```
import pandas as pd
df = pd.read_csv('federalist.csv')
df[:10]
```

	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...
5	HAMILTON	FEDERALIST No. 6 Concerning Dangers from Disse...
6	HAMILTON	FEDERALIST. No. 7 The Same Subject Continued (...
7	HAMILTON	FEDERALIST No. 8 The Consequences of Hostiliti...
8	HAMILTON	FEDERALIST No. 9 The Union as a Safeguard Agai...
9	MADISON	FEDERALIST No. 10 The Same Subject Continued (...

```
df = df.astype({"author":'category'})
```

```
df['author'].unique()
```

```
['HAMILTON', 'JAY', 'MADISON', 'HAMILTON AND MADISON', 'HAMILTON OR MADISON']
Categories (5, object): ['HAMILTON', 'HAMILTON AND MADISON', 'HAMILTON OR MADISON',
'JAY',
                         'MADISON']
```

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

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

```
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, rand
print("X_train: ", X_train.shape)
print("X_test: ", X_test.shape)
print("y_train: ", y_train.shape)
print("y_test: ", y_test.shape)
```

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

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

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


X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

```
print('train size:', X_train.shape)
print('test size:', X_test.shape)
```

```
train size: (66, 7876)
test size: (17, 7876)
```

```
from sklearn.naive_bayes import BernoulliNB

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

```
BernoulliNB()
```

```
pred = naive_bayes.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.5882352941176471
```

Vectorizer with max_features set to 1000:

```
vectorizer2 = TfidfVectorizer(stop_words=stopwords, max_features=1000)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, rand
```

```
X_train = vectorizer2.fit_transform(X_train)
X_test = vectorizer2.transform(X_test)
```

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

```
     BernoulliNB()
```

```
pred = naive_bayes.predict(X_test)
```

WITHOUT BIGRAMS:

```
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
     accuracy score:  0.9411764705882353
```

Vectorizer with bigrams:

```
vectorizer3 = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1, 2), stop_words=stopwords,
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, rand
```

```
X_train = vectorizer3.fit_transform(X_train)
X_test = vectorizer3.transform(X_test)
```

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

```
     BernoulliNB()
```

```
pred = naive_bayes.predict(X_test)
```

WITH BIGRAMS:

```
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
     accuracy score:  0.9411764705882353
```

Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, rand

# vectorizer
vectorizer4 = TfidfVectorizer(stop_words=stopwords)
```

```
X_train = vectorizer4.fit_transform(X_train)
X_test = vectorizer4.transform(X_test)

#train
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
```

Neural Network:

```
from sklearn.neural_network import MLPClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, rand

# vectorizer
vectorizer5 = TfidfVectorizer(stop_words=stopwords)
X_train = vectorizer5.fit_transform(X_train)
X_test = vectorizer5.transform(X_test)

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

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

```
     accuracy score:  0.7647058823529411
```