```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")


import nltk
import re
import re,string,unicodedata
from nltk.corpus import stopwords

from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential

df = pd.read_json("C:\\Users\\user\\Desktop\\
Sarcasm_Headlines_Dataset.json",lines=True)
df.head()
```

```
                                          article_link  \
0  https://www.huffingtonpost.com/entry/versace-b...
1  https://www.huffingtonpost.com/entry/roseanne-...
2  https://local.theonion.com/mom-starting-to-fea...
3  https://politics.theonion.com/boehner-just-wan...
4  https://www.huffingtonpost.com/entry/jk-rowlin...

                                            headline  is_sarcastic
0  former versace store clerk sues over secret 'b...             0
1  the 'roseanne' revival catches up to our thorn...             0
2  mom starting to fear son's web series closest ...             1
3  boehner just wants wife to listen, not come up...             1
4  j.k. rowling wishes snape happy birthday in th...             0
```

```
df.info
```

```
<bound method DataFrame.info of
article_link  \
0      https://www.huffingtonpost.com/entry/versace-b...
1      https://www.huffingtonpost.com/entry/roseanne-...
2      https://local.theonion.com/mom-starting-to-fea...
3      https://politics.theonion.com/boehner-just-wan...
4      https://www.huffingtonpost.com/entry/jk-rowlin...
...                                                  ...
26704  https://www.huffingtonpost.com/entry/american-...
26705  https://www.huffingtonpost.com/entry/americas-...
26706  https://www.huffingtonpost.com/entry/reparatio...
26707  https://www.huffingtonpost.com/entry/israeli-b...
```

```
26708  https://www.huffingtonpost.com/entry/gourmet-g...

                                                 headline  is_sarcastic
0      former versace store clerk sues over secret 'b...             0
1      the 'roseanne' revival catches up to our thorn...             0
2      mom starting to fear son's web series closest ...             1
3      boehner just wants wife to listen, not come up...             1
4      j.k. rowling wishes snape happy birthday in th...             0
...                                                  ...           ...
26704               american politics in moral free-fall             0
26705                            america's best 20 hikes             0
26706                              reparations and obama             0
26707  israeli ban targeting boycott supporters raise...             0
26708                  gourmet gifts for the foodie 2014             0

[26709 rows x 3 columns]>
```

```
df.shape
```

```
(26709, 3)
```

```
df.isnull().sum()
```

```
article_link    0
headline        0
is_sarcastic    0
dtype: int64
```

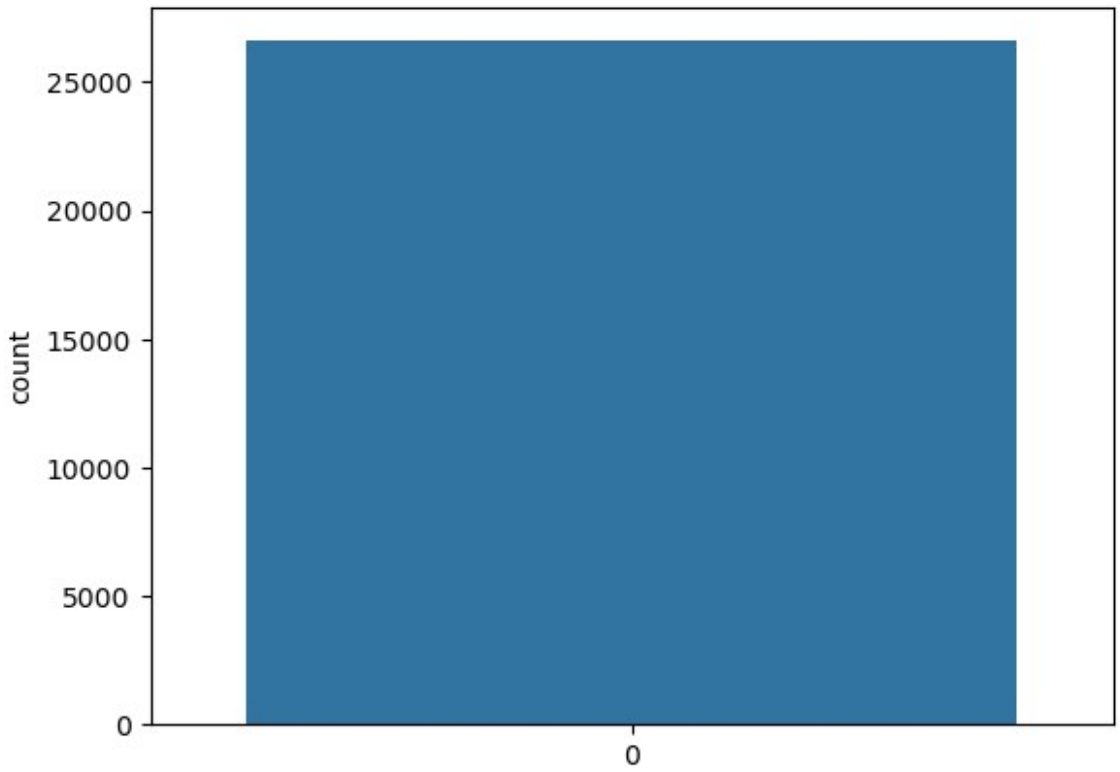
```
df.describe(include='object')
```

```
                                             article_link
headline
count                                               26709
26709
unique                                              26708
26602
top     https://www.huffingtonpost.comhttp://nymag.com...  sunday
roundup
freq                                                    2
10
```

```
# checking for duplicate values

df['headline'].duplicated().sum()
```

```
107
```

```
df=df.drop(df[df['headline'].duplicated()].index,axis=0)

sns.countplot(df['is_sarcastic']);
```




```
import nltk
nltk.download('stopwords')
stop = set(stopwords.words('english'))
punctuation = list(string.punctuation)
stop.update(punctuation)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

```
# Removing the stopwords from the text

def split_into_words(text):
    # split into words by white space
    words = text.split()
    return words
```

```
def to_lower_case(words):
    # convert the words to the lower case
    words = [word.lower() for word in words]
    return words

def remove_punctuation(words):

    # prepare regex for char filtering

    re_punc = re.compile('[%s]' % re.escape(string.punctuation))

    # remove punctuation from each word

    stripped = [re_punc.sub('', w) for w in words]
    return stripped

def keep_alphabetic(words):
    # remove remaining tokens that are not alphabetic
    words = [word for word in words if word.isalpha()]
    return words

def remove_stopwords(words):
    # filter out stop words
    stop_words = set(stopwords.words('english'))
    words = [w for w in words if not w in stop_words]
    return words

def to_sentence(words):
    # join words to a sentence
    return ' '.join(words)

# Removing the noisy text

def denoise_text(text):
    words = split_into_words(text)
    words = to_lower_case(words)
    words = remove_punctuation(words)
    words = keep_alphabetic(words)
    words = remove_stopwords(words)
    return to_sentence(words)

# Apply function on review column
df['headline']=df['headline'].apply(denoise_text)

labels = (df['is_sarcastic'])
data = (df['headline'])

train_ratio = 0.80

train_size = int(len(labels)*train_ratio)
```

```
train_data = data[:train_size]
train_labels= labels[:train_size]

test_data = data[train_size:]
test_labels = labels[train_size:]

# Get the vocabulary size

num_words=len(tokenizer.word_index)
print (num_words)

train_sequences = tokenizer.texts_to_sequences(train_data)
test_sequences = tokenizer.texts_to_sequences(test_data)

24522

maxlen=max([len(i) for i in train_sequences])

train_padded = pad_sequences(train_sequences, maxlen=maxlen,
padding='post')
test_padded = pad_sequences(test_sequences, maxlen=maxlen,
padding='post')


# Printing  a sample headline

index = 10

print(f'sample headline: {train_sequences[index]}')
print(f'padded sequence: {train_padded[index]} \n')

print(f'Original Sentence:  \n
{tokenizer.sequences_to_texts(train_sequences[index:index+1])} \n')



# Printing the  dimensions of padded sequences

print(f'shape of padded sequences: {train_padded.shape}')

sample headline: [3050, 1791, 4182, 4, 4771, 6794, 1792, 821]
padded sequence: [3050 1791 4182    4 4771 6794 1792  821    0    0
0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0]

Original Sentence:
 ['airline passengers tackle man rushes cockpit bomb threat']

shape of padded sequences: (21281, 25)

model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size+1,100,input_length=maxlen),
```

```
    tf.keras.layers.Bidirectional( tf.keras.layers.LSTM(128)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.50),
    tf.keras.layers.Dense(64,activation='relu'),
    tf.keras.layers.Dense(1,activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.summary()
```

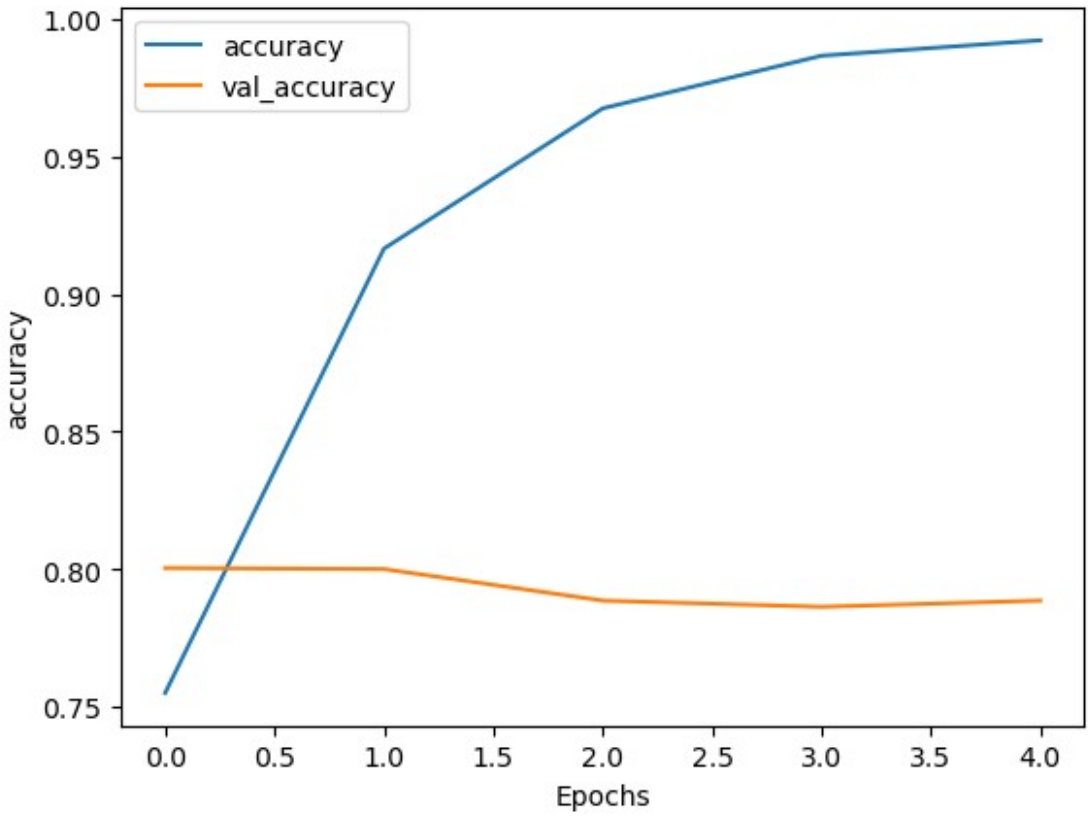
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 25, 100)           2452300

 bidirectional (Bidirection  (None, 256)               234496
 al)

 flatten (Flatten)           (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 64)                16448

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 2703309 (10.31 MB)
Trainable params: 2703309 (10.31 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

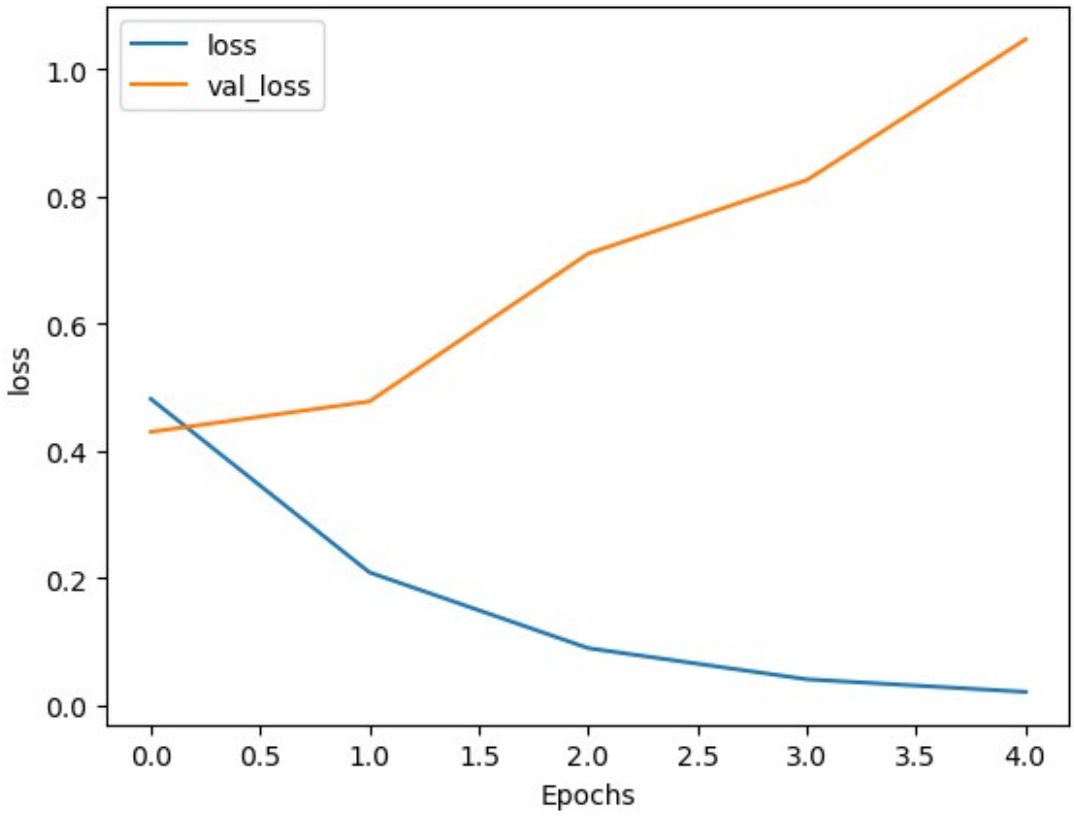
```
history=model.fit(train_padded, np.array(train_labels),validation_data
= (test_padded,np.array(test_labels)) , epochs = 5 , verbose=2)
```

```
Epoch 1/5
666/666 - 49s - loss: 0.4820 - accuracy: 0.7550 - val_loss: 0.4302 -
val_accuracy: 0.8004 - 49s/epoch - 73ms/step
Epoch 2/5
666/666 - 44s - loss: 0.2093 - accuracy: 0.9165 - val_loss: 0.4780 -
val_accuracy: 0.8000 - 44s/epoch - 66ms/step
Epoch 3/5
666/666 - 44s - loss: 0.0903 - accuracy: 0.9675 - val_loss: 0.7104 -
val_accuracy: 0.7886 - 44s/epoch - 66ms/step
Epoch 4/5
666/666 - 44s - loss: 0.0414 - accuracy: 0.9866 - val_loss: 0.8259 -
val_accuracy: 0.7863 - 44s/epoch - 66ms/step
Epoch 5/5
```

```
666/666 - 44s - loss: 0.0216 - accuracy: 0.9922 - val_loss: 1.0470 -
val_accuracy: 0.7886 - 44s/epoch - 66ms/step
```

```
# Plot utility

def plot_graphs(model, string):
  plt.plot(model.history[string])
  plt.plot(model.history['val_'+string])

  plt.xlabel("Epochs")
  plt.ylabel(string)

  plt.legend([string, 'val_'+string])

  plt.show()

# Plotting the accuracy and loss

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




loss
val_loss
1.0
0.8
0.6
0.4
0.2
0.0
loss
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs