In [1]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
```

1

In [2]:
```python
data = pd.read_csv("federalist.csv")
data['author'] = data.author.astype('category')
data.head()
```

Out[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 [3]:
```python
data.author.value_counts()
```

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

In [4]:
```python
data.dtypes
```

Out[4]:
```
author    category
text        object
dtype: object
```

2

In [5]:
```python
X = data.text
y = data.author

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

Out[5]:
```
((66,), (17,), (66,), (17,))
```

3

In [6]:
```
vectorizer = TfidfVectorizer(stop_words='english')

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

print('train size:', X_train.shape)
print(X_train.toarray()[:5])

print('\ntest size:', X_test.shape)
print(X_test.toarray()[:5])
```

```
train size: (66, 7727)
[[0.         0.         0.03056353 ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.03999681 0.         0.        ]]

test size: (17, 7727)
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.02406012 0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

4

In [7]:
```
naive_bayes = BernoulliNB()
naive_bayes.fit(X_train, y_train)
```

Out[7]:
```
BernoulliNB()
```

In [8]:
```
pred = naive_bayes.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.5882352941176471
```

5

In [9]:
```
X = data.text
y = data.author
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[9]:
```
((66,), (17,), (66,), (17,))
```

In [10]:
```
vectorizer = TfidfVectorizer(stop_words='english', max_features=1000, ngram_range=(1,2)
# vectorize
X_train = vectorizer.fit_transform(X_train) # fit the training data
X_test = vectorizer.transform(X_test) # transform only

print('train size:', X_train.shape)
print(X_train.toarray()[:5])

print('\ntest size:', X_test.shape)
print(X_test.toarray()[:5])
```

```
train size: (66, 1000)
[[0.02001784 0.02001784 0.01929126 ... 0.03268402 0.01766216 0.        ]
 [0.02282457 0.02282457 0.         ... 0.03726669 0.02013859 0.        ]
 [0.01136023 0.01136023 0.         ... 0.10510731 0.01002337 0.        ]
 [0.         0.         0.         ... 0.01044773 0.         0.        ]
 [0.01676058 0.01676058 0.08076111 ... 0.01824383 0.01478821 0.04437088]]

test size: (17, 1000)
[[0.02956605 0.02956605 0.0569858  ... 0.03218254 0.02608674 0.        ]
 [0.0233583  0.0233583  0.02251047 ... 0.02542542 0.02060951 0.03091864]
 [0.02531887 0.02531887 0.02439988 ... 0.02755951 0.02233937 0.        ]
 [0.0241714  0.0241714  0.         ... 0.02631049 0.02132693 0.        ]
 [0.01918632 0.01918632 0.01848992 ... 0.02088424 0.01692849 0.        ]]
```

In [11]:

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

Out[11]:

```
BernoulliNB()
```

In [12]:

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

```
accuracy score:  0.9411764705882353
```

6

In [13]:

```
classifier = LogisticRegression()
classifier.fit(X_train, y_train)

# evaluate
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.5882352941176471
```

In [14]:

```
classifier = LogisticRegression(solver='lbfgs', class_weight='balanced')
classifier.fit(X_train, y_train)

# evaluate
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.7058823529411765
```

7

```
In [15]:
classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                           hidden_layer_sizes=(100,), random_state=1)
classifier.fit(X_train, y_train)

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

```
accuracy score:  0.7058823529411765
```

```
In [16]:
classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                           hidden_layer_sizes=(50,), random_state=1)
classifier.fit(X_train, y_train)

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

```
accuracy score:  0.7647058823529411
```