In [57]:
```
import pandas as pd
df = pd.read_csv('federalist.csv') #reads in csv
df.head() #prints first 5 elements
df.author = df.author.astype('category')
df.author.value_counts()
```

Out[57]:
```
HAMILTON               49
MADISON                15
HAMILTON OR MADISON    11
JAY                     5
HAMILTON AND MADISON    3
Name: author, dtype: int64
```

In [58]:
```
X = df.text
y = df.author
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1234, stratify = y) #
print("X_train shape: " + str(X_train.shape))
print("X_test shape: " + str(X_test.shape))
print("y_train shape: " + str(y_train.shape))
print("y_test shape: " + str(y_test.shape))
```

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

In [59]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words = 'english') #creates vectorizer
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
print("X_train shape: " + str(X_train.shape))
print("X_test shape: " + str(X_test.shape))
```

```
X_train shape: (66, 7532)
X_test shape: (17, 7532)
```

In [60]:
```
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score
nb = BernoulliNB()
nb.fit(X_train, y_train)
nbPredict = nb.predict(X_test) #performs Beroulli Naive Bayes
print("Bernoulli Naive Bayes Accuracy: ", accuracy_score(y_test, nbPredict))
```

```
Bernoulli Naive Bayes Accuracy:  0.5882352941176471
```

In [61]:
```
vectorizer = TfidfVectorizer(stop_words = 'english', max_features= 1000, ngram_range= (1,2)) #creates new vecto
X = df.text #recreates variables for splitting
y = df.author
X_train, X_test, y_train, y_test= train_test_split(X, y, test_size = 0.2, random_state=1234) #resplits data
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
nb = BernoulliNB() #creates new Bernoulli Naive Bayes
nb.fit(X_train, y_train)
nbPredict = nb.predict(X_test) #performs Bernoulli Naive Bayes
print("New Bernoulli Naive Bayes Accuracy: ", accuracy_score(y_test, nbPredict))
```

```
New Bernoulli Naive Bayes Accuracy:  0.9411764705882353
```

In [62]:
```
from sklearn.linear_model import LogisticRegression
lrPlain = LogisticRegression() #creates plain Logistic Regression object
lrPlain.fit(X_train, y_train)
lrPlainPredict = lrPlain.predict(X_test) #runs plain Logistic Regression object
print("Plain Logistic Regression Accuracy: ", accuracy_score(y_test, lrPlainPredict))
parameters = LogisticRegression(class_weight = 'balanced') #creates Logistic Regression object with a parameter
parameters.fit(X_train, y_train)
lrParameterPredict = parameters.predict(X_test) #runs Logistic Regression with parameter
print("Parameter Logistic Regression Accuracy: ", accuracy_score(y_test, lrParameterPredict))
```

```
Plain Logistic Regression Accuracy:  0.5882352941176471
Parameter Logistic Regression Accuracy:  0.7058823529411765
```

In [63]:
```
from sklearn.neural_network import MLPClassifier
nn = MLPClassifier(random_state = 1234, hidden_layer_sizes = (1000,)) #creates neural network object
nn.fit(X_train, y_train)
nnPredict = nn.predict(X_test) #runs neural network
print("Neural Network Accuracy: ", accuracy_score(y_test, nnPredict))
```

```
Neural Network Accuracy:  0.7647058823529411
```