Libraries

In [1]:
```
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string
import sklearn
nltk.download()
from nltk.corpus import brown
brown.words()
import re
import tensorflow as tf
from nltk import word_tokenize
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn import model_selection, naive_bayes, svm
from sklearn import svm
from sklearn import preprocessing
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import Dropout
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml (https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)
```

Dataset

In [2]:
```
df = pd.read_csv('C:\\Users\\Laptop inn\\Downloads\\fake-news\\train.csv')
df.head()
```

Out[2]:

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

In [3]:
```
Bal = df = pd.read_csv('C:\\Users\\Laptop inn\\Downloads\\fake-news\\train.csv')
```

Convert the data into numerical

In [4]:
```
le = preprocessing.LabelEncoder()
Bal = Bal.apply(le.fit_transform)
```

In [5]:
```
Bal.head()
```

Out[5]:

	id	title	author	text	label
0	0	7609	940	8021	1
1	1	5854	908	6297	0
2	2	18702	826	19125	1
3	3	145	1776	17464	1
4	4	8529	1498	13019	1

Label Data

In [6]:
```
X = Bal.drop(['label'], axis=1)
Y = Bal['label']
```

Classes

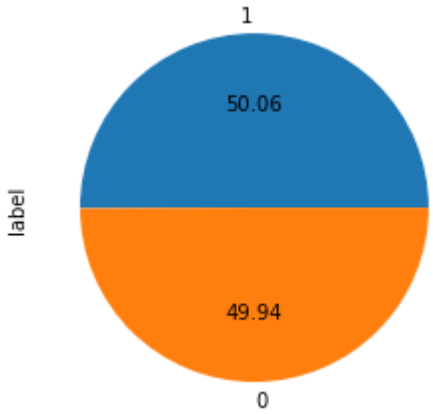
In [7]:
```
Y.value_counts()
```

Out[7]:
```
1    10413
0    10387
Name: label, dtype: int64
```

Plot shows classes are Balanced

In [8]:
```
Y.value_counts().plot.pie(autopct='%.2f')
```

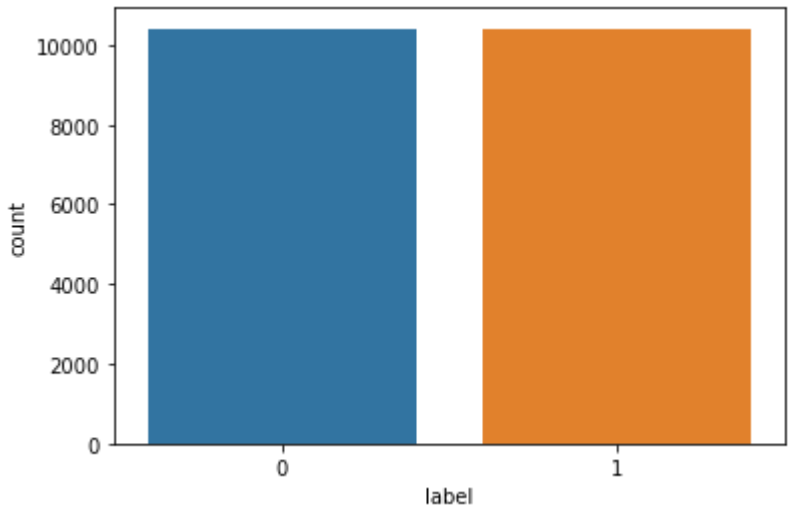
Out[8]:
```
<AxesSubplot:ylabel='label'>
```



Counter plot of classes

In [9]:
```
sns.countplot(data=Bal, x='label')
```

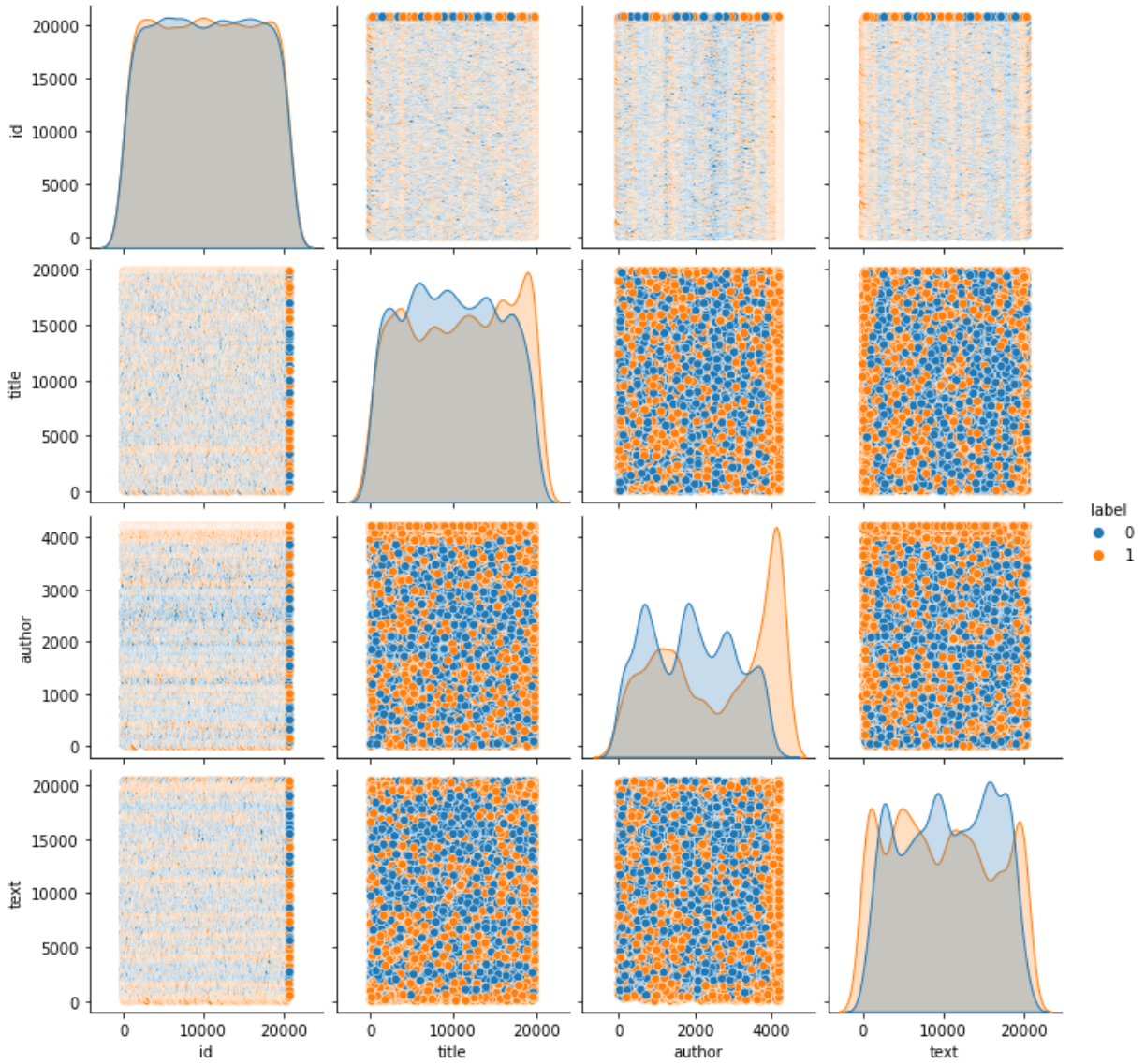
Out[9]:
```
<AxesSubplot:xlabel='label', ylabel='count'>
```



Pairplot of dataset

In [10]:
```
sns.pairplot(Bal[['id','title','author','text','label']],hue='label')
```

Out[10]: <seaborn.axisgrid.PairGrid at 0x2413b933be0>



In [11]:
```
df.shape
```

Out[11]: (20800, 5)

In [12]:
```
df.drop_duplicates(inplace= True)
df.shape
```

Out[12]: (20800, 5)

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
id           0
title      558
author    1957
text        39
label        0
dtype: int64
```

droped empty strings

In [14]:
```
df.dropna(axis=0, inplace= True)
df.shape
```

Out[14]:
```
(18285, 5)
```

Combine import columns

In [15]:
```
df['combined']= df['author'] +' '+df['title']+ df['text']
```

In [16]:
```
df['combined'].head()
```

Out[16]:
```
0    Darrell Lucus House Dem Aide: We Didn’t Even S...
1    Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
2    Consortiumnews.com Why the Truth Might Get You...
3    Jessica Purkiss 15 Civilians Killed In Single ...
4    Howard Portnoy Iranian woman jailed for fictio...
Name: combined, dtype: object
```

In [17]:
```
print(string.punctuation)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Define the function to remove the punctuation

In [18]:
```
def remove_punctuations(text):
    for punctuation in string.punctuation:
        text = text.replace(punctuation, '')
    return text

df['Removed_Puntuations'] = df['combined'].apply(remove_punctuations)
```

In [19]:
```
df['Removed_Puntuations']
```

Out[19]:
```
0        Darrell Lucus House Dem Aide We Didn’t Even Se...
1        Daniel J Flynn FLYNN Hillary Clinton Big Woman...
2        Consortiumnewscom Why the Truth Might Get You ...
3        Jessica Purkiss 15 Civilians Killed In Single ...
4        Howard Portnoy Iranian woman jailed for fictio...
                              ...
20795    Jerome Hudson Rapper TI Trump a ’Poster Child ...
20796    Benjamin Hoffman NFL Playoffs Schedule Matchup...
20797    Michael J de la Merced and Rachel Abrams Macy’...
20798    Alex Ansary NATO Russia To Hold Parallel Exerc...
20799    David Swanson What Keeps the F35 Alive  David ...
Name: Removed_Puntuations, Length: 18285, dtype: object
```

Tokenized dataset

In [20]:
```
def custom_tokenize(text):
    if not text:
        print('The text to be tokenized is a None type. Defaulting to blank strir
        text = ''
    return word_tokenize(text)

df['data_tokenize'] = df['Removed_Puntuations'].str.lower().apply(word_tokenize)
```

In [21]:
```
df['data_tokenize']
```

Out[21]:
```
0        [darrell, lucus, house, dem, aide, we, didn, ’...
1        [daniel, j, flynn, flynn, hillary, clinton, bi...
2        [consortiumnewscom, why, the, truth, might, ge...
3        [jessica, purkiss, 15, civilians, killed, in, ...
4        [howard, portnoy, iranian, woman, jailed, for,...
                              ...
20795    [jerome, hudson, rapper, ti, trump, a, ’, post...
20796    [benjamin, hoffman, nfl, playoffs, schedule, m...
20797    [michael, j, de, la, merced, and, rachel, abra...
20798    [alex, ansary, nato, russia, to, hold, paralle...
20799    [david, swanson, what, keeps, the, f35, alive,...
Name: data_tokenize, Length: 18285, dtype: object
```

Removed stopword

In [22]:
```
import nltk
stopwords = nltk.corpus.stopwords.words('english')
```

In [23]:
```
def remove_stopwords(txt_tokenized):
    txt_clean = [word for word in txt_tokenized if word not in stopwords]
    return txt_clean

df['Removed_stopwords'] = df['data_tokenize'].apply(lambda x: remove_stopwords(x)
```

In [24]:
```
df['Removed_stopwords']
```

Out[24]:
```
0        [darrell, lucus, house, dem, aide, ’, even, se...
1        [daniel, j, flynn, flynn, hillary, clinton, bi...
2        [consortiumnewscom, truth, might, get, firedwh...
3        [jessica, purkiss, 15, civilians, killed, sing...
4        [howard, portnoy, iranian, woman, jailed, fict...
                              ...
20795    [jerome, hudson, rapper, ti, trump, ’, poster,...
20796    [benjamin, hoffman, nfl, playoffs, schedule, m...
20797    [michael, j, de, la, merced, rachel, abrams, m...
20798    [alex, ansary, nato, russia, hold, parallel, e...
20799    [david, swanson, keeps, f35, alive, david, swa...
Name: Removed_stopwords, Length: 18285, dtype: object
```

In [25]:
```
# visualize the frequent words
#all_words = " ".join([sentence for sentence in df['Removed_stopwords']])
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd
all_words = ','.join(str(v) for v in df['Removed_stopwords'])
wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).

# plot the graph
plt.figure(figsize=(15, 9))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Separating the data & label

In [26]:
```
X = df.drop(columns='label', axis=1)
y = df['label']
```

Stemm or lemmatization use for normalize text data

In [27]:
```
port_stem = PorterStemmer()
```

In [28]:
```
def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]',' ',content)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [port_stem.stem(word) for word in stemmed_content if not wo
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content
```

In [29]:
```
df['after_stemming']=df['Removed_stopwords'].apply(str)
```

In [30]:
```
df['after_stemming']
```

Out[30]:
```
0        ['darrell', 'lucus', 'house', 'dem', 'aide', '...
1        ['daniel', 'j', 'flynn', 'flynn', 'hillary', '...
2        ['consortiumnewscom', 'truth', 'might', 'get',...
3        ['jessica', 'purkiss', '15', 'civilians', 'kil...
4        ['howard', 'portnoy', 'iranian', 'woman', 'jai...
                              ...
20795    ['jerome', 'hudson', 'rapper', 'ti', 'trump', ...
20796    ['benjamin', 'hoffman', 'nfl', 'playoffs', 'sc...
20797    ['michael', 'j', 'de', 'la', 'merced', 'rachel...
20798    ['alex', 'ansary', 'nato', 'russia', 'hold', '...
20799    ['david', 'swanson', 'keeps', 'f35', 'alive', ...
Name: after_stemming, Length: 18285, dtype: object
```

Feature Extraction

In [31]:
```
X =  df['after_stemming'].values
y = df['label'].values
```

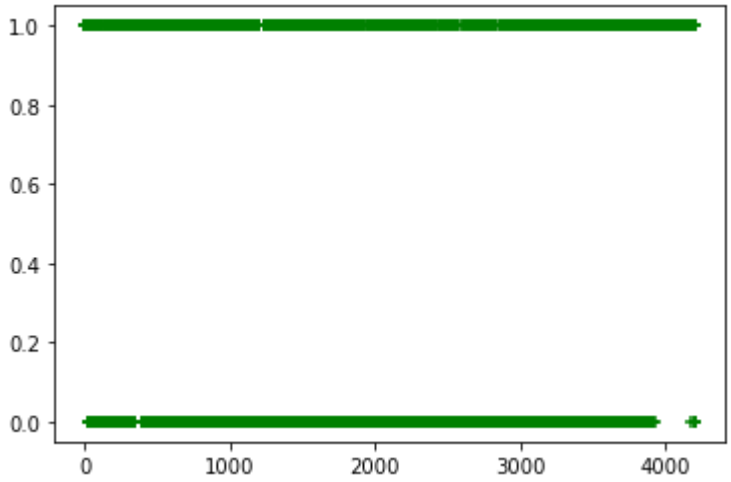
In [32]:
```
vectorizer = TfidfVectorizer()
vectorizer.fit(X)

X = vectorizer.transform(X)
```

In [33]:
```
print(X)
```

```
  (0, 174982)   0.04702508794926275
  (0, 174894)   0.010737513593484611
  (0, 172747)   0.04237181555679773
  (0, 172734)   0.06839775508466542
  (0, 172603)   0.036069571145668534
  (0, 172516)   0.013181356141822318
  (0, 171771)   0.024812039575453446
  (0, 170821)   0.03785009769123916
  (0, 170431)   0.016255525143627488
  (0, 170212)   0.012228133224596408
  (0, 170084)   0.028588071514040057
  (0, 169963)   0.013596793945404946
  (0, 169668)   0.011700758835554213
  (0, 168427)   0.027186186894261576
  (0, 168391)   0.020950363670322228
  (0, 168307)   0.016953845057030847
  (0, 166924)   0.02069813604176485
  (0, 165220)   0.06282885905786612
  (0, 164065)   0.033476174039273165
  (0, 163724)   0.016701412932223082
  (0, 162534)   0.03880242992319804
  (0, 161634)   0.010538871702038745
  (0, 161605)   0.04132937808352647
  (0, 161485)   0.08168057470268444
  (0, 161480)   0.07893382388208524
  :     :
  (18284, 9730) 0.025892466745593597
  (18284, 9723) 0.026146900611270056
  (18284, 9715) 0.02715186392290396
  (18284, 9628) 0.01543098088717602
  (18284, 9547) 0.03809663217859168
  (18284, 8205) 0.017880683068178402
  (18284, 8005) 0.03897715071969432
  (18284, 7961) 0.018776888089097
  (18284, 7844) 0.036769834639660395
  (18284, 7420) 0.022693548433666406
  (18284, 7049) 0.013313690963402236
  (18284, 7031) 0.016765281829091674
  (18284, 6512) 0.04022706558171179
  (18284, 6324) 0.03860911144175071
  (18284, 6249) 0.044070590629630854
  (18284, 4788) 0.03066148787179068
  (18284, 3748) 0.019103375339364584
  (18284, 3638) 0.014627274824988561
  (18284, 2693) 0.020165326655588883
  (18284, 2679) 0.026762354757698125
  (18284, 2569) 0.021029650707242126
  (18284, 1404) 0.03483085679682759
  (18284, 1344) 0.044070590629630854
  (18284, 1339) 0.03473991746001765
  (18284, 689)  0.01518053315070621
```

Logistic Regression Model

In [34]:
```
plt.scatter(Bal.author, Bal.label ,marker='+',color='green')

plt.rcParams["figure.figsize"] = (30,19)
```



In [35]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_
```

In [36]:
```
from sklearn.linear_model import LogisticRegression
```

In [37]:
```
model = LogisticRegression()
```

In [38]:
```
model.fit(X_train, y_train)
```

Out[38]:
```
LogisticRegression()
```

In [39]:
```
model.predict(X_test)
```

Out[39]:
```
array([1, 0, 0, ..., 0, 1, 1], dtype=int64)
```

In [40]:
```
y_predict = model.predict(X_test)
```

In [41]:
```
lg_prediction =  model.predict(X_test)
```

In [42]:
```
lg_y_prediction = accuracy_score(lg_prediction, y_test)
```

In [43]:
```
print(lg_prediction)
```

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

In [44]:
```
lg_train_prediction = model.predict(X_train)
lg_score_train = accuracy_score(lg_train_prediction, y_train)
```

Accuracy of Logistic Regression

In [45]:
```
print('Accuracy score of the test data : ',lg_y_prediction)
```

```
Accuracy score of the test data :  0.956089478044739
```

In [46]:
```
from sklearn.metrics import classification_report
print(classification_report(lg_prediction, y_test))
```

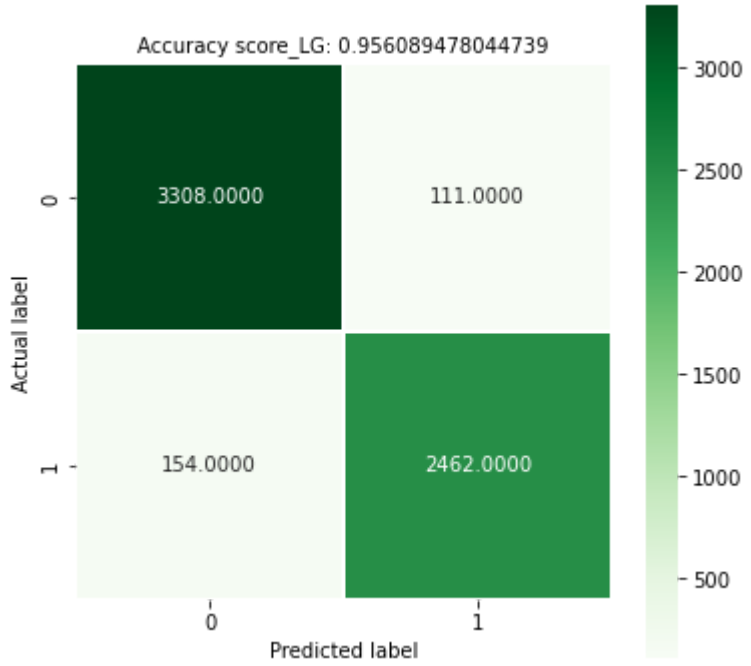
```
              precision    recall  f1-score   support

           0       0.97      0.96      0.96      3462
           1       0.94      0.96      0.95      2573

    accuracy                           0.96      6035
   macro avg       0.95      0.96      0.96      6035
weighted avg       0.96      0.96      0.96      6035
```

In [47]:
```
from sklearn.metrics import precision_score,recall_score,plot_roc_curve
```

In [48]:
```
y_predict = model.predict(X_test)
cm = metrics.confusion_matrix(y_test, y_predict)
print(cm)
```

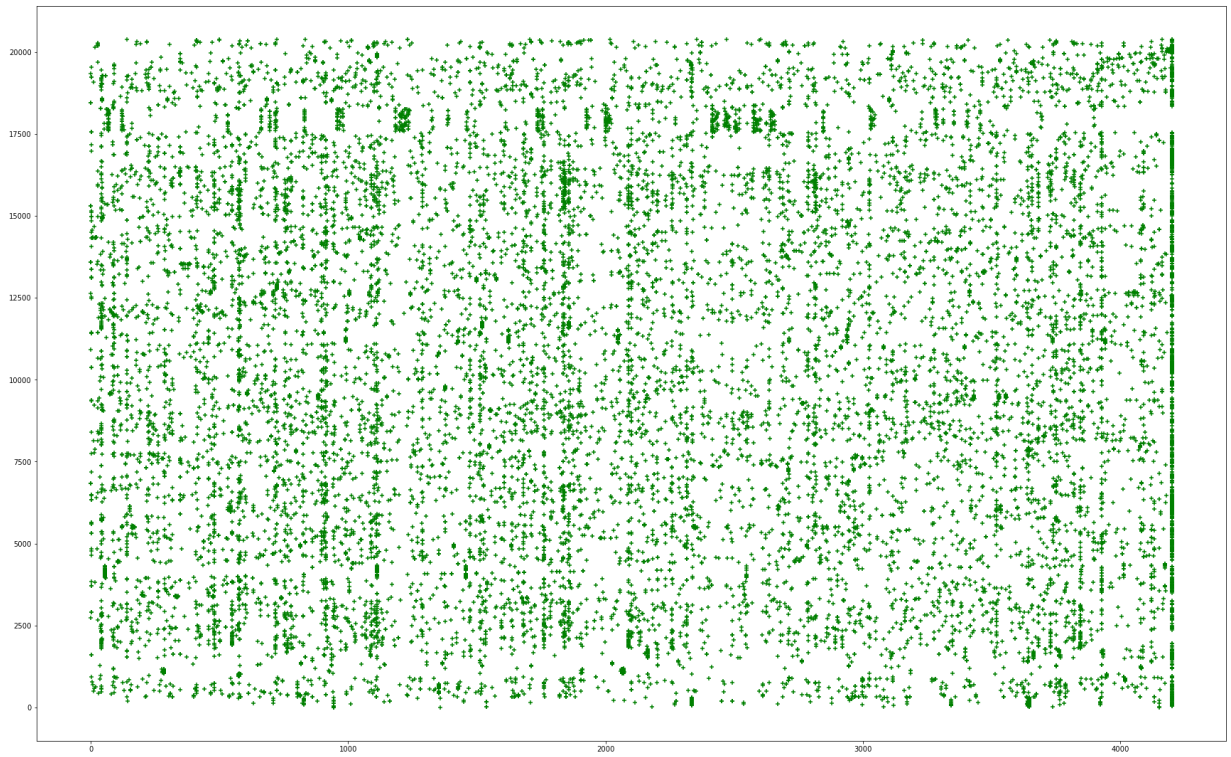
```
[[3308  111]
 [ 154 2462]]
```

In [49]:
```
plt.figure(figsize=(6,6))
sns.heatmap(cm, annot=True, fmt=".4f", linewidths=.3, square = True, cmap = 'Gree
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title ='Accuracy score_LG: {0}'.format(lg_y_prediction)
plt.title(all_sample_title, size = 10);
```



SVM classifier

In [50]:
```
plt.scatter(Bal['author'],Bal['text'],color='green',marker='+')

plt.rcParams["figure.figsize"] = (10,8)
```



In [51]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_
```

In [52]:
```
from sklearn.svm import SVC
```

In [53]:
```
classifier = svm.SVC(kernel = 'linear', gamma= 'auto', C=2)

classifier.fit(X_test, y_test)
y_predict = classifier.predict(X_test)
```

Accuracy score on the test data

In [54]:
```
svm_prediction = classifier.predict(X_test)
```

In [55]:
```

svm_y_predict = accuracy_score(svm_prediction, y_test)
```

In [56]:
```
print(svm_prediction)
```

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

In [57]:
```
svm_train_prediction = classifier.predict(X_train)
svm_score_train = accuracy_score(svm_train_prediction, y_train)
```

In [58]:
```
print('Accuracy score of the test data : ',svm_y_predict)
```

```
Accuracy score of the test data :  0.9995028997514499
```

In [59]:
```
from sklearn.metrics import classification_report
print(classification_report(svm_prediction, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3418
           1       1.00      1.00      1.00      2617

    accuracy                           1.00      6035
   macro avg       1.00      1.00      1.00      6035
weighted avg       1.00      1.00      1.00      6035
```

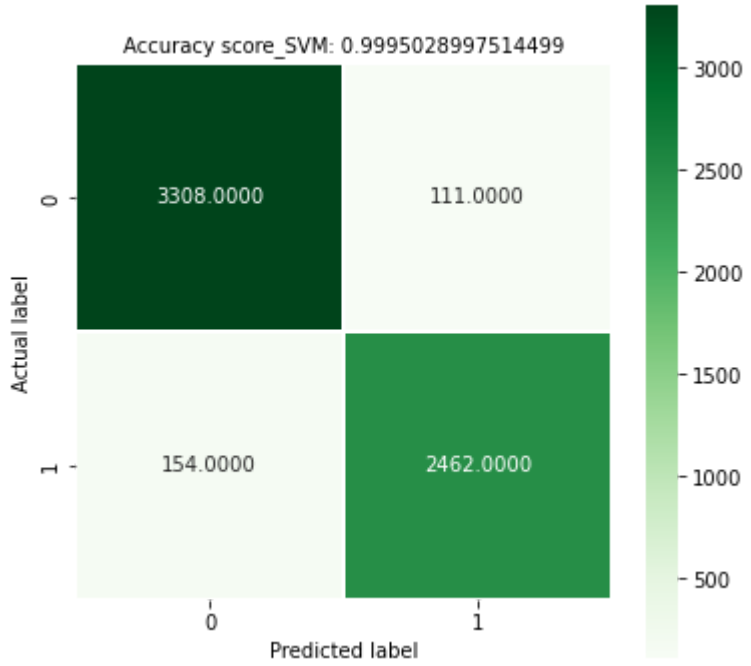
In [60]:
```
classifier = []
classifier.append(('SVM', svm.SVC()))
```

In [61]:
```
from sklearn.metrics import precision_score,recall_score,plot_roc_curve
```

In [62]:
```
y_predict = model.predict(X_test)
cm = metrics.confusion_matrix(y_test, y_predict)
print(cm)
```

```
[[3308  111]
 [ 154 2462]]
```

In [63]:
```
plt.figure(figsize=(6,6))
sns.heatmap(cm, annot=True, fmt=".4f", linewidths=.3, square = True, cmap = 'Gree
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title ='Accuracy score_SVM: {0}'.format(svm_y_predict)
plt.title(all_sample_title, size = 10);
```



Naive Bayes Classifier

In [64]:
```
#from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_
```

In [65]:
```
model = MultinomialNB()
```

In [66]:
```
model.fit(X_train, y_train)
```

Out[66]:
```
MultinomialNB()
```

In [67]:
```
NB = model.predict(X_test)
```

In [68]:
```
NB_y_predict = accuracy_score(NB, y_test)
```

In [69]:
```
NB_train_prediction = model.predict(X_train)
NB_score_train = accuracy_score(NB_train_prediction, y_train)
```

Accuracy of Naive bayes

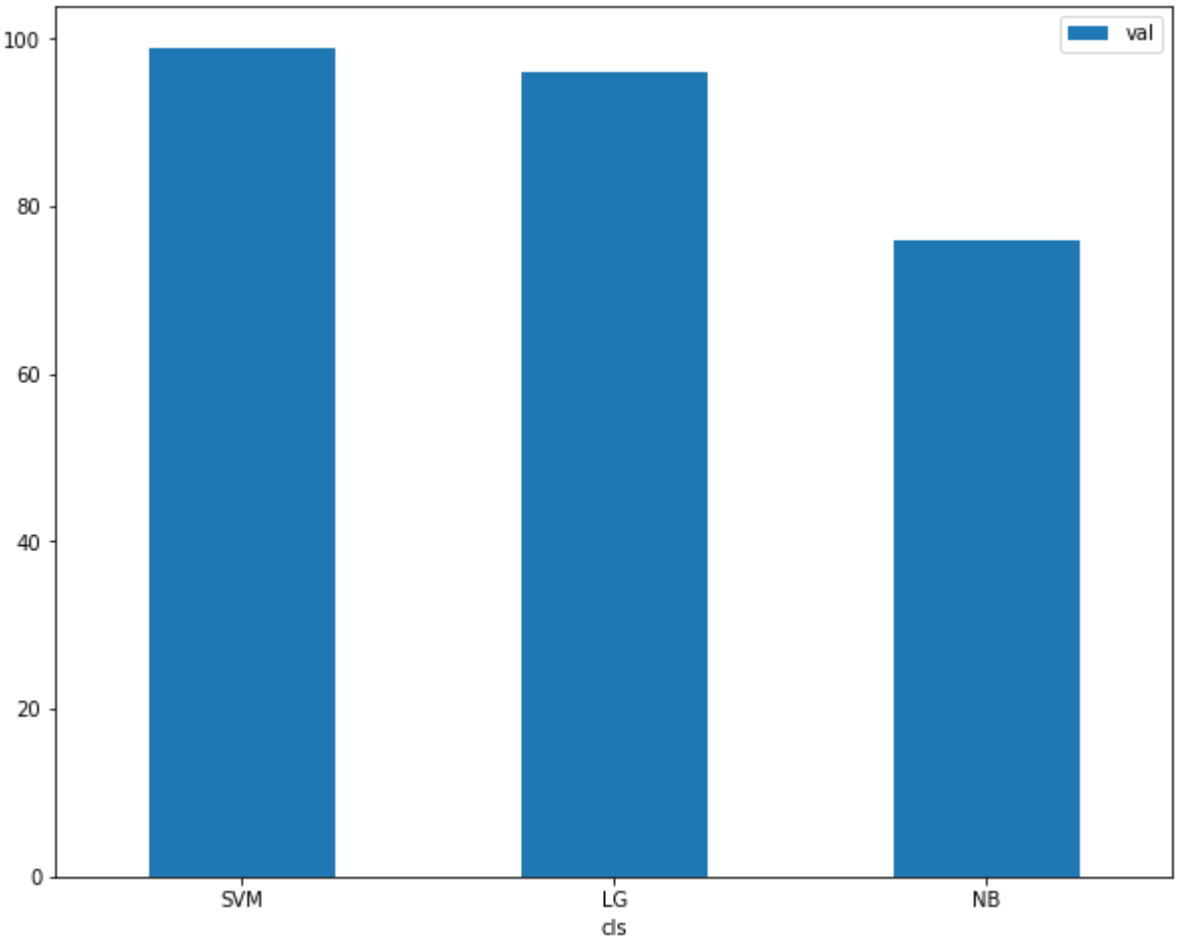
In [70]:
```
print('Accuracy score of the test data : ',NB_y_predict)
```

```
Accuracy score of the test data :  0.762373530216024
```

In [71]:
```
from sklearn.metrics import classification_report
print(classification_report(NB, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.71      0.83      2947
           1       0.45      1.00      0.62       710

    accuracy                           0.76      3657
   macro avg       0.72      0.85      0.72      3657
weighted avg       0.89      0.76      0.79      3657
```

In [72]:
```
Bal = pd.DataFrame({'cls':['SVM','LG','NB'], 'val':[99,96,76]})
ax = Bal.plot.bar(x='cls', y='val', rot=0)
```



In [73]:
```
X_new = X_test[670]
print('original =' ,Y[0])
prediction = model.predict(X_new)
print(prediction)

if (prediction[0]==0):
  print('The news is Real')
else:
  print('The news is fake')
```

```
original = 1
[0]
The news is Real
```

RNN MODEL

Important libraries

In [74]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import re
from tensorflow.keras.preprocessing.text import Tokenizer
import tensorflow as tf
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, re
import seaborn as sns
plt.style.use('ggplot')
```

In [75]:
```
real_df = pd.read_csv("C:\\Users\\Laptop inn\\Desktop\Fake.csv")
fake_df = pd.read_csv("C:\\Users\\Laptop inn\\Desktop\True.csv")
real_df.head()
```

Out[75]:

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017

In [76]:
```
fake_df.isnull().sum()
```

Out[76]:
```
title      0
text       0
subject    0
date       0
dtype: int64
```

In [77]:
```
real_df.isnull().sum()
```

Out[77]:
```
title      0
text       0
subject    0
date       0
dtype: int64
```

Checking for unique values for subject

In [78]:
```
fake_df.subject.unique()
```

Out[78]:
```
array(['politicsNews', 'worldnews'], dtype=object)
```

In [79]:
```
real_df.subject.unique()
```

Out[79]:
```
array(['News', 'politics', 'Government News', 'left-news', 'US_News',
       'Middle-east'], dtype=object)
```

Drop the date from the dataset, I don't think there is a strong correlation between date and validity of the news.

In [80]:
```
fake_df.drop(['date', 'subject'], axis=1, inplace=True)
real_df.drop(['date', 'subject'], axis=1, inplace=True)
```

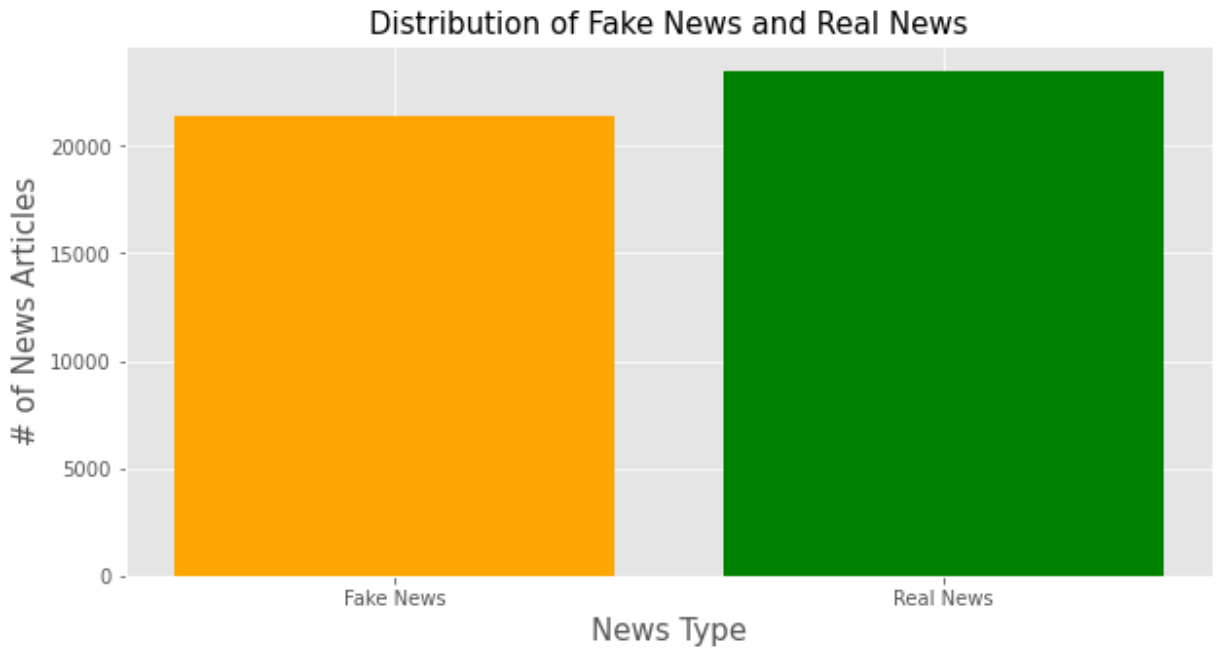
0 for fake news, and 1 for real news

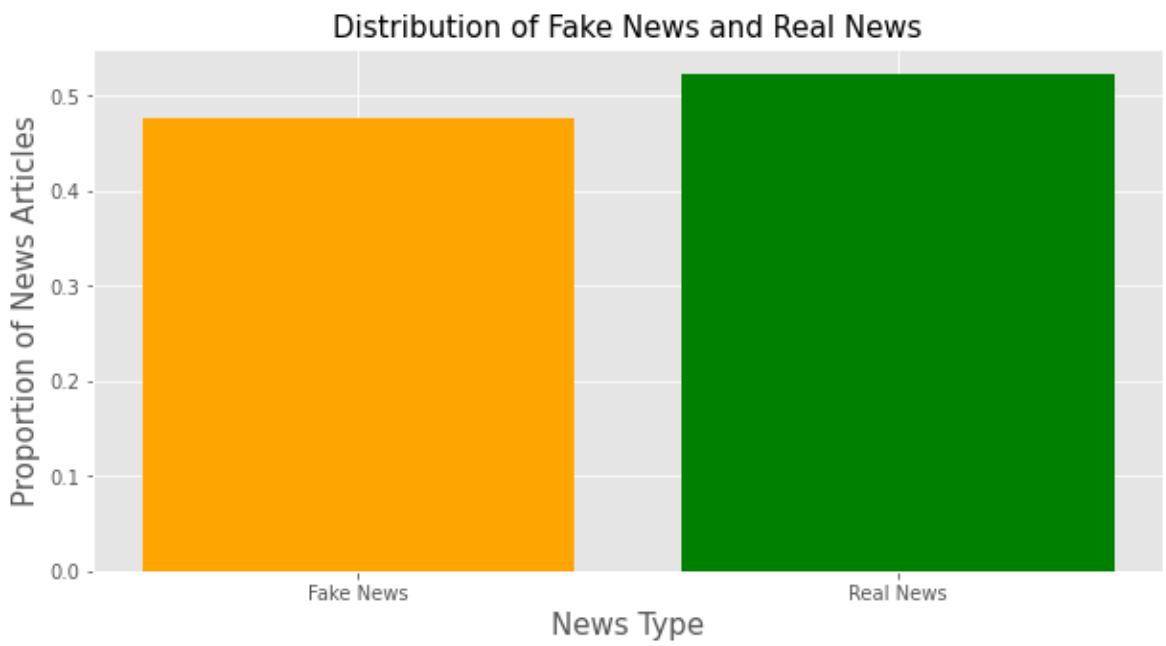
In [81]:
```
fake_df['class'] = 0
real_df['class'] = 1
```

Check out the distribution of fake news compare to real news

In [82]:
```
plt.figure(figsize=(10, 5))
plt.bar('Fake News', len(fake_df), color='orange')
plt.bar('Real News', len(real_df), color='green')
plt.title('Distribution of Fake News and Real News', size=15)
plt.xlabel('News Type', size=15)
plt.ylabel('# of News Articles', size=15)


total_len = len(fake_df) + len(real_df)
plt.figure(figsize=(10, 5))
plt.bar('Fake News', len(fake_df) / total_len, color='orange')
plt.bar('Real News', len(real_df) / total_len, color='green')
plt.title('Distribution of Fake News and Real News', size=15)
plt.xlabel('News Type', size=15)
plt.ylabel('Proportion of News Articles', size=15)
```

Out[82]:
```
Text(0, 0.5, 'Proportion of News Articles')
```





In [83]:
```
print('Difference in news articles:',len(fake_df)-len(real_df))
```

```
Difference in news articles: -2064
```

In [84]:
```
news_df = pd.concat([fake_df, real_df], ignore_index=True, sort=False)
news_df
```

Out[84]:

	title	text	class
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	0
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	0
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	0
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	0
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	0
...	...	...	...
44893	McPain: John McCain Furious That Iran Treated ...	21st Century Wire says As 21WIRE reported earl...	1
44894	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...	21st Century Wire says It s a familiar theme. ...	1
44895	Sunnistan: US and Allied ‘Safe Zone’ Plan to T...	Patrick Henningsen 21st Century WireRemember ...	1
44896	How to Blow $700 Million: Al Jazeera America F...	21st Century Wire says Al Jazeera America will...	1
44897	10 U.S. Navy Sailors Held by Iranian Military ...	21st Century Wire says As 21WIRE predicted in ...	1

44898 rows × 3 columns

Combining the title with the text, it is much easier to process this way.

In [85]:
```
news_df['text'] = news_df['title'] + news_df['text']
news_df.drop('title', axis=1, inplace=True)
```

Split into training and testing

In [86]:
```
features = news_df['text']
targets = news_df['class']

X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size=
```

Normalizing our data

In [87]:
```
def normalize(data):
    normalized = []
    for i in data:
        i = i.lower()
        # get rid of urls
        i = re.sub('https?://\S+|www\.\S+', '', i)
        # get rid of non words and extra spaces
        i = re.sub('\\W', ' ', i)
        i = re.sub('\n', '', i)
        i = re.sub(' +', ' ', i)
        i = re.sub('^ ', '', i)
        i = re.sub(' $', '', i)
        normalized.append(i)
    return normalized

X_train = normalize(X_train)
X_test = normalize(X_test)
```

In [88]:
```
max_vocab = 10000
tokenizer = Tokenizer(num_words=max_vocab)
tokenizer.fit_on_texts(X_train)
```

Convert text to vectors, our classifier only takes numerical data.

In [89]:
```
# tokenize the text into vectors
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

Apply padding so we have the same length for each article

In [90]:
```
X_train = tf.keras.preprocessing.sequence.pad_sequences(X_train, padding='post',
X_test = tf.keras.preprocessing.sequence.pad_sequences(X_test, padding='post', ma
```

Building the RNN.

In [91]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(max_vocab, 128),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,  return_sequences=True
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1)
])

model.summary()
```

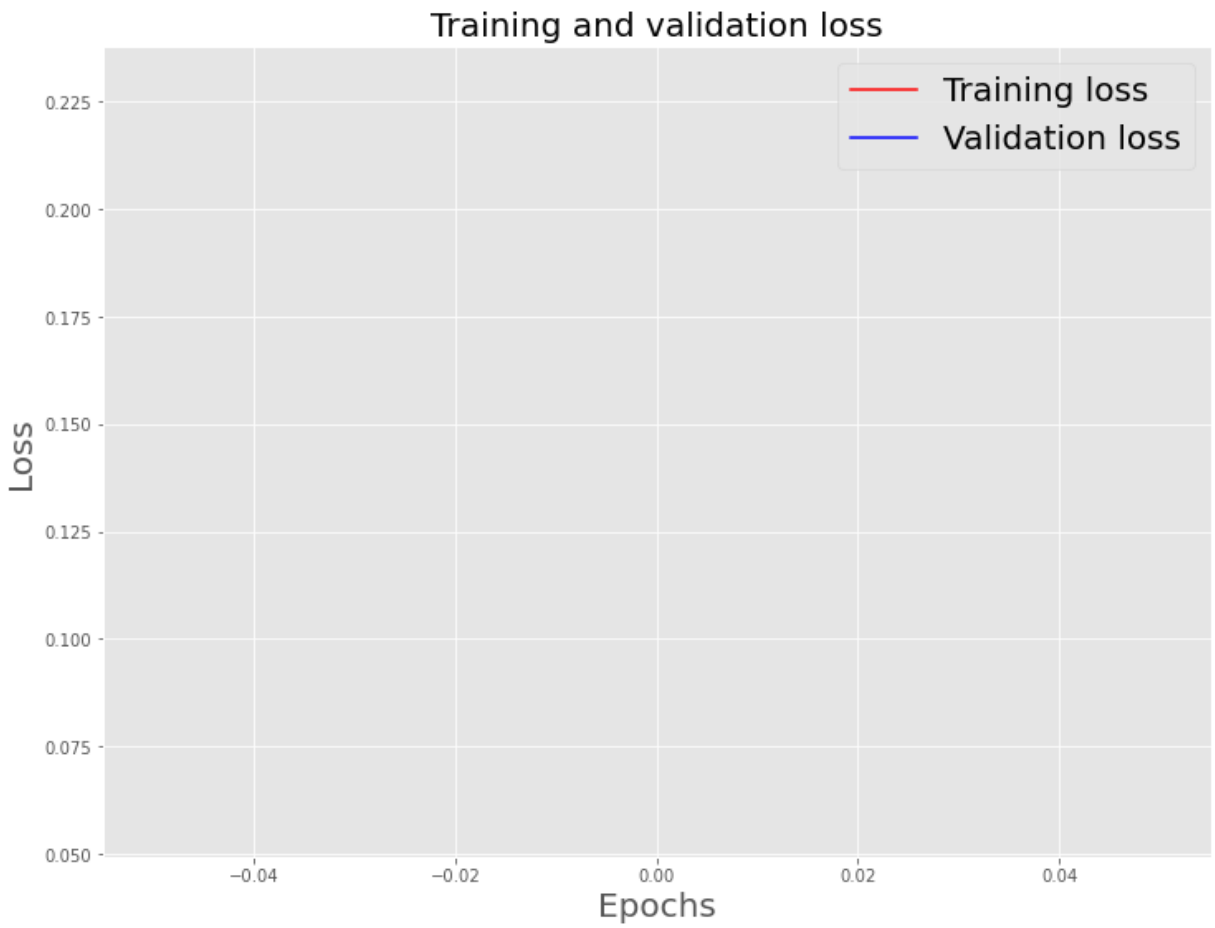
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 128)         1280000

 bidirectional (Bidirectiona  (None, None, 128)        98816
 l)

 bidirectional_1 (Bidirectio  (None, 32)               18560
 nal)

 dense (Dense)               (None, 64)                2112

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 1,399,553
Trainable params: 1,399,553
Non-trainable params: 0
_________________________________________________________________
```

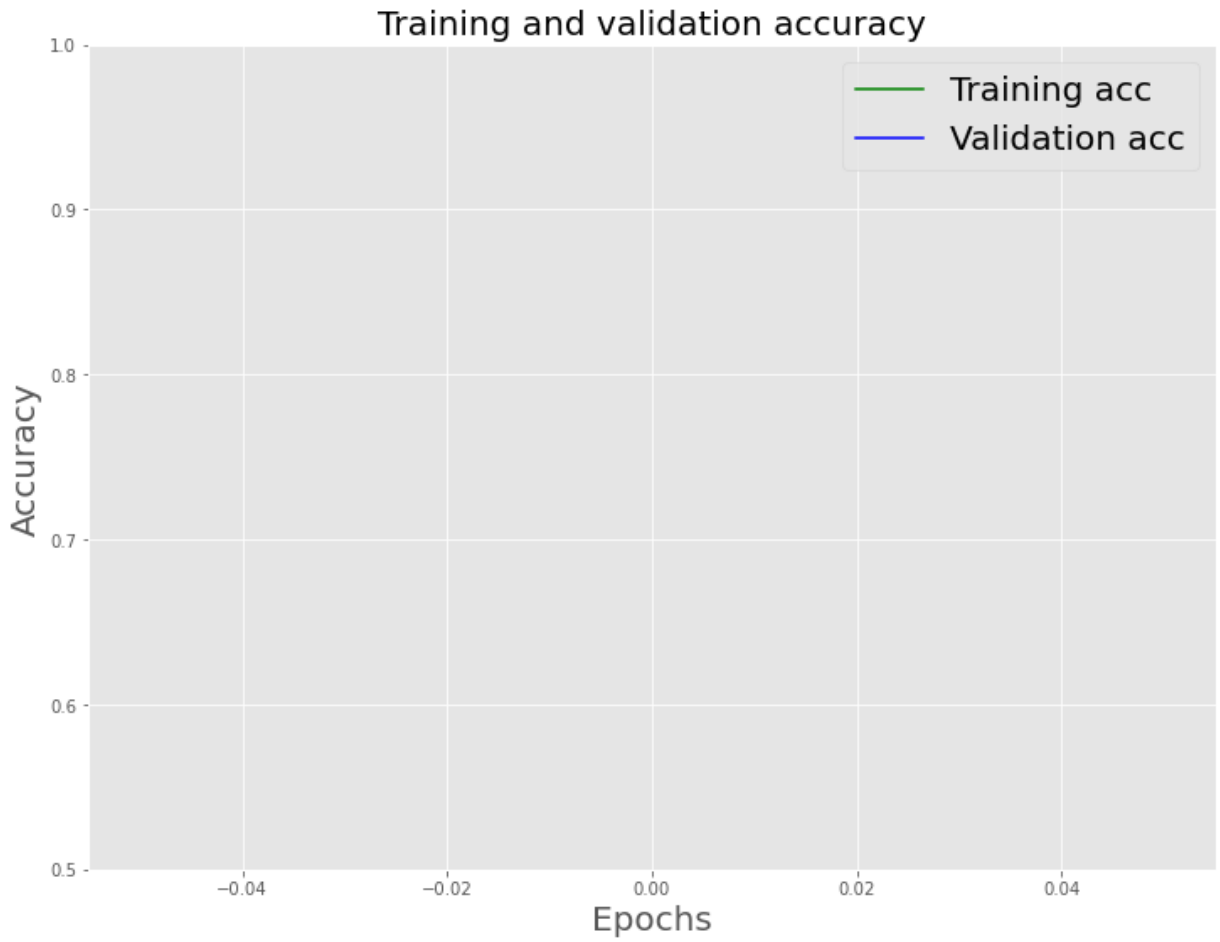
In [92]:
```
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=2, res
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=1,validation_split=0.1, batch_size=3
```

```
1078/1078 [==============================] - 551s 500ms/step - loss: 0.2294 - a
ccuracy: 0.8720 - val_loss: 0.0579 - val_accuracy: 0.9864
```

Visualize our training over time

In [93]:
```
history_dict = history.history

acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']
loss = history_dict['loss']
val_loss = history_dict['val_loss']
epochs = history.epoch

plt.figure(figsize=(12,9))
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss', size=20)
plt.xlabel('Epochs', size=20)
plt.ylabel('Loss', size=20)
plt.legend(prop={'size': 20})
plt.show()

plt.figure(figsize=(12,9))
plt.plot(epochs, acc, 'g', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy', size=20)
plt.xlabel('Epochs', size=20)
plt.ylabel('Accuracy', size=20)
plt.legend(prop={'size': 20})
plt.ylim((0.5,1))
plt.show()
```





Evaluate the testing set

In [94]:
```
model.evaluate(X_test, y_test)
```

```
281/281 [==============================] - 32s 99ms/step - loss: 0.0707 - accur
acy: 0.9792
```

Out[94]:
```
[0.07074250280857086, 0.9791759252548218]
```

In [95]:
```
pred = model.predict(X_test)

binary_predictions = []

for i in pred:
    if i >= 0.5:
        binary_predictions.append(1)
    else:
        binary_predictions.append(0)
```

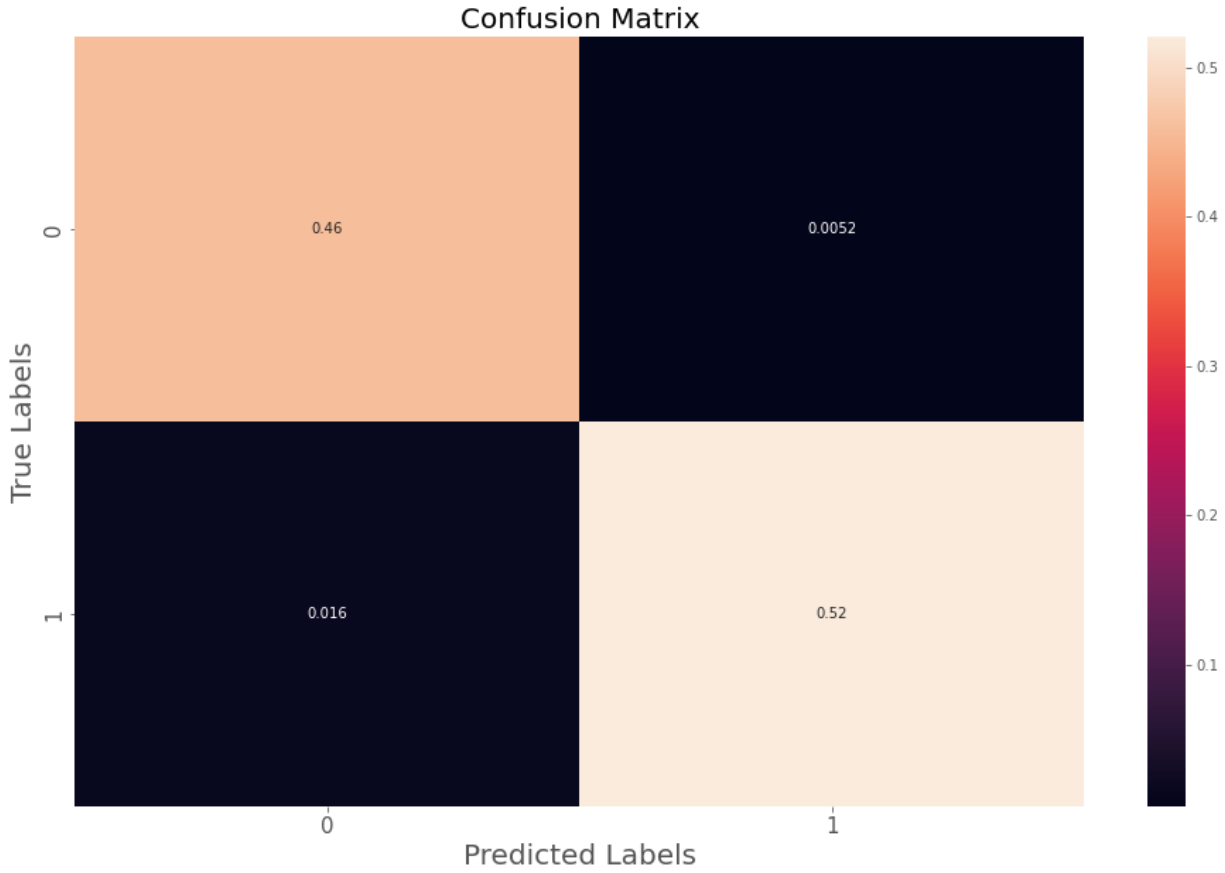
In [96]:
```
print('Accuracy on testing set:', accuracy_score(binary_predictions, y_test))
print('Precision on testing set:', precision_score(binary_predictions, y_test))
print('Recall on testing set:', recall_score(binary_predictions, y_test))
```

```
Accuracy on testing set: 0.9791759465478842
Precision on testing set: 0.9900465904277849
Recall on testing set: 0.9709241952232607
```

Confusion matrix

In [97]:
```
matrix = confusion_matrix(binary_predictions, y_test, normalize='all')
plt.figure(figsize=(16, 10))
ax= plt.subplot()
sns.heatmap(matrix, annot=True, ax = ax)

# labels, title and ticks
ax.set_xlabel('Predicted Labels', size=20)
ax.set_ylabel('True Labels', size=20)
ax.set_title('Confusion Matrix', size=20)
ax.xaxis.set_ticklabels([0,1], size=15)
ax.yaxis.set_ticklabels([0,1], size=15)
```

Out[97]:
```
[Text(0, 0.5, '0'), Text(0, 1.5, '1')]
```



LSTM Model

In [98]:
```
import pandas as pd
```

In [99]:
```
df = pd.read_csv('C:\\Users\\Laptop inn\\Downloads\\fake-news\\train.csv')
df.head()
```

Out[99]:

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

In [100]:
```
###Drop Nan Values
df=df.dropna()
```

In [101]:
```
## Get the Independent Features

X=df.drop('label',axis=1)
```

In [102]:
```
## Get the Dependent features
y=df['label']
```

In [103]:
```
X.shape
```

Out[103]: (18285, 4)

In [104]:
```
y.shape
```

Out[104]: (18285,)

In [105]:
```
import tensorflow as tf
```

In [106]:
```
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
```

In [107]:
```
### Vocabulary size
voc_size=5000
```

Onehot Representation

In [108]:
```
messages=X.copy()
```

In [109]:
```
messages['title'][1]
```

Out[109]:
```
'FLYNN: Hillary Clinton, Big Woman on Campus - Breitbart'
```

In [110]:
```
messages.reset_index(inplace=True)
```

In [111]:
```
import nltk
import re
from nltk.corpus import stopwords
```

In [112]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:\Users\Laptop
[nltk_data]     inn\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[112]:
```
True
```

In [113]:
```
### Dataset Preprocessing
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
corpus = []
for i in range(0, len(messages)):
    print(i)
    review = re.sub('[^a-zA-Z]', ' ', messages['title'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('er
    review = ' '.join(review)
    corpus.append(review)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

In [114]:
```
corpus
```

Out[114]:
```
['hous dem aid even see comey letter jason chaffetz tweet',
 'flynn hillari clinton big woman campu breitbart',
 'truth might get fire',
 'civilian kill singl us airstrik identifi',
 'iranian woman jail fiction unpublish stori woman stone death adulteri',
 'jacki mason hollywood would love trump bomb north korea lack tran bathroom
exclus video breitbart',
 'beno hamon win french socialist parti presidenti nomin new york time',
 'back channel plan ukrain russia courtesi trump associ new york time',
 'obama organ action partner soro link indivis disrupt trump agenda',
 'bbc comedi sketch real housew isi caus outrag',
 'russian research discov secret nazi militari base treasur hunter arctic ph
oto',
 'us offici see link trump russia',
 'ye paid govern troll social media blog forum websit',
 'major leagu soccer argentin find home success new york time',
 'well fargo chief abruptli step new york time',
 'anonym donor pay million releas everyon arrest dakota access pipelin',
 'fbi close hillari',
```

In [115]:
```
onehot_repr=[one_hot(words,voc_size)for words in corpus]
# onehot_repr
```

Embedding Representation

In [116]:
```
sent_length=20
embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)
print(embedded_docs)
```

```
[[   0    0    0 ... 1337 4962 4801]
 [   0    0    0 ... 3439 2894 4005]
 [   0    0    0 ... 4590 4727 4641]
 ...
 [   0    0    0 ... 2569 3720 3661]
 [   0    0    0 ... 2137 2722 2906]
 [   0    0    0 ...  162 4650 4136]]
```

In [117]:
```
embedded_docs[0]
```

Out[117]:
```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,   33,
        123, 4816, 2527, 1617, 3954, 2109, 1337, 4962, 4801])
```

Creating model

In [118]:
```
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(LSTM(100))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 20, 40)            200000

 lstm_2 (LSTM)               (None, 100)               56400

 dense_2 (Dense)             (None, 1)                 101

=================================================================
Total params: 256,501
Trainable params: 256,501
Non-trainable params: 0
_________________________________________________________________
None
```

In [119]:
```
len(embedded_docs),y.shape
```

Out[119]:
```
(18285, (18285,))
```

In [120]:
```
import numpy as np
X_final=np.array(embedded_docs)
y_final=np.array(y)
```

In [121]:
```
X_final.shape,y_final.shape
```

Out[121]:
```
((18285, 20), (18285,))
```

In [122]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_final, y_final, test_size=0
```

Model Training

In [137]:
```
### Finally Training
history=model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=3,batch_
```

```
Epoch 1/3
192/192 [==============================] - 16s 57ms/step - loss: 0.0753 - accur
acy: 0.9748 - val_loss: 0.2603 - val_accuracy: 0.9127
Epoch 2/3
192/192 [==============================] - 11s 56ms/step - loss: 0.0582 - accur
acy: 0.9806 - val_loss: 0.2513 - val_accuracy: 0.9175
Epoch 3/3
192/192 [==============================] - 10s 52ms/step - loss: 0.0455 - accur
acy: 0.9847 - val_loss: 0.3106 - val_accuracy: 0.9178
```

Performance Metrics And Accuracy

In [132]:
```
#y_pred=model.predict_classes(X_test)
#predict_x=model.predict(X_test)
#classes_x=np.argmax(predict_x,axis=1)
#predictions = model.predict_classes(x_test)
y_pred = (model.predict(X_test) > 0.5).astype("int32")
```

In [133]:
```
from sklearn.metrics import confusion_matrix
```

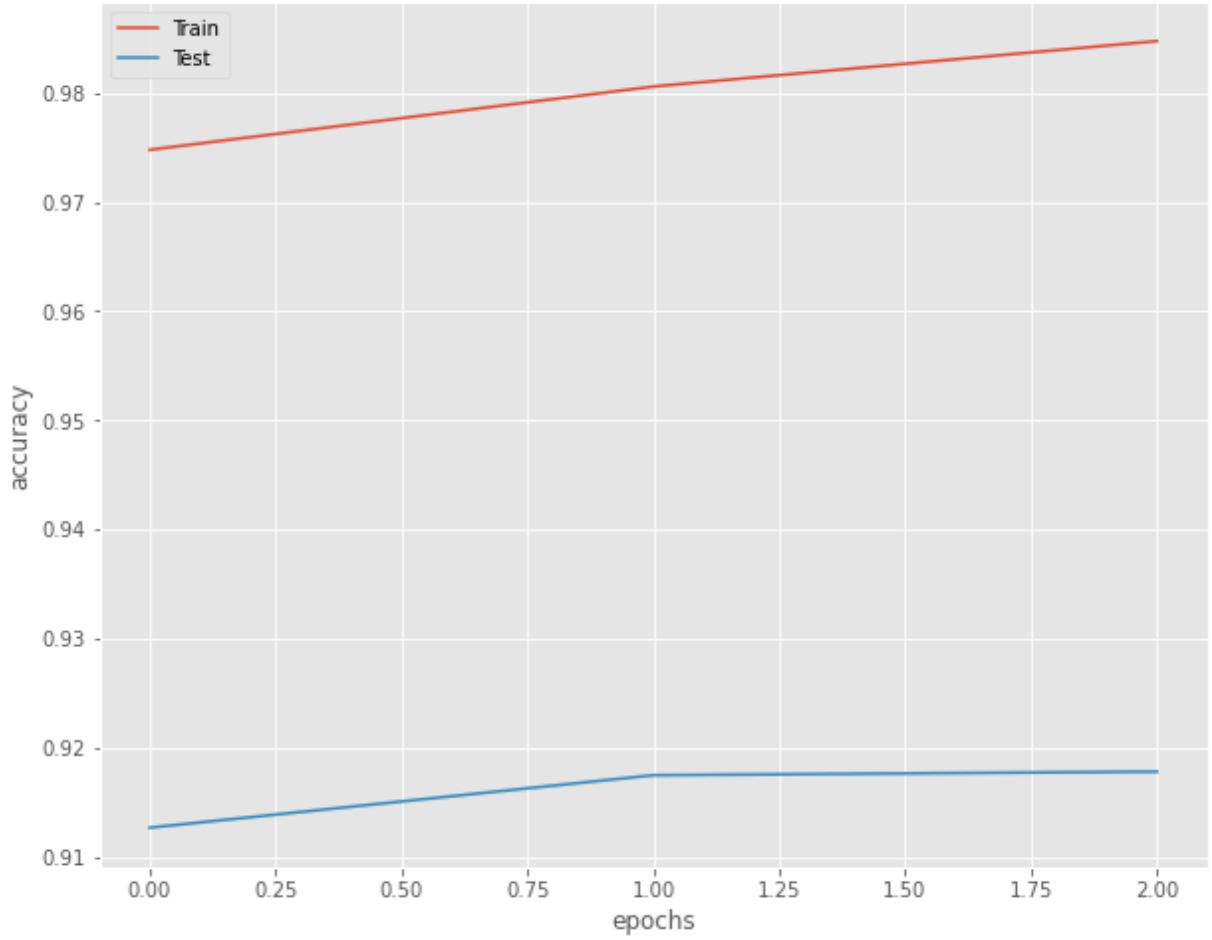
In [134]:
```
confusion_matrix(y_test,y_pred)
```

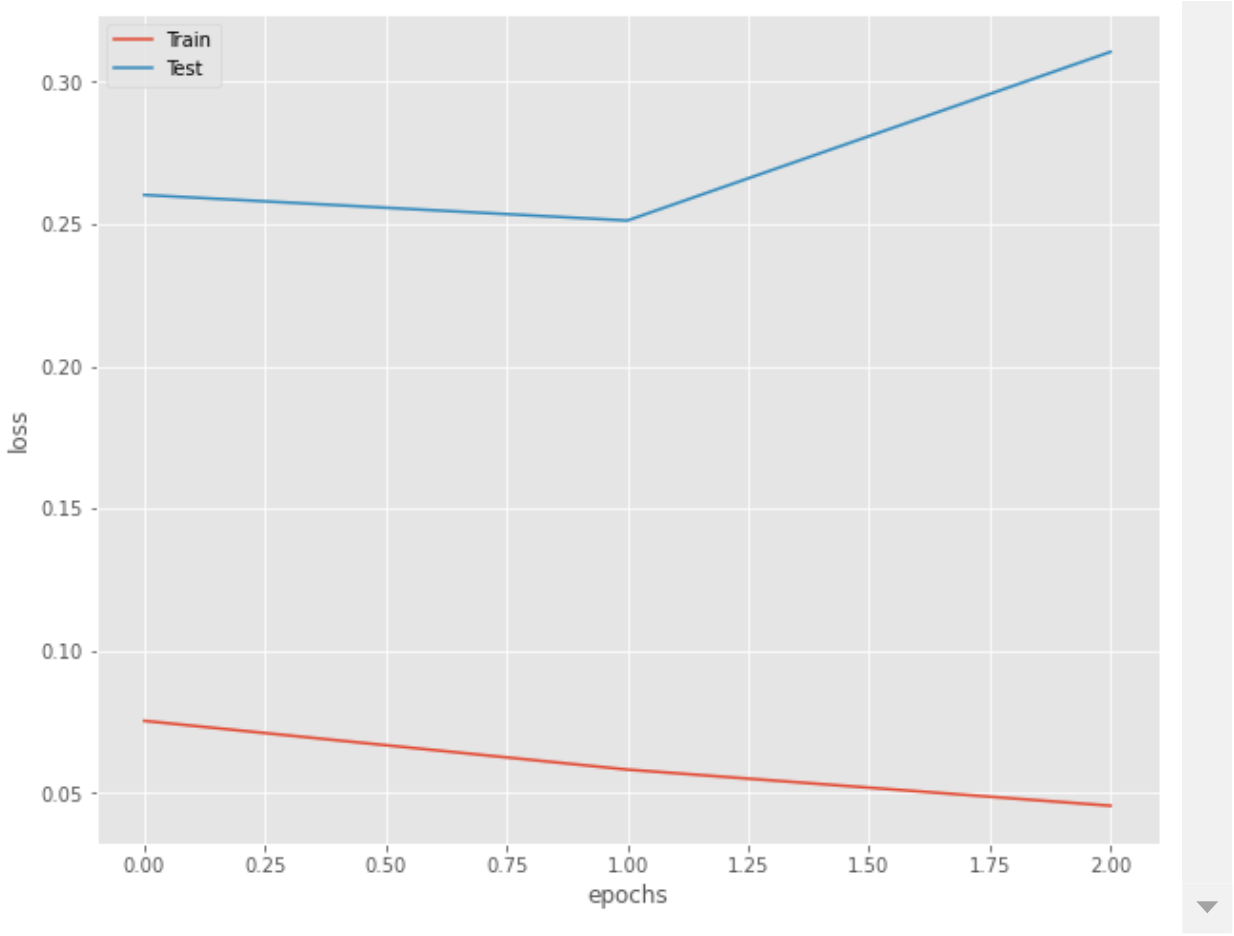
Out[134]:
```
array([[3143,  276],
       [ 208, 2408]], dtype=int64)
```

In [136]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[136]:
```
0.91980115990058
```

In [138]:
```
# visualize the results
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend(['Train', 'Test'])
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend(['Train', 'Test'])
plt.show()
```




Train
Test
0.30
0.25
0.20
0.15
0.10
0.05
loss
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
epochs

In [ ]: