Deep Learning Techniques

Mini Project - 3

Subject Code: CS6005

Fake and Real News Classification

Name: Jerrick Gerald

Batch: N

Regno: 2018103031

Introduction:

News media has become a channel to pass on the information of what's happening in the world to the people living. Often people perceive whatever conveyed in the news to be true. There were circumstances where even the news channels acknowledged that their news is not true as they wrote. But some news has a significant impact not only on the people or government but also on the economy. One news can shift the curves up and down depending on the emotions of people and political situation.

It is important to identify the fake news from the real true news. The problem has been taken over and resolved with the help of Natural Language Processing tools which help us identify fake or true news based on historical data.

Problem Statement:

The authenticity of Information has become a longstanding issue affecting businesses and society, both for printed and digital media. On social networks, the reach and effects of information spread occur at such a fast pace and so amplified that distorted, inaccurate or false information acquires a tremendous potential to cause real world impacts, within minutes, for millions of users. Recently, several public concerns about this problem and some approaches to mitigate the problem were expressed.. The sensationalism of not-so-accurate eye catching and intriguing headlines aimed at retaining the attention of audiences to sell information has persisted all throughout the history of all kinds of information broadcast. On social networking websites, the reach and effects of information spread are however significantly amplified and occur at such a fast pace, that distorted, inaccurate or false information acquires a tremendous potential to cause real impacts, within minutes, for millions of users.

Dataset: This metadata has two csv files where one dataset contains fake news and the other contains true/real news and has nearly 23481 **fake news and 21417 true news**

1. title- contains news headlines
2. text-contains news content/article
3. subject- type of news

Importing Libraries:

[1]:
```
#Basic libraries
import pandas as pd
import numpy as np

#Visualization libraries
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sns
from textblob import TextBlob
from plotly import tools


#NLTK libraries
import nltk
import re
import string
from nltk.corpus import stopwords
from wordcloud import WordCloud,STOPWORDS
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split


#Metrics libraries
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

from collections import Counter
```

Importing Dataset:

```
fake_news = pd.read_csv('../input/fake-and-real-news-dataset/Fake.csv')
true_news = pd.read_csv('../input/fake-and-real-news-dataset/True.csv')
print ("The shape of the  data is (row, column):"+ str(fake_news.shape))
print (fake_news.info())
print("\n ---------------------------------------- \n")
print ("The shape of the  data is (row, column):"+ str(true_news.shape))
print (true_news.info())
```

```
The shape of the  data is (row, column):(23481, 4)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23481 entries, 0 to 23480
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   title    23481 non-null  object
 1   text     23481 non-null  object
 2   subject  23481 non-null  object
 3   date     23481 non-null  object
dtypes: object(4)
memory usage: 733.9+ KB
None

 ----------------------------------------

The shape of the  data is (row, column):(21417, 4)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21417 entries, 0 to 21416
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   title    21417 non-null  object
 1   text     21417 non-null  object
 2   subject  21417 non-null  object
 3   date     21417 non-null  object
dtypes: object(4)
memory usage: 669.4+ KB
None
```

```
[5]: fake_news['date'].value_counts()
```

```
Out[5]: May 10, 2017
        46
        May 5, 2016
        44
        May 26, 2016
        44
        May 6, 2016
        44
        May 11, 2016
        43

        ..
        November 12, 2017
        1
        https://100percentfedup.com/video-hillary-asked-about-trump-i-just-want-to-eat-some-pie/
        1
        https://100percentfedup.com/served-roy-moore-vietnamletter-veteran-sets-record-straight-honorable-decent-respectable-patriot
        ic-commander-soldier/     1
        14-Feb-18
        1
        December 11, 2017
        1
        Name: date, Length: 1681, dtype: int64
```

Text Processing

This is an important phase for any text analysis application. There will be many unusual contents in the news which can be an obstacle when feeding to a machine learning model. Unless we remove them, the machine learning model doesn't work efficiently. Let's go step by step.

```
clean_news=news_dataset.copy()
```

```
def review_cleaning(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

```
clean_news['news']=clean_news['news'].apply(lambda x:review_cleaning(x))
clean_news.head()
```

```
[1]:
```

	subject	date	news	output
0	News	2017-12-31	donald trump sends out embarrassing new year'...	0
1	News	2017-12-31	drunk bragging trump staffer started russian ...	0
2	News	2017-12-30	sheriff david clarke becomes an internet joke...	0
3	News	2017-12-29	trump is so obsessed he even has obama's name...	0
4	News	2017-12-25	pope francis just called out donald trump dur...	0

STOPWORD REMOVAL:

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words.

```
2]: stop = stopwords.words('english')
clean_news['news'] = clean_news['news'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
clean_news.head()
```

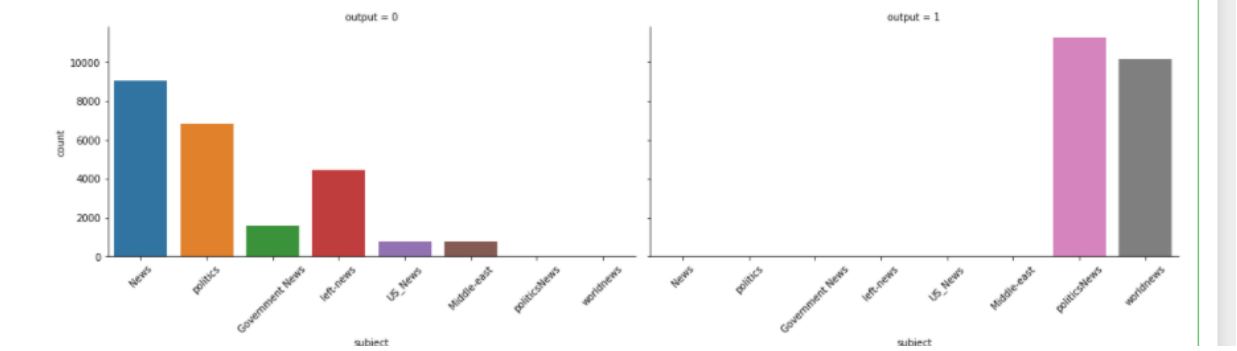
ut[12]:

	subject	date	news	output
0	News	2017-12-31	donald trump sends embarrassing new year's eve...	0
1	News	2017-12-31	drunk bragging trump staffer started russian c...	0
2	News	2017-12-30	sheriff david clarke becomes internet joke thr...	0
3	News	2017-12-29	trump obsessed even obama's name coded website...	0
4	News	2017-12-25	pope francis called donald trump christmas spe...	0

Count of news subject based on true or fake

```
g = sns.catplot(x="subject", col="output",
                data=clean_news, kind="count",
                height=4, aspect=2)
g.set_xticklabels(rotation=45)
```

14]: <seaborn.axisgrid.FacetGrid at 0x7ff7c0e53e90>



Deriving new features from the news

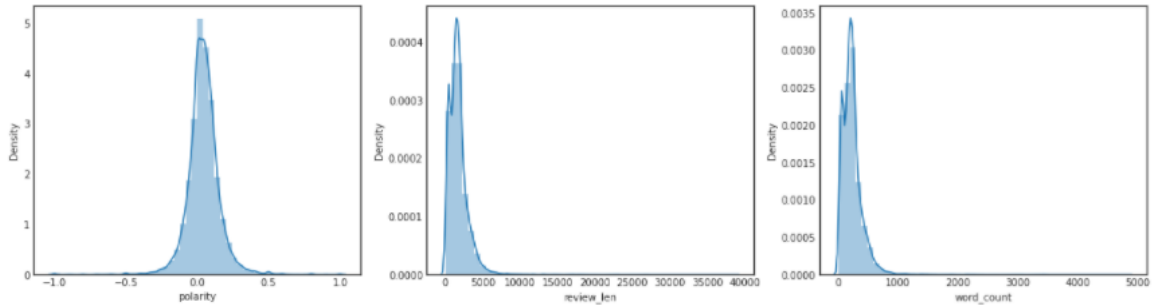
Extracting more features from the news feature such as

1. Polarity: The measure which signifies the sentiment of th news

2. Review length: Length of the news (number of letters and spaces)

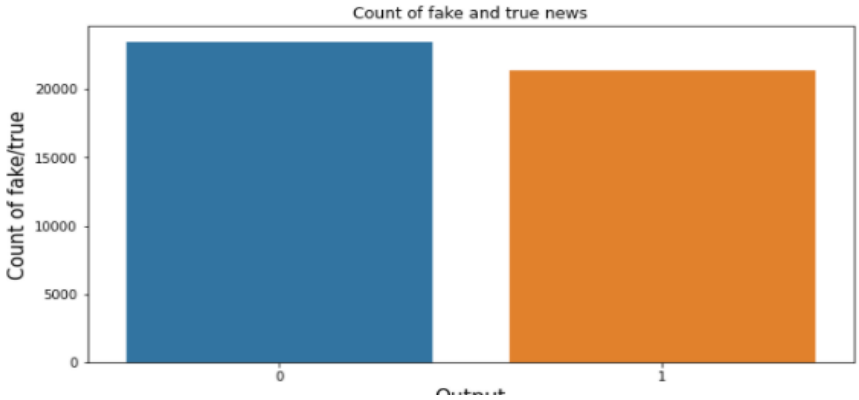
3. Word Count: Number of words in the news

```
clean_news['polarity'] = clean_news['news'].map(lambda text: TextBlob(text).sentiment.polarity)
clean_news['review_len'] = clean_news['news'].astype(str).apply(len)
clean_news['word_count'] = clean_news['news'].apply(lambda x: len(str(x).split()))
plt.figure(figsize = (20, 5))
plt.style.use('seaborn-white')
plt.subplot(131)
sns.distplot(clean_news['polarity'])
fig = plt.gcf()
plt.subplot(132)
sns.distplot(clean_news['review_len'])
fig = plt.gcf()
plt.subplot(133)
sns.distplot(clean_news['word_count'])
fig = plt.gcf()
```



Count of fake news and true news

```
ax=sns.countplot(x="output", data=clean_news)
ax.set(xlabel='Output', ylabel='Count of fake/true',title='Count of fake and true news')
ax.xaxis.get_label().set_fontsize(15)
ax.yaxis.get_label().set_fontsize(15)
```



The top 20 words from the news which could give us a brief idea on what news are popular in our dataset

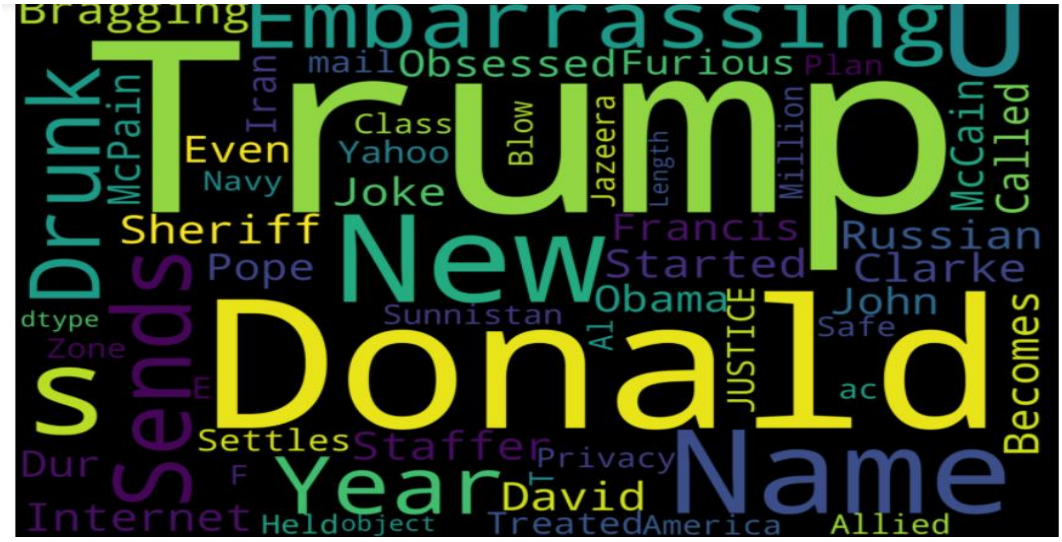
```
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer().fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

common_words = get_top_n_words(clean_news['news'], 20)

for word, freq in common_words:
    print(word, freq)
df1 = pd.DataFrame(common_words, columns = ['news' , 'count'])

#Group by words and plot the sum
df1.groupby('news').sum()['count'].sort_values(ascending=False).iplot(
    kind='bar', yTitle='Count', linecolor='black', title='Top 20 words in news')
```

```
trump 140400
said 130258
us 68081
would 55422
president 53189
people 41718
one 36146
state 33190
new 31799
also 31209
obama 29881
clinton 29003
house 28716
government 27392
donald 27376
reuters 27348
states 26331
```

WORDCLOUD IMAGES

In [20]:

```
text = fake_news["news"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



STEMMING:

Stemming is a method of deriving root word from the inflected word. Here we extract the reviews and convert the words in reviews to its root word. for example,

* Going->go

* Finally->final

```
news_features=news_features[['news']].reset_index(drop=True)
news_features.head()
```

23]:

	news
0	donald trump sends embarrassing new year's eve...
1	drunk bragging trump staffer started russian c...
2	sheriff david clarke becomes internet joke thr...
3	trump obsessed even obama's name coded website...
4	pope francis called donald trump christmas spe...

```
stop_words = set(stopwords.words("english"))

ps = PorterStemmer()

corpus = []
for i in range(0, len(news_features)):
    news = re.sub('[^a-zA-Z]', ' ', news_features['news'][i])
    news= news.lower()
    news = news.split()
    news = [ps.stem(word) for word in news if not word in stop_words]
    news = ' '.join(news)
    corpus.append(news)
```

```
corpus[1]
```

25]:

```
'drunk brag trump staffer start russian collus investigationhous intellig committe chairman devin nune go bad day assumpt li
ke mani us christoph steeledossi prompt russia investig lash depart justic fbi order protect trump happen dossier start inve
stig accord document obtain new york timesform trump campaign advis georg papadopoulo drunk wine bar reveal knowledg russian
opposit research hillari clintonon top papadopoulo covfef boy trump administr alleg much larger role none damn drunken fool
wine bar coffe boy help arrang new york meet trump presid abdel fattah elsisi egypt two month elect known former aid set mee
t world leader trump team trump ran mere coffe boyin may papadopoulo reveal australian diplomat alexand downer russian offic
i shop around possibl dirt thendemocrat presidenti nomine hillari clinton exactli much mr papadopoulo said night kensington
wine room australian alexand downer unclear report state two month later leak democrat email began appear onlin australian o
ffici pass inform mr papadopoulo american counterpart accord four current former american foreign offici direct knowledg aus
```

TFIDF (Term Frequency — Inverse Document Frequency)

TF-IDF stands for "Term Frequency — Inverse Document Frequency". This is a technique to quantify a word in documents, we generally compute a weight to each word which signifies the importance of the word in the document and corpus. This method is a widely used technique in Information Retrieval and Text Mining.

Here we are splitting as bigram (two words) and consider their combined weight. Also, we are taking only the top 5000 words from the news.

```
]: tfidf_vectorizer = TfidfVectorizer(max_features=5000,ngram_range=(2,2))
   X= tfidf_vectorizer.fit_transform(news_features['news'])
   X.shape
```

```
:[26]: (44888, 5000)
```

As we have considered 5000 words, we can confirm that we have 5000 columns from the shape.

```
]: y=clean_news['output']
```

CLASSIFIER:

```
]: logreg_cv = LogisticRegression(random_state=0)
   dt_cv=DecisionTreeClassifier()
   knn_cv=KNeighborsClassifier()
   nb_cv=MultinomialNB(alpha=0.1)
   cv_dict = {0: 'Logistic Regression', 1: 'Decision Tree',2:'KNN',3:'Naive Bayes'}
   cv_models=[logreg_cv,dt_cv,knn_cv,nb_cv]
   for i,model in enumerate(cv_models):
       print("{} Test Accuracy: {}".format(cv_dict[i],cross_val_score(model, X, y, cv=10, scoring ='accuracy').mean()))
```

```
Logistic Regression Test Accuracy: 0.9660040199274997
Decision Tree Test Accuracy: 0.9354608353396239
KNN Test Accuracy: 0.613172841991654
Naive Bayes Test Accuracy: 0.9373328405462511
```

From the results, we can see logistic regression outdone the rest of the algorithms followed by Naive Bayes and Decision Tree.

Classification report:

Considering Fake news, we should seriously consider precision score (False positive). We can't afford the mistakes when the model classifies fake news as true which is wrong.

```
]: print("Classification Report:\n",classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      5892
           1       0.98      0.98      0.98      5330

    accuracy                           0.98     11222
   macro avg       0.98      0.98      0.98     11222
weighted avg       0.98      0.98      0.98     11222
```

All our scores are 98%

ROC-AUC Curve

This is a very important curve where we decide on which threshold to setup based upon the objective criteria

```
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([-0.01, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

We should consider the AUC score here which is 98%. Very well. All metrics are performing good. The more far left the curve is better our model We can adjust our threshold based on our ROC curve to get results based on model requirements

Deep learning-LSTM

Here in this part, we use neural network to predict whether the given news is fake or not. We aren't going to use normal neural network like ANN to classify but LSTM (long short-term memory) which helps in containing sequence information. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more.

One hot for Embedding layers

While one hot encoding the words in sentences will take the index from the vocabulary size. Let's fix the vocabulary size to 10000.

```
7]: corpus[1]

t[37]: 'drunk brag trump staffer start russian collus investigationhous intellig committe chairman devin nune go bad day assumpt li
ke mani us christoph steeledossi prompt russia investig lash depart justic fbi order protect trump happen dossier start inve
stig accord document obtain new york timesform trump campaign advis georg papadopoulo drunk wine bar reveal knowledg russian
opposit research hillari clintonon top papadopoulo covfef boy trump administr alleg much larger role none damn drunken fool
wine bar coffe boy help arrang new york meet trump presid abdel fattah elsisi egypt two month elect known former aid set mee
t world leader trump team trump ran mere coffe boyin may papadopoulo reveal australian diplomat alexand downer russian offic
i shop around possibl dirt thendemocrat presidenti nomine hillari clinton exactli much mr papadopoulo said night kensington
wine room australian alexand downer unclear report state two month later leak democrat email began appear onlin australian o
ffici pass inform mr papadopoulo american counterpart accord four current former american foreign offici direct knowledg aus
tralian role papadopoulo plead guilti lie fbi cooper wit special counsel robert mueller teamthi presid badli script realiti
tv showphoto win mcnameegetti imag'

3]: voc_size=10000
onehot_repr=[one_hot(words,voc_size)for words in corpus]
```

Padding embedded documents

All the neural networks require to have inputs that have the same shape and size. However, when we pre-process and use the texts as inputs for our LSTM model, not all the sentences have the same length. In other words, naturally, some of the sentences are longer or shorter. We need to have the inputs with the same size,

this is where the padding is necessary. Here we take the common length as 5000 and perform padding using pad sequence () function. Also, we are going to 'pre' pad so that zeros are added before the sentences to make the sentence of equal length

```
[39]: sent_length=5000
      embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)
      print(embedded_docs)

      [[   0    0    0 ... 6298 2241 2722]
       [   0    0    0 ... 1360 4634 2722]
       [   0    0    0 ... 8222 6352 2722]
       ...
       [   0    0    0 ... 8130  586 7933]
       [   0    0    0 ... 7854 5752 6917]
       [   0    0    0 ... 1216 3409 4232]]
```

```
[40]: embedded_docs[1]

Out[40]: array([   0,    0,    0, ..., 1360, 4634, 2722], dtype=int32)
```

LSTM Model

We developed the base model and compiled it. The first layer will be the embedding layer which has the input of vocabulary size, vector features and sentence length. Later we add 30% dropout layer to prevent overfitting and the LSTM layer which has 100 neurons in the layer. In final layer we use sigmoid activation function. Later we compile the model using Adam optimizer and binary cross entropy as loss function since we have only two outputs.

```
41]: embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(Dropout(0.3))
model.add(LSTM(100))
model.add(Dropout(0.3))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 5000, 40)          400000
_________________________________________________________________
dropout (Dropout)            (None, 5000, 40)          0
_________________________________________________________________
lstm (LSTM)                  (None, 100)               56400
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 456,501
Trainable params: 456,501
Non-trainable params: 0
_________________________________________________________________
None
```

```
42]: len(embedded_docs),y.shape
```

```
Out[42]: (44888, (44888,))
```

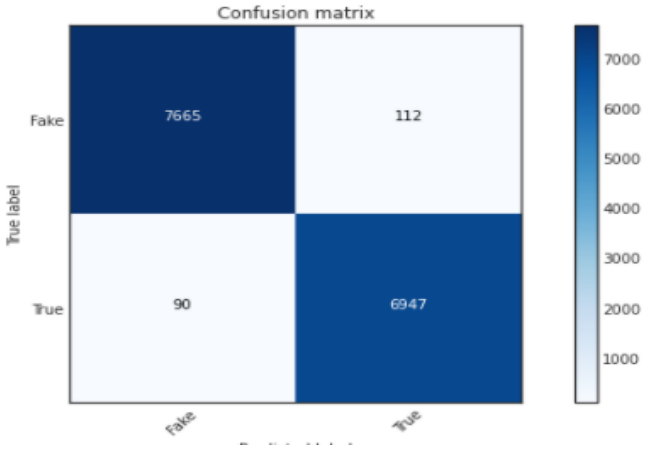
Let's split our new X and y variable into train and test and proceed with fitting the model to the data. We have considered **10 epochs and 64 as batch size**. It can be varied to get better results.

```
Epoch 1/10
470/470 [==============================] - 139s 286ms/step - loss: 0.3447 - accuracy: 0.8509 - val_loss: 0.0774 - val_accura
cy: 0.9766
Epoch 2/10
470/470 [==============================] - 133s 284ms/step - loss: 0.0815 - accuracy: 0.9753 - val_loss: 0.0722 - val_accura
cy: 0.9772
Epoch 3/10
470/470 [==============================] - 133s 283ms/step - loss: 0.0574 - accuracy: 0.9823 - val_loss: 0.0596 - val_accura
cy: 0.9820
Epoch 4/10
470/470 [==============================] - 133s 284ms/step - loss: 0.0311 - accuracy: 0.9909 - val_loss: 0.0464 - val_accura
cy: 0.9852
Epoch 5/10
470/470 [==============================] - 134s 286ms/step - loss: 0.0471 - accuracy: 0.9849 - val_loss: 0.0632 - val_accura
cy: 0.9832
Epoch 6/10
470/470 [==============================] - 135s 287ms/step - loss: 0.0306 - accuracy: 0.9916 - val_loss: 0.0862 - val_accura
cy: 0.9738
Epoch 7/10
470/470 [==============================] - 137s 291ms/step - loss: 0.1278 - accuracy: 0.9534 - val_loss: 0.1018 - val_accura
cy: 0.9704
Epoch 8/10
470/470 [==============================] - 137s 292ms/step - loss: 0.0403 - accuracy: 0.9868 - val_loss: 0.1338 - val_accura
cy: 0.9466
Epoch 9/10
470/470 [==============================] - 136s 289ms/step - loss: 0.0308 - accuracy: 0.9898 - val_loss: 0.0505 - val_accura
cy: 0.9831
Epoch 10/10
470/470 [==============================] - 138s 293ms/step - loss: 0.0188 - accuracy: 0.9943 - val_loss: 0.0484 - val_accura
cy: 0.9864
```

Confusion Matrix

```
[45]: y_pred=model.predict_classes(X_test)
      cm = metrics.confusion_matrix(y_test, y_pred)
      plot_confusion_matrix(cm,classes=['Fake','True'])
```

```
Confusion matrix, without normalization
```



Classification Report

```
In [46]: accuracy_score(y_test,y_pred)
Out[46]: 0.9863642500337518
```

```
In [47]: print(classification_report(y_test,y_pred))
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      7777
           1       0.98      0.99      0.99      7037

    accuracy                           0.99     14814
   macro avg       0.99      0.99      0.99     14814
weighted avg       0.99      0.99      0.99     14814
```

Conclusion:

From the classification report we can see the accuracy value is nearly around 96% for LSTM, we have to concentrate on precision score and it is 98% where our model performs well and from the AUC score which is 98%. The more far left the curve is better our model.