FAKE NEWS DETECTION USING NLP

PHASE 4: DEVELOPMENT PART2

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

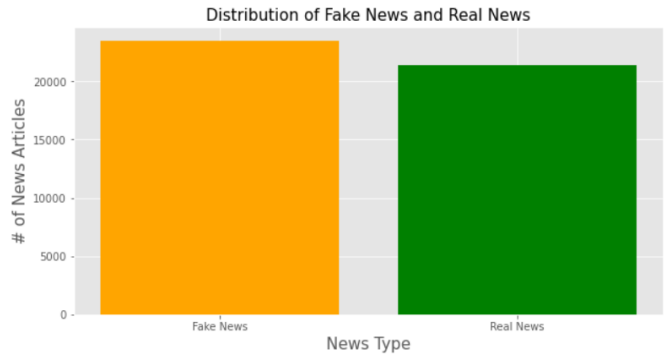
DATA VISUALIZATION:

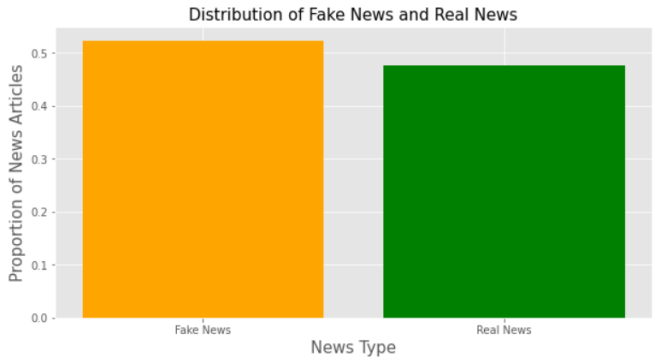
CODE:

```
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:

```
Text(0, 0.5, 'Proportion of News Articles')
```





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:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 128)         1280000
_________________________________________________________________
bidirectional (Bidirectional (None, None, 128)         98816
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32)                18560
_________________________________________________________________
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
_________________________________________________________________
```

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: