Configuration Manual

MSc Research Project
Data Analytics

Daksh Sharma
Student ID: 22165665

School of Computing
National College of Ireland

Supervisor: Mrs. Harshani Nagahamulla

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Daksh Sharma……………………………………………………………………………………

Student ID: ………22165665…………………………………………………………………………………..……

Programme: ……Data Analytics……… **Year:** …2023-2024

Module: ..….MSc. Research Project……………………………………………………………………

Lecturer: ……14/12/2023……………………………………………………………………………….………

Submission Due Date: …………………………………………………………………………………………………….……
…

Project Title: …. Opinion Mining using Twitter data for Ukraine-Russia War……………

Word Count: ………619……………………… **Page Count:** …………………7…………..……..………

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: …………Daksh Sharma…………………………………………………………………………

Date: ………14/12/2023………………………………………………………………………………

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

Daksh Sharma
Student ID: 22165665

1 Introduction

The configuration manual serves to guide users in replicating and understanding the experimental setup of this project on the topic – Opinion Mining using Twitter data for Ukraine-Russia war, detailing software, data processing steps, and model configurations for transparency and reproducibility. This article encapsulates the information about the required libraries, system specifications, models used, and code execution.

2 System Specification

This research project was created on the system with following configurations –

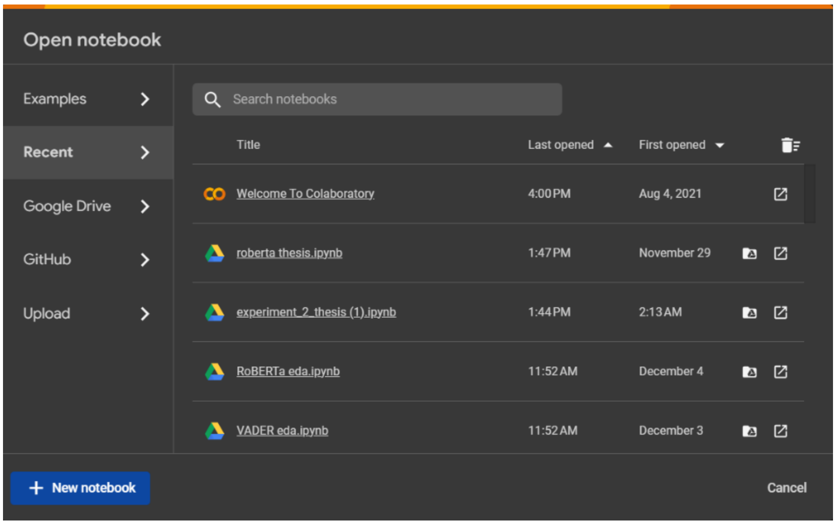
- Operating System: Windows 11 – 64 bits
- Processor: Intel i5 11th Gen
- RAM: 8 GB
- Hard Drive: 512 GB

3 Software Tools

The following software tools were used in the project –

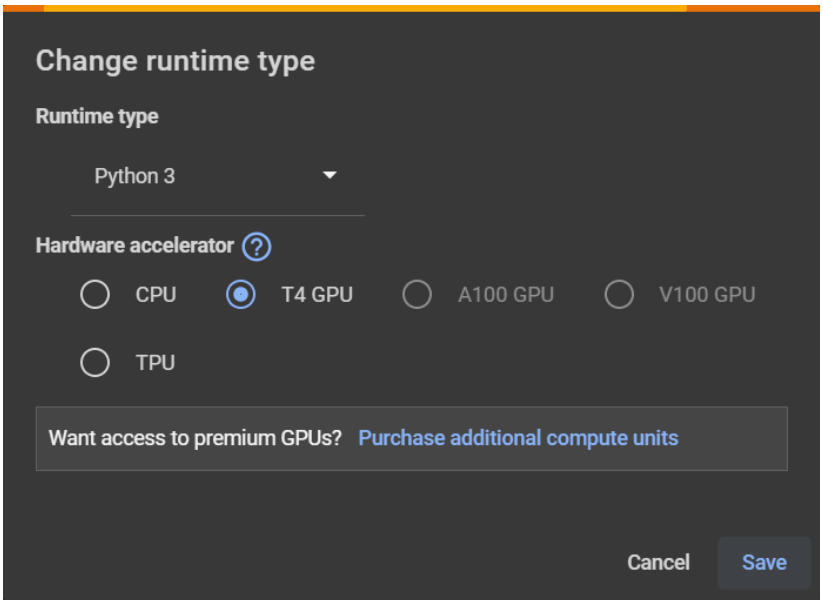
- Python 3.10.12
- Google Colab
  Google Colab can be accessed by –
  https://colab.google/

  To run a .ipynb file, you can open a downloaded file or create a new file as follows –



Title	Last opened	First opened
Welcome To Colaboratory	4:00 PM	Aug 4, 2021
roberta thesis.ipynb	1:47 PM	November 29
experiment_2_thesis (1).ipynb	1:44 PM	2:13 AM
RoBERTa eda.ipynb	11:52 AM	December 4
VADER eda.ipynb	11:52 AM	December 3

- To run the python code using GPU instead of CPU –



Change runtime type

Runtime type

Python 3

Hardware accelerator

○ CPU ● T4 GPU ○ A100 GPU ○ V100 GPU

○ TPU

Want access to premium GPUs? Purchase additional compute units

4 Required Libraries

The following libraries are to be installed to run the code –

- Numpy 1.23.5 – Numerical computing library for Python, providing efficient array operations and mathematical functions.

- Pandas 1.5.3 – Data manipulation library, offering data structures like DataFrame for easy handling and analysis of structured data.

- Emoji 2.9.0 – Python library for handling emojis, facilitating their encoding, decoding, and manipulation in text strings.

- Nltk 3.8.1 – Natural Language Toolkit for Python, providing tools for text processing, tokenization, and more.

- Seaborn 0.12.2 – Data visualization library based on Matplotlib, simplifying the creation of informative and attractive statistical graphics.

- Matplotlib 3.7.1 – Comprehensive plotting library for Python, generating static, animated, and interactive visualizations across various formats.

- Tensorflow 2.14.0 – Open-source machine learning framework, facilitating the development and training of deep learning models.

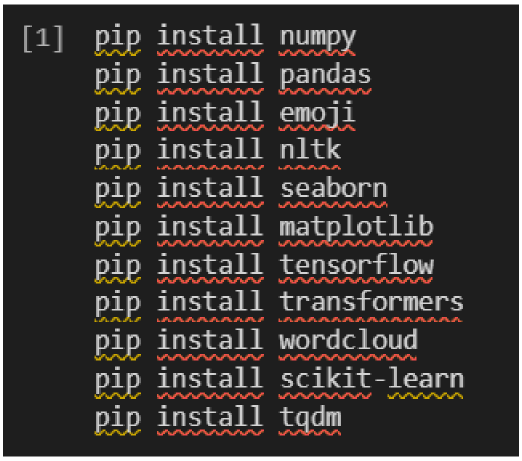
- Transformers 4.35.2 – Hugging Face’s library for state-of-the-art natural language processing models, including BERT and GPT.

- Wordcloud 1.9.2 – Python library for creating word clouds from text data, visually representing word frequency in a graphical manner.

- Scikit-learn 1.2.2 – Machine learning library for Python which facilitates tasks like train_test_split, CountVectorizer, and metrics assessment for classification models.

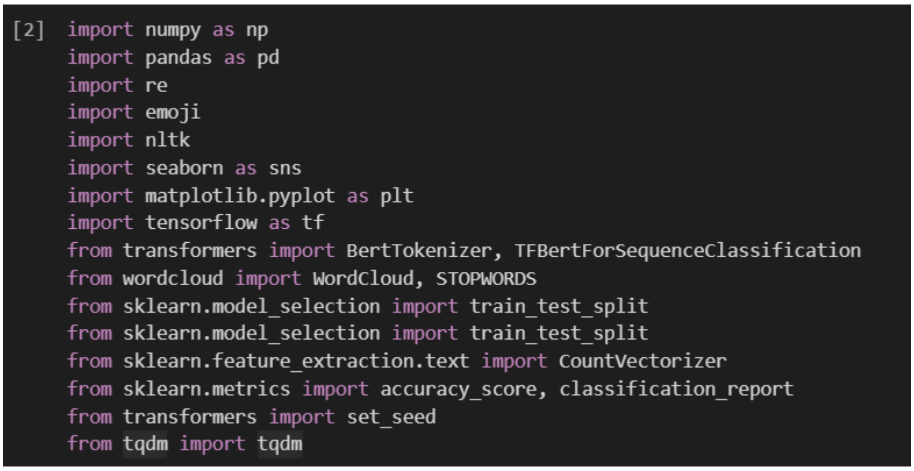
- Tqdm 4.66.1 – Fast, extensible progress bar library for Python, providing visual feedback on the progress of iterations or tasks.

These libraries are needed to be installed before the code execution and can be installed as -



```
[1]  pip install numpy
     pip install pandas
     pip install emoji
     pip install nltk
     pip install seaborn
     pip install matplotlib
     pip install tensorflow
     pip install transformers
     pip install wordcloud
     pip install scikit-learn
     pip install tqdm
```

After installation, these libraries have to be imported as follows –



```
[2]  import numpy as np
     import pandas as pd
     import re
     import emoji
     import nltk
     import seaborn as sns
     import matplotlib.pyplot as plt
     import tensorflow as tf
     from transformers import BertTokenizer, TFBertForSequenceClassification
     from wordcloud import WordCloud, STOPWORDS
     from sklearn.model_selection import train_test_split
     from sklearn.model_selection import train_test_split
     from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.metrics import accuracy_score, classification_report
     from transformers import set_seed
     from tqdm import tqdm
```

5 Dataset Source

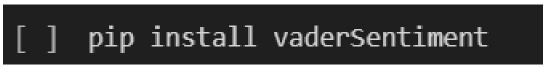
The dataset used in this project has been downloaded from Kaggle. The “Russia vs Ukraine Tweets Dataset (Daily Updated)” can be downloaded using the following URL –

https://www.kaggle.com/datasets/towhidultonmoy/russia-vs-ukraine-tweets-datasetdaily-updated/?select=filename.csv

6 Models Used

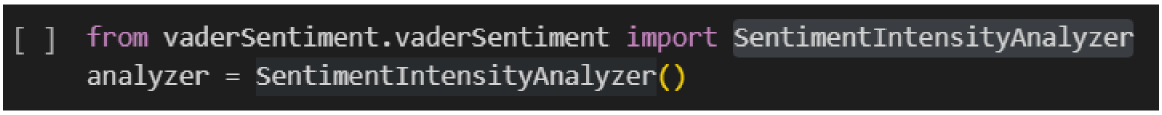
6.1 VADER

To install the VADER sentiment analysis tool, use the pip install method –



```
[ ]  pip install vaderSentiment
```

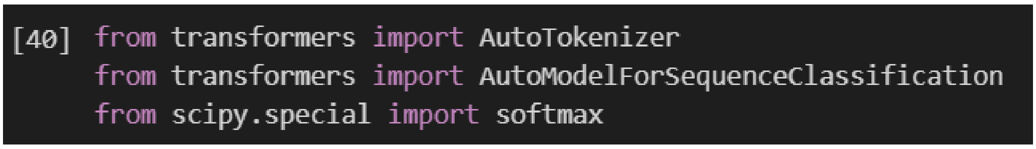
After installation, import ‘SentimentIntensityAnalyzer’, and create an instance of the analyzer-



```
[ ]  from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
     analyzer = SentimentIntensityAnalyzer()
```

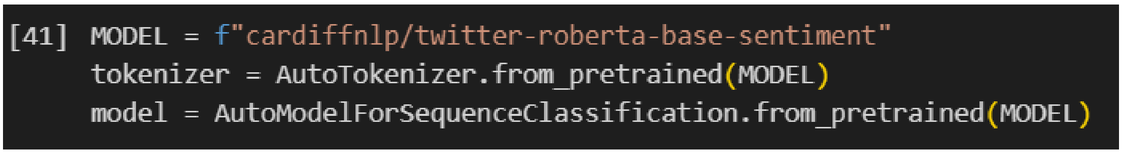
6.2 RoBERTa

From ‘Transformers’ library, import the following –



```
[40]  from transformers import AutoTokenizer
      from transformers import AutoModelForSequenceClassification
      from scipy.special import softmax
```

Then, we need to download the model from the Hugging Face’s website using the following set of code –

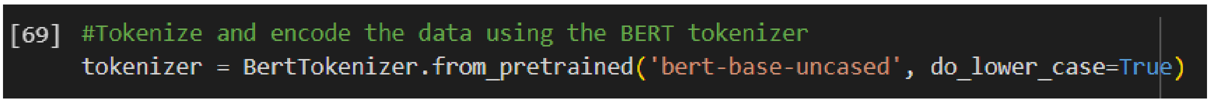


```
[41]  MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
      tokenizer = AutoTokenizer.from_pretrained(MODEL)
      model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

6.3 BERT base

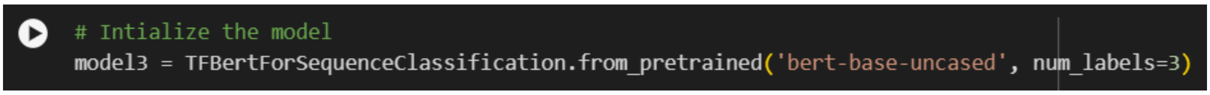
The ‘BertTokenizer’, ‘TFBertForSequenceClassification’ has already been imported from the ‘Transformers’ library in Section 4.

To tokenize and encode the data using BERT tokenizer, use the following code –



```
[69]  #Tokenize and encode the data using the BERT tokenizer
      tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
```

To initialize the model –



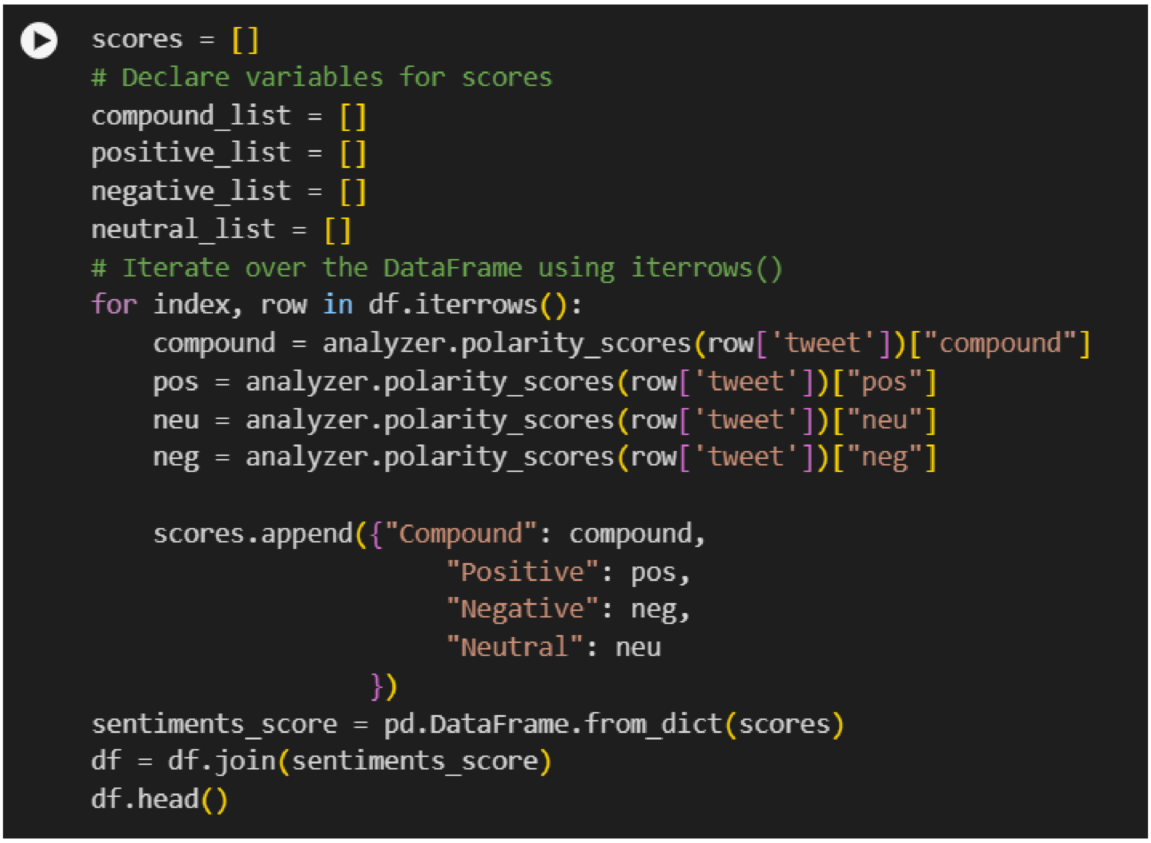
```
# Intialize the model
model3 = TFBertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=3)
```

7 Code Execution

This section contains few of the important parts of the code and the method to run them –

7.1 VADER Sentiment Analysis

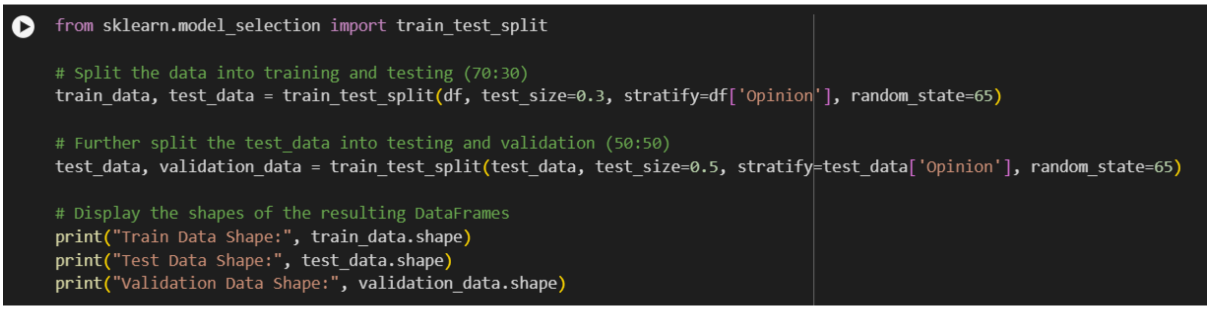
- The polarity scores using VADER are assigned to the tweets using the following code-



```
scores = []
# Declare variables for scores
compound_list = []
positive_list = []
negative_list = []
neutral_list = []
# Iterate over the DataFrame using iterrows()
for index, row in df.iterrows():
    compound = analyzer.polarity_scores(row['tweet'])["compound"]
    pos = analyzer.polarity_scores(row['tweet'])["pos"]
    neu = analyzer.polarity_scores(row['tweet'])["neu"]
    neg = analyzer.polarity_scores(row['tweet'])["neg"]

    scores.append({"Compound": compound,
                   "Positive": pos,
                   "Negative": neg,
                   "Neutral": neu
               })
sentiments_score = pd.DataFrame.from_dict(scores)
df = df.join(sentiments_score)
df.head()
```

- The train, test, and validation splits were performed in the following ratio –



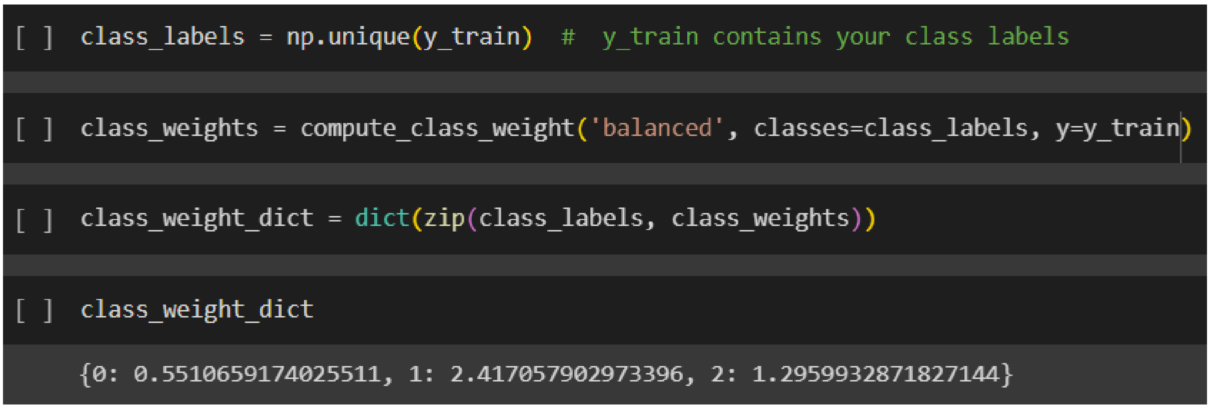
```
from sklearn.model_selection import train_test_split

# Split the data into training and testing (70:30)
train_data, test_data = train_test_split(df, test_size=0.3, stratify=df['Opinion'], random_state=65)

# Further split the test_data into testing and validation (50:50)
test_data, validation_data = train_test_split(test_data, test_size=0.5, stratify=test_data['Opinion'], random_state=65)

# Display the shapes of the resulting DataFrames
print("Train Data Shape:", train_data.shape)
print("Test Data Shape:", test_data.shape)
print("Validation Data Shape:", validation_data.shape)
```

- To tackle the problem of class imbalance, we have used class weights –



```
[ ]  class_labels = np.unique(y_train)  #  y_train contains your class labels
```

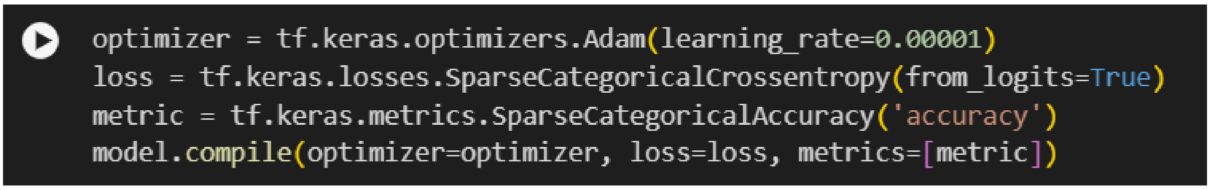
```
[ ]  class_weights = compute_class_weight('balanced', classes=class_labels, y=y_train)
```

```
[ ]  class_weight_dict = dict(zip(class_labels, class_weights))
```

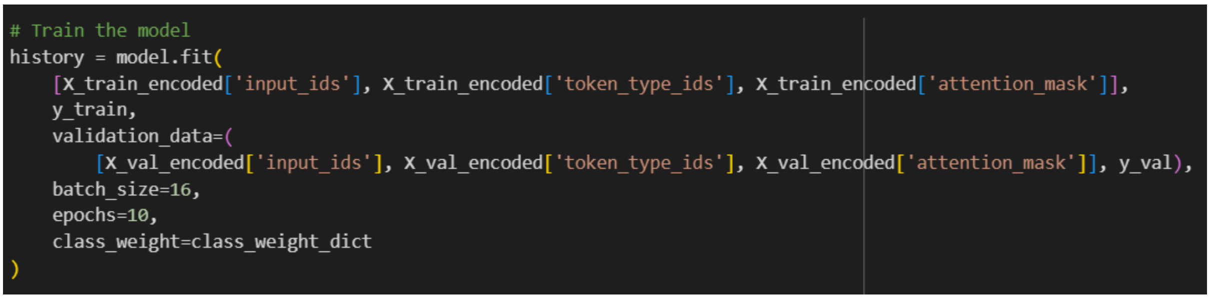
```
[ ]  class_weight_dict

     {0: 0.5510659174025511, 1: 2.417057902973396, 2: 1.2959932871827144}
```

- To train the BERT model, we fine-tune the following parameters –



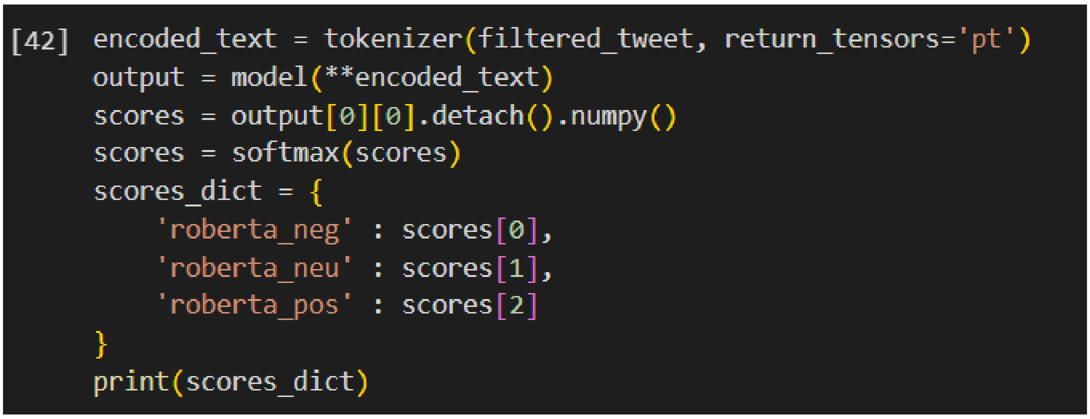
```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.00001)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
model.compile(optimizer=optimizer, loss=loss, metrics=[metric])
```



```
# Train the model
history = model.fit(
    [X_train_encoded['input_ids'], X_train_encoded['token_type_ids'], X_train_encoded['attention_mask']],
    y_train,
    validation_data=(
        [X_val_encoded['input_ids'], X_val_encoded['token_type_ids'], X_val_encoded['attention_mask']], y_val),
    batch_size=16,
    epochs=10,
    class_weight=class_weight_dict
)
```

7.2 RoBERTa

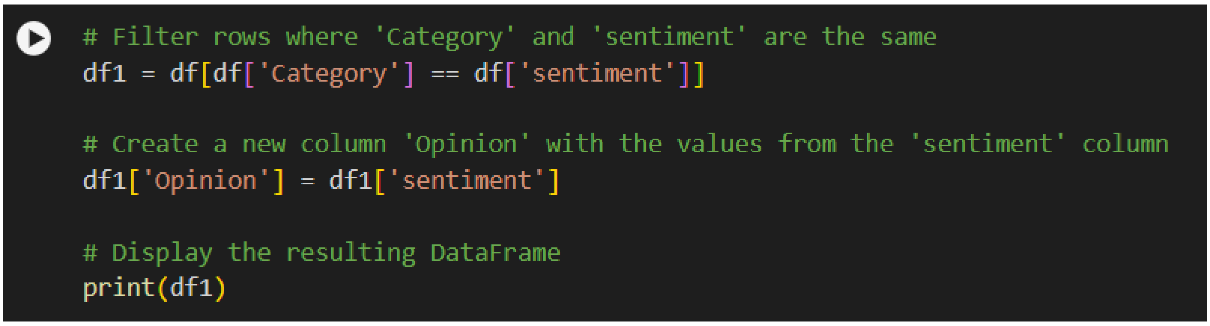
- To extract sentiment score from a filtered tweet using RoBERTa, we use the following code –



```
[42] encoded_text = tokenizer(filtered_tweet, return_tensors='pt')
     output = model(**encoded_text)
     scores = output[0][0].detach().numpy()
     scores = softmax(scores)
     scores_dict = {
         'roberta_neg' : scores[0],
         'roberta_neu' : scores[1],
         'roberta_pos' : scores[2]
     }
     print(scores_dict)
```

7.3 Approach 2 – Using both VADER and RoBERTa to obtain true labels

- After labelling the data using VADER and RoBERTa, we use the following code to combine the outputs of these two approaches and obtain a dataset with true labels –



```
# Filter rows where 'Category' and 'sentiment' are the same
df1 = df[df['Category'] == df['sentiment']]

# Create a new column 'Opinion' with the values from the 'sentiment' column
df1['Opinion'] = df1['sentiment']

# Display the resulting DataFrame
print(df1)
```