In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

In [2]:
```
federalist = pd.read_csv('federalist.csv')
```

In [3]:
```
# change author to a category variable and show the first few rows in datafram
e and how many each author wrote
federalist.author = federalist.author.astype('category')
print(federalist.head(10))
print(federalist['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
```

In [4]:
```
#split into X and Y
X = federalist['text']
Y = federalist['author']
```

In [5]:
```
#do train test split and print the shape of the Train and test
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, rando
m_state=1234)
print(X_train.shape)
print(X_test.shape)
```

```
(66,)
(17,)
```

In [6]:
```
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

#tfidf vectorizer and remove stopwords
stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords)

#fit-transform the train and only transform the test
Xtrain = vectorizer.fit_transform(X_train)
Xtest = vectorizer.transform(X_test)

#print shape of train and test
print(Xtrain.shape)
print(Xtest.shape)
```

```
(66, 7876)
(17, 7876)
```

In [7]:
```
from sklearn.naive_bayes import BernoulliNB
#create bernoulli naive bayes model and fit to training data that was vectoriz
ed
bnb = BernoulliNB()
bnb.fit(Xtrain, y_train)


from sklearn.metrics import accuracy_score
#get accuracy score of model by predicting the test data
pred = bnb.predict(Xtest)
print(accuracy_score(y_test, pred))
print("The accuracy on the test set for the Bernoulli Model is 0.59")
```

```
0.5882352941176471
The accuracy on the test set for the Bernoulli Model is 0.59
```

In [8]:
```
#adjust vectorizer based on criteria and do same process for train and test da
ta as above
vectorizer2 = TfidfVectorizer(stop_words=stopwords, max_features = 1000, ngram
_range = (1,2))
Xtrain2 = vectorizer2.fit_transform(X_train)
Xtest2 = vectorizer2.transform(X_test)

#fit anouther Bernoulli Naive Bayes model to the new vectorizer's train data
bnb2 = BernoulliNB()
bnb2.fit(Xtrain2, y_train)

#get accuracy score of second model by predicting the new test data
pred2 = bnb2.predict(Xtest2)
print(accuracy_score(y_test, pred2))
print("The accuracy on the test set for the Bernoulli Model with the improved
 vectorizer is 0.94.",
     "Compared to the previous model,  this model did significantly better wit
h the concentrated features,",
      "and using bigrams as well.")
```

```
0.9411764705882353
The accuracy on the test set for the Bernoulli Model with the improved vector
izer is 0.94. Compared to the previous model,  this model did significantly b
etter with the concentrated features, and using bigrams as well.
```

In [9]:
```
from sklearn.linear_model import LogisticRegression
#try plain logistric regression on the training data and get accuracy
classifierlognorm = LogisticRegression()
classifierlognorm.fit(Xtrain2, y_train)

predlognorm = classifierlognorm.predict(Xtest2)
print(accuracy_score(y_test, predlognorm))
```

```
0.5882352941176471
```

In [10]:
```
#adjust hyperparameters and see if logistic regression accuracy improves
classifierlog = LogisticRegression(solver='sag', class_weight='balanced', rand
om_state = 1234)
classifierlog.fit(Xtrain2, y_train)

predlog = classifierlog.predict(Xtest2)
print(accuracy_score(y_test, predlog))

print("By adjusting parameters over the model with no parameters improves the
 accuracy significantly. \n",
     "The previous logistic regression model with no parameters had an accurac
y of 0.59, whereas the accuracy of",
     "the model with  adjusted parameters has an accuracy of 0.76.")
```

```
0.7647058823529411
By adjusting parameters over the model with no parameters improves the accura
cy significantly.
 The previous logistic regression model with no parameters had an accuracy of
0.59, whereas the accuracy of the model with  adjusted parameters has an accu
racy of 0.76.
```

In [11]:
```
#The next few code chunks are trying out various neural networks in which the
 MLPClassifier tries to get the best
#accuracy by changing the topology

from sklearn.neural_network import MLPClassifier
classifiernn1 = MLPClassifier(solver='lbfgs',alpha=1e-5,
                    hidden_layer_sizes=(12,1000), random_state=1234)
classifiernn1.fit(Xtrain2, y_train)

prednn1 = classifiernn1.predict(Xtest2)
print(accuracy_score(y_test, prednn1))
```

```
0.7058823529411765
```

In [12]:
```
classifiernn2 = MLPClassifier(solver='lbfgs',alpha=1e-5,
                    hidden_layer_sizes=(15,1000), random_state=1234)
classifiernn2.fit(Xtrain2, y_train)

prednn2 = classifiernn2.predict(Xtest2)
print(accuracy_score(y_test, prednn2))
```

```
0.7647058823529411
```

In [13]:
```
classifiernn3 = MLPClassifier(solver='adam',alpha=1e-5,
                    hidden_layer_sizes=(12,1000), random_state=1234)
classifiernn3.fit(Xtrain2, y_train)

prednn3 = classifiernn3.predict(Xtest2)
print(accuracy_score(y_test, prednn3))
```

```
0.8235294117647058
```

In [14]:
```
classifiernn = MLPClassifier(solver='adam',alpha=1e-5,
                    hidden_layer_sizes=(15,1000), random_state=1234)
classifiernn.fit(Xtrain2, y_train)

prednn = classifiernn.predict(Xtest2)
print(accuracy_score(y_test, prednn))
```

```
0.8823529411764706
```

In [15]:
```
print("After trying various methods, the MLPClassifier with an adam solver and
a hidden_layer_sizes = (15,1000)",
     "seemed to work the best and had an accuracy of 0.88")
```

```
After trying various methods, the MLPClassifier with an adam solver and a hid
den_layer_sizes = (15,1000) seemed to work the best and had an accuracy of 0.
88
```

In [ ]:
```

```