Author_Identification

September 25, 2022

[1]:
```
import pandas as pd
train_data = pd.read_csv("train-authors.csv")
test_data = pd.read_csv("test-authors.csv")
```

[2]:
```
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

[3]:
```
train_data.head(5)
```

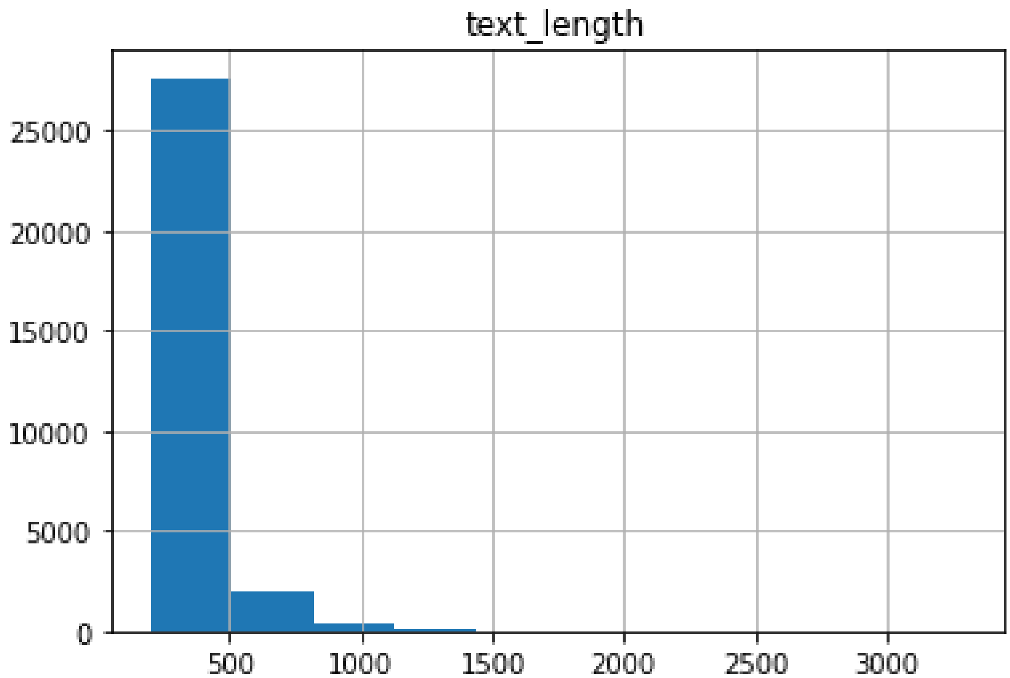
[3]:
```
                                                text   author
0   She wanted clothes to keep her warm, and food...  dickens
1   The question now was, who was the man,\nand w...    doyle
2   I therefore\n      smoked a great number of t...    doyle
3   I am partial to the modern\nFrench school. \n...    doyle
4  ’’ She stood smiling, holding up a little slip ...    doyle
```

[4]:
```
train_data['author'].value_counts()
```

[4]:
```
defoe      7569
dickens    7493
twain      7478
doyle      7460
Name: author, dtype: int64
```

[5]:
```
train_data['text_length'] = train_data['text'].str.len()
```

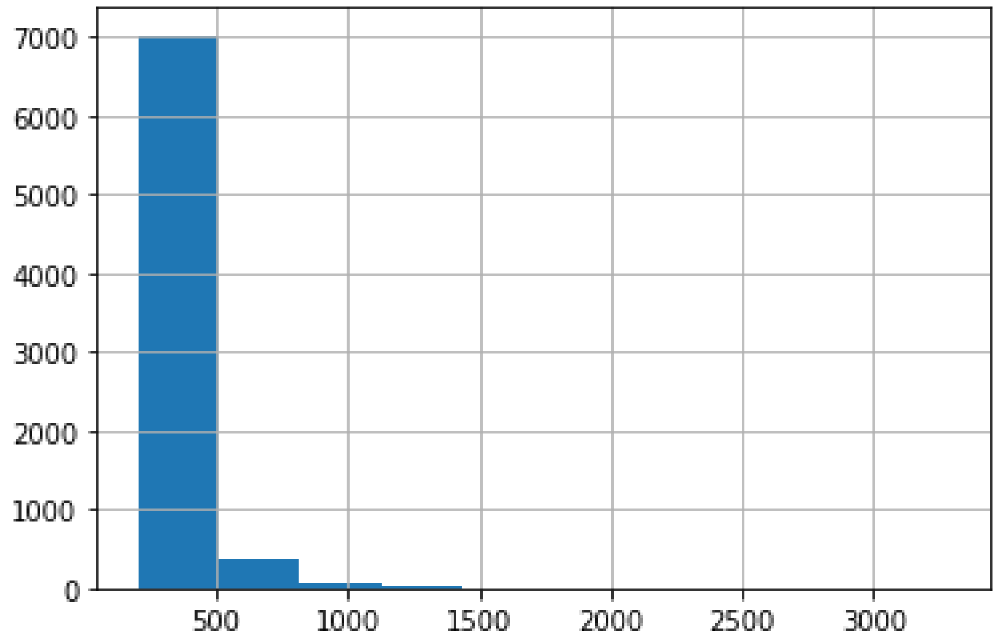
[6]:
```
train_data.hist()
plt.show()
```



[7]:
```
twain_train = train_data[train_data['author'] =='twain']['text_length']
twain_train.describe()
```

[7]:
```
count    7478.000000
mean      309.776678
std       135.313290
min       200.000000
25%       231.000000
50%       272.000000
75%       340.000000
max      3289.000000
Name: text_length, dtype: float64
```

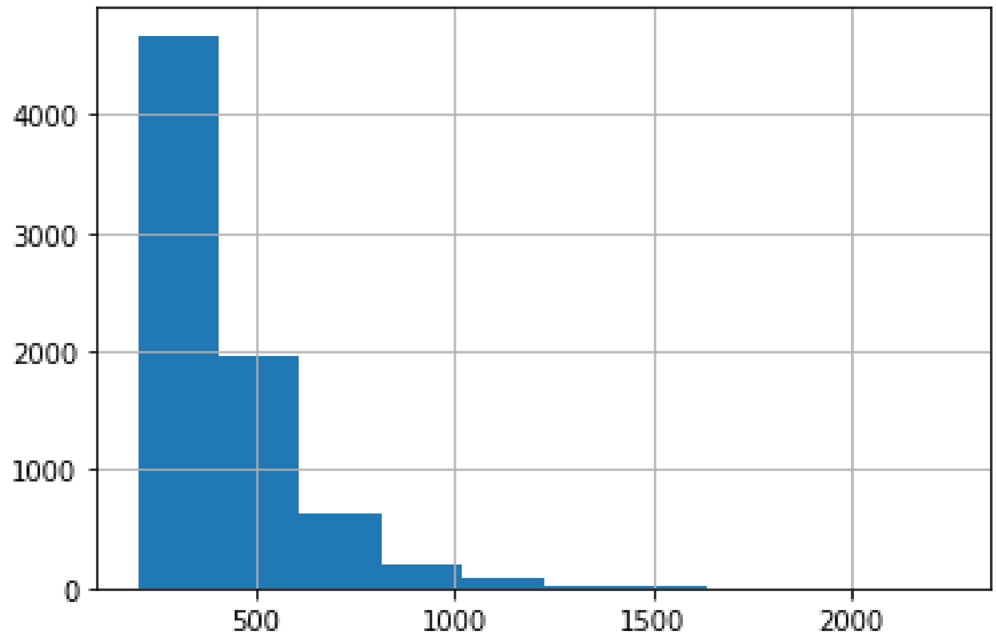
[8]:
```
twain_train.hist()
plt.show()
```



[9]:
```
defoe_train = train_data[train_data['author'] == 'defoe']['text_length']
defoe_train.describe()
```

[9]:
```
count    7569.000000
mean      405.952041
std       195.312379
min       200.000000
25%       267.000000
50%       353.000000
75%       483.000000
max      2249.000000
Name: text_length, dtype: float64
```

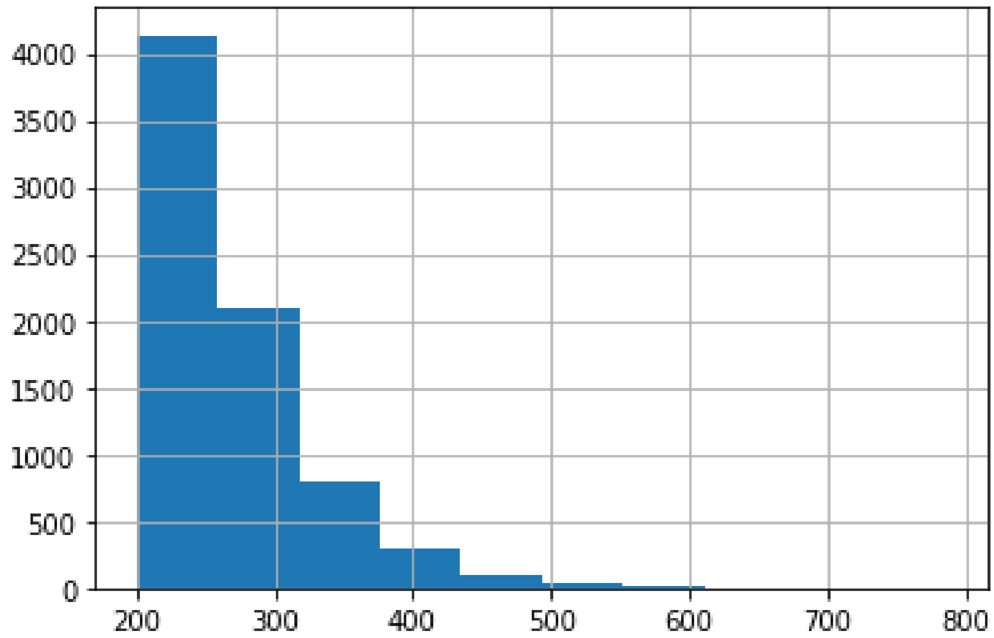
[10]:
```
defoe_train.hist()
plt.show()
```



[11]:
```
doyle_train = train_data[train_data['author'] == 'doyle']['text_length']
doyle_train.describe()
```

[11]:
```
count    7460.000000
mean      266.766622
std        58.795911
min       200.000000
25%       224.000000
50%       251.000000
75%       294.000000
max       787.000000
Name: text_length, dtype: float64
```

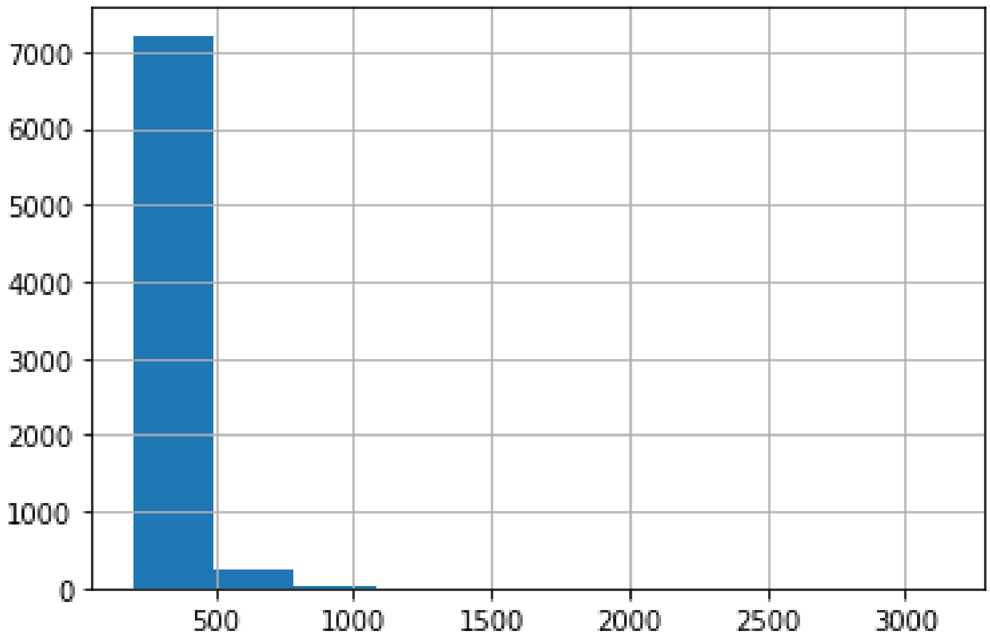
[12]:
```
doyle_train.hist()
plt.show()
```



[13]:
```
dickens_train = train_data[train_data['author'] == 'dickens']['text_length']
dickens_train.describe()
```

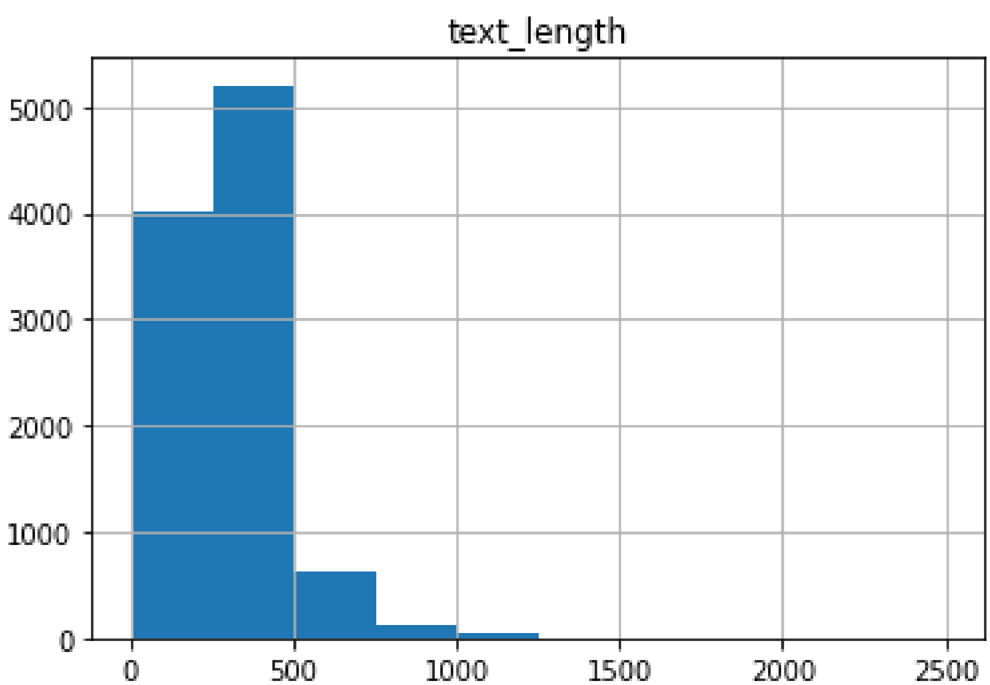
[13]:
```
count    7493.000000
mean      288.498065
std        99.546325
min       200.000000
25%       225.000000
50%       260.000000
75%       321.000000
max      3141.000000
Name: text_length, dtype: float64
```

[14]:
```
dickens_train.hist()
plt.show()
```



Similarly examine the text length & distribution in test data

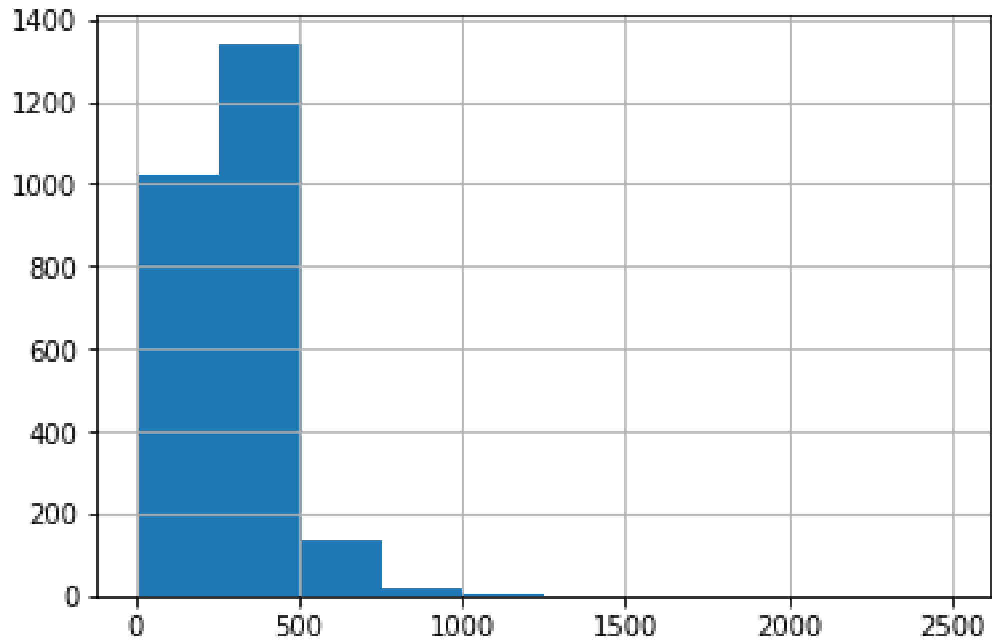
[15]:
```
test_data['text_length'] = test_data['text'].str.len()
test_data.hist()
plt.show()
```



[16]:
```
twain_test = test_data[test_data['author'] =='twain']['text_length']
twain_test.describe()
```

[16]:
```
count    2522.000000
mean      307.407613
std       125.125268
min         6.000000
25%       230.250000
50%       270.000000
75%       339.000000
max      2494.000000
Name: text_length, dtype: float64
```

[17]:
```
twain_test.hist()
plt.show()
```

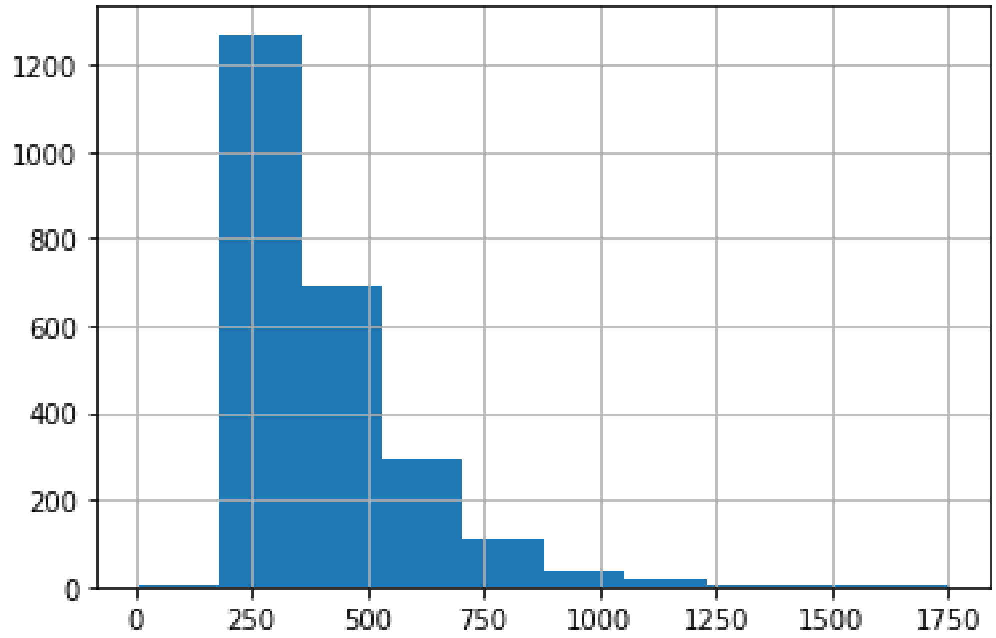


[18]:
```
defoe_test = test_data[test_data['author'] =='defoe']['text_length']
defoe_test.describe()
```

[18]:
```
count    2431.000000
mean      400.674619
std       189.809590
min         6.000000
```

```
25%       262.000000
50%       346.000000
75%       484.500000
max      1755.000000
Name: text_length, dtype: float64
```

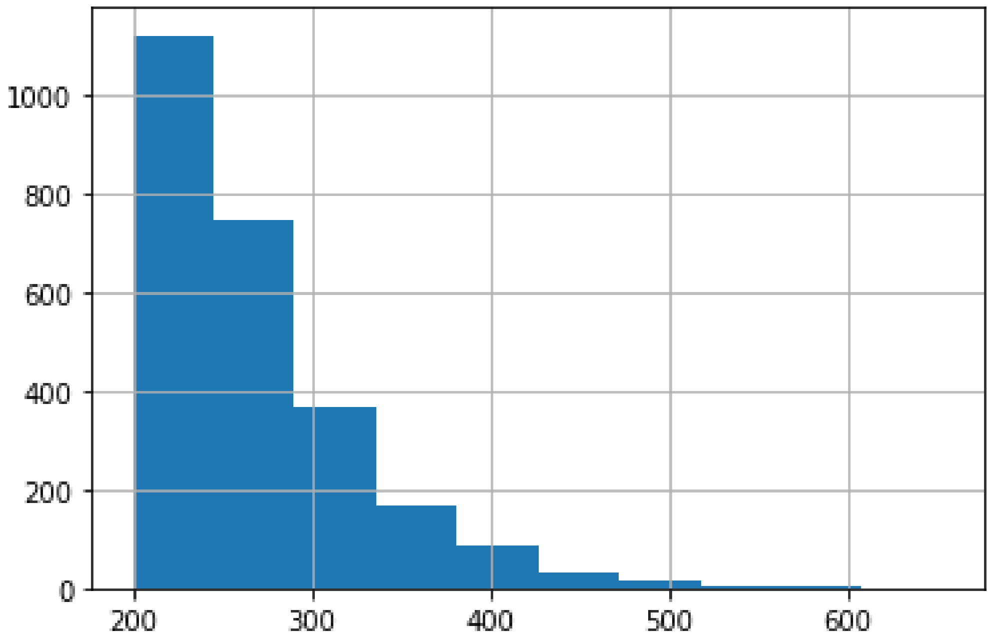
[19]:
```
defoe_test.hist()
plt.show()
```



[20]:
```
doyle_test = test_data[test_data['author'] =='doyle']['text_length']
doyle_test.describe()
```

[20]:
```
count    2540.000000
mean      268.123622
std        59.771638
min       200.000000
25%       224.000000
50%       254.000000
75%       295.000000
max       654.000000
Name: text_length, dtype: float64
```

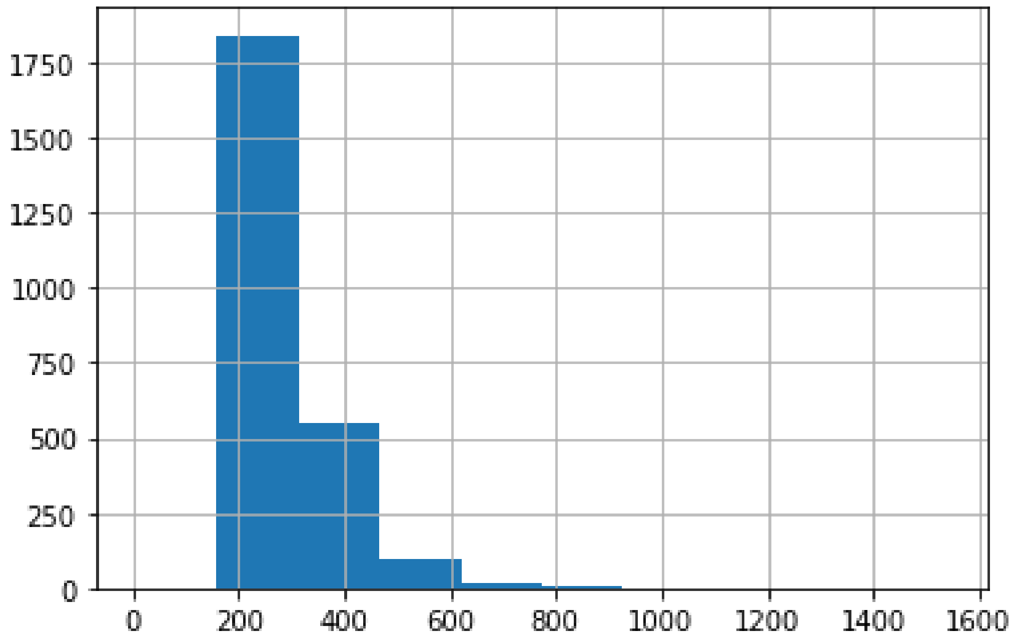
[21]:
```
doyle_test.hist()
plt.show()
```



[22]:
```
dickens_test = test_data[test_data['author'] =='dickens']['text_length']
dickens_test.describe()
```

[22]:
```
count    2507.000000
mean      286.886318
std        93.134536
min         6.000000
25%       226.000000
50%       260.000000
75%       317.000000
max      1541.000000
Name: text_length, dtype: float64
```

[23]:
```
dickens_test.hist()
plt.show()
```



Some preprocessing of the target variable to facilitate modelling

[24]:
```
# Encoding author label into numbers
train_data['author_num'] = train_data.author.map({'twain':0, 'defoe':1, 'doyle':
 ↪2, 'dickens':3})
train_data.head()
```

[24]:
```
                                                text   author  text_length  \
0   She wanted clothes to keep her warm, and food...  dickens          251
1   The question now was, who was the man,\nand w...    doyle          208
2   I therefore\n      smoked a great number of t...    doyle          444
3   I am partial to the modern\nFrench school. \n...    doyle          292
4  ” She stood smiling, holding up a little slip ...    doyle          227

   author_num
0           3
1           2
2           2
3           2
4           2
```

Limiting all text length to 700 characters for both train and test for less outliers in data

[25]:
```
train_data = train_data.rename(columns={'text':'original_text'})
train_data['text'] = train_data['original_text'].str[:700]
train_data['text_length'] = train_data['text'].str.len()
```

[26]:

```
test_data = test_data.rename(columns={'text':'original_text'})
test_data['text'] = test_data['original_text'].str[:700]
test_data['text_length'] = test_data['text'].str.len()
```

Define X and y from train data for use in tokenization by Vectorizers

[27]:

```
train_data
```

[27]:

```
                                            original_text   author  \
0       She wanted clothes to keep her warm, and food...  dickens
1       The question now was, who was the man,\nand w...    doyle
2       I therefore\n      smoked a great number of t...    doyle
3       I am partial to the modern\nFrench school. \n...    doyle
4      ’’ She stood smiling, holding up a little slip ...    doyle
...                                                  ...      ...
29995   It ain't anything.  There ain't no harm in it...    twain
29996    In my\nyouth the monarchs of England had cea...    twain
29997   Bob Sawyer nodded. \n\n‘So are you, sir,’ sai...  dickens
29998   He was out on the lawn, in through the window...    doyle
29999   ‘‘Here he is,’’ said he, sitting down and flatt...    doyle

       text_length  author_num  \
0              251           3
1              208           2
2              444           2
3              292           2
4              227           2
...            ...         ...
29995          429           0
29996          366           0
29997          297           3
29998          361           2
29999          201           2

                                                    text
0       She wanted clothes to keep her warm, and food...
1       The question now was, who was the man,\nand w...
2       I therefore\n      smoked a great number of t...
3       I am partial to the modern\nFrench school. \n...
4      ’’ She stood smiling, holding up a little slip ...
...                                                  ...
29995   It ain't anything.  There ain't no harm in it...
29996    In my\nyouth the monarchs of England had cea...
29997   Bob Sawyer nodded. \n\n‘So are you, sir,’ sai...
29998   He was out on the lawn, in through the window...
29999   ‘‘Here he is,’’ said he, sitting down and flatt...

[30000 rows x 5 columns]
```

[28]:
```
test_data
```

[28]:
```
                                         original_text   author  text_length  \
0      Carton,'' said the man of business.  ``Good nig...  dickens          214
1      _Is taken, and\nhow_, 154.  _Tried, condemned...    defoe          237
2      Through a cousin who\n      works with Gelder...    doyle          207
3     \n\nIndeed, nothing was more strange than to s...    defoe          282
4     \n\nOn the rocks above the present city of Alt...    twain          318
...                                                 ...      ...          ...
9995   I was very glad to\nsee her too, and, after a...  dickens          365
9996   ``And yet we manage to make ourselves fairly h...    doyle          205
9997   `Why, here they are. '\n\n`No, no; I mean the...  dickens          249
9998  ''\n\n``Was Peter Wilks well off?''\n\n``Oh, yes, ...    twain          225
9999  \n\n`Shall I go away, aunt?' I asked, tremblin...  dickens          214

                                                  text
0      Carton,'' said the man of business.  ``Good nig...
1      _Is taken, and\nhow_, 154.  _Tried, condemned...
2      Through a cousin who\n      works with Gelder...
3     \n\nIndeed, nothing was more strange than to s...
4     \n\nOn the rocks above the present city of Alt...
...                                                 ...
9995   I was very glad to\nsee her too, and, after a...
9996   ``And yet we manage to make ourselves fairly h...
9997   `Why, here they are. '\n\n`No, no; I mean the...
9998  ''\n\n``Was Peter Wilks well off?''\n\n``Oh, yes, ...
9999  \n\n`Shall I go away, aunt?' I asked, tremblin...

[10000 rows x 4 columns]
```

[29]:
```
X = train_data['text']
y = train_data['author_num']
```

Split train data into a training and a test se

[30]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=123)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(24000,) (24000,) (6000,) (6000,)
```

Examine the class distribution in y_train and y_test

[31]:
```
print(y_train.value_counts(),'\n', y_test.value_counts())
```

```
1    6087
3    5996
2    5962
```

```
0    5955
Name: author_num, dtype: int64
 0    1523
2    1498
3    1497
1    1482
Name: author_num, dtype: int64
```

Vectorize the data using Vectorizer

[32]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

[33]:
```
vect = CountVectorizer(lowercase=False, token_pattern=r'(?u)\b\w+\b|\,|\.|\;|\:')
vect
```

[33]:
```
CountVectorizer(lowercase=False, token_pattern='(?u)\\b\\w+\\b|\\,|\\.|\\;|\\:')
```

Learn the vocabulary in the training data, then use it to create a document-term matrix

[34]:
```
X_train_dtm = vect.fit_transform(X_train)
```

Examine the document-term matrix created from X_train

[35]:
```
print(X_train_dtm)
```

```
  (0, 5237)     3
  (0, 36190)    1
  (0, 39461)    2
  (0, 0)        5
  (0, 32712)    1
  (0, 36212)    1
  (0, 39463)    1
  (0, 26868)    1
  (0, 32387)    1
  (0, 13647)    3
  (0, 30931)    1
  (0, 28586)    1
  (0, 23975)    2
  (0, 28869)    1
  (0, 1)        4
  (0, 11058)    1
  (0, 23674)    2
  (0, 22834)    1
  (0, 27099)    1
  (0, 22363)    1
  (0, 26331)    1
  (0, 24118)    1
  (0, 33452)    1
  (0, 16422)    1
```

```
  (0, 19762)	1
  :	:
  (23999, 36236)	1
  (23999, 20944)	1
  (23999, 38860)	1
  (23999, 21128)	1
  (23999, 35895)	1
  (23999, 25748)	1
  (23999, 36235)	1
  (23999, 34918)	1
  (23999, 17682)	1
  (23999, 29765)	1
  (23999, 31392)	1
  (23999, 25865)	1
  (23999, 38342)	1
  (23999, 13408)	1
  (23999, 6043)	1
  (23999, 21110)	1
  (23999, 34251)	1
  (23999, 29237)	1
  (23999, 32855)	1
  (23999, 19964)	1
  (23999, 1553)	2
  (23999, 29844)	1
  (23999, 7130)	1
  (23999, 23885)	1
  (23999, 8166)	1
```

Transform the test data using the earlier fitted vocabulary, into a document-term matrix

[36]:
```
X_test_dtm = vect.transform(X_test)
```

Examine the document-term matrix from X_test

[37]:
```
print(X_test_dtm)
```

```
  (0, 0)	5
  (0, 1)	4
  (0, 1835)	1
  (0, 3436)	1
  (0, 5237)	3
  (0, 6883)	2
  (0, 6884)	1
  (0, 9403)	1
  (0, 12598)	1
  (0, 13016)	1
  (0, 13524)	1
  (0, 13528)	1
  (0, 13827)	1
```

```
  (0, 15964)	1
  (0, 17218)	1
  (0, 23491)	1
  (0, 23674)	2
  (0, 23945)	1
  (0, 26177)	1
  (0, 27099)	1
  (0, 27919)	1
  (0, 28315)	1
  (0, 28860)	1
  (0, 32712)	1
  (0, 35855)	1
  :	:
  (5999, 26261)	1
  (5999, 26463)	1
  (5999, 26636)	1
  (5999, 27099)	1
  (5999, 27671)	1
  (5999, 28390)	1
  (5999, 28509)	1
  (5999, 29976)	1
  (5999, 30330)	1
  (5999, 33507)	3
  (5999, 33611)	1
  (5999, 36075)	1
  (5999, 36117)	1
  (5999, 36189)	1
  (5999, 36204)	1
  (5999, 36212)	2
  (5999, 36333)	1
  (5999, 36360)	1
  (5999, 36600)	2
  (5999, 38626)	1
  (5999, 38860)	1
  (5999, 38867)	1
  (5999, 39025)	1
  (5999, 39445)	1
  (5999, 39468)	2
```

Add character counts as a features to the sparse matrix using function add_feature

[38]:
```
def add_feature(X, feature_to_add):
    from scipy.sparse import csr_matrix, hstack
    return hstack([X, csr_matrix(feature_to_add).T], 'csr')
```

[39]:
```
from string import punctuation
X_train_chars = X_train.str.len()
```

```
X_train_punc = X_train.apply(lambda x: len([c for c in str(x) if c in␣
↪punctuation]))
X_test_chars = X_test.str.len()
X_test_punc = X_test.apply(lambda x: len([c for c in str(x) if c in␣
↪punctuation]))
X_train_dtm = add_feature(X_train_dtm, [X_train_chars, X_train_punc])
X_test_dtm = add_feature(X_test_dtm, [X_test_chars, X_test_punc])
```

[40]:
```
print(X_train_dtm)
```

```
  (0, 0)	5
  (0, 1)	4
  (0, 4930)	1
  (0, 5237)	3
  (0, 10684)	1
  (0, 11058)	1
  (0, 13261)	1
  (0, 13465)	1
  (0, 13647)	3
  (0, 14073)	1
  (0, 14220)	1
  (0, 16281)	1
  (0, 16422)	1
  (0, 19762)	1
  (0, 21439)	1
  (0, 22363)	1
  (0, 22834)	1
  (0, 23422)	1
  (0, 23512)	1
  (0, 23674)	2
  (0, 23704)	1
  (0, 23975)	2
  (0, 24118)	1
  (0, 26047)	1
  (0, 26177)	1
  :	:
  (23999, 28509)	1
  (23999, 28586)	1
  (23999, 28758)	1
  (23999, 28777)	1
  (23999, 29237)	1
  (23999, 29765)	1
  (23999, 29844)	1
  (23999, 31392)	1
  (23999, 32855)	1
  (23999, 34251)	1
  (23999, 34918)	1
  (23999, 35895)	1
```

```
(23999, 36212)        1
(23999, 36228)        3
(23999, 36230)        1
(23999, 36235)        1
(23999, 36236)        1
(23999, 36274)        1
(23999, 36531)        1
(23999, 38342)        1
(23999, 38860)        1
(23999, 38966)        2
(23999, 39461)        1
(23999, 39503)        347
(23999, 39504)        5
```

[41]:
```
print(X_test_dtm)
```

```
(0, 0)        5
(0, 1)        4
(0, 1835)     1
(0, 3436)     1
(0, 5237)     3
(0, 6883)     2
(0, 6884)     1
(0, 9403)     1
(0, 12598)    1
(0, 13016)    1
(0, 13524)    1
(0, 13528)    1
(0, 13827)    1
(0, 15964)    1
(0, 17218)    1
(0, 23491)    1
(0, 23674)    2
(0, 23945)    1
(0, 26177)    1
(0, 27099)    1
(0, 27919)    1
(0, 28315)    1
(0, 28860)    1
(0, 32712)    1
(0, 35855)    1
:      :
(5999, 26636) 1
(5999, 27099) 1
(5999, 27671) 1
(5999, 28390) 1
(5999, 28509) 1
(5999, 29976) 1
```

```
  (5999, 30330)	1
  (5999, 33507)	3
  (5999, 33611)	1
  (5999, 36075)	1
  (5999, 36117)	1
  (5999, 36189)	1
  (5999, 36204)	1
  (5999, 36212)	2
  (5999, 36333)	1
  (5999, 36360)	1
  (5999, 36600)	2
  (5999, 38626)	1
  (5999, 38860)	1
  (5999, 38867)	1
  (5999, 39025)	1
  (5999, 39445)	1
  (5999, 39468)	2
  (5999, 39503)	288
  (5999, 39504)	7
```

Build and evaluate an author classification model using Multinomial Naive Bayes

[42]:
```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb
```

[42]:
```
MultinomialNB()
```

Tune hyperparameter alpha = [0.01, 0.1, 1, 10, 100]

[43]:
```
from sklearn.model_selection import GridSearchCV
grid_values = {'alpha':[0.01, 0.1, 1.0, 10.0, 100.0]}
grid_nb = GridSearchCV(nb, param_grid=grid_values, scoring='neg_log_loss')
grid_nb.fit(X_train_dtm, y_train)
grid_nb.best_params_
```

[43]:
```
{'alpha': 1.0}
```

Set with recommended hyperparameters

[44]:
```
nb = MultinomialNB(alpha=1.0)
```

Train the model using X_train_dtm & y_train

[45]:
```
nb.fit(X_train_dtm, y_train)
```

[45]:
```
MultinomialNB()
```

Make author predictions for X_test_dtm

[46]:
```
y_pred_test = nb.predict(X_test_dtm)
```

Accuracy

[47]:
```
from sklearn import metrics
metrics.accuracy_score(y_test, y_pred_test)
```

[47]:
```
0.9261666666666667
```

F1 score

[48]:
```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred_test)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.93      0.90      0.91      1523
           1       0.94      0.97      0.95      1482
           2       0.93      0.94      0.94      1498
           3       0.92      0.90      0.91      1497

    accuracy                           0.93      6000
   macro avg       0.93      0.93      0.93      6000
weighted avg       0.93      0.93      0.93      6000
```

Compute the accuracy of training data predictions

[49]:
```
y_pred_train = nb.predict(X_train_dtm)
metrics.accuracy_score(y_train, y_pred_train)
```

[49]:
```
0.9587083333333334
```

Look at the confusion matrix for y_test

[50]:
```
metrics.confusion_matrix(y_test, y_pred_test)
```

[50]:
```
array([[1371,   46,   50,   56],
       [  12, 1432,    8,   30],
       [  31,   15, 1414,   38],
       [  67,   37,   53, 1340]], dtype=int64)
```

Calculate predicted probabilities for X_test_dtm

[51]:
```
y_pred_prob = nb.predict_proba(X_test_dtm)
y_pred_prob[:10]
```

[51]:
```
array([[1.75701289e-12, 5.62686466e-18, 1.15173981e-09, 9.99999999e-01],
       [2.05188323e-09, 5.75150351e-13, 3.08776118e-06, 9.99996910e-01],
       [2.93878431e-01, 1.10697485e-05, 2.79557848e-04, 7.05830941e-01],
```

```
       [9.61738619e-01, 4.92103725e-08, 3.82613049e-02, 2.70085307e-08],
       [2.34526393e-06, 3.89169584e-08, 3.19793403e-07, 9.99997296e-01],
       [9.99328736e-01, 4.16798946e-07, 6.66253151e-04, 4.59382505e-06],
       [9.06365827e-08, 1.42241346e-04, 9.99836445e-01, 2.12234778e-05],
       [9.63120868e-09, 9.99999577e-01, 1.91982529e-08, 3.94366738e-07],
       [2.49280811e-19, 2.40764063e-17, 1.22969696e-08, 9.99999988e-01],
       [7.19508098e-09, 1.74787994e-10, 9.96407122e-01, 3.59287109e-03]])
```

Compute the log loss number

[52]:
```
metrics.log_loss(y_test, y_pred_prob)
```

[52]:
```
0.32429446190320216
```

[53]:
```
test = test_data['text']
# transform the test data using the earlier fitted vocabulary, into a␣
↪document-term matrix
test_dtm = vect.transform(test)
# examine the document-term matrix from X_test
test_dtm
```

[53]:
```
<10000x39503 sparse matrix of type '<class 'numpy.int64'>'
        with 463465 stored elements in Compressed Sparse Row format>
```

[54]:
```
test_chars = test.str.len()
test_punc = test.str.count(r'\W')
test_dtm = add_feature(test_dtm, [test_chars, test_punc])
test_dtm
```

[54]:
```
<10000x39505 sparse matrix of type '<class 'numpy.int64'>'
        with 483465 stored elements in Compressed Sparse Row format>
```

[55]:
```
NB_y_pred = nb.predict(test_dtm)
print(NB_y_pred)
```

```
[3 1 0 ... 3 0 3]
```

[56]:
```
NB_y_pred_prob = nb.predict_proba(test_dtm)
NB_y_pred_prob[:10]
```

[56]:
```
array([[1.14323576e-07, 5.19415896e-14, 2.66765160e-11, 9.99999886e-01],
       [1.57537590e-23, 1.00000000e+00, 1.29594645e-27, 1.28732480e-20],
       [8.13708962e-01, 1.58019702e-01, 2.51053415e-03, 2.57608013e-02],
       [1.06970687e-08, 9.99999779e-01, 2.15308600e-15, 2.10712891e-07],
       [9.99999999e-01, 8.71917762e-10, 2.46155020e-11, 3.05540480e-11],
       [7.10211074e-09, 9.99999993e-01, 9.44439407e-27, 2.02373928e-10],
       [1.35406099e-07, 2.77606748e-07, 9.99551625e-01, 4.47961669e-04],
       [4.51392973e-11, 1.01613453e-10, 1.72908678e-12, 1.00000000e+00],
       [7.82381656e-13, 9.28101475e-20, 5.12265002e-17, 1.00000000e+00],
```

```
[8.89002582e-01, 1.20911512e-08, 1.95103863e-07, 1.10997211e-01]])
```

[57]:
```
result = pd.DataFrame(NB_y_pred_prob,␣
 ↪columns=['defoe','dickens','twain','doyle'])
result.head()
```

[57]:

	defoe	dickens	twain	doyle
0	1.143236e-07	5.194159e-14	2.667652e-11	9.999999e-01
1	1.575376e-23	1.000000e+00	1.295946e-27	1.287325e-20
2	8.137090e-01	1.580197e-01	2.510534e-03	2.576080e-02
3	1.069707e-08	9.999998e-01	2.153086e-15	2.107129e-07
4	1.000000e+00	8.719178e-10	2.461550e-11	3.055405e-11

[ ]:

[ ]:

[ ]: