

National  
College of  
Ireland

# Configuration Manual

MSc Research Project  
Data Analytics

Ravi Sahal  
Student ID: x21161984

School of Computing  
National College of Ireland

Supervisor: Dr. Catherine Mulwa

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



**Student Name:** Ravi Sahal  
**Student ID:** x21161984  
**Programme:** Data Analytics **Year:** 2022 - 2023  
**Module:** MSc Research Project  
**Lecturer:** Dr. Catherine Mulwa  
**Submission Due Date:** 15<sup>th</sup> December 2022  
**Project Title:** Predicting Optimal Cryptocurrency using Social Media Sentimental Analysis  
**Word Count:** 877 **Page Count:** 9

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:** .....

**Date:** 15th December 2022

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Ravi Sahal  
Student ID: x21161984

## 1 Introduction

Implementation of a framework that combines contextual-based embedding ELMo Embedding with a recurrent neural network and recommends to investors a list of the top 10 cryptocurrencies to maximize their profits. This configuration manual includes information on the system configuration, software, and hardware requirements, and the procedures used to accomplish the Research Project.

## 2 System Configuration

The system configuration that was used to carry out the project is described in this part of the configuration manual.

### 2.1 Hardware Requirement

Operating System	macOS 13.0.1
Processor	Mac M1 Chip
RAM	8.00GB
Storage	256GB

### 2.2 Software Requirements

Programming Language	Python 3.9.15
Tools	Jupyter Notebook, PyCharm IDE, Excel

## 3 Project Development

The framework's development is primarily separated into two sections: the first focuses on the model's construction, and the second on its deployment on the console-based application.

### 3.1 Important Libraries

The implementation of this framework requires different libraries like pandas, numpy, seaborn, matplotlib, os, tensorflow, tensorflow\_hub, keras, sklearn, glob, datetime, nltk, time, json, and twint. The libraries of sklearn and TensorFlow was used for model development, calculation of model performance and loading the different neural network modules for which the relevant code snippets are mentioned in this sections.

Name	Last Modified	File size
..	seconds ago	
datasets	13 hours ago	
dataCleanTransform.ipynb	13 hours ago	21.4 kB
dataPreparationAndLabelling.ipynb	13 hours ago	28.7 kB
ELMoWithBiLSTM.ipynb	13 hours ago	115 kB
ELMoWithSimpleRNN.ipynb	13 hours ago	70.4 kB
clean_Dataset.csv	10 days ago	17.5 MB
coinData.json	10 days ago	803 B
cryptos.csv	22 days ago	188 MB
geckodriver	a month ago	4.54 MB
geckodriver.log	10 days ago	97.7 kB
main.py	14 hours ago	1.76 kB
Model_deployment.py	13 hours ago	3.88 kB
modelElmo.h5	3 days ago	77.2 MB
results.py	14 hours ago	449 B
scrapCoin.py	13 hours ago	900 B
simple_rnn.h5	5 days ago	9.88 MB
topTen.json	3 days ago	1.62 kB
Training_Dataset.csv	8 days ago	18.3 MB
tweets.py	13 hours ago	812 B

**Figure 1: Framework Overview**

### 3.2 Model Development

Data preparation, data transformation, and data modelling utilizing ELMo embedding with the recurrent neural network are all included in this section. These developments were all carried out using Jupyter Notebook.

#### Data Preparation

To prepare the training data for data modelling, the data preparation and transformation phase for model construction include data collection, cleaning, and transformation employing labelling of the cleaned and transformed data using the Flair framework and K-means Clustering. The figures below depict each phase of this development:

```

1 import twint
2
3 # This script is using Twint API to scrape the tweets
4
5 def fetchTweets(tags):
6     HashTags= tags
7     # search wit one by one hashtag from HashTags list
8     for tag in HashTags:
9         c = twint.Config() # get configuration
10        c.Search = tag # search hashtag
11        # c.Limit = 10000 # limit number of Tweets to 10000 for scraping
12        c.Store_csv = True # store tweets in a csv file
13        c.Hide_output=True
14        c.Pandas = True
15        c.Since = "2022-01-23"
16        c.Output = './cryptos.csv' # This is for training data acquisition
17        # c.Output = './'+ tags[0]+' .csv'
18        twint.run.Search(c)
19        import pandas as pd
20        df = pd.read_csv('./cryptos.csv', low_memory=False)
21        print(df.__len__())
22
23
24 fetchTweets(['crypto', 'Cryptocurrency', 'Bitcoin', 'ETH', 'digital currency']) #almost 4+hrs for scrapping tweets

```

**Figure 2: Scrapping Tweets for Model Training**

```

In [1]: import pandas as pd
import numpy as np
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
from nltk.stem import PorterStemmer
import string
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')

[nltk_data] Downloading package stopwords to
[nltk_data] /Users/ravishah/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

Out[1]: True

Data loading and preprocessing

In [2]: df = pd.read_csv('./cryptos.csv', usecols=['tweet', 'language'])
df.head()

Out[2]:
   tweet language
0  #bitnews crypto #ftc ftx attacker continues... en
1  @bunny228 FTX to refund people's money... FTX... en
2  Tell me @GaryGensler Do I own real shares of... en
3  DYDX resumes USDC Liquidity provision/ staking... en
4  (4) Digital asset brokerage Genesis received o... en

In [4]: df[df['language'] == 'en']
df['tweet'], df['language']

Out[4]:
0      #catnews #crypto #teeh FTX attacker continues...
1  @bunny228 FTX to refund people's money... FTX...
2  Tell me @GaryGensler Do I own real shares of...
3  DYDX resumes USDC Liquidity provision/ staking...
4  (4) Digital asset brokerage Genesis received o...

381838      ..
381838      @thehondr3ds GM +80M people! 🤩
381838      @cozyprompt the standards are back on the rise ...
381835      Good art from @basegofficial https://t.co/3h...
381836      Rings, Phillips of Cam Smith are a must @Lions
381838      Fun Thanksgiving Desserts for Kids - Food Netw...
Name: tweet, Length: 157182, dtype: object

In [5]: df.____len__()

Out[5]: 157182

```

Figure 3: Loading the Important Libraries and Visualization of the Data

```

In [9]: # Data cleaning for tweets
stop_words = set(stopwords.words('english'))
if 'not' in stop_words:
    print(stop_words.remove('not'))
def dataPreprocessing(tweet):
    cleanTweet = tweet.lower()
    pattern = re.compile(r'[\u4e00-\u9fff]+')
    if pattern.search(cleanTweet) != None:
        return

    cleanTweet = re.sub(r'@[A-Za-z0-9_+]|[\^\w\]|#|http\S+', '', cleanTweet)

    cleanTweet = cleanTweet.translate(str.maketrans('', '', string.punctuation))
    cleanTweet_tokens = word_tokenize(cleanTweet)
    # print('---', cleanTweet_tokens)
    # cleanTweet_words = [word for word in cleanTweet_tokens if word not in stop_words]
    if len(cleanTweet_tokens) < 5:
        return
    filtered_sentence = [w for w in cleanTweet_tokens if not w.lower() in stop_words]

    # stemObj = PorterStemmer()
    # stemWords = [stemObj.stem(word) for word in filtered_sentence]

    lemObj = WordNetLemmatizer()
    lemWords = [lemObj.lemmatize(word, pos='a') for word in filtered_sentence]

    return ' '.join(lemWords)

```

Figure 4: Data Cleaning Method

```

In [7]: from flair.models import TextClassifier
from flair.data import Sentence

classifier = TextClassifier.load('en-sentiment')
sentence = Sentence('The food was great!')
classifier.predict(sentence)

# print sentence with predicted labels
print('Sentence above is: ', sentence.labels)

2022-12-07 13:07:37,913 loading file /Users/ravishah/flair/models/sentiment-en-mix-distillbert_4.pt
Sentence above is: ['Sentence: "The food was great!" - POSITIVE' (0.9961)]

In [8]: # print sentence with predicted labels
print('Sentence above is: ', sentence)

Sentence above is: Sentences: "The food was great!" - POSITIVE (0.9961)

In [13]: def getSentiment(tweet):
# classifier = TextClassifier.load('en-sentiment')
sentence = Sentence(tweet)
classifier.predict(sentence)
sentScore = ''
if sentence.labels[0].value.lower() == 'positive':
    return '+' + str(sentence.labels[0].score[:4])
return '-' + str(sentence.labels[0].score[:4])

In [9]: type(sentence.labels[0].to_dict())

Out[9]: dict

In [26]: # Classifying labels with KMeans with K size =2
from sklearn.cluster import KMeans
labels = KMeans(n_clusters=2, random_state=0).fit(sentiment)
df['sentiment'] = labels.labels_

In [29]: df.head()

Out[29]:
   Unnamed: 0      cleanedTweets      tweet      sentimentScore      sentiment
0      0  catnews crypto tech ftx attacker continues...  #bitnews crypto #ftc ftx attacker continues...  0.550000      1
1      1  2288 ftx refund peoples money ftx value going...  @bunny228 FTX to refund people's money... FTX...  -0.990234      0
2      2  tell real shares anc lame ass ftx tokens irma...  Tell me @GaryGensler Do I own real shares of...  -0.990234      0
3      3  4 digital asset brokerage genesis received 1 b...  (4) Digital asset brokerage Genesis received o...  -0.990234      0
4      4  5 ftx employees paid ft token oof JUST IN: Some FTX employees were paid with the...  ftx employees paid ft token oof JUST IN: Some FTX employees were paid with the...  0.890157      1

In [30]: df['sentiment'].value_counts()

Out[30]:
0      38652
1      27801
Name: sentiment, dtype: int64

In [33]: df[df['sentimentScore'] == 0.5]['sentiment'].value_counts()

Out[33]:
1      27801
Name: sentiment, dtype: int64

In [34]: df.to_csv('./Training_Dataset.csv')

```

Figure 5: Data Labelling using Flair and K-means Clustering

## Data Modelling

The Elmo embedding is utilized to create the model, which uses two deep-learning models. This part covers model creation since the cleaned and labelled dataset is prepared for model training.

# 1. Implementation of ELMo with biLSTM model

```
In [1]: !pip install "tensorflow==1.7.0"
!pip install tensorflow-hub

Requirement already satisfied: tensorflow==1.7.0 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (1.7.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (from tensorflow==1.7.0) (1.46.3)
Requirement already satisfied: keras<2.10.0,>=2.9.0rc8 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (from tensorflow==1.7.0) (2.9.0)
Requirement already satisfied: keras-preprocessing==1.1.1 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (from tensorflow==1.7.0) (1.1.2)
Requirement already satisfied: astunparse==1.6.0 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (from tensorflow==1.7.0) (1.6.3)
Requirement already satisfied: tensorboard<2.10,>=2.9 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (from tensorflow==1.7.0) (2.9.0)
Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc8 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (from tensorflow==1.7.0) (2.9.0)
Requirement already satisfied: h5py<2.9.0 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (from tensorflow==1.7.0) (3.2.1)
Requirement already satisfied: flatbuffers<2,>=1.12 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (from tensorflow==1.7.0) (1.12)
Requirement already satisfied: numpy==1.20 in /Users/ravisaha/opt/anaconda3/lib/python3.9/site-packages (from tensorflow==1.7.0) (1.23.5)

In [2]: import tensorflow.compat.v1 as tf
#to make it 2.0 compatible with tf1.0 code, we disable the tf2.0 functionalities
tf.disable_eager_execution()

In [3]: # to hide warnings
import warnings
# warnings.filterwarnings('ignore')
warnings.filterwarnings('ignore', category=DeprecationWarning)

import tensorflow_hub as hub
import tensorflow as tf

elmo = hub.Module("https://tfhub.dev/google/elmo/2", trainable=True)

In [5]: import pandas as pd
In [6]: df= pd.read_csv('./Training_Dataset.csv')

In [7]: df.head()
Out [7]:
```

	Unnamed: 0	Unnamed: 0	cleanedTweets	tweet	sentimentScore	sentiment
0	0	0	catfishes crypto tech fix attacker continues see...	Roathres Kryptojtech FTX attacker continues...	0.52	1
1	1	1	2028 fix refund peoples money fix value going ...	@bunny228 FTX to refund people's money... FTX...	-0.99	0
2	2	2	tell real shares smc lame ass fix tokens imma ...	Tell me @GaryGensler Do i own real shares of...	-0.99	0
3	3	4	4 digital asset brokerage genesis received 1 b...	(4) Digital asset brokerage Genesis received o...	-0.99	0
4	4	5	ftx employees paid fit token oof	JUST IN: Some FTX employees were paid with the...	0.89	1

```
In [8]: df2=df[['cleanedTweets','sentiment']]
In [9]: df2.head()
Out [9]:
```

	cleanedTweets	sentiment
0	catfishes crypto tech fix attacker continues see...	1
1	2028 fix refund peoples money fix value going ...	0
2	tell real shares smc lame ass fix tokens imma ...	0
3	4 digital asset brokerage genesis received 1 b...	0
4	ftx employees paid fit token oof	1

Figure 6: Loading Important Packages and Preprocessed Dataset into Data frame

```
In [12]: from sklearn.preprocessing import LabelEncoder

In [13]: train_df, val_df, test_df = train_validate_test_split(df2)

print('Train: {}, Validation: {}, Test: {}'.format(train_df.shape, val_df.shape, test_df.shape))

print(colored('\nTRAIN DATA', 'magenta', attrs=['bold']))
display(train_df.head())

train_text = train_df['cleanedTweets'].tolist()
train_text = np.array(train_text, dtype=object)[:, np.newaxis]
# train_label = np.asarray(pd.get_dummies(train_df['sentiment']), dtype = np.int8)
train_label = train_df['sentiment'].values

val_text = val_df['cleanedTweets'].tolist()
val_text = np.array(val_text, dtype=object)[:, np.newaxis]
# val_label = np.asarray(pd.get_dummies(val_df['sentiment']), dtype = np.int8)
val_label = val_df['sentiment'].values

test_text = test_df['cleanedTweets'].tolist()
test_text = np.array(test_text, dtype=object)[:, np.newaxis]
# test_label = np.asarray(pd.get_dummies(test_df['sentiment']), dtype = np.int8)
test_label = test_df['sentiment'].values
test_label

Train: (39391, 2), Validation: (13130, 2), Test: (13132, 2)

TRAIN DATA
```

	cleanedTweets	sentiment
26706	craftingthefuture web2 web3 web3gaming minecra...	1
20884	eth price 117676 right compared last tweet pri...	0
34184	im arms cleo sol	1
15367	ftx claims owes 31 billion top 50 creditors re...	0
30881	cyberpunk solympus 271 purchased magic eden ra...	0

```
Out [13]: array([0, 0, 0, ..., 1, 0, 1])
```

Figure 7: Data Splitting and Model Creation

```
In [14]: import numpy as np
from tensorflow import keras
from tensorflow.keras.layers import Input, Lambda, Bidirectional, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras import layers
from tensorflow.python.keras import backend as K

In [15]: def ELMoEmbedding(input_text):
return elmo(tf.reshape(tf.cast(input_text, tf.string), [-1]), signature="default", as_dict=True)["elmo"]
def build_model():
input_layer = Input(shape=(1,), dtype="string", name="Input_layer")
embedding_layer = Lambda(ELMoEmbedding, output_shape=(1024, ), name="Elmo_Embedding")(input_layer)
BiLSTM = Bidirectional(layers.LSTM(512, return_sequences=False, recurrent_dropout=0.2, dropout=0.2), name="BiLSTM")
Dense_layer_1 = Dense(128, activation='relu')(BiLSTM)
Dropout_layer_1 = Dropout(0.5)(Dense_layer_1)
# Dense_layer_2 = Dense(64, activation='relu')(Dropout_layer_1)
# Dropout_layer_2 = Dropout(0.5)(Dense_layer_2)
output_layer = Dense(1, activation='sigmoid')(Dropout_layer_1)
model = Model(inputs=[input_layer], outputs=output_layer, name="BiLSTM with ELMo Embeddings")
model.summary()
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
return model

In [17]: elmo_BiDirectional_model = build_model()
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
```

Figure 8: Model Creation for First Experiment

```

In [23]: with tf.Session() as session:
          K.set_session(session)
          session.run(tf.global_variables_initializer())
          session.run(tf.tables_initializer())

          model_elmo = elmo.BidirectionalModel.fit(train_text, train_label, validation_data=(val_text, val_label), epochs=7
          # train_prediction = elmo.BidirectionalModel.predict(train_text)
          elmo.BidirectionalModel.save('model_elmo.h5')

Train on 39391 samples, validate on 13130 samples
Epoch 1/7
39391/39391 [#####] - 10281s 261ms/sample - loss: 0.4481 - accuracy: 0.7861 - val_loss: 0.3888 - val_accuracy: 0.8289
Epoch 2/7
39391/39391 [#####] - 6849s 174ms/sample - loss: 0.3497 - accuracy: 0.8429 - val_loss: 0.3438 - val_accuracy: 0.8430
Epoch 3/7
39391/39391 [#####] - 7099s 180ms/sample - loss: 0.3061 - accuracy: 0.8638 - val_loss: 0.3487 - val_accuracy: 0.8583
Epoch 4/7
39391/39391 [#####] - 7562s 192ms/sample - loss: 0.2630 - accuracy: 0.8849 - val_loss: 0.3399 - val_accuracy: 0.8535
Epoch 5/7
39391/39391 [#####] - 7348s 186ms/sample - loss: 0.2230 - accuracy: 0.9058 - val_loss: 0.3543 - val_accuracy: 0.8496
Epoch 6/7
39391/39391 [#####] - 7367s 187ms/sample - loss: 0.1847 - accuracy: 0.9228 - val_loss: 0.3997 - val_accuracy: 0.8590
Epoch 7/7
39391/39391 [#####] - 7567s 192ms/sample - loss: 0.1512 - accuracy: 0.9369 - val_loss: 0.3962 - val_accuracy: 0.8634

In [24]: from numpy import loadtxt
          from tensorflow.keras.models import load_model

          # load model
          with tf.Session() as session:
              K.set_session(session)
              session.run(tf.global_variables_initializer())
              session.run(tf.tables_initializer())
              elmo.BidirectionalModel.load_model('model_elmo.h5')
              # train_acc = elmo.BidirectionalModel.evaluate(train_text, train_label)
              # test_acc = elmo.BidirectionalModel.evaluate(test_text, test_label)
              print('Train Accuracy: {}, Test Accuracy: {}'.format(round(train_acc,4), round(test_acc,4)))

Train Accuracy: 0.9628000259399434, Test Accuracy: 0.8629999756813849

In [23]: # load model
          with tf.Session() as session:
              K.set_session(session)
              session.run(tf.global_variables_initializer())
              session.run(tf.tables_initializer())
              predict = elmo.BidirectionalModel.predict(np.array(['now days eth is not performing well',]))
              print(predict)

INFO:tensorflowSaver not created because there are no variables in the graph to restore
INFO:tensorflowSaver not created because there are no variables in the graph to restore
[[0.00043027]]

```

Figure 9: Model Training, Results and Prediction

```

In [31]: def plot_graphs(history, metric):
          plt.plot(history.history[metric])
          plt.plot(history.history['val_'+metric], '')
          plt.xlabel("Epochs")
          plt.ylabel(metric)
          plt.legend([metric, 'val_'+metric])

In [32]: plt.figure(figsize=(16, 8))
          plt.subplot(1, 2, 1)
          plot_graphs(model_elmo, 'accuracy')
          plt.ylim(None, 1)
          plt.subplot(1, 2, 2)
          plot_graphs(model_elmo, 'loss')
          plt.ylim(0, None)

Out[32]: (0.0, 0.4629148677333719)

```

Figure 10: Method for Plotting accuracy and loss learning graph

## 2. Implementation of ELMo with Simple RNN model

For both the first experiment, the data preparation, dataset splitting functions and plotting, graphs are the same.

```

In [14]: import numpy as np
          from tensorflow import keras
          from tensorflow.keras.layers import Input, Lambda, Bidirectional, Dense, Dropout
          from tensorflow.keras.models import Model
          from tensorflow.keras import layers
          from tensorflow.python.keras import backend as K

In [15]: def ELMoEmbedding(input_text):
          return elmo(tf.reshape(tf.cast(input_text, tf.string), [-1]), signature="default",
          def build_model():
              input_layer = Input(shape=(1,), dtype="string", name="Input_layer")
              embedding_layer = Lambda(ELMoEmbedding, output_shape=(1024,)), name="Elmo_Embedding"
              simpleRNN = layers.SimpleRNN(512, return_sequences=False, kernel_regularizer=keras.r
              # Dense_layer_1 = Dense(128, activation='relu', name='dense_layer_1')(simpleRNN)
              # Dropout_layer_1 = Dropout(0.5)(Dense_layer_1)
              Dense_layer_2 = Dense(64, activation='relu')(simpleRNN)
              Dropout_layer_2 = Dropout(0.5)(Dense_layer_2)
              output_layer = Dense(1, activation='sigmoid', name='output_layer')(Dropout_layer_2)
              model = Model(inputs=[input_layer], outputs=output_layer, name="Simple RNN with ELM
              model.summary()
              model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
              return model

In [16]: elmo_rnn_model = build_model()

```

Figure 11: Model Creation for Second Implementation

```

In [26]: with tf.Session() as session:
          K.set_session(session)
          session.run(tf.global_variables_initializer())
          session.run(tf.tables_initializer())
          model_elmo = elmo_rnn_model.fit(train_text, train_label, validation_data=(val_text, va
          # train_prediction = elmo_bidirectional_model.predict(train_text)
          elmo_rnn_model.save('simple_rnn.h5')

Train on 39391 samples, validate on 13130 samples
Epoch 1/7
39391/39391 [=====] - 5930s 151ms/sample - loss: 0.8472 - acc
uracy: 0.5876 - val_loss: 0.6920 - val_accuracy: 0.5856
Epoch 2/7
39391/39391 [=====] - 5913s 150ms/sample - loss: 0.6873 - acc
uracy: 0.5889 - val_loss: 0.6787 - val_accuracy: 0.5853
Epoch 3/7
39391/39391 [=====] - 5930s 151ms/sample - loss: 0.6774 - acc
uracy: 0.5901 - val_loss: 0.6786 - val_accuracy: 0.5853
Epoch 4/7
39391/39391 [=====] - 6291s 160ms/sample - loss: 0.6768 - acc
uracy: 0.5901 - val_loss: 0.6792 - val_accuracy: 0.5853
Epoch 5/7
39391/39391 [=====] - 6844s 174ms/sample - loss: 0.6769 - acc
uracy: 0.5901 - val_loss: 0.6785 - val_accuracy: 0.5853
Epoch 6/7
39391/39391 [=====] - 6894s 175ms/sample - loss: 0.6768 - acc
uracy: 0.5901 - val_loss: 0.6785 - val_accuracy: 0.5853
Epoch 7/7
39391/39391 [=====] - 7439s 189ms/sample - loss: 0.6768 - acc
uracy: 0.5901 - val_loss: 0.6786 - val_accuracy: 0.5853

In [27]: from numpy import loadtxt
          from tensorflow.keras.models import load_model

          # load model
          with tf.Session() as session:
              K.set_session(session)
              session.run(tf.global_variables_initializer())
              session.run(tf.tables_initializer())
              elmo_rnn_model.load_model('simple_rnn.h5')
              train_acc = elmo_rnn_model.evaluate(train_text, train_label)
              test_acc = elmo_rnn_model.evaluate(test_text, test_label)

          print('__Train Accuracy__: {}, __Test Accuracy__: {}'.format(round(train_acc,4), round(
INFO:tensorflow:Saver not created because there are no variables in the graph to resto
re
INFO:tensorflow:Saver not created because there are no variables in the graph to resto
re
__Train Accuracy__: 0.590999903678894, __Test Accuracy__: 0.587999995231628

```

Figure 12: Model Training and Results

### 3.3 Model Deployment

We used the PyCharm IDE to build a console-based application that also helped us grasp the structure of the project for quicker deployment. An application called main.py is developed to establish a one-touch automation framework. This script carries out several operations, including scraping the top thirty trending cryptocurrency names, retrieving tweets mentioning cryptocurrency names, loading the best-trained TensorFlow saved model, and emotional categorization for each of the thirty cryptocurrencies. Additionally, the main.py script will show the outcome on the console.

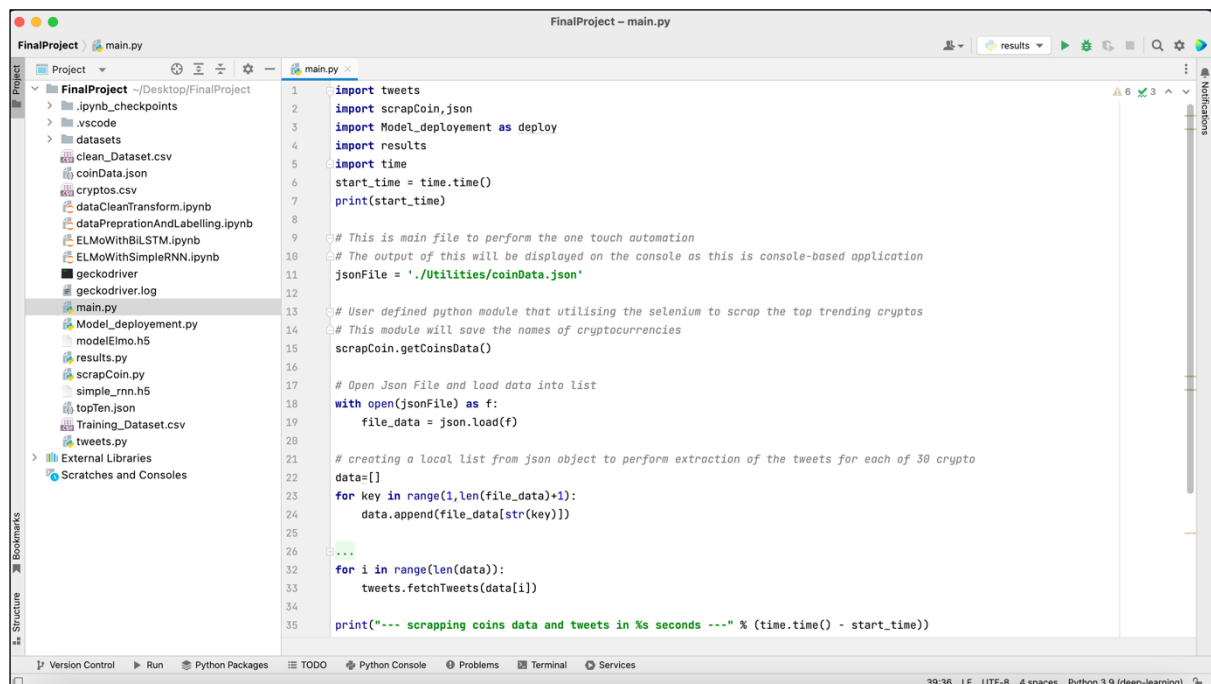


Figure 13: Project Framework Overview in PyCharm



```

1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 import json
4
5 def getCoinsData():
6     driver = webdriver.Firefox(executable_path='./geckodriver')
7     driver.maximize_window()
8     driver.implicitly_wait(20)
9
10    # To access the top trending coins data such that names and symbol
11    driver.get('https://coinmarketcap.com/trending-cryptocurrencies/')
12
13    finalData={}
14    key=1
15    # Iterate over top thirty row for trending cryptos
16    for i in range(1,31):
17        coins = driver.find_elements(By.XPATH, "//tbody/tr["+str(i)+"]/td[3]")[0]
18        coin = coins.text.split('\n')
19        finalData[str(key)]=coin
20        key = key + 1
21    driver.implicitly_wait(20)
22
23    time.sleep(5)
24    driver.quit()
25
26    # Dump the coins name and symbol into JSON file
27    with open('./coinData.json', "w") as outfile:
28        json.dump(finalData, outfile)
29    driver.quit()
30
31 getCoinsData()

```

**Figure 14: Scrapping and Storing the top 30 trending Cryptocurrency Names using Selenium**

```

1 import twint
2
3 # This script is using Twint API to scrape the tweets
4
5 def fetchTweets(tags):
6     HashTags= tags
7     # search wit one by one hashtag from HashTags list
8     for tag in HashTags:
9         c = twint.Config() # get configuration
10        c.Search = tag+' crypto' # search hashtag
11        # c.Limit = 10000 # limit number of Tweets to 10000 for scraping
12        c.Store_csv = True # store tweets in a csv file
13        c.Hide_output=True
14        c.Pandas = True
15        c.Since = "2022-01-23"
16        # c.Output = './cryptos.csv' # This is for training data acquisition
17        c.Output = './'+ tags[0]+' .csv'
18        twint.run.Search(c)
19
20    # import pandas as pd
21    # df = pd.read_csv('./cryptos.csv', low_memory=False)
22    # print(df.__len__())
23
24    # twitterDataset(['Ethereum eth ETH'])

```

**Figure 15: Fetching Tweets for each of the top 30 Cryptocurrencies**

```

35 # Reading each CSV in dataframe to perform cleaning and modelling
36 def readCsv(file):...
41
42 # Cleaning the dataset
43 def cleanDataset(df):...
71
72 # Load the saved model and predict the sentiment for each of the cryptos
73 def calculatePolarity(data):
74     with tf.Session() as session:
75         K.set_session(session)
76         session.run(tf.global_variables_initializer())
77         session.run(tf.tables_initializer())
78         elmo_BiDirectional_model = load_model('modelElmo.h5')
79         predict = elmo_BiDirectional_model.predict(data)
80         # print(predict)
81     return predict
82
83 # below method is to perform reading, cleaning and modelling for all the 30 cryptos
84 # this method is called in main.py script for one touch automation
85 def getSentimentClassification():
86     topTen = []
87     # loop over the list of csv files
88     for f in csv_files:...
116     print(topTen)
117
118     import json
119     with open('./topTen.json', 'w') as fout:
120         json.dump(topTen , fout)

```

Figure 16: Generic Method Sentimental Classification for each of the top 30 cryptocurrencies

<pre> 42 #cleaning the dataset 43 def cleanDataset(df): 44     def dataPreprocessing(tweet): 45         cleanTweet = tweet.lower() 46         pattern = re.compile(r'[!@#0-9%\*\&amp;*\~\.\?&amp;#39;"]') 47         if pattern.search(cleanTweet) != None: 48             return 49         cleanTweet = re.sub(r'@[A-Za-z0-9_]+ https?://.*', '', cleanTweet) 50 51         cleanTweet = cleanTweet.translate(str.maketrans('', '', string.punctuation)) 52         cleanTweet_tokens = word_tokenize(cleanTweet) 53         # print('---', cleanTweet_tokens) 54         # cleanTweet_tokens = [word for word in cleanTweet_tokens if word not in stop_words] 55         if len(cleanTweet_tokens) &lt; 4: 56             return 57         filtered_sentence = [w for w in cleanTweet_tokens if not w.lower() in stop_words] 58 59         # stemObj = PorterStemmer() 60         # stemWords = [stemObj.stem(word) for word in filtered_sentence] 61 62         lemObj = WordNetLemmatizer() 63         lemWords = [lemObj.lemmatize(word, pos='a') for word in filtered_sentence] 64 65         return ' '.join(lemWords) 66 67 </pre>	<pre> 41 #cleaning the dataset 42 def cleanDataset(df):... 71 72 # Load the saved model and predict the sentiment for each of the cryptos 73 def calculatePolarity(data): 74     with tf.Session() as session: 75         K.set_session(session) 76         session.run(tf.global_variables_initializer()) 77         session.run(tf.tables_initializer()) 78         elmo_BiDirectional_model = load_model('modelElmo.h5') 79         predict = elmo_BiDirectional_model.predict(data) 80         # print(predict) 81     return predict 82 83 # below method is to perform reading, cleaning and modelling for all the 30 cryptos 84 # this method is called in main.py script for one touch automation 85 def getSentimentClassification():... </pre>
--	--

Figure 17: Cleaning and Loading the Saved Model

```

92     df = cleanDataset(df)
93
94     # dropping null values
95     # print(df.head())
96     df = df.dropna()
97     predict = calculatePolarity(df['cleanedTweets'])
98
99     # Classifying the tweets with 0 and 1
100    # print(predict)
101    def sentiment(sentimentScore):
102        if sentimentScore > 0.5:
103            return 1
104        else:
105            return 0
106    df['sentiments'] = predict
107    df['sentiment'] = df['sentiments'].apply(sentiment)
108    print(df['sentiment'].value_counts())
109    pos = df['sentiment'].value_counts()[1]
110    total = pos + df['sentiment'].value_counts()[0]
111    print(pos,total)
112    data['crypto']=os.path.basename(f).split('.')[0]
113    data['pos']= pos/total*100
114    topTen.append(data)
115    # print(topTen)
116    print(topTen)
117
118    import json
119    with open('./topTen.json', 'w') as fout:
120        json.dump(topTen , fout)

```

**Figure 18: Sentiment Score Mapping with 0 and 1 after Sentimental Classification**

```

1
2     import json
3     def getResults():
4         with open("./topTen.json", "r") as read_file:
5             data = json.load(read_file)
6             print(data)
7             t1=sorted(data, key=lambda i: i['pos'],reverse=True)
8             print('\n\nRecommending Top Ten Cryptocurrencies of The Day')
9             from prettytable import PrettyTable
10            t = PrettyTable(['Cryptos', 'Positive Public Opinion(in%)'])
11            for i in range(10):
12                t.add_row([t1[i]['crypto'],t1[i]['pos']])
13            print(t)
14

```

**Figure 19: Method to Show the list of Top 10 Most Favourable Cryptocurrencies based on the Sentimental Classification**