Author Attribution

Read and Explore the Data

```
import pandas as pd
df = pd.read_csv('federalist.csv')
df.author = df.author.astype('category')
df.head()
```

	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...

```
df.author.value_counts()
```

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

Divide into Train and Test

```
from sklearn.model_selection import train_test_split

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, random_state=1234)

print('Shape of training data:', X_train.shape)
print('Shape of testing data:', X_test.shape)
```

```
Shape of training data: (66,)
Shape of testing data: (17,)
```

Performing TF-IDF Vectorization

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

stopwords = set(stopwords.words('english'))
vectorizer1 = TfidfVectorizer(stop_words=stopwords)

X_train1 = vectorizer1.fit_transform(X_train)
X_test1 = vectorizer1.transform(X_test)
```

```
print('Shape of training data:', X_train1.shape)
print('Shape of testing data:', X_test1.shape)
```

```
Shape of training data: (66, 7876)
Shape of testing data: (17, 7876)
```

Bernoulli Naïve Bayes Model 1

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

naive_bayes = MultinomialNB()
naive_bayes.fit(X_train1, y_train)

pred1 = naive_bayes.predict(X_test1)
print('Naïve Bayes Model 1 Accuracy:', accuracy_score(y_test, pred1))
```

```
Naïve Bayes Model 1 Accuracy: 0.5882352941176471
```

Redo the Vectorization

```
vectorizer2 = TfidfVectorizer(stop_words=stopwords, max_features=1000, ngram_range=(2,2))
X_train2 = vectorizer2.fit_transform(X_train)
X_test2 = vectorizer2.transform(X_test)

print('Shape of training data:', X_train2.shape)
print('Shape of testing data:', X_test2.shape)
```

```
Shape of training data: (66, 1000)
Shape of testing data: (17, 1000)
```

Bernoulli Naïve Bayes Model 2

```
naive_bayes = MultinomialNB()
naive_bayes.fit(X_train2, y_train)

pred2 = naive_bayes.predict(X_test2)
print('Naïve Bayes Model 2 Accuracy:', accuracy_score(y_test, pred2))
```

```
Naïve Bayes Model 2 Accuracy: 0.5882352941176471
```

Comparison:

Both Bernoulli Naïve Bayes models performed similar, despite the different vectorization of the training data.

Logistic Regression Model 1

```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression()
classifier.fit(X_train2, y_train)
```

```
pred = classifier.predict(X_test2)
print('Logistic Regression Model 1 Accuracy:', accuracy_score(y_test, pred))
```

```
Logistic Regression Model 1 Accuracy: 0.5882352941176471
```

Logistic Regression Model 2

```
classifier = LogisticRegression(class_weight='balanced')
classifier.fit(X_train2, y_train)

pred = classifier.predict(X_test2)
print('Logistic Regression Model 2 Accuracy:', accuracy_score(y_test, pred))
```

```
Logistic Regression Model 2 Accuracy: 0.7647058823529411
```

Comparison:

Logistic Regression model with balanced weights performed 18% better than the model with unbalanced weights.

Neural Network Model 1

```
from sklearn.neural_network import MLPClassifier

classifier = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(7,6))
classifier.fit(X_train2, y_train)

pred = classifier.predict(X_test2)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.7647058823529411
```

Neural Network Model 2

```
classifier = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(13))
classifier.fit(X_train2, y_train)

pred = classifier.predict(X_test2)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.8235294117647058
```

Final Accuracy:

After running all three algorithms with different parameters, the best accuracy of 82.3% was achieved by the Neural Network with 1 hidden layer and 13 nodes.

Colab paid products - Cancel contracts here

 0s completed at 5:05 PM