main

November 12, 2022

```
[766]: import pandas as pd
       import nltk
       from nltk.corpus import stopwords
       from sklearn.model_selection import train_test_split
       from sklearn.feature_extraction.text import TfidfVectorizer
       from sklearn.naive_bayes import BernoulliNB
       from sklearn.linear_model import LogisticRegression
       from sklearn.neural_network import MLPClassifier
       from sklearn.metrics import accuracy_score
```

0.0.1 Step 1

```
[767]: df = pd.read_csv("federalist.csv")
       df["author"] = df.author.astype("category")
       print(df.head())
       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…
```

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

0.0.2 Step 2

[768]:
```
X, y = df.text, 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(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

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

0.0.3 Step 3

[769]:
```
stopwords = set(stopwords.words("english"))
vectorizer = TfidfVectorizer(stop_words=stopwords)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

print(X_train.shape)
print(X_test.shape)
```

```
(66, 7876)
(17, 7876)
```

0.0.4 Step 4

[770]:
```
nb = BernoulliNB()
nb.fit(X_train, y_train)
pred = nb.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, pred))
```

```
Accuracy Score: 0.5882352941176471
```

0.0.5 Step 5

[771]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪train_size=0.8, random_state=1234)
vectorizer = TfidfVectorizer(stop_words=stopwords, max_features=1000,␣
↪ngram_range=(1, 2))
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

nb.fit(X_train, y_train)
pred = nb.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, pred))
```

```
Accuracy Score: 0.9411764705882353
```

0.0.6 Step 6

```
[772]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
        ↪train_size=0.8, random_state=1234)
       vectorizer = TfidfVectorizer(binary=True, stop_words=stopwords,␣
        ↪max_features=1000, ngram_range=(1, 2))
       X_train = vectorizer.fit_transform(X_train)
       X_test = vectorizer.transform(X_test)

       lr = LogisticRegression(solver="lbfgs", class_weight="balanced")
       lr.fit(X_train, y_train)
       pred = lr.predict(X_test)

       print("Accuracy Score:", accuracy_score(y_test, pred))
```

```
Accuracy Score: 0.8823529411764706
```

0.0.7 Step 6 - model param change

I changed bigrams -> trigrams, introduced a min_df, and increased the C value. But I did not see any improvements

```
[773]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
        ↪train_size=0.8, random_state=1234)
       vectorizer = TfidfVectorizer(binary=True, stop_words=stopwords,␣
        ↪max_features=1000, ngram_range=(1, 3), min_df=5)
       X_train = vectorizer.fit_transform(X_train)
       X_test = vectorizer.transform(X_test)

       lr = LogisticRegression(solver="lbfgs", class_weight="balanced", C=2)
       lr.fit(X_train, y_train)
       pred = lr.predict(X_test)

       print("Accuracy Score:", accuracy_score(y_test, pred))
```

```
Accuracy Score: 0.8823529411764706
```

0.0.8 Step 7

I tried a couple of layzer sizes but I couldn't find better results than a simple (15, 2) layer

```
[774]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
        ↪train_size=0.8, random_state=1234)
       vectorizer = TfidfVectorizer(binary=True, stop_words=stopwords,␣
        ↪max_features=1000, ngram_range=(1, 2))
       X_train = vectorizer.fit_transform(X_train)
       X_test = vectorizer.transform(X_test)
```

```
nn = MLPClassifier(solver="lbfgs", alpha=1e-5, hidden_layer_sizes=(15,2),␣
 ↪random_state=1)
nn.fit(X_train, y_train)
pred = nn.predict(X_test)

print("Accuracy Score:", accuracy_score(y_test, pred))
```

```
Accuracy Score: 0.7058823529411765
```

[ ]:

```

```