```
pip install nlpaug
```

```
Requirement already satisfied: nlpaug in
/usr/local/lib/python3.7/dist-packages (1.1.7)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import nlpaug.augmenter.word as naw

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding,LSTM
from tensorflow.keras.preprocessing.sequence import pad_sequences

train_data=pd.read_csv("/content/drive/MyDrive/NLP
PROJECT/TWITTER/train.csv")
test_data=pd.read_csv("/content/drive/MyDrive/NLP
PROJECT/TWITTER/test.csv")

train_data.head()
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

```
test_data.head()
```

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedic...
1	31964	@user #white #supremacists want everyone to s...
2	31965	safe ways to heal your #acne!! #altwaystohe...
3	31966	is the hp and the cursed child book up for res...
4	31967	3rd #bihday to my amazing, hilarious #nephew...

```
train_data.shape
```

```
(31962, 3)
```

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31962 entries, 0 to 31961
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      31962 non-null  int64
 1   label   31962 non-null  int64
 2   tweet   31962 non-null  object
dtypes: int64(2), object(1)
memory usage: 749.2+ KB
```

PREPROCESSING TRAING DATA

```
clean_data = train_data.append(test_data,ignore_index= True)

# removes pattern in the input text
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for word in r:
        input_txt = re.sub(word, "", input_txt)
    return input_txt

# remove twitter handles (@user)
clean_data['clean_tweet'] = np.vectorize(remove_pattern)
(clean_data['tweet'], "@user[\w]*")

clean_data.head()
```

```
   id  ...                                        clean_tweet
0   1  ...    when a father is dysfunctional and is so sel...
1   2  ...    thanks for #lyft credit i can't use cause th...
2   3  ...                                 bihday your majesty
3   4  ...  #model   i love u take with u all the time in ...
4   5  ...             factsguide: society now    #motivation

[5 rows x 4 columns]
```

```
# remove special characters, numbers and punctuations
clean_data['clean_tweet'] =
clean_data['clean_tweet'].str.replace("[^a-zA-Z#]", " ")
clean_data.head()
```

```
   id  ...                                        clean_tweet
0   1  ...    when a father is dysfunctional and is so sel...
1   2  ...    thanks for #lyft credit i can t use cause th...
2   3  ...                                 bihday your majesty
3   4  ...  #model   i love u take with u all the time in ...
4   5  ...             factsguide  society now    #motivation

[5 rows x 4 columns]
```

```
# remove short words
clean_data['clean_tweet'] = clean_data['clean_tweet'].apply(lambda x:
" ".join([w for w in x.split() if len(w)>3]))
clean_data.head()
```

```
   id  ...                                          clean_tweet
0   1  ...  when father dysfunctional selfish drags kids i...
1   2  ...  thanks #lyft credit cause they offer wheelchai...
2   3  ...                                bihday your majesty
3   4  ...                           #model love take with time
4   5  ...                      factsguide society #motivation

[5 rows x 4 columns]
```

```
# individual words considered as tokens
tokenized_tweet = clean_data['clean_tweet'].apply(lambda x: x.split())
tokenized_tweet.head()
```

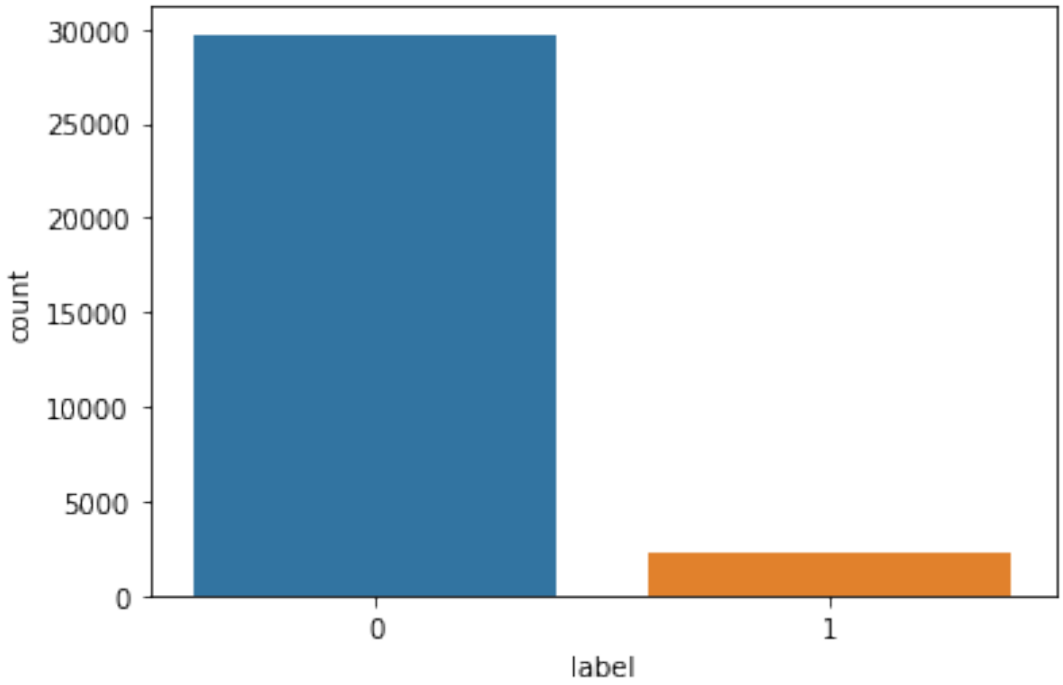
```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, cause, they, offer, wh...
2                           [bihday, your, majesty]
3                    [#model, love, take, with, time]
4                [factsguide, society, #motivation]
Name: clean_tweet, dtype: object
```

```
# combine words into single sentence
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = " ".join(tokenized_tweet[i])

clean_data['clean_tweet'] = tokenized_tweet
clean_data.head()
```

```
   id  ...                                          clean_tweet
0   1  ...  when father dysfunctional selfish drags kids i...
1   2  ...  thanks #lyft credit cause they offer wheelchai...
2   3  ...                                bihday your majesty
3   4  ...                           #model love take with time
4   5  ...                      factsguide society #motivation

[5 rows x 4 columns]
```

```
clean_data.label.value_counts()
sns.countplot(x = 'label',data = train_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd922474f10>
```



DATA BALANCING (USING TEXT AUGUMENTATION)

```
import nltk
nltk.download('wordnet')
```

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

```
True
```

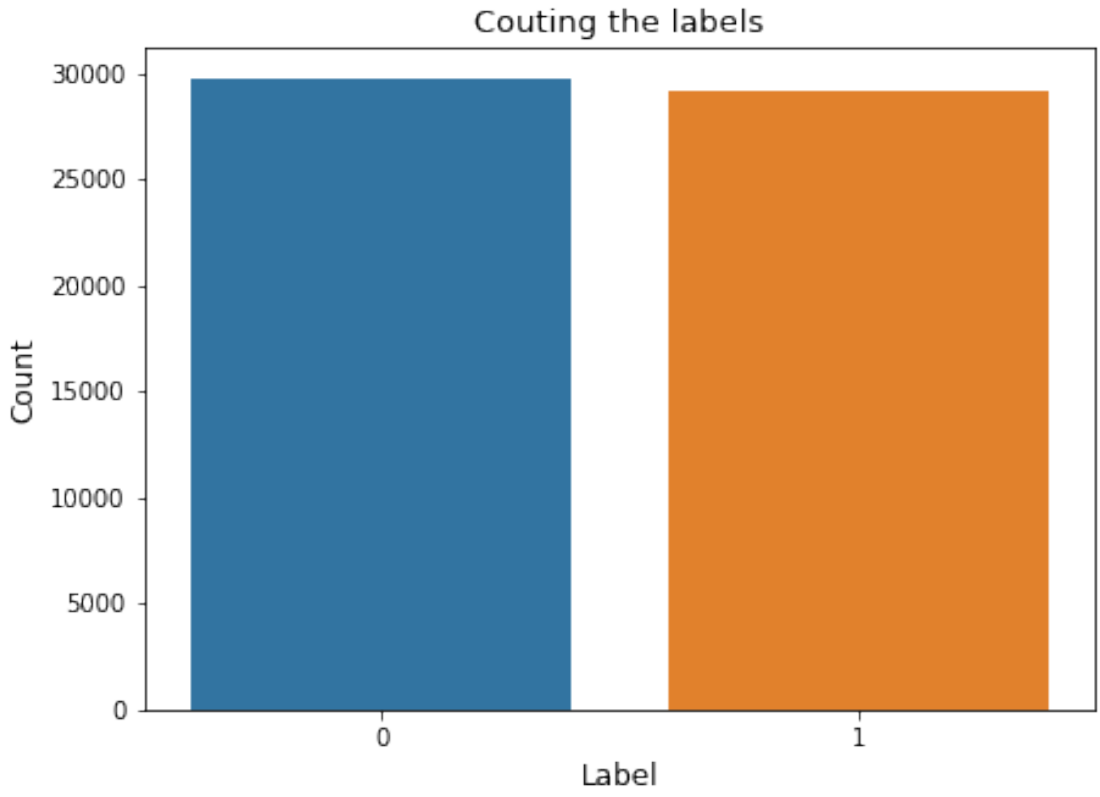
```
# nplaug
import nltk
nltk.download('averaged_perceptron_tagger')
data_resampled_nlpaug = train_data.copy()

aug_texts = []
minority_data = data_resampled_nlpaug[data_resampled_nlpaug['label']
== 1]
aug = naw.SynonymAug(aug_src='wordnet')

texts = minority_data['tweet'].tolist()

for text in texts:
    augmented_texts = aug.augment(text, n=12)

    for augmented in augmented_texts:
        aug_texts.append(augmented)
```

```
print(len(aug_texts))

temp = pd.DataFrame({
    'tweet': aug_texts
})

temp['label'] = 1

data_resampled_nlpaug = pd.concat([data_resampled_nlpaug, temp],
axis=0)
data_resampled_nlpaug = data_resampled_nlpaug.reset_index()
data_resampled_nlpaug = data_resampled_nlpaug.drop(columns=['index'])
del temp, minority_data
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
26904
```

```
plt.figure(figsize=(7, 5))
plt.title('Couting the labels', fontsize=13)
sns.countplot(data=data_resampled_nlpaug, x='label')
plt.xlabel('Label',fontsize=12)
plt.ylabel('Count', fontsize=12)
```

```
Text(0, 0.5, 'Count')
```

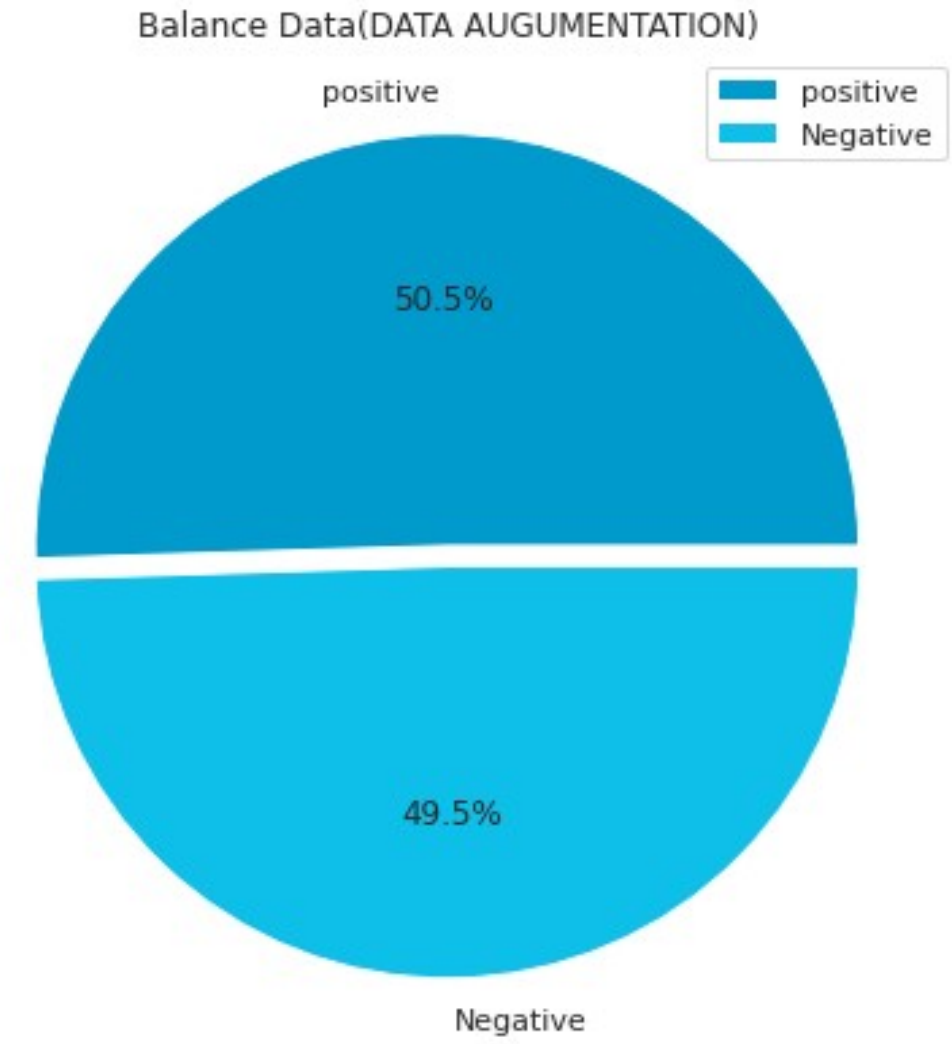



```
counts = pd.DataFrame({
    'Label': data_resampled_nlpaug['label'].value_counts().index,
    'Count': data_resampled_nlpaug['label'].value_counts().values,
    'Percentage':
data_resampled_nlpaug['label'].value_counts().values/data_resampled_nl
paug.shape[0]
})

counts.head()
```

	Label	Count	Percentage
0	0	29720	0.504875
1	1	29146	0.495125

```
sns.set(style="whitegrid")
labels = ["positive", "Negative"]
size = data_resampled_nlpaug["label"].value_counts(sort=True)
colors =['#009ACD','#0EBFE9']
explode = (0.05,0)

plt.figure(figsize=(7,7))
plt.pie(size,explode=explode,labels=labels,colors=colors,autopct='%1.1
f%%')

plt.title('Balance Data(DATA AUGUMENTATION)')
```

```
plt.legend()
plt.show()
```



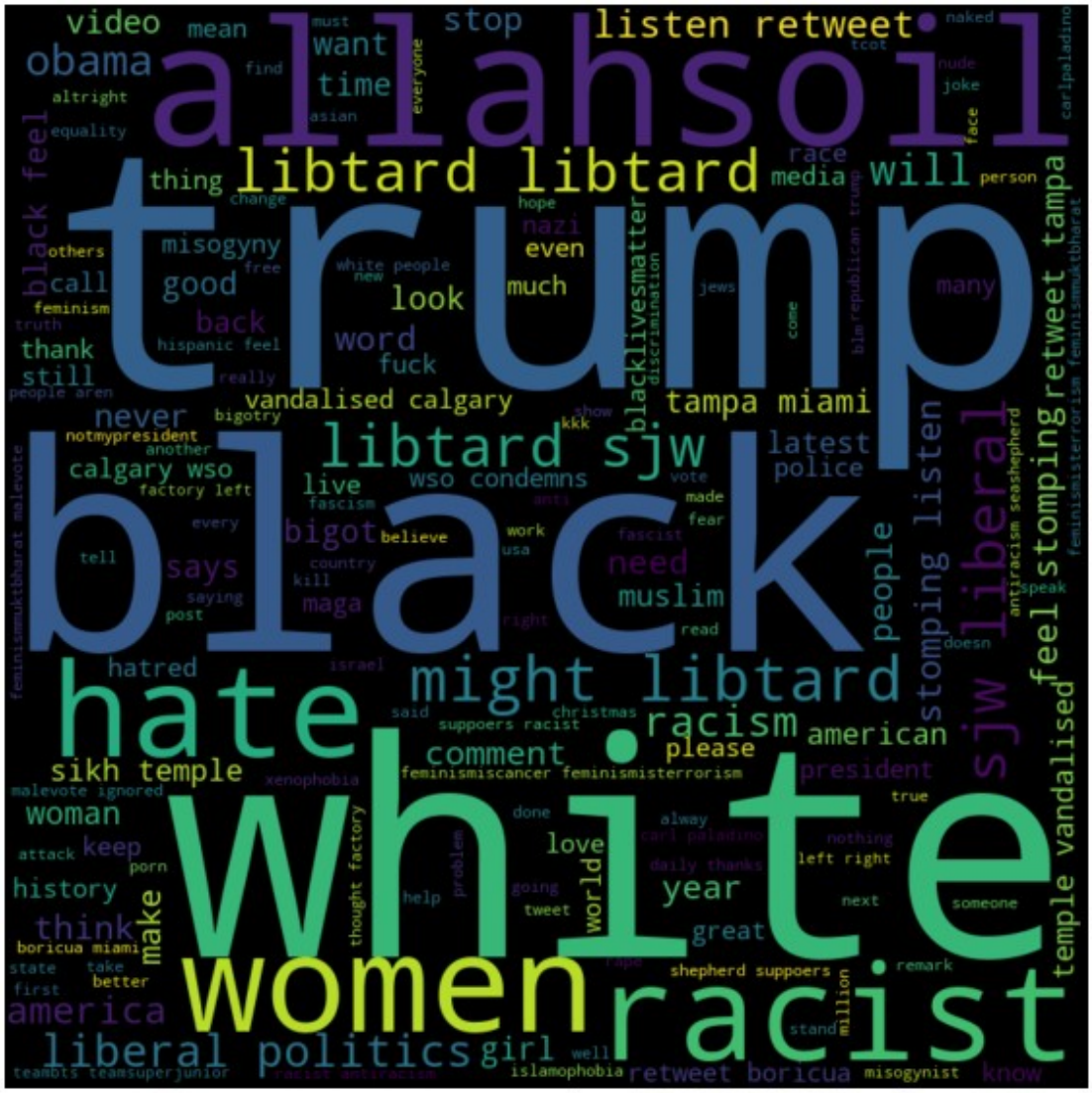
EXPLORATORY DATA ANALYSIS

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

wc = WordCloud(width=800,
               height=800,
               background_color="black",
               min_font_size=10)

positive_reviews = " ".join(clean_data[clean_data["label"] == 0]
["clean_tweet"])
```

```
wc.generate(positive_reviews)

plt.figure(figsize=(10,12))
plt.imshow(wc,interpolation="bilinear")
plt.axis("off")
plt.show()
```


```
wc = WordCloud(width=800,
               height=800,
               background_color="black",
               min_font_size=10)

negative_reviews = " ".join(clean_data[clean_data["label"] == 1]
["clean_tweet"])

wc.generate(negative_reviews)
```

```
plt.figure(figsize=(10,12))
plt.imshow(wc,interpolation="bilinear")
plt.axis("off")
plt.show()
```



CREATING MACHINE LEARNING MODEL

```
X = data_resampled_nlpaug["tweet"]
y = data_resampled_nlpaug["label"]

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,
random_state=1)

from sklearn.feature_extraction.text import CountVectorizer
```

```
# CountVectorizer
cv = CountVectorizer(min_df=0,max_df=1,binary=False,ngram_range=(1,3))
X_train_cv = cv.fit_transform(X_train)
X_test_cv = cv.transform(X_test)

# import Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(X_train_cv, y_train)

RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
class_weight=None,
                       criterion='gini', max_depth=None,
max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0,
min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False,
random_state=None,
                       verbose=0, warm_start=False)

y_pred = rfc.predict(X_test_cv)
print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       1.00      0.11      0.20      9000
           1       0.52      1.00      0.68      8660

    accuracy                           0.55     17660
   macro avg       0.76      0.55      0.44     17660
weighted avg       0.76      0.55      0.43     17660


from sklearn.feature_extraction.text import TfidfVectorizer

# TfidfVectorizer
tf = TfidfVectorizer(stop_words= 'english', ngram_range= (1,3),
lowercase= True, max_features= 5000)
X_train_tf = tf.fit_transform(X_train)
X_test_tf = tf.transform(X_test)

rfc = RandomForestClassifier()
rfc.fit(X_train_tf, y_train)

RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
class_weight=None,
                       criterion='gini', max_depth=None,
max_features='auto',
```

```
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0,
min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False,
random_state=None,
                       verbose=0, warm_start=False)

y_pred = rfc.predict(X_test_tf)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      9000
           1       0.97      0.95      0.96      8660

    accuracy                           0.96     17660
   macro avg       0.96      0.96      0.96     17660
weighted avg       0.96      0.96      0.96     17660
```

CREATING NEURALNETWORK MODEL

```
# Processing text

tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)

vocabulary = tokenizer.index_word
vocabulary
```

```
{1: 'user',
 2: 'the',
 3: 'to',
 4: 'a',
 5: '\x80',
 6: 'you',
 7: 'of',
 8: 'in',
 9: 'and',
 10: '¦',
 11: "'",
 12: 'i',
 13: 'for',
 14: 'is',
 15: 'be',
 16: 'on',
 17: 'this',
 18: 'â',
```

19: 'my',
20: 'that',
21: 'are',
22: 'it',
23: 'with',
24: 'amp',
25: 's',
26: 'all',
27: 'we',
28: 'love',
29: 'so',
30: 'your',
31: 'have',
32: 'at',
33: 'if',
34: 'day',
35: 'by',
36: 'not',
37: 't',
38: 'me',
39: 'what',
40: 'trump',
41: 'like',
42: 'as',
43: 'do',
44: 'just',
45: 'from',
46: 'they',
47: 'when',
48: 'libtard',
49: 'up',
50: 'will',
51: 'no',
52: 'how',
53: 'but',
54: '\x9f',
55: 'out',
56: 'happy',
57: 'ð',
58: 'u',
59: 'exploiter',
60: 'about',
61: 'substance',
62: 'am',
63: 'abuser',
64: 'drug',
65: 'get',
66: 'his',
67: 'one',
68: 'was',

69: 'white',
70: 'new',
71: 'people',
72: 'can',
73: 'time',
74: 'black',
75: 'make',
76: 'who',
77: 'our',
78: 'good',
79: 'more',
80: 'an',
81: 'life',
82: '2',
83: 'allahsoil',
84: 'today',
85: 'world',
86: 'â\x80¦',
87: 'because',
88: 'now',
89: 'he',
90: 'why',
91: 'has',
92: "it's",
93: 'information',
94: 'go',
95: 'us',
96: 'technology',
97: 'take',
98: "i'm",
99: 'only',
100: 'obama',
101: 'or',
102: 'sjw',
103: 'see',
104: 'her',
105: 'positive',
106: 'racist',
107: 'thankful',
108: 'want',
109: '\x98',
110: 'live',
111: '\x99',
112: 'their',
113: 'non',
114: 'against',
115: 'right',
116: 'bihday',
117: 'than',
118: 'feel',

```
119: 'retweet',
120: 'hate',
121: 'being',
122: 'work',
123: 'great',
124: 'over',
125: 'woman',
126: 'should',
127: 're',
128: 'via',
129: "can't",
130: 'smile',
131: 'women',
132: '1',
133: 'think',
134: 'would',
135: 'back',
136: 'there',
137: 'need',
138: 'after',
139: 'politics',
140: 'got',
141: 'here',
142: 'him',
143: 'off',
144: 'america',
145: 'been',
146: 'thanks',
147: 'really',
148: 'girl',
149: 'some',
150: '2016',
151: 'never',
152: 'might',
153: 'she',
154: 'hatred',
155: 'follow',
156: 'way',
157: 'these',
158: 'liberal',
159: "don't",
160: 'much',
161: 'family',
162: 'man',
163: 'first',
164: 'fun',
165: 'weekend',
166: 'sex',
167: 'come',
168: 'say',
```

```
169: 'best',
170: 'know',
171: 'them',
172: 'too',
173: 'health',
174: 'still',
175: 'going',
176: 'stop',
177: 'listen',
178: 'many',
179: 'healthy',
180: 'racism',
181: 'down',
182: 'year',
183: 'number',
184: 'bull',
185: 'had',
186: 'wait',
187: '2017',
188: 'look',
189: 'well',
190: 'very',
191: 'summer',
192: 'don',
193: 'political',
194: 'even',
195: 'days',
196: '3',
197: 'into',
198: 'music',
199: 'video',
200: 'most',
201: 'friends',
202: 'astir',
203: 'friday',
204: 'sad',
205: '4',
206: 'beautiful',
207: 'always',
208: 'then',
209: 'home',
210: 'call',
211: 'week',
212: 'anti',
213: 'show',
214: 'free',
215: 'state',
216: 'wherefore',
217: 'morning',
218: 'ever',
```

```
219: 'let',
220: 'any',
221: 'atomic',
222: 'race',
223: 'find',
224: 'organization',
225: 'every',
226: 'miamiâ',
227: 'brexit',
228: 'news',
229: 'men',
230: 'those',
231: 'pay',
232: 'makes',
233: 'next',
234: 'its',
235: 'gt',
236: 'orlando',
237: 'again',
238: 'last',
239: '\x9d',
240: 'tomorrow',
241: "father's",
242: 'blog',
243: 'fathersday',
244: 'please',
245: 'thank',
246: 'cute',
247: 'tampa',
248: 'power',
249: 'blm',
250: 'girls',
251: 'everyone',
252: 'own',
253: 'where',
254: 'give',
255: 'another',
256: 'hope',
257: 'w',
258: 'same',
259: 'keep',
260: 'stomping',
261: 'bigot',
262: 'model',
263: 'night',
264: 'nothing',
265: '\x9c',
266: 'ampere',
267: 'usa',
268: 'guy',
```

```
269: 'acid',
270: 'little',
271: 'school',
272: 'porn',
273: 'sunday',
274: 'help',
275: 'leave',
276: 'exist',
277: 'monophosphate',
278: 'american',
279: 'adenylic',
280: 'm',
281: 'game',
282: 'real',
283: 'thought',
284: 'affirmation',
285: 'believe',
286: 'old',
287: 'god',
288: 'fuck',
289: 'happiness',
290: "you're",
291: 'person',
292: 'dad',
293: 'other',
294: 'says',
295: 'police',
296: 'finally',
297: 'two',
298: 'things',
299: 'won',
300: 'comments',
301: 'act',
302: '5',
303: 'paladino',
304: 'peace',
305: 'left',
306: 'amazing',
307: 'calgary',
308: 'suppoers',
309: 'cost',
310: 'did',
311: 'wish',
312: 'president',
313: 'selfie',
314: 'someone',
315: 'sikh',
316: 'father',
317: 'made',
318: 'adenosine',
```

```
 319: 'city',
 320: 'watch',
 321: 'such',
 322: 'end',
 323: 'away',
 324: 'wso',
 325: 'condemns',
 326: 'racial',
 327: 'maga',
 328: '\x82',
 329: 'd',
 330: 'without',
 331: 'racialist',
 332: 'urð\x9f\x93±',
 333: 'ð\x9f\x98\x99ð\x9f\x98\x8eð\x9f\x91\x84ð\x9f\x91\x85ð\x9f\
x92¦ð\x9f\x92¦ð\x9f\x92¦',
 334: 'gold',
 335: 'silver',
 336: 'thing',
 337: 'book',
 338: 'before',
 339: 'fathers',
 340: 'word',
 341: 'ready',
 342: 'true',
 343: 'were',
 344: 'malevote',
 345: 'yes',
 346: 'feeling',
 347: 'may',
 348: 'â\x86\x9d',
 349: 'done',
 350: 'states',
 351: 'read',
 352: 'big',
 353: 'lol',
 354: 'embody',
 355: 'represent',
 356: 'also',
 357: 'tonight',
 358: 'proud',
 359: 'personify',
 360: 'single',
 361: 'comprise',
 362: 'looking',
 363: 'everything',
 364: 'altwaystoheal',
 365: 'mean',
 366: 'years',
 367: 'r',
```

```
368: 'equal',
369: 'friend',
370: 'matter',
371: 'food',
372: 'seashepherd',
373: 'n',
374: 'tell',
375: 'better',
376: 'pa',
377: 'yourself',
378: 'kkk',
379: 'lost',
380: 'attack',
381: 'media',
382: 'bear',
383: 'does',
384: 'yet',
385: 'human',
386: 'buffalo',
387: 'around',
388: 'funny',
389: 'job',
390: 'until',
391: 'constitute',
392: 'long',
393: 'sta',
394: 'kids',
395: 'bad',
396: 'hot',
397: 'latest',
398: 'having',
399: 'play',
400: 'check',
401: 'place',
402: 'mightiness',
403: 'ppl',
404: 'change',
405: 'b',
406: 'face',
407: 'country',
408: 'team',
409: 'hard',
410: 'tweet',
411: 'forex',
412: 'enjoy',
413: 'feminismiscancer',
414: 'feminismisterrorism',
415: 'feminismmuktbharat',
416: 'healing',
417: '\x8f',
```

```
418: 'guys',
419: 'discrimination',
420: 'already',
421: 'young',
422: 'post',
423: 'government',
424: 'ignorance',
425: 'science',
426: 'simply',
427: 'history',
428: 've',
429: 'mind',
430: 'uk',
431: 'united',
432: 'whatever',
433: '\x96',
434: 'climb',
435: 'hey',
436: 'antiracism',
437: 'coming',
438: 'said',
439: 'others',
440: 'business',
441: 'hispanic',
442: 'could',
443: 'getting',
444: 'vote',
445: 'blacklivesmatter',
446: 'miss',
447: 'helium',
448: 'use',
449: 'baby',
450: 'isn',
451: 'lt',
452: 'blessed',
453: 'nude',
454: 'shit',
455: 'republican',
456: 'christmas',
457: 'similar',
458: 'nice',
459: 'trumpet',
460: 'saying',
461: 'altright',
462: 'daily',
463: 'win',
464: 'americans',
465: 'sun',
466: 'carl',
467: 'saturday',
```

```
468: 'gets',
469: 'angry',
470: 'must',
471: 'gorilla',
472: '\x92',
473: '\x87',
474: 'found',
475: 'card',
476: 'im',
477: 'notmypresident',
478: 'money',
479: 'muslim',
480: 'awesome',
481: 'hear',
482: 'misogyny',
483: 'instagood',
484: 'far',
485: 'misogynist',
486: 'dog',
487: 'temple',
488: 'horn',
489: 'both',
490: 'ur',
491: 'doing',
492: 'boy',
493: 'house',
494: 'put',
495: 'hea',
496: 'wow',
497: 'through',
498: 'direct',
499: "i've",
500: 'rest',
501: '10',
502: 'boycott',
503: 'excited',
504: 'while',
505: 'extent',
506: 'soon',
507: 'suppo',
508: '8',
509: 'holiday',
510: 'cool',
511: 'vandalised',
512: 'truth',
513: 'full',
514: 'polar',
515: 'doesn',
516: 'carlpaladino',
517: 'tree',
```

518: 'strong',
519: 'ain',
520: '\x91',
521: 'gonna',
522: '¤',
523: 'resist',
524: 'male',
525: 'enough',
526: 'oh',
527: '\x95',
528: 'ally',
529: 'quote',
530: 'twitter',
531: 'cornet',
532: 'run',
533: 'oil',
534: 'something',
535: 'waiting',
536: 'beach',
537: '¸',
538: 'lot',
539: 'comment',
540: 'wedding',
541: 'watching',
542: 'greater',
543: 'few',
544: 'hair',
545: 'hillary',
546: "that's",
547: 'dominate',
548: 'try',
549: 'needs',
550: 'cnn',
551: 'point',
552: 'yeah',
553: 'kind',
554: 'sexy',
555: 'self',
556: 'leadership',
557: 'teambts',
558: 'body',
559: 'grateful',
560: 'class',
561: 'which',
562: 'lady',
563: 'emiratis',
564: 'fear',
565: 'children',
566: 'videos',
567: 'picture',

568: '\x8e',
569: 'forward',
570: 'together',
571: 'bigotry',
572: 'tcot',
573: 'homophobic',
574: 'israel',
575: 'sick',
576: 'talk',
577: 'looks',
578: 'top',
579: 'story',
580: 'bit',
581: '7',
582: 'lovely',
583: 'motivation',
584: 'times',
585: 'southafrica',
586: 'ace',
587: 'towards',
588: 'fashion',
589: 'anything',
590: 'xenophobia',
591: 'making',
592: 'hold',
593: 'sea',
594: 'putinschoice',
595: 'unity',
596: '6',
597: 'travel',
598: 'care',
599: 'course',
600: 'once',
601: 'late',
602: '\x93',
603: 'gay',
604: 'son',
605: 'open',
606: 'fascism',
607: 'ass',
608: 'sleep',
609: 'female',
610: 'talking',
611: 'child',
612: 'african',
613: 'stay',
614: 'playing',
615: 'monday',
616: 'aren',
617: 'wrong',

```
618: 'fucking',
619: 'omg',
620: 'joy',
621: 'forget',
622: 'bing',
623: 'ã',
624: 'mad',
625: 'lgbt',
626: 'c',
627: 'naked',
628: 'half',
629: 'o',
630: 'vs',
631: 'operating',
632: 'bong',
633: 'hour',
634: 'order',
635: 'buy',
636: 'fans',
637: 'couple',
638: 'weeks',
639: 'michelle',
640: 'comes',
641: 'social',
642: 'trying',
643: 'merely',
644: 'reading',
645: 'pretty',
646: 'cause',
647: 'crazy',
648: 'newyork',
649: 'death',
650: 'flag',
651: 'remember',
652: 'relation',
653: 'islamophobia',
654: '9',
655: 'problem',
656: 'words',
657: 'â\x9d¤ï¸\x8f',
658: 'each',
659: 'pine',
660: 'cow',
661: 'south',
662: 'boricua',
663: 'head',
664: 'yay',
665: 'tweets',
666: 'factory',
667: 'early',
```

668: 'theresistance',
669: 'dead',
670: 'ignored',
671: 'rip',
672: 'followme',
673: 'x',
674: 'equality',
675: 'whites',
676: 'fight',
677: 'june',
678: 'photooftheday',
679: 'meet',
680: 'else',
681: 'islam',
682: 'anyone',
683: 'fuhered',
684: 'speak',
685: 'uselections2016',
686: 'join',
687: '©',
688: 'high',
689: 'mass',
690: 'color',
691: 'war',
692: 'boricuaâ',
693: 'goes',
694: 'complete',
695: "we're",
696: 'force',
697: 'disgusting',
698: 'campaign',
699: 'hither',
700: 'thats',
701: 'didn',
702: 'poetry',
703: 'present',
704: 'living',
705: 'hours',
706: 'lives',
707: 'send',
708: 'experience',
709: 'tbt',
710: 'shepherd',
711: 'bring',
712: 'culture',
713: 'national',
714: 'since',
715: 'list',
716: 'movement',
717: 'religion',

718: 'ï',
719: 'racing',
720: 'japan',
721: 'turn',
722: 'cat',
723: 'lie',
724: 'close',
725: 'impoant',
726: 'share',
727: 'joke',
728: 'racialism',
729: 'jews',
730: 'song',
731: 'came',
732: 'co',
733: 'worst',
734: 'asian',
735: 'semite',
736: 'nazi',
737: 'mindset',
738: 'working',
739: 'fascist',
740: 'shop',
741: 'group',
742: 'themselves',
743: 'photo',
744: 'shooting',
745: 'office',
746: 'church',
747: 'action',
748: 'chair',
749: 'latino',
750: 'miami',
751: "doesn't",
752: '\x9a',
753: 'cold',
754: 'public',
755: 'dear',
756: 'dark',
757: 'become',
758: 'quotes',
759: 'less',
760: 'sorry',
761: 'teambtsâ',
762: 'ð\x9f\x98\x8d',
763: 'almost',
764: 'muslims',
765: '\x84',
766: '\x8c',
767: 'mother',

768: 'target',
769: 'alone',
770: 'gop',
771: '¬',
772: 'ð\x9f\x98\x8a',
773: 'law',
774: 'gone',
775: 'stupid',
776: 'euro2016',
777: 'service',
778: 'islamic',
779: 'perfect',
780: "didn't",
781: 'room',
782: 'kill',
783: 'stand',
784: 'football',
785: 'land',
786: "ne'er",
787: 'loving',
788: 'under',
789: 'wants',
790: 'heed',
791: 'moment',
792: 'education',
793: 'clean',
794: 'message',
795: 'military',
796: 'fire',
797: 'tear',
798: 'bbc',
799: 'dads',
800: 'wall',
801: 'christian',
802: 'community',
803: 'account',
804: 'ask',
805: 'e',
806: 'set',
807: 'allow',
808: 'maine',
809: 'calling',
810: 'aicle',
811: 'maybe',
812: 'obamas',
813: 'dance',
814: 'remark',
815: 'rape',
816: 'london',
817: 'ocean',

```
818: 'delete',
819: 'ugly',
820: "i'll",
821: 'agree',
822: 'though',
823: '\x94',
824: 'disease',
825: 'different',
826: 'inspiration',
827: 'brown',
828: 'mom',
829: 'fall',
830: 'month',
831: 'cleaning',
832: 'actually',
833: 'pic',
834: 'environment',
835: 'called',
836: 'adult',
837: "won't",
838: 'super',
839: 'celebrate',
840: 'newyear',
841: 'york',
842: 'australia',
843: 'bed',
844: 'means',
845: 'ok',
846: 'green',
847: 'depression',
848: '2016in4words',
849: 'feminism',
850: 'ane',
851: 'systemic',
852: 'nyc',
853: 'guess',
854: 'line',
855: 'beauty',
856: 'ago',
857: 'board',
858: 'seems',
859: 'thursday',
860: 'denial',
861: 'ignorant',
862: 'hand',
863: 'serve',
864: 'season',
865: 'pussy',
866: 'truly',
867: 'conference',
```

868: 'side',
869: 'success',
870: 'street',
871: 'wtf',
872: 'boys',
873: 'terrorism',
874: 'liar',
875: 'till',
876: 'movie',
877: 'fitness',
878: 'shows',
879: 'election',
880: '¡',
881: 'kid',
882: 'fact',
883: 'tv',
884: 'tired',
885: 'wishes',
886: 'fair',
887: 'calls',
888: 'simulator',
889: 'adapt',
890: 'hello',
891: 'tech',
892: 'gift',
893: 'terrorist',
894: 'sympathies',
895: 'ii',
896: '53',
897: 'stopracism',
898: 'special',
899: 'donaldtrump',
900: 'profiling',
901: 'months',
902: 'form',
903: 'due',
904: 'gender',
905: 'mood',
906: 'forever',
907: 'russia',
908: 'justice',
909: 'happened',
910: 'thoughts',
911: 'protesting',
912: 'india',
913: 'smh',
914: 'youtube',
915: 'victims',
916: 'die',
917: 'freedom',

```
918: 'blue',
919: 'practice',
920: 'whitepeople',
921: 'flight',
922: 'reason',
923: "isn't",
924: 'understand',
925: 'learn',
926: 'seeing',
927: 'sure',
928: 'corresponding',
929: 'yr',
930: '¢',
931: 'piece',
932: 'texas',
933: "he's",
934: 'dream',
935: 'small',
936: 'dont',
937: 'simulation',
938: 'coffee',
939: 'africa',
940: 'idea',
941: 'anymore',
942: 'accept',
943: 'alt',
944: 'ahead',
945: 'medium',
946: 'security',
947: 'hollywood',
948: 'vacation',
949: 'woh',
950: 'realize',
951: '2016in4wordsâ',
952: 'network',
953: 'fucked',
954: 'putin',
955: 'along',
956: 'either',
957: 'loved',
958: 'stamp',
959: 'term',
960: 'correct',
961: 'daughter',
962: 'went',
963: 'chick',
964: 'laugh',
965: 'international',
966: 'age',
967: 'running',
```

```
 968: 'soul',
 969: 'welcome',
 970: 'eah',
 971: 'wanna',
 972: 'wife',
 973: 'market',
 974: 'light',
 975: 'everyday',
 976: 'past',
 977: 'instagram',
 978: 'walk',
 979: 'pig',
 980: 'future',
 981: 'treason',
 982: 'officer',
 983: 'ð\x9f\x98\x81',
 984: 'least',
 985: 'homes',
 986: 'donald',
 987: 'development',
 988: 'p',
 989: 'despite',
 990: 'wear',
 991: 'inequality',
 992: 'seen',
 993: 'damn',
 994: 'ð\x9f\x98\x82',
 995: 'wonderful',
 996: 'ag',
 997: 'shot',
 998: 'lose',
 999: 'lily',
 1000: 'used',
 ...}
```

```
vocab_len = len(vocabulary)
vocab_len
```

```
41102
```

```
train_sequence = tokenizer.texts_to_sequences(X_train)
```

```
doc_len = []
for doc in train_sequence:
    doc_len.append(len(doc))
```

```
max(doc_len)
```

```
68
```

```
np.quantile(doc_len, 0.99)
```

```
30.0
```

```
max_len = 30

train_sequence_matrix = sequence.pad_sequences(train_sequence, maxlen=
max_len)
```

```
train_sequence_matrix
```

```
array([[    0,     0,     0, ...,   114,  1226,  2525],
       [    0,     0,     0, ...,   335,   334,   411],
       [    0,     0,     0, ...,     1,     1,  5572],
       ...,
       [    0,     0,     0, ...,   145, 13929, 41098],
       [    0,     0,     0, ..., 41101, 16258, 41102],
       [    0,     0,     0, ..., 17584,  7020,   893]], dtype=int32)
```

```
# testing
test_sequence = tokenizer.texts_to_sequences(X_test)
test_sequence_matrix = sequence.pad_sequences(test_sequence, maxlen=
max_len)
```

```
test_sequence_matrix
```

```
array([[   11,    37,    15, ...,  7506,     5,    10],
       [    0,     0,     0, ...,    14,  2272,   173],
       [    0,     0,     0, ...,  8078,     5,    10],
       ...,
       [    0,     0,     0, ...,  2981,   751,  2426],
       [    0,     0,     0, ...,     3,     2,  1551],
       [    0,     0,     0, ..., 15390,   240, 11580]], dtype=int32)
```

```
# LSTM
model = Sequential()
model.add(Embedding(input_dim=vocab_len+1,output_dim=100,input_length=
max_len,mask_zero=True))
model.add(LSTM(32, activation="tanh"))
model.add(Dense(64,activation="tanh"))
model.add(Dense(64,activation="tanh"))
model.add(Dense(1,activation="sigmoid"))
```

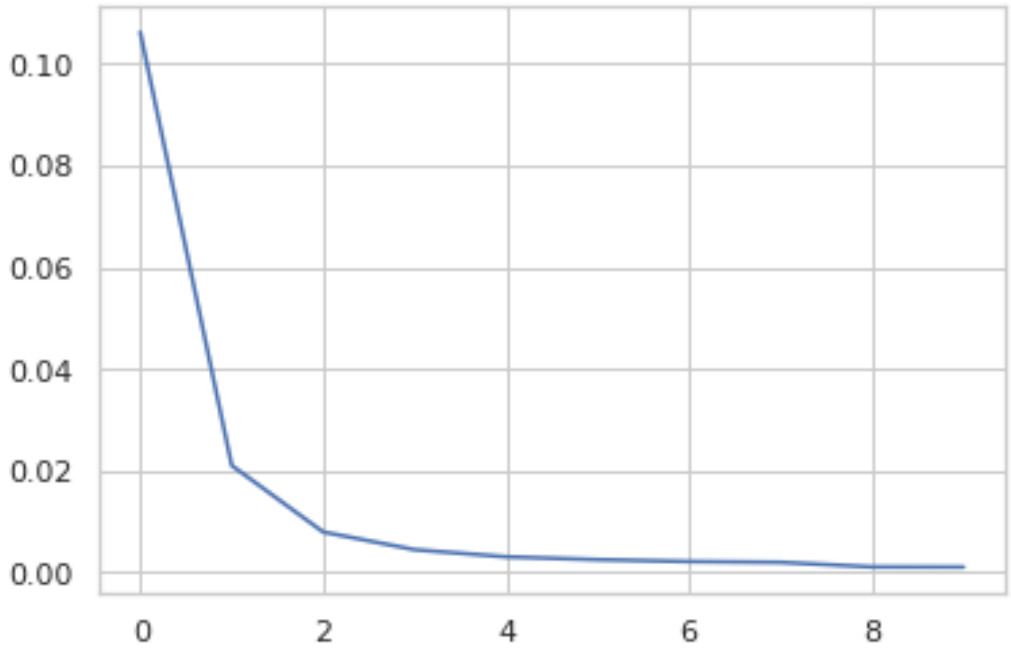
```
model.compile(optimizer="adam", loss="binary_crossentropy")
trained_model=model.fit(train_sequence_matrix,y_train,
batch_size=32,epochs=10)
```

```
Epoch 1/10
1288/1288 [==============================] - 86s 64ms/step - loss:
0.1062
Epoch 2/10
1288/1288 [==============================] - 82s 64ms/step - loss:
0.0210
Epoch 3/10
```

```
1288/1288 [==============================] - 82s 64ms/step - loss:
0.0079
Epoch 4/10
1288/1288 [==============================] - 82s 64ms/step - loss:
0.0045
Epoch 5/10
1288/1288 [==============================] - 82s 64ms/step - loss:
0.0030
Epoch 6/10
1288/1288 [==============================] - 82s 63ms/step - loss:
0.0025
Epoch 7/10
1288/1288 [==============================] - 81s 63ms/step - loss:
0.0021
Epoch 8/10
1288/1288 [==============================] - 81s 63ms/step - loss:
0.0020
Epoch 9/10
1288/1288 [==============================] - 81s 63ms/step - loss:
0.0011
Epoch 10/10
1288/1288 [==============================] - 81s 63ms/step - loss:
0.0010
```

```
plt.plot(trained_model.history["loss"])
```

```
[<matplotlib.lines.Line2D at 0x7fd907d47c90>]
```



```
y_pred = model.predict(test_sequence_matrix)
y_pred = np.where(y_pred >= 0.5, 1, 0)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      9000
           1       0.97      0.99      0.98      8660

    accuracy                           0.98     17660
   macro avg       0.98      0.98      0.98     17660
weighted avg       0.98      0.98      0.98     17660
```

PREPROCESSING TESTING DATA

```
test_data.head()
```

```
      id                                                tweet
0  31963  #studiolife #aislife #requires #passion #dedic...
1  31964   @user #white #supremacists want everyone to s...
2  31965  safe ways to heal your #acne!!    #altwaystohe...
3  31966  is the hp and the cursed child book up for res...
4  31967    3rd #bihday to my amazing, hilarious #nephew...
```

```
# removes pattern in the input text
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for word in r:
        input_txt = re.sub(word, "", input_txt)
    return input_txt

# remove twitter handles (@user)
test_data['clean_tweet'] = np.vectorize(remove_pattern)
(test_data['tweet'], "@[\w]*")

test_data.head()
```

```
      id  ...                                        clean_tweet
0  31963  ...  #studiolife #aislife #requires #passion #dedic...
1  31964  ...    #white #supremacists want everyone to see th...
2  31965  ...  safe ways to heal your #acne!!    #altwaystohe...
3  31966  ...  is the hp and the cursed child book up for res...
4  31967  ...    3rd #bihday to my amazing, hilarious #nephew...

[5 rows x 3 columns]
```

```
# remove special characters, numbers and punctuations
test_data['clean_tweet'] = test_data['clean_tweet'].str.replace("[^a-
zA-Z#]", " ")
test_data.head()
```

```
      id  ...                                            clean_tweet
0  31963  ...  #studiolife #aislife #requires #passion #dedic...
1  31964  ...    #white #supremacists want everyone to see th...
2  31965  ...  safe ways to heal your #acne      #altwaystohe...
3  31966  ...  is the hp and the cursed child book up for res...
4  31967  ...     rd #bihday to my amazing  hilarious #nephew...

[5 rows x 3 columns]
```

```
# remove short words
test_data['clean_tweet'] = test_data['clean_tweet'].apply(lambda x: "
".join([w for w in x.split() if len(w)>3]))
test_data.head()
```

```
      id  ...                                            clean_tweet
0  31963  ...  #studiolife #aislife #requires #passion #dedic...
1  31964  ...  #white #supremacists want everyone #birds #mov...
2  31965  ...  safe ways heal your #acne #altwaystoheal #heal...
3  31966  ...  cursed child book reservations already where w...
4  31967  ...  #bihday amazing hilarious #nephew ahmir uncle ...

[5 rows x 3 columns]
```

```
# individual words considered as tokens
tokenized_tweet = test_data['clean_tweet'].apply(lambda x: x.split())
tokenized_tweet.head()
```

```
0    [#studiolife, #aislife, #requires, #passion, #...
1    [#white, #supremacists, want, everyone, #birds...
2    [safe, ways, heal, your, #acne, #altwaystoheal...
3    [cursed, child, book, reservations, already, w...
4    [#bihday, amazing, hilarious, #nephew, ahmir, ...
Name: clean_tweet, dtype: object
```

```
# combine words into single sentence
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = " ".join(tokenized_tweet[i])

test_data['clean_tweet'] = tokenized_tweet
test_data.head()
```

```
      id  ...                                            clean_tweet
0  31963  ...  #studiolife #aislife #requires #passion #dedic...
1  31964  ...  #white #supremacists want everyone #birds #mov...
2  31965  ...  safe ways heal your #acne #altwaystoheal #heal...
3  31966  ...  cursed child book reservations already where w...
4  31967  ...  #bihday amazing hilarious #nephew ahmir uncle ...

[5 rows x 3 columns]
```

```
test=test_data.clean_tweet
```

```
# Processing text
# testing
test_sequence = tokenizer.texts_to_sequences(test)
test_sequence_matrix = sequence.pad_sequences(test_sequence, maxlen=
max_len)

y_pred=model.predict(test_sequence_matrix)
y_pred = np.where(y_pred >= 0.5, 1, 0)

print(y_pred)

[[0]
 [1]
 [0]
 ...
 [0]
 [0]
 [0]]

test['label'] = (y_pred >= 0.5).astype(np.int)

predictions = pd.DataFrame(y_pred, columns=['Prediction'])

df = pd.concat([test, predictions], axis =1)

df.head()
```

	clean_tweet	Prediction
0	#studiolife #aislife #requires #passion #dedic...	0.0
1	#white #supremacists want everyone #birds #mov...	1.0
2	safe ways heal your #acne #altwaystoheal #heal...	0.0
3	cursed child book reservations already where w...	0.0
4	#bihday amazing hilarious #nephew ahmir uncle ...	0.0