```
import pandas as pd
import numpy as np
import seaborn as sns
import re
import nltk
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Embedding,Dropout
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM,Bidirectional,GRU
from tensorflow.keras.layers import Dense
from sklearn.metrics import classification_report,accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.naive_bayes import MultinomialNB
%matplotlib inline
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/feature_extraction/image.py:167: Depre
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  dtype=np.int):
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:35: Deprec
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  eps=np.finfo(np.float).eps,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:597: Depre
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  eps=np.finfo(np.float).eps, copy_X=True, fit_path=True,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:836: Depre
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  eps=np.finfo(np.float).eps, copy_X=True, fit_path=True,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:862: Depre
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  eps=np.finfo(np.float).eps, positive=False):
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:1097: Depr
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  max_n_alphas=1000, n_jobs=None, eps=np.finfo(np.float).eps,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:1344: Depr
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  max_n_alphas=1000, n_jobs=None, eps=np.finfo(np.float).eps,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/least_angle.py:1480: Depr
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  eps=np.finfo(np.float).eps, copy_X=True, positive=False):
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/randomized_l1.py:152: Dep
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  precompute=False, eps=np.finfo(np.float).eps,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/randomized_l1.py:320: Dep
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  eps=np.finfo(np.float).eps, random_state=None,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/randomized_l1.py:580: Dep
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  eps=4 * np.finfo(np.float).eps, n_jobs=None,
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/gradient_boosting.py:34: Depr
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  from ._gradient_boosting import predict_stages
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/gradient_boosting.py:34: Depr
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  from ._gradient_boosting import predict_stages
```

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
train_data = pd.read_csv('/content/sample_data/train.csv')
test_data =  pd.read_csv('/content/sample_data/test.csv')
```

```
print(train_data.head())
```

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1

```
print(test_data.head())
```

	id	title	author	text
0	20800	Specter of Trump Loosens Tongues, if Not Purse...	David Streitfeld	PALO ALTO, Calif. — After years of scorning...
1	20801	Russian warships ready to strike terrorists ne...	NaN	Russian warships ready to strike terrorists ne...
2	20802	#NoDAPL: Native American Leaders Vow to Stay A...	Common Dreams	Videos #NoDAPL: Native American Leaders Vow to...
3	20803	Tim Tebow Will Attempt Another	[illegible]	If at first you don’t succeed, try a

```
print(train_data.shape)
```

```
(20800, 5)
```

```
print(test_data.shape)
```

```
(5200, 4)
```

```
print(train_data.describe())
```

	id	label
count	20800.000000	20800.000000
mean	10399.500000	0.500625
std	6004.587135	0.500012
min	0.000000	0.000000
25%	5199.750000	0.000000
50%	10399.500000	1.000000
75%	15599.250000	1.000000
max	20799.000000	1.000000

```
print(train_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20800 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      20800 non-null  int64 
 1   title   20242 non-null  object
 2   author  18843 non-null  object
 3   text    20761 non-null  object
 4   label   20800 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 812.6+ KB
```

```
print(train_data.isnull().sum())
```

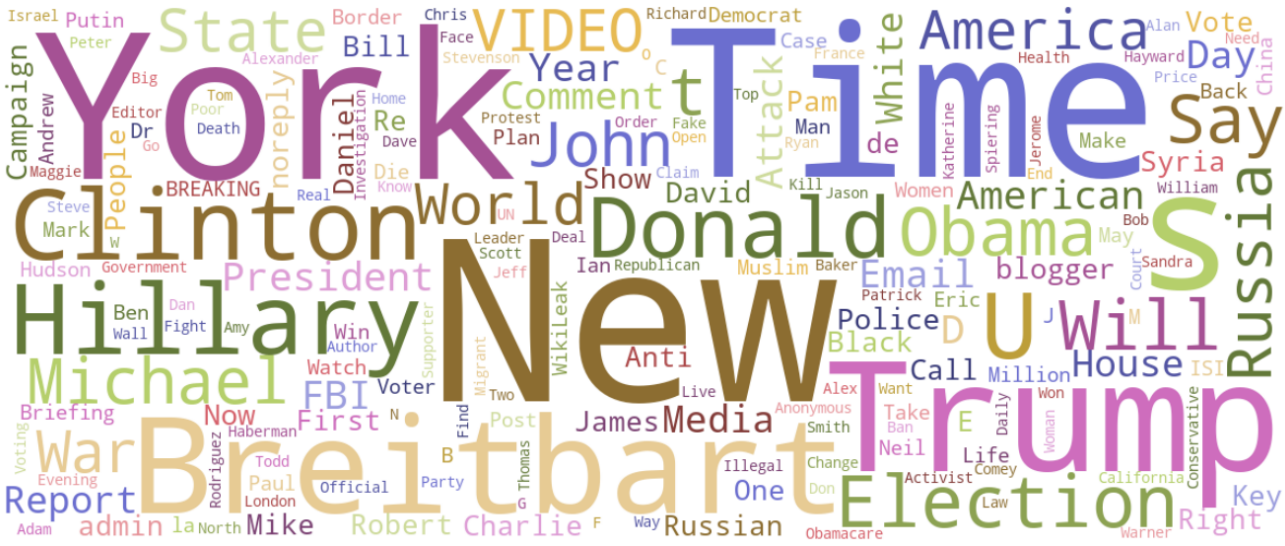
```
id           0
title      558
author    1957
text        39
label        0
dtype: int64
```

```
print(test_data.isnull().sum())
```

```
id          0
title     122
author    503
text        7
dtype: int64
```

```
# Adding empty spaces to the Nan values in the dataset

def handle_null(train_data, test_data):
  train = train_data.fillna(" ")
```

```
  test  = test_data.fillna(" ")
  return train,test

train_data,test_data = handle_null(train_data,test_data)


train_data['Author&Title'] = train_data['author'] + ' ' + train_data['title']
test_data['Author&Title'] = test_data['author'] + ' ' + test_data['title']



print(train_data['Author&Title'])

     0        Darrell Lucus House Dem Aide: We Didn’t Even S...
     1        Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
     2        Consortiumnews.com Why the Truth Might Get You...
     3        Jessica Purkiss 15 Civilians Killed In Single ...
     4        Howard Portnoy Iranian woman jailed for fictio...
                                ...
     20795    Jerome Hudson Rapper T.I.: Trump a ’Poster Chi...
     20796    Benjamin Hoffman N.F.L. Playoffs: Schedule, Ma...
     20797    Michael J. de la Merced and Rachel Abrams Macy...
     20798    Alex Ansary NATO, Russia To Hold Parallel Exer...
     20799              David Swanson What Keeps the F-35 Alive
     Name: Author&Title, Length: 20800, dtype: object


from wordcloud import WordCloud
from collections import Counter

# Plotting Word Cloud

# Finding the most common word

words = list(train_data['Author&Title'].apply(lambda x:x.split()))
words = [x for y in words for x in y]

most_common40 = Counter(words).most_common(40)
wc = WordCloud(width=1200, height=500,collocations=False, background_color="white",colorma

# collocations to False  is set to ensure that the word cloud doesn't appear as if it cont
plt.figure(figsize=(25,10))
# generate word cloud, interpolation
plt.imshow(wc, interpolation='bilinear')
_ = plt.axis("off")
```




```
X = train_data.drop(columns = 'label', axis = 1)
Y = train_data['label']
```

```
print(X)
```

	id	title	author	text	Author&Title
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus House Dem Aide: We Didn’t Even S...
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	Consortiumnews.com Why the Truth Might Get You...
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	Jessica Purkiss 15 Civilians Killed In Single ...
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	Howard Portnoy Iranian woman jailed for fictio...

```
print(Y)
```

```
0        1
1        0
```

```
2        1
3        1
4        1
        ..
20795    0
20796    0
20797    0
20798    1
20799    1
Name: label, Length: 20800, dtype: int64
```

```
print(test_data)
```

	id	title	author	text	Author&Title
0	20800	Specter of Trump Loosens Tongues, if Not Purse...	David Streitfeld	PALO ALTO, Calif. — After years of scorning...	David Streitfeld Specter of Trump Loosens Tong...
1	20801	Russian warships ready to strike terrorists ne...		Russian warships ready to strike terrorists ne...	Russian warships ready to strike terrorists ...
2	20802	#NoDAPL: Native American Leaders Vow to Stay A...	Common Dreams	Videos #NoDAPL: Native American Leaders Vow to...	Common Dreams #NoDAPL: Native American Leaders...
3	20803	Tim Tebow Will Attempt Another Comeback, This ...	Daniel Victor	If at first you don't succeed, try a different...	Daniel Victor Tim Tebow Will Attempt Another C...
4	20804	Keiser Report: Meme Wars (E995)	Truth Broadcast Network	42 mins ago 1 Views 0 Comments 0 Likes 'For th...	Truth Broadcast Network Keiser Report: Meme Wa...
...	...	...	...	...	...
		The Bangladeshi	Jody	Of all the dysfunctions	Jody Rosen The

```
#Resetting the index

messages = X.copy()
messages.reset_index(inplace=True)
messages_test = test_data.copy()
messages_test.reset_index(inplace=True)


stem = PorterStemmer()
def preprocess(input):
  corpus = []
  for i in range(0,len(input)):
    data = re.sub('[^a-zA-Z]',' ',input['Author&Title'][i])
    data = data.lower()
    data = data.split()
    data = [stem.stem(word) for word in data if not word in stopwords.words('english')]
    data = ' '.join(data)
    corpus.append(data)
  return corpus
```

```
train_corpus = preprocess(messages)
test_corpus = preprocess(messages_test)


print(train_corpus[1])
print(test_corpus[1])
```

```
     daniel j flynn flynn hillari clinton big woman campu breitbart
     russian warship readi strike terrorist near aleppo
```

```
vocab_size = 5000
one_hot_train = [one_hot(word,vocab_size) for word in train_corpus]
one_hot_test  = [one_hot(word,vocab_size) for word in test_corpus]


# Embedding Representation
sent_length = 20
embed_train = pad_sequences(one_hot_train,padding='pre',maxlen=sent_length)
embed_test  = pad_sequences(one_hot_test,padding='pre',maxlen=sent_length)



print(embed_train)
print(embed_test)
```

```
     [[   0    0    0 ...  906 3336 3866]
      [   0    0    0 ... 2645  381  960]
      [   0    0    0 ... 4466 2882 3670]
      ...
      [   0    0    0 ... 3592  871 2096]
      [   0    0    0 ... 2032 4665 4720]
      [   0    0    0 ... 4566 1189 2684]]
     [[   0    0    0 ... 3592  871 2096]
      [   0    0    0 ... 4121 1787 3647]
      [   0    0    0 ... 3523 1998 3285]
      ...
      [   0    0    0 ... 3592  871 2096]
      [   0    0    0 ... 1395 4281  117]
      [   0    0    0 ... 3592  871 2096]]
```

```
x_final = np.array(embed_train)
y_final = np.array(Y)
x_test_final = np.array(embed_test)


X_train, X_test, y_train, y_test = train_test_split(x_final, y_final, test_size=0.20, rand


# LOGISTIC REGRESSION
model_LR = LogisticRegression()
model_LR.fit(X_train,y_train)
predictions_LR = model_LR.predict(X_test)
classification_report_LR = classification_report(y_test,predictions_LR)
print('Classification Report\n', classification_report_LR)
print('Acuracy Score\n', accuracy_score(y_test,predictions_LR))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/logistic.py:433: FutureWa
  FutureWarning)
Classification Report
              precision    recall  f1-score   support

           0       0.76      0.81      0.78      2107
           1       0.79      0.74      0.76      2053

   micro avg       0.77      0.77      0.77      4160
   macro avg       0.77      0.77      0.77      4160
weighted avg       0.77      0.77      0.77      4160

Acuracy Score
 0.7737980769230769
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/base.py:283: DeprecationW
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  indices = (scores > 0).astype(np.int)
```

```
# NAIVE BAYES

model_NB = MultinomialNB()
model_NB.fit(X_train,y_train)
predictions_NB = model_NB.predict(X_test)
classification_report_NB = classification_report(y_test,predictions_NB)
print('Classification Report\n', classification_report_NB)
print('Acuracy Score\n', accuracy_score(y_test,predictions_NB))
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.75      0.66      0.70      2107
           1       0.69      0.77      0.73      2053

   micro avg       0.71      0.71      0.71      4160
   macro avg       0.72      0.71      0.71      4160
weighted avg       0.72      0.71      0.71      4160

Acuracy Score
 0.7134615384615385
```

```
# RANDOM FOREST

model_RFC = RandomForestClassifier()
model_RFC.fit(X_train,y_train)
predictions_RFC = model_RFC.predict(X_test)
classification_report_RFC = classification_report(y_test,predictions_RFC)
print('Classification Report\n', classification_report_RFC)
print('Acuracy Score\n', accuracy_score(y_test,predictions_RFC))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/forest.py:246: FutureWarning
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/forest.py:487: DeprecationWar
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_store_unique_indices = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
```

```
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
Classification Report
               precision    recall  f1-score   support

           0       0.95      0.87      0.90      2107
           1       0.87      0.95      0.91      2053

   micro avg       0.91      0.91      0.91      4160
   macro avg       0.91      0.91      0.91      4160
weighted avg       0.91      0.91      0.91      4160

Acuracy Score
 0.9072115384615385
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/tree/tree.py:149: DeprecationWarning:
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  y_encoded = np.zeros(y.shape, dtype=np.int)
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/base.py:158: DeprecationWarni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  dtype=np.int)
```

```
# XGBOOST

model_XGB = XGBClassifier()
model_XGB.fit(X_train,y_train)
predictions_XGB = model_XGB.predict(X_test)
classification_report_XGB = classification_report(y_test,predictions_XGB)
print('Classification Report\n', classification_report_XGB)
print('Acuracy Score\n', accuracy_score(y_test,predictions_XGB))
```

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

```
                0       1.00      0.82      0.90      2107
                1       0.85      1.00      0.91      2053

         micro avg       0.91      0.91      0.91      4160
         macro avg       0.92      0.91      0.91      4160
      weighted avg       0.92      0.91      0.91      4160

      Acuracy Score
       0.9084134615384616
```

```
# NEURAL NETWORKS: LSTM

# Creating the LSTM Model for prediction
embedding_feature_vector = 40
model = Sequential()
model.add(Embedding(vocab_size,embedding_feature_vector,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, 20, 40)            200000

 dropout (Dropout)           (None, 20, 40)            0

 lstm (LSTM)                 (None, 100)               56400

 dropout_1 (Dropout)         (None, 100)               0

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 256,501
Trainable params: 256,501
Non-trainable params: 0
_________________________________________________________________
None
```

```
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=10,batch_size=64)
```

```
Epoch 1/10
260/260 [==============================] - 15s 43ms/step - loss: 0.2023 - accuracy: 0
Epoch 2/10
260/260 [==============================] - 16s 61ms/step - loss: 0.0319 - accuracy: 0
Epoch 3/10
260/260 [==============================] - 11s 42ms/step - loss: 0.0174 - accuracy: 0
Epoch 4/10
260/260 [==============================] - 9s 36ms/step - loss: 0.0086 - accuracy: 0
Epoch 5/10
260/260 [==============================] - 9s 36ms/step - loss: 0.0052 - accuracy: 0
Epoch 6/10
```

```
260/260 [==============================] - 10s 39ms/step - loss: 0.0043 - accuracy: 0
Epoch 7/10
260/260 [==============================] - 10s 37ms/step - loss: 0.0032 - accuracy: 0
Epoch 8/10
260/260 [==============================] - 9s 36ms/step - loss: 0.0016 - accuracy: 0
Epoch 9/10
260/260 [==============================] - 15s 58ms/step - loss: 0.0031 - accuracy: 0
Epoch 10/10
260/260 [==============================] - 17s 65ms/step - loss: 0.0023 - accuracy: 0
<keras.callbacks.History at 0x7f8c89ea5dd0>
```

```
# # Making Predictions on test data with XGBOOST MODEL
labels = pd.read_csv('/content/sample_data/labels.csv')
target = labels['label']
predictions_test = model_XGB.predict(x_test_final)
```

```
print(labels)
```

```
        id  label
0     20800      0
1     20801      1
2     20802      0
3     20803      1
4     20804      1
...     ...    ...
5195  25995      0
5196  25996      1
5197  25997      0
5198  25998      1
5199  25999      0

[5200 rows x 2 columns]
```

```
error = np.mean(predictions_test != target)
print("Error rate  = ",error*100," %")
```

```
Error rate  =  31.0  %
```

✓ 0s completed at 2:56 AM