```
import nltk
nltk.download('stopwords')
from google.colab import files
#Uploading federalist.csv
uploaded = files.upload()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Choose Files federalist.csv

- **federalist.csv**(text/csv) - 1100616 bytes, last modified: 10/31/2022 - 100% done

```
Saving federalist.csv to federalist.csv
```

```
import pandas as pd
#Reading in csv file
df = pd.read_csv('federalist.csv')
#Convert author column to categorical data
df['author'] = df['author'].astype('category')
#Display first few rows
print(df.head())
#Display counts by author
print(df['author'].value_counts())
```

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

```
from sklearn.model_selection import train_test_split
X = df['text']
y = df['author']
#Divide into train and test using random state 1234
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
#Display shape of train and test
print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)
```

```
Train shape: (66,)
Test shape: (17,)
```

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

```
#Removing stop words and performing tf-idf vectorization
stop_words = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stop_words)
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)
#Display training set shape and test set shape
print("Vectorized train shape:", X_train_vect.shape)
print("Vectorized test shape:", X_test_vect.shape)
```

```
Vectorized train shape: (66, 7678)
Vectorized test shape: (17, 7678)
```

Bernoulli Naive Bayes

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

#First Attempt
naive_bayes = BernoulliNB()
naive_bayes.fit(X_train_vect, y_train)

nb_pred = naive_bayes.predict(X_test_vect)

print('Accuracy Score: ', accuracy_score(y_test, nb_pred))
print("Classification Report:\n", classification_report(y_test, nb_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, nb_pred))
```

```
Accuracy Score:  0.5882352941176471
Classification Report:
                      precision    recall  f1-score   support

             HAMILTON       0.59      1.00      0.74        10
HAMILTON AND MADISON       0.00      0.00      0.00         1
 HAMILTON OR MADISON       0.00      0.00      0.00         2
                JAY       0.00      0.00      0.00         1
            MADISON       0.00      0.00      0.00         3

           accuracy                           0.59        17
          macro avg       0.12      0.20      0.15        17
       weighted avg       0.35      0.59      0.44        17

Confusion Matrix:
 [[10  0  0  0  0]
 [ 1  0  0  0  0]
 [ 2  0  0  0  0]
 [ 1  0  0  0  0]
 [ 3  0  0  0  0]]
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from nltk.classify import naivebayes
#Second Attempt adjusting Max Features and Bigram
vectorizer = TfidfVectorizer(stop_words=stop_words, max_features=1000, ngram_range=(1,2))

X_train_vect2 = vectorizer.fit_transform(X_train)
X_test_vect2 = vectorizer.transform(X_test)

naive_bayes2 = BernoulliNB()
naive_bayes2.fit(X_train_vect2, y_train)

nb_pred2 = naive_bayes2.predict(X_test_vect2)

print('Accuracy Score: ', accuracy_score(y_test, nb_pred2))
print("Classification Report:\n", classification_report(y_test, nb_pred2))
print("Confusion Matrix:\n", confusion_matrix(y_test, nb_pred2))
```

```
Accuracy Score:  0.7058823529411765
Classification Report:
                      precision    recall  f1-score   support

             HAMILTON       0.71      1.00      0.83        10
HAMILTON AND MADISON       0.00      0.00      0.00         1
 HAMILTON OR MADISON       1.00      0.50      0.67         2
                 JAY       0.00      0.00      0.00         1
             MADISON       0.50      0.33      0.40         3

            accuracy                           0.71        17
           macro avg       0.44      0.37      0.38        17
        weighted avg       0.63      0.71      0.64        17

Confusion Matrix:
 [[10  0  0  0  0]
 [ 1  0  0  0  0]
 [ 0  0  1  0  1]
 [ 1  0  0  0  0]
 [ 2  0  0  0  1]]
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
```

The first Naive Bayes attempt resulted in only 54% accuracy which is low. However, by lowering max features to only 1000 most frequent words and including bigrams as well, the model's accuracy went up to 71% from 54%.

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import log_loss
#No parameters
log_reg = LogisticRegression()
log_reg.fit(X_train_vect2, y_train)

lr_pred = log_reg.predict(X_test_vect2)

print('Accuracy Score: ', accuracy_score(y_test, lr_pred))
probs = log_reg.predict_proba(X_test_vect2)
print('Log Loss: ', log_loss(y_test, probs))
print("Classification Report:\n", classification_report(y_test, lr_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, lr_pred))
```

```
Accuracy Score:  0.5882352941176471
Log Loss:  0.958375361498397
Classification Report:
                      precision    recall  f1-score   support

             HAMILTON       0.59      1.00      0.74        10
HAMILTON AND MADISON        0.00      0.00      0.00         1
 HAMILTON OR MADISON        0.00      0.00      0.00         2
                 JAY        0.00      0.00      0.00         1
             MADISON        0.00      0.00      0.00         3

            accuracy                           0.59        17
           macro avg       0.12      0.20      0.15        17
        weighted avg       0.35      0.59      0.44        17

Confusion Matrix:
 [[10  0  0  0  0]
 [ 1  0  0  0  0]
 [ 2  0  0  0  0]
 [ 1  0  0  0  0]
 [ 3  0  0  0  0]]
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
```

+ Code + Text

```
#Adjusted with parameters
log_reg2 = LogisticRegression(class_weight='balanced', max_iter=1)
log_reg2.fit(X_train_vect2, y_train)

lr_pred2 = log_reg2.predict(X_test_vect2)
```

```
print('Accuracy Score: ', accuracy_score(y_test, lr_pred2))
probs = log_reg2.predict_proba(X_test_vect2)
print('Log Loss: ', log_loss(y_test, probs))
print("Classification Report:\n", classification_report(y_test, lr_pred2))
print("Confusion Matrix:\n", confusion_matrix(y_test, lr_pred2))
```

```
Accuracy Score:  0.8823529411764706
Log Loss:  1.5507651123349315
Classification Report:
                      precision    recall  f1-score   support

             HAMILTON       1.00      0.90      0.95        10
HAMILTON AND MADISON       1.00      1.00      1.00         1
 HAMILTON OR MADISON       1.00      0.50      0.67         2
                 JAY       1.00      1.00      1.00         1
             MADISON       0.60      1.00      0.75         3

            accuracy                           0.88        17
           macro avg       0.92      0.88      0.87        17
        weighted avg       0.93      0.88      0.89        17

Confusion Matrix:
 [[9 0 0 0 1]
 [0 1 0 0 0]
 [0 0 1 0 1]
 [0 0 0 1 0]
 [0 0 0 0 3]]
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Convergenc
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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
#Lowering log loss
log_reg2 = LogisticRegression(class_weight='balanced', C=100)

log_reg2.fit(X_train_vect2, y_train)

lr_pred2 = log_reg2.predict(X_test_vect2)

print('Accuracy Score: ', accuracy_score(y_test, lr_pred2))
probs = log_reg2.predict_proba(X_test_vect2)
print('Log Loss: ', log_loss(y_test, probs))
print("Classification Report:\n", classification_report(y_test, lr_pred2))
print("Confusion Matrix:\n", confusion_matrix(y_test, lr_pred2))
```

```
Accuracy Score:  0.7647058823529411
Log Loss:  0.480552802234012
Classification Report:
```

```
                         precision    recall  f1-score   support

              HAMILTON       0.83      1.00      0.91        10
     HAMILTON AND MADISON       0.00      0.00      0.00         1
      HAMILTON OR MADISON       0.00      0.00      0.00         2
                   JAY       1.00      1.00      1.00         1
               MADISON       0.50      0.67      0.57         3

              accuracy                           0.76        17
             macro avg       0.47      0.53      0.50        17
          weighted avg       0.64      0.76      0.69        17

Confusion Matrix:
 [[10  0  0  0  0]
 [ 1  0  0  0  0]
 [ 0  0  0  0  2]
 [ 0  0  0  1  0]
 [ 1  0  0  0  2]]
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
```

The Logistic Regression without parameters resulted in only 59% accuracy which is low. However, by changing the class weight to balanced and making max iteration before convergence to 1, the model's accuracy went up to 88% from 59%. This come with the issue of having a really high log loss at 1.55. Instead, by switching from max iteration parameter to C which control penalty strength and setting it to 100, we can get 76% accuracy while only being at 0.48 for log loss.

Neural Networks

```
from sklearn.neural_network import MLPClassifier
#First topology
nn = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(10), random_state=1234)
nn.fit(X_train_vect2, y_train)

nn_pred = nn.predict(X_test_vect2)

print('Accuracy Score: ', accuracy_score(y_test, nn_pred))
print("Classification Report:\n", classification_report(y_test, nn_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, nn_pred))
```

```
Accuracy Score:  0.8235294117647058
Classification Report:
                         precision    recall  f1-score   support
```

```
                HAMILTON       0.91      1.00      0.95        10
      HAMILTON AND MADISON       0.00      0.00      0.00         1
       HAMILTON OR MADISON       0.00      0.00      0.00         2
                     JAY       1.00      1.00      1.00         1
                  MADISON       0.60      1.00      0.75         3

                accuracy                           0.82        17
               macro avg       0.50      0.60      0.54        17
            weighted avg       0.70      0.82      0.75        17

      Confusion Matrix:
       [[10  0  0  0  0]
       [ 1  0  0  0  0]
       [ 0  0  0  0  2]
       [ 0  0  0  1  0]
       [ 0  0  0  0  3]]
      /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
        _warn_prf(average, modifier, msg_start, len(result))
      /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
        _warn_prf(average, modifier, msg_start, len(result))
      /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
        _warn_prf(average, modifier, msg_start, len(result))
```

```
#Second topology
nn2 = MLPClassifier(solver='lbfgs', random_state=1234) # size 100
nn2.fit(X_train_vect2, y_train)

nn2_pred = nn2.predict(X_test_vect2)

print('Accuracy Score: ', accuracy_score(y_test, nn2_pred))
print("Classification Report:\n", classification_report(y_test, nn2_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, nn2_pred))
```

```
      Accuracy Score:  0.8823529411764706
      Classification Report:
                           precision    recall  f1-score   support

                HAMILTON       1.00      1.00      1.00        10
      HAMILTON AND MADISON       0.50      1.00      0.67         1
       HAMILTON OR MADISON       1.00      0.50      0.67         2
                     JAY       0.00      0.00      0.00         1
                  MADISON       0.75      1.00      0.86         3

                accuracy                           0.88        17
               macro avg       0.65      0.70      0.64        17
            weighted avg       0.87      0.88      0.86        17

      Confusion Matrix:
       [[10  0  0  0  0]
       [ 0  1  0  0  0]
       [ 0  0  1  0  1]
       [ 0  1  0  0  0]
       [ 0  0  0  0  3]]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
```

```
#Third topology
nn3 = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(200), random_state=1234)
nn3.fit(X_train_vect2, y_train)

nn3_pred = nn3.predict(X_test_vect2)

print('Accuracy Score: ', accuracy_score(y_test, nn3_pred))
print("Classification Report:\n", classification_report(y_test, nn3_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, nn3_pred))
```

```
Accuracy Score:  0.9411764705882353
Classification Report:
                      precision    recall  f1-score   support

             HAMILTON       1.00      1.00      1.00        10
HAMILTON AND MADISON       1.00      1.00      1.00         1
 HAMILTON OR MADISON       1.00      0.50      0.67         2
                JAY       1.00      1.00      1.00         1
            MADISON       0.75      1.00      0.86         3

           accuracy                           0.94        17
          macro avg       0.95      0.90      0.90        17
       weighted avg       0.96      0.94      0.94        17

Confusion Matrix:
 [[10  0  0  0  0]
 [ 0  1  0  0  0]
 [ 0  0  1  0  1]
 [ 0  0  0  1  0]
 [ 0  0  0  0  3]]
```

After 3 different attempts for higher accuracy, my final accuracy is 94%.

After 3 different attempts for higher accuracy, my final accuracy is 94%.

Colab paid products - Cancel contracts here

✓ 1s completed at 12:33 AM