Proj74ThesisReport_SuryaMunjal2

October 9, 2022

[1]:
```
pip install lime
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
     |                 | 275 kB 28.4 MB/s
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.7/dist-packages (from lime) (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from lime) (1.21.6)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from lime) (1.7.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
(from lime) (4.64.1)
Requirement already satisfied: scikit-learn>=0.18 in
/usr/local/lib/python3.7/dist-packages (from lime) (1.0.2)
Requirement already satisfied: scikit-image>=0.12 in
/usr/local/lib/python3.7/dist-packages (from lime) (0.18.3)
Requirement already satisfied: tifffile>=2019.7.26 in
/usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime)
(2021.11.2)
Requirement already satisfied: PyWavelets>=1.1.1 in
/usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime) (1.3.0)
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime) (7.1.2)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-
packages (from scikit-image>=0.12->lime) (2.9.0)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-
packages (from scikit-image>=0.12->lime) (2.6.3)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->lime) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-
packages (from matplotlib->lime) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->lime) (1.4.4)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->lime) (2.8.2)
```

```
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from
kiwisolver>=1.0.1->matplotlib->lime) (4.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-
packages (from python-dateutil>=2.1->matplotlib->lime) (1.15.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn>=0.18->lime) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.18->lime) (3.1.0)
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) … done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283857
sha256=2d489edb3227bd90f833d68b3c71d84a3c171d0f85d9674deb5eb506bbc997ce
  Stored in directory: /root/.cache/pip/wheels/ca/cb/e5/ac701e12d365a08917bf4c61
71c0961bc880a8181359c66aa7
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import sqlite3
from sqlalchemy import create_engine # database connection
import csv
import os
warnings.filterwarnings("ignore")
import datetime as dt
import numpy as np
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
#from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.metrics import log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
#from sklearn.cross_validation import StratifiedKFold
```

```
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier



from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
#from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_recall_curve, auc, roc_curve
!pip install -q kaggle
import re
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from wordcloud import WordCloud, STOPWORDS
from os import path

import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import sqlite3
from sqlalchemy import create_engine # database connection
import csv
import os
warnings.filterwarnings("ignore")
import datetime as dt
import numpy as np
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
```

```
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
#from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.metrics import log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
#from sklearn.cross_validation import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier



from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
#from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_recall_curve, auc, roc_curve
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
```

[3]:
```
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving kaggle.json to kaggle (1).json
```

[3]:
```
{'kaggle.json':
b'{"username":"suryamunjal","key":"376b73a1d8c76f470856f9ab7d5318e5"}'}
```

[4]:
```
!mkdir ~/.kaggle
```

[5]:
```
!cp kaggle.json ~/.kaggle/
```

[6]:
```
!chmod 600 ~/.kaggle/kaggle.json
```

[7]:
```
!kaggle datasets download -d akshayaki/fakenews
```

```
Downloading fakenews.zip to /content
 67% 24.0M/35.6M [00:00<00:00, 247MB/s]
100% 35.6M/35.6M [00:00<00:00, 283MB/s]
```

[8]:
```
!unzip fakenews.zip
```

```
Archive:  fakenews.zip
  inflating: News.csv
```

Reading the Data Set

[9]:
```
df=pd.read_csv('News.csv')
```

[10]:
```
df.head(2)
```

[10]:
```
   Unnamed: 0                                              title  \
0           0  GRAPHIC RIOT VIDEOS EXPOSE THUGS ATTACKING ELD…
1           1  BIG BROTHER: FEDS WANT YOUR DOCTOR TO WARN YOU…

                                                text          subject  \
0  youngers these days are becoming so moist pic…         politics
1  totally out of bounds! This is so wrong and so…  Government News

           date Labels
0  Sep 22, 2016   Fake
1  Jun 26, 2015   Fake
```

[11]:
```
df['combined']=df['title']+ " " + df["text"]
```

[12]:
```
df['Labels']=df['Labels'].map({'Fake': 0, 'True': 1})
```

[13]:
```
df=df[['combined','Labels']]
```

[14]:
```
df.head(1)
```

[14]:
```
                                            combined  Labels
0  GRAPHIC RIOT VIDEOS EXPOSE THUGS ATTACKING ELD…       0
```

data cleaning

[15]:
```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

def preproc(sentance):
#from tqdm import tqdm
# tqdm is for printing the status bar
#for sentance in tqdm(df['combined'].values):
    sentance = re.sub(r"http\S+", "url", sentance)  #removing urls with space
    sentance = BeautifulSoup(sentance, 'lxml').get_text() # removes tags like␣
 ↪<br>
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()  # remove words with␣
 ↪numbers
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)  ##remove spacial character:
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in␣
 ↪stopwords)

    return(sentance.strip())
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[16]:
```
df['Final_text']=df['combined'].apply(lambda x: preproc(str(x)))


#dropping the original combined column
df.drop(['combined'],inplace=True,axis=1)
```

[17]:
```
df.head(2)
```

[17]:

	Labels	Final_text
0	0	graphic riot videos expose thugs attacking eld…
1	0	big brother feds want doctor warn global warmi…

TRAIN AND TEST SPLIT

[18]:
```
# split data into test and train

X_train, X_test, y_train, y_test = train_test_split(df['Final_text'], df.
↪Labels, test_size = 0.3)
```

[19]:
```
X_train.shape
```

[19]:
```
(31428,)
```

[20]:
```
X_test.shape
```

[20]:
```
(13470,)
```

preparing datasets for rnn models

[21]:
```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize, sent_tokenize

maxlen = -1
for i in df['Final_text']:
    tokens = nltk.word_tokenize(i)
    if(maxlen<len(tokens)):
        maxlen = len(tokens)
print("The maximum number of words in any document is =", maxlen)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
The maximum number of words in any document is = 4963
```

[22]:
```
list_of_words = []
for i in df['Final_text']:
    for j in i.split():
        list_of_words.append(j)
```

[23]:
```
len(list_of_words)
```

[23]:
```
10833916
```

[24]:
```
total=len(set(list_of_words))
print("total no of unique words are",total)
```

```
total no of unique words are 105035
```

[25]:
```
from tensorflow.keras.preprocessing.text import one_hot, Tokenizer
voc_size=total #we could have giveb max like 5000
#onehot_repr=[one_hot(words,total)for words in df['Final_text']]
#onehot_repr[0]

# Create a tokenizer to tokenize the words and create sequences of tokenized␣
 ↪words
tokenizer = Tokenizer(num_words = total)
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)

X_test = tokenizer.texts_to_sequences(X_test)

#X_train = tokenizer.sequences_to_matrix(train_sequences, mode='binary')
#X_test = tokenizer.sequences_to_matrix(test_sequences, mode='binary')
```

[26]:
```
#https://machinelearningknowledge.ai/
 ↪keras-tokenizer-tutorial-with-examples-for-fit_on_texts-texts_to_sequences-texts_to_matrix-
 ↪
'''
docs =['Machine Learning Knowledge',
       'Machine Learning and Deep Learning',
       'Deep Learning',
       'Artificial Intelligence']

# create the tokenizer
t = Tokenizer()

t.fit_on_texts(docs)

sequences = t.texts_to_sequences(docs)
print(sequences)

encoded_docs = t.sequences_to_matrix(sequences, mode='binary')
print(encoded_docs)

'''
```

[26]:
```
"\ndocs =['Machine Learning Knowledge',\n       'Machine Learning and Deep
Learning',\n       'Deep Learning',\n       'Artificial Intelligence']\n\n#
create the tokenizer\nt = Tokenizer()\n\nt.fit_on_texts(docs)\n\nsequences =
t.texts_to_sequences(docs)\nprint(sequences)\n\nencoded_docs =
```

```
t.sequences_to_matrix(sequences, mode='binary')\nprint(encoded_docs)\n\n"
```

[27]:
```
len(X_train)
```

[27]:
```
31428
```

[28]:
```
# Add padding can either be maxlen or smaller number maxlen = 600 seems to work␣
↪well based on results
from tensorflow.keras.preprocessing.sequence import pad_sequences
# truncate and/or pad input sequences
max_length = 600
X_train = sequence.pad_sequences(X_train, maxlen=max_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_length)

print(X_train.shape)
print(X_train[1])
```

```
(31428, 600)
[    0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
```

```
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0  3319  1598 11147   802  3138  1658     1 19982    19
 4938  4464 14826    15     1   351    39   238 12241  3691  2895  5810
 5753  4361    19  4438    19  2766 13799  5901   201  1119    19  2767
  282  2755   927  1702  1755   394  1846  3555    43  1066   282 19983
 1357   640    19  2766   557 10074    91   129  9702  5035   202   129
  456   100   885  8211   800  1024  4464   223   410     1  1846    73
 2698   132   282    84    37   264  1560   330  1030  4476 19001  2234
18163     6  3080   596   260   264    19  2766   251     1   524  4299
  225    35  5810  5753  4361  1598 11147   802  3138  6800   899  8529
    1    73   684  1978   800  1531     4     1  7730  4668   499   684
 2755  3655  1388   909  4114  2813   400   802  3138 42880   539   388
 1094  2939   388     1  2154   267   225    15     1   481    19  2767
12241    18    29    29    19  4438     1   771   141  4694   684   187
  293  2847     1     4   117     4    19  4438  3033  3033  9918    18
   37  2861  2704   516    29   151   107    76  1677  5431   438   370]
```

CREATING A LSTM MODEL

[29]:
```
# create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(total, embedding_vecor_length, input_length=max_length))
model.add(LSTM(8))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
print(model.summary())
#Refer: https://datascience.stackexchange.com/questions/10615/
 ↪number-of-parameters-in-an-lstm-model

#log loss is  loss='binary_crossentropy' specifies that your model should␣
 ↪optimize the log loss for binary classification.
#metrics=['accuracy'] specifies that accuracy should be printed out, but log␣
 ↪loss is also printed out by default.
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding (Embedding)        (None, 600, 32)           3361120

 lstm (LSTM)                  (None, 8)                 1312
```

```
 dense (Dense)               (None, 1)                 9

=================================================================
Total params: 3,362,441
Trainable params: 3,362,441
Non-trainable params: 0
_________________________________________________________________
None
```

[30]:
```
history=model.fit(X_train, y_train, epochs=5, batch_size=64,validation_split=0.
↪3)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Epoch 1/5
344/344 [==============================] - 16s 25ms/step - loss: 0.2733 -
accuracy: 0.9179 - val_loss: 0.0993 - val_accuracy: 0.9785
Epoch 2/5
344/344 [==============================] - 8s 23ms/step - loss: 0.0671 -
accuracy: 0.9859 - val_loss: 0.0689 - val_accuracy: 0.9833
Epoch 3/5
344/344 [==============================] - 8s 23ms/step - loss: 0.0321 -
accuracy: 0.9937 - val_loss: 0.0647 - val_accuracy: 0.9837
Epoch 4/5
344/344 [==============================] - 8s 23ms/step - loss: 0.0242 -
accuracy: 0.9945 - val_loss: 0.0576 - val_accuracy: 0.9846
Epoch 5/5
344/344 [==============================] - 8s 23ms/step - loss: 0.0129 -
accuracy: 0.9978 - val_loss: 0.0606 - val_accuracy: 0.9836
Accuracy: 98.67%
```

[31]:
```
print(history.history.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[32]:
```
print(history.history['val_loss'])
```

```
[0.09927283972501755, 0.0689343586564064, 0.06465856730937958,
0.057585202157497406, 0.060601186007261276]
```

[33]:
```
print(history.history['val_accuracy'])
```

```
[0.9784706830978394, 0.9833492636680603, 0.9836674332618713, 0.9846218824386597,
0.9835613369941711]
```

[34]:
```
print(scores)
```

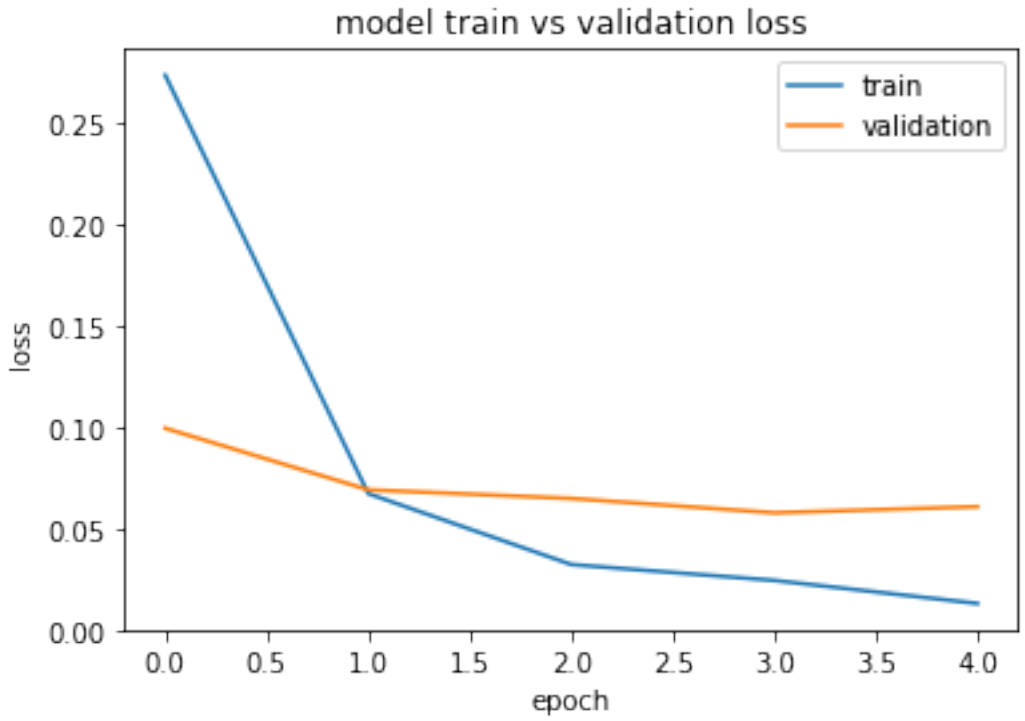
```
[0.053384117782115936, 0.9867112040519714]
```

[35]:
```
print(model.metrics_names)
```

```
['loss', 'accuracy']
```

1 so we are getting a test log loss of 0.05 in lstm

[36]:
```
# plot train and validation loss
from matplotlib import pyplot
pyplot.plot(history.history['loss'])
pyplot.plot(history.history['val_loss'])
pyplot.title('model train vs validation loss')
pyplot.ylabel('loss')
pyplot.xlabel('epoch')
pyplot.legend(['train', 'validation'], loc='upper right')
pyplot.show()
```



[37]:
```
from sklearn.metrics import confusion_matrix
y_pred = model.predict(X_test)
confusion_matrix = confusion_matrix(y_test, np.rint(y_pred))
print(confusion_matrix)
```

```
from sklearn.metrics import classification_report
target_names = ['fake 0', 'real 1']
print(classification_report(y_test, np.rint(y_pred),target_names=target_names))
```

```
[[6995   99]
 [  80 6296]]
              precision    recall  f1-score   support

      fake 0       0.99      0.99      0.99      7094
      real 1       0.98      0.99      0.99      6376

    accuracy                           0.99     13470
   macro avg       0.99      0.99      0.99     13470
weighted avg       0.99      0.99      0.99     13470
```

[38]:
```
print(confusion_matrix)
```

```
[[6995   99]
 [  80 6296]]
```

[39]:
```
from sklearn.metrics import classification_report
target_names = ['fake 0', 'real 1']
print(classification_report(y_test, np.rint(y_pred),target_names=target_names))
```

```
              precision    recall  f1-score   support

      fake 0       0.99      0.99      0.99      7094
      real 1       0.98      0.99      0.99      6376

    accuracy                           0.99     13470
   macro avg       0.99      0.99      0.99     13470
weighted avg       0.99      0.99      0.99     13470
```

[40]:
```
!pip install lime
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: lime in /usr/local/lib/python3.7/dist-packages
(0.2.0.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from lime) (1.7.3)
Requirement already satisfied: scikit-image>=0.12 in
/usr/local/lib/python3.7/dist-packages (from lime) (0.18.3)
Requirement already satisfied: scikit-learn>=0.18 in
/usr/local/lib/python3.7/dist-packages (from lime) (1.0.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
```

```
(from lime) (4.64.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-
packages (from lime) (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from lime) (1.21.6)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-
packages (from scikit-image>=0.12->lime) (2.6.3)
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime) (7.1.2)
Requirement already satisfied: PyWavelets>=1.1.1 in
/usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime) (1.3.0)
Requirement already satisfied: tifffile>=2019.7.26 in
/usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime)
(2021.11.2)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-
packages (from scikit-image>=0.12->lime) (2.9.0)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->lime) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->lime) (1.4.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->lime) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-
packages (from matplotlib->lime) (0.11.0)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from
kiwisolver>=1.0.1->matplotlib->lime) (4.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-
packages (from python-dateutil>=2.1->matplotlib->lime) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.18->lime) (3.1.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn>=0.18->lime) (1.1.0)
```

[41]:
```
from lime.lime_text import LimeTextExplainer
class_names=['0','1']
explainer= LimeTextExplainer(class_names=class_names)
def predict_proba(arr):
  processed=[]
  for i in arr:
    processed.append(preproc(i))
  list_tokenized_ex = tokenizer.texts_to_sequences(processed)
  Ex = pad_sequences(list_tokenized_ex, maxlen=max_length)
  pred=model.predict(Ex)
  returnable=[]
  for i in pred:
    temp=i[0]
```

```
    returnable.append(np.array([1-temp,temp])) #I would recommend rounding temp␣
↪and 1-temp off to 2 places
 return np.array(returnable)
```

[42]:
```
data=pd.read_csv('News.csv')
data['combined']=data['title']+" "+data['text']
data.head(1)
```

[42]:
```
   Unnamed: 0                                              title  \
0           0  GRAPHIC RIOT VIDEOS EXPOSE THUGS ATTACKING ELD…

                                                text    subject          date  \
0  youngers these days are becoming so moist pic…  politics  Sep 22, 2016

  Labels                                                combined
0   Fake  GRAPHIC RIOT VIDEOS EXPOSE THUGS ATTACKING ELD…
```

[43]:
```
print("Actual",data['Labels'][666])
explainer.explain_instance(data['combined'][666],predict_proba).
↪show_in_notebook(text=True)
```

```
Actual Fake
```

```
<IPython.core.display.HTML object>
```

[44]:
```
print("Actual",data['Labels'][4437])
explainer.explain_instance(data['combined'][4437],predict_proba).
↪show_in_notebook(text=True)
```

```
Actual True
```

```
<IPython.core.display.HTML object>
```

2 LETS CREATE A BIDIRECTIONAL RNN MODEL

[45]:
```
## Creating model
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import Dropout
# create the model

embedding_vecor_length = 32
model1 = Sequential()
model1.add(Embedding(total, embedding_vecor_length, input_length=max_length))
model1.add(Bidirectional(LSTM(8)))
model1.add(Dropout(0.3))
```

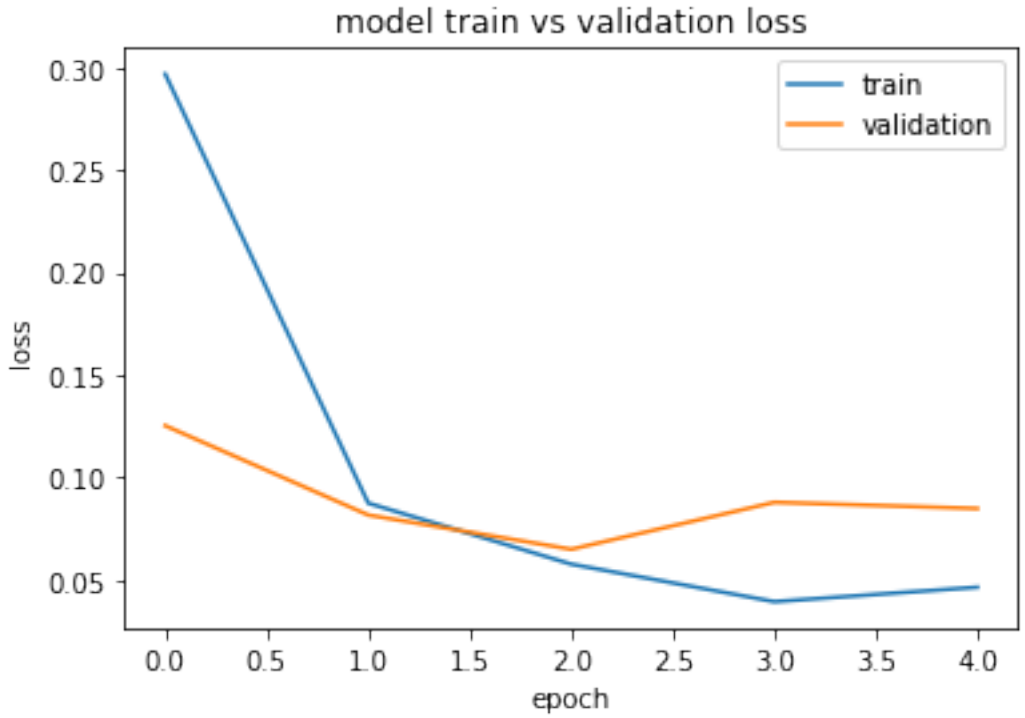
```
model1.add(Dense(1, activation='sigmoid'))
model1.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
print(model1.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, 600, 32)           3361120

 bidirectional (Bidirectiona  (None, 16)               2624
 l)

 dropout (Dropout)            (None, 16)                0

 dense_1 (Dense)              (None, 1)                 17

=================================================================
Total params: 3,363,761
Trainable params: 3,363,761
Non-trainable params: 0
_________________________________________________________________
None
```

[46]:
```
history=model1.fit(X_train, y_train, epochs=5, batch_size=64,validation_split=0.
 ↪3)
# Final evaluation of the model
scores = model1.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Epoch 1/5
344/344 [==============================] - 27s 69ms/step - loss: 0.2967 -
accuracy: 0.9174 - val_loss: 0.1252 - val_accuracy: 0.9722
Epoch 2/5
344/344 [==============================] - 16s 48ms/step - loss: 0.0874 -
accuracy: 0.9850 - val_loss: 0.0816 - val_accuracy: 0.9773
Epoch 3/5
344/344 [==============================] - 14s 41ms/step - loss: 0.0576 -
accuracy: 0.9886 - val_loss: 0.0649 - val_accuracy: 0.9822
Epoch 4/5
344/344 [==============================] - 14s 41ms/step - loss: 0.0393 -
accuracy: 0.9933 - val_loss: 0.0878 - val_accuracy: 0.9736
Epoch 5/5
344/344 [==============================] - 14s 41ms/step - loss: 0.0464 -
accuracy: 0.9903 - val_loss: 0.0848 - val_accuracy: 0.9750
Accuracy: 97.49%
```

[47]:
```
# plot train and validation loss
from matplotlib import pyplot
pyplot.plot(history.history['loss'])
pyplot.plot(history.history['val_loss'])
pyplot.title('model train vs validation loss')
pyplot.ylabel('loss')
pyplot.xlabel('epoch')
pyplot.legend(['train', 'validation'], loc='upper right')
pyplot.show()
```



[48]:
```
from sklearn.metrics import confusion_matrix
y_pred1 = model1.predict(X_test)
confusion_matrix1 = confusion_matrix(y_test, np.rint(y_pred1))
print(confusion_matrix1)
from sklearn.metrics import classification_report
target_names = ['fake 0', 'real 1']
print(classification_report(y_test, np.rint(y_pred1),target_names=target_names))
```

```
[[6933  161]
 [ 177 6199]]
              precision    recall  f1-score   support

      fake 0       0.98      0.98      0.98      7094
      real 1       0.97      0.97      0.97      6376
```

```
    accuracy                           0.97     13470
   macro avg       0.97      0.97      0.97     13470
weighted avg       0.97      0.97      0.97     13470
```

[49]:
```
from lime.lime_text import LimeTextExplainer
class_names=['0','1']
explainer= LimeTextExplainer(class_names=class_names)
def predict_proba(arr):
  processed=[]
  for i in arr:
    processed.append(preproc(i))
  list_tokenized_ex = tokenizer.texts_to_sequences(processed)
  Ex = pad_sequences(list_tokenized_ex, maxlen=max_length)
  pred=model.predict(Ex)
  returnable=[]
  for i in pred:
    temp=i[0]
    returnable.append(np.array([1-temp,temp])) #I would recommend rounding temp␣
 ↪and 1-temp off to 2 places
  return np.array(returnable)
```

[50]:
```
from lime.lime_text import LimeTextExplainer
class_names=['0','1']
explainer1= LimeTextExplainer(class_names=class_names)
def predict_proba1(arr):
  processed=[]
  for i in arr:
    processed.append(preproc(i))
  list_tokenized_ex = tokenizer.texts_to_sequences(processed)
  Ex = pad_sequences(list_tokenized_ex, maxlen=max_length)
  pred=model1.predict(Ex)
  returnable=[]
  for i in pred:
    temp=i[0]
    returnable.append(np.array([1-temp,temp])) #I would recommend rounding temp␣
 ↪and 1-temp off to 2 places
  return np.array(returnable)
```

[51]:
```
print("Actual",data['Labels'][769])
explainer1.explain_instance(data['combined'][669],predict_proba1).
 ↪show_in_notebook(text=True)
```

```
Actual Fake
```

```
<IPython.core.display.HTML object>
```

[53]:
```
import joblib
joblib.dump(model,'model.pkl')
joblib.dump(tokenizer,'tokenizer.pkl')
```

```
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn,
lstm_cell_layer_call_and_return_conditional_losses while saving (showing 2 of
2). These functions will not be directly callable after loading.
WARNING:absl:<keras.layers.recurrent.LSTMCell object at 0x7f52355e3e10> has the
same name 'LSTMCell' as a built-in Keras object. Consider renaming <class
'keras.layers.recurrent.LSTMCell'> to avoid naming conflicts when loading with
`tf.keras.models.load_model`. If renaming is not possible, pass the object in
the `custom_objects` parameter of the load function.
```

[53]:
```
['tokenizer.pkl']
```

[ ]:
```

```