

```
In [1]: import pandas as pd
```

```
In [2]: #1
df = pd.read_csv("federalist.csv") ## read in the csv
df['author'] = df.author.astype('category') ## convert the author column to categorical
df.head() ## display the first few rows
```

```
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]: df['author'].value_counts() ## Display the counts by author
```

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

```
In [4]: #2
x = df.text ## x var
y = df.author ## y var
```

```
In [5]: 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, train_size=0.8)
## display the shapes of the trains and tests
print('Shape of x_train : ' + str(x_train.shape))
print('Shape of y_train : ' + str(y_train.shape))
print('Shape of x_test : ' + str(x_test.shape))
print('Shape of y_test : ' + str(y_test.shape))
```

```
Shape of x_train : (66,)
Shape of y_train : (66,)
Shape of x_test : (17,)
Shape of y_test : (17,)
```

```
In [6]: #3
from nltk.corpus import stopwords
import re
from sklearn.feature_extraction.text import TfidfVectorizer
stopwords = set(stopwords.words('english')) ## set the stopwords
```

```
In [7]: vectorizer = TfidfVectorizer(stop_words=stopwords) ## create the vectorizer
```

```
x_train_vec = vectorizer.fit_transform(x_train) ## vectorize the training
x_test_vec = vectorizer.transform(x_test) ## vectorize the test
```

```
In [8]: ## display the new shapes
print('Shape of x_train : ' + str(x_train_vec.shape))
print('Shape of x_test : ' + str(x_test_vec.shape))
```

Shape of x_train : (66, 7876)
Shape of x_test : (17, 7876)

```
In [9]: #4
from sklearn.naive_bayes import BernoulliNB
naive_bayes = BernoulliNB() ## create the model
naive_bayes.fit(x_train_vec, y_train) ## train the model
```

```
Out[9]: ▾ BernoulliNB
BernoulliNB()
```

```
In [10]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, co
pred = naive_bayes.predict(x_test_vec) ## predict on the test data
print('Accuracy for Bernoulli Naive Bayes: ', accuracy_score(y_test, pred)) ## output t
```

Accuracy for Bernoulli Naive Bayes: 0.5882352941176471

```
In [11]: #5
vectorizer = TfidfVectorizer(stop_words=stopwords, max_features = 1000, ngram_range = (
x_train_vec = vectorizer.fit_transform(x_train) ## vectorize the train
x_test_vec = vectorizer.transform(x_test) ## vectorize the test
```

```
In [12]: naive_bayes = BernoulliNB() ## create the model
naive_bayes.fit(x_train_vec, y_train) ## train the model
pred = naive_bayes.predict(x_test_vec) ## predict on the test data
print('Accuracy for Bernoulli Naive Bayes with max features and ngrams: ', accuracy_sco
```

Accuracy for Bernoulli Naive Bayes with max features and ngrams: 0.9411764705882353

```
In [13]: #6
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

pipe1 = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('logreg', LogisticRegression(multi_class='multinomial', solver='lbfgs', class_w
]) ## create the model pipeline with the vectorizer and the logistic regression model
## here we change the solver to lbfgs and change the class weight to balanced

pipe1.fit(x_train, y_train) ## fit the model
pred = pipe1.predict(x_test) ## predict on the test data
import numpy as np
print("Accuracy for Logistic regression: ", np.mean(pred==y_test)) ## output the accura
```

Accuracy for Logistic regression: 0.9411764705882353

In [14]:

```
#7
from sklearn.neural_network import MLPClassifier

pipe1 = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('neuralnet', MLPClassifier(solver='lbfgs', alpha=1e-5,
                               hidden_layer_sizes=(10, 5), random_state=1234)),
])## create the model pipeline with the vectorizer and the neural network mode
## here we add 5 hidden layers with 10 nodes
pipe1.fit(x_train, y_train) ## fit the model
pred = pipe1.predict(x_test) ## predict on the test data
print("Accuracy for Neural Network regression: ", np.mean(pred==y_test)) ## output the
```

Accuracy for Neural Network regression: 0.5882352941176471