rcd180001-HW7

November 6, 2022

1 Author Attribution

Joshua Durana rcd180001

[ ]:
```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.linear_model._logistic import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPClassifier

from nltk.corpus import stopwords
```

1.1 Load Data

[ ]:
```
#Load CSV
federalistCSV = pd.read_csv('Data/federalist.csv')

#Set Author to Categorical
federalistCSV.author = federalistCSV.author.astype('category')

#Print Head
print(federalistCSV.head())

#Number of rows for each author
print(federalistCSV.groupby(['author']).size())
```

```
     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…
author
HAMILTON               49
```

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

1.2 Divide to Train and Test

[ ]:
```
#Divide to test and train
textTrain, textTest, authorsTrain, authorsTest = train_test_split(federalistCSV.
 ↪text, federalistCSV.author, test_size = .2, random_state=1234)

#Print Dimensions
print("Train Dimensions:", textTrain.shape)
print("Test Dimensions:", textTest.shape)
```

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

1.3 Process Text

[ ]:
```
#Create Vectorizer
vectorizer = TfidfVectorizer(stop_words = set(stopwords.words('english')))

#Remove Stopwords and Transform Train
textTrainV = vectorizer.fit_transform(textTrain)
textTestV = vectorizer.transform(textTest)


#Shape
print("Train Shape: ", textTrainV.shape)
print("Test Shape: ", textTestV.shape)

textTrainV.toarray()
```

```
Train Shape:  (66, 7876)
Test Shape:  (17, 7876)
```

[ ]:
```
array([[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.        , 0.        ,
```

```
        0.        ],
       [0.        , 0.        , 0.        , …, 0.02275824, 0.        ,
        0.        ]])
```

1.4 Bernoulli Naive Bayes

[ ]:
```
#Create Model
fedNN = MultinomialNB()
fedNN.fit(textTrainV, authorsTrain)

#Test Model
fedNNPredict = fedNN.predict(textTestV)

#Metrics
print(classification_report(authorsTest, fedNNPredict, zero_division=False))

#Confusion Matrix
print(confusion_matrix(authorsTest, fedNNPredict))
```

```
                    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

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

1.5 Edit Training and Test Vectors

[ ]:
```
#Edit Vectorizer
vectorizer = TfidfVectorizer(stop_words = set(stopwords.words('english')),␣
 ↪max_features = 1000, ngram_range = (1,2))

#Vectorize Text
textTrainMF = vectorizer.fit_transform(textTrain)
textTestMF = vectorizer.transform(textTest)

#Shape
print("Train Shape: ", textTrainMF.shape)
```

```
print("Test Shape: ", textTestMF.shape)

textTrainMF.toarray()
```

```
Train Shape:  (66, 1000)
Test Shape:  (17, 1000)
```

[ ]:
```
array([[0.01943752, 0.01943752, 0.01873201, …, 0.03173652, 0.01715013,
        0.        ],
       [0.02016878, 0.02016878, 0.        , …, 0.03293047, 0.01779534,
        0.        ],
       [0.01158532, 0.01158532, 0.        , …, 0.10718994, 0.01022197,
        0.        ],
       …,
       [0.01842101, 0.01842101, 0.        , …, 0.03007681, 0.01625325,
        0.        ],
       [0.        , 0.        , 0.        , …, 0.02535216, 0.        ,
        0.        ],
       [0.02044439, 0.02044439, 0.        , …, 0.02225365, 0.01803851,
        0.02706159]])
```

1.6 Naive Bayes

[ ]:
```
#Create Model
fedNN.fit(textTrainMF, authorsTrain)

#Predict
fedNNPredict = fedNN.predict(textTestMF)

#Metrics
print(classification_report(authorsTest, fedNNPredict, zero_division=False))

#Confusion Matrix
print(confusion_matrix(authorsTest, fedNNPredict))
```

```
                    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

[[10  0  0  0]
```

```
 [ 3  0  0  0]
 [ 2  0  0  0]
 [ 2  0  0  0]]
```

There doesn't seem to have a change between the 2 runs of Naive Bayes, most likely due to the majority of the Federalist Papers were written by Hamilton

1.7 Logistic Regression

[ ]:
```
#Make Pipeline
pipeLR = Pipeline([
('tfidf', TfidfVectorizer(stop_words = set(stopwords.words('english')))),
('logreg', LogisticRegression()),
])

#Make model
pipeLR.fit(textTrain, authorsTrain)

#Evaluate
predictionLR = pipeLR.predict(textTest)

#Report
print(classification_report(authorsTest, predictionLR, zero_division=False))

print(confusion_matrix(authorsTest, predictionLR))
```

```
                    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

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

[ ]:
```
##Improved LR
pipeLR = Pipeline([
('tfidf', TfidfVectorizer(stop_words = set(stopwords.words('english')))),
('logreg', LogisticRegression(multi_class='multinomial',␣
 ↪solver='lbfgs',class_weight='balanced')),
])
```

```
#Make model
pipeLR.fit(textTrain, authorsTrain)

#Evaluate
predictionLR = pipeLR.predict(textTest)

#Report
print(classification_report(authorsTest, predictionLR, zero_division=False))

#Confusion Matrix
print(confusion_matrix(authorsTest, predictionLR))
```

```
                    precision    recall  f1-score   support

           HAMILTON       0.71      1.00      0.83        10
HAMILTON OR MADISON       1.00      0.67      0.80         3
                JAY       0.00      0.00      0.00         2
            MADISON       0.00      0.00      0.00         2

           accuracy                           0.71        17
          macro avg       0.43      0.42      0.41        17
       weighted avg       0.60      0.71      0.63        17

[[10  0  0  0]
 [ 0  2  0  1]
 [ 2  0  0  0]
 [ 2  0  0  0]]
```

1.8 Neural Network

[ ]:
```
#Create Pipe
pipeNN = Pipeline([
('tfidf', TfidfVectorizer(stop_words = set(stopwords.words('english')))),
('mlp', MLPClassifier(solver = 'lbfgs', hidden_layer_sizes=(30, 15))),
])

#Fit
pipeNN.fit(textTrain, authorsTrain)

#Predict
predictionsNN = pipeNN.predict(textTest)

#Report
print(classification_report(authorsTest, predictionsNN, zero_division=False))

#Confusion Matrix
```

```
print(confusion_matrix(authorsTest, predictionsNN))
```

```
                    precision    recall  f1-score   support

           HAMILTON       0.91      1.00      0.95        10
HAMILTON OR MADISON       0.50      0.67      0.57         3
                JAY       0.00      0.00      0.00         2
            MADISON       0.50      0.50      0.50         2

           accuracy                           0.76        17
          macro avg       0.48      0.54      0.51        17
       weighted avg       0.68      0.76      0.72        17

[[10  0  0  0]
 [ 0  2  0  1]
 [ 1  1  0  0]
 [ 0  1  0  1]]
```