CS 4395 Author Attribution Assignment

This assignment will use sklearn to predict the author of a given document.

Part 1

```
from google.colab import files
data = files.upload()
```

Choose Files federalist.csv

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

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

```
import io
import pandas as pd
df = pd.read_csv(io.BytesIO(data['federalist.csv']))
```

```
df.author = df.author.astype('category')
print("Counts: ")
print(df.author.value_counts())
df.head()
```

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

	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...

Part 2 - Train/Test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.text, df.author, test_size=0.2, random
print(X_train.shape)
print(X_test.shape)
```

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

Part 3 - Text processing & tf-idf vectorization

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords = set(stopwords.words('english'))
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words=stopwords)
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)
print(X_train_vect.shape)
print(X_test_vect.shape)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
(66, 7678)
(17, 7678)
```

Part 4 - Naive Bayes Model

```
from sklearn.naive_bayes import BernoulliNB
bnb = BernoulliNB()
bnb.fit(X_train_vect, y_train)
print(bnb.score(X_train_vect, y_train), '\n')
prediction = bnb.predict(X_test_vect)

from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, prediction))
```

```
0.803030303030303

                      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

/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))
```

Part 5 - Naive Bayes with Vectorization

```
from sklearn.feature_extraction.text import TfidfVectorizer
vector = TfidfVectorizer(stop_words=stopwords, max_features=1000, ngram_range=(2,2))
X_train_vect2 = vector.fit_transform(X_train)
X_test_vect2 = vector.transform(X_test)
print(X_train_vect2.shape)
print(X_test_vect2.shape)
```

```
(66, 1000)
(17, 1000)
```

```
bnb2 = BernoulliNB()
bnb2.fit(X_train_vect2, y_train)
print(bnb2.score(X_train_vect2, y_train), '\n')
prediction2 = bnb2.predict(X_test_vect2)
print(classification_report(y_test, prediction2))
```

```
0.8939393939393939

                      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

/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))
```

Part 6 - Logistic Regression

First algorithm using default solver ('lbfgs')

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(solver='lbfgs', multi_class='multinomial', class_weight='balance
log_reg.fit(X_train_vect2, y_train)
prediction = log_reg.predict(X_test_vect2)
print(classification_report(y_test, prediction), '\n')
```

```
                        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       1.00      0.50      0.67         2
                   JAY       1.00      1.00      1.00         1
               MADISON       0.75      1.00      0.86         3

              accuracy                           0.88        17
             macro avg       0.73      0.70      0.70        17
          weighted avg       0.84      0.88      0.85        17


/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 algorithm using solver 'liblinear'

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(solver='liblinear', class_weight='balanced', random_state=1234)
log_reg.fit(X_train_vect2, y_train)
prediction = log_reg.predict(X_test_vect2)
print(classification_report(y_test, prediction), '\n')
```

```
                        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


/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 algorithms using solver 'sag'

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(solver='sag', multi_class='multinomial', class_weight='balanced'
log_reg.fit(X_train_vect2, y_train)
prediction = log_reg.predict(X_test_vect2)
print(classification_report(y_test, prediction), '\n')
```

```
                        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       1.00      0.50      0.67         2
                    JAY       1.00      1.00      1.00         1
                MADISON       0.75      1.00      0.86         3

               accuracy                           0.88        17
              macro avg       0.73      0.70      0.70        17
           weighted avg       0.84      0.88      0.85        17


/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))
```

Fourth algorithm using solver 'saga'

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(solver='saga', multi_class='multinomial', class_weight='balanced
log_reg.fit(X_train_vect2, y_train)
prediction = log_reg.predict(X_test_vect2)
```

```
print(classification_report(y_test, prediction), '\n')
```

```
                        precision    recall  f1-score   support

              HAMILTON       1.00      0.60      0.75        10
     HAMILTON AND MADISON       0.20      1.00      0.33         1
      HAMILTON OR MADISON       1.00      0.50      0.67         2
                   JAY       1.00      1.00      1.00         1
               MADISON       0.50      0.67      0.57         3

              accuracy                           0.65        17
             macro avg       0.74      0.75      0.66        17
          weighted avg       0.86      0.65      0.70        17


/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_sag.py:354: ConvergenceWarn
  ConvergenceWarning,
```

Lbfgs and sag boast the highest accuracy at 0.88 (but only if multi_class is set to multinomial, otherwise the accuracy is lower), which is a significant improvement over the naive bayes classifier.

Part 7 - Neural Networks

```
from sklearn.neural_network import MLPClassifier
nn = MLPClassifier(activation='tanh',solver='lbfgs', alpha=1e-4, hidden_layer_sizes=(10, 15,
nn.fit(X_train_vect2, y_train)
prediction = nn.predict(X_test_vect2)
print(classification_report(y_test, prediction), '\n')
```

```
                        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       0.50      0.50      0.50         2
                   JAY       1.00      1.00      1.00         1
               MADISON       0.50      0.33      0.40         3

              accuracy                           0.82        17
             macro avg       0.70      0.77      0.71        17
          weighted avg       0.82      0.82      0.82        17
```

Of all the combinations, lbgfs with tanh activation function provided the best accuracy, and the hidden layers were tweaked until I reached the highest at 82% accuracy. It seems the accuracy is

always 59%, 65%, 71%, 76%, and 82%. No matter what I changed, the accuracy is always one of those five. Overall, logistic regression provides the best accuracy at 88%.