```
!pip install wget
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Requirement already satisfied: wget in /usr/local/lib/python3.9/dist-packages (3.2)
```

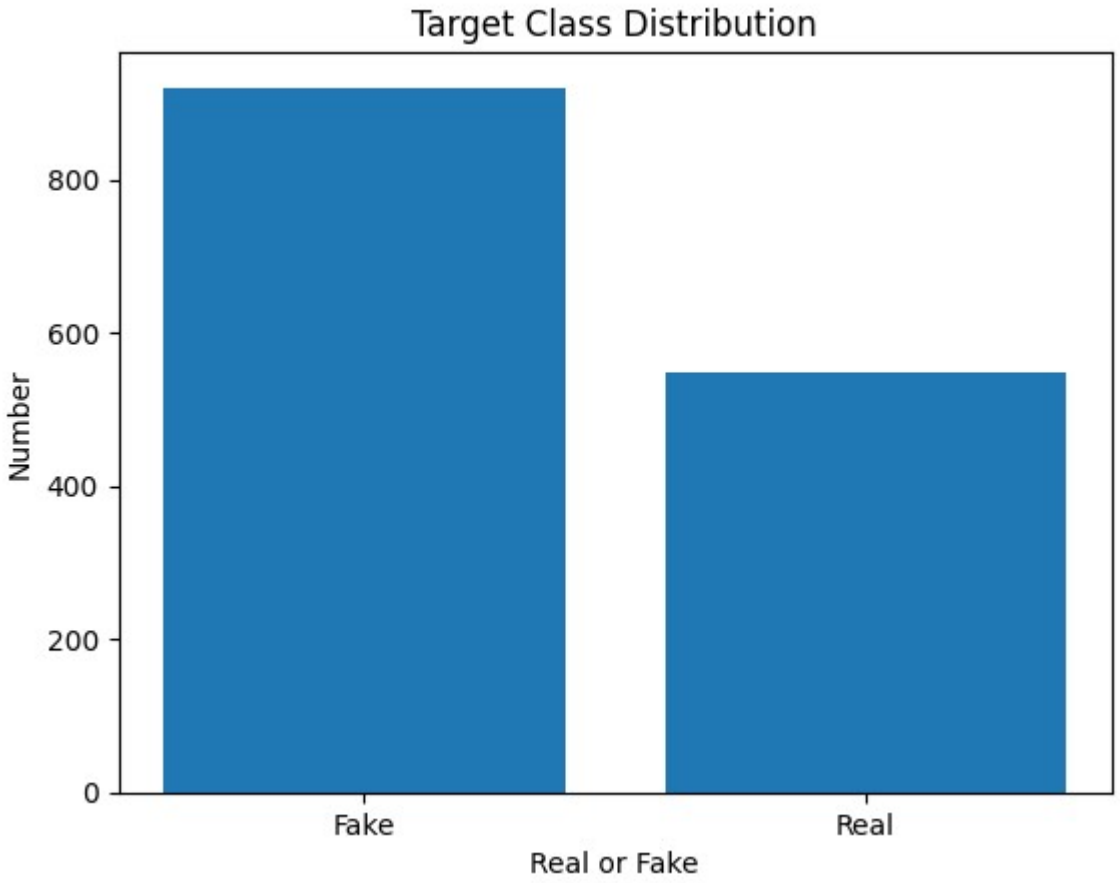
```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, Dropout
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Embedding, Conv1D, MaxPooling1D, GlobalMaxPooling1D, Dropo
from keras.utils import to_categorical
```

```
#dataset from https://www.kaggle.com/datasets/ruchi798/source-based-news-classification
df = pd.read_csv('news_articles.csv')
print(df.head(3))
train, test = train_test_split(df, test_size=0.3, random_state=12322)
```

```
                  author                       published  \
0     Barracuda Brigade  2016-10-26T21:41:00.000+03:00
1  reasoning with facts  2016-10-29T08:47:11.259+03:00
2     Barracuda Brigade  2016-10-31T01:41:49.479+02:00

                                               title  \
0  muslims busted they stole millions in govt ben...
1  re why did attorney general loretta lynch plea...
2  breaking weiner cooperating with fbi on hillar...

                                                text language  \
0  print they should pay all the back all the mon...  english
1  why did attorney general loretta lynch plead t...  english
2  red state  \nfox news sunday reported this mor...  english

              site_url                                           main_img_url  \
0  100percentfedup.com  http://bb4sp.com/wp-content/uploads/2016/10/Fu...
1  100percentfedup.com  http://bb4sp.com/wp-content/uploads/2016/10/Fu...
2  100percentfedup.com  http://bb4sp.com/wp-content/uploads/2016/10/Fu...

   type label                          title_without_stopwords  \
0  bias  Real        muslims busted stole millions govt benefits
1  bias  Real         attorney general loretta lynch plead fifth
2  bias  Real  breaking weiner cooperating fbi hillary email ...

                            text_without_stopwords  hasImage
0  print pay back money plus interest entire fami...       1.0
1  attorney general loretta lynch plead fifth bar...       1.0
2  red state fox news sunday reported morning ant...       1.0
```

✓ 1m 30s completed at 9:51 PM

```
labels = train["label"]
class_counts = labels.value_counts()
fig, ax = plt.subplots()
ax.bar(class_counts.index, class_counts.values)
ax.set_title("Target Class Distribution")
ax.set_xlabel("Real or Fake")
ax.set_ylabel("Number")
plt.show()
```



The data set includes a variety of news article titles along with the medium it was printed in, among other attributes. The model should be able to predict, using said features, if the article is "fake news."

```
#PREPROCESSING
train = train.dropna(subset=["text"])
train = train[train["text"] != ""]
test = test.dropna(subset=["text"])
test = test[test["text"] != ""]


tokenizer = Tokenizer(num_words=2500)
tokenizer.fit_on_texts(train["text"])
```

```
X_train = tokenizer.texts_to_sequences(train["text"])
X_test = tokenizer.texts_to_sequences(test["text"])
size_1 = len(tokenizer.word_index) + 1


X_train = pad_sequences(X_train, padding='post', maxlen=150)
X_test = pad_sequences(X_test, padding='post', maxlen=150)


y_train = train["label"].replace({"Fake": 0, "Real": 1})
y_test = test["label"].replace({"Fake": 0, "Real": 1})

model = Sequential()


#SEQUENTIAL
model.add(Dense(300, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(25, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=7, batch_size=32)

print("accuracy:", model.evaluate(X_test, y_test)[1])
```

```
Epoch 1/7
45/45 [==============================] - 2s 4ms/step - loss: 37.9616 - accuracy: 0.5
Epoch 2/7
45/45 [==============================] - 0s 5ms/step - loss: 8.0134 - accuracy: 0.64
Epoch 3/7
45/45 [==============================] - 0s 7ms/step - loss: 2.7624 - accuracy: 0.73
Epoch 4/7
45/45 [==============================] - 0s 7ms/step - loss: 1.1603 - accuracy: 0.80
Epoch 5/7
45/45 [==============================] - 0s 6ms/step - loss: 0.4400 - accuracy: 0.89
Epoch 6/7
45/45 [==============================] - 0s 5ms/step - loss: 0.2684 - accuracy: 0.93
Epoch 7/7
45/45 [==============================] - 0s 6ms/step - loss: 0.1474 - accuracy: 0.96
20/20 [==============================] - 0s 4ms/step - loss: 6.3761 - accuracy: 0.59
accuracy: 0.5921696424484253
```

```
#CNN-type model

model = Sequential() #reset model

model.add(Embedding(input_dim=size_1, output_dim=128, input_length=150))
model.add(Conv1D(128, 5, activation='relu'))
model.add(GlobalMaxPooling1D())
```

```
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))


model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=7, batch_size=128, validation_split=0.3)


print("accuracy:", model.evaluate(X_test, y_test)[1])
```

```
Epoch 1/7
8/8 [==============================] - 8s 792ms/step - loss: 0.6803 - accuracy: 0.58
Epoch 2/7
8/8 [==============================] - 5s 612ms/step - loss: 0.6397 - accuracy: 0.64
Epoch 3/7
8/8 [==============================] - 5s 603ms/step - loss: 0.6183 - accuracy: 0.64
Epoch 4/7
8/8 [==============================] - 4s 479ms/step - loss: 0.5984 - accuracy: 0.65
Epoch 5/7
8/8 [==============================] - 3s 374ms/step - loss: 0.5700 - accuracy: 0.65
Epoch 6/7
8/8 [==============================] - 3s 335ms/step - loss: 0.5228 - accuracy: 0.69
Epoch 7/7
8/8 [==============================] - 3s 325ms/step - loss: 0.4670 - accuracy: 0.76
20/20 [==============================] - 0s 19ms/step - loss: 0.6070 - accuracy: 0.6
accuracy: 0.6411092877388
```

```
#EMBEDDING focus

model = Sequential() #reset model
model.add(Embedding(input_dim=size_1, output_dim=64, input_length=150))
model.add(LSTM(32, dropout=0.3, recurrent_dropout=0.3))
model.add(Dense(1, activation='sigmoid'))


model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=7, batch_size=32, validation_split=0.3)


print("accuracy:", model.evaluate(X_test, y_test)[1])
```

```
Epoch 1/7
32/32 [==============================] - 23s 473ms/step - loss: 0.6702 - accuracy: 0
Epoch 2/7
32/32 [==============================] - 8s 248ms/step - loss: 0.6346 - accuracy: 0.
Epoch 3/7
32/32 [==============================] - 9s 272ms/step - loss: 0.5778 - accuracy: 0.
Epoch 4/7
32/32 [==============================] - 9s 288ms/step - loss: 0.4991 - accuracy: 0.
Epoch 5/7
32/32 [==============================] - 7s 233ms/step - loss: 0.4258 - accuracy: 0.
```

```
Epoch 6/7
32/32 [==============================] - 9s 288ms/step - loss: 0.3761 - accuracy: 0.
Epoch 7/7
32/32 [==============================] - 8s 258ms/step - loss: 0.3309 - accuracy: 0.
20/20 [==============================] - 1s 25ms/step - loss: 0.7325 - accuracy: 0.6
accuracy: 0.631321370601654
```

All the models performed at the same relative accuracy, which was higher than simply guessing (60-65% chance of getting prediction correct vs 50%). The CNN Type model performed the best with an accuracy of 64. The Embedded focused model performed closely next at 63%. Lastly, the sequential approach had about 60% accuracy. The accuracies being this close in value suggest that the approach itself may not be significantly important in the context of the current data (if deciding between the 3 models). Other approaches may be more effective and, furthermore, tuning the current model (for example, with different layers and parameters) could improve accuracy. Lastly, it's worth noting that the Sequential type approach took much less time (5 seconds vs almost a minute!) for a relatively close accuracy.