PROJECT - (23rd April, 2021 - 09th May, 2021)

1. PART ONE

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
%tensorflow_version 2.x
import tensorflow
tensorflow.__version__
```

```
'2.4.1'
```

```
project_path = '/content/drive/MyDrive/My Files/AIML Workbooks'
```

```
import os                   # Importing os library
import pandas as pd         # To read the data set
import numpy as np          # Importing numpy library
import seaborn as sns       # For data visualization
import matplotlib.pyplot as plt      # Necessary library for plotting graphs
from glob import glob       # Importing necessary library
import tensorflow as tf     # Importing library
%matplotlib inline
sns.set(color_codes = True)

from sklearn import metrics          # Importing metrics
from sklearn.model_selection import train_test_split       # Splitting data into
from sklearn.metrics import classification_report, accuracy_score, recall_score,
from sklearn.preprocessing import StandardScaler          # Importing to standa
from sklearn.impute import SimpleImputer                  # Importing to fill i
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import PolynomialFeatures      # Importing polynomia
from sklearn.decomposition import PCA           # Importing to run pca analysis
from sklearn import svm              # Importing necessary library for model bui
from sklearn.ensemble import RandomForestClassifier       # Importing necessary
from sklearn.neighbors import KNeighborsClassifier        # Importing necessary
from sklearn import preprocessing              # Importing preprocessing librar
```

```
from sklearn.model_selection import KFold, cross_val_score       # Importing
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV    #
from sklearn.cluster import KMeans            # For KMeans cluster model build
from scipy.stats import zscore      # Import zscore library
from scipy.spatial.distance import cdist       # Importing cdist functionality
import tensorflow           # Importing tensorflow library
from tensorflow.keras.models import Sequential, Model             # Importing
from tensorflow.keras.utils import to_categorical         # Importing tensorflo
from tensorflow.keras import optimizers                   # Importing optimizer
from tensorflow.keras.layers import Dense, Dropout, Activation, BatchNormalizati
from tensorflow.keras.applications.mobilenet import preprocess_input          #
from tensorflow.python.keras.preprocessing.text import Tokenizer    # Importing
from tensorflow.python.keras.preprocessing.sequence import pad_sequences       #
from tensorflow.python.keras.models import load_model              # Importing
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLRO
from tensorflow.keras.applications.mobilenet import MobileNet       # Importing
from tensorflow.keras.losses import binary_crossentropy            # Importing
from tensorflow.keras.backend import log, epsilon         # Importing necessary
from keras.utils import np_utils     # Importing necessary library
from sklearn import svm             # Importing necessary library for model bui
from sklearn.svm import SVC         # Import svc library for model building

from skimage.color import rgb2gray             # Loading color library
from sklearn.preprocessing import OneHotEncoder           # Library for one hot
from sklearn.metrics import confusion_matrix              # Loading necessary l
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, i
from keras.preprocessing import image          # Importing necessary image libr
from tensorflow import keras        # Loading keras libaray
from tensorflow.keras.optimizers import Adam, SGD         # Importing optimizer
import cv2                  # Importing necessary library
from PIL import ImageFile           # Importing image library
from tqdm import tqdm               # Importing necessary library
import time                 # Importing time library
from mpl_toolkits.axes_grid1 import ImageGrid             # Importing necessary
from PIL import Image       # Importing image library

import re                   # Importing regular expression library
import nltk                 # Import necessary library
from nltk.corpus import stopwords              # Importing necessary library
from sklearn.feature_extraction.text import CountVectorizer          # Importing
from sklearn.preprocessing import MultiLabelBinarizer     # Imorting necessary
import json                 # Importing json to import file
import urllib               # Importing necessary library
from tensorflow.keras.preprocessing.text import Tokenizer           # Importing
from tensorflow.keras.preprocessing.sequence import pad_sequences             #
from wordcloud import WordCloud      # Importing necessary library
nltk.download("stopwords")          # Loading necessary datasets from nltk
nltk.download("punkt")              # Loading necessary datasets from nltk
nltk.download('punkt')              # Loadiing necessary datasets from nltk
```

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

Loading and checking the data

```
df = pd.read_csv('/content/drive/MyDrive/My Files/AIML Workbooks/IMDB Dataset.cs
```

```
df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

```
df.tail()
```

	review	sentiment
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

```
df.shape
```

```
(50000, 2)
```

```
df.size
```

```
100000
```

```
df.columns
```

```
Index(['review', 'sentiment'], dtype='object')
```

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

```
review       0
sentiment    0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

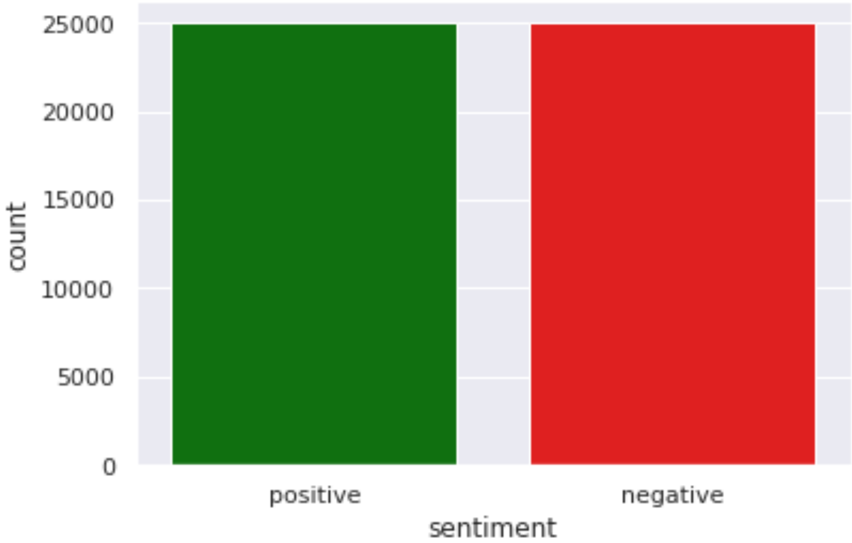
```
df.describe()
```

	review	sentiment
count	50000	50000
unique	49582	2
top	Loved today's show!!! It was a variety and not...	positive
freq	5	25000

```
# Visualizing the positive and negative sentiments

sns.countplot(df['sentiment'], palette = ['green','red'])
plt.show()
print(df.sentiment.value_counts())
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWar
  FutureWarning
```




```
positive    25000
negative    25000
Name: sentiment, dtype: int64
```

We can observe from the above graphical representation that both positive and negative sentiments are equal.

```
# Converting our sentiments into integer values

df.sentiment  = [ 1 if each == 'positive' else 0 for each in df.sentiment]
```

```
df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

```
df.tail()
```

	review	sentiment
49995	I thought this movie did a down right good job...	1
49996	Bad plot, bad dialogue, bad acting, idiotic di...	0
49997	I am a Catholic taught in parochial elementary...	0
49998	I'm going to have to disagree with the previou...	0
49999	No one expects the Star Trek movies to be high...	0

Checking to see if the sentiments have been coverted to integer values

Cleaning the dataset

Process of clearing punctuation marks in data.

Cleaning unnecessary marks in data.

Capitalization to lowercase.

Cleaning extra spaces.

Removal of stopwords in sentences.

[ ] ↳ *4 cells hidden*

Splittting data into train and test set

[ ] ↳ *12 cells hidden*

Model Buidling

```
model = Sequential()

embedding_size = 50

model.add(Embedding(input_dim = 10000, output_dim = embedding_size, input_length

model.add(LSTM(units = 16, return_sequences = True))
model.add(Dropout(0.1))

model.add(LSTM(units = 8, return_sequences= True))
model.add(Dropout(0.1))

model.add(LSTM(units = 4))
model.add(Dropout(0.1))

model.add(Dense(1, activation = 'sigmoid'))


optimizer = Adam(lr = 1e-3)
model.compile(loss = 'binary_crossentropy', metrics = ['accuracy'], optimizer =
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_layer (Embedding)  (None, 272, 50)           500000
_________________________________________________________________
lstm_3 (LSTM)                (None, 272, 16)           4288
_________________________________________________________________
dropout_3 (Dropout)          (None, 272, 16)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 272, 8)            800
_________________________________________________________________
dropout_4 (Dropout)          (None, 272, 8)            0
_________________________________________________________________
lstm_5 (LSTM)                (None, 4)                 208
_________________________________________________________________
dropout_5 (Dropout)          (None, 4)                 0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 5
=================================================================
Total params: 505,301
Trainable params: 505,301
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(x_train_pad, y_train, validation_split = 0.3, batch_size = 1
```

```
Epoch 1/10
28/28 [==============================] - 7s 144ms/step - loss: 0.6898 - acc
Epoch 2/10
28/28 [==============================] - 3s 117ms/step - loss: 0.5824 - acc
Epoch 3/10
28/28 [==============================] - 3s 115ms/step - loss: 0.4445 - acc
Epoch 4/10
28/28 [==============================] - 3s 116ms/step - loss: 0.3774 - acc
Epoch 5/10
28/28 [==============================] - 3s 117ms/step - loss: 0.3299 - acc
Epoch 6/10
28/28 [==============================] - 3s 117ms/step - loss: 0.2925 - acc
Epoch 7/10
28/28 [==============================] - 3s 116ms/step - loss: 0.2732 - acc
Epoch 8/10
28/28 [==============================] - 3s 116ms/step - loss: 0.2458 - acc
Epoch 9/10
28/28 [==============================] - 3s 117ms/step - loss: 0.2261 - acc
Epoch 10/10
28/28 [==============================] - 3s 117ms/step - loss: 0.2018 - acc
```

```
result = model.evaluate(x_test_pad, y_test)
```

```
313/313 [==============================] - 6s 15ms/step - loss: 0.3755 - ac
```

```
x = model.predict(x_test_pad)
```

```
print(x)
```

```
[[0.9118545 ]
 [0.9136807 ]
 [0.08736128]
 ...
 [0.9157319 ]
 [0.86391574]
 [0.8981884 ]]
```

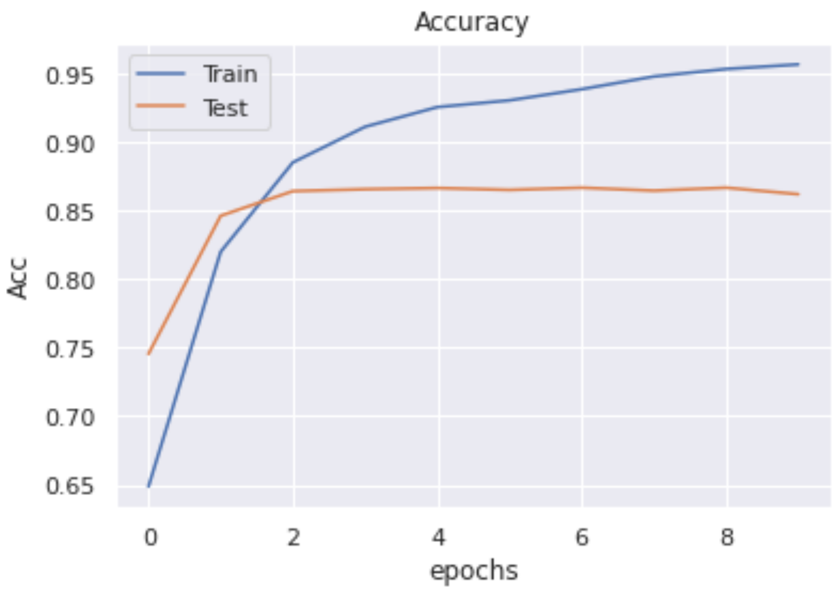
```
y = []
a = 0
for i in x:
  if i >= 0.5:
    a = 1
  else:
    a = 0
  y.append(a)
```

```
print(y)
```

```
[1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
```

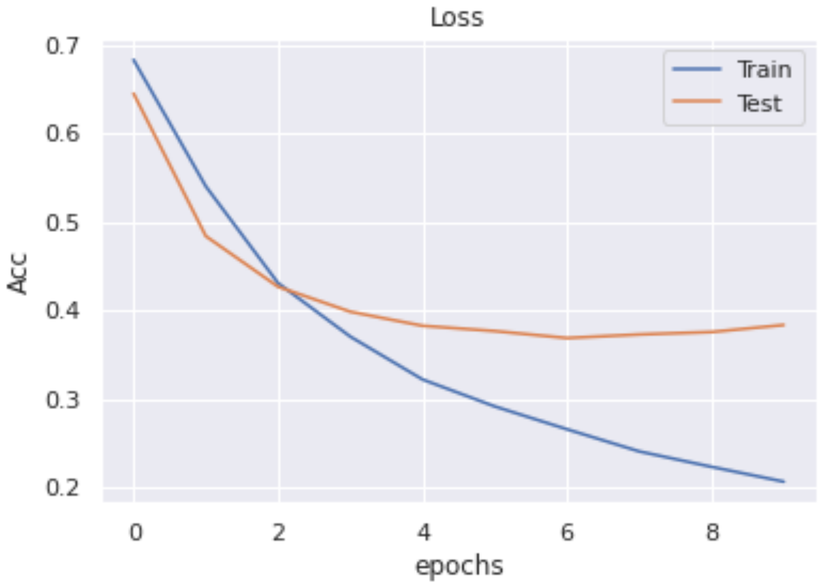
```
# Visualizing accuracy through graphical representation

plt.figure()
plt.plot(history.history["accuracy"], label = "Train")
plt.plot(history.history["val_accuracy"], label = "Test")
plt.title("Accuracy")
plt.ylabel("Acc")
plt.xlabel("epochs")
plt.legend()
plt.show()
```



```
# Visualizing loss through graphical representation

plt.figure()
plt.plot(history.history["loss"], label = "Train")
plt.plot(history.history["val_loss"], label = "Test")
plt.title("Loss")
plt.ylabel("Acc")
plt.xlabel("epochs")
plt.legend()
plt.show()
```




```
dataset_predict = df.copy()
dataset_predict = pd.DataFrame(dataset_predict)
dataset_predict.columns = ['review']
dataset_predict = dataset_predict.reset_index()
dataset_predict = dataset_predict.drop(['index'], axis=1)
dataset_predict.head()
```

	review
0	one reviewers mentioned watching oz episode ho...
1	wonderful little production br br filming tech...
2	thought wonderful way spend time hot summer we...
3	basically family little boy jake thinks zombie...
4	petter mattei love time money visually stunnin...

```
test_actual_label = sentiment.copy()
test_actual_label = pd.DataFrame(test_actual_label)
test_actual_label.columns = ['sentiment']
test_actual_label['sentiment'] = test_actual_label['sentiment'].replace({1: 'pos
test_actual_label['sentiment'].head()
```

```
0    positive
1    positive
2    positive
3    negative
4    positive
Name: sentiment, dtype: object
```

```
test_predicted_label = y.copy()
test_predicted_label = pd.DataFrame(test_predicted_label)
test_predicted_label.columns = ['predicted_sentiment']
test_predicted_label['predicted_sentiment'] = test_predicted_label['predicted_se
```

```
test_result = pd.concat([dataset_predict, test_actual_label, test_predicted_labe
test_result.head()
```

	review	sentiment	predicted_sentiment
0	one reviewers mentioned watching oz episode ho...	positive	positive
1	wonderful little production br br filming tech...	positive	positive
2	thought wonderful way spend time hot summer we...	positive	negative
3	basically family little boy jake thinks zombie...	negative	positive
4	petter mattei love time money visually stunnin...	positive	negative

2. PART TWO

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, ca
```

```
%tensorflow_version 2.x
import tensorflow
tensorflow.__version__
```

```
'2.4.1'
```

```
project_path = '/content/drive/MyDrive/My Files/AIML Workbooks'
```

```
import os                   # Importing os library
import pandas as pd         # To read the data set
import numpy as np          # Importing numpy library
import seaborn as sns       # For data visualization
import matplotlib.pyplot as plt      # Necessary library for plotting graphs
from glob import glob       # Importing necessary library
import tensorflow as tf     # Importing library
%matplotlib inline
sns.set(color_codes = True)

from sklearn import metrics          # Importing metrics
from sklearn.model_selection import train_test_split       # Splitting data into
from sklearn.metrics import classification_report, accuracy_score, recall_score,
from sklearn.preprocessing import StandardScaler           # Importing to standa
from sklearn.impute import SimpleImputer                   # Importing to fill i
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import PolynomialFeatures       # Importing polynomia
from sklearn.decomposition import PCA          # Importing to run pca analysis
from sklearn import svm              # Importing necessary library for model bui
from sklearn.ensemble import RandomForestClassifier        # Importing necessary
from sklearn.neighbors import KNeighborsClassifier         # Importing necessary
from sklearn import preprocessing              # Importing preprocessing librar

from sklearn.model_selection import KFold, cross_val_score           # Importing
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV           #
from sklearn.cluster import KMeans             # For KMeans cluster model build
from scipy.stats import zscore       # Import zscore library
from scipy.spatial.distance import cdist       # Importing cdist functionality
import tensorflow           # Importing tensorflow library
from tensorflow.keras.models import Sequential, Model              # Importing
from tensorflow.keras.utils import to_categorical          # Importing tensorflo
from tensorflow.keras import optimizers                    # Importing optimizer
from tensorflow.keras.layers import Dense, Dropout, Activation, BatchNormalizati
from tensorflow.keras.applications.mobilenet import preprocess_input           #
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLRO
from tensorflow.keras.applications.mobilenet import MobileNet      # Importing
from tensorflow.keras.losses import binary_crossentropy            # Importing
from tensorflow.keras.backend import log, epsilon          # Importing necessary
```

```
from keras.utils import np_utils     # Importing necessary library
from sklearn import svm              # Importing necessary library for model bui
from sklearn.svm import SVC          # Import svc library for model building

from skimage.color import rgb2gray              # Loading color library
from sklearn.preprocessing import OneHotEncoder           # Library for one hot
from sklearn.metrics import confusion_matrix              # Loading necessary l
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, i
from keras.preprocessing import image           # Importing necessary image libr
from tensorflow import keras         # Loading keras libaray
from tensorflow.keras.optimizers import Adam, SGD         # Importing optimizer
import cv2                  # Importing necessary library
from PIL import ImageFile            # Importing image library
from tqdm import tqdm                # Importing necessary library
import time                 # Importing time library
from mpl_toolkits.axes_grid1 import ImageGrid             # Importing necessary
from PIL import Image       # Importing image library

import re                   # Importing regular expression library
import nltk                 # Import necessary library
from nltk.corpus import stopwords               # Importing necessary library
from sklearn.feature_extraction.text import CountVectorizer         # Importing
from sklearn.preprocessing import MultiLabelBinarizer     # Imorting necessary
import json                 # Importing json to import file
import urllib               # Importing necessary library
from tensorflow.keras.preprocessing.text import Tokenizer           # Importing
from tensorflow.keras.preprocessing.sequence import pad_sequences             #
from wordcloud import WordCloud      # Importing necessary library


def parse_data(file):
  for I in open(file, 'r'):
    yield json.loads(I)

df = list(parse_data('/content/drive/MyDrive/My Files/AIML Workbooks/Sarcasm_Hea


len(df)

    26709
```

```
# The dataset is a list of dictionaries as seen below

df[0:10]
```

```
[{'article_link': 'https://www.huffingtonpost.com/entry/versace-black-code_
  'headline': "former versace store clerk sues over secret 'black code' for
  'is_sarcastic': 0},
 {'article_link': 'https://www.huffingtonpost.com/entry/roseanne-revival-re
  'headline': "the 'roseanne' revival catches up to our thorny political mo
  'is_sarcastic': 0},
 {'article_link': 'https://local.theonion.com/mom-starting-to-fear-son-s-we
  'headline': "mom starting to fear son's web series closest thing she will
  'is_sarcastic': 1},
 {'article_link': 'https://politics.theonion.com/boehner-just-wants-wife-to
  'headline': 'boehner just wants wife to listen, not come up with alternat
  'is_sarcastic': 1},
 {'article_link': 'https://www.huffingtonpost.com/entry/jk-rowling-wishes-s
  'headline': 'j.k. rowling wishes snape happy birthday in the most magical
  'is_sarcastic': 0},
 {'article_link': 'https://www.huffingtonpost.com/entry/advancing-the-world
  'headline': "advancing the world's women",
  'is_sarcastic': 0},
 {'article_link': 'https://www.huffingtonpost.com/entry/how-meat-is-grown-i
  'headline': 'the fascinating case for eating lab-grown meat',
  'is_sarcastic': 0},
 {'article_link': 'https://www.huffingtonpost.com/entry/boxed-college-tuiti
  'headline': 'this ceo will send your kids to school, if you work for his
  'is_sarcastic': 0},
 {'article_link': 'https://politics.theonion.com/top-snake-handler-leaves-s
  'headline': 'top snake handler leaves sinking huckabee campaign',
  'is_sarcastic': 1},
 {'article_link': 'https://www.huffingtonpost.com/entry/fridays-morning-ema
  'headline': "friday's morning email: inside trump's presser for the ages"
  'is_sarcastic': 0}]
```

```
# Seperating data into features and labels for futher processing

headline_tmp = []
labels = []

for items in df:
  headline_tmp.append(items['headline'])
  labels.append(items['is_sarcastic'])
```

```
headline_tmp[0:10]
```

```
["former versace store clerk sues over secret 'black code' for minority sho
 "the 'roseanne' revival catches up to our thorny political mood, for bette
 "mom starting to fear son's web series closest thing she will have to gran
 'boehner just wants wife to listen, not come up with alternative debt-redu
 'j.k. rowling wishes snape happy birthday in the most magical way',
 "advancing the world's women",
 'the fascinating case for eating lab-grown meat',
 'this ceo will send your kids to school, if you work for his company',
 'top snake handler leaves sinking huckabee campaign',
 "friday's morning email: inside trump's presser for the ages"]
```

```
labels[0:10]
```

```
[0, 0, 1, 1, 0, 0, 0, 0, 1, 0]
```

```
# Steps to remove stopwords and punctuations
nltk.download('stopwords')
stop = list(stopwords.words('english'))

# Remove stopwords from the headlines
headline = []
for sent in headline_tmp:
   filtered_list = []
   for word in sent.split():
     if not word in stop:
       filtered_list.append(word)
   join_str = ' '.join([str(ele) for ele in filtered_list])
   headline.append(join_str)
```

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

```
headline[0:10]
```

```
["former versace store clerk sues secret 'black code' minority shoppers",
 "'roseanne' revival catches thorny political mood, better worse",
 "mom starting fear son's web series closest thing grandchild",
 'boehner wants wife listen, come alternative debt-reduction ideas',
 'j.k. rowling wishes snape happy birthday magical way',
 "advancing world's women",
 'fascinating case eating lab-grown meat',
 'ceo send kids school, work company',
 'top snake handler leaves sinking huckabee campaign',
 "friday's morning email: inside trump's presser ages"]
```

Word Cloud

```
# Creating a dataframe from the oirginal json file

sarcastic = pd.read_json('/content/drive/MyDrive/My Files/AIML Workbooks/Sarcasm

sarcastic.head()
```

	article_link	headline	is_sarcastic
0	https://www.huffingtonpost.com/entry/versace-b...	former versace store clerk sues over secret 'b...	0
1	https://www.huffingtonpost.com/entry/roseanne-...	the 'roseanne' revival catches up to our thorn...	0
2	https://local.theonion.com/mom-starting-to-fea...	mom starting to fear son's web series closest	1

```
sarcastic.tail()
```

	article_link	headline	is_sarcastic
26704	https://www.huffingtonpost.com/entry/american-...	american politics in moral free-fall	0
26705	https://www.huffingtonpost.com/entry/americas-...	america's best 20 hikes	0
26706	https://www.huffingtonpost.com/entry/reparatio...	reparations and obama	0
		israeli ban targeting	

```
# Remvoing the article column from the dataframe we do not require it for out an

sarcastic.drop('article_link', axis =1, inplace = True )
```

```
sarcastic.head()
```

	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

```
# Filtering rows to display is_sarcastic = 1

sarcastic[sarcastic.is_sarcastic == 1]
```

	headline	is_sarcastic
2	mom starting to fear son's web series closest ...	1
3	boehner just wants wife to listen, not come up...	1
8	top snake handler leaves sinking huckabee camp...	1
15	nuclear bomb detonates during rehearsal for 's...	1
16	cosby lawyer asks why accusers didn't come for...	1
...	...	...
26693	new bailiff tired of hearing how old bailiff d...	1
26694	breaking: 'the onion' in kill range of boston ...	1
26695	seaworld crowd applauds for dolphin playfully ...	1
26702	pentagon to withhold budget figures out of res...	1
26703	pope francis wearing sweater vestments he got ...	1

11724 rows × 2 columns

```
# Filtering rows to display is_sarcastic = 0

sarcastic[sarcastic.is_sarcastic == 0]
```

	headline	is_sarcastic
0	former versace store clerk sues over secret 'b...	0
1	the 'roseanne' revival catches up to our thorn...	0
4	j.k. rowling wishes snape happy birthday in th...	0
5	advancing the world's women	0
6	the fascinating case for eating lab-grown meat	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

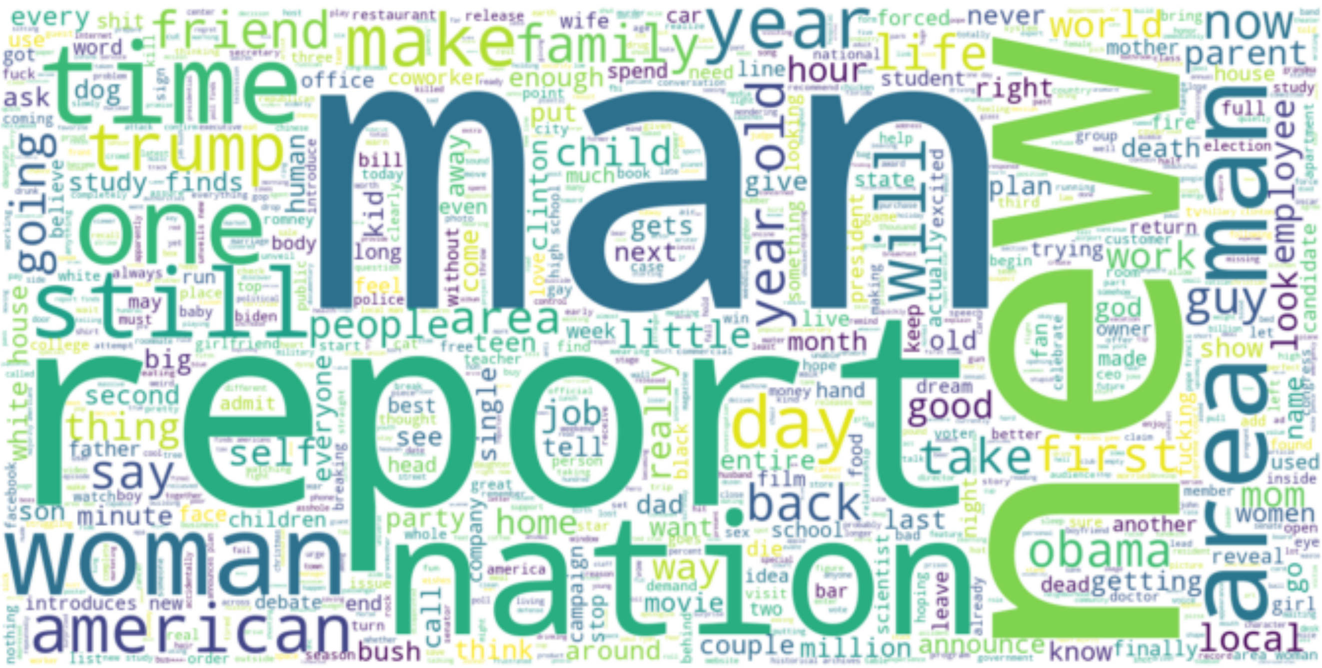
14985 rows × 2 columns

```
# Defining a word cloud for 1000 most frequent words in sarcastic headlines

wc = WordCloud(background_color='white', max_words = 1000, width= 1400, height =
wc.generate(' '.join(sarcastic[sarcastic.is_sarcastic == 1]['headline']))

print('Word cloud for 1000 most frequent words in sarcastic headline')
plt.figure(figsize = (12,12))
plt.imshow(wc, interpolation= 'bilinear')
plt.axis('off')
plt.show()
```

```
Word cloud for 1000 most frequent words in sarcastic headline
```



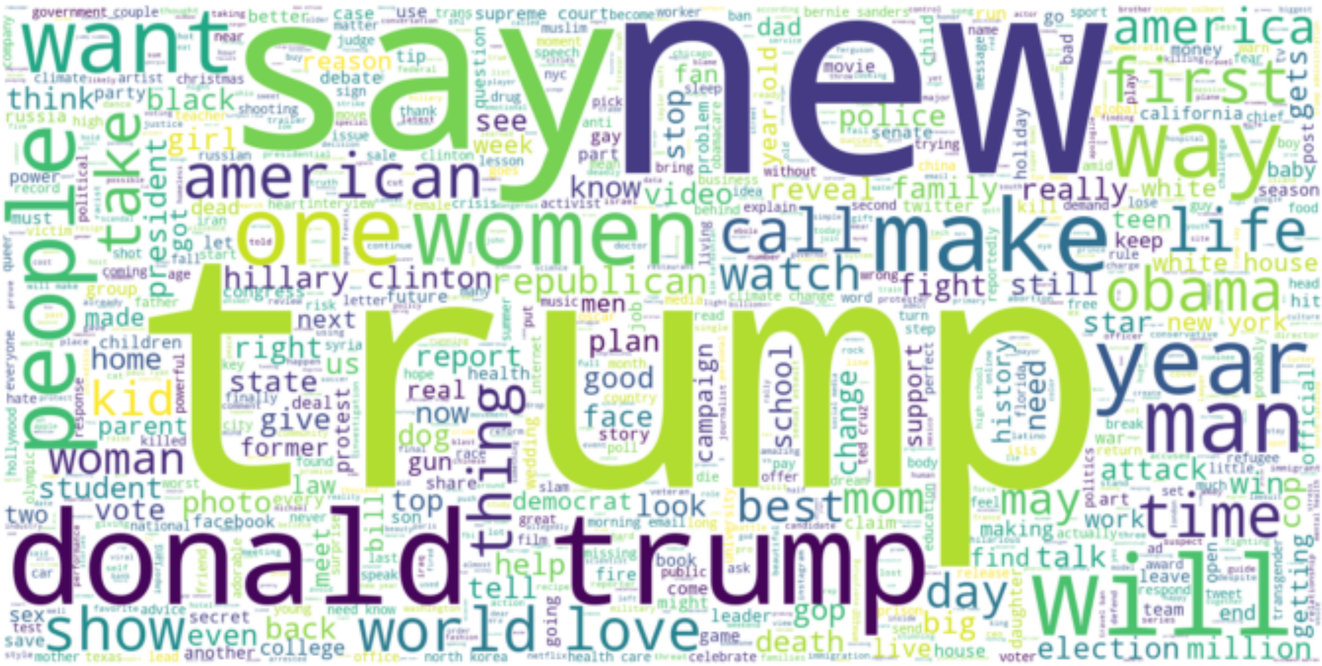
```
# Defining a word cloud for 1000 most frequent words in non saracatic headlines

wc = WordCloud(background_color = 'white', max_words = 1000, width = 1400, heigh
wc.generate(' '.join(sarcastic[sarcastic.is_sarcastic == 0]['headline']))

print('Words cloud for 1000 most frequent words in non sarcastic headlines')

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

```
Words cloud for 1000 most frequent words in non sarcastic headlines
```



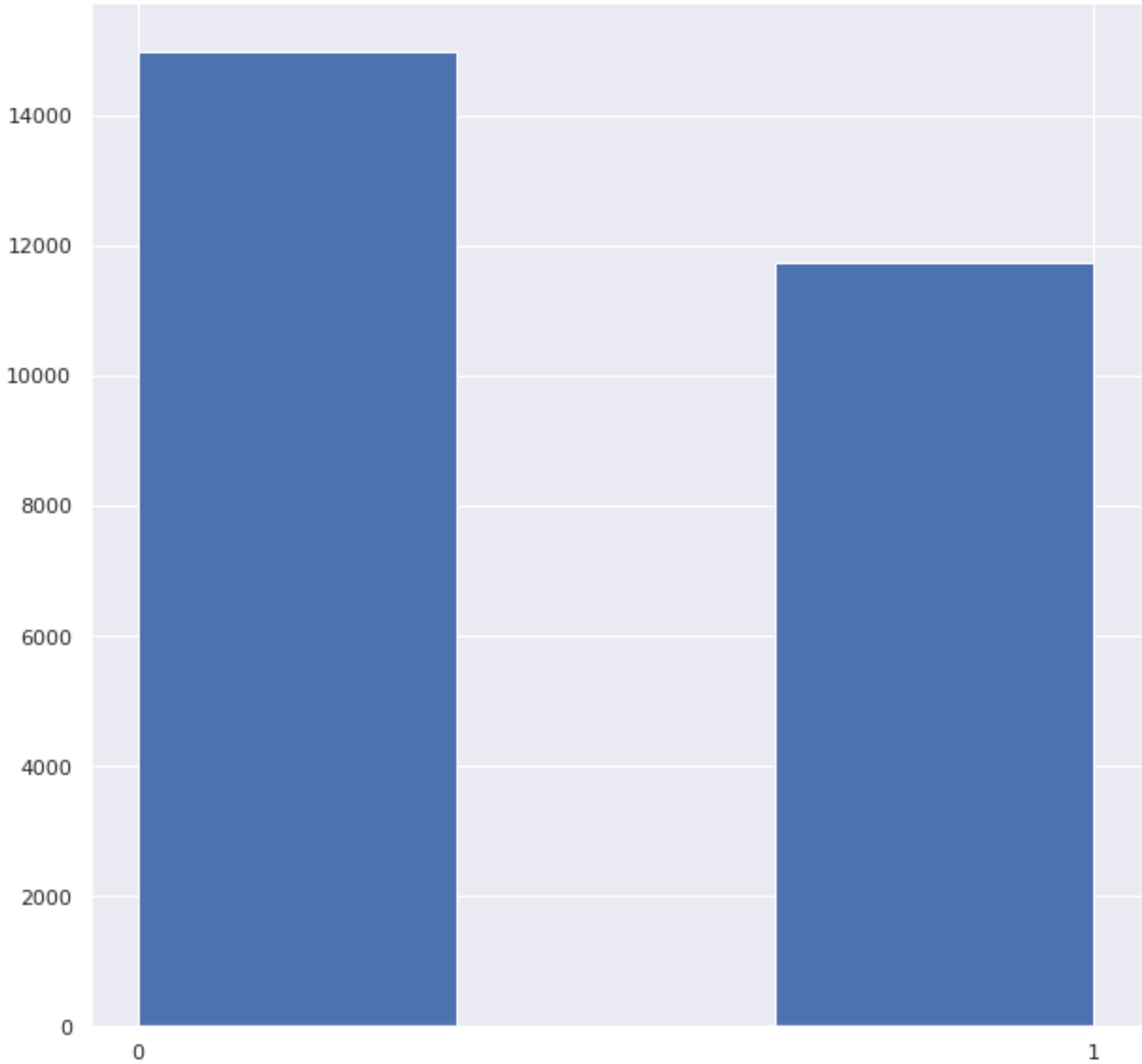
Distribution of Labels

```
sarcastic['is_sarcastic'].value_counts()
```

```
0    14985
1    11724
Name: is_sarcastic, dtype: int64
```

```
# Visualizing the count via histogram plot

plt.figure(figsize = (10,10))
plt.hist(sarcastic['is_sarcastic'], bins = 3)
plt.xticks([0,1])
plt.show()
```



Distribution of length of headlines

```
# Creating a new column in the dataframe with the number of words for each headl

sarcastic['headline_length'] = sarcastic['headline'].apply(lambda x: len(x.split
```

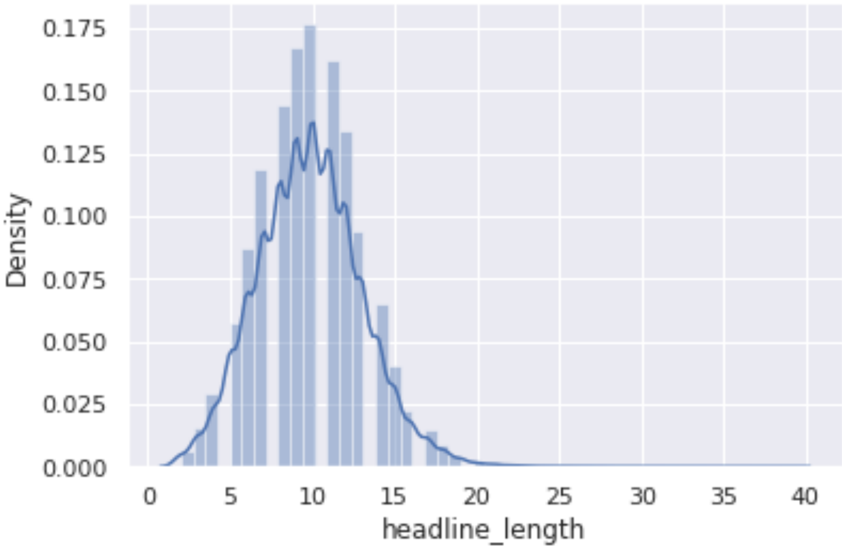
```
sarcastic['headline_length'].head()
```

```
0    12
1    14
2    14
3    13
4    11
Name: headline_length, dtype: int64
```

```
# Visualizing density plot of headline lengths

sns.distplot(sarcastic['headline_length']);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: Futur
  warnings.warn(msg, FutureWarning)
```




```
sarcastic['headline_length'].describe()
```

```
count    26709.000000
mean         9.845820
std          3.168955
min          2.000000
25%          8.000000
50%         10.000000
75%         12.000000
max         39.000000
Name: headline_length, dtype: float64
```

```
# Getting the 99.99th percentile using quantile method

sarcastic['headline_length'].quantile(0.9999)
```

```
30.32919999999649
```

Data Processing

```
# Define parameters

#embedding_dim = the dimnesion to which each of the words in the sentence will b
#max_length = Maximum length to be retained of each sentence(headline) for the t
#trunc_type = trucate(suffix) the sentence from the back if the sentence length
#padding_type = pad(suffix) the sentence with 0's at the back if the sentence le
#oov_token = Out Of Vocubulary token to be used if the word is not part of the v

embedding_dim = 100
max_length = 32
trunc_type='post'
padding_type='post'
oov_tok = "<OOV>"
training_size = 20000
```

```
# Split data into training and validation datasets using the training_size param

train_sentences = headline[:training_size]
train_labels = labels[:training_size]

validation_sentences = headline[training_size:]
validation_labels = labels[training_size:]

print(training_size)
print(len(train_sentences))
print(len(train_labels))
print(len(validation_sentences))
print(len(validation_labels))
```

```
20000
20000
20000
6709
6709
```

```
# Use tokenizer from Keras to tokenize and transform the words into numerical da
# Use pad_sequences from keras to pad the data to make it of same length(max_len

tokenizer = Tokenizer(oov_token= oov_tok)
tokenizer.fit_on_texts(headline)
word_index = tokenizer.word_index

training_sequences = tokenizer.texts_to_sequences(train_sentences)
training_padded = pad_sequences(training_sequences, maxlen = max_length, padding

validation_sequences = tokenizer.texts_to_sequences(validation_sentences)
validation_padded = pad_sequences(validation_sequences, maxlen = max_length, pad
```

```
# Vocabulary size will be the total number of the words in the word_index identi

len(word_index)
```

```
29590
```

```
training_sequences[0]
```

```
[216, 15046, 572, 3237, 2192, 287, 2472, 15047, 2473, 8352]
```

```
training_padded[0]
```

```
array([  216, 15046,   572,  3237,  2192,   287,  2472, 15047,  2473,
        8352,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0], dtype=int32)
```

```
# Convert into numpy array

training_padded = np.array(training_padded)
validation_padded = np.array(validation_padded)
training_labels_pad = np.array(train_labels)
validation_labels_pad = np.array(validation_labels)
```

Model 1. Using default embedding layer of of keras

```
# Define model

vocab_size = len(word_index)

tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

# Defining early stopping if validation accuracy does not change within 5 epochs

callback = EarlyStopping(patience = 5, verbose = 1, monitor = 'val_accuracy', re

model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size+1, embedding_dim, input_length=max_leng
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences = Tr
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')])


optimizer = tf.keras.optimizers.Adam()
model.compile(loss = 'binary_crossentropy', metrics = ['accuracy'], optimizer =
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 32, 100)           2959100
_________________________________________________________________
dropout (Dropout)            (None, 32, 100)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 32, 128)           84480
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 128)           0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32, 64)            41216
_________________________________________________________________
global_average_pooling1d (Gl (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 256)               16640
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 3,134,461
Trainable params: 3,134,461
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(training_padded, training_labels_pad, validation_data=(valid
```

```
Epoch 1/30
157/157 [==============================] - 5s 33ms/step - loss: 0.0049 - ac
Epoch 2/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0048 - ac
Epoch 3/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0054 - ac
Epoch 4/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0034 - ac
Epoch 5/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0046 - ac
Epoch 6/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0041 - ac
Epoch 7/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0030 - ac
Epoch 8/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0032 - ac
Epoch 9/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0031 - ac
Epoch 10/30
```

```
157/157 [==============================] - 4s 25ms/step - loss: 0.0038 - ac
Epoch 11/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0040 - ac
Epoch 12/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0032 - ac
Epoch 13/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0029 - ac
Epoch 14/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0014 - ac
Epoch 15/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0020 - ac
Epoch 16/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0038 - ac
Epoch 17/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0050 - ac
Epoch 18/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0039 - ac
Epoch 19/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0043 - ac
Epoch 20/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0011 - ac
Epoch 21/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0015 - ac
Epoch 22/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0012 - ac
Epoch 23/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0012 - ac
Epoch 24/30
157/157 [==============================] - 4s 26ms/step - loss: 9.3129e-05
Epoch 25/30
157/157 [==============================] - 4s 25ms/step - loss: 9.5440e-04
Epoch 26/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0026 - ac
Epoch 27/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0013 - ac
Epoch 28/30
157/157 [==============================] - 4s 26ms/step - loss: 8.2499e-04
Epoch 29/30
157/157 [==============================] - 4s 25ms/step - loss: 7.5011e-04
Epoch 30/30
157/157 [==============================] - 4s 26ms/step - loss: 6.5150e-04
```

Model 2. Using Glove Embeddings

```
vocab_size = len(word_index) #30813

# for each line in the glove embedding text file, the first value is the word an
# Store the values into a dictionary
embeddings_index = {}
with open('/content/drive/MyDrive/My Files/AIML Workbooks/glove.6B.100d.txt') as
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word]=coefs

# initialize a matrix of zeros and then assign the encoding for the words in the
embeddings_matrix = np.zeros((vocab_size+1, embedding_dim))
for key in sorted(word_index, key=word_index.get)[:vocab_size]:
    embedding_vector = embeddings_index.get(key)
    if embedding_vector is not None:
        embeddings_matrix[word_index[key]] = embedding_vector
```

```
embeddings_matrix.shape
```

```
(29591, 100)
```

```
# First 10 words and the corresponding index from the word_index dictionary whic

for key in sorted(word_index, key=word_index.get)[:10]:
  print(key,end=' ')
  print(word_index.get(key))
```

```
<OOV> 1
new 2
trump 3
man 4
one 5
report 6
year 7
area 8
donald 9
u 10
```

```
# Embeddings for the first 10 words

for key in sorted(word_index, key=word_index.get)[:10]:
  print(key,end=' ')
  print(embeddings_index.get(key))
```

```
<OOV> None
```

```
new [-4.3959e-02  1.8936e-01  6.6110e-01 -4.9007e-01  3.2211e-01 -3.4161e-0
 -6.8480e-02  3.1364e-01 -7.1142e-01  5.7436e-01 -3.3588e-01 -5.2279e-01
 -3.9075e-01 -8.9694e-02  4.6371e-01 -3.5610e-01  8.4576e-01 -2.6188e-02
 -1.9328e-01 -8.3846e-02  3.1806e-01 -1.9812e-01  3.0009e-01  6.9189e-02
  5.4470e-01 -5.9193e-01  5.4221e-01 -6.2876e-01 -5.3447e-01  4.2334e-01
  3.0869e-02  9.7164e-01 -5.6222e-01  4.5752e-02 -5.7100e-01  8.0185e-02
 -8.1434e-02 -6.0260e-01  1.6466e-01 -4.0281e-01 -4.7701e-01 -5.1950e-01
  1.2777e-01 -4.3775e-01  2.6602e-01  4.8752e-01 -6.0220e-02 -5.2622e-01
  3.7687e-01 -1.8007e-01  3.0166e-02 -9.4577e-02  1.6330e-01  5.9041e-01
 -4.8877e-01 -3.4230e+00  1.3113e-01 -8.0386e-02  1.8978e+00  1.8857e-01
 -5.7300e-01  8.6358e-01  2.1116e-03  3.6060e-01  8.0475e-01 -1.3954e-01
 -5.3935e-02  3.8873e-01  3.0673e-01 -3.1395e-01  8.3238e-02 -4.1737e-01
 -1.0998e+00 -8.8005e-01  2.1550e-01 -2.6132e-01 -1.0091e-01  7.9584e-02
 -1.2341e+00 -6.5281e-01  6.3363e-01 -9.8491e-02  3.3518e-01  2.6332e-01
 -9.6427e-01 -1.4150e-02  3.0849e-01 -3.1418e-01 -4.0793e-01 -4.2900e-01
  8.5451e-02 -2.0073e-01  5.5050e-02 -4.0922e-02 -9.4015e-01  6.9544e-02
 -4.5397e-01 -1.4168e-01  9.2789e-01  5.9058e-01]
trump [-0.15731  -0.75503   0.36845  -0.18958  -0.16896  -0.23157  -0.22658
 -0.30186   0.24372   0.61896   0.58995   0.047638 -0.055164 -0.70211
  0.22084  -0.69232   0.49419   1.4285   -0.25362   0.20031  -0.26192
  0.05315  -0.048418 -0.44982   0.54644  -0.014645 -0.015531 -0.61197
 -0.91964  -0.7528    0.64843   1.0934    0.052682  0.33345   0.10532
  0.59517   0.023104 -0.37105   0.29749  -0.23683   0.079566 -0.10326
  0.35885  -0.28935  -0.19881   0.22908  -0.061435  0.56127  -0.017115
 -0.32868  -0.78417  -0.49375   0.34944   0.16278  -0.061168 -1.3106
  0.39152   0.124    -0.20873  -0.18473  -0.56184   0.55693   0.012114
 -0.54545  -0.31409   0.1       0.31543   0.74757  -0.47734  -0.18332
 -0.65623   0.40768  -0.30697  -0.47247  -0.7421   -0.44978  -0.078122
 -0.52673  -0.70633   1.3271    0.26298  -0.91      0.91632  -0.51643
  0.20284  -0.25402  -1.2566    0.20271   0.92105  -0.57574  -0.15105
 -0.24831   0.36673  -0.53987   0.18534   0.25713   0.38794  -0.54137
  0.67817  -0.17251 ]
man [ 3.7293e-01  3.8503e-01  7.1086e-01 -6.5911e-01 -1.0128e-03  9.2715e-0
  2.7615e-01 -5.6203e-02 -2.4294e-01  2.4632e-01 -1.8449e-01  3.1398e-01
  4.8983e-01  9.2560e-02  3.2958e-01  1.5056e-01  5.7317e-01 -1.8529e-01
 -5.2277e-01  4.6191e-01  9.2038e-01  3.1001e-02 -1.6246e-01 -4.0567e-01
  7.8621e-01  5.7722e-01 -5.3501e-01 -6.8228e-01  1.6987e-01  3.6310e-01
 -7.1773e-02  4.7233e-01  2.7806e-02 -1.4951e-01  1.7543e-01 -3.7573e-01
 -7.8517e-01  5.8171e-01  8.6859e-01  3.1445e-02 -4.5897e-01 -4.0917e-02
  9.5897e-01 -1.6975e-01  1.3045e-01  2.7434e-01 -6.9485e-02  2.2402e-02
  2.4977e-01 -2.1536e-01 -3.2406e-01 -3.9867e-01  6.8613e-01  1.7923e+00
 -3.7848e-01 -2.2477e+00 -7.7025e-01  4.6582e-01  1.2411e+00  5.7756e-01
  4.1151e-01  8.4328e-01 -5.4259e-01 -1.6715e-01  7.3927e-01 -9.3477e-02
  9.0278e-01  5.0889e-01 -5.0031e-01  2.6451e-01  1.5443e-01 -2.9432e-01
  1.0906e-01 -2.6667e-01  3.5438e-01  4.9079e-02  1.8018e-01 -5.8590e-01
 -5.5542e-01 -2.8987e-01  7.4278e-01  3.4530e-01 -2.8757e-02 -2.2646e-01
 -1.3113e+00 -5.7190e-01 -5.2306e-01 -1.2670e-01 -9.8678e-02 -5.3463e-01
  2.8607e-01 -3.7501e-01  4.5742e-01  4.5975e-02 -2.4675e-01  4.5656e-02
 -3.8302e-01 -9.3711e-01  3.9138e-02 -5.3911e-01]
one [-0.22557   0.49418   0.4861   -0.4332    0.13738   0.50617   0.26058
  0.30103  -0.091486  0.10876   0.3058    0.051028  0.22303   0.054236
  0.068838 -0.24701   0.32689  -0.082203 -0.28866   0.3734    0.73804
 -0.040969  0.040201  0.11384   0.69987  -0.49745  -0.06755  -0.42599
 -0.10725  -0.010697 -0.01479   0.55976   0.3064    0.053053  0.058034
```

```
 0.32756  -0.37233   0.46513   0.14285  -0.085003 -0.45476   0.19773
 0.6383   -0.31148   0.10858   0.31557   0.36682  -0.35135  -0.48414
-0.33235  -0.33816  -0.39678   0.1908    1.3513   -0.39044  -2.8795
-0.14276  -0.087754  1.7713    0.99332  -0.14129   0.94389   0.050897
```

```
embeddings_matrix[word_index['new']]
```

```
array([-4.39589992e-02,  1.89359993e-01,  6.61099970e-01, -4.90069985e-01,
        3.22109997e-01, -3.41610014e-01, -6.84799999e-02,  3.13639998e-01,
       -7.11420000e-01,  5.74360013e-01, -3.35880011e-01, -5.22790015e-01,
       -3.90749991e-01, -8.96940008e-02,  4.63710010e-01, -3.56099993e-01,
        8.45759988e-01, -2.61879992e-02, -1.93279997e-01, -8.38460028e-02,
        3.18060011e-01, -1.98119998e-01,  3.00089985e-01,  6.91889971e-02,
        5.44700027e-01, -5.91929972e-01,  5.42209983e-01, -6.28759980e-01,
       -5.34470022e-01,  4.23339993e-01,  3.08689997e-02,  9.71639991e-01,
       -5.62219977e-01,  4.57520001e-02, -5.70999980e-01,  8.01850036e-02,
       -8.14339966e-02, -6.02599978e-01,  1.64660007e-01, -4.02810007e-01,
       -4.77010012e-01, -5.19500017e-01,  1.27770007e-01, -4.37750012e-01,
        2.66020000e-01,  4.87520009e-01, -6.02199994e-02, -5.26220024e-01,
        3.76870006e-01, -1.80069998e-01,  3.01660001e-02, -9.45769995e-02,
        1.63299993e-01,  5.90409994e-01, -4.88770008e-01, -3.42300010e+00,
        1.31129995e-01, -8.03859979e-02,  1.89779997e+00,  1.88569993e-01,
       -5.73000014e-01,  8.63579988e-01,  2.11160001e-03,  3.60599995e-01,
        8.04750025e-01, -1.39540002e-01, -5.39349988e-02,  3.88729990e-01,
        3.06730002e-01, -3.13950002e-01,  8.32379982e-02, -4.17369992e-01,
       -1.09979999e+00, -8.80050004e-01,  2.15499997e-01, -2.61319995e-01,
       -1.00910001e-01,  7.95840025e-02, -1.23409998e+00, -6.52809978e-01,
        6.33629978e-01, -9.84909981e-02,  3.35180014e-01,  2.63319999e-01,
       -9.64269996e-01, -1.41500002e-02,  3.08490008e-01, -3.14179987e-01,
       -4.07929987e-01, -4.28999990e-01,  8.54509994e-02, -2.00729996e-01,
        5.50500005e-02, -4.09220010e-02, -9.40150023e-01,  6.95440024e-02,
       -4.53969985e-01, -1.41680002e-01,  9.27890003e-01,  5.90579987e-01])
```

```
# Define model

vocab_size = len(word_index)

tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

# Defining early stopping if validation accuracy does not change within 5 epochs

callback = EarlyStopping(patience = 5, verbose = 1, monitor = 'val_accuracy', re

model_ge = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size+1, embedding_dim, input_length=max_leng
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences = Tr
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')])


optimizer_ge = tf.keras.optimizers.Adam()
model_ge.compile(loss = 'binary_crossentropy', metrics = ['accuracy'], optimizer
```

```
model_ge.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 32, 100)           2959100
_________________________________________________________________
dropout (Dropout)            (None, 32, 100)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 32, 128)           84480
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 128)           0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32, 64)            41216
_________________________________________________________________
global_average_pooling1d (Gl (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 256)               16640
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 3,134,461
Trainable params: 3,134,461
Non-trainable params: 0
_________________________________________________________________
```

```
history_ge = model_ge.fit(training_padded, training_labels_pad, batch_size = 128
```

```
Epoch 1/30
157/157 [==============================] - 10s 37ms/step - loss: 0.6307 - a
Epoch 2/30
157/157 [==============================] - 4s 25ms/step - loss: 0.2713 - ac
Epoch 3/30
157/157 [==============================] - 4s 26ms/step - loss: 0.1321 - ac
Epoch 4/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0779 - ac
Epoch 5/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0529 - ac
Epoch 6/30
157/157 [==============================] - 4s 25ms/step - loss: 0.0356 - ac
Epoch 7/30
157/157 [==============================] - 4s 26ms/step - loss: 0.0273 - ac
Restoring model weights from the end of the best epoch.
Epoch 00007: early stopping
```

```
# Saving the model

model.save("sarcasm_glove_mymodel.h5")


# Retrieve and evaluate the model with the validation set

model = keras.models.load_model('sarcasm_glove_mymodel.h5')


model.evaluate(validation_padded,validation_labels_pad)
```

```
210/210 [==============================] - 2s 4ms/step - loss: 2.6553 - acc
[2.6552867889404297, 0.7707557082176208]
```

```
# Prediction using the same validation dataset to plot confusion matrix

pred = (model.predict(validation_padded) > 0.5).astype("int32")


# Confusion Matrix

from sklearn import metrics

cm = metrics.confusion_matrix(validation_labels_pad,pred)
print(cm)
```

```
[[3155  624]
 [ 914 2016]]
```

```
df_cm = pd.DataFrame(cm, index = [i for i in ['Not_sarcastic','Sarcastic']],
                                  columns = [i for i in ['Not_sarcastic','Sarcas
plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True, fmt = '1g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0bf22aaa50>
```