Step 1

We read in the file

```
import pandas as pd
df = pd.read_csv('federalist.csv')
print(df[:10])
df = df.astype({"author":'category'})
df['author'].value_counts()
```

```
          author                                               text
     0  HAMILTON  FEDERALIST. No. 1 General Introduction For the...
     1       JAY  FEDERALIST No. 2 Concerning Dangers from Forei...
     2       JAY  FEDERALIST No. 3 The Same Subject Continued (C...
     3       JAY  FEDERALIST No. 4 The Same Subject Continued (C...
     4       JAY  FEDERALIST No. 5 The Same Subject Continued (C...
     5  HAMILTON  FEDERALIST No. 6 Concerning Dangers from Disse...
     6  HAMILTON  FEDERALIST. No. 7 The Same Subject Continued (...
     7  HAMILTON  FEDERALIST No. 8 The Consequences of Hostiliti...
     8  HAMILTON  FEDERALIST No. 9 The Union as a Safeguard Agai...
     9   MADISON  FEDERALIST No. 10 The Same Subject Continued (...
     HAMILTON               49
     MADISON                15
     HAMILTON OR MADISON    11
     JAY                     5
     HAMILTON AND MADISON    3
     Name: author, dtype: int64
```

Step 2

We divide the data and show the shape

```
from sklearn.model_selection import train_test_split
import seaborn as sb
from sklearn import datasets
import matplotlib.pyplot as plt

X = df.text     # features
y = df.author    # targets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, rand

print('train size:', X_train.shape)
print('\ntest size:', X_test.shape)
```

```
     train size: (66,)

     test size: (17,)
```

Step 3

We remove stopwords and vectorize the data

```
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words = stopwords)
```

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

```
X_train = vectorizer.fit_transform(X_train) # fit the training data
X_test = vectorizer.transform(X_test) # transform only

print(X_train.toarray())
print('train size:', X_train.shape)
print('\ntest size:', X_test.shape)
```

```
[[0.         0.         0.02956872 ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.02275824 0.         0.        ]]
train size: (66, 7876)

test size: (17, 7876)
```

Step 4

We try a Bernoulli NB model

Accuracy: 59%

```
from sklearn.naive_bayes import BernoulliNB
naive_bayes = BernoulliNB()
naive_bayes.fit(X_train, y_train)

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusio
```

```
pred = naive_bayes.predict(X_test)
print(confusion_matrix(y_test, pred))
print('accuracy score: ', accuracy_score(y_test, pred))

from sklearn.metrics import classification_report
print(classification_report(y_test, pred))
```

```
[[10  0  0  0]
 [ 3  0  0  0]
 [ 2  0  0  0]
 [ 2  0  0  0]]
accuracy score:  0.5882352941176471
                    precision    recall  f1-score   support

           HAMILTON       0.59      1.00      0.74        10
HAMILTON OR MADISON       0.00      0.00      0.00         3
                JAY       0.00      0.00      0.00         2
            MADISON       0.00      0.00      0.00         2

           accuracy                           0.59        17
          macro avg       0.15      0.25      0.19        17
       weighted avg       0.35      0.59      0.44        17

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
```

Step 5

We adjust the max words the network looks at and also add in a bigram feature.

Accuracy: 94%

```
vectorizer = TfidfVectorizer(max_features=1000, ngram_range=(1,2), stop_words = stopwords)
X_train = vectorizer.fit_transform(X_train) # fit the training data
X_test = vectorizer.transform(X_test) # transform only

naive_bayes = BernoulliNB()
naive_bayes.fit(X_train, y_train)

pred = naive_bayes.predict(X_test)
print(confusion_matrix(y_test, pred))
print('accuracy score: ', accuracy_score(y_test, pred))
print(classification_report(y_test, pred))
```

```
[[10  0  0  0]
 [ 0  3  0  0]
```

```
 [ 1  0  1  0]
 [ 0  0  0  2]]
accuracy score:  0.9411764705882353
                     precision    recall  f1-score   support

           HAMILTON       0.91      1.00      0.95        10
HAMILTON OR MADISON       1.00      1.00      1.00         3
                JAY       1.00      0.50      0.67         2
            MADISON       1.00      1.00      1.00         2

           accuracy                           0.94        17
          macro avg       0.98      0.88      0.90        17
       weighted avg       0.95      0.94      0.93        17
```

Step 6

We try logistic regression with adjusted variables

Accuracy without variable adjustment: 59%

Accuracy with adjustment: 71%

```
from sklearn.linear_model import LogisticRegression

X_train = vectorizer.fit_transform(X_train) # fit the training data
X_test = vectorizer.transform(X_test) # transform only
classifier = LogisticRegression(C=2)
classifier.fit(X_train, y_train)

pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred, average='micro'))
print('recall score: ', recall_score(y_test, pred, average='micro'))
print('f1 score: ', f1_score(y_test, pred, average='micro'))
probs = classifier.predict_proba(X_test)
```

```
accuracy score:  0.7058823529411765
precision score:  0.7058823529411765
recall score:  0.7058823529411765
f1 score:  0.7058823529411765
```

```
# Step 7
We try a neural network

Final accuracy: 82%
```

Step 7

We try a neural network

Final accuracy: 82%

```
from sklearn.neural_network import MLPClassifier
classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(4,2), max_iter=300, random_state=1234)
classifier.fit(X_train, y_train)
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred, average='micro'))
print('recall score: ', recall_score(y_test, pred, average='micro'))
print('f1 score: ', f1_score(y_test, pred, average='micro'))
```

```
accuracy score:  0.8235294117647058
precision score:  0.8235294117647058
recall score:  0.8235294117647058
f1 score:  0.8235294117647058
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

Colab paid products - Cancel contracts here

✓ 2s completed at 8:24 AM