```
import nltk
nltk.download('stopwords')
import pandas as pd
import warnings
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from google.colab import files
uploaded = files.upload() # used to upload a file to the google drive
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Choose Files federalist.csv

- **federalist.csv**(text/csv) - 1100616 bytes, last modified: 11/5/2022 - 100% done

```
Saving federalist.csv to federalist (3).csv
```

Step 1

```
df = pd.read_csv('federalist.csv')
df['author'] = pd.Categorical(df.author)
print(df.head(10))
dfx = df.groupby(['author'])['author'].count()
print(dfx)
```

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

Step 2

```
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(f"The shape of X is {X.shape}, X train is {X_train.shape} and X test is {X_test.shape}")
print(f"The shape of Y is {Y.shape}, Y train is {Y_train.shape} and Y test is {Y_test.shape}")
```

```
The shape of X is (83,), X train is (66,) and X test is (17,)
The shape of Y is (83,), Y train is (66,) and Y test is (17,)
```

Step 3

```
stopwords = stopwords.words('english')
stopwords = set(stopwords)
tf_idf = TfidfVectorizer(stop_words=stopwords)
X_train_set = tf_idf.fit_transform(X_train)
X_test_set = tf_idf.transform(X_test)
print(f"Training set shape - {X_train_set.shape}")
print(f"Test set shape - {X_test_set.shape}")
```

```
Training set shape - (66, 7876)
```

```
Test set shape - (17, 7876)
```

Step 4

```
bernoulli = BernoulliNB()
bernoulli.fit(X_train_set, Y_train)
pred = bernoulli.predict(X_test_set)
print('accuracy score: ', accuracy_score(Y_test, pred))
print(classification_report(Y_test, pred, zero_division=0))
```

```
accuracy score:  0.5882352941176471
                    precision    recall  f1-score   support

          HAMILTON       0.59      1.00      0.74        10
HAMILTON OR MADISON       0.00      0.00      0.00         3
               JAY       0.00      0.00      0.00         2
           MADISON       0.00      0.00      0.00         2

          accuracy                           0.59        17
         macro avg       0.15      0.25      0.19        17
      weighted avg       0.35      0.59      0.44        17
```

Step 5

```
# ngram_range=(1,2) uses unigrams and bigrams
tf_idf_update = TfidfVectorizer(stop_words=stopwords,ngram_range=(1,2), max_features=1000)
X_train_set = tf_idf_update.fit_transform(X_train)
X_test_set = tf_idf_update.transform(X_test)
print(f"\nTraining set shape - {X_train_set.shape}")
print(f"Test set shape - {X_test_set.shape}")
naive_bayes = BernoulliNB()
naive_bayes.fit(X_train_set, Y_train)
pred = naive_bayes.predict(X_test_set)
print("_____________________________________________________________")
print('\n accuracy score: ', accuracy_score(Y_test, pred))
print(classification_report(Y_test, pred, zero_division=0))
print("_____________________________________________________________")
```

```
Training set shape - (66, 1000)
Test set shape - (17, 1000)
_____________________________________________________________

 accuracy score:  0.9411764705882353
                    precision    recall  f1-score   support

          HAMILTON       0.91      1.00      0.95        10
HAMILTON OR MADISON       1.00      1.00      1.00         3
               JAY       1.00      0.50      0.67         2
           MADISON       1.00      1.00      1.00         2

          accuracy                           0.94        17
         macro avg       0.98      0.88      0.90        17
      weighted avg       0.95      0.94      0.93        17

_____________________________________________________________
```

Step 6

```
# ngram_range=(1,2) uses unigrams and bigrams
tf_idf_update = TfidfVectorizer(stop_words=stopwords,ngram_range=(1,2), max_features=1000)
X_train_set = tf_idf_update.fit_transform(X_train)
X_test_set = tf_idf_update.transform(X_test)
print("LogisticRegression no param__________________________________")
naive_bayes = LogisticRegression()
naive_bayes.fit(X_train_set, Y_train)
pred = naive_bayes.predict(X_test_set)
print('\n accuracy score: ', accuracy_score(Y_test, pred))
print(classification_report(Y_test, pred, zero_division=0))
print("_____________________________________________________________")
```

```
LogisticRegression no param__________________________________
```

```
 accuracy score:  0.5882352941176471
                    precision    recall  f1-score   support

          HAMILTON       0.59      1.00      0.74        10
HAMILTON OR MADISON       0.00      0.00      0.00         3
               JAY       0.00      0.00      0.00         2
           MADISON       0.00      0.00      0.00         2

          accuracy                           0.59        17
         macro avg       0.15      0.25      0.19        17
      weighted avg       0.35      0.59      0.44        17

_____________________________________________________________
```

```
print("LogisticRegression with param________________________________")
naive_bayes = LogisticRegression(solver='newton-cg',warm_start="True", multi_class='multinomial',class_weight='balanced', C = 0.8)
naive_bayes.fit(X_train_set, Y_train)
pred = naive_bayes.predict(X_test_set)
print('\n accuracy score: ', accuracy_score(Y_test, pred))
print(classification_report(Y_test, pred, zero_division=0))
print("_____________________________________________________________")
```

```
LogisticRegression with param________________________________

 accuracy score:  0.8235294117647058
                    precision    recall  f1-score   support

          HAMILTON       0.83      1.00      0.91        10
HAMILTON OR MADISON       0.75      1.00      0.86         3
               JAY       1.00      0.50      0.67         2
           MADISON       0.00      0.00      0.00         2

          accuracy                           0.82        17
         macro avg       0.65      0.62      0.61        17
      weighted avg       0.74      0.82      0.76        17

_____________________________________________________________
```

Step 7

```
warnings.filterwarnings('ignore')
print("1st MLPClassifier with param_____________________________________")
naive_bayes = MLPClassifier(random_state=1, activation = 'tanh', learning_rate = 'invscaling', max_iter= 300)
naive_bayes.fit(X_train_set, Y_train)
pred_max = naive_bayes.predict(X_test_set)
accur_max = accuracy_score(Y_test, pred)
print(f'\n accuracy score: {accur_max}')
print(classification_report(Y_test, pred, zero_division=0))

print("2nd MLPClassifier with param_____________________________________")
naive_bayes = MLPClassifier(random_state=1, activation = 'tanh', hidden_layer_sizes= (25,11,7,5,3,), learning_rate = 'invscaling', ma
naive_bayes.fit(X_train_set, Y_train)
pred = naive_bayes.predict(X_test_set)
accur = accuracy_score(Y_test, pred)
print(f'\n accuracy score: {accur}')
print(classification_report(Y_test, pred, zero_division=0))

if accur>accur_max:
    accur_max = accur


print("3rd MLPClassifier with param_____________________________________")
naive_bayes = MLPClassifier(random_state=1, activation = 'tanh',hidden_layer_sizes= (45,11,2,), learning_rate = 'invscaling', max_it
naive_bayes.fit(X_train_set, Y_train)
pred = naive_bayes.predict(X_test_set)
accur = accuracy_score(Y_test, pred)
print(f'\n accuracy score: {accur}')
print(classification_report(Y_test, pred, zero_division=0))
print("_____________________________________________________________")

if accur>accur_max:
    accur_max = accur

print(f"My best precision is {accur_max}")
```

```
1st MLPClassifier with param_____________________________________
```

```
 accuracy score: 0.8235294117647058
                    precision    recall  f1-score   support

          HAMILTON       0.83      1.00      0.91        10
HAMILTON OR MADISON       0.75      1.00      0.86         3
               JAY       1.00      0.50      0.67         2
           MADISON       0.00      0.00      0.00         2

          accuracy                           0.82        17
         macro avg       0.65      0.62      0.61        17
      weighted avg       0.74      0.82      0.76        17

2nd MLPClassifier with param_____________________________________

 accuracy score: 0.6470588235294118
                    precision    recall  f1-score   support

          HAMILTON       0.71      1.00      0.83        10
HAMILTON OR MADISON       0.00      0.00      0.00         3
               JAY       0.00      0.00      0.00         2
           MADISON       0.33      0.50      0.40         2

          accuracy                           0.65        17
         macro avg       0.26      0.38      0.31        17
      weighted avg       0.46      0.65      0.54        17

3rd MLPClassifier with param_____________________________________

 accuracy score: 0.8823529411764706
                    precision    recall  f1-score   support

          HAMILTON       0.83      1.00      0.91        10
HAMILTON OR MADISON       1.00      1.00      1.00         3
               JAY       1.00      0.50      0.67         2
           MADISON       1.00      0.50      0.67         2

          accuracy                           0.88        17
         macro avg       0.96      0.75      0.81        17
      weighted avg       0.90      0.88      0.87        17

_____________________________________________________________
My best precision is 0.8823529411764706
```

✓ 13s completed at 1:51 AM