In [ ]:
```
import pandas as pd
import nltk
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\josho\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[ ]:
```
True
```

In [ ]:
```
df = pd.read_csv('federalist.csv', dtype={'author' : 'category'})
```

In [ ]:
```
print('rows and columns:', df.shape)
print(df.head())
```

```
rows and columns: (83, 2)
     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...
```

In [ ]:
```
df['author'].value_counts()
```

Out[ ]:
```
HAMILTON               49
MADISON                15
HAMILTON OR MADISON    11
JAY                     5
HAMILTON AND MADISON    3
Name: author, dtype: int64
```

In [ ]:
```
X = df.text
y = df.author

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8
print("X_train:", X_train.shape)
print("X_test:",X_test.shape)
print("y_train:",y_train.shape)
print("y_test:",y_test.shape)
```

```
X_train: (66,)
X_test: (17,)
y_train: (66,)
y_test: (17,)
```

In [ ]:
```
stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words = stopwords, binary = True)
```

In [ ]:
```
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
print("X_train:", X_train.shape)
print("X_test:",X_test.shape)
```

```
X_train: (66, 7876)
X_test: (17, 7876)
```

In [ ]:
```
from sklearn.naive_bayes import BernoulliNB
```

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

Out[ ]:
```
▾ BernoulliNB

BernoulliNB()
```

In [ ]:
```
pred = naive_bayes.predict(X_test)

from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, pred)
```

Out[ ]:
```
array([[10,  0,  0,  0],
       [ 3,  0,  0,  0],
       [ 2,  0,  0,  0],
       [ 2,  0,  0,  0]], dtype=int64)
```

In [ ]:
```
from sklearn.metrics import accuracy_score
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.5882352941176471
```

In [ ]:
```
X = df.text
y = df.author

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8
vectorizer = TfidfVectorizer(stop_words = stopwords, binary = True, max_features = 1000
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
print("X_train:", X_train.shape)
print("X_test:",X_test.shape)
```

```
X_train: (66, 1000)
X_test: (17, 1000)
```

In [ ]:
```
naive_bayes = BernoulliNB()
naive_bayes.fit(X_train, y_train)
```

Out[ ]:
```
▾ BernoulliNB

BernoulliNB()
```

In [ ]:
```
pred = naive_bayes.predict(X_test)

from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, pred)
```

Out[ ]:
```
array([[10,  0,  0,  0],
       [ 0,  2,  0,  1],
       [ 1,  0,  1,  0],
       [ 0,  0,  0,  2]], dtype=int64)
```

In [ ]:
```
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.8823529411764706
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression
X = df.text
y = df.author
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8
vectorizer = TfidfVectorizer(stop_words = stopwords, binary = True, max_features = 1000
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
print("X_train:", X_train.shape)
print("X_test:",X_test.shape)
```

```
X_train: (66, 1000)
X_test: (17, 1000)
```

In [ ]:
```
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
```

Out[ ]:
```
LogisticRegression()
```

In [ ]:
```
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.5882352941176471
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression
X = df.text
y = df.author

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8
vectorizer = TfidfVectorizer(stop_words = stopwords, binary = True, max_features = 1000
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
print("X_train:", X_train.shape)
print("X_test:",X_test.shape)
```

```
X_train: (66, 1000)
X_test: (17, 1000)
```

In [ ]:
```
classifier = LogisticRegression(class_weight='balanced')
classifier.fit(X_train, y_train)
```

Out[ ]:
```
LogisticRegression(class_weight='balanced')
```

In [ ]:
```
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.8823529411764706
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression
X = df.text
y = df.author

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8
vectorizer = TfidfVectorizer(stop_words = stopwords, binary = True, max_features = 1000
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
print("X_train:", X_train.shape)
print("X_test:",X_test.shape)
```

```
X_train: (66, 1000)
```

```
X_test: (17, 1000)
```

In [ ]:
```
from sklearn.neural_network import MLPClassifier

classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(15, 8), random_state=1)
classifier.fit(X_train, y_train)
```

Out[ ]:
```
MLPClassifier

MLPClassifier(alpha=1e-05, hidden_layer_sizes=(15, 8), random_state=1,
              solver='lbfgs')
```

The topology that gave me the best results was (15,8)

In [ ]:
```
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.7058823529411765
```