the bitch who told it to ya 
5	3	1	2	0	1	!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just blows me..claim you so faithful and down for somebody but still fucking with hoes! 😂😂😂"
6	3	0	3	0	1	!!!!!"@__BrighterDays: I can not just sit up and HATE on another bitch .. I got too much shit going on!"
7	3	0	3	0	1	!!!!“@selfiequeenbri: cause I'm tired of you big bitches coming for us skinny girls!!”
8	3	0	3	0	1	" & you might not get ya bitch back & thats that "
9	3	1	2	0	1	" @rhythmixx_ :hobbies include: fighting Mariam" bitch

Figure 14: Before preprocessing

Unnamed:	count	hate_spee	offensive_l	neither	class	tweet	tweet_leng	sentiment
0	-0.275721	-0.443966	-1.72478	2.201388	2	rt mayasolov woman shouldnt complain clean hous amp man trash	10	0.3666667
1	-0.275721	-0.443966	0.418948	-0.493361	1	rt mleew17 boy dat coldtyga dwn bad cuffin dat hoe 1st place	12	-0.7
2	-0.275721	-0.443966	0.418948	-0.493361	1	rt urkindofbrand dawg rt 80sbaby4lif fuck bitch start confus shit	10	-0.3
3	-0.275721	-0.443966	-0.295628	0.4048884	1	rt cganderson vivabas look like tranni	6	0
4	3.1216249	-0.443966	2.5626761	-0.493361	1	rt shenikarobert shit hear true faker bitch told ya 57361	10	0.075
5	-0.275721	1.1387172	-0.295628	-0.493361	1	tmadisonx shit just blow meclaim faith somebodi fuck hoe 128514128514128514	10	-0.3
6	-0.275721	-0.443966	0.418948	-0.493361	1	brighterday just sit hate bitch got shit go	8	-0.5
7	-0.275721	-0.443966	0.418948	-0.493361	1	8220selfiequeenbri caus im tire big bitch come skinni girls8221	9	0
8	-0.275721	-0.443966	0.418948	-0.493361	1	amp ya bitch amp that	5	0
9	-0.275721	1.1387172	-0.295628	-0.493361	1	rhythmixx hobbi includ fight mariam bitch	6	0
10	-0.275721	-0.443966	0.418948	-0.493361	1	keek bitch curv lol walk convers like smh	8	0.8
11	-0.275721	-0.443966	0.418948	-0.493361	1	murda gang bitch gang land	5	0
12	-0.275721	-0.443966	-0.295628	0.4048884	1	hoe smoke loser yea ig	5	0
13	-0.275721	-0.443966	0.418948	-0.493361	1	bad bitch thing like	4	-0.7
14	-0.275721	1.1387172	-0.295628	-0.493361	1	bitch	1	0
15	-0.275721	-0.443966	0.418948	-0.493361	1	bitch nigga miss	3	0
16	-0.275721	-0.443966	0.418948	-0.493361	1	bitch plz	2	0
17	-0.275721	1.1387172	-0.295628	-0.493361	1	bitch love	2	0.5
18	-0.275721	-0.443966	0.418948	-0.493361	1	bitch cut everyday b	4	-0.2

Figure 15: After Preprocessing

4.12 Performing one-hot encoding on the dataset.

```
[ ]  # Convert class labels to categorical (one-hot encoding)
     data['class'] = data['class'].astype(int)
```

Figure 16: One-hot encoding

4.13 Ensuring there are no null values in the dataset.

```
[ ]  # Ensure there are no null values in the dataset
     data = data.dropna()
```

Figure 17: Dropping the null values

5 Model Implementation

5.1 Splitting the dataset into train and test partitions.

```
[ ]  # Splitting data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(
         data['tweet'].values,  # Features (tweets)
         data['class'].values,  # Target variable (class labels)
         test_size=0.2,  # Percentage of data to use for testing (20%)
         random_state=42  # Random seed for reproducibility
     )
```

Figure 18: Splitting data

5.2 Data Augmentation

```
[ ]  # Data Augmentation: Synonym Replacement
     def synonym_replacement(text, n):
         words = text.split()
         new_words = words.copy()
         random_word_list = list(set(words))
         random.shuffle(random_word_list)
         num_replaced = 0
         for random_word in random_word_list:
             synonyms = wordnet.synsets(random_word)
             if synonyms:
                 synonym = synonyms[0].lemmas()[0].name()
                 new_words = [synonym if word == random_word else word for word in new_words]
                 num_replaced += 1
             if num_replaced >= n:
                 break
         sentence = ' '.join(new_words)
         return sentence

     data['tweet_aug'] = data['tweet'].apply(lambda x: synonym_replacement(x, 3))
     data = pd.concat([data, data[['tweet_aug', 'class']].rename(columns={'tweet_aug': 'tweet'})], axis=0)
```

Figure 19: Data Augmentation

5.3 Advanced Tokenization

```
# Advanced Tokenization
tokenizer = RobertaTokenizerFast.from_pretrained('roberta-base')
train_encodings = tokenizer(X_train.tolist(), truncation=True, padding=True, max_length=128)
test_encodings = tokenizer(X_test.tolist(), truncation=True, padding=True, max_length=128)

train_dataset = tf.data.Dataset.from_tensor_slices((dict(train_encodings), y_train)).shuffle(len(X_train)).batch(8)
test_dataset = tf.data.Dataset.from_tensor_slices((dict(test_encodings), y_test)).batch(8)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens)
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100% 25.0/25.0 [00:00<00:00, 1.89kB/s]
vocab.json: 100% 899k/899k [00:00<00:00, 3.83MB/s]
merges.txt: 100% 456k/456k [00:00<00:00, 2.14MB/s]
tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 6.20MB/s]
config.json: 100% 481/481 [00:00<00:00, 39.0kB/s]
```

Figure 20: Tokenization

5.4 Defining function and running optuna

```
[ ] # Define objective function for Optuna
    def objective(trial):
        learning_rate = trial.suggest_loguniform('learning_rate', 1e-6, 1e-4)
        num_train_epochs = trial.suggest_int('num_train_epochs', 1, 3)
        batch_size = trial.suggest_categorical('batch_size', [8, 16])
        optimizer_name = trial.suggest_categorical('optimizer', ['adam', 'adamw'])

        def create_model_with_scheduler():
            model = TFRobertaForSequenceClassification.from_pretrained('roberta-base', num_labels=len(label_encoder.classes_))
            steps_per_epoch = len(X_train) // batch_size
            num_train_steps = steps_per_epoch * num_train_epochs
            optimizer, schedule = create_optimizer(init_lr=learning_rate, num_train_steps=num_train_steps, num_warmup_steps=0)
            model.compile(optimizer=optimizer, loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])
            return model

        model = create_model_with_scheduler()

        model.fit(train_dataset, epochs=num_train_epochs, validation_data=test_dataset)

        loss, accuracy = model.evaluate(test_dataset)
        return accuracy
```

```
[ ] # Run Optuna optimization
    study = optuna.create_study(direction='maximize')
    study.optimize(objective, n_trials=3)

    best_params = study.best_params
    best_score = study.best_value

    print(f"Best params: {best_params}")
    print(f"Best score: {best_score}")
```

Figure 21: Code for Defining function and running optuna.

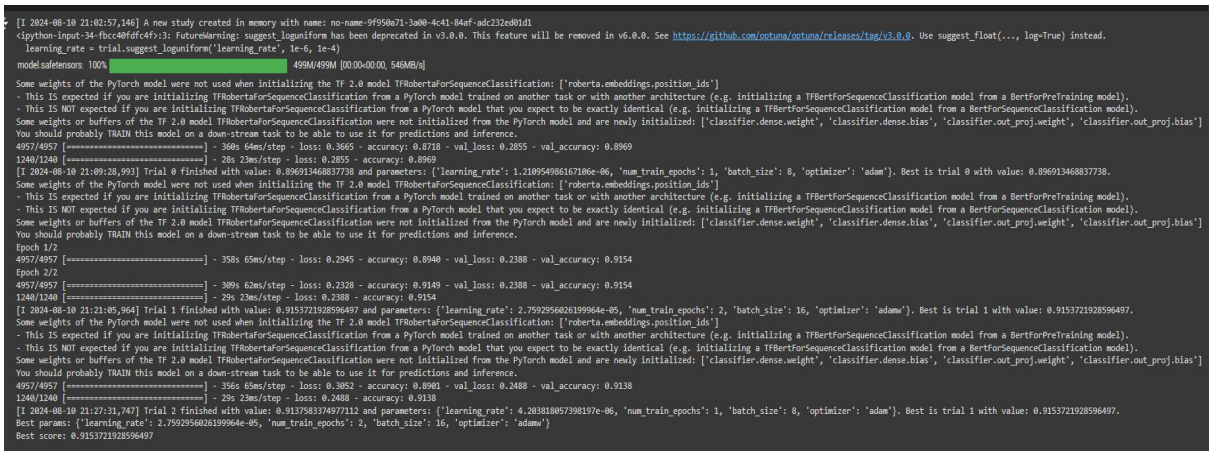

Figure 22: Output for Optuna

5.5Training the final model based on the best parameters found out by optuna.

Figure 23: Final model output

5.6 Save the model and Evaluate the final model on test dataset.

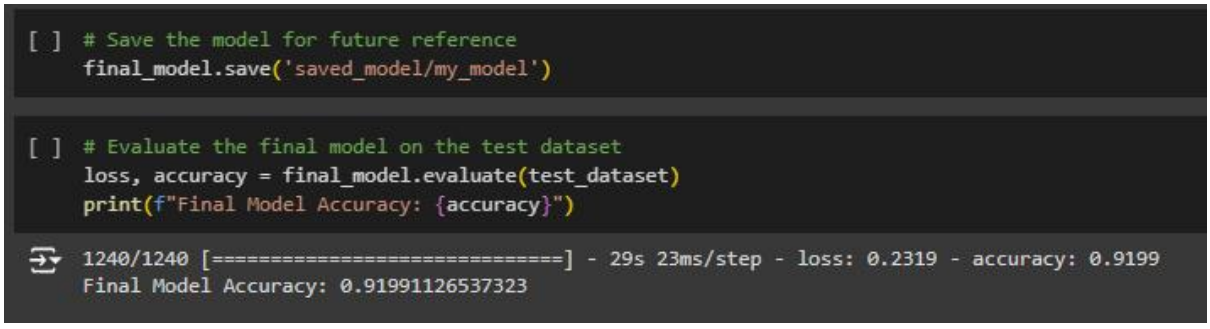

Figure 24: Final model accuracy

5.7 F1-Score for the final model

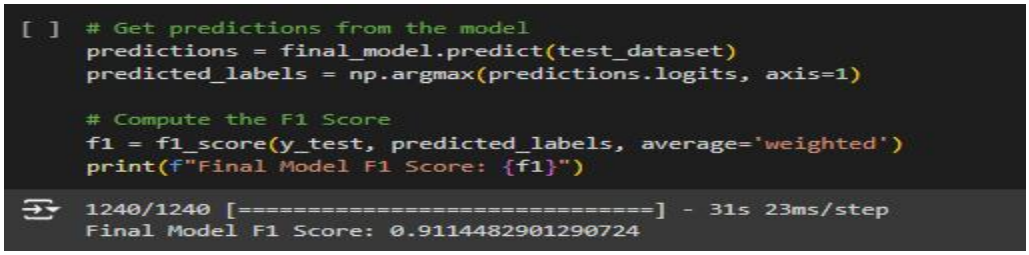

Figure 25: Final model F1-Score

References

distilbert/distilbert-base-uncased · Hugging Face (2024). Available at: https://huggingface.co/distilbert/distilbert-base-uncased (Accessed: 11 August 2024).

FacebookAI/xlm-roberta-base · Hugging Face (no date). Available at: https://huggingface.co/FacebookAI/xlm-roberta-base (Accessed: 11 August 2024).

Hate Speech and Offensive Language Dataset (no date). Available at: https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset (Accessed: 10 August 2024).