**FAKE NEWS DETECTION USING NLP**

**PHASE 4: DEVELOPMENT PART2**

**Dataset Link:  https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news dataset**

**DATA VISUALIZATION:**

**CODE:**

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')

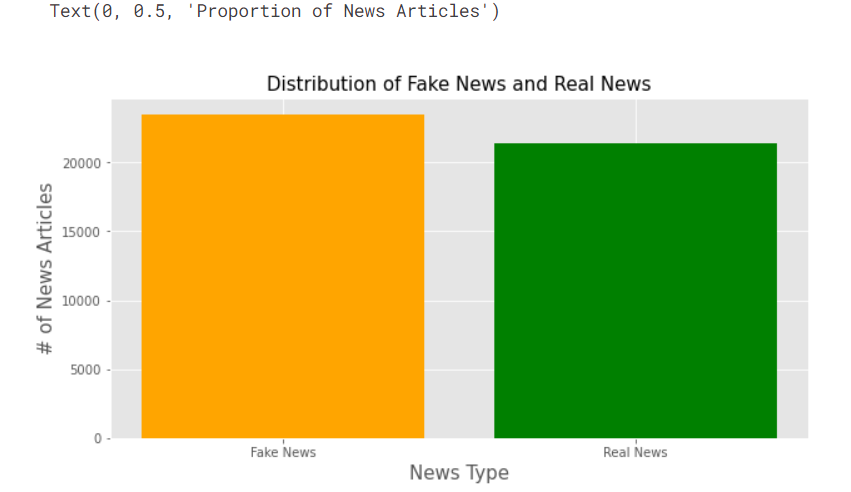
plt.bar('Real News', len(real\_df) / total\_len, color='green')

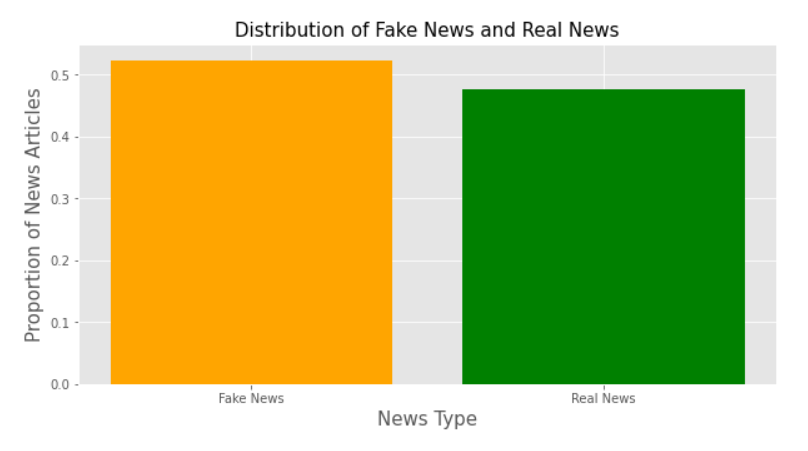
plt.title('Distribution of Fake News and Real News', size=15)

plt.xlabel('News Type', size=15)

plt.ylabel('Proportion of News Articles', size=15)

**OUTPUT:**





**VECTORIZATION:**

**CODE:**

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)

max\_vocab = 10000

tokenizer = Tokenizer(num\_words=max\_vocab)

tokenizer.fit\_on\_texts(X\_train)

# tokenize the text into vectors

X\_train = tokenizer.texts\_to\_sequences(X\_train)

X\_test = tokenizer.texts\_to\_sequences(X\_test)

X\_train = tf.keras.preprocessing.sequence.pad\_sequences(X\_train, padding='post', maxlen=256)

X\_test = tf.keras.preprocessing.sequence.pad\_sequences(X\_test, padding='post', maxlen=256)

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'),

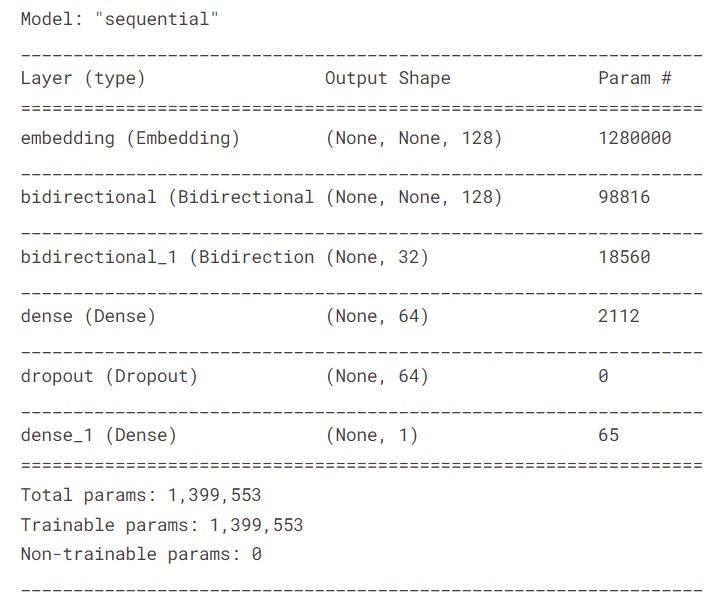
tf.keras.layers.Dropout(0.5),

tf.keras.layers.Dense(1)

])

model.summary()

**OUTPUT:**



**DATA SPLITTING INTO THE TRAIN AND TEST:**

**CODE:**

model.evaluate(X\_test, y\_test)

**OUTPUT:**

281/281 [==============================] - 5s 19ms/step - loss: 0.0501 - accuracy: 0.9861

[0.050108399242162704, 0.9860801696777344]

**MODEL EVALUATION:**

**CODE:**

pred = model.predict(X\_test)

binary\_predictions = []

for i in pred:

if i >= 0.5:

binary\_predictions.append(1)

else:

binary\_predictions.append(0)

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))

**OUTPUT:**

Accuracy on testing set: 0.9860801781737194

Precision on testing set: 0.9812413154238073

Recall on testing set: 0.9897220275636534

**CODE:**

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)

**OUTPUT:**

