Configuration Manual

MSc Research Project
Msc Data Analytics

Snehal Ransing
Student ID: x19200714

School of Computing
National College of Ireland

Supervisor: Mr. Prashanth Nayak

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	…Snehal Nagnath Ransing……………………………………………………		
Student ID:	…x19200714…………………………………………………………………………….……		
Programme:	Msc Data Analytics……………………………	**Year:**	…2021-2022..
Module:	MSc Research Project………………………………………………………………….………		
Lecturer:	Mr. Prashanth Nayak…………………………………………………………………….………		
Submission Due Date:	…31-01-2023………………………………………………………………………….………		
Project Title:	Brand Reviews of e-wallet applications using Twitter sentiments		
Word Count:	……………………………………	**Page Count:**	…………………………………….…….………

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: ……Snehal Nagnath Ransing…………………………………………………………

Date: ……15-12-2022………………………………………………………………………

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

Forename Surname
Student ID:

1 Introduction

The paper includes comprehensive information on the equipment and programs used to support the project's development from beginning to end. This setup manual paper, which is included with the research project report, enables readers to comprehend the study better. Therefore, any technical details necessary for project completion but are not permitted to be included in the report are discussed here.

2 Hardware and Software requirement

To study the architecture of Machine learning classification models (Linear SVC, Random Forest, KNN Algorithm, Logistic Regression) and BERT, RoBERTa model we have used software. The software and hardware necessary to complete the work are thus described here.2.1Software used

Table 1: Software used

Tools used for programming	Anaconda navigator,Jupyter Notebook, Google Colab
Tools used to build the report	MS. Excel, MS. PowerPoint and MS. Word.
Programing Language used	Python.
Data storage	Google Drive, GitHub, Local system

2.2 Hardware required

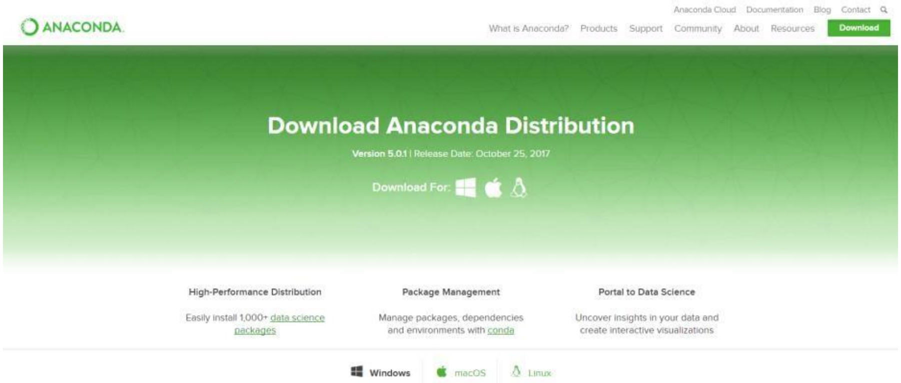
Table 2: Hardware used

System	**Specification**
Operating System	Windows 10 pro
Processor	Intel core i5-7th Gen
RAM	8 GB
System type	64-bit OS, x64-based processor
Graphic card	NVidia GeForce

3 Software installation

3.1 Steps to install Anaconda navigator and Jupyter Notebook On windows.

1) Go to the downloads page for Anaconda.[1]. Visit the following website: Anaconda.com/downloads



Download Anaconda Distribution

Version 5.0.1 | Release Date: October 25, 2017

High-Performance Distribution

Easily install 1,000+ data science packages

Package Management

Manage packages, dependencies and environments with conda

Portal to Data Science

Uncover insights in your data and create interactive visualizations

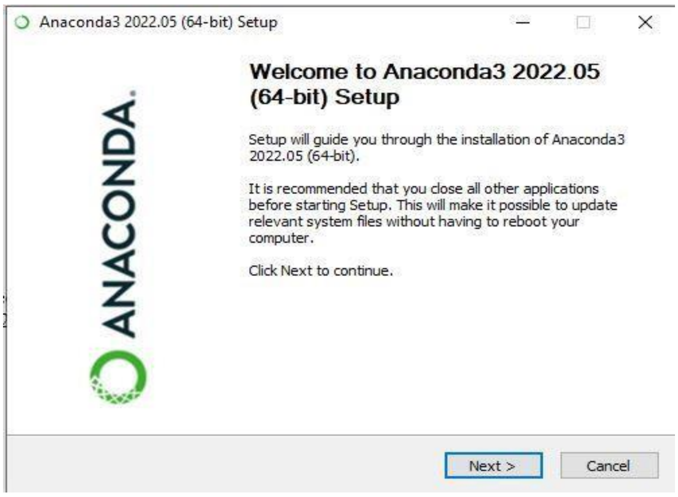
Figure 1: The Anaconda Downloads Page will look something like this

2) Choose Windows. The three operating systems are offered when you choose Windows, as seen in figure 2 below.

Figure 2: Select window option

3) Then the .Exe file gets downloaded.
4) After downloading the .exe file, we need to install the anaconda in the system. To do this we follow the below step

[1] https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/


Anaconda3 2022.05 (64-bit) Setup

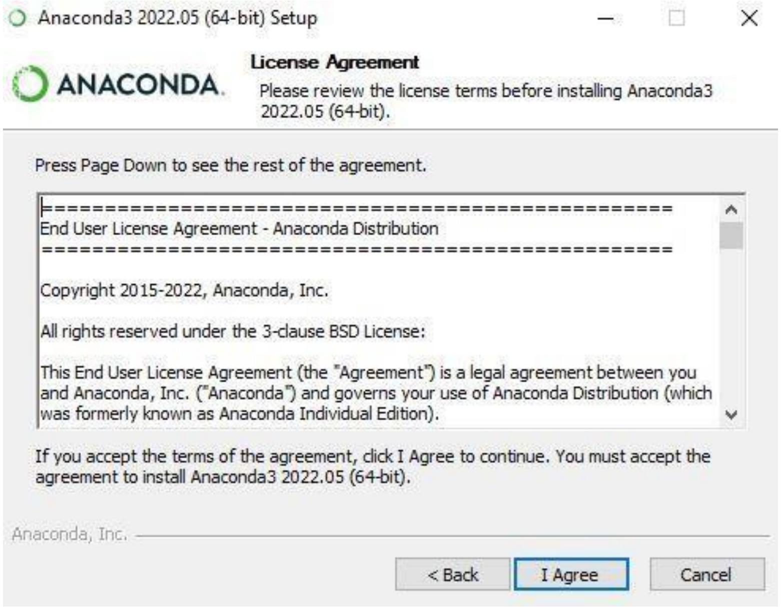
Welcome to Anaconda3 2022.05 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2022.05 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 3: Installation window


Anaconda3 2022.05 (64-bit) Setup

License Agreement

Please review the license terms before installing Anaconda3 2022.05 (64-bit).

Press Page Down to see the rest of the agreement.

```
==================================================
End User License Agreement - Anaconda Distribution
==================================================
```

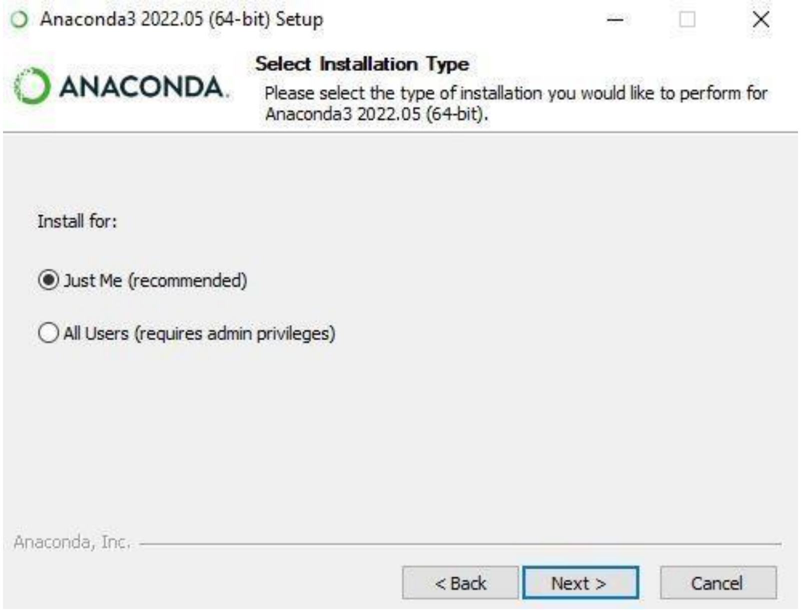
Copyright 2015-2022, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

This End User License Agreement (the "Agreement") is a legal agreement between you and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Distribution (which was formerly known as Anaconda Individual Edition).

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2022.05 (64-bit).

Figure 4: License agreement window



Anaconda3 2022.05 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2022.05 (64-bit).

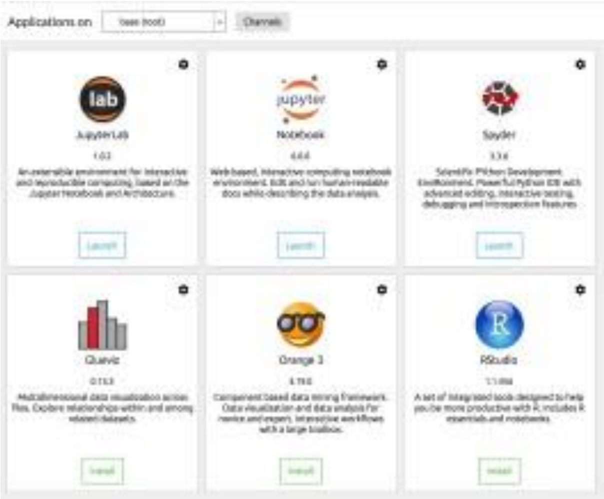
Install for:

● Just Me (recommended)

○ All Users (requires admin privileges)

Figure 5: Selection of installing type window

5) After choosing the user in the aforementioned figure, click next and wait for Anaconda to fully install on your machine.
6) After the installation is finished, run the program from the Start menu to see a screen similar to the one in the accompanying image. By default, JupyterLab and Jupyter Notebook are installed.



An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Multidimensional data visualization across files. Explore relationships within and among related datasets.

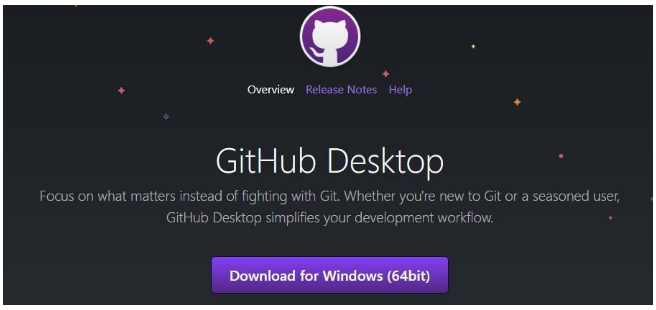
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 6: Anaconda interface

3.2 Installation of GitHub desktop

i. Click on this link https://desktop.github.com/ and Select Windows Download[2].

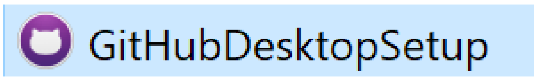


GitHub Desktop

Focus on what matters instead of fighting with Git. Whether you're new to Git or a seasoned user, GitHub Desktop simplifies your development workflow.

Figure 6: GitHub download page

ii. After downloading the setup file, we must install the setup.



GitHubDesktopSetup

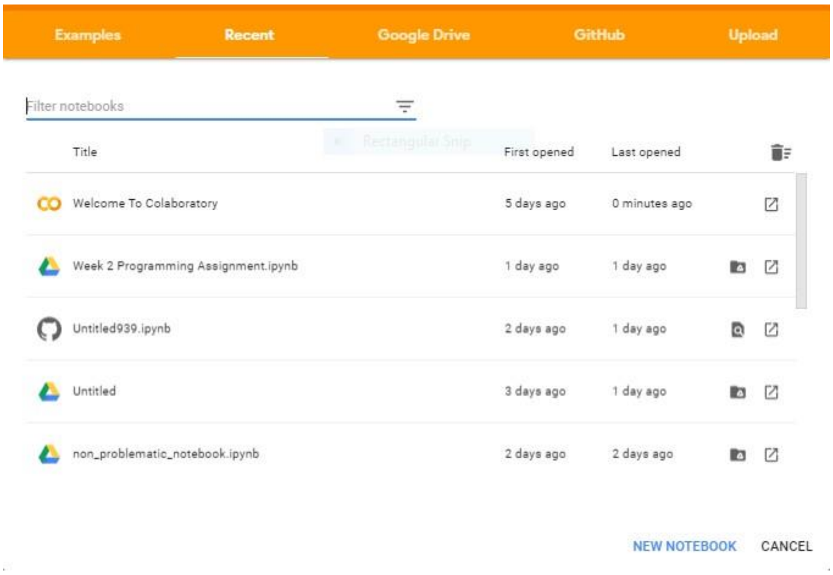
iii. Following the successful installation of GitHub. launch of the application. We also need to create a repository where we can store all of the files and share them with everyone.

3.3 Google colaboratory

[2] https://www.techrepublic.com/article/how-to-install-github-desktop/

Google Colab has being utilized for additional programming-related parts. Considering its advantages, including a free GPU, keep notebooks on Google Drive. Additionally, it can work with Github and local memory, which is helpful while utilizing it. Additionally, the programming notebook may be stored straight to without having to install it on the PC.

Go to https://colab.research.google.com/ to create a new notebook on Colab. It will automatically display your previous notebooks and offer you the option to start a new one[3].



Title	First opened	Last opened
Welcome To Colaboratory	5 days ago	0 minutes ago
Week 2 Programming Assignment.ipynb	1 day ago	1 day ago
Untitled939.ipynb	2 days ago	1 day ago
Untitled	3 days ago	1 day ago
non_problematic_notebook.ipynb	2 days ago	2 days ago

Figure 6: google colab interface to for new working notebook

The fact that Colab provides free GPU and TPU support is its greatest perk. You may choose GPU or TPU for your software by selecting Runtime > Change runtime type. may contribute to accelerating the runtime.

[3] https://www.kdnuggets.com/2020/06/google-colab-deep-learning.html

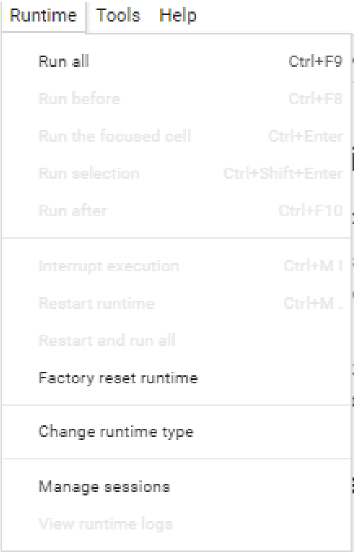


Figure 7: Changing runtime

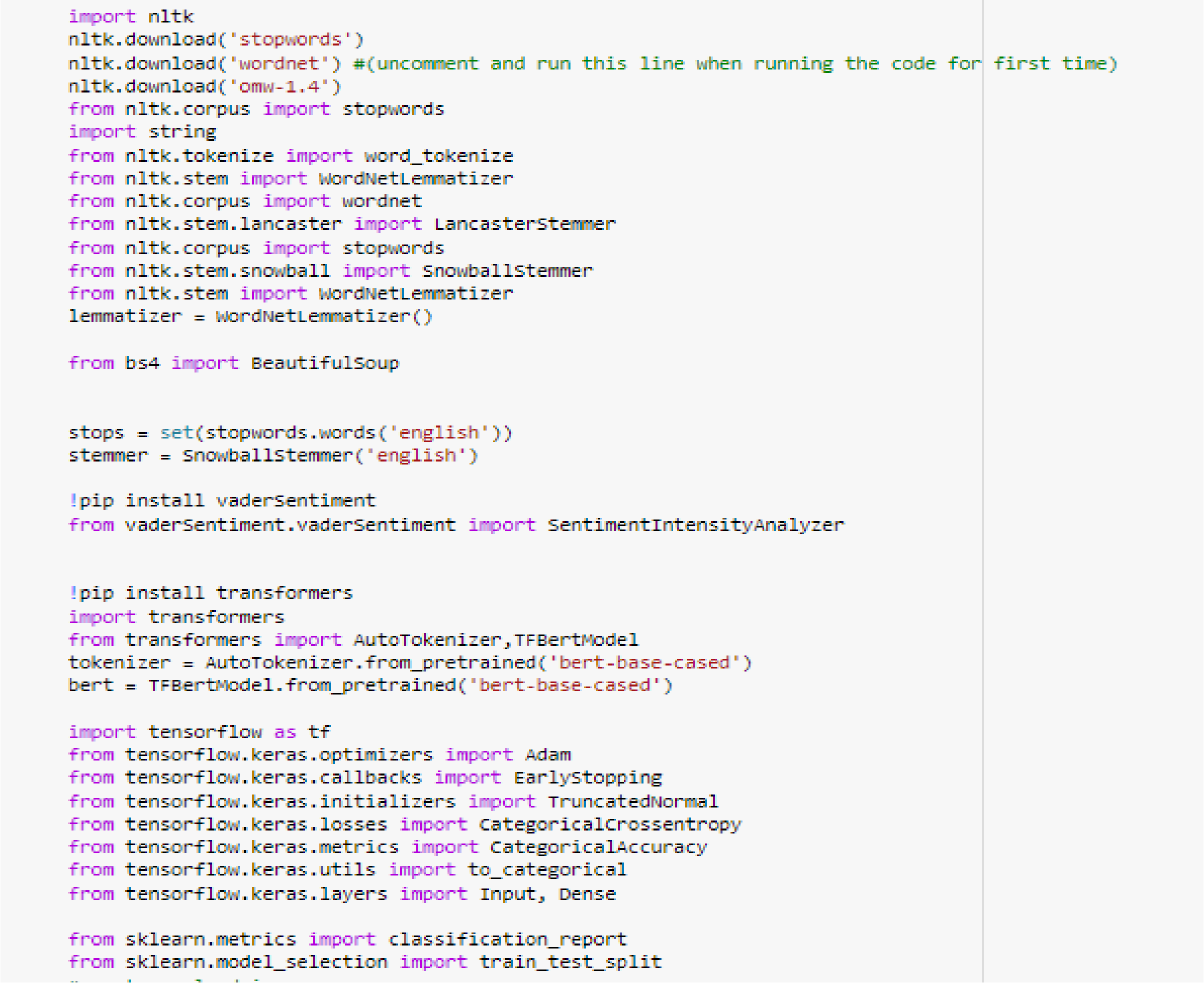
3.4 Microsoft Word, Microsoft PowerPoint, Microsoft excel

The report is put together using Microsoft Word, Microsoft PowerPoint, and Microsoft Excel, which also help with the production of graphs and the presentation of the research project. All of these programs, which assist with the writing portion of the research, are convenient to use and simple to comprehend.

Figure 8: Microsoft tool used for report building

4 Python Libraries used

For the deep learning and Machine learning task and for the Exploratory data Analysis different libraries are used in python that are showed in figure 9 below.



```
import nltk
nltk.download('stopwords')
nltk.download('wordnet') #(uncomment and run this line when running the code for first time)
nltk.download('omw-1.4')
from nltk.corpus import stopwords
import string
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.stem.lancaster import LancasterStemmer
from nltk.corpus import stopwords
from nltk.stem.snowball import SnowballStemmer
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

from bs4 import BeautifulSoup


stops = set(stopwords.words('english'))
stemmer = SnowballStemmer('english')

!pip install vaderSentiment
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer


!pip install transformers
import transformers
from transformers import AutoTokenizer,TFBertModel
tokenizer = AutoTokenizer.from_pretrained('bert-base-cased')
bert = TFBertModel.from_pretrained('bert-base-cased')

import tensorflow as tf
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.initializers import TruncatedNormal
from tensorflow.keras.losses import CategoricalCrossentropy
from tensorflow.keras.metrics import CategoricalAccuracy
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Input, Dense

from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

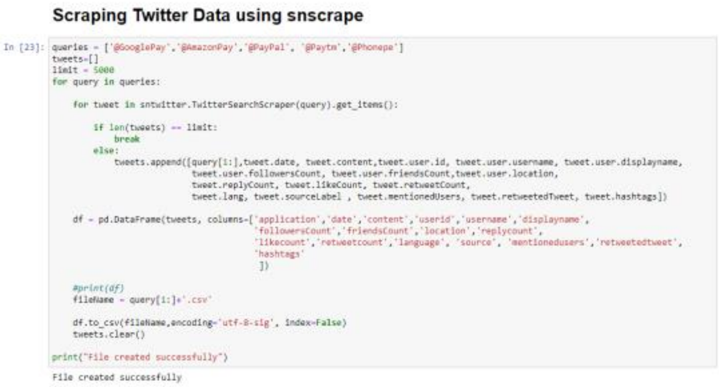
Figure 9: Python libraries used

Libraries	**Version**
Sklearn	1.0.2
Pandas	1.41
Matplotlib	3.5.1
Tensorflow	2.8.0
Numpy	1.12.2
matplotlib	3.0.0

Table 3. Python Libraries Version

5 Data Understanding and Pre-processing Step

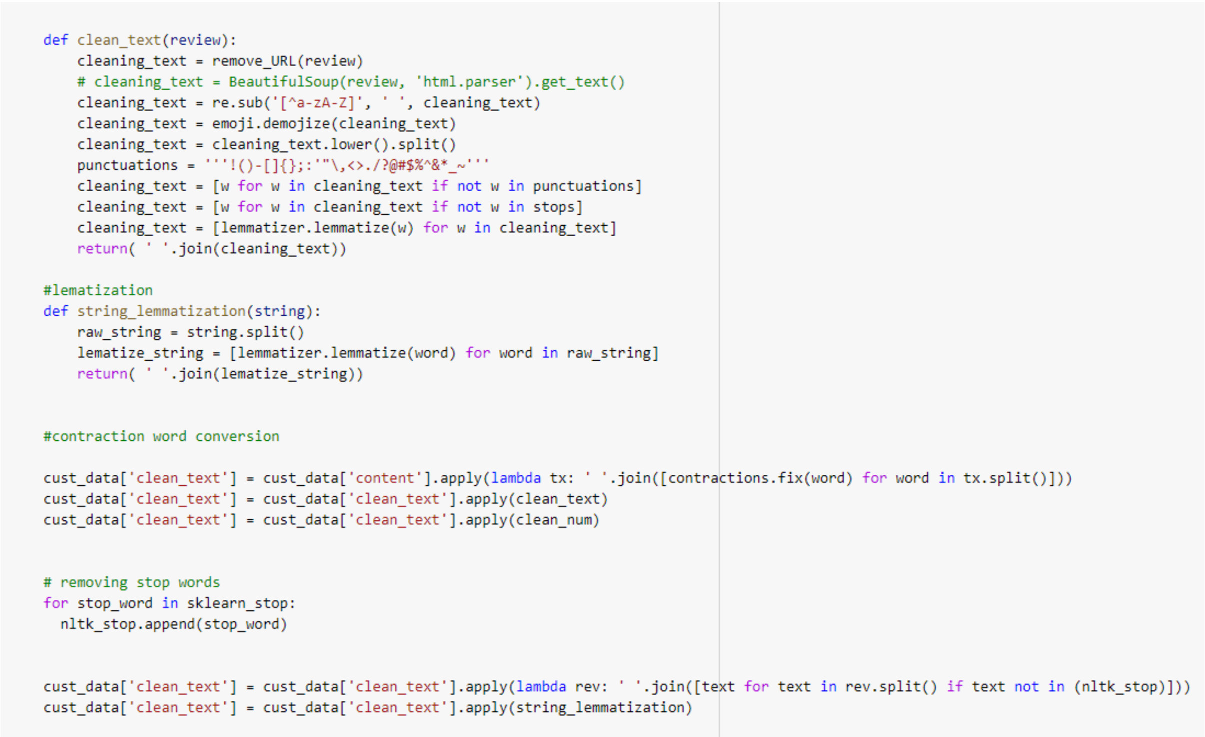
- The dataset is obtained by scraping tweets directly from twitter
- Nature of dataset is raw with multiple anomalies.
- Technique used to scrape data is my using @mentions (@GooglePay, @Phonepe, @Paytm, @AmazonPay, @PayPal)
- The shape of dataset is Row=30000,Columns=17
- 6000 records present for each brand



Scraping Twitter Data using snscrape

```
In [23]: queries = ['@GooglePay','@AmazonPay','@PayPal', '@Paytm','@Phonepe']
tweets=[]
limit = 5000
for query in queries:

    for tweet in sntwitter.TwitterSearchScraper(query).get_items():

        if len(tweets) == limit:
            break
        else:
            tweets.append([query[1:],tweet.date, tweet.content,tweet.user.id, tweet.user.username, tweet.user.displayname,
                           tweet.user.followersCount, tweet.user.friendsCount,tweet.user.location,
                           tweet.replyCount, tweet.likeCount, tweet.retweetCount,
                           tweet.lang, tweet.sourceLabel , tweet.mentionedUsers, tweet.retweetedTweet, tweet.hashtags])

    df = pd.DataFrame(tweets, columns=['application','date','content','userid','username','displayname',
                                       'followersCount','friendsCount','location','replycount',
                                       'likecount','retweetcount','language', 'source', 'mentionedusers','retweetedtweet',
                                       'hashtags'
                                        ])

    #print(df)
    fileName = query[1:]+'.csv'

    df.to_csv(fileName,encoding='utf-8-sig', index=False)
    tweets.clear()

print("File created successfully")
File created successfully
```

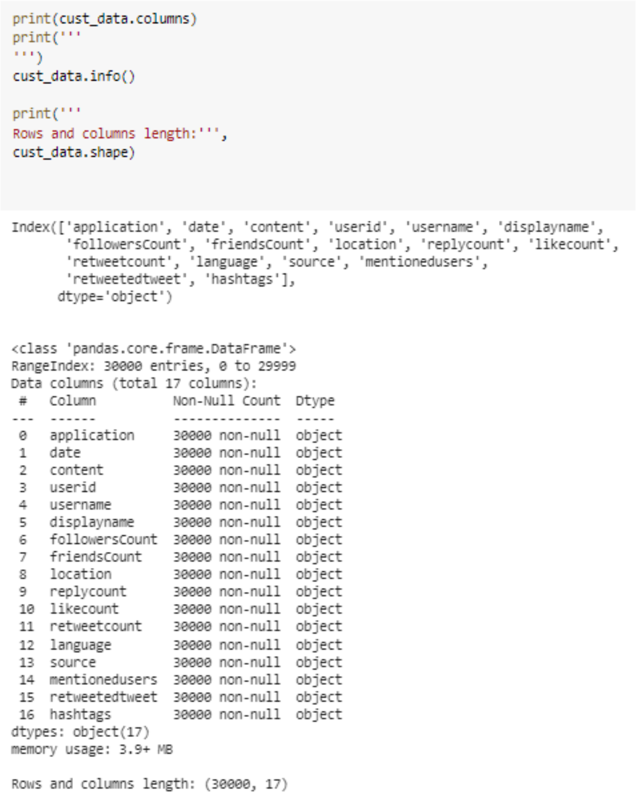
Figure 10. Twitter Data Extraction



```
def clean_text(review):
    cleaning_text = remove_URL(review)
    # cleaning_text = BeautifulSoup(review, 'html.parser').get_text()
    cleaning_text = re.sub('[^a-zA-Z]', ' ', cleaning_text)
    cleaning_text = emoji.demojize(cleaning_text)
    cleaning_text = cleaning_text.lower().split()
    punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
    cleaning_text = [w for w in cleaning_text if not w in punctuations]
    cleaning_text = [w for w in cleaning_text if not w in stops]
    cleaning_text = [lemmatizer.lemmatize(w) for w in cleaning_text]
    return( ' '.join(cleaning_text))

#lematization
def string_lemmatization(string):
    raw_string = string.split()
    lematize_string = [lemmatizer.lemmatize(word) for word in raw_string]
    return( ' '.join(lematize_string))


#contraction word conversion

cust_data['clean_text'] = cust_data['content'].apply(lambda tx: ' '.join([contractions.fix(word) for word in tx.split()]))
cust_data['clean_text'] = cust_data['clean_text'].apply(clean_text)
cust_data['clean_text'] = cust_data['clean_text'].apply(clean_num)


# removing stop words
for stop_word in sklearn_stop:
  nltk_stop.append(stop_word)


cust_data['clean_text'] = cust_data['clean_text'].apply(lambda rev: ' '.join([text for text in rev.split() if text not in (nltk_stop)]))
cust_data['clean_text'] = cust_data['clean_text'].apply(string_lemmatization)
```

Figure 10. Data Preprocessing

- Data Preprocessing involves following:
- Cleaning the text by removing Punctuations, #, @mentions
- Removing contractions and emojis.
- Removing the Stopwords, lemmatizing

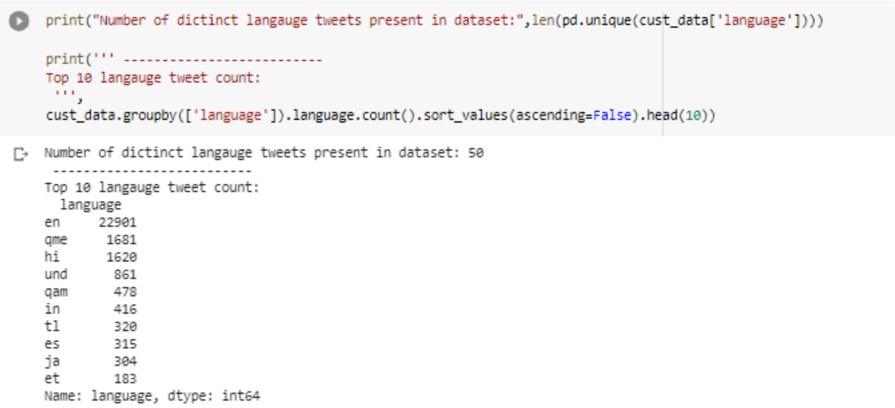


```
print(cust_data.columns)
print('''
''')
cust_data.info()

print('''
Rows and columns length:''',
cust_data.shape)
```

```
Index(['application', 'date', 'content', 'userid', 'username', 'displayname',
       'followersCount', 'friendsCount', 'location', 'replycount', 'likecount',
       'retweetcount', 'language', 'source', 'mentionedusers',
       'retweetedtweet', 'hashtags'],
      dtype='object')


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   application     30000 non-null  object
 1   date            30000 non-null  object
 2   content         30000 non-null  object
 3   userid          30000 non-null  object
 4   username        30000 non-null  object
 5   displayname     30000 non-null  object
 6   followersCount  30000 non-null  object
 7   friendsCount    30000 non-null  object
 8   location        30000 non-null  object
 9   replycount      30000 non-null  object
 10  likecount       30000 non-null  object
 11  retweetcount    30000 non-null  object
 12  language        30000 non-null  object
 13  source          30000 non-null  object
 14  mentionedusers  30000 non-null  object
 15  retweetedtweet  30000 non-null  object
 16  hashtags        30000 non-null  object
dtypes: object(17)
memory usage: 3.9+ MB

Rows and columns length: (30000, 17)
```

Figure 11. EDA

- Exploratory data analysis is performed to see the characteristics of various attributes like username, location, language and sentiment columns.
- There are 30000 rows with 17 columns.
- Languages used: 50



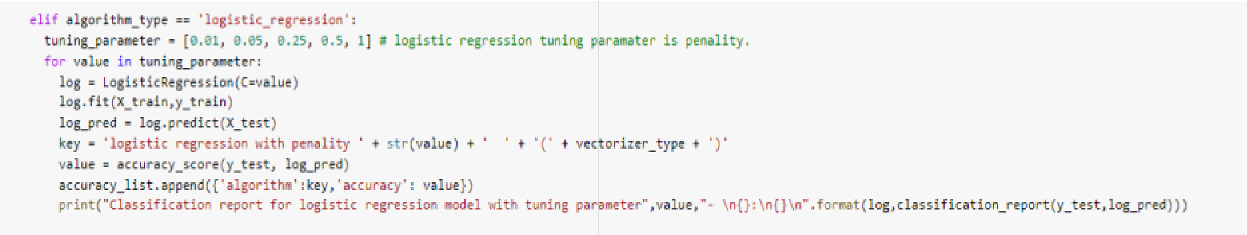
```
print("Number of dictinct langauge tweets present in dataset:",len(pd.unique(cust_data['language'])))

print(''' -------------------------
Top 10 langauge tweet count:
 ''',
cust_data.groupby(['language']).language.count().sort_values(ascending=False).head(10))
```

```
Number of dictinct langauge tweets present in dataset: 50
 -------------------------
Top 10 langauge tweet count:
  language
en     22901
qme     1681
hi      1620
und      861
qam      478
in       416
tl       320
es       315
ja       304
et       183
Name: language, dtype: int64
```

Figure 12. Language used in Dataset

Logistic Regression:

To stay away from the gamble of over fitting the model, parameter tuning is performed by passing the value of 24 C as [0.01, 0.05, 0.25, 0.5, 1]. By Iterating over these values during the model execution, the best-fit model will be accomplished.

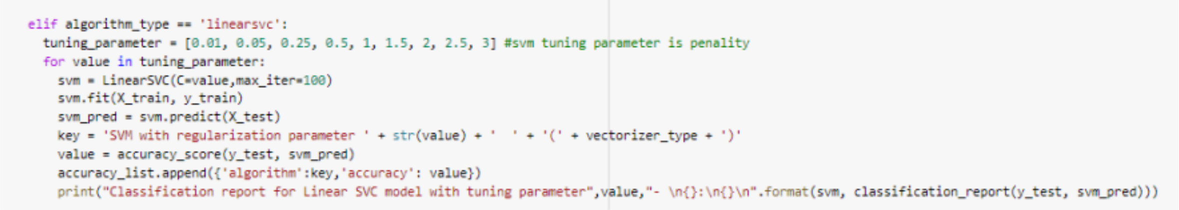


```
elif algorithm_type == 'logistic_regression':
  tuning_parameter = [0.01, 0.05, 0.25, 0.5, 1] # logistic regression tuning paramater is penality.
  for value in tuning_parameter:
    log = LogisticRegression(C=value)
    log.fit(X_train,y_train)
    log_pred = log.predict(X_test)
    key = 'logistic regression with penality ' + str(value) + '  ' + '(' + vectorizer_type + ')'
    value = accuracy_score(y_test, log_pred)
    accuracy_list.append({'algorithm':key,'accuracy': value})
    print("Classification report for logistic regression model with tuning parameter",value,"- \n{}:\n{}\n".format(log,classification_report(y_test,log_pred)))
```

Figure 13. Logistic Regression

Linear SVC:

Post Evaluating Linear SVC model, the results obtained were quite remarkable in accuracy. For Count vectorizer the accuracy is maximum approaching at 93% for C=1 and maximum iteration set to 100. For tf—idf also the accuracy is 91% for C=1 and for n-gram count vectorizer the accuracy is 89%

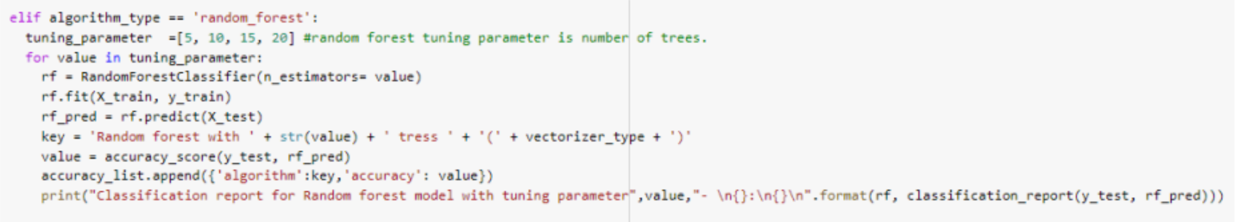


```
elif algorithm_type == 'linearsvc':
  tuning_parameter = [0.01, 0.05, 0.25, 0.5, 1, 1.5, 2, 2.5, 3] #svm tuning parameter is penality
  for value in tuning_parameter:
    svm = LinearSVC(C=value,max_iter=100)
    svm.fit(X_train, y_train)
    svm_pred = svm.predict(X_test)
    key = 'SVM with regularization parameter ' + str(value) + '  ' + '(' + vectorizer_type + ')'
    value = accuracy_score(y_test, svm_pred)
    accuracy_list.append({'algorithm':key,'accuracy': value})
    print("Classification report for Linear SVC model with tuning parameter",value,"- \n{}:\n{}\n".format(svm, classification_report(y_test, svm_pred)))
```

Figure 14. Linear SVC

Random Forest:

The values used are [5, 10, 15, 20], the values in the list signifies the number of trees the model is considering at the time of single execution. Post evaluating the Random Forest model, the accuracy for 20 estimators is 83% for count vectorizer, 81% for tf-idf vectorizer and 81% for n-gram count vectorizer.

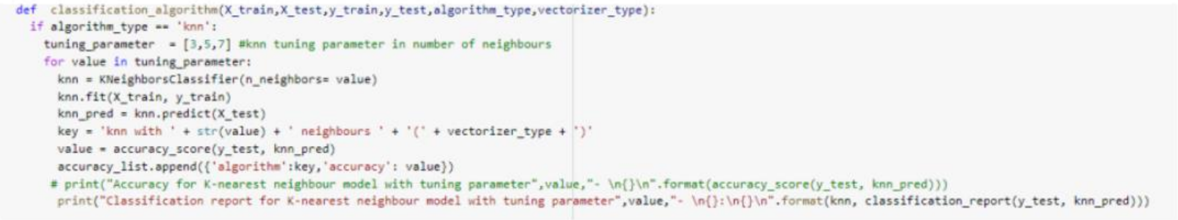


```
elif algorithm_type == 'random_forest':
  tuning_parameter  =[5, 10, 15, 20] #random forest tuning parameter is number of trees.
  for value in tuning_parameter:
    rf = RandomForestClassifier(n_estimators= value)
    rf.fit(X_train, y_train)
    rf_pred = rf.predict(X_test)
    key = 'Random forest with ' + str(value) + ' tress ' + '(' + vectorizer_type + ')'
    value = accuracy_score(y_test, rf_pred)
    accuracy_list.append({'algorithm':key,'accuracy': value})
    print("Classification report for Random forest model with tuning parameter",value,"- \n{}:\n{}\n".format(rf, classification_report(y_test, rf_pred)))
```

Figure 15. Random Forest

knn-algorithm:

The data with comparable sort of values are isolated together and named as one, and remaining information focuses are similarly isolated and marked. Here, in this study, the value for k is taken as [3,5,7]. The justification for picking all the odd adjoining point is fundamentally to try not to any kind of get between two distinct classes incorrect.

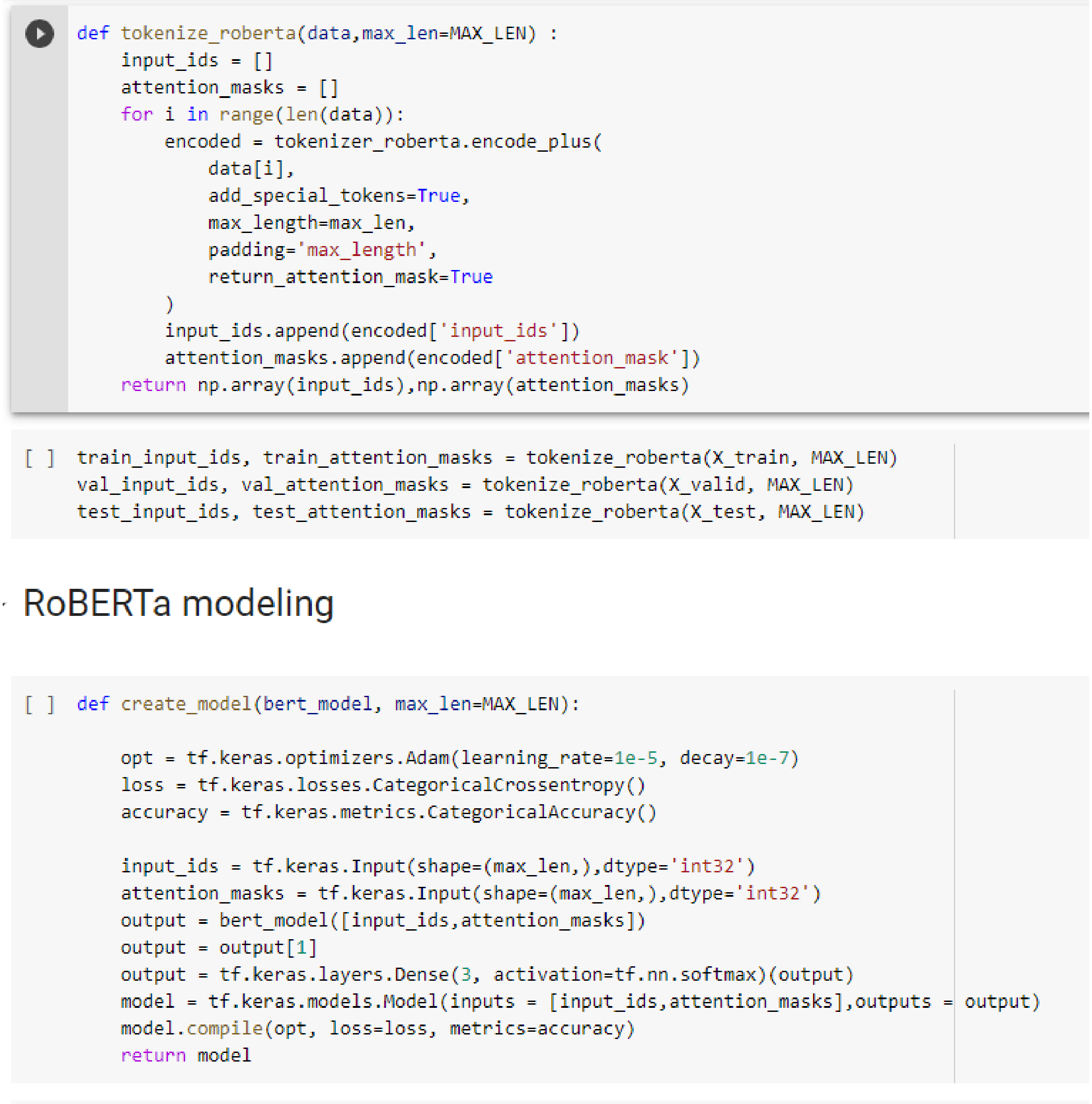


```
def  classification_algorithm(X_train,X_test,y_train,y_test,algorithm_type,vectorizer_type):
  if algorithm_type == 'knn':
    tuning_parameter  = [3,5,7] #knn tuning parameter in number of neighbours
    for value in tuning_parameter:
      knn = KNeighborsClassifier(n_neighbors= value)
      knn.fit(X_train, y_train)
      knn_pred = knn.predict(X_test)
      key = 'knn with ' + str(value) + ' neighbours ' + '(' + vectorizer_type + ')'
      value = accuracy_score(y_test, knn_pred)
      accuracy_list.append({'algorithm':key,'accuracy': value})
     # print("Accuracy for K-nearest neighbour model with tuning parameter",value,"- \n{}\n".format(accuracy_score(y_test, knn_pred)))
      print("Classification report for K-nearest neighbour model with tuning parameter",value,"- \n{}:\n{}\n".format(knn, classification_report(y_test, knn_pred)))
```

Figure 16. knn-algorithm

RoBERTa:

With the use of a dynamic masking technique called RoBERTa, the BERT pre-trained model's next sentence prediction is removed. The RoBERTa model, is an advance on the BERT model masking method.



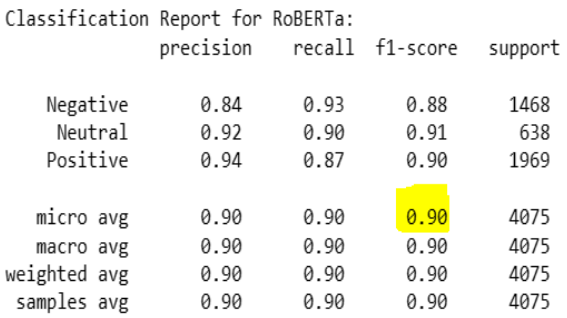
```
def tokenize_roberta(data,max_len=MAX_LEN) :
    input_ids = []
    attention_masks = []
    for i in range(len(data)):
        encoded = tokenizer_roberta.encode_plus(
            data[i],
            add_special_tokens=True,
            max_length=max_len,
            padding='max_length',
            return_attention_mask=True
        )
        input_ids.append(encoded['input_ids'])
        attention_masks.append(encoded['attention_mask'])
    return np.array(input_ids),np.array(attention_masks)
```

```
train_input_ids, train_attention_masks = tokenize_roberta(X_train, MAX_LEN)
val_input_ids, val_attention_masks = tokenize_roberta(X_valid, MAX_LEN)
test_input_ids, test_attention_masks = tokenize_roberta(X_test, MAX_LEN)
```

RoBERTa modeling

```
def create_model(bert_model, max_len=MAX_LEN):

    opt = tf.keras.optimizers.Adam(learning_rate=1e-5, decay=1e-7)
    loss = tf.keras.losses.CategoricalCrossentropy()
    accuracy = tf.keras.metrics.CategoricalAccuracy()

    input_ids = tf.keras.Input(shape=(max_len,),dtype='int32')
    attention_masks = tf.keras.Input(shape=(max_len,),dtype='int32')
    output = bert_model([input_ids,attention_masks])
    output = output[1]
    output = tf.keras.layers.Dense(3, activation=tf.nn.softmax)(output)
    model = tf.keras.models.Model(inputs = [input_ids,attention_masks],outputs = output)
    model.compile(opt, loss=loss, metrics=accuracy)
    return model
```

Figure 17. RoBERTa

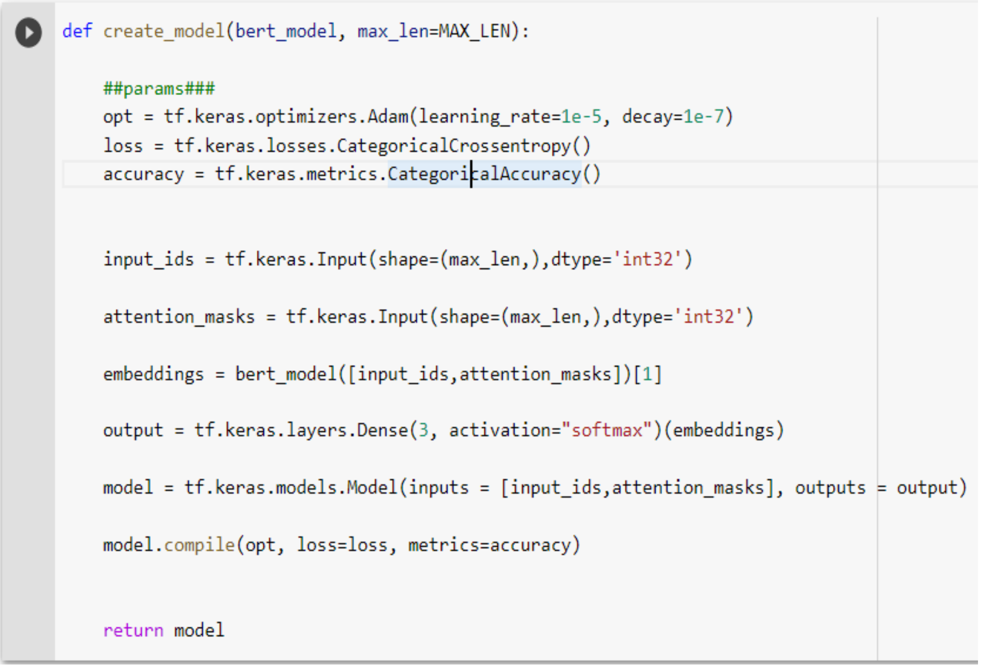


```
Classification Report for RoBERTa:
              precision    recall  f1-score   support

    Negative       0.84      0.93      0.88      1468
     Neutral       0.92      0.90      0.91       638
    Positive       0.94      0.87      0.90      1969

   micro avg       0.90      0.90      0.90      4075
   macro avg       0.90      0.90      0.90      4075
weighted avg       0.90      0.90      0.90      4075
 samples avg       0.90      0.90      0.90      4075
```

Figure 18. Classification report for RoBERTa

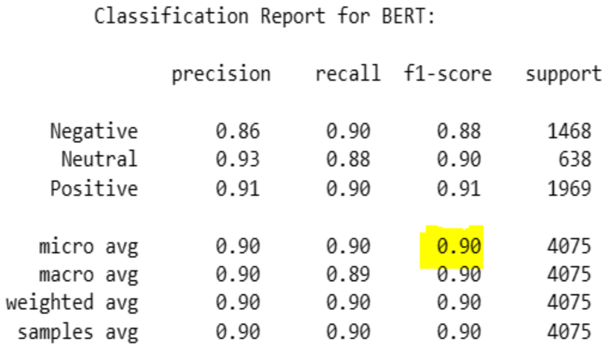
BERT:

After performing some tests, by using one hot encoding on the target variable we achieved higher accuracy. For this reason, have chosen one hot encoding over label encoding and resulted in better accuracy. Then, have created a custom function to host the pre trained BERT model, and attach to it a 3 neurons output layer, necessary to perform the classification of the 3 different classes of the dataset (the 3 emotions). After evaluating the BERT model it’s observed that the accuracy for the BERT validation dataset is 90%. This result is good accurate as depicted by other machine learning models.



```
def create_model(bert_model, max_len=MAX_LEN):

    ##params###
    opt = tf.keras.optimizers.Adam(learning_rate=1e-5, decay=1e-7)
    loss = tf.keras.losses.CategoricalCrossentropy()
    accuracy = tf.keras.metrics.CategoricalAccuracy()


    input_ids = tf.keras.Input(shape=(max_len,),dtype='int32')

    attention_masks = tf.keras.Input(shape=(max_len,),dtype='int32')

    embeddings = bert_model([input_ids,attention_masks])[1]

    output = tf.keras.layers.Dense(3, activation="softmax")(embeddings)

    model = tf.keras.models.Model(inputs = [input_ids,attention_masks], outputs = output)

    model.compile(opt, loss=loss, metrics=accuracy)


    return model
```

Figure 19. BERT



```
Classification Report for BERT:

              precision    recall  f1-score   support

    Negative       0.86      0.90      0.88      1468
     Neutral       0.93      0.88      0.90       638
    Positive       0.91      0.90      0.91      1969

   micro avg       0.90      0.90      0.90      4075
   macro avg       0.90      0.89      0.90      4075
weighted avg       0.90      0.90      0.90      4075
 samples avg       0.90      0.90      0.90      4075
```

Figure 20. Classification report for BERT

We can see that both the algorithms performed well on the classification task, with performance scores around 90%

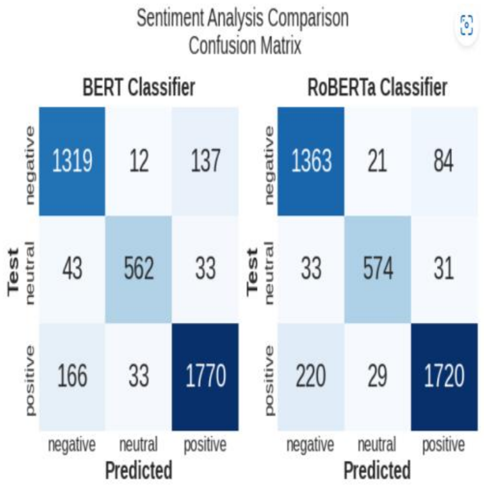


Figure 21. Sentiment Analysis Comparison Confusion Matrix