budowa_krok_7_2

April 9, 2024

1 Predicting tweet sentiment

Dataset from https://www.kaggle.com/datasets/bhavikjikadara/tweets-dataset

Context

This is the sentiment140 dataset. It contains 1,600,000 tweets extracted using the Twitter API. The tweets have been annotated (0 = negative, 4 = positive) and can be used to detect sentiment.

Content

It contains the following 6 fields:

- target: the polarity of the tweet (0 = negative and 4 = positive)
- ids: The id of the tweet (2087)
- date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
- flag: The query (lyx). If there is no query, then this value is NO_QUERY.
- user: the user that tweeted.
- text: the text of the tweet.

1.1 1. Exploratory Data Analysis

[ ]:
```
# import libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

# %pip install contractions, wordcloud, tensorflow

# text processing libraries
import re
import contractions

from collections import Counter
# import string
import nltk
# import warnings
# %matplotlib inline
# warnings.filterwarnings("ignore")
```

```
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud

from nltk.stem import WordNetLemmatizer
nltk.download("wordnet")
nltk.download("omw-1.4")

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, accuracy_score, confusion_matrix

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\flang\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\flang\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Pandas and Numpy have been used for data manipulation and numerical calculations
Matplotlib and Seaborn have been used for data visualizations

[ ]:
```
# import data
tweets = pd.read_csv("..//data//tweets.csv", encoding="latin-1")
```

[ ]:
```
tweets.head()
```

[ ]:

	Target	ID	Date	flag	User	Text
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all...
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

1.2 2. Splitting dataset into training, valid and testing parts

[ ]:
```
x_train_valid, x_test, y_train_valid, y_test = train_test_split(
    tweets.drop(columns=['Target']), # X
    tweets['Target'], # y
    test_size=0.3, random_state=42)
```

[ ]:
```
x_train_valid.shape, y_train_valid.shape, x_test.shape, y_test.shape
```

[ ]:
```
((734002, 5), (734002,), (314573, 5), (314573,))
```

[ ]:
```
x_train, x_valid, y_train, y_valid = train_test_split(
    x_train_valid, # X
    y_train_valid, # y
    test_size=0.3, random_state=42)
```

[ ]:
```
x_train.shape, y_train.shape, x_valid.shape, y_valid.shape
```

[ ]:
```
((513801, 5), (513801,), (220201, 5), (220201,))
```

[ ]:
```
# saving to files
# x_train.to_csv("..//data//x_train.csv", index=False)
# y_train.to_csv("..//data//y_train.csv", index=False)
# x_valid.to_csv("..//data//x_valid.csv", index=False)
# y_valid.to_csv("..//data//y_valid.csv", index=False)
# x_test.to_csv("..//data//x_test.csv", index=False)
# y_test.to_csv("..//data//y_test.csv", index=False)
```

1.3 EDA

[ ]:
```
# check the shape of the dataframe
# df = x_train
# df['Target'] = y_train
df = x_valid
df['Target'] = y_valid
print("Shape of the dataframe:", df.shape)
```

```
Shape of the dataframe: (220201, 6)
```

[ ]:
```
# display the first few rows of the dataframe
df.head()
```

[ ]:

	ID	Date	flag	User	Text	Target
240689	1980936366	Sun May 31 08:02:09 PDT 2009	NO_QUERY	JustMaddie	Tierd and it's school tomorrow Last week atle...	0
413003	2060489943	Sat Jun 06 19:00:12 PDT 2009	NO_QUERY	tyla_da_queen		
950284	1823968497	Sat May 16 23:35:04 PDT 2009	NO_QUERY	ileftmycookie		
672298	2247129196	Fri Jun 19 18:37:58 PDT 2009	NO_QUERY	geekonomics		
852721	1573026482	Mon Apr 20 23:26:00 PDT 2009	NO_QUERY	NovaWildstar		

```
413003  twitter gets boring n boring everyday!!!no sta...       0
950284  I'm watching Guy Ripley, right now...haha...       4
672298  @mhisham that's the way indoor stadium toilets...       0
852721   @hannahpoulton it must be all that bike riding!       4
```

[ ]:
```
# display the last few rows of the dataframe
df.tail()
```

[ ]:
```
                ID                          Date      flag            User  \
55759   1685191660  Sat May 02 23:23:47 PDT 2009  NO_QUERY      pnwfitness
175608  1964891086  Fri May 29 14:58:49 PDT 2009  NO_QUERY  Brandonnnnnnnn
661283  2243073656  Fri Jun 19 12:59:35 PDT 2009  NO_QUERY    emmalouisex3
43369   1676483427  Fri May 01 22:10:50 PDT 2009  NO_QUERY     DonniesDiva
401275  2057629187  Sat Jun 06 13:21:43 PDT 2009  NO_QUERY      lovesmiles

                                                     Text  Target
55759   @LisaKLong Wantd 2b comedian when lil boy. I m...       0
175608  Omg I can't believe jay leno is going off the ...       0
661283  @Nickjonas: i dont know! my days are all messe...       0
43369   So I am guessin @donniewahlberg meant midnight...       0
401275  shit! fuckin fever, fuckin body ..think im gon...       0
```

[ ]:
```
# display information about data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 220201 entries, 240689 to 401275
Data columns (total 6 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   ID      220201 non-null  int64
 1   Date    220201 non-null  object
 2   flag    220201 non-null  object
 3   User    220201 non-null  object
 4   Text    220201 non-null  object
 5   Target  220201 non-null  int64
dtypes: int64(2), object(4)
memory usage: 11.8+ MB
```

[ ]:
```
# check for duplication
df.nunique()
```

[ ]:
```
ID        220172
Date      194930
flag           1
User      162707
Text      218810
Target         2
dtype: int64
```

[ ]:
```
# check for missing values
df.isnull().sum()
```

[ ]:
```
ID        0
Date      0
flag      0
User      0
Text      0
Target    0
dtype: int64
```

[ ]:
```
# summary statistics of numerical columns
df.describe()
```

[ ]:
```
                 ID         Target
count  2.202010e+05  220201.000000
mean   1.975621e+09       0.949115
std    2.302542e+08       1.701662
min    1.467811e+09       0.000000
25%    1.824298e+09       0.000000
50%    1.990733e+09       0.000000
75%    2.198698e+09       0.000000
max    2.329203e+09       4.000000
```

Data reduction

Some columns or variables can be dropped if they do not add value to our analysis

In our dataset, columns ID, Date, flag, User don't have any predictive power to predict the dependent variable

[ ]:
```
data = df.drop(['ID', 'Date', 'flag', 'User'], axis = 'columns')
data
```

[ ]:
```
                                                     Text  Target
240689  Tierd and it's school tomorrow  Last week atle...       0
413003  twitter gets boring n boring everyday!!!no sta...       0
950284  I'm watching Guy Ripley, right now...haha...       4
672298  @mhisham that's the way indoor stadium toilets...       0
852721   @hannahpoulton it must be all that bike riding!       4
...                                                   ...     ...
55759   @LisaKLong Wantd 2b comedian when lil boy. I m...       0
175608  Omg I can't believe jay leno is going off the ...       0
661283  @Nickjonas: i dont know! my days are all messe...       0
43369   So I am guessin @donniewahlberg meant midnight...       0
401275  shit! fuckin fever, fuckin body ..think im gon...       0

[220201 rows x 2 columns]
```

Data cleaning

Some names of the variables are not relevant and not easy to understand

Some data may have data entry errors, and some variables may need data type conversion. We need to fix this issue in the data

[ ]:
```
# adjusting target values
data['Target'] = data['Target'].replace(4, 1)
data
```

[ ]:

	Text	Target
240689	Tierd and it's school tomorrow Last week atle...	0
413003	twitter gets boring n boring everyday!!!no sta...	0
950284	I'm watching Guy Ripley, right now...haha...	1
672298	@mhisham that's the way indoor stadium toilets...	0
852721	@hannahpoulton it must be all that bike riding!	1
...	...	...
55759	@LisaKLong Wantd 2b comedian when lil boy. I m...	0
175608	Omg I can't believe jay leno is going off the ...	0
661283	@Nickjonas: i dont know! my days are all messe...	0
43369	So I am guessin @donniewahlberg meant midnight...	0
401275	shit! fuckin fever, fuckin body ..think im gon...	0

[220201 rows x 2 columns]

[ ]:
```
# removing unnecessary user tags
data['Text'] = data['Text'].replace(r"@\w+", "", regex=True)
data
```

[ ]:

	Text	Target
240689	Tierd and it's school tomorrow Last week atle...	0
413003	twitter gets boring n boring everyday!!!no sta...	0
950284	I'm watching Guy Ripley, right now...haha...	1
672298	that's the way indoor stadium toilets are	0
852721	it must be all that bike riding!	1
...	...	...
55759	Wantd 2b comedian when lil boy. I memrize com...	0
175608	Omg I can't believe jay leno is going off the ...	0
661283	: i dont know! my days are all messed up since...	0
43369	So I am guessin meant midnight Pacific time	0
401275	shit! fuckin fever, fuckin body ..think im gon...	0

[220201 rows x 2 columns]

[ ]:
```
# resolving contractions (and slang)
data['Text'] = data['Text'].apply(lambda x: contractions.fix(x))
data
```

[ ]:

	Text	Target
240689	Tierd and it is school tomorrow Last week atl...	0
413003	twitter gets boring n boring everyday!!!no sta...	0
950284	I am watching Guy Ripley, right now...haha...	1
672298	that is the way indoor stadium toilets are	0
852721	it must be all that bike riding!	1
...	...	...

```
55759    Wantd 2b comedian when lil boy. I memrize com...       0
175608  Omg I cannot believe jay leno is going off the...       0
661283  : i do not know! my days are all messed up sin...       0
43369       So I am guessin  meant midnight Pacific time        0
401275  shit! fuckin fever, fuckin body ..think i am g...       0

[220201 rows x 2 columns]
```

[ ]:
```
# removing punctuation marks
data['Text'] = data['Text'].apply(lambda x: re.sub(r'[^\w\s]', '', x))
data
```

[ ]:
```
                                                     Text  Target
240689  Tierd and it is school tomorrow  Last week atl...       0
413003  twitter gets boring n boring everydayno star w...       0
950284   I am watching Guy Ripley right nowhahahilarious        1
672298        that is the way indoor stadium toilets are        0
852721                   it must be all that bike riding        1
...                                                   ...     ...
55759    Wantd 2b comedian when lil boy I memrize comm...       0
175608  Omg I cannot believe jay leno is going off the...       0
661283   i do not know my days are all messed up since...       0
43369       So I am guessin  meant midnight Pacific time        0
401275  shit fuckin fever fuckin body think i am going...       0

[220201 rows x 2 columns]
```

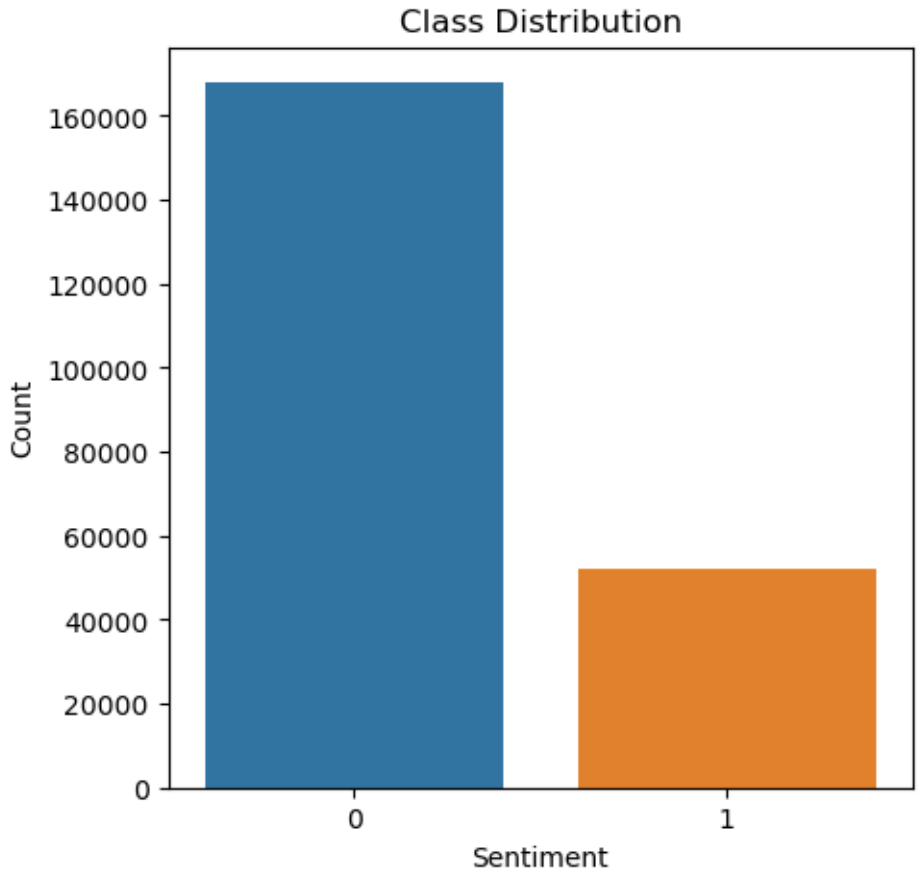
[ ]:
```
# lowercasing letters in the text
data['Text'] = data['Text'].str.lower()
data
```

[ ]:
```
                                                     Text  Target
240689  tierd and it is school tomorrow  last week atl...       0
413003  twitter gets boring n boring everydayno star w...       0
950284   i am watching guy ripley right nowhahahilarious        1
672298        that is the way indoor stadium toilets are        0
852721                   it must be all that bike riding        1
...                                                   ...     ...
55759    wantd 2b comedian when lil boy i memrize comm...       0
175608  omg i cannot believe jay leno is going off the...       0
661283   i do not know my days are all messed up since...       0
43369       so i am guessin  meant midnight pacific time        0
401275  shit fuckin fever fuckin body think i am going...       0

[220201 rows x 2 columns]
```

Visualization

[ ]:
```
# visualize class distribution
plt.figure(figsize=(5, 5))
```

```
sns.countplot(x = 'Target' , data = data)
plt.title('Class Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```

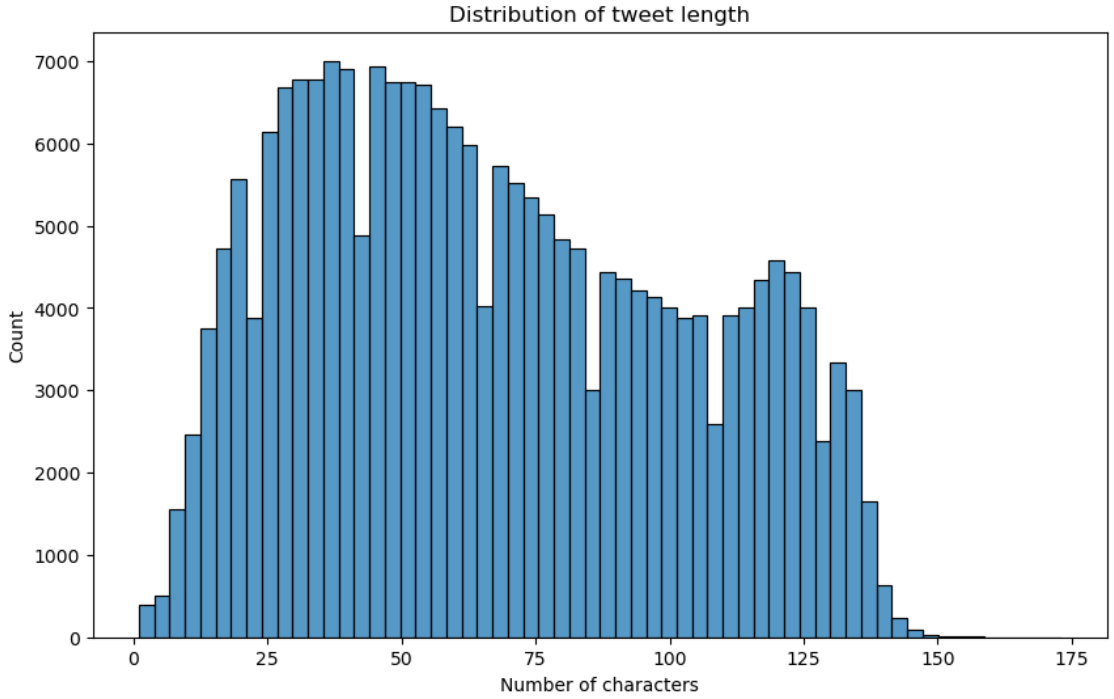


[ ]:
```
# checking the percentage of target 1
target_counts = data['Target'].value_counts()
percentage_target_1 = (target_counts[1] / target_counts.sum()) * 100
percentage_target_1
```

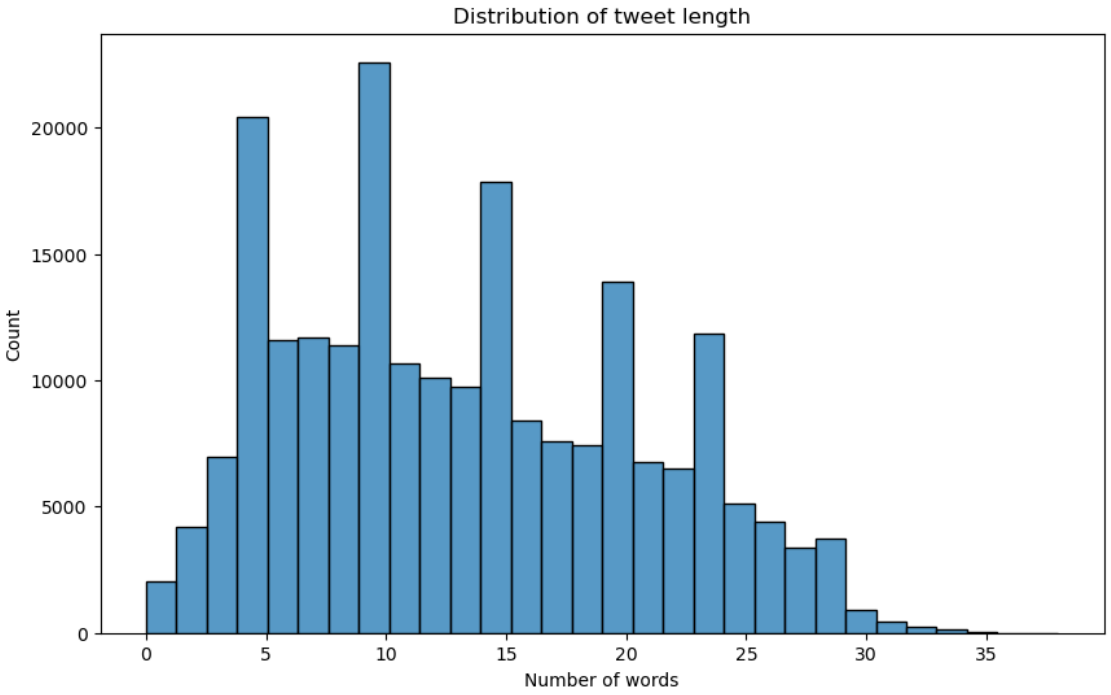
[ ]: 23.72786681259395

[ ]:
```
# explore tweet length
data['characters'] = data['Text'].apply(lambda x: len(x))

# visualize tweet length distribution
plt.figure(figsize = (10, 6))
sns.histplot(data['characters'], bins = 60)
plt.title('Distribution of tweet length')
plt.xlabel('Number of characters')
```

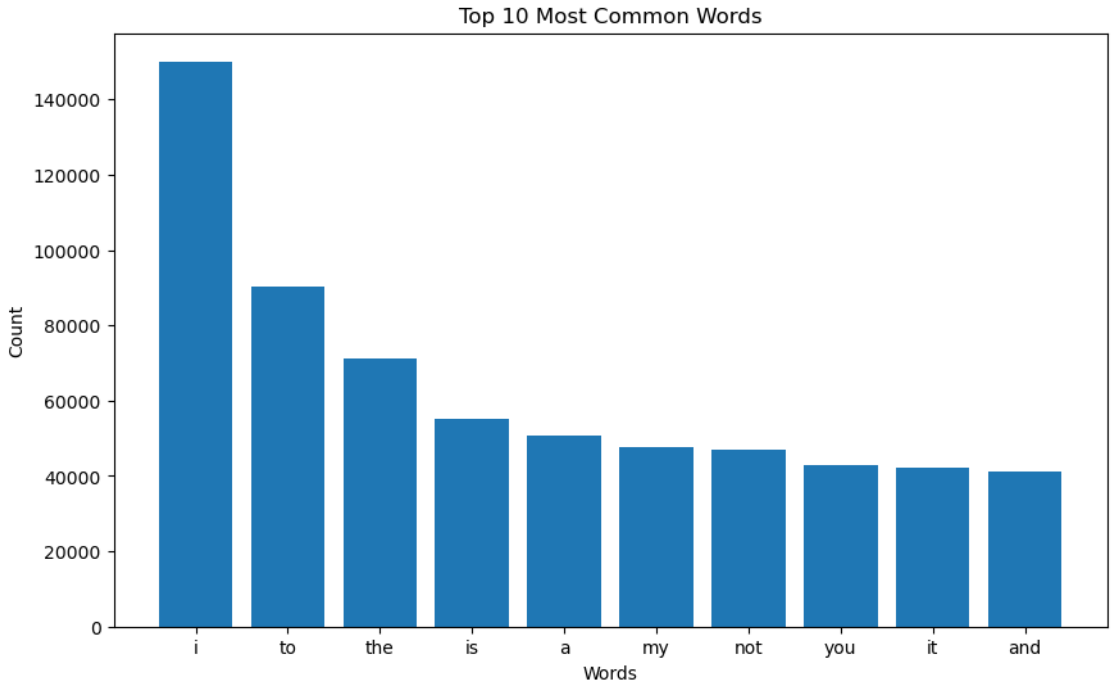
```
plt.ylabel('Count')
plt.show()
```



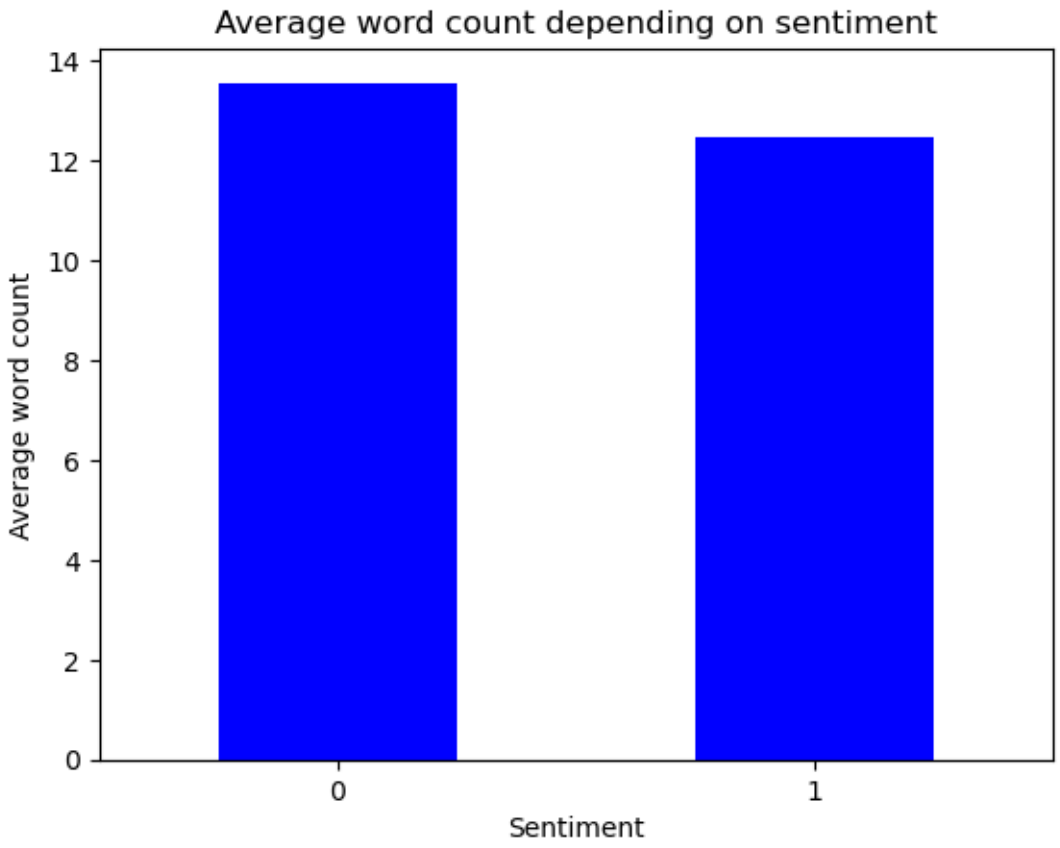
[ ]:
```
# explore tweet length
data['words'] = data['Text'].apply(lambda x: len(x.split()))

# visualize tweet length distribution
plt.figure(figsize = (10, 6))
sns.histplot(data['words'], bins = 30)
plt.title('Distribution of tweet length')
plt.xlabel('Number of words')
plt.ylabel('Count')
plt.show()
```



[ ]:
```
# combine all the text into a single string
all_text = ' '.join(data['Text'])

# split the text into individual words
words = all_text.split()

# count the frequency of each word
word_counts = Counter(words)

# get the top 10 most common words
top_10_words = word_counts.most_common(10)

# extract the words and their counts
top_10_words, top_10_counts = zip(*top_10_words)

# plot the bar chart
plt.figure(figsize=(10, 6))
plt.bar(top_10_words, top_10_counts)
plt.title('Top 10 Most Common Words')
plt.xlabel('Words')
plt.ylabel('Count')
plt.show()
```



[ ]:
```
# average word count depending on sentiment

d = data.groupby("Target").words.agg("mean")

d.plot(kind = 'bar', color = 'blue')

plt.title('Average word count depending on sentiment')
plt.xlabel('Sentiment')
plt.ylabel('Average word count')
plt.xticks(rotation = 0)
plt.show()
```



[ ]:
```
# is # in tweet?

data['has_hashtag'] = tweets['Text'].str.contains(r'#\w+')
data
```

[ ]:
```
                                                     Text  Target  characters  \
240689  tierd and it is school tomorrow  last week atl...       0          51
413003  twitter gets boring n boring everydayno star w...       0          84
950284   i am watching guy ripley right nowhahahilarious        1          48
672298        that is the way indoor stadium toilets are        0          44
852721                    it must be all that bike riding       1          33
...                                                     ...     ...         ...
55759    wantd 2b comedian when lil boy i memrize comm...       0         126
175608  omg i cannot believe jay leno is going off the...       0          51
661283   i do not know my days are all messed up since...       0          94
43369       so i am guessin  meant midnight pacific time        0          45
401275  shit fuckin fever fuckin body think i am going...       0          92

        words  has_hashtag
240689      9        False
413003     12        False
```

```
950284      7        False
672298      8        False
852721      7        False
...       ...          ...
55759      20        False
175608     11        False
661283     21        False
43369       8        False
401275     18        False

[220201 rows x 5 columns]
```

[ ]:
```
# is hashtag present in negatives tweets?

data[data['Target'] == 0]['has_hashtag'].value_counts().apply(lambda x: x /␣
 ↪len(data[data['Target'] == 0]) * 100)
```

[ ]:
```
False    98.136968
True      1.863032
Name: has_hashtag, dtype: float64
```

[ ]:
```
# is hashtag present in positives tweets?

data[data['Target'] == 1]['has_hashtag'].value_counts().apply(lambda x: x /␣
 ↪len(data[data['Target'] == 1]) * 100)
```

[ ]:
```
False    97.536795
True      2.463205
Name: has_hashtag, dtype: float64
```

[ ]:
```
# is "not" in tweet?

data['has_not'] = data['Text'].str.contains('not')
data
```

[ ]:
```
                                                     Text  Target  characters  \
240689  tierd and it is school tomorrow  last week atl...       0          51
413003  twitter gets boring n boring everydayno star w...       0          84
950284   i am watching guy ripley right nowhahahilarious        1          48
672298        that is the way indoor stadium toilets are       0          44
852721                    it must be all that bike riding       1          33
...                                                  ...     ...         ...
55759    wantd 2b comedian when lil boy i memrize comm...       0         126
175608  omg i cannot believe jay leno is going off the...       0          51
661283   i do not know my days are all messed up since...       0          94
43369       so i am guessin  meant midnight pacific time        0          45
401275  shit fuckin fever fuckin body think i am going...       0          92

        words  has_hashtag  has_not
240689      9        False    False
```

```
413003     12        False    False
950284      7        False    False
672298      8        False    False
852721      7        False    False
...       ...          ...      ...
55759      20        False     True
175608     11        False     True
661283     21        False     True
43369       8        False    False
401275     18        False    False

[220201 rows x 6 columns]
```

[ ]:
```
# is "not" present in negatives tweets?

data[data['Target'] == 0]['has_not'].value_counts().apply(lambda x: x /␣
↪len(data[data['Target'] == 0]) * 100)
```

[ ]:
```
False    70.515385
True     29.484615
Name: has_not, dtype: float64
```

[ ]:
```
# is "not" present in positives tweets?

data[data['Target'] == 1]['has_not'].value_counts().apply(lambda x: x /␣
↪len(data[data['Target'] == 1]) * 100)
```

[ ]:
```
False    86.196865
True     13.803135
Name: has_not, dtype: float64
```

[ ]:
```
# extract hour from the Date column

data['Hour'] = pd.to_datetime(tweets['Date']).dt.hour
data
```

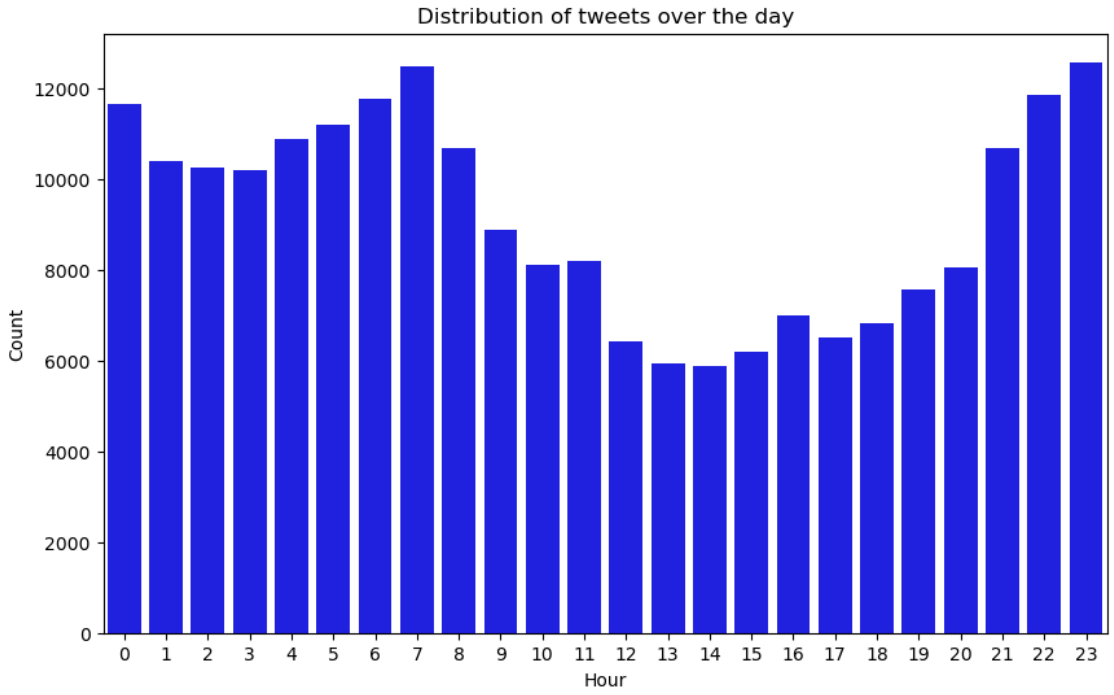
```
c:\Users\flang\anaconda3\lib\site-packages\dateutil\parser\_parser.py:1207:
UnknownTimezoneWarning: tzname PDT identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  warnings.warn("tzname {tzname} identified but not understood.  "


      ␣
↪---------------------------------------------------------------------------

       KeyboardInterrupt                         Traceback (most recent call␣
↪last)

       Cell In[36], line 3
```

```
      1 # extract hour from the Date column
----> 3 data['Hour'] = pd.to_datetime(tweets['Date']).dt.hour
      4 data


File c:
↪\Users\flang\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1051,␣
↪in to_datetime(arg, errors, dayfirst, yearfirst, utc, format, exact, unit,␣
↪infer_datetime_format, origin, cache)
   1049         result = arg.map(cache_array)
   1050     else:
-> 1051         values = convert_listlike(arg._values, format)
   1052         result = arg._constructor(values, index=arg.index, name=arg.
↪name)
   1053 elif isinstance(arg, (ABCDataFrame, abc.MutableMapping)):


File c:
↪\Users\flang\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:402,␣
↪in _convert_listlike_datetimes(arg, format, name, tz, unit, errors,␣
↪infer_datetime_format, dayfirst, yearfirst, exact)
    400 assert format is None or infer_datetime_format
    401 utc = tz == "utc"
--> 402 result, tz_parsed = objects_to_datetime64ns(
    403     arg,
    404     dayfirst=dayfirst,
    405     yearfirst=yearfirst,
    406     utc=utc,
    407     errors=errors,
    408     require_iso8601=require_iso8601,
    409     allow_object=True,
    410 )
    412 if tz_parsed is not None:
    413     # We can take a shortcut since the datetime64 numpy array
    414     # is in UTC
    415     dta = DatetimeArray(result, dtype=tz_to_dtype(tz_parsed))


File c:
↪\Users\flang\anaconda3\lib\site-packages\pandas\core\arrays\datetimes.py:2199,␣
↪in objects_to_datetime64ns(data, dayfirst, yearfirst, utc, errors,␣
↪require_iso8601, allow_object, allow_mixed)
   2197 order: Literal["F", "C"] = "F" if flags.f_contiguous else "C"
   2198 try:
-> 2199     result, tz_parsed = tslib.array_to_datetime(
   2200         data.ravel("K"),
   2201         errors=errors,
```

```
      2202         utc=utc,
      2203         dayfirst=dayfirst,
      2204         yearfirst=yearfirst,
      2205         require_iso8601=require_iso8601,
      2206         allow_mixed=allow_mixed,
      2207     )
      2208     result = result.reshape(data.shape, order=order)
      2209 except ValueError as err:


      File c:\Users\flang\anaconda3\lib\site-packages\pandas\_libs\tslib.pyx:
↪381, in pandas._libs.tslib.array_to_datetime()


      File c:\Users\flang\anaconda3\lib\site-packages\pandas\_libs\tslib.pyx:
↪536, in pandas._libs.tslib.array_to_datetime()


      File c:
↪\Users\flang\anaconda3\lib\site-packages\pandas\_libs\tslibs\parsing.pyx:281,␣
↪in pandas._libs.tslibs.parsing.parse_datetime_string()


      File c:\Users\flang\anaconda3\lib\site-packages\dateutil\parser\_parser.
↪py:1368, in parse(timestr, parserinfo, **kwargs)
      1366     return parser(parserinfo).parse(timestr, **kwargs)
      1367 else:
   -> 1368     return DEFAULTPARSER.parse(timestr, **kwargs)


      File c:\Users\flang\anaconda3\lib\site-packages\dateutil\parser\_parser.
↪py:640, in parser.parse(self, timestr, default, ignoretz, tzinfos, **kwargs)
      636 if default is None:
      637     default = datetime.datetime.now().replace(hour=0, minute=0,
      638                                               second=0,␣
↪microsecond=0)
   --> 640 res, skipped_tokens = self._parse(timestr, **kwargs)
      642 if res is None:
      643     raise ParserError("Unknown string format: %s", timestr)


      File c:\Users\flang\anaconda3\lib\site-packages\dateutil\parser\_parser.
↪py:719, in parser._parse(self, timestr, dayfirst, yearfirst, fuzzy,␣
↪fuzzy_with_tokens)
      716     yearfirst = info.yearfirst
      718 res = self._result()
   --> 719 l = _timelex.split(timestr)         # Splits the timestr into tokens
```

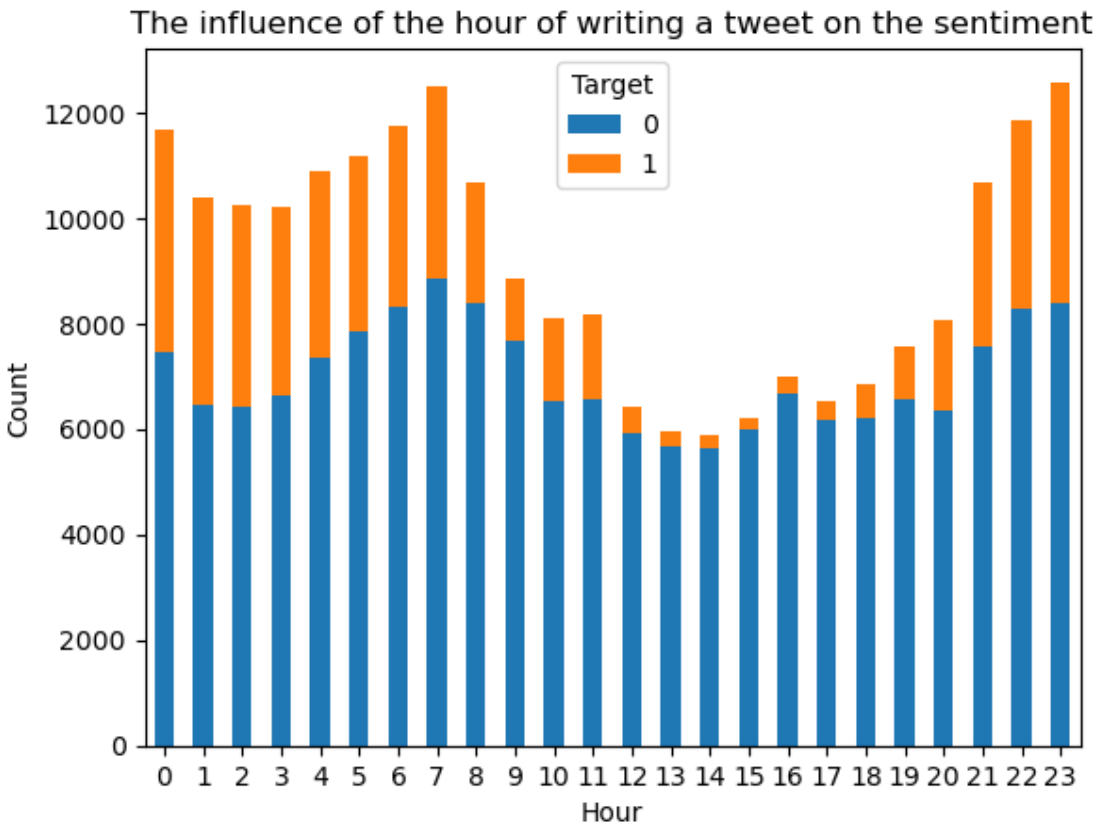
```
      721 skipped_idxs = []
      723 # year/month/day list


      File c:\Users\flang\anaconda3\lib\site-packages\dateutil\parser\_parser.
↪py:201, in _timelex.split(cls, s)
      199 @classmethod
      200 def split(cls, s):
  --> 201     return list(cls(s))


      File c:\Users\flang\anaconda3\lib\site-packages\dateutil\parser\_parser.
↪py:190, in _timelex.__next__(self)
      189 def __next__(self):
  --> 190     token = self.get_token()
      191     if token is None:
      192         raise StopIteration


      File c:\Users\flang\anaconda3\lib\site-packages\dateutil\parser\_parser.
↪py:98, in _timelex.get_token(self)
       95 token = None
       96 state = None
  ---> 98 while not self.eof:
       99     # We only realize that we've reached the end of a token when we
      100     # find a character that's not part of the current token - since
      101     # that character may be part of the next token, it's stored in␣
↪the
      102     # charstack.
      103     if self.charstack:
      104         nextchar = self.charstack.pop(0)


      KeyboardInterrupt:
```

[ ]:
```
# visualize the distribution of tweets over the day

plt.figure(figsize=(10, 6))
sns.countplot(x = 'Hour', data = data, color = 'blue')
plt.title('Distribution of tweets over the day')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.show()
```



[ ]:
```
# visualize the the influence of the hour of writing a tweet on the Target␣
 ↪variable
hourly_target_counts = data.groupby('Hour')['Target'].value_counts().
 ↪unstack(fill_value=0)
plt.figure(figsize=(15, 6))
hourly_target_counts.plot(kind='bar', stacked=True)

plt.title('The influence of the hour of writing a tweet on the sentiment')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.xticks(rotation = 0)

plt.show()
```

```
<Figure size 1500x600 with 0 Axes>
```



[ ]:
```
# deleting words which have less characters than 3

data['clean_text'] = data["Text"].apply(lambda x: " ".join([w for w in x.
 ↪split() if len(w)>=3]))
data
```

[ ]:
```
                                                     Text  Target  characters  \
240689  tierd and it is school tomorrow  last week atl...       0          51
413003  twitter gets boring n boring everydayno star w...       0          84
950284   i am watching guy ripley right nowhahahilarious        1          48
672298        that is the way indoor stadium toilets are        0          44
852721                    it must be all that bike riding       1          33
...                                                     ...     ...         ...
55759    wantd 2b comedian when lil boy i memrize comm...       0         126
175608  omg i cannot believe jay leno is going off the...       0          51
661283   i do not know my days are all messed up since...       0          94
43369       so i am guessin  meant midnight pacific time        0          45
401275  shit fuckin fever fuckin body think i am going...       0          92

        words  has_hashtag  has_not  Hour  \
240689      9        False    False     8
```

```
413003     12        False    False    19
950284      7        False    False    23
672298      8        False    False    18
852721      7        False    False    23
...       ...          ...      ...   ...
55759      20        False     True    23
175608     11        False     True    14
661283     21        False     True    12
43369       8        False    False    22
401275     18        False    False    13

                                                clean_text
240689        tierd and school tomorrow last week atleast
413003  twitter gets boring boring everydayno star wan...
950284         watching guy ripley right nowhahahilarious
672298            that the way indoor stadium toilets are
852721                             must all that bike riding
...                                                   ...
55759   wantd comedian when lil boy memrize commercial...
175608      omg cannot believe jay leno going off the air
661283  not know days are all messed since got out sch...
43369                  guessin meant midnight pacific time
401275  shit fuckin fever fuckin body think going die ...

[220201 rows x 8 columns]
```

[ ]:
```
# individual words considered as tokens

tokenized_tweet = data['clean_text'].apply(lambda x: x.split())
tokenized_tweet
```

[ ]:
```
240689    [tierd, and, school, tomorrow, last, week, atl...
413003    [twitter, gets, boring, boring, everydayno, st...
950284     [watching, guy, ripley, right, nowhahahilarious]
672298      [that, the, way, indoor, stadium, toilets, are]
852721                       [must, all, that, bike, riding]
                                  ...
55759     [wantd, comedian, when, lil, boy, memrize, com...
175608    [omg, cannot, believe, jay, leno, going, off, ...
661283    [not, know, days, are, all, messed, since, got...
43369              [guessin, meant, midnight, pacific, time]
401275    [shit, fuckin, fever, fuckin, body, think, goi...
Name: clean_text, Length: 220201, dtype: object
```

[ ]:
```
# stem the words
# stemmer = PorterStemmer()
```

```
# tokenized_tweet = tokenized_tweet.apply(lambda s: [stemmer.stem(word) for␣
 ↪word in s]) # stemming
# tokenized_tweet
# Initialize wordnet lemmatizer only on verbs - makes the biggest sense
wnl = WordNetLemmatizer()
tokenized_tweet = tokenized_tweet.apply(lambda s: [wnl.lemmatize(word, pos="v")␣
 ↪for word in s]) # lemmatization
```

[ ]:
```
tokenized_tweet.iloc[34]
```

[ ]:
```
['have',
 'just',
 'look',
 'your',
 'list',
 'and',
 'not',
 'there',
 'httpwwwdiigocomuserdaibarnesmoodlefairytab250']
```

[ ]:
```
# combining to sentences
combined_sentences = [' '.join(tokens) for tokens in tokenized_tweet]
data['combined_tweet'] = combined_sentences
data
```

[ ]:
```
                                                     Text  Target  characters  \
240689  tierd and it is school tomorrow  last week atl...       0          51
413003  twitter gets boring n boring everydayno star w...       0          84
950284   i am watching guy ripley right nowhahahilarious        1          48
672298        that is the way indoor stadium toilets are       0          44
852721                    it must be all that bike riding       1          33
...                                                     ...     ...         ...
55759    wantd 2b comedian when lil boy i memrize comm...       0         126
175608  omg i cannot believe jay leno is going off the...       0          51
661283   i do not know my days are all messed up since...       0          94
43369       so i am guessin  meant midnight pacific time        0          45
401275  shit fuckin fever fuckin body think i am going...       0          92

        words  has_hashtag  has_not  Hour  \
240689      9        False    False     8
413003     12        False    False    19
950284      7        False    False    23
672298      8        False    False    18
852721      7        False    False    23
...       ...          ...      ...   ...
55759      20        False     True    23
175608     11        False     True    14
661283     21        False     True    12
43369       8        False    False    22
```

```
401275     18        False    False    13

                                                clean_text  \
240689        tierd and school tomorrow last week atleast
413003  twitter gets boring boring everydayno star wan...
950284         watching guy ripley right nowhahahilarious
672298            that the way indoor stadium toilets are
852721                             must all that bike riding
...                                                   ...
55759   wantd comedian when lil boy memrize commercial...
175608      omg cannot believe jay leno going off the air
661283  not know days are all messed since got out sch...
43369                  guessin meant midnight pacific time
401275  shit fuckin fever fuckin body think going die ...

                                            combined_tweet
240689        tierd and school tomorrow last week atleast
413003  twitter get bore bore everydayno star want rep...
950284            watch guy ripley right nowhahahilarious
672298             that the way indoor stadium toilets be
852721                               must all that bike rid
...                                                   ...
55759   wantd comedian when lil boy memrize commercial...
175608         omg cannot believe jay leno go off the air
661283  not know days be all mess since get out school...
43369                   guessin mean midnight pacific time
401275  shit fuckin fever fuckin body think go die hea...

[220201 rows x 9 columns]
```

[ ]:
```
all_words = ' '.join([text for text in data['clean_text']])
all_words_pos = ' '.join([text for text in data['clean_text'][data['Target'] ==␣
 ↪1]])
all_words_neg = ' '.join([text for text in data['clean_text'][data['Target'] ==␣
 ↪0]])
wordcloud = WordCloud(width=800, height=500, random_state=42,␣
 ↪max_font_size=100).generate(all_words)
wordcloud_pos = WordCloud(width=800, height=500, random_state=42,␣
 ↪max_font_size=100).generate(all_words_pos)
wordcloud_neg = WordCloud(width=800, height=500, random_state=42,␣
 ↪max_font_size=100).generate(all_words_neg)

# plot the graph

fig, ax = plt.subplots(1, 3, figsize=(15, 10))
ax[0].imshow(wordcloud, interpolation="bilinear")
ax[0].set_title('All words')
```

```
ax[0].axis('off')
ax[1].imshow(wordcloud_pos, interpolation="bilinear")
ax[1].set_title('Words target 1 - Positive')
ax[1].axis('off')
ax[2].imshow(wordcloud_neg, interpolation="bilinear")
ax[2].set_title('Words target 0 - Negative')
ax[2].axis('off')
fig.show()
```

```
C:\Users\flang\AppData\Local\Temp\ipykernel_1284\208684624.py:20: UserWarning:
FigureCanvasAgg is non-interactive, and thus cannot be shown
  fig.show()
```

[ ]:
```
def hashtag_extract(tweetss):
    hashtags = []
    for tweet in tweetss:
        ht = re.findall(r"#(\w+)", tweet)
        hashtags.append(ht)
    return hashtags
```

[ ]:
```
# extracting hashtags from positive tweets
ht_positive = hashtag_extract(df['Text'][data['Target'] == 1])

# extracting hashtags from negative tweets
ht_negative = hashtag_extract(df['Text'][data['Target'] == 0])
```

[ ]:
```
# unnest list
ht_positive = sum(ht_positive, [])
ht_negative = sum(ht_negative, [])
```

[ ]:
```
ht_positive[:5]
```

[ ]:
```
['tek09', 'innovatechurch', 'yaymen', 'TwitterTakeover', 'Win7']
```

[ ]:
```
ht_negative[:5]
```

[ ]:
```
['aquarium', '1', 'dontyouhate', 'deli', 'Conwy']
```

[ ]:
```
# converting dictionary to dataframe
freq = nltk.FreqDist(ht_positive)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
```

```
                  'Count': list(freq.values())
                  })
d.sort_values(by='Count', ascending=False)
```

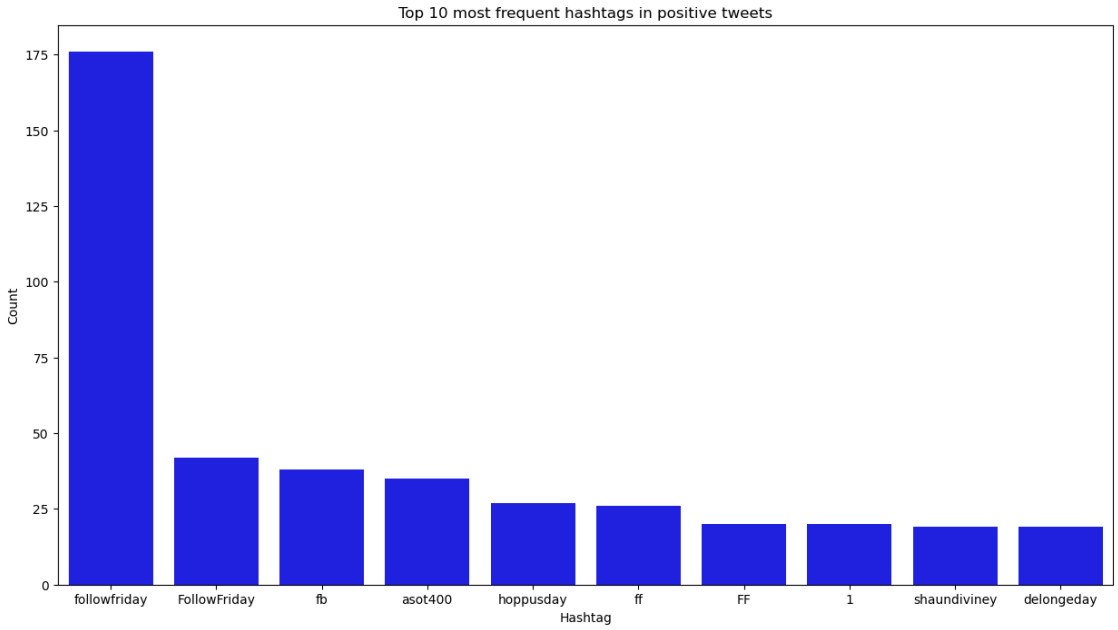
[ ]:

	Hashtag	Count
13	followfriday	176
38	FollowFriday	42
7	fb	38
27	asot400	35
19	hoppusday	27
..	...	...
340	terminator	1
341	sarahconnor	1
342	tscc	1
344	dbnerd	1
840	fuckyoufriday	1

[841 rows x 2 columns]

[ ]:
```
# selecting top 10 most frequent hashtags positive
d = d.nlargest(columns="Count", n = 10)
plt.figure(figsize=(15,8))
sns.barplot(data=d, x= "Hashtag", y = "Count", color="blue")
plt.title('Top 10 most frequent hashtags in positive tweets')
```

[ ]:
```
Text(0.5, 1.0, 'Top 10 most frequent hashtags in positive tweets')
```



[ ]:
```
# converting dictionary to dataframe
freq = nltk.FreqDist(ht_negative)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                  'Count': list(freq.values())
                  })
d.sort_values(by='Count', ascending=False)
```

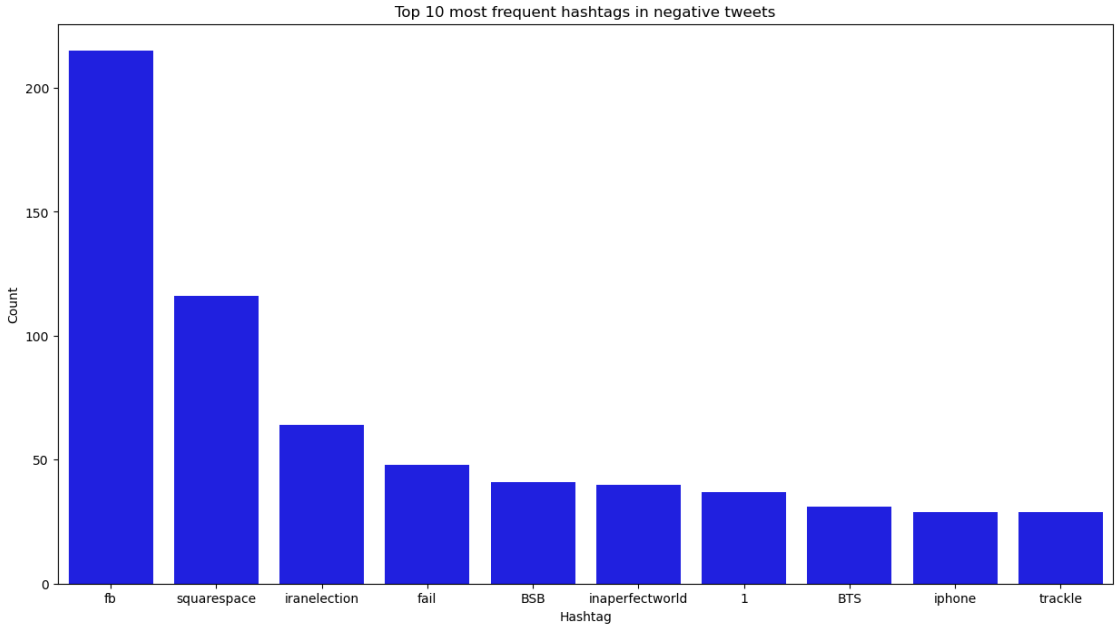
[ ]:

	Hashtag	Count
13	fb	215
6	squarespace	116
28	iranelection	64
53	fail	48
178	BSB	41
...	...	...
812	bcp3	1
811	GetWellSoonJB	1
810	macbooks	1
809	imacs	1
2003	bottomless	1

[2004 rows x 2 columns]

[ ]:
```
# selecting top 10 most frequent hashtags negative
d = d.nlargest(columns="Count", n = 10)
plt.figure(figsize=(15,8))
sns.barplot(data=d, x= "Hashtag", y = "Count", color="blue")
plt.title('Top 10 most frequent hashtags in negative tweets')
```

[ ]:
```
Text(0.5, 1.0, 'Top 10 most frequent hashtags in negative tweets')
```



1.4 2. Feature engineering

[ ]:
```
# import libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split


# text processing libraries
import re
import contractions

from collections import Counter
# import string
import nltk
# import warnings
# %matplotlib inline
# warnings.filterwarnings("ignore")
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud

from nltk.stem import WordNetLemmatizer
nltk.download("wordnet")
nltk.download("omw-1.4")

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, accuracy_score, confusion_matrix,␣
 ↪roc_auc_score
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\flang\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\flang\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

[ ]:
```
# reading splited data
x_train = pd.read_csv("..//data//x_train.csv", encoding="latin-1")
y_train = pd.read_csv("..//data//y_train.csv", encoding="latin-1")
x_test = pd.read_csv("..//data//x_test.csv", encoding="latin-1")
y_test = pd.read_csv("..//data//y_test.csv", encoding="latin-1")
```

```
x_valid = pd.read_csv("..//data//x_valid.csv", encoding="latin-1")
y_valid = pd.read_csv("..//data//y_valid.csv", encoding="latin-1")
```

[ ]:
```
# for building team
df_x = x_valid
df_y = y_valid
```

[ ]:
```
# for validation team
# df_x = x_test
# df_y = y_test
```

[ ]:
```
def clear_data(x):
    # removing unnecessary columns
    data_frame = x.drop(['ID', 'Date', 'flag', 'User'], axis = 'columns')

    # removing unnecessary user tags
    data_frame['Text'] = data_frame['Text'].replace(r"@\w+", "", regex=True)

    # resolving contractions (and slang)
    data_frame['Text'] = data_frame['Text'].apply(lambda x: contractions.fix(x))

    # removing punctuation marks
    data_frame['Text'] = data_frame['Text'].apply(lambda x: re.sub(r'[^\w\s]',␣
 ↪'', x))

    # deleting websites
    data_frame['Text'] = data_frame['Text'].apply(lambda x: re.sub(r'http\S+',␣
 ↪'', x))

    # lowercasing letters in the text
    data_frame['Text'] = data_frame['Text'].str.lower()

    # removing words with less than 3 characters
    data_frame['Text'] = data_frame['Text'].apply(lambda x: " ".join([w for w in␣
 ↪x.split() if len(w) >= 2]))

    return data_frame
```

[ ]:
```
# preparing data for the model
x_train = clear_data(x_train)
```

[ ]:
```
# preparing data for the model validation
df_x = clear_data(df_x)
```

[ ]:
```
# lemmatization
def lemmatization(x):
    data_frame = x
    # individual words considered as tokens
    tokenized_tweet = data_frame['Text'].apply(lambda x: x.split())
```

```
    # Initialize wordnet lemmatizer
    wnl = WordNetLemmatizer()
    tokenized_tweet = tokenized_tweet.apply(lambda s: [wnl.lemmatize(word,␣
↪pos='v') for word in s])
    tokenized_tweet = tokenized_tweet.apply(lambda s: [wnl.lemmatize(word,␣
↪pos='n') for word in s])
    tokenized_tweet = tokenized_tweet.apply(lambda s: [wnl.lemmatize(word,␣
↪pos='a') for word in s])
    tokenized_tweet = tokenized_tweet.apply(lambda s: [wnl.lemmatize(word,␣
↪pos='r') for word in s])

    # combining to sentences
    combined_sentences = [' '.join(tokens) for tokens in tokenized_tweet]
    data_frame['combined_tweet'] = combined_sentences
    return data_frame
```

[ ]:
```
# lemmatization data for the model
x_train = lemmatization(x_train)
```

[ ]:
```
# lemmitization data for the model validation
df_x = lemmatization(df_x)
```

[ ]:
```
# selecting stop words to be removed
custom_stop_words = CountVectorizer(stop_words='english').get_stop_words()
custom_stop_words = set(custom_stop_words) -␣
↪{'not','alone','why','well','very','together','such','nobody','noone','nothing','myself','c
custom_stop_words = list(custom_stop_words)
custom_stop_words
```

[ ]:
```
['always',
 'who',
 'meanwhile',
 'over',
 'thereupon',
 'interest',
 'almost',
 'about',
 'etc',
 'go',
 'more',
 'never',
 'down',
 'thus',
 'through',
 'whole',
 'enough',
 'none',
```

```
'ourselves',
'themselves',
'last',
'whose',
'be',
'three',
'one',
'and',
'this',
'detail',
'amount',
'back',
'others',
'anyhow',
'wherein',
'sometime',
'everyone',
'during',
'again',
'although',
'to',
'they',
'thru',
'it',
'else',
'afterwards',
'thence',
'only',
'besides',
'up',
'everything',
'there',
'twenty',
'sincere',
'ltd',
'something',
'often',
'same',
'behind',
'ie',
'eleven',
'own',
'first',
'might',
'neither',
'take',
'anything',
```

```
'latterly',
'into',
'where',
'at',
'third',
'much',
'so',
'is',
'above',
'becomes',
'while',
'perhaps',
'two',
'we',
'us',
'system',
'however',
'though',
'please',
'its',
'these',
'within',
'whereupon',
'sometimes',
'his',
'our',
'twelve',
'anyone',
'her',
'upon',
'an',
'thereby',
'seems',
'when',
'either',
'how',
'than',
'without',
'thick',
'will',
'done',
'found',
'whoever',
'being',
'him',
'seemed',
'ten',
```

```
'whither',
'the',
'nor',
'she',
'several',
'hereby',
'am',
'find',
'per',
'amongst',
'eg',
'move',
'whatever',
'indeed',
'whence',
'describe',
'no',
'otherwise',
'everywhere',
'former',
'under',
'what',
'therefore',
'front',
'become',
'somewhere',
'further',
'he',
'towards',
'herein',
'all',
'namely',
'elsewhere',
'empty',
'off',
'since',
'moreover',
'because',
'of',
'along',
'then',
'somehow',
'fill',
'them',
'other',
'due',
'side',
```

```
'been',
'for',
'by',
'another',
'himself',
'keep',
'forty',
'around',
'ours',
'wherever',
'any',
'next',
'out',
'via',
'throughout',
'nowhere',
'most',
'few',
'rather',
'herself',
'hereupon',
'whereas',
'must',
'six',
'hence',
'against',
'con',
'whether',
'on',
'whereby',
'should',
'latter',
'each',
'thereafter',
'from',
'onto',
'many',
'would',
'co',
'bill',
'formerly',
'serious',
'now',
'fifteen',
'un',
'itself',
'seeming',
```

```
'may',
'among',
'toward',
'five',
'mine',
'have',
'you',
'yet',
'their',
'get',
'beside',
'but',
'seem',
'once',
'sixty',
'except',
'until',
'full',
'put',
'give',
'whereafter',
'even',
'do',
'call',
'made',
'hasnt',
'if',
'mill',
'too',
'a',
'nine',
'me',
'becoming',
'fifty',
'whom',
'amoungst',
'de',
'yourself',
'hers',
'still',
'yourselves',
'inc',
'thin',
'was',
'became',
're',
'were',
```

```
'mostly',
'therein',
'name',
'beyond',
'whenever',
'also',
'are',
'my',
'your',
'that',
'show',
'fire',
'has',
'which',
'nevertheless',
'i',
'yours',
'anyway',
'anywhere',
'least',
'with',
'hundred',
'four',
'top',
'here',
'less',
'every',
'bottom',
'ever',
'hereafter',
'part',
'in',
'both',
'after',
'some',
'beforehand',
'or',
'across',
'those',
'as',
'already',
'below',
'between',
'someone',
'see',
'before',
'had',
```

```
'eight']
```

1.4.1 Bag of words model

[ ]:
```
# bag of words conditions and vectorization
bow_vectorizer = CountVectorizer(max_df = 0.95, min_df = 5, max_features =␣
↪5000, stop_words=custom_stop_words)
bow = bow_vectorizer.fit_transform(x_train['combined_tweet'])
```

[ ]:
```
# training the model
model = LogisticRegression(max_iter=5000)
model.fit(bow, y_train)
```

```
c:\Users\flang\anaconda3\lib\site-packages\sklearn\utils\validation.py:1184:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[ ]:
```
LogisticRegression(max_iter=5000)
```

[ ]:
```
# vectorization of the validation data
bow = bow_vectorizer.transform(df_x['combined_tweet'])
```

[ ]:
```
# testing the model
pred = model.predict(bow)
```

[ ]:
```
# metrics
f1_score(df_y, pred, pos_label=4)
```

[ ]:
```
0.5537982098505085
```

[ ]:
```
auc = roc_auc_score(df_y, pred)
gini = 2 * auc - 1
gini
```

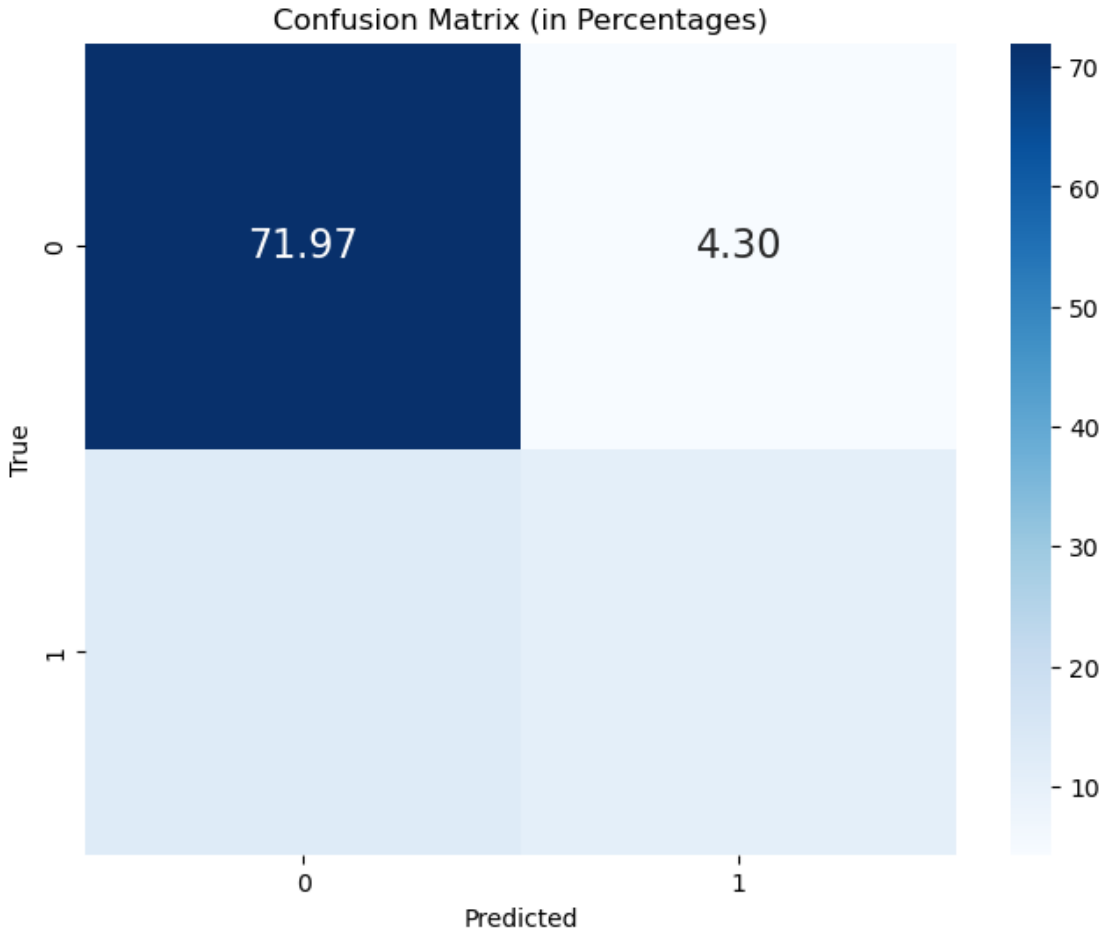
[ ]:
```
0.3959560892559759
```

[ ]:
```
# accuracy
accuracy_score(df_y, pred)
```

[ ]:
```
0.8270398408726573
```

[ ]:
```
# plotting the confusion matrix
cm = confusion_matrix(df_y, pred)

# Calculate the total number of samples
total_samples = np.sum(cm)

# Convert the values in the confusion matrix to percentages
cm_percent = (cm / total_samples) * 100
```

```
# Plotting the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_percent, annot=True, fmt='.2f', cmap='Blues', annot_kws={"size":␣
↪16})
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix (in Percentages)')
plt.show()
```



1.4.2 Tensorflow model

[ ]:
```
# read the CSV file
x_train = pd.read_csv('../data/x_train.csv')
x_valid = pd.read_csv('../data/x_valid.csv')
y_train = pd.read_csv('../data/y_train.csv')
y_valid = pd.read_csv('../data/y_valid.csv')
x_test = pd.read_csv("..//data//x_test.csv")
```

```
y_test = pd.read_csv("..//data//y_test.csv")
```

[ ]:
```
# for building team
df_x = x_valid
df_y = y_valid
```

[ ]:
```
# for validation team
# df_x = x_test
# df_y = y_test
```

[ ]:
```
# replacing 4 with 1 in the target column to make it binary
y_train['Target'] = y_train['Target'].replace(4, 1)
df_y['Target'] = df_y['Target'].replace(4, 1)
```

[ ]:
```
# making training and testing sentences
training_sentences = x_train['Text'].tolist()
testing_sentences = df_x['Text'].tolist()
```

[ ]:
```
# making training and testing labels
training_labels = y_train['Target'].tolist()
testing_labels = df_y['Target'].tolist()
```

[ ]:
```
# some necessary variables
vocab_size = 10000
oov_tok = "<OOV>"
max_length = 80
embedding_dim = 16
```

[ ]:
```
# changing the sentences into sequences
tokenizer = Tokenizer(num_words=vocab_size,
                      oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)

word_index = tokenizer.word_index

training_sequences = tokenizer.texts_to_sequences(training_sentences)
training_padded = pad_sequences(training_sequences,
                                maxlen = max_length,
                                padding = 'post',
                                truncating = 'post')

testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sequences,
                                maxlen = max_length,
                                padding = 'post',
                                truncating = 'post')
```

[ ]:
```
# changing the lists into arrays for the model
training_padded = np.array(training_padded)
training_labels = np.array(training_labels)
testing_padded = np.array(testing_padded)
```

```
testing_labels = np.array(testing_labels)
```

[ ]:
```
# creating the model
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
```

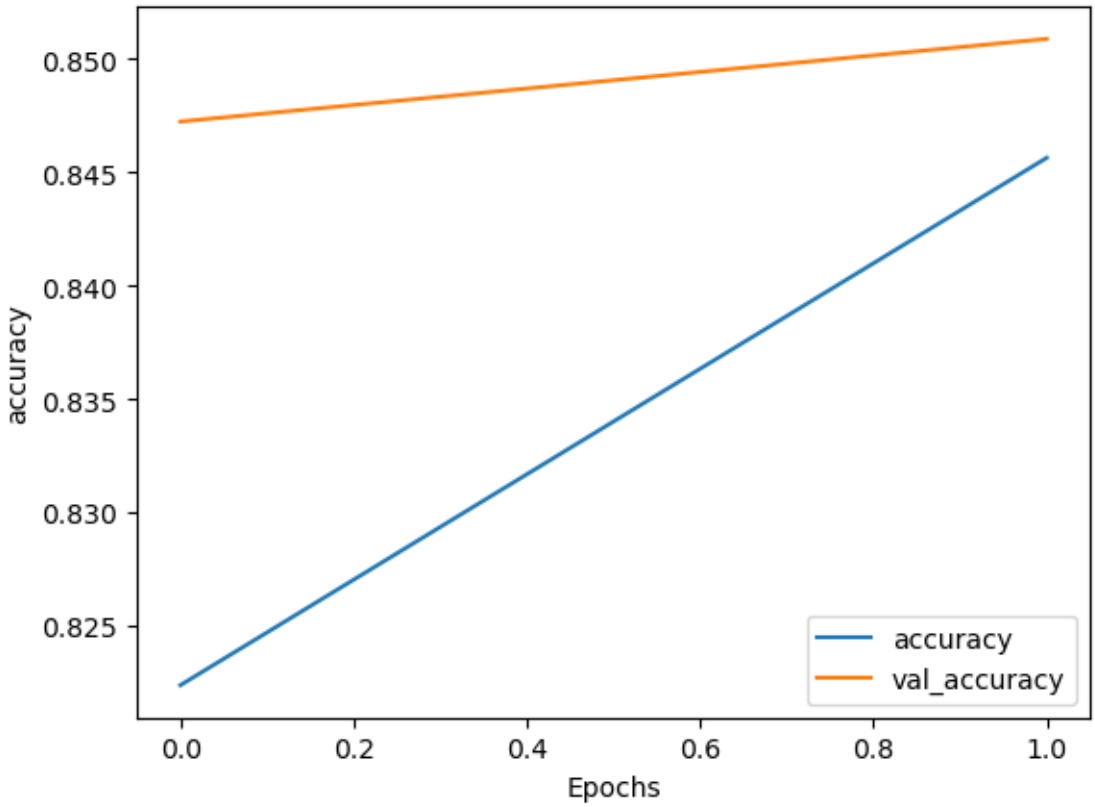
[ ]:
```
# model.summary()
```

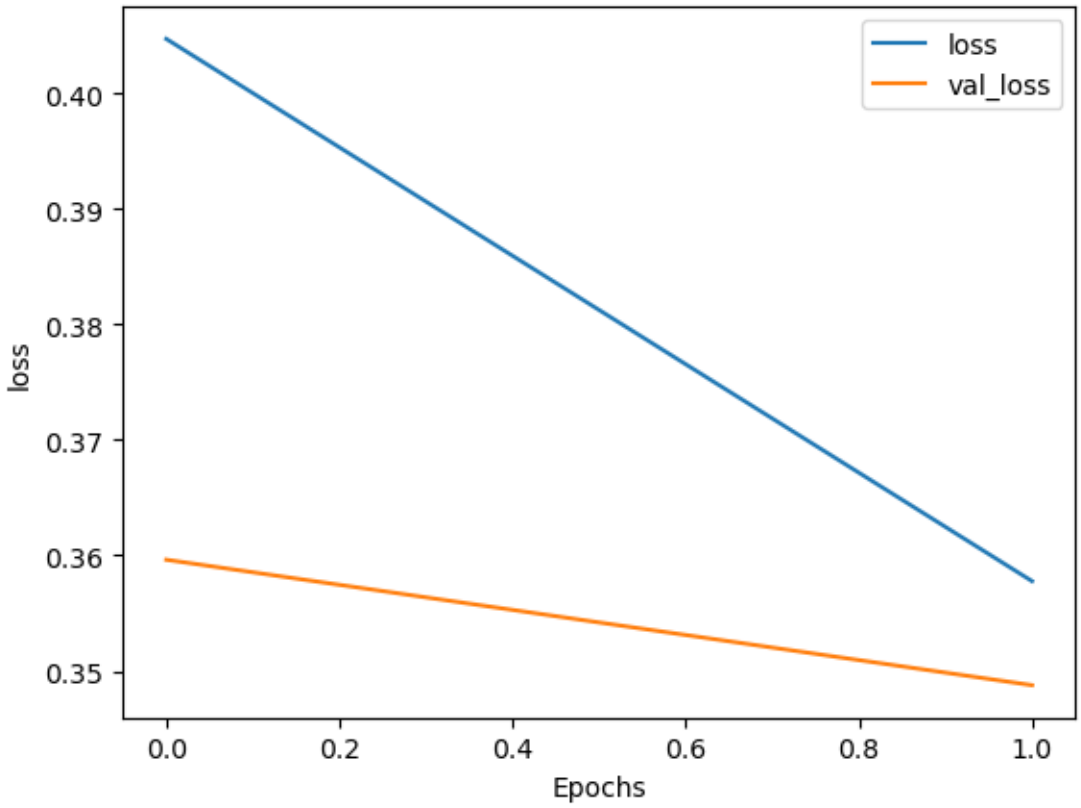
[ ]:
```
# number of epochs to train the model
num_epochs = 2
```

[ ]:
```
# training and testing the model
history = model.fit(training_padded,
                    training_labels,
                    epochs=num_epochs,
                    validation_data=(testing_padded,
                                     testing_labels),
                    verbose=2)
```

```
Epoch 1/2
16057/16057 - 59s - 4ms/step - accuracy: 0.8224 - loss: 0.4047 - val_accuracy:
0.8472 - val_loss: 0.3596
Epoch 2/2
16057/16057 - 55s - 3ms/step - accuracy: 0.8456 - loss: 0.3578 - val_accuracy:
0.8509 - val_loss: 0.3488
```

[ ]:
```
# plotting the accuracy and loss
def plot_graphs(history, string):
  plt.plot(history.history[string])
  plt.plot(history.history['val_'+string])
  plt.xlabel("Epochs")
  plt.ylabel(string)
  plt.legend([string, 'val_'+string])
  plt.show()

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
```


0.850
0.845
0.840
0.835
0.830
0.825
accuracy
0.0
0.2
0.4
0.6
0.8
1.0
Epochs
accuracy
val_accuracy



[ ]:
```
pred = model.predict(testing_padded)
```

```
6882/6882  12s 2ms/step
```

[ ]:
```
auc = roc_auc_score(testing_labels, pred)
gini = 2 * auc - 1
gini
```

[ ]: 0.7626808211743885

[ ]:

[ ]:

[ ]:

[ ]: