Earthquake detection using Twitter Tweets

Anurag Gautam MT22015
Muskaan Gupta MT22113
Saloni Garg MT22063
Saloni Srivastava MT22301
Surbhi Goyal MT22076
Sumit Bhagat MT22302

1 PROBLEM STATEMENT

This project addresses the need for a faster and more efficient earthquake detection and alerting system that uses social media channels, notably Twitter. Traditional earthquake monitoring systems are frequently constrained by their reliance on seismic sensors, which can be costly and time-consuming to install. Furthermore, there is frequently a lag in sending information to appropriate authorities and non-governmental organizations (NGOs), which can stymie disaster response operations. This project intends to create a real-time earthquake detection and notification system that can detect seismic events quickly and correctly and warn NGOs and other monitoring committees utilizing Twitter data analysis techniques and machine learning algorithms. The goal is to provide a cost-effective and timely solution that can help save lives and mitigate the impact of earthquakes.

2 MOTIVATION

Firstly, Social media occupies a large part of our day-to-day lives. People keep updating their accounts with their recent activities, places they have been to, or events they attend. Social media can also be very helpful in posting about natural calamities. The use of social media during natural disasters has allowed people to broadcast their safety.
Secondly, Twitter is a useful tool for real-time earthquake detection since it allows people to submit real-time information. This can facilitate timely action and precautions to be taken.
Thirdly, Twitter is a useful tool for communicating important information regarding seismic activity since it is generally available to a large audience. When phone networks crashed during the earthquake, many people turned to Twitter, a social-networking website in the country.
Moreover, analyzing tweets can yield accurate information about an earthquake's location, magnitude, and other relevant details.
Finally, analyzing social media data, including tweets, can offer insights into people's reactions to natural disasters, which can help develop better disaster response strategies.

3 LITERATURE REVIEW

[1] This paper discusses how Twitter can be used as a source of information and proposes a method for detecting real-time events using Convolution Neural Network (CNN). The CNN model is trained on past earthquake tweets labeled by crowdsourcing and used to predict whether a tweet containing the earthquake keyword is informative. The informative tweets are then used as input streaming data for the real-time event detection algorithm. This system can detect earthquakes with a tolerance level faster than official government disaster websites.

[2]This paper investigates using Twitter as a real-time interaction platform to detect events such as earthquakes. The authors propose an algorithm that uses a classifier to monitor tweets and detect target events based on features such as keywords and context. They also devise a probabilistic spatiotemporal model for the event and use Kalman filtering and particle filtering for location estimation. The proposed system is applied to construct an earthquake reporting system in Japan that detects earthquakes with high probability and sends emails to registered users much faster than official announcements by the Japan Meteorological Agency.

[3]The proposed system in this paper uses deep learning techniques such as RNN/LSTM to validate Twitter tweets and provide real-time detection of earthquakes. Machine learning models are trained from past labeled tweets related to earthquakes and used as classifiers to predict the validity of tweets. The system listens to particular keywords like "earthquake" using Twitter's API and converts tweets to word embeddings using BERT before inputting them into the model. The proposed system can detect earthquakes with a higher tolerance level than existing websites and provide earlier warnings to the public.

[4]This paper outlines two main approaches to tracking and analyzing hashtag-based Twitter activities during a crisis: using an open-source tool like yourTwapperkeeper and additional tools to process and visualize Twitter activities. The other requires custom-designed tracking and analysis tools like Gawk.

4 NOVELTY

When an earthquake is struck, it takes approximately two to three days to get it covered in the news. This delays the process of reaching out and providing aid to the people and places where the calamity has been hit. Through this project, we provide a platform for the nearby NGOs to inform them where the earthquake has been detected in real-time. This accelerates the process of helping the ones in need.

5 METHODOLOGY

Our problem solution first requires extracting the tweets from the historic data streamed as live data using Kafka stream. The tweets will be extracted based on the word and hashtags like earthquake, stuck, help,#earthquakes, and many more. These extracted tweets will become our dataset. The dataset will have Tweet, the user's location who has tweeted, as a significant feature that can be used further.

5.1 BERT

In our project, we have used the BERT algorithm. BERT is used to classify the dataset as relevant(talking about an earthquake Hit) or irrelevant.

5.2 Kafka Streaming

Kafka Streaming is used to stream tweets in real time. The process involves importing a historic tweet dataset into Kafka and setting up a Kafka Connect Source Connector that reads the data from the Kafka topic and writes it to a new topic in real time.

5.3 Bing API

Used to find the coordinates of the locations extracted.

5.4 Flask

Used to build a web application.

5.5 Folium

Creating a Map using the coordinates obtained from locations.

5.6 Socket

Used for two-way communication between Front-end and Back-end.

5.7 React JS

To provide the users and NGOs with an easy platform to get notified where the earthquake is occurring.

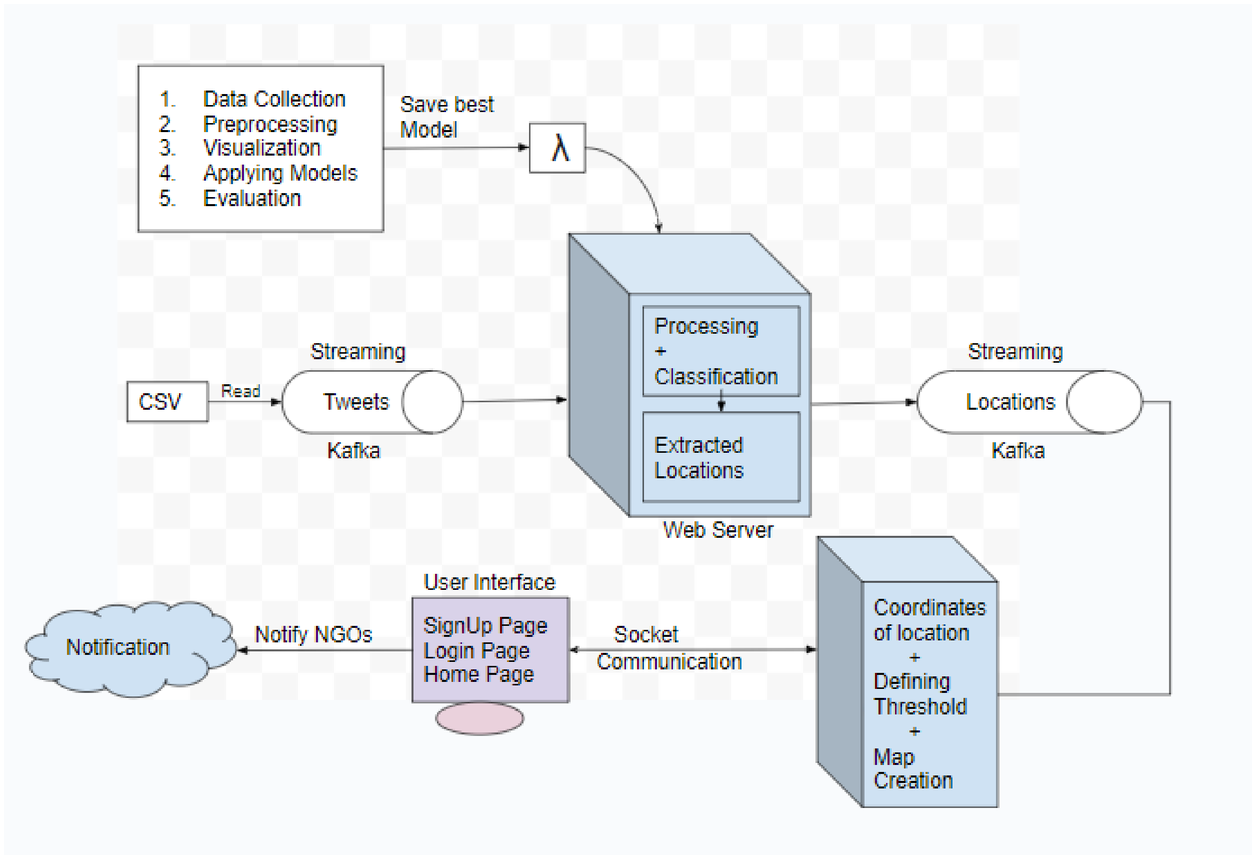


Figure 1: Architecture of our application

6 DATABASE

6.1 Database

We used **Firebase**, a cloud-based platform provided by Google, for user authentication, allowing us to add secure authentication to our Shaky app easily. In addition to authentication, it offers us a real-time database and cloud storage to store user data.

6.2 Datasets

6.2.1 Classification Model Training Data. We used a dataset from CrisisLex, which has 33095 rows and 3 columns. The column includes:- Id, Relevance, and Text

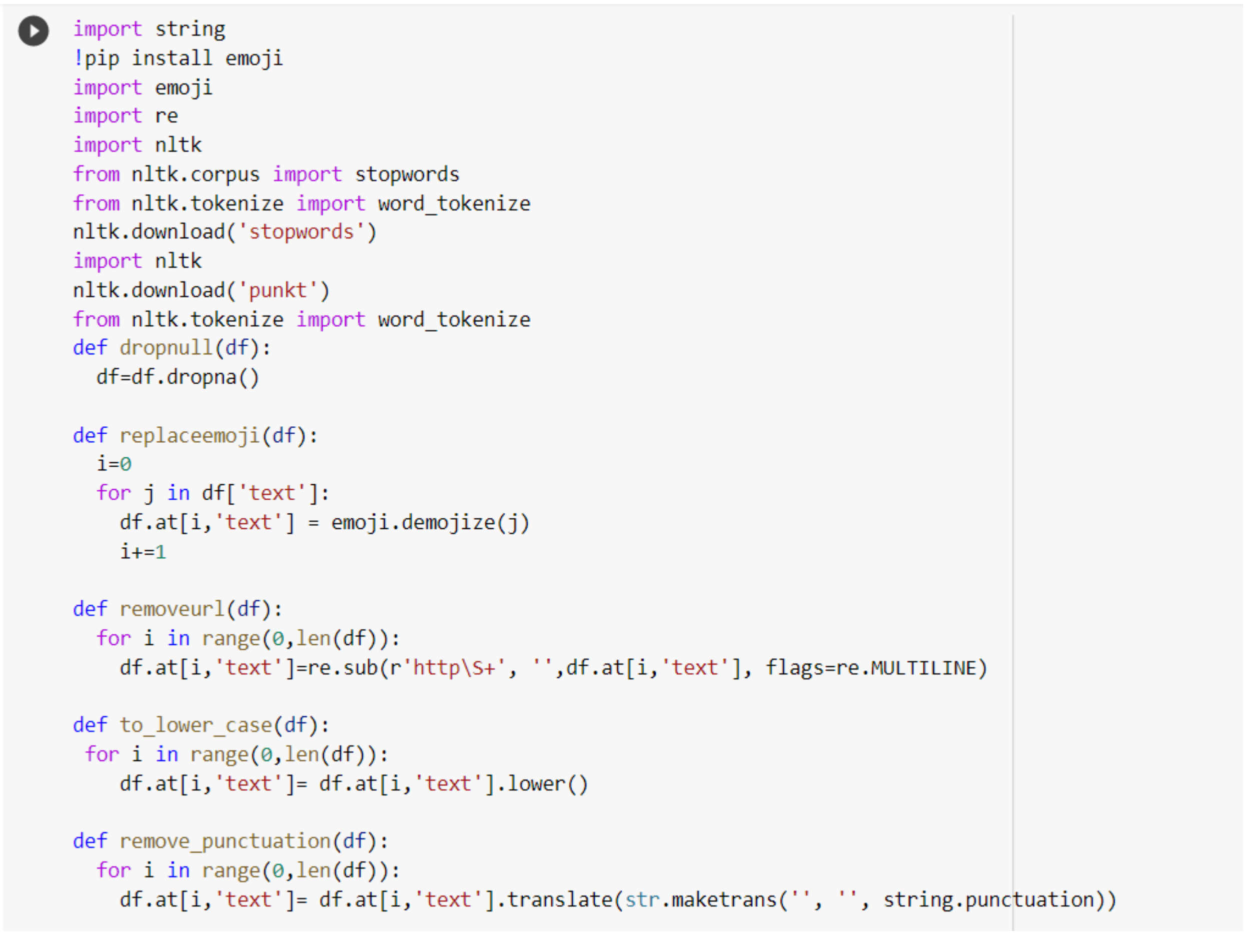
6.2.2 Classification Model Testing Data. We have used a dataset from CrisisLex, which has 4463 rows and 3 columns. The column includes:- Id, Text, and Relevance

6.2.3 Dataset Used in Kafka Streaming. Here we have used a dataset that has 98000 rows and four columns. The column includes- named id, event,src, and text
This data is historical Twitter data which has been stimulated by Kafka streaming to create a visual representation of tweets live streaming.

7 CODE

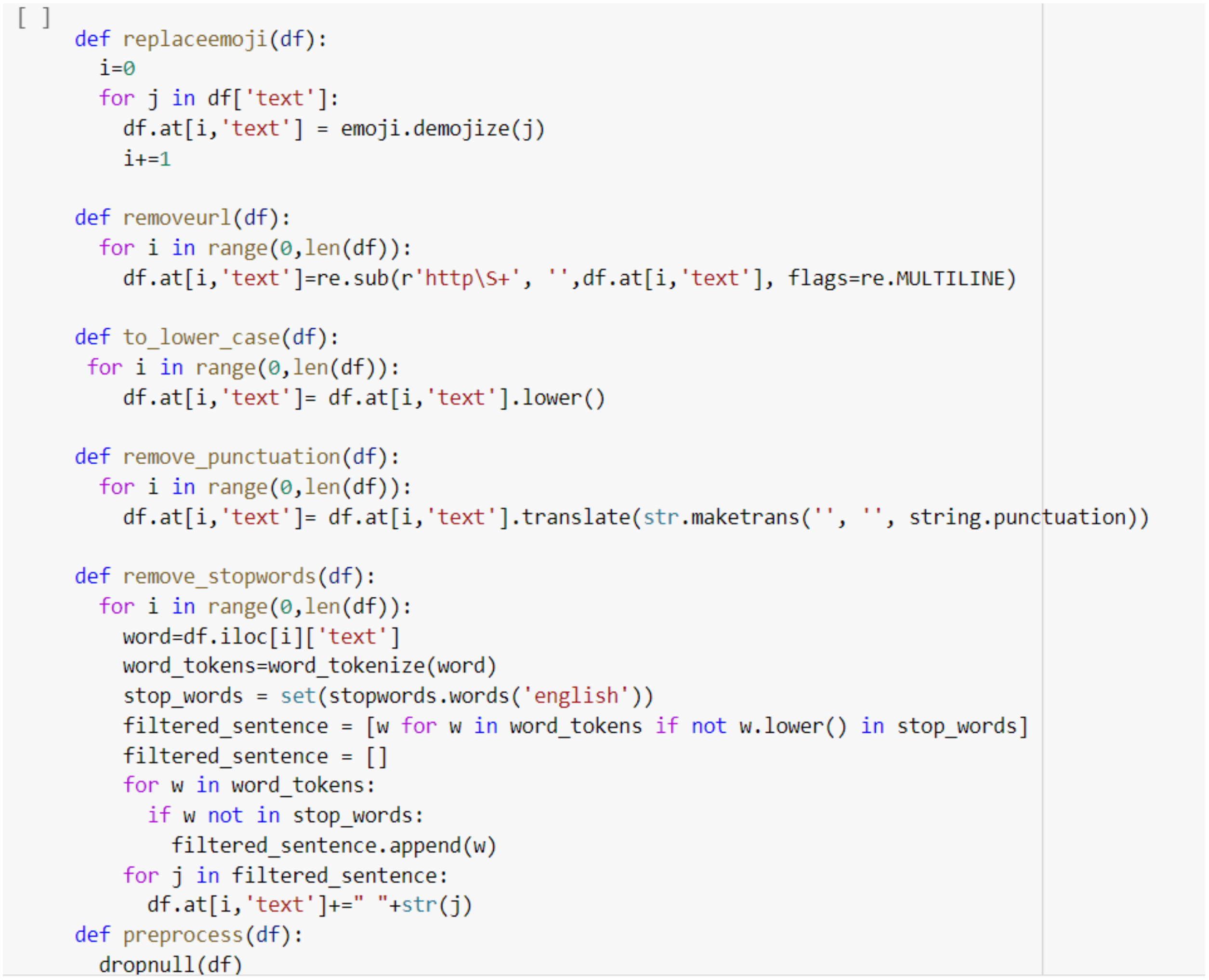
7.1 Model for Classification of tweets

7.1.1 Preprocessing



```
import string
!pip install emoji
import emoji
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
def dropnull(df):
  df=df.dropna()

def replaceemoji(df):
  i=0
  for j in df['text']:
    df.at[i,'text'] = emoji.demojize(j)
    i+=1

def removeurl(df):
  for i in range(0,len(df)):
    df.at[i,'text']=re.sub(r'http\S+', '',df.at[i,'text'], flags=re.MULTILINE)

def to_lower_case(df):
 for i in range(0,len(df)):
    df.at[i,'text']= df.at[i,'text'].lower()

def remove_punctuation(df):
  for i in range(0,len(df)):
    df.at[i,'text']= df.at[i,'text'].translate(str.maketrans('', '', string.punctuation))
```

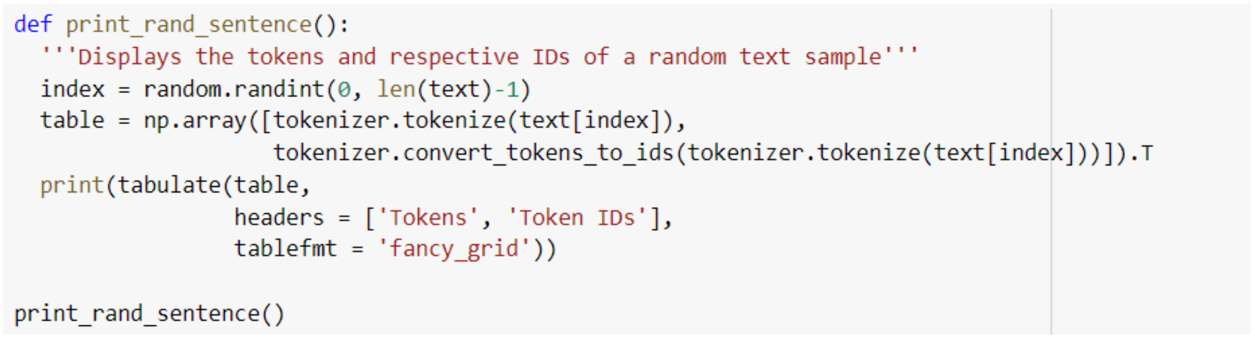
Figure 2: Code snippet



```
[ ]
def replaceemoji(df):
  i=0
  for j in df['text']:
    df.at[i,'text'] = emoji.demojize(j)
    i+=1

def removeurl(df):
  for i in range(0,len(df)):
    df.at[i,'text']=re.sub(r'http\S+', '',df.at[i,'text'], flags=re.MULTILINE)

def to_lower_case(df):
 for i in range(0,len(df)):
    df.at[i,'text']= df.at[i,'text'].lower()

def remove_punctuation(df):
  for i in range(0,len(df)):
    df.at[i,'text']= df.at[i,'text'].translate(str.maketrans('', '', string.punctuation))

def remove_stopwords(df):
  for i in range(0,len(df)):
    word=df.iloc[i]['text']
    word_tokens=word_tokenize(word)
    stop_words = set(stopwords.words('english'))
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
    filtered_sentence = []
    for w in word_tokens:
      if w not in stop_words:
        filtered_sentence.append(w)
    for j in filtered_sentence:
      df.at[i,'text']+=" "+str(j)
def preprocess(df):
  dropnull(df)
```

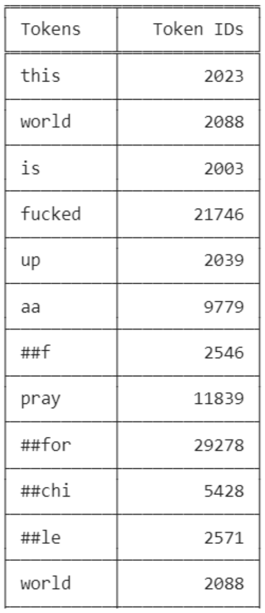
Figure 3: Code snippet

7.1.2 Tokenization



```
def print_rand_sentence():
  '''Displays the tokens and respective IDs of a random text sample'''
  index = random.randint(0, len(text)-1)
  table = np.array([tokenizer.tokenize(text[index]),
                    tokenizer.convert_tokens_to_ids(tokenizer.tokenize(text[index]))]).T
  print(tabulate(table,
                 headers = ['Tokens', 'Token IDs'],
                 tablefmt = 'fancy_grid'))

print_rand_sentence()
```

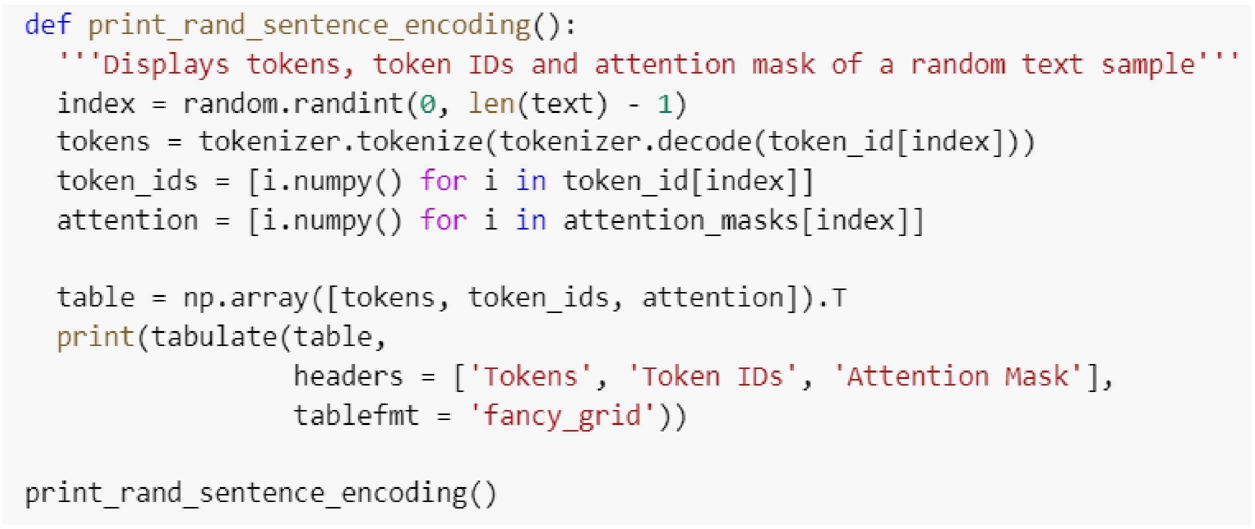
Figure 4: Code snippet



Tokens	Token IDs
this	2023
world	2088
is	2003
fucked	21746
up	2039
aa	9779
##f	2546
pray	11839
##for	29278
##chi	5428
##le	2571
world	2088

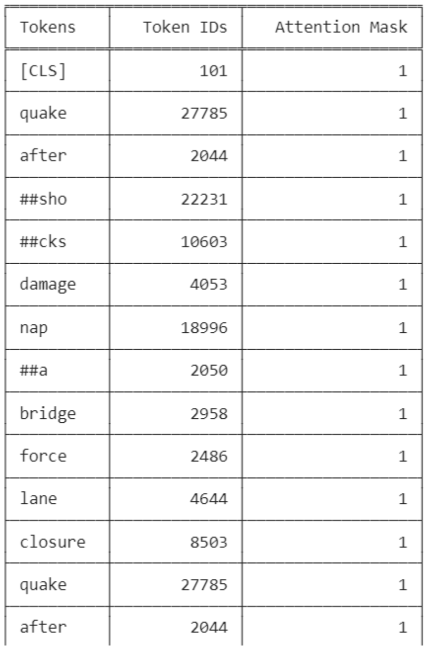
Table 1: Representation of tokens with tokenId

7.1.3 Adding attention mask



```python
def print_rand_sentence_encoding():
  '''Displays tokens, token IDs and attention mask of a random text sample'''
  index = random.randint(0, len(text) - 1)
  tokens = tokenizer.tokenize(tokenizer.decode(token_id[index]))
  token_ids = [i.numpy() for i in token_id[index]]
  attention = [i.numpy() for i in attention_masks[index]]

  table = np.array([tokens, token_ids, attention]).T
  print(tabulate(table,
                 headers = ['Tokens', 'Token IDs', 'Attention Mask'],
                 tablefmt = 'fancy_grid'))

print_rand_sentence_encoding()
```

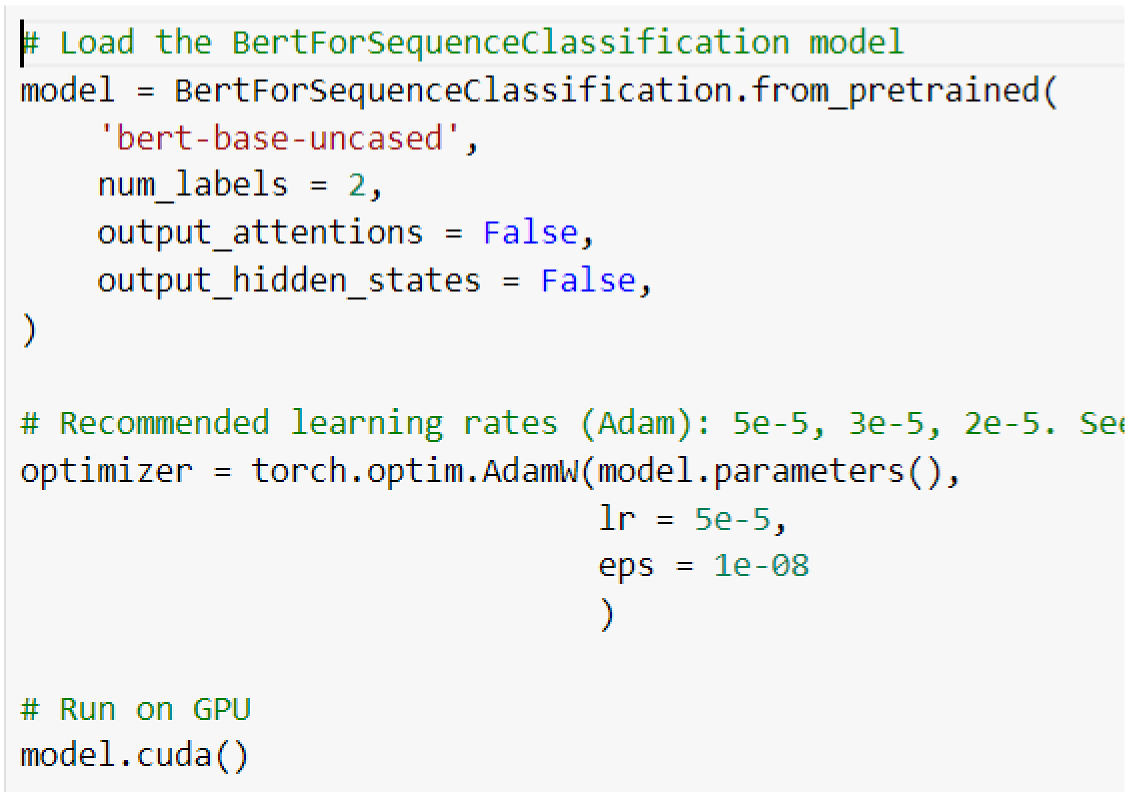
Figure 5: Code snippet



Tokens	Token IDs	Attention Mask
[CLS]	101	1
quake	27785	1
after	2044	1
##sho	22231	1
##cks	10603	1
damage	4053	1
nap	18996	1
##a	2050	1
bridge	2958	1
force	2486	1
lane	4644	1
closure	8503	1
quake	27785	1
after	2044	1

Table 2: Tokens with Attention Mask

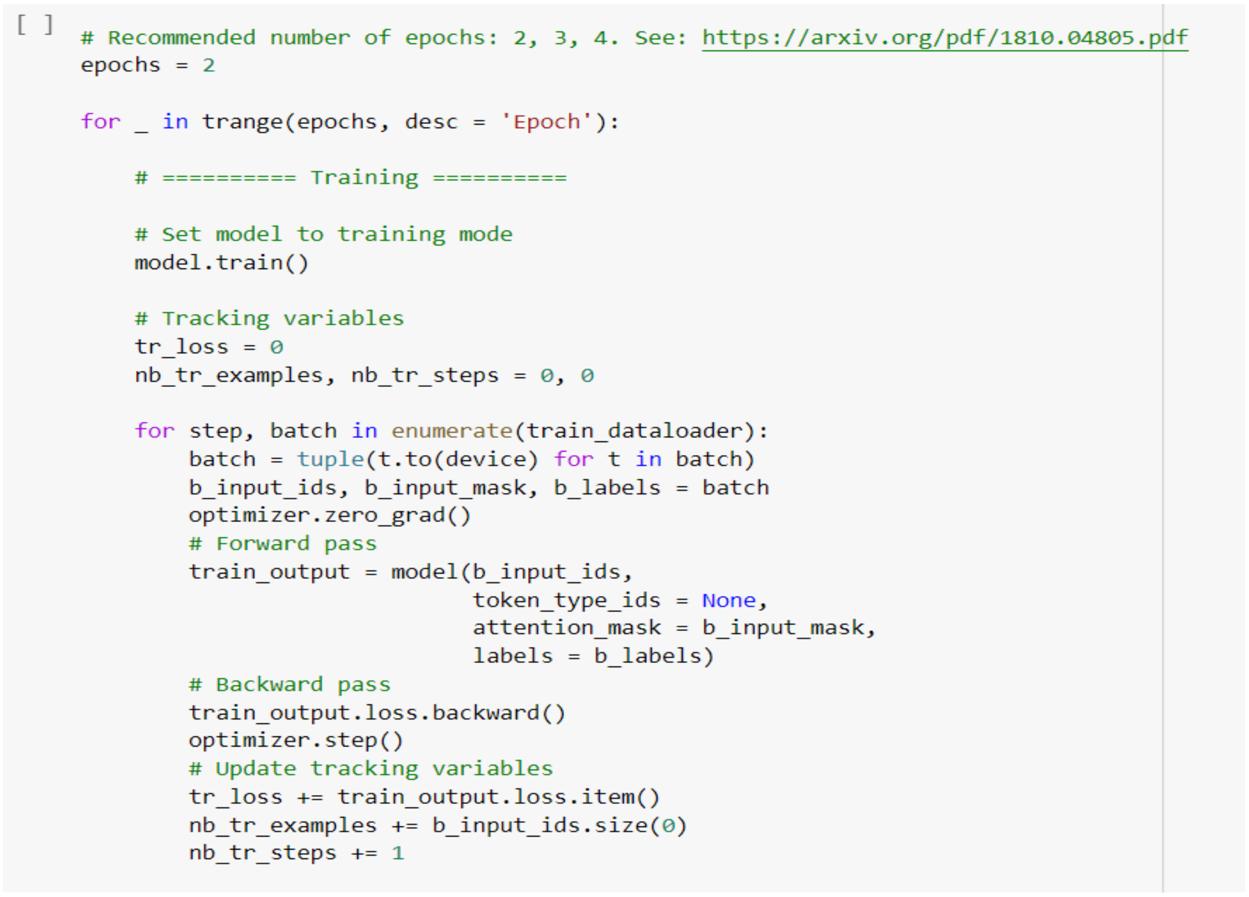
7.1.4 Applying BERT Model



```
# Load the BertForSequenceClassification model
model = BertForSequenceClassification.from_pretrained(
    'bert-base-uncased',
    num_labels = 2,
    output_attentions = False,
    output_hidden_states = False,
)

# Recommended learning rates (Adam): 5e-5, 3e-5, 2e-5. Se
optimizer = torch.optim.AdamW(model.parameters(),
                              lr = 5e-5,
                              eps = 1e-08
                              )

# Run on GPU
model.cuda()
```

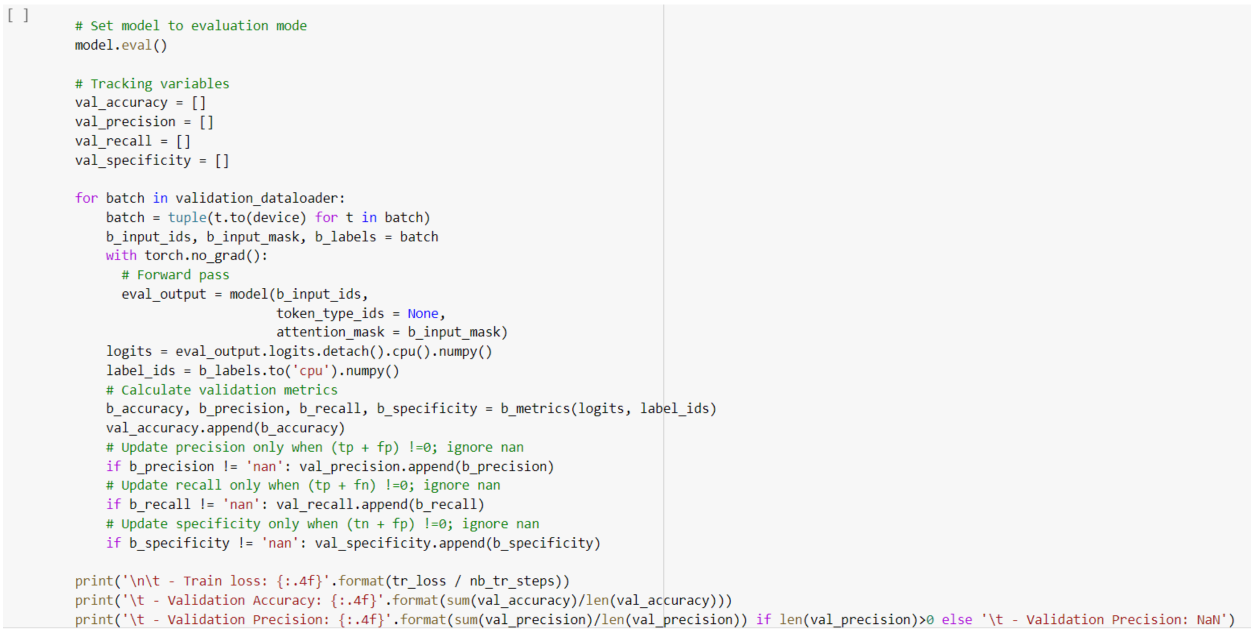
Figure 6: Code snippet

7.1.5 *Training Model Using Train Dataset*



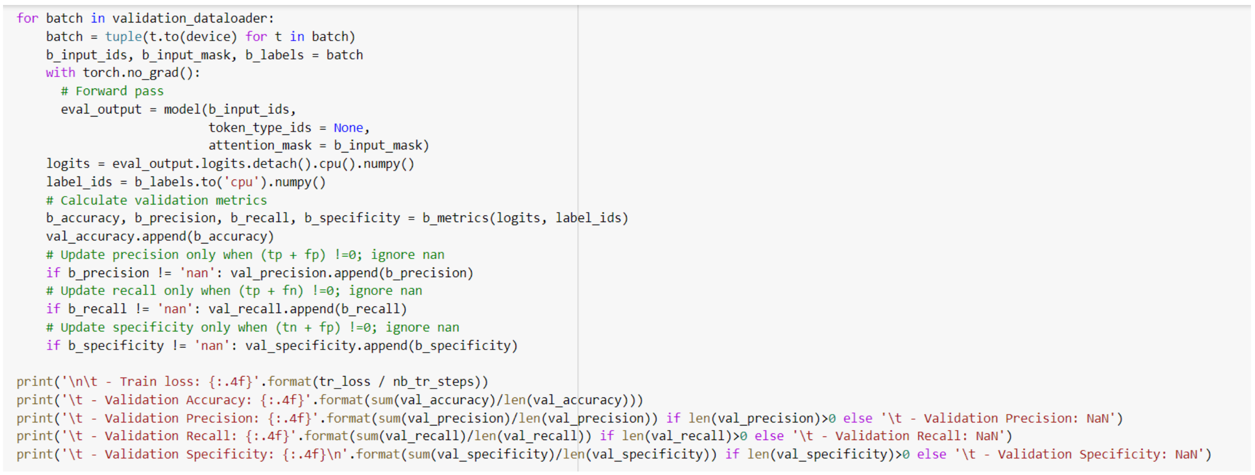
```
[ ]
    # Recommended number of epochs: 2, 3, 4. See: https://arxiv.org/pdf/1810.04805.pdf
    epochs = 2

    for _ in trange(epochs, desc = 'Epoch'):

        # ========== Training ==========

        # Set model to training mode
        model.train()

        # Tracking variables
        tr_loss = 0
        nb_tr_examples, nb_tr_steps = 0, 0

        for step, batch in enumerate(train_dataloader):
            batch = tuple(t.to(device) for t in batch)
            b_input_ids, b_input_mask, b_labels = batch
            optimizer.zero_grad()
            # Forward pass
            train_output = model(b_input_ids,
                                 token_type_ids = None,
                                 attention_mask = b_input_mask,
                                 labels = b_labels)
            # Backward pass
            train_output.loss.backward()
            optimizer.step()
            # Update tracking variables
            tr_loss += train_output.loss.item()
            nb_tr_examples += b_input_ids.size(0)
            nb_tr_steps += 1
```

Figure 7: Code snippet



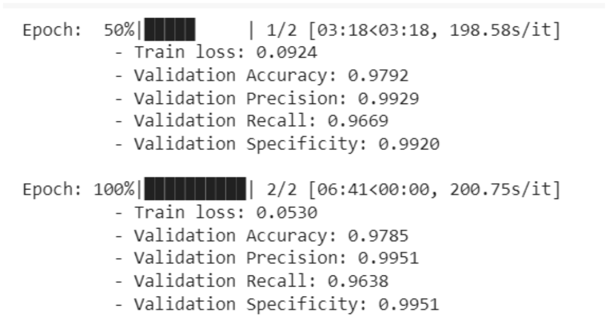
```
[ ]
    # Set model to evaluation mode
    model.eval()

    # Tracking variables
    val_accuracy = []
    val_precision = []
    val_recall = []
    val_specificity = []

    for batch in validation_dataloader:
        batch = tuple(t.to(device) for t in batch)
        b_input_ids, b_input_mask, b_labels = batch
        with torch.no_grad():
          # Forward pass
          eval_output = model(b_input_ids,
                              token_type_ids = None,
                              attention_mask = b_input_mask)
        logits = eval_output.logits.detach().cpu().numpy()
        label_ids = b_labels.to('cpu').numpy()
        # Calculate validation metrics
        b_accuracy, b_precision, b_recall, b_specificity = b_metrics(logits, label_ids)
        val_accuracy.append(b_accuracy)
        # Update precision only when (tp + fp) !=0; ignore nan
        if b_precision != 'nan': val_precision.append(b_precision)
        # Update recall only when (tp + fn) !=0; ignore nan
        if b_recall != 'nan': val_recall.append(b_recall)
        # Update specificity only when (tn + fp) !=0; ignore nan
        if b_specificity != 'nan': val_specificity.append(b_specificity)

    print('\n\t - Train loss: {:.4f}'.format(tr_loss / nb_tr_steps))
    print('\t - Validation Accuracy: {:.4f}'.format(sum(val_accuracy)/len(val_accuracy)))
    print('\t - Validation Precision: {:.4f}'.format(sum(val_precision)/len(val_precision)) if len(val_precision)>0 else '\t - Validation Precision: NaN')
```

Figure 8: Code snippet



```
for batch in validation_dataloader:
    batch = tuple(t.to(device) for t in batch)
    b_input_ids, b_input_mask, b_labels = batch
    with torch.no_grad():
      # Forward pass
      eval_output = model(b_input_ids,
                          token_type_ids = None,
                          attention_mask = b_input_mask)
    logits = eval_output.logits.detach().cpu().numpy()
    label_ids = b_labels.to('cpu').numpy()
    # Calculate validation metrics
    b_accuracy, b_precision, b_recall, b_specificity = b_metrics(logits, label_ids)
    val_accuracy.append(b_accuracy)
    # Update precision only when (tp + fp) !=0; ignore nan
    if b_precision != 'nan': val_precision.append(b_precision)
    # Update recall only when (tp + fn) !=0; ignore nan
    if b_recall != 'nan': val_recall.append(b_recall)
    # Update specificity only when (tn + fp) !=0; ignore nan
    if b_specificity != 'nan': val_specificity.append(b_specificity)

print('\n\t - Train loss: {:.4f}'.format(tr_loss / nb_tr_steps))
print('\t - Validation Accuracy: {:.4f}'.format(sum(val_accuracy)/len(val_accuracy)))
print('\t - Validation Precision: {:.4f}'.format(sum(val_precision)/len(val_precision)) if len(val_precision)>0 else '\t - Validation Precision: NaN')
print('\t - Validation Recall: {:.4f}'.format(sum(val_recall)/len(val_recall)) if len(val_recall)>0 else '\t - Validation Recall: NaN')
print('\t - Validation Specificity: {:.4f}\n'.format(sum(val_specificity)/len(val_specificity)) if len(val_specificity)>0 else '\t - Validation Specificity: NaN')
```

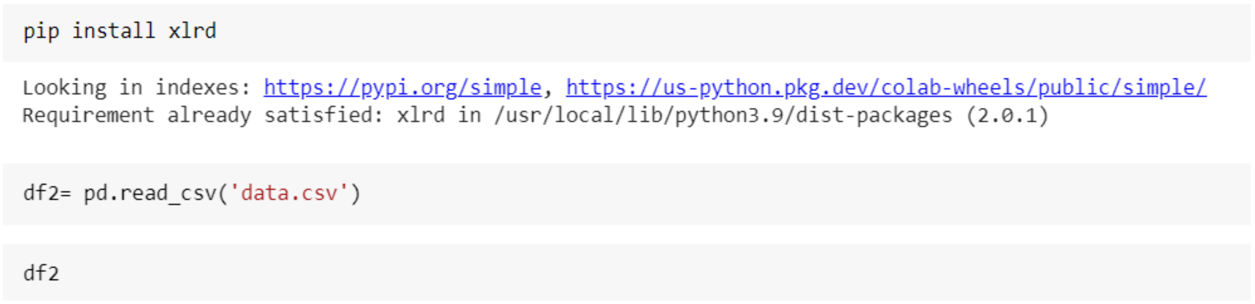
Figure 9: Code snippet



```
Epoch:  50%|█████     | 1/2 [03:18<03:18, 198.58s/it]
         - Train loss: 0.0924
         - Validation Accuracy: 0.9792
         - Validation Precision: 0.9929
         - Validation Recall: 0.9669
         - Validation Specificity: 0.9920

Epoch: 100%|██████████| 2/2 [06:41<00:00, 200.75s/it]
         - Train loss: 0.0530
         - Validation Accuracy: 0.9785
         - Validation Precision: 0.9951
         - Validation Recall: 0.9638
         - Validation Specificity: 0.9951
```

Figure 10: Result snippet

7.1.6 Classification on Testing Data



```
pip install xlrd
```

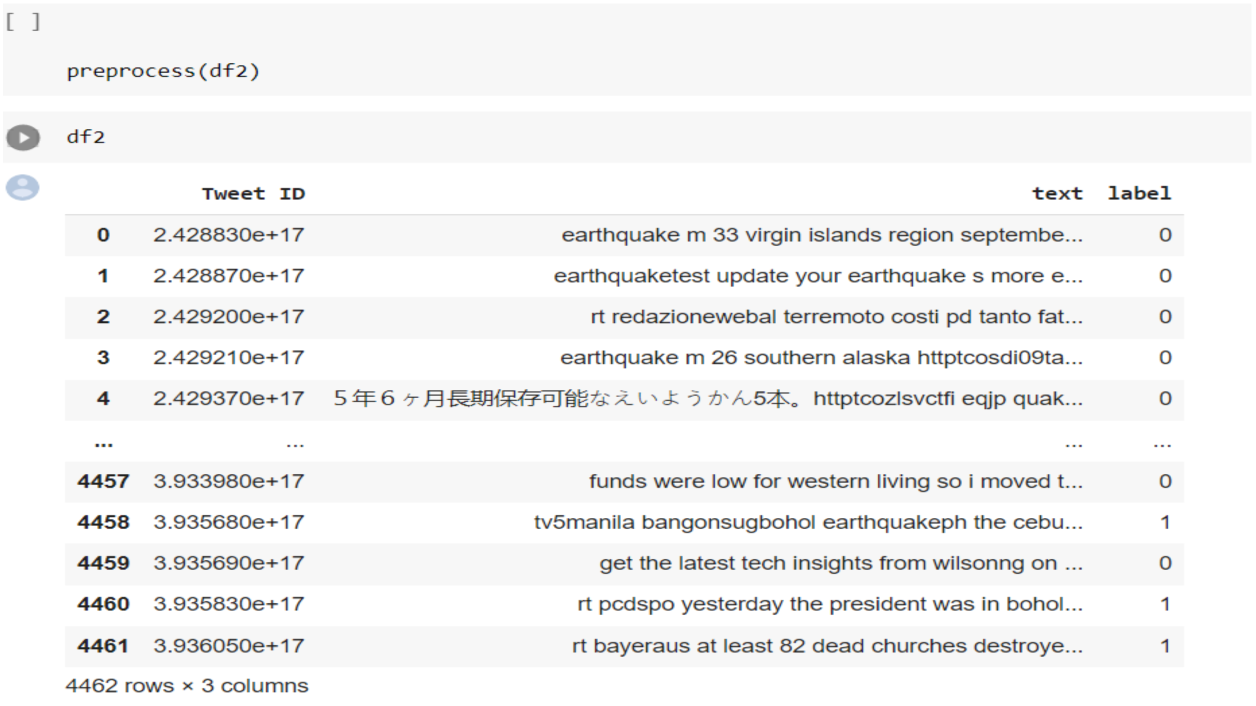
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: xlrd in /usr/local/lib/python3.9/dist-packages (2.0.1)
```

```
df2= pd.read_csv('data.csv')
```

```
df2
```

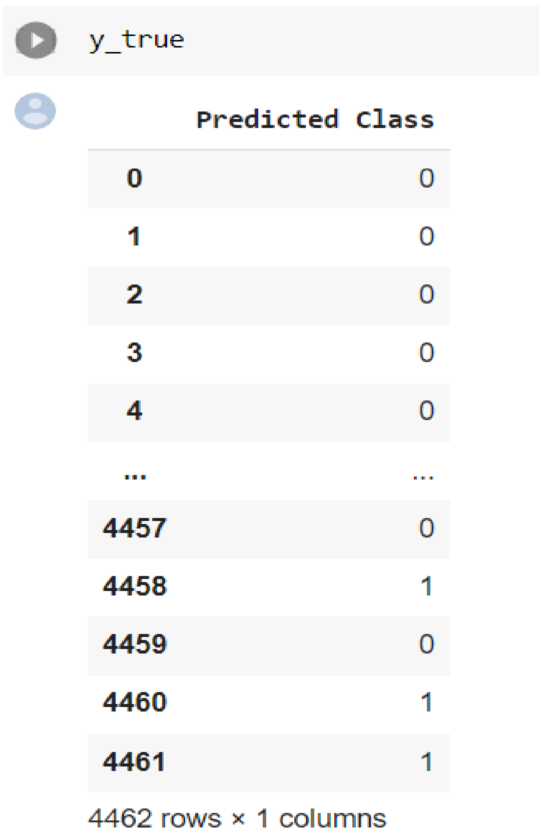
Figure 11: Code snippet

7.1.7 Preprocessing on new dataset



```
[ ]
    preprocess(df2)

    df2

          Tweet ID                                                          text  label
0     2.428830e+17                 earthquake m 33 virgin islands region septembe...      0
1     2.428870e+17                 earthquaketest update your earthquake s more e...      0
2     2.429200e+17                    rt redazionewebal terremoto costi pd tanto fat...      0
3     2.429210e+17                earthquake m 26 southern alaska httptcosdi09ta...      0
4     2.429370e+17  ５年６ヶ月長期保存可能なえいようかん5本。httptcozlsvctfi eqjp quak...      0
...            ...                                                           ...    ...
4457  3.933980e+17                 funds were low for western living so i moved t...      0
4458  3.935680e+17            tv5manila bangonsugbohol earthquakeph the cebu...      1
4459  3.935690e+17                   get the latest tech insights from wilsonng on ...      0
4460  3.935830e+17                  rt pcdspo yesterday the president was in bohol...      1
4461  3.936050e+17                 rt bayeraus at least 82 dead churches destroye...      1
4462 rows × 3 columns
```

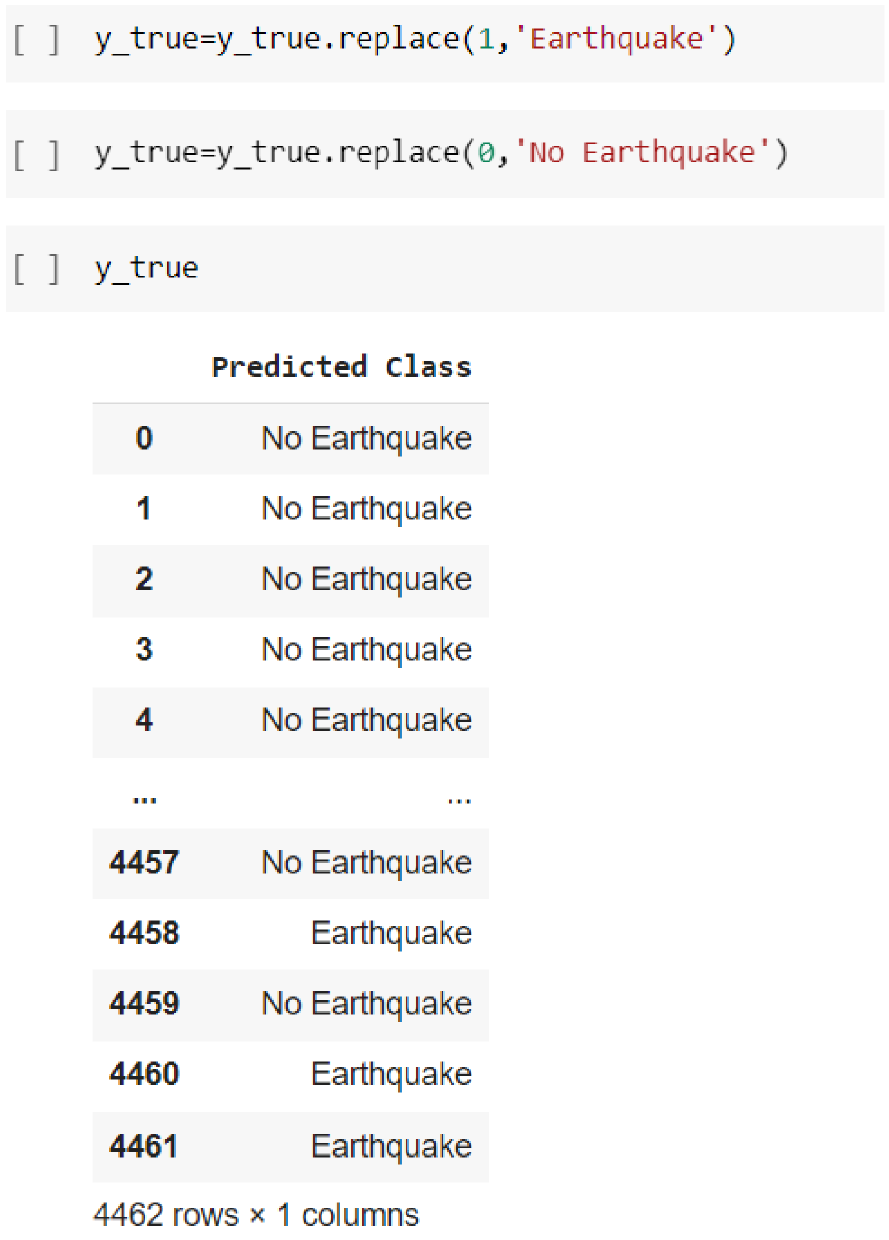
Figure 12: Code snippet



```
y_true

      Predicted Class
0                   0
1                   0
2                   0
3                   0
4                   0
...               ...
4457                0
4458                1
4459                0
4460                1
4461                1
4462 rows × 1 columns
```

Figure 13: Code snippet

7.1.8 Replacing 0's with no earthquake and 1's with earthquake



```
[ ] y_true=y_true.replace(1,'Earthquake')
```

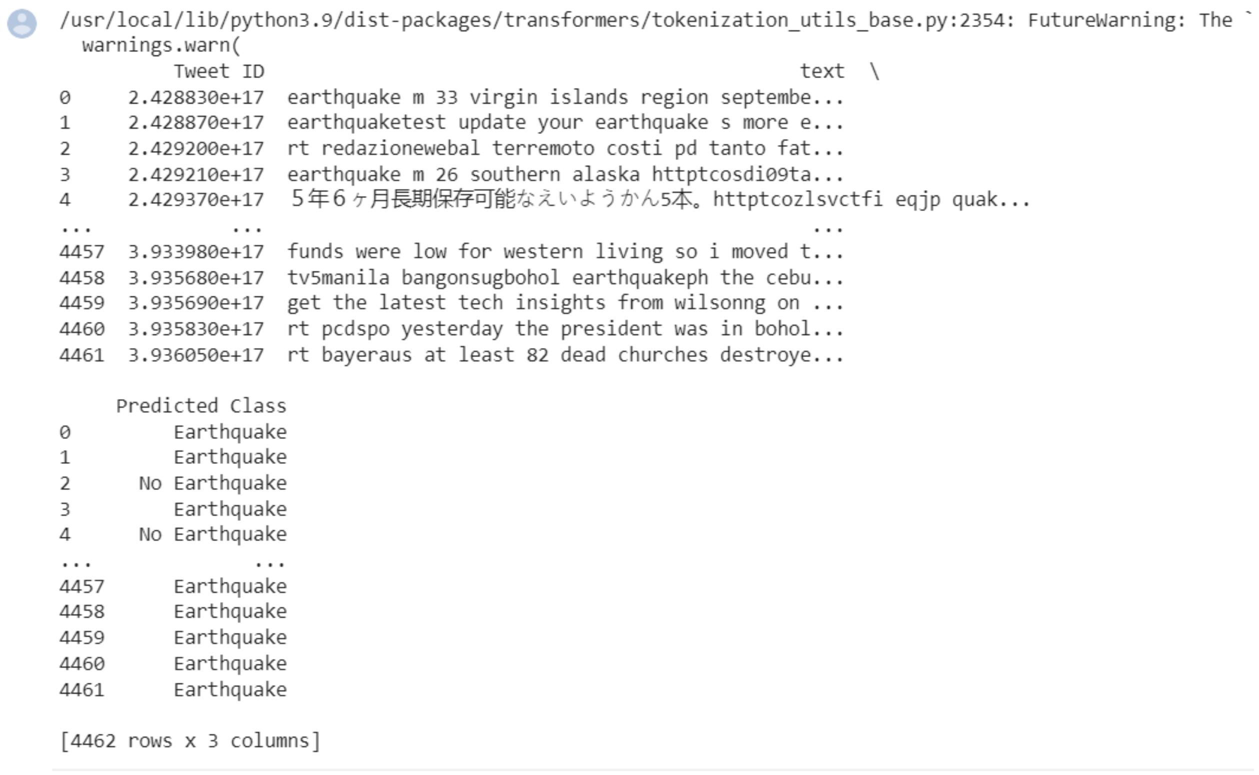
```
[ ] y_true=y_true.replace(0,'No Earthquake')
```

```
[ ] y_true
```

```
      Predicted Class
   0    No Earthquake
   1    No Earthquake
   2    No Earthquake
   3    No Earthquake
   4    No Earthquake
 ...              ...
4457    No Earthquake
4458       Earthquake
4459    No Earthquake
4460       Earthquake
4461       Earthquake
4462 rows × 1 columns
```

Figure 14: Code snippet

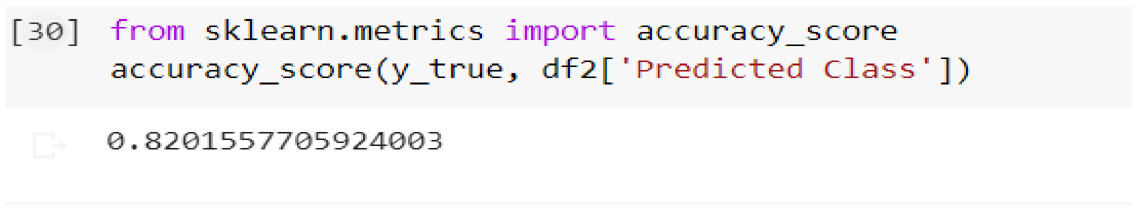
7.1.9 Tokenization and assigning attention mask using BERT



```
/usr/local/lib/python3.9/dist-packages/transformers/tokenization_utils_base.py:2354: FutureWarning: The `
  warnings.warn(
          Tweet ID                                               text  \
0     2.428830e+17  earthquake m 33 virgin islands region septembe...
1     2.428870e+17  earthquaketest update your earthquake s more e...
2     2.429200e+17  rt redazionewebal terremoto costi pd tanto fat...
3     2.429210e+17  earthquake m 26 southern alaska httptcosdi09ta...
4     2.429370e+17  ５年６ヶ月長期保存可能なえいようかん5本。httptcozlsvctfi eqjp quak...
...            ...                                                ...
4457  3.933980e+17  funds were low for western living so i moved t...
4458  3.935680e+17  tv5manila bangonsugbohol earthquakeph the cebu...
4459  3.935690e+17  get the latest tech insights from wilsonng on ...
4460  3.935830e+17  rt pcdspo yesterday the president was in bohol...
4461  3.936050e+17  rt bayeraus at least 82 dead churches destroye...

     Predicted Class
0         Earthquake
1         Earthquake
2      No Earthquake
3         Earthquake
4      No Earthquake
...              ...
4457      Earthquake
4458      Earthquake
4459      Earthquake
4460      Earthquake
4461      Earthquake

[4462 rows x 3 columns]
```

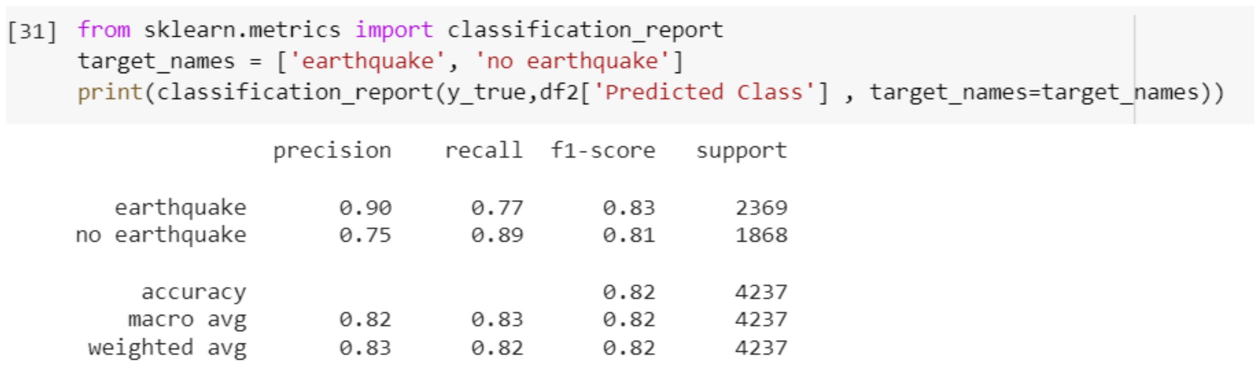
Figure 15: Code snippet

7.2.0 Final Accuracy



```
[30] from sklearn.metrics import accuracy_score
     accuracy_score(y_true, df2['Predicted Class'])

     0.8201557705924003
```

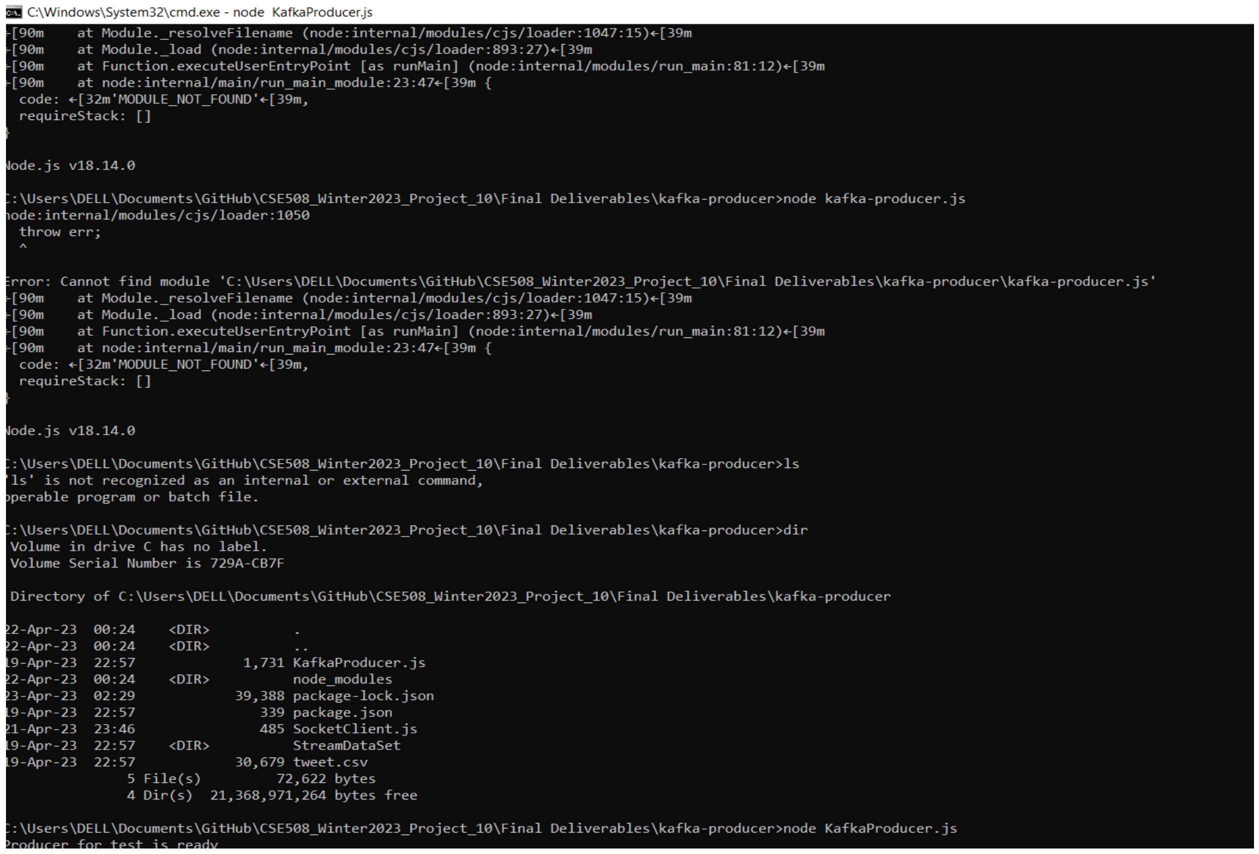
Figure 16: Result snippet



```
[31] from sklearn.metrics import classification_report
     target_names = ['earthquake', 'no earthquake']
     print(classification_report(y_true,df2['Predicted Class'] , target_names=target_names))

                   precision    recall  f1-score   support

      earthquake       0.90      0.77      0.83      2369
   no earthquake       0.75      0.89      0.81      1868

        accuracy                           0.82      4237
       macro avg       0.82      0.83      0.82      4237
    weighted avg       0.83      0.82      0.82      4237
```

Figure 17: Result snippet

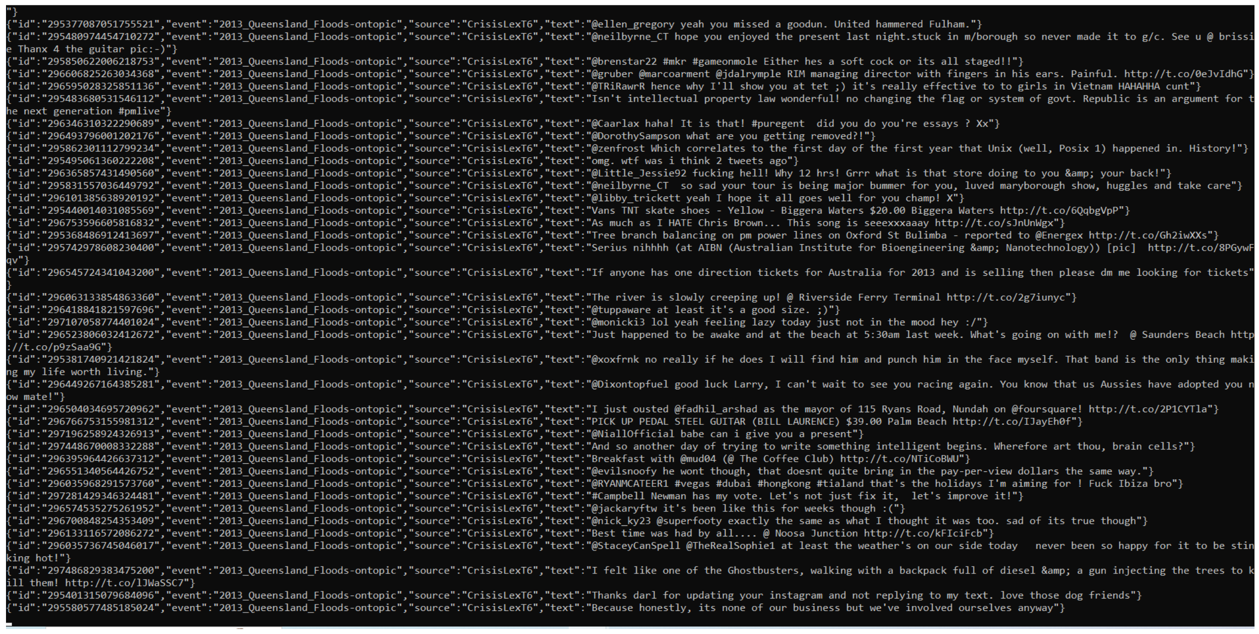
7.2 Kafka Producer



```
C:\Windows\System32\cmd.exe - node  KafkaProducer.js
-[90m    at Module._resolveFilename (node:internal/modules/cjs/loader:1047:15)←[39m
-[90m    at Module._load (node:internal/modules/cjs/loader:893:27)←[39m
-[90m    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)←[39m
-[90m    at node:internal/main/run_main_module:23:47←[39m {
  code: ←[32m'MODULE_NOT_FOUND'←[39m,
  requireStack: []
}

Node.js v18.14.0

:\Users\DELL\Documents\GitHub\CSE508_Winter2023_Project_10\Final Deliverables\kafka-producer>node kafka-producer.js
node:internal/modules/cjs/loader:1050
  throw err;
  ^

Error: Cannot find module 'C:\Users\DELL\Documents\GitHub\CSE508_Winter2023_Project_10\Final Deliverables\kafka-producer\kafka-producer.js'
-[90m    at Module._resolveFilename (node:internal/modules/cjs/loader:1047:15)←[39m
-[90m    at Module._load (node:internal/modules/cjs/loader:893:27)←[39m
-[90m    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)←[39m
-[90m    at node:internal/main/run_main_module:23:47←[39m {
  code: ←[32m'MODULE_NOT_FOUND'←[39m,
  requireStack: []
}

Node.js v18.14.0

:\Users\DELL\Documents\GitHub\CSE508_Winter2023_Project_10\Final Deliverables\kafka-producer>ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

:\Users\DELL\Documents\GitHub\CSE508_Winter2023_Project_10\Final Deliverables\kafka-producer>dir
 Volume in drive C has no label.
 Volume Serial Number is 729A-CB7F

 Directory of C:\Users\DELL\Documents\GitHub\CSE508_Winter2023_Project_10\Final Deliverables\kafka-producer

22-Apr-23  00:24    <DIR>          .
22-Apr-23  00:24    <DIR>          ..
19-Apr-23  22:57             1,731 KafkaProducer.js
22-Apr-23  00:24    <DIR>          node_modules
23-Apr-23  02:29            39,388 package-lock.json
19-Apr-23  22:57               339 package.json
21-Apr-23  23:46               485 SocketClient.js
19-Apr-23  22:57    <DIR>          StreamDataSet
19-Apr-23  22:57            30,679 tweet.csv
               5 File(s)         72,622 bytes
               4 Dir(s)  21,368,971,264 bytes free

:\Users\DELL\Documents\GitHub\CSE508_Winter2023_Project_10\Final Deliverables\kafka-producer>node KafkaProducer.js
Producer for test is ready
```

Figure 18: Kafka Producer- creating a queue of tweets

7.3 Kafka Relevant Live Stream Dataset



```
"}
{"id":"295377087051755521","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@ellen_gregory yeah you missed a goodun. United hammered Fulham."}
{"id":"295480974454710272","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@neilbyrne_CT hope you enjoyed the present last night.stuck in m/borough so never made it to g/c. See u @ brissi
e Thanx 4 the guitar pic:-)"}
{"id":"295850622006218753","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@brenstar22 #mkr #gameonmole Either hes a soft cock or its all staged!!"}
{"id":"296606825263034368","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@gruber @marcoarment @jdalrymple RIM managing director with fingers in his ears. Painful. http://t.co/0eJvIdhG"}
{"id":"296595028325851136","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@TRiRawrR hence why I'll show you at tet ;) it's really effective to to girls in Vietnam HAHAHHA cunt"}
{"id":"295483680531546112","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"Isn't intellectual property law wonderful! no changing the flag or system of govt. Republic is an argument for t
he next generation #pmlive"}
{"id":"296346310322290689","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@Caarlax haha! It is that! #puregent  did you do you're essays ? Xx"}
{"id":"296493796001202176","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@DorothySampson what are you getting removed?!"}
{"id":"295862301112799234","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@zenfrost Which correlates to the first day of the first year that Unix (well, Posix 1) happened in. History!"}
{"id":"295495061360222208","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"omg. wtf was i think 2 tweets ago"}
{"id":"296365857431490560","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@Little_Jessie92 fucking hell! Why 12 hrs! Grrr what is that store doing to you &amp; your back!"}
{"id":"295831557036449792","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@neilbyrne_CT  so sad your tour is being major bummer for you, luved maryborough show, huggles and take care"}
{"id":"296101385638920192","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@libby_trickett yeah I hope it all goes well for you champ! X"}
{"id":"295440014031085569","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"Vans TNT skate shoes - Yellow - Biggera Waters $20.00 Biggera Waters http://t.co/6QqbgVpP"}
{"id":"296753596605816832","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"As much as I HATE Chris Brown... This song is seeexxxaaay http://t.co/sJnUnWgx"}
{"id":"295368486912413697","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"Tree branch balancing on pm power lines on Oxford St Bulimba - reported to @Energex http://t.co/Gh2iwXXs"}
{"id":"295742978608230400","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"Serius nihhhh (at AIBN (Australian Institute for Bioengineering &amp; Nanotechnology)) [pic]  http://t.co/8PGywF
qv"}
{"id":"296545724341043200","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"If anyone has one direction tickets for Australia for 2013 and is selling then please dm me looking for tickets"
}
{"id":"296063133854863360","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"The river is slowly creeping up! @ Riverside Ferry Terminal http://t.co/2g7iunyc"}
{"id":"296418841821597696","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@tuppaware at least it's a good size. ;)"}
{"id":"297107058774401024","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@monicki3 lol yeah feeling lazy today just not in the mood hey :/"}
{"id":"296523806032412672","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"Just happened to be awake and at the beach at 5:30am last week. What's going on with me!?  @ Saunders Beach http
://t.co/p9zSaa9G"}
{"id":"295381740921421824","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@xoxfrnk no really if he does I will find him and punch him in the face myself. That band is the only thing maki
ng my life worth living."}
{"id":"296449267164385281","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@Dixontopfuel good luck Larry, I can't wait to see you racing again. You know that us Aussies have adopted you n
ow mate!"}
{"id":"296504034695720962","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"I just ousted @fadhil_arshad as the mayor of 115 Ryans Road, Nundah on @foursquare! http://t.co/2P1CYTla"}
{"id":"296766753155981312","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"PICK UP PEDAL STEEL GUITAR (BILL LAURENCE) $39.00 Palm Beach http://t.co/IJayEh0f"}
{"id":"297196258924326913","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@NiallOfficial babe can i give you a present"}
{"id":"297448670008332288","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"And so another day of trying to write something intelligent begins. Wherefore art thou, brain cells?"}
{"id":"296395964426637312","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"Breakfast with @mud04 (@ The Coffee Club) http://t.co/NTiCoBWU"}
{"id":"296551340564426752","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@evilsnoofy he wont though, that doesnt quite bring in the pay-per-view dollars the same way."}
{"id":"296035968291573760","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@RYANMCATEER1 #vegas #dubai #hongkong #tialand that's the holidays I'm aiming for ! Fuck Ibiza bro"}
{"id":"297281429346324481","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"#Campbell Newman has my vote. Let's not just fix it,  let's improve it!"}
{"id":"296574535275261952","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@jackaryftw it's been like this for weeks though :("}
{"id":"296700848254353409","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@nick_ky23 @superfooty exactly the same as what I thought it was too. sad of its true though"}
{"id":"296133116572086272","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"Best time was had by all.... @ Noosa Junction http://t.co/kFIciFcb"}
{"id":"296035736745046017","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"@StaceyCanSpell @TheRealSophie1 at least the weather's on our side today   never been so happy for it to be stin
king hot!"}
{"id":"297486829383475200","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"I felt like one of the Ghostbusters, walking with a backpack full of diesel &amp; a gun injecting the trees to k
ill them! http://t.co/lJWaSSC7"}
{"id":"295401315079684096","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"Thanks darl for updating your instagram and not replying to my text. love those dog friends"}
{"id":"295580577485185024","event":"2013_Queensland_Floods-ontopic","source":"CrisisLexT6","text":"Because honestly, its none of our business but we've involved ourselves anyway"}
```

Figure 19: Kafka Producer- creating queue of tweets

7.4 UI Notification Platform

7.4.1 Login Page

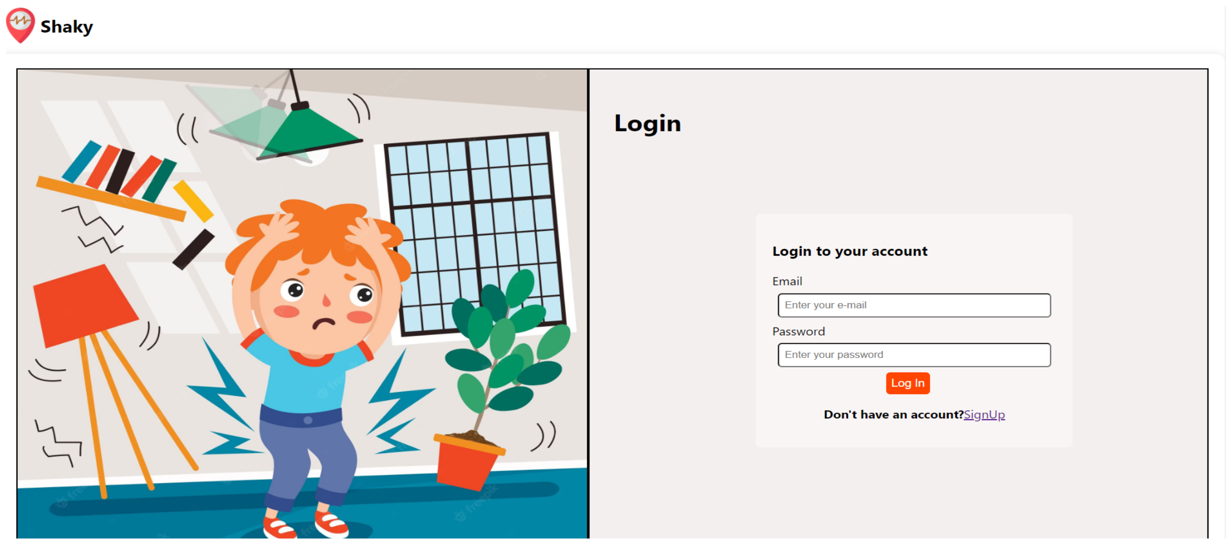


Figure 20: UI snippet

7.4.2 Sign Up Page

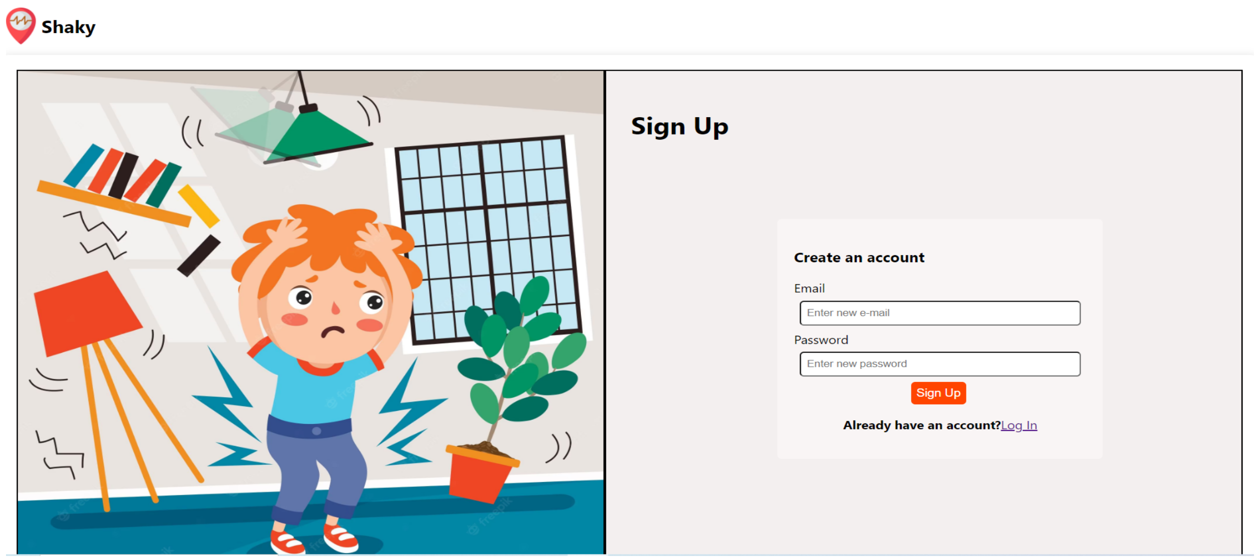
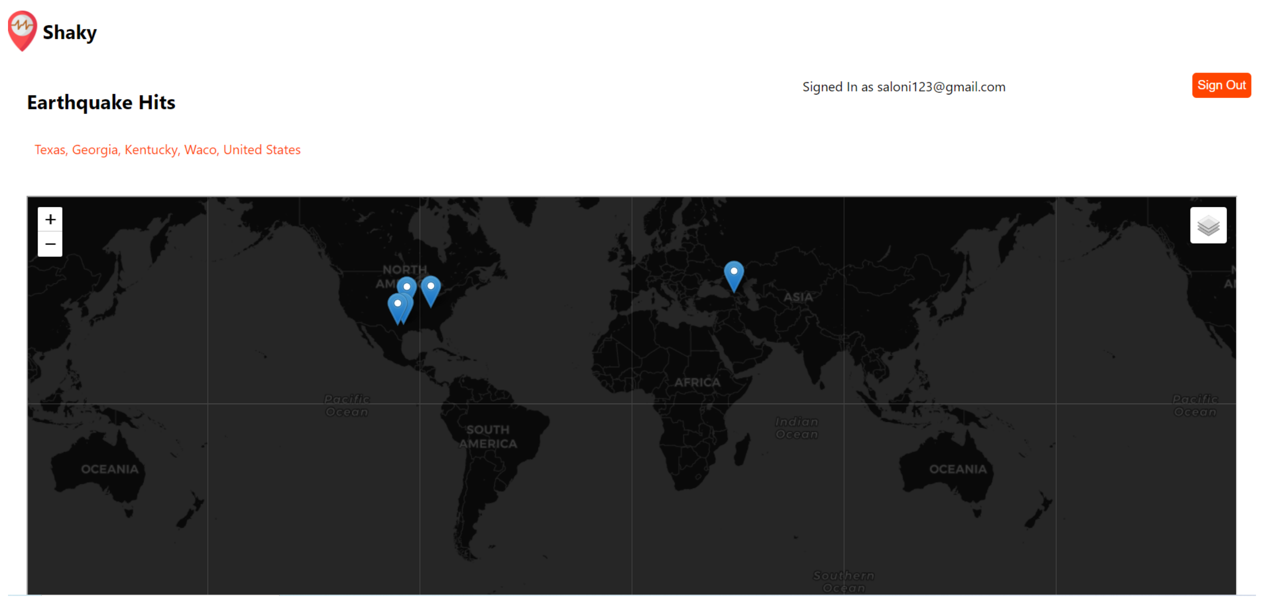


Figure 21: UI snippet

7.4.3 Notification Page


Shaky

Earthquake Hits

Signed In as saloni123@gmail.com

Sign Out

Texas, Georgia, Kentucky, Waco, United States

Figure 22: UI snippet

8 EVALUATION

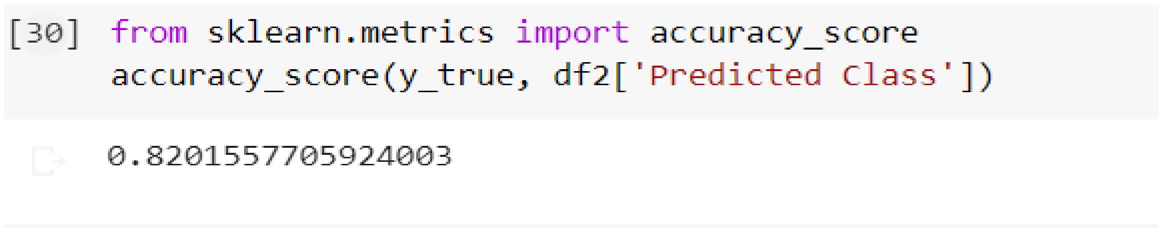
As we planned on using Twitter API and fetching the data as a live stream, due to the unavailability of Twitter Developer Portal access, we performed a Kafka Live Streaming on historical data that we acquired from CrisisLex.

- Comparison with baselines and the system's performance on existing data

  In the baseline results, we have shown that the system's performance was around

  With accuracy= 52%

  And now it has been increased to 82%.

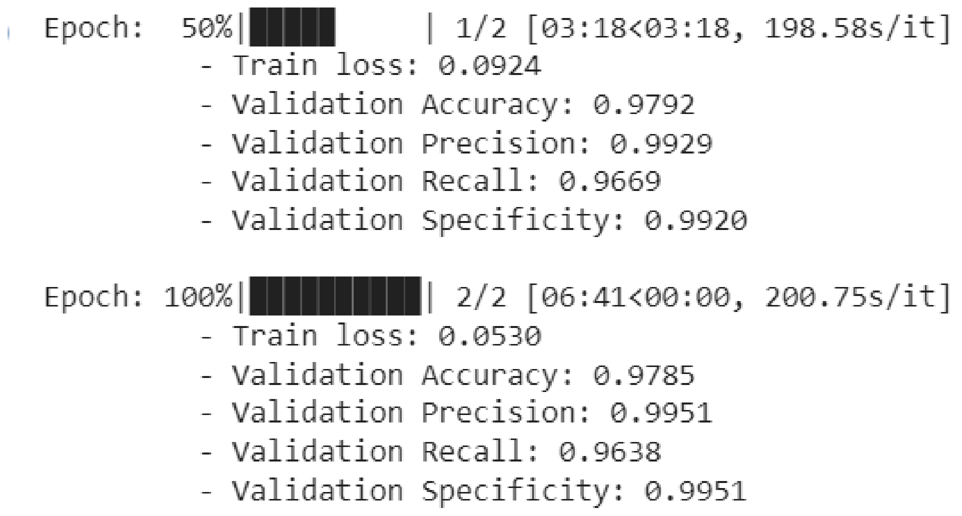


```
[30] from sklearn.metrics import accuracy_score
     accuracy_score(y_true, df2['Predicted Class'])

     0.8201557705924003
```

Figure 23: Result snippet

- SOTA on different evaluation metrics + how the system performs on new data / handles different cases

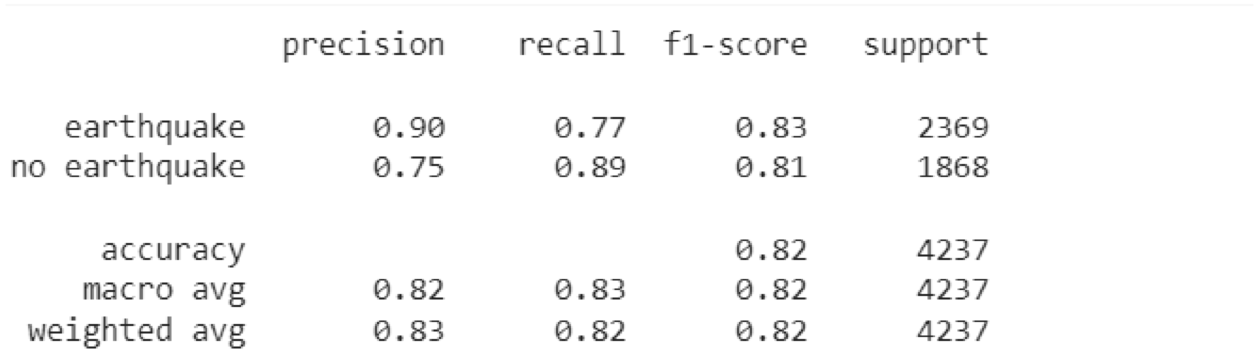
  Our system performance metrics for the training data and validation data is shown below:-



```
Epoch:  50%|█████     | 1/2 [03:18<03:18, 198.58s/it]
          - Train loss: 0.0924
          - Validation Accuracy: 0.9792
          - Validation Precision: 0.9929
          - Validation Recall: 0.9669
          - Validation Specificity: 0.9920

Epoch: 100%|██████████| 2/2 [06:41<00:00, 200.75s/it]
          - Train loss: 0.0530
          - Validation Accuracy: 0.9785
          - Validation Precision: 0.9951
          - Validation Recall: 0.9638
          - Validation Specificity: 0.9951
```

Figure 24: Result snippet

Our system performance metrics for the new data or the test data is shown below:-



```
               precision    recall  f1-score   support

   earthquake       0.90      0.77      0.83      2369
no earthquake       0.75      0.89      0.81      1868

     accuracy                           0.82      4237
    macro avg       0.82      0.83      0.82      4237
 weighted avg       0.83      0.82      0.82      4237
```

Figure 25: Result snippet

9 CONCLUSION

The papers described in the Literature Review mainly focus on the location present within the tweets. Therefore to improve and for better location prediction, we have added an additional feature to get the locations present in the geolocations of the user who has tweeted and compare these locations to get the most accurate location. We have also trained our model to classify the tweets, based on the BERT algorithm and it is efficiently able to classify on any given data.

An additional feature has been implemented that none of the paper has implemented, that is a feature to notify the NGOs or users in real-time where the earthquake has been stuck. Our system provides a user-friendly platform which informs the user/NGOs so that help can be provided faster.

10 REFERENCES

[1] Van Quan, N., Yang, H. J., Kim, K., & Oh, A. R. (2017, April). Real-time earthquake detection using convolutional neural networks and social data. In 2017 IEEE Third International Conference on Multimedia Big Data (BigMM) (pp. 154–157). IEEE.CNN model trained from the tweet related to an earthquake is used for classification. https://ieeexplore.ieee.org/abstract/document/7966735
[2] Sakaki, T., Okazaki, M., & Matsuo, Y. (2010, April). Earthquake shakes Twitter users:
real-time event detection by social sensors. In Proceedings of the 19th international conference on the World wide web (pp. 851–860).
https://dl.acm.org/doi/abs/10.1145/1772690.1772777
[3] Real-time earthquake detection using Twitter tweets. AIP Conference Proceedings 2520, 030014 (2022); https://doi.org/10.1063/5.0102903 Eldho Ittan Georgea) and Cerene Mariam Abrahamb)
[4] Alex Burns, Yuxian Eugene Liang "Tools and methods for capturing Twitter data during
natural disasters," FirstMonday, Volume 17, Number 4 - 2 April 2012
https://firstmonday.org/article/view/3937/3193