```
import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/KinseyMellon/CS4395-HLT/main/federalist.c
df['author'] = pd.Categorical(df.author)
print(df.dtypes)
print(df.head())
print(df['author'].value_counts())
```

```
author    category
text        object
dtype: object
     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...
HAMILTON               49
MADISON                15
HAMILTON OR MADISON    11
JAY                     5
HAMILTON AND MADISON    3
Name: author, dtype: int64
```

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

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, rand
print('train size before tfidf vectorizer: ',X_train.shape)
print('test size before tfidf vectorizer: ',X_test.shape)

stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True)

X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)


print('train size after tfidf vectorizer: ',X_train.shape)
print('test size after tfidf vectorizer: ',X_test.shape)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
train size before tfidf vectorizer:  (66,)
test size before tfidf vectorizer:  (17,)
```

```
train size after tfidf vectorizer:  (66, 7876)
test size after tfidf vectorizer:  (17, 7876)
```

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

naive_bayes = BernoulliNB()
naive_bayes.fit(X_train,y_train)
pred = naive_bayes.predict(X_test)

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

```
accuracy score:  0.5882352941176471
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords


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, rand

stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True, max_features=1000, ngram_rang

X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

naive_bayes = BernoulliNB()
naive_bayes.fit(X_train,y_train)
pred = naive_bayes.predict(X_test)

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

```
accuracy score:  0.8823529411764706
```

With the new train and test sets the accuracy increased about 30%

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords

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, rand

stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True, max_features=1000, ngram_rang

X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

logReg = LogisticRegression()
logReg.fit(X_train,y_train)
pred = logReg.predict(X_test)
print("accuracy score with no parameters: ", accuracy_score(y_test,pred))

logReg2 = LogisticRegression(solver='lbfgs', class_weight='balanced', multi_class='multinomia
logReg2.fit(X_train,y_train)
pred3 = logReg2.predict(X_test)
print("accuracy score with parameters: ", accuracy_score(y_test,pred3))
```

```
accuracy score with no parameters:  0.5882352941176471
accuracy score with parameters:  0.8823529411764706
```

The logistic regression model before adding any parameters got around 58% but I added the class weight parameter since I noticed that the data is very skewed and I changed the solver too to one that works well with multiclass data and got an increase in accuracy of about 30%.

```
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

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, rand

stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True, max_features=1000, ngram_rang

X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

#classifier = MLPClassifier(solver='lbfgs',hidden_layer_sizes=(100,20,2),max_iter=500)
classifier = MLPClassifier(solver='lbfgs',hidden_layer_sizes=(100,20,2),max_iter=500,random_s
classifier.fit(X_train,y_train)
pred4 = classifier.predict(X_test)
```

```
print('accuracy score: ', accuracy_score(y_test, pred4))
```

```
accuracy score:  0.7647058823529411
```

The highest final accuracy I could get after trying mutliple different topologies was 76%