```
import sys
import re
import os
import requests
import pandas as pd
import numpy as np
from sklearn import metrics
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline

## save the book locally
if not os.path.exists('sherlock-holmes.txt'):
    text = requests.get('https://www.gutenberg.org/files/1661/1661-0.txt').text
    with open("sherlock-holmes.txt", "w") as text_file:
        text_file.write(text)

## read in the book
text = open('sherlock-holmes.txt', 'r').read()

## do some basic parsing and cleaning of sentences
stop_pattern = '\.|\?|\!'
sentences = re.split(stop_pattern, text)
sentences = [re.sub("\r|\n"," ",s.lower()) for s in sentences][3:]

text = requests.get('https://www.gutenberg.org/files/1661/1661-0.txt').text

## extract a few features and create a pandas df
stop_pattern = '\.|\?|\!'
sentences = re.split(stop_pattern, text)
sentences = [re.sub("\r|\n"," ",s.lower()) for s in sentences]
has_sherlock =  [True if re.search("sherlock|holmes",s) else False for s in sentences]
has_watson = [True if re.search("john|watson",s) else False for s in sentences]
df = pd.DataFrame({'text':sentences,'has_sherlock':has_sherlock,'has_watson':has_watson})

## extract the data to be used in the model from the df
labels = np.zeros(df.shape[0])
labels[(df['has_sherlock'] == True)] = 1
labels[(df['has_watson'] == True)] = 2
df['labels'] = labels
df = df[df['labels']!=0]
X = df['text'].values
y = df['labels'].values

## carry out the train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

from sklearn.linear_model import SGDClassifier
text_clf = Pipeline([
     ('vect', CountVectorizer()),
     ('tfidf', TfidfTransformer()),
     ('clf', SGDClassifier(loss='hinge', penalty='l2',
                           alpha=1e-3, random_state=42,
                           max_iter=5, tol=None))
    ])

## train a model
text_clf.fit(X_train, y_train)

## evaluate the model performance
predicted = text_clf.predict(X_test)
print(np.mean(predicted == y_test))
print(metrics.classification_report(y_test, predicted,
      target_names=['sherlock','watson']))
```