ARTIFICIAL
INTELLIGENCE

SUDHARSHAN EDUCATIONAL TRUST

Creating Pathways to Wisdom

SUDHARSAN
ENGINEERING COLLEGE

Name : S. Manikandan

Reg no: 814421243015

Dept : Artificial intelligence and Data Science

Title : Fake News detection using NLP

Table of contents:

1. Executive Summary:

- **Project's goal:** Develop a model to detect fake news articles.
- Gather data from Kaggle with article titles and labels.
- Clean and prepare textual data for analysis.
- Transform text into numerical features (TF-IDF, word embeddings).
- Choose the best classification algorithm for fake news detection.
- Train and evaluate the model using metrics like accuracy and F1-score.
- Explore advanced techniques (LSTM, BERT) for improved accuracy.
- **Project's significance**: Safeguard information credibility and informed decision-making.

2. Introduction:

- The project's focus is on developing a model for fake news detection.
- Fake news, misleading or false information, poses significant challenges in today's digital information landscape.
- Addressing this challenge is crucial to ensure the accuracy and integrity of information dissemination.
- The project aims to distinguish genuine from fake news articles using NLP techniques and machine learning.
- It leverages data collection, preprocessing, and advanced modeling to achieve this goal.
- Ultimately, the project's impact extends to promoting informed decision-making and safeguarding information credibility in the digital age.

3. Problem Statement Revisited:

☆ "The project aims to build a model for distinguishing real news from fake news, addressing the challenge of misinformation in the digital era"

4. Data Collection and Preprocessing:

- The project begins by gathering a dataset from Kaggle, comprising news articles' titles and text, along with authenticity labels.
- A thorough data preprocessing phase follows, involving text cleaning, tokenization, and encoding to ensure the data is ready for analysis.
- The goal is to prepare textual data meticulously, making it suitable for feature extraction and model training.

5. Feature Extraction:

- Feature extraction techniques, including TF-IDF and word embeddings, are employed to convert textual data into numerical representations.
- These numerical features are vital for training and evaluating classification models for fake news detection.
- Feature extraction bridges the gap between raw text data and the model, enabling effective analysis of the text's content and meaning.

6. Model Selection:

- The project carefully selects an appropriate classification algorithm, aligning it with the objective of fake news detection.
- The choice may involve algorithms like Logistic Regression, Random Forest, or Neural Networks, depending on their suitability for the task.
- The selected model serves as the core engine for distinguishing between genuine and fake news articles, and its effectiveness plays a crucial role in the project's success.

7. Model Training:

- The selected classification model is trained using the preprocessed data.
- During training, the model learns to differentiate between real and fake news articles based on the features extracted from the text data.
- This phase is pivotal in preparing the model to make accurate predictions in the later stages of the project.

8. Model Evaluation:

- The project rigorously evaluates the model's performance using a range of metrics, including accuracy, precision, recall, F1-score, and ROC-AUC.
- These metrics provide insights into the model's effectiveness in distinguishing between genuine and fake news articles.
- Model evaluation serves as a critical step in assessing the model's strengths and areas for improvement, ensuring it meets the project's objectives effectively.

9. Innovation:

- The project explores advanced techniques, such as deep learning models like LSTM and BERT, to enhance the accuracy of fake news detection.
- These innovative approaches offer the potential for improved model performance and more precise identification of fake news articles.
- Innovation in model techniques underscores the project's commitment to staying at the forefront of fake news detection methods.

10. Conclusion:

- The project's primary objective was to develop an effective model for distinguishing fake news from genuine news articles.
- By employing data collection, preprocessing, feature extraction, and rigorous model training, the project made substantial progress in addressing the challenge of misinformation.
- The exploration of advanced techniques, including deep learning models, holds promise for improved detection accuracy.
- In conclusion, the project is a vital step in countering the spread of misinformation, safeguarding information credibility, and promoting informed decision-making in the digital age.

11. Acknowledgments:

- Acknowledge individuals or organizations that provided support, data, or resources during the project.
- Express gratitude for their contributions or assistance.

Program:

[1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/input/'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/input/Fake.csv
/input/True.csv
/input/glove-twitter\glove.twitter.27B.100d.txt
/input/glove-twitter\glove.twitter.27B.100d.txt.zip
/input/glove-twitter\glove.twitter.27B.200d.txt
/input/glove-twitter\glove.twitter.27B.200d.txt.zip
/input/glove-twitter\glove.twitter.27B.25d.txt
/input/glove-twitter\glove.twitter.27B.25d.txt.zip
/input/glove-twitter\glove.twitter.27B.50d.txt
/input/glove-twitter\glove.twitter.27B.50d.txt.zip
```

[2]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud,STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
from bs4 import BeautifulSoup
import re,string,unicodedata
from keras.preprocessing import text, sequence
from sklearn.metrics import␣
 ↪classification_report,confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
from string import punctuation
from nltk import pos_tag
from nltk.corpus import wordnet
import keras
from keras.models import Sequential
from keras.layers import Dense,Embedding,LSTM,Dropout
from keras.callbacks import ReduceLROnPlateau
import tensorflow as tf
```

[3]:

```
true = pd.read_csv("/input/True.csv")
false  =  pd.read_csv("/input/Fake.csv")
```

[4]:

```
true.head()
```

[4]:

```
                                               title  \
0  As U.S. budget fight looms, Republicans flip t...
1  U.S. military to accept transgender recruits o...
2  Senior U.S. Republican senator: 'Let Mr. Muell...
3  FBI Russia probe helped by Australian diplomat...
4  Trump wants Postal Service to charge 'much mor...

                                                text       subject  \
0  WASHINGTON (Reuters) - The head of a conservat...  politicsNews
1  WASHINGTON (Reuters) - Transgender people will...  politicsNews
2  WASHINGTON (Reuters) - The special counsel inv...  politicsNews
3  WASHINGTON (Reuters) - Trump campaign adviser ...  politicsNews
4  SEATTLE/WASHINGTON (Reuters) - President Donal...  politicsNews

                 date
0  December 31, 2017
1  December 29, 2017
2  December 31, 2017
3  December 30, 2017
4  December 29, 2017
```

[5]:

```
false.head()
```

[5]:

```
                                               title  \
0   Donald Trump Sends Out Embarrassing New Year'...
1   Drunk Bragging Trump Staffer Started Russian ...
2   Sheriff David Clarke Becomes An Internet Joke...
3   Trump Is So Obsessed He Even Has Obama's Name...
4   Pope Francis Just Called Out Donald Trump Dur...
```

```
                                                text subject  \
0  Donald Trump just couldn t wish all Americans ...    News
1  House Intelligence Committee Chairman Devin Nu...    News
2  On Friday, it was revealed that former Milwauk...    News
3  On Christmas day, Donald Trump announced that ...    News
4  Pope Francis used his annual Christmas Day mes...    News

                date
0  December 31, 2017
1  December 31, 2017
2  December 30, 2017
3  December 29, 2017
4  December 25, 2017
```

[6]:

```
true["category"] = 1
false["category"] = 0
```

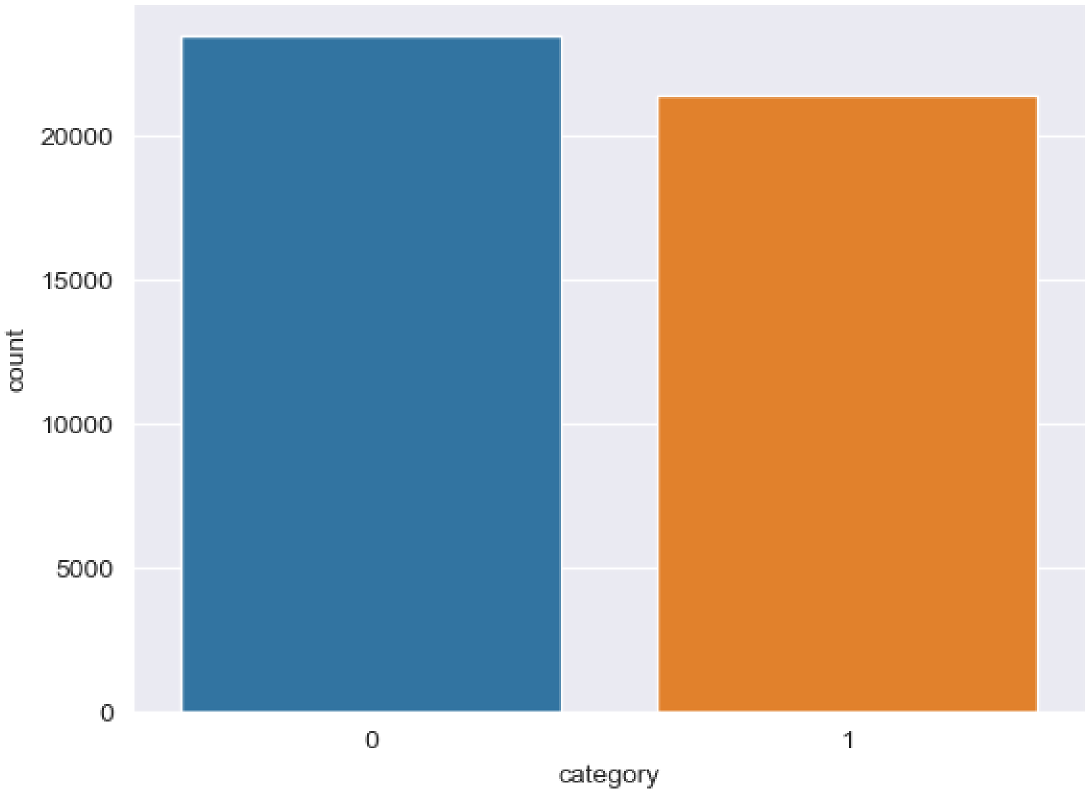
[7]:

```
df = pd.concat([true,false]) #Merging the 2 datasets
sns.set_style("darkgrid")
sns.countplot(df.category)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[7]:

```
<AxesSubplot:xlabel='category', ylabel='count'>
```



```
[8]: df.head()
```

```
[8]:                                              title  \
0  As U.S. budget fight looms, Republicans flip t...
1  U.S. military to accept transgender recruits o...
2  Senior U.S. Republican senator: 'Let Mr. Muell...
3  FBI Russia probe helped by Australian diplomat...
4  Trump wants Postal Service to charge 'much mor...

                                               text       subject  \
0  WASHINGTON (Reuters) - The head of a conservat...  politicsNews
1  WASHINGTON (Reuters) - Transgender people will...  politicsNews
2  WASHINGTON (Reuters) - The special counsel inv...  politicsNews
3  WASHINGTON (Reuters) - Trump campaign adviser ...  politicsNews
4  SEATTLE/WASHINGTON (Reuters) - President Donal...  politicsNews

                  date  category
0   December 31, 2017          1
1   December 29, 2017          1
2   December 31, 2017          1
3   December 30, 2017          1
```

```
4   December 29, 2017          1
```

```
[9]:  df.isna().sum() # Checking for nan Values
```

```
[9]:  title       0
      text        0
      subject     0
      date        0
      category    0
      dtype: int64
```

```
[10]: df.title.count()
```

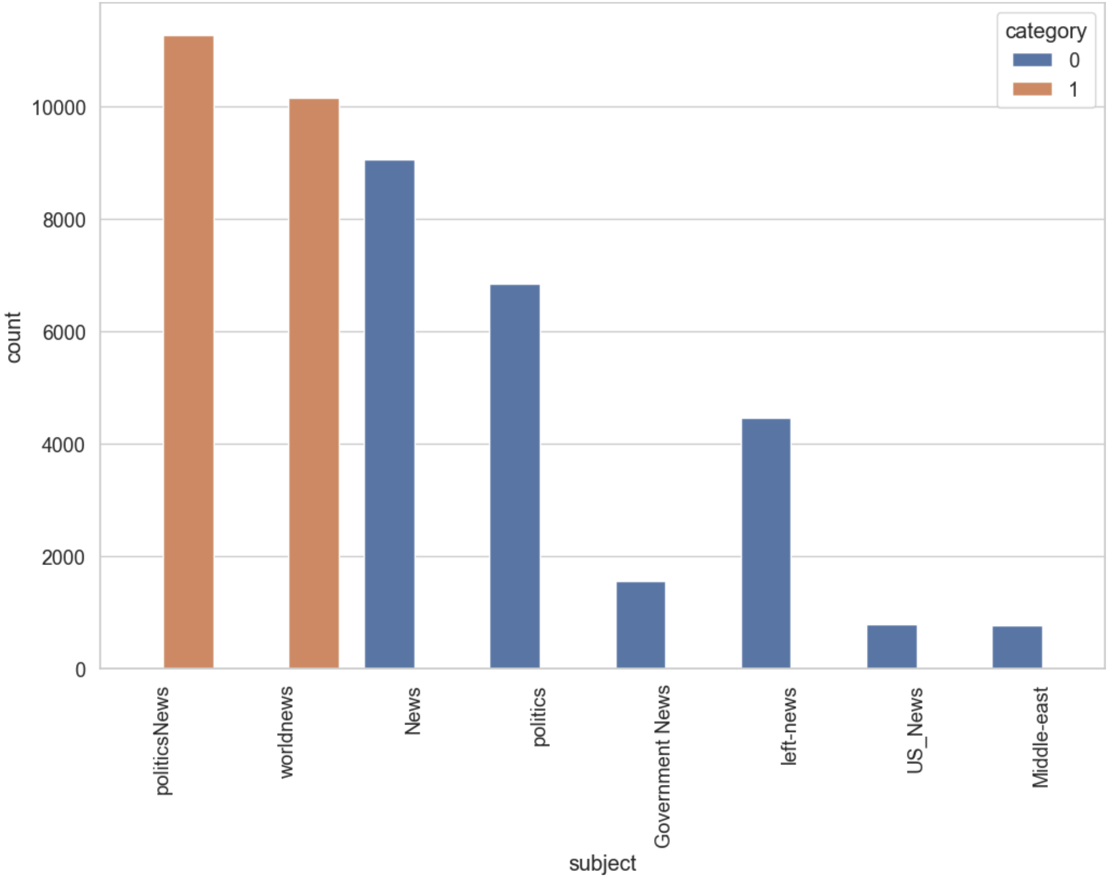
```
[10]: 44898
```

```
[11]: df.subject.value_counts()
```

```
[11]: politicsNews       11272
      worldnews          10145
      News                9050
      politics            6841
      left-news           4459
      Government News     1570
      US_News              783
      Middle-east          778
      Name: subject, dtype: int64
```

```
[12]: plt.figure(figsize = (12,8))
      sns.set(style = "whitegrid",font_scale = 1.2)
      chart = sns.countplot(x = "subject", hue = "category" , data = df)
      chart.set_xticklabels(chart.get_xticklabels(),rotation=90)
```

```
[12]: [Text(0, 0, 'politicsNews'),
       Text(1, 0, 'worldnews'),
       Text(2, 0, 'News'),
       Text(3, 0, 'politics'),
       Text(4, 0, 'Government News'),
       Text(5, 0, 'left-news'),
       Text(6, 0, 'US_News'),
       Text(7, 0, 'Middle-east')]
```



[13]:

```
df['text'] = df['text'] + " " + df['title']
del df['title']
del df['subject']
del df['date']
```

[14]:

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

[15]:

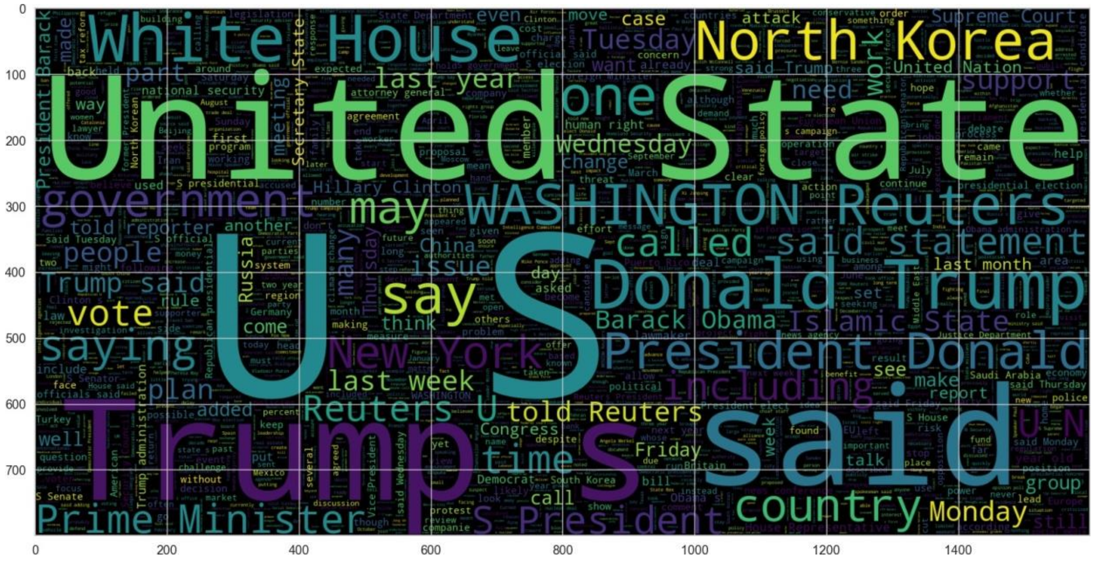
```
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

#Removing the square brackets
def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)
# Removing URL's
def remove_between_square_brackets(text):
    return re.sub(r'http\S+', '', text)
```

```
#Removing the stopwords from text
def remove_stopwords(text):
    final_text = []
    for i in text.split():
        if i.strip().lower() not in stop:
            final_text.append(i.strip())
    return " ".join(final_text)
#Removing the noisy text
def  denoise_text(text):
    text = strip_html(text)
    text = remove_between_square_brackets(text)
    text = remove_stopwords(text)
    return text
#Apply function on review column
df['text']=df['text'].apply(denoise_text)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\bs4\_init_.py:435:
MarkupResemblesLocatorWarning: The input looks more like a filename than markup.
You may want to open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
```

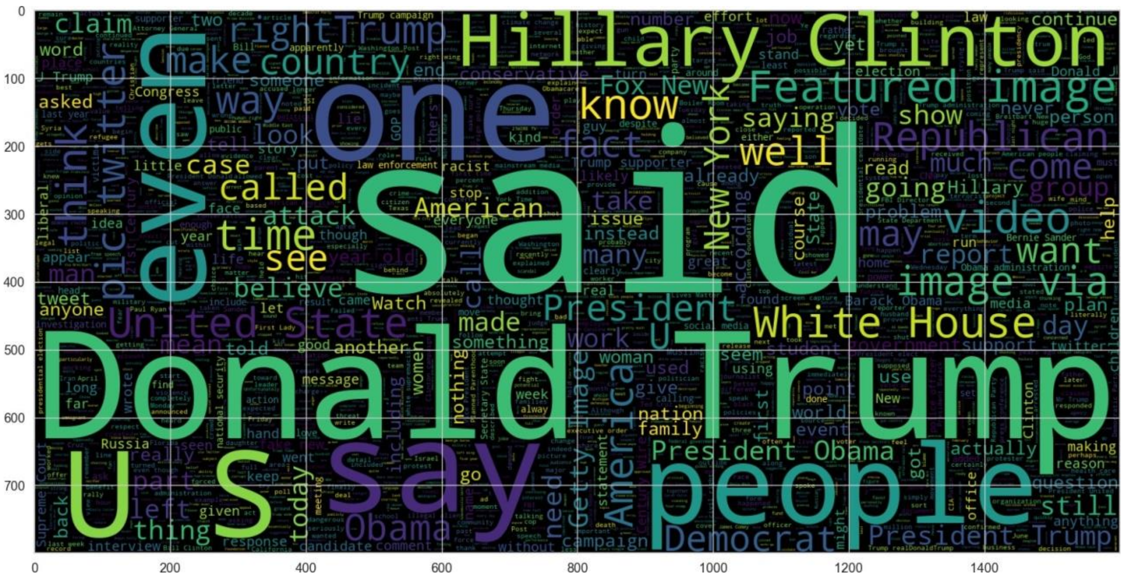
[16]:
```
plt.figure(figsize = (20,20)) # Text that is not Fake
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800 , stopwords =␣
 ↪STOPWORDS).generate(" ".join(df[df.category == 1].text))
plt.imshow(wc , interpolation = 'bilinear')
```

[16]:
```
<matplotlib.image.AxesImage at 0x1cfb6f8edc0>
```

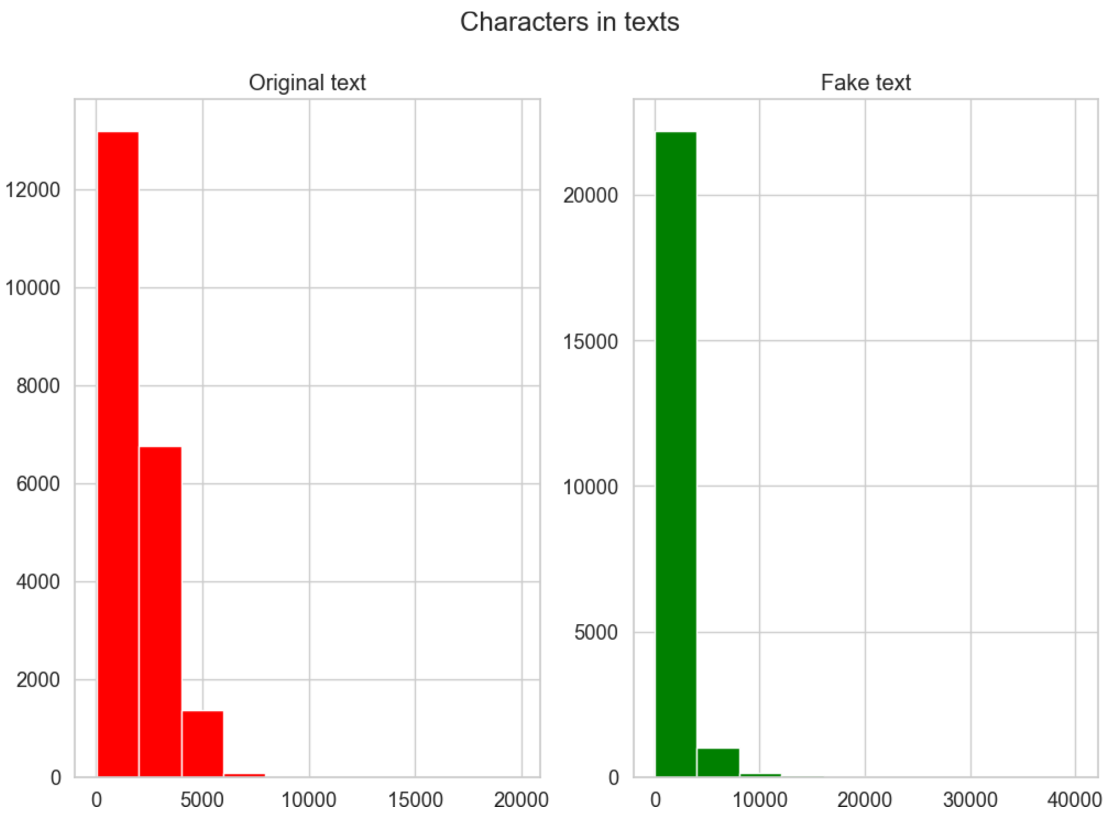


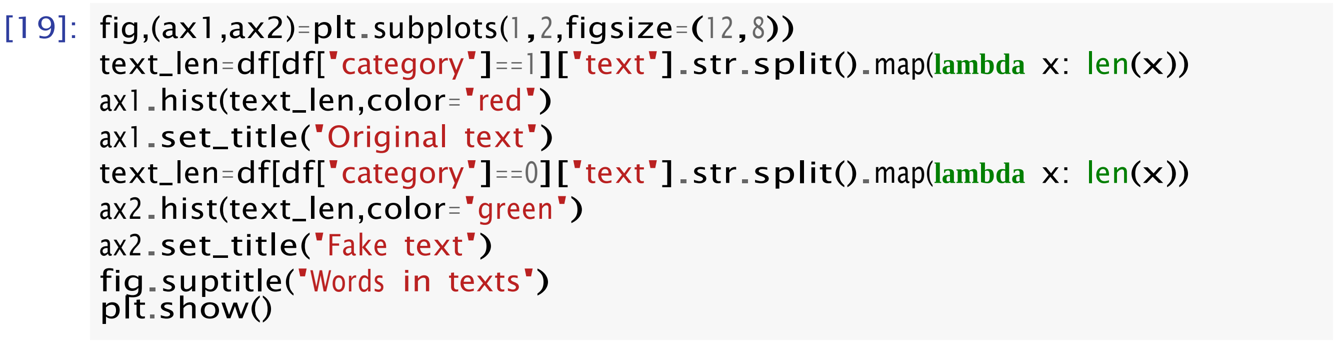
[17]:
```
plt.figure(figsize = (20,20)) # Text that is Fake
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800 , stopwords =␣
 ↪STOPWORDS).generate(" ".join(df[df.category == 0].text))
plt.imshow(wc , interpolation = 'bilinear')
```

[17]:
```
<matplotlib.image.AxesImage at 0x1d009a20280>
```

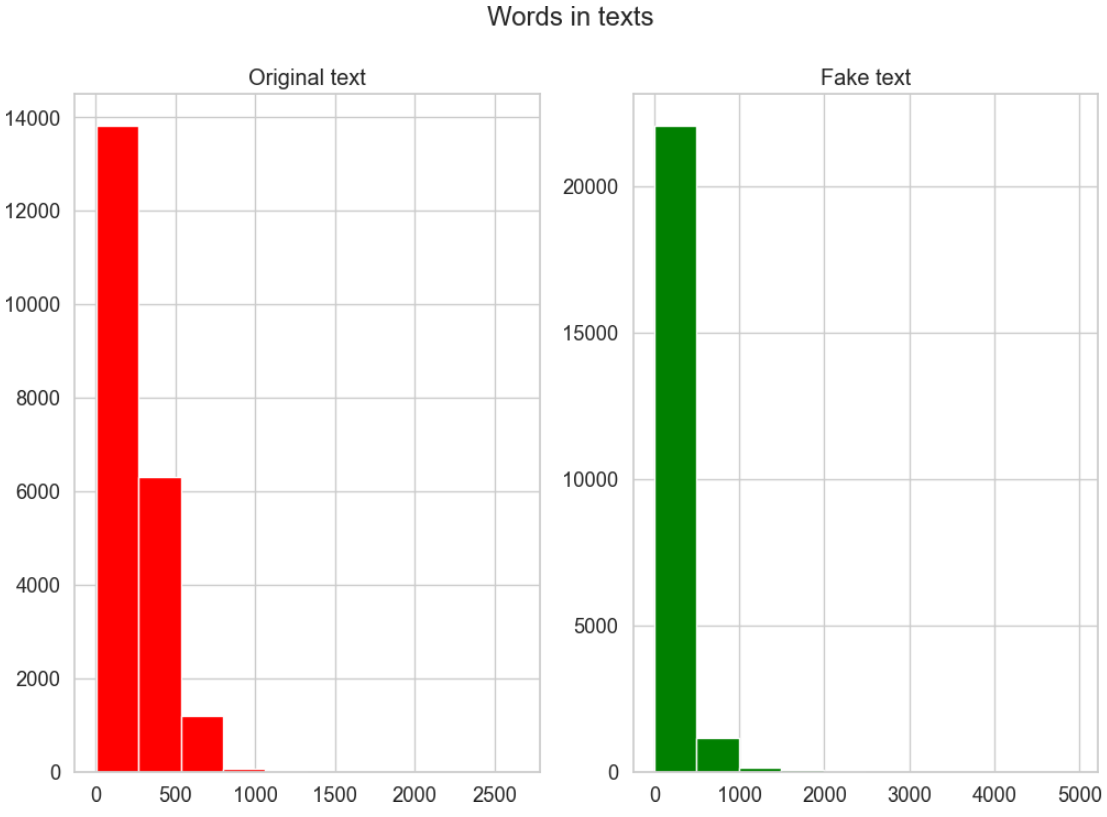


[18]:
```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(12,8))
text_len=df[df['category']==1]['text'].str.len()
ax1.hist(text_len,color='red')
ax1.set_title('Original text')
text_len=df[df['category']==0]['text'].str.len()
ax2.hist(text_len,color='green')
ax2.set_title('Fake text')
fig.suptitle('Characters in texts')
plt.show()
```





```
[19]: fig,(ax1,ax2)=plt.subplots(1,2,figsize=(12,8))
      text_len=df[df['category']==1]['text'].str.split().map(lambda x: len(x))
      ax1.hist(text_len,color='red')
      ax1.set_title('Original text')
      text_len=df[df['category']==0]['text'].str.split().map(lambda x: len(x))
      ax2.hist(text_len,color='green')
      ax2.set_title('Fake text')
      fig.suptitle('Words in texts')
      plt.show()
```



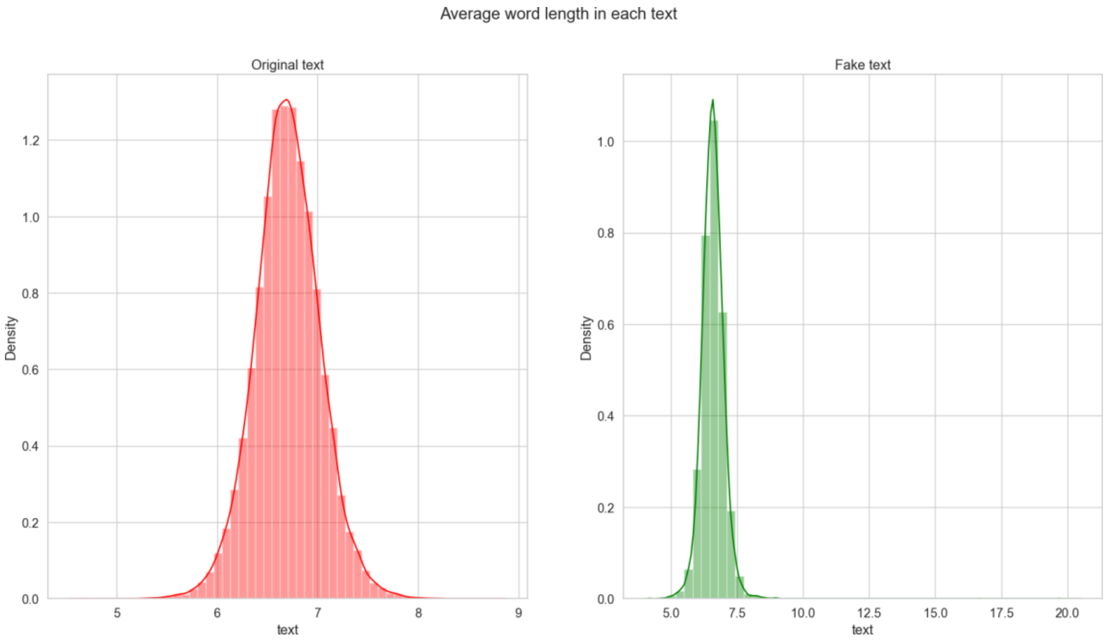
[20]:

```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(20,10))
word=df[df['category']==1]['text'].str.split().apply(lambda x : [len(i) for i␣
 ↪in x])
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax1,color='red')
ax1.set_title('Original text')
word=df[df['category']==0]['text'].str.split().apply(lambda x : [len(i) for i␣
 ↪in x])
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax2,color='green')
ax2.set_title('Fake text')
fig.suptitle('Average word length in each text')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3464:
RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_methods.py:192:
```

```
RuntimeWarning: invalid value encountered in scalar divide
  ret = ret.dtype.type(ret / rcount)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[20] :
```
Text(0.5, 0.98, 'Average word length in each text')
```



[21]:
```
def get_corpus(text):
    words = []
    for i in text:
        for j in i.split():
            words.append(j.strip())
    return words
corpus = get_corpus(df.text)
corpus[:5]
```

[21] :
```
['WASHINGTON', '(Reuters)', 'head', 'conservative', 'Republican']
```

[22]:
```
from collections import Counter
counter = Counter(corpus)
most_common = counter.most_common(10)
most_common = dict(most_common)
```

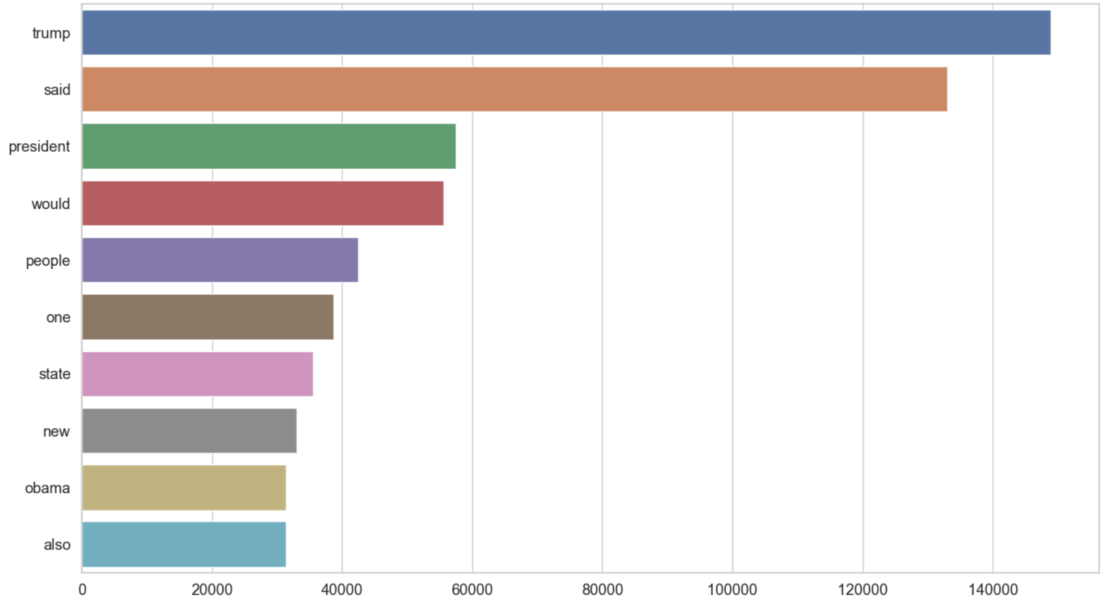
```
most_common
```

[22]:
```
{'Trump': 111503,
 'said': 93162,
 'would': 54613,
 'U.S.': 50441,
 'President': 33180,
 'people': 33115,
 'also': 30325,
 'one': 29370,
 'Donald': 27795,
 'said.': 26194}
```

[23]:
```
from sklearn.feature_extraction.text import CountVectorizer
def get_top_text_ngrams(corpus, n, g):
    vec = CountVectorizer(ngram_range=(g, g)).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]
```

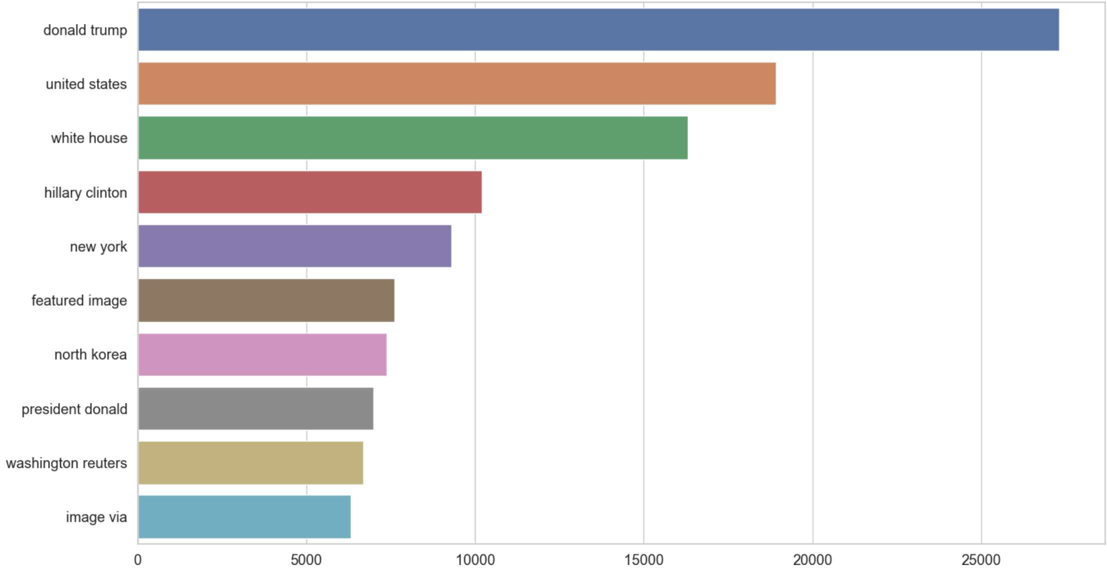
[24]:
```
plt.figure(figsize = (16,9))
most_common_uni = get_top_text_ngrams(df.text,10,1)
most_common_uni = dict(most_common_uni)
sns.barplot(x=list(most_common_uni.values()),y=list(most_common_uni.keys()))
```

[24] :
```
<AxesSubplot:>
```



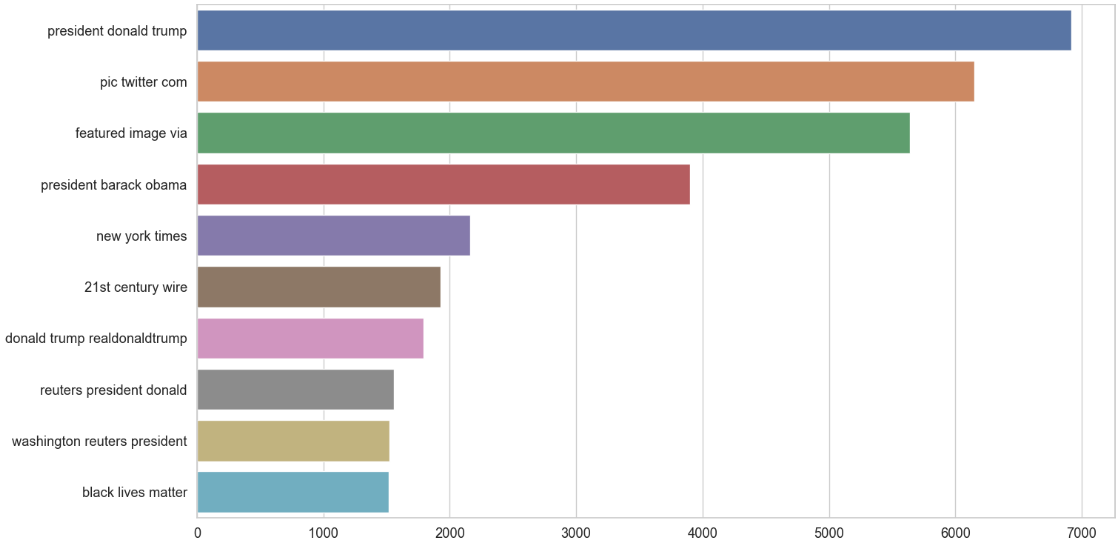
[25]:
```
plt.figure(figsize = (16,9))
most_common_bi = get_top_text_ngrams(df.text,10,2)
most_common_bi = dict(most_common_bi)
sns.barplot(x=list(most_common_bi.values()),y=list(most_common_bi.keys()))
```

[25] :
```
<AxesSubplot:>
```



[26]:
```
plt.figure(figsize = (16,9))
most_common_tri = get_top_text_ngrams(df.text,10,3)
most_common_tri = dict(most_common_tri)
sns.barplot(x=list(most_common_tri.values()),y=list(most_common_tri.keys()))
```

[26] :
```
<AxesSubplot:>
```



[27]:
```
x_train,x_test,y_train,y_test = train_test_split(df.text,df.category,random_state = 0)
```

[28]:
```
max_features = 10000
maxlen = 300
```

[29]:
```
tokenizer = text.Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(x_train)
tokenized_train = tokenizer.texts_to_sequences(x_train)
x_train = sequence.pad_sequences(tokenized_train, maxlen=maxlen)
```

[30]:
```
tokenized_test = tokenizer.texts_to_sequences(x_test)
X_test = sequence.pad_sequences(tokenized_test, maxlen=maxlen)
```

[31]:
```
EMBEDDING_FILE = "C:\\input\\glove-twitter\\glove.twitter.27B.100d.txt"
```

[32]:
```
def get_coefs(word, *arr):
    return word, np.asarray(arr, dtype='float32')

# Specify the correct encoding, e.g., 'utf-8'
with open(EMBEDDING_FILE, 'r', encoding='utf-8') as f:
    embeddings_index = dict(get_coefs(*o.rstrip().rsplit(' ')) for o in f)
```

[33]:
```
all_embs = np.stack(embeddings_index.values())
emb_mean,emb_std = all_embs.mean(), all_embs.std()
embed_size = all_embs.shape[1]

word_index = tokenizer.word_index
nb_words = min(max_features, len(word_index))
```

```
#change below line if computing normal stats is too slow
embedding_matrix = embedding_matrix = np.random.normal(emb_mean, emb_std,␣
 ↪(nb_words, embed_size))
for word, i in word_index.items():
    if i >= max_features: continue
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None: embedding_matrix[i] = embedding_vector
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3377: FutureWarning: arrays to stack
must be passed as a "sequence" type such as list or tuple. Support for non-
sequence iterables such as generators is deprecated as of NumPy 1.16 and will
raise an error in the future.
  if (await self.run_code(code, result,  async_=asy)):
```

[34]:
```
batch_size = 256
epochs = 3
embed_size = 100
```

[35]:
```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience =␣
 ↪2, verbose=1,factor=0.5, min_lr=0.00001)
```

[36]:
```
#Defining Neural Network
model = Sequential()
#Non-trainable embeddidng layer
model.add(Embedding(max_features, output_dim=embed_size,␣
 ↪weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128 , return_sequences = True , recurrent_dropout = 0.25 ,␣
 ↪dropout = 0.25))
model.add(LSTM(units=64 , recurrent_dropout = 0.1 , dropout = 0.1))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=keras.optimizers.Adam(lr = 0.01),␣
 ↪loss='binary_crossentropy', metrics=['accuracy'])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate`
or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

[37]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 300, 100)          1000000
 lstm (LSTM)                 (None, 300, 128)          117248

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 32)                2080

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 1168769 (4.46 MB)
Trainable params: 168769 (659.25 KB)
Non-trainable params: 1000000 (3.81 MB)
_________________________________________________________________
```

[38]:
```
history = model.fit(x_train, y_train, batch_size = batch_size , validation_data
 ↪= (X_test,y_test) , epochs = epochs , callbacks = [learning_rate_reduction])
```

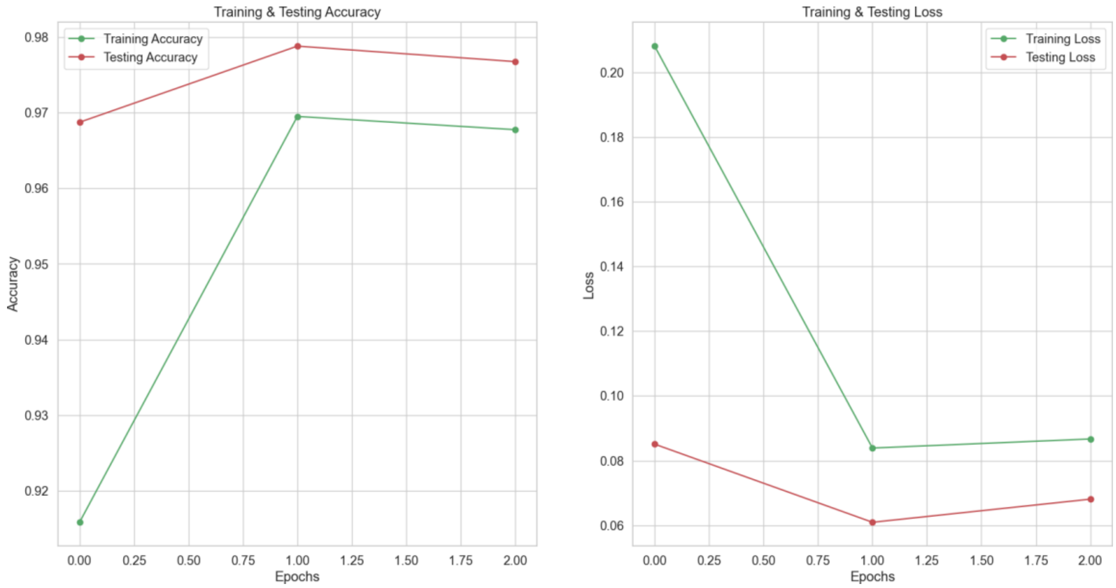
```
Epoch 1/3
132/132 [==============================] - 1571s 12s/step - loss: 0.2080 -
accuracy: 0.9158 - val_loss: 0.0851 - val_accuracy: 0.9687 - lr: 0.0010
Epoch 2/3
132/132 [==============================] - 1811s 14s/step - loss: 0.0839 -
accuracy: 0.9695 - val_loss: 0.0609 - val_accuracy: 0.9788 - lr: 0.0010
Epoch 3/3
132/132 [==============================] - 1913s 14s/step - loss: 0.0867 -
accuracy: 0.9677 - val_loss: 0.0681 - val_accuracy: 0.9767 - lr: 0.0010
```

[39]:
```
print("Accuracy of the model on Training Data is - " , model.
 ↪evaluate(x_train,y_train)[1]*100 , "%")
print("Accuracy of the model on Testing Data is - " , model.
 ↪evaluate(X_test,y_test)[1]*100 , "%")
```

```
1053/1053 [==============================] - 79s 75ms/step - loss: 0.0637 -
accuracy: 0.9773
Accuracy of the model on Training Data is -  97.73111939430237 %
351/351 [==============================] - 26s 75ms/step - loss: 0.0681 -
accuracy: 0.9767
Accuracy of the model on Testing Data is -  97.67483472824097 %
```

[40]:
```
epochs = [i for i in
range(3)]fig , ax =
plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
fig.set_size_inches(20,10)
```

```
ax[0].plot(epochs , val_acc , 'ro-' , label = 'Testing Accuracy')
ax[0].set_title('Training & Testing Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")

ax[1].plot(epochs , train_loss , 'go-' , label = 'Training Loss')
ax[1].plot(epochs , val_loss , 'ro-' , label = 'Testing Loss')
ax[1].set_title('Training & Testing Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Loss")
plt.show()
```



[47]:

```
import numpy as np

# Assuming 'model' is your Sequential model and 'X_test' is your test data

# Get the predicted probabilities for each class
pred_probabilities = model.predict(X_test)

# Get the class with the highest probability for each sample
pred_classes = np.argmax(pred_probabilities, axis=1)
pred_classes[:5]  # Print the first 5 predicted classes
```

```
351/351 [==============================] - 24s 68ms/step
```

[47]:

```
array([0, 0, 0, 0, 0], dtype=int64)
```

[49]:

```
import numpy as np
from sklearn.metrics import classification_report

# Assuming you have the necessary data and model for prediction
# Make predictions using your model
pred = model.predict(X_test)  # Assuming 'model' is your trained model and␣
  ↪'X_test' is your test data

# Convert probabilities to class predictions
pred_classes = np.argmax(pred, axis=1)

# Print the classification report
print(classification_report(y_test, pred_classes, target_names=["Fake", "Not␣
  ↪Fake"]))
```

```
351/351 [==============================] - 24s 68ms/step
              precision    recall  f1-score   support

        Fake       0.52      1.00      0.69      5858
    Not Fake       0.00      0.00      0.00      5367

    accuracy                           0.52     11225
   macro avg       0.26      0.50      0.34     11225
weighted avg       0.27      0.52      0.36     11225
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[51]:

```
import numpy as np
from sklearn.metrics import confusion_matrix

# Assuming 'y_test' contains the true labels and 'pred' contains the predicted␣
  ↪values
```

```
# Convert the predicted probabilities to class labels
pred_classes = np.argmax(pred, axis=1)

# Print the confusion matrix
cm = confusion_matrix(y_test, pred_classes)
print(cm)
```

```
[[5858    0]
 [5367    0]]
```

[52]:

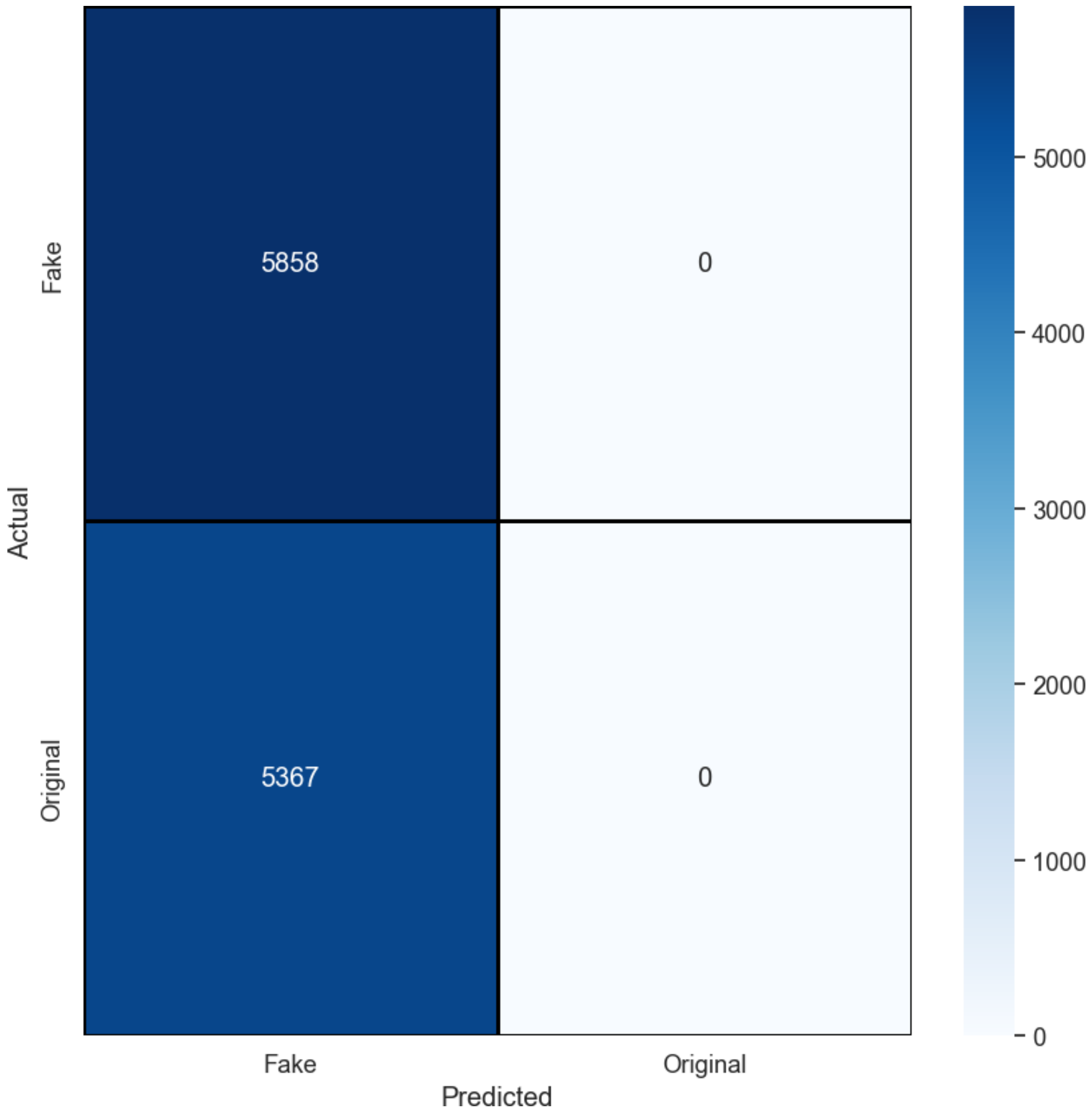
```
cm = pd.DataFrame(cm , index = ['Fake','Original'] , columns =␣
 ↪['Fake','Original'])
```

[53]:

```
plt.figure(figsize = (10,10))
sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth = 1 , annot =␣
 ↪True, fmt='' , xticklabels = ['Fake','Original'] , yticklabels =␣
 ↪['Fake','Original'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

[53]:

```
Text(88.25, 0.5, 'Actual')
```


Actual
Fake
Original
5858
0
5367
0
Fake
Original
Predicted
5000
4000
3000
2000
1000
0

```
[ ]:
```