```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import nltk
nltk.download('all')
from nltk.corpus import stopwords
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusic
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.neural_network import MLPClassifier
```

```
df = pd.read_csv('federalist.csv')#, index_col='author')
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...

```
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, ranc
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

```
stopwordSet = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwordSet)
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, 7876)
[[0.         0.         0.02956872 ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.03741484 0.         0.        ]]

test size: (17, 7876)
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.02314673 0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

```
naive_bayes = BernoulliNB()
naive_bayes.fit(X_train, y_train)
print(naive_bayes.class_log_prior_[1])
print(naive_bayes.feature_log_prob_)
```

```
-3.091042453358315
[[-3.02042489 -2.61495978 -2.61495978 ... -1.9218126  -3.71357207
  -2.32727771]
 [-1.60943791 -1.60943791 -0.91629073 ... -0.91629073 -0.91629073
  -1.60943791]
 [-2.30258509 -2.30258509 -2.30258509 ... -1.2039728  -2.30258509
  -2.30258509]
 [-1.60943791 -1.60943791 -1.60943791 ... -1.60943791 -1.60943791
  -1.60943791]
 [-2.7080502  -2.01490302 -2.01490302 ... -0.62860866 -2.7080502
  -2.01490302]]
```

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

```
[[10  0  0  0]
 [ 3  0  0  0]
 [ 2  0  0  0]
 [ 2  0  0  0]]
accuracy score:  0.5882352941176471
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, ranc
vectorizer = TfidfVectorizer(stop_words=stopwordSet, ngram_range=(1,2), max_features=1000)
X_train = vectorizer.fit_transform(X_train) # fit the training data
X_test = vectorizer.transform(X_test) # transform only
```

```
naive_bayes.fit(X_train, y_train)
print(naive_bayes.class_log_prior_[1])
print(naive_bayes.feature_log_prob_)

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

```
-3.091042453358315
[[-1.00552187 -1.00552187 -0.71783979 ... -0.02469261 -0.71783979
  -1.9218126 ]
 [-1.60943791 -1.60943791 -1.60943791 ... -0.22314355 -0.91629073
  -0.91629073]
 [-0.22314355 -0.22314355 -0.91629073 ... -0.10536052 -0.22314355
  -1.2039728 ]
 [-1.60943791 -1.60943791 -0.51082562 ... -0.22314355 -1.60943791
  -1.60943791]
 [-0.62860866 -0.62860866 -0.76214005 ... -0.06899287 -0.51082562
  -0.62860866]]
[[10  0  0  0]
 [ 0  3  0  0]
 [ 1  0  1  0]
 [ 0  0  0  2]]
accuracy score:  0.9411764705882353
```

```
clf = LogisticRegression()
clf.fit(X_train, y_train)
pred2 = clf.predict(X_test)
print(confusion_matrix(y_test, pred2))
print(accuracy_score(y_test, pred2))
```

```
[[10  0  0  0]
 [ 3  0  0  0]
 [ 2  0  0  0]
 [ 2  0  0  0]]
0.5882352941176471
```

```
clf = LogisticRegression(C=2.5, n_jobs=4, solver='lbfgs', random_state=17, verbose=1)
clf.fit(X_train, y_train)
pred2 = clf.predict(X_test)
print(confusion_matrix(y_test, pred2))
print(accuracy_score(y_test, pred2))
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[[10  0  0  0]
 [ 0  2  0  1]
 [ 2  0  0  0]
 [ 2  0  0  0]]
0.7058823529411765
[Parallel(n_jobs=4)]: Done   1 out of   1 | elapsed:    2.3s finished
```

```
classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(15, 2), random_state=1)
classifier.fit(X_train, y_train)

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

classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(6, 3), random_state=1)
classifier.fit(X_train, y_train)

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

```
accuracy score:  0.5882352941176471
accuracy score:  0.6470588235294118
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

Colab paid products - Cancel contracts here

✓ 3s completed at 8:09 PM