Author Attribution

Aleena Syed

CS 4395.001

In [71]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
```

In [72]:
```
# Convert author column to categorical data
df = pd.read_csv('federalist.csv')
df['author'] = df['author'].astype('category')
print(df.head())
```

```
     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...
```

In [73]:
```
# display counts by author
df['author'].value_counts()
```

Out[73]:
```
HAMILTON                49
MADISON                 15
HAMILTON OR MADISON     11
JAY                      5
HAMILTON AND MADISON     3
Name: author, dtype: int64
```

In [74]:
```
# divide into train and test, with 80% in train, using random state 1234
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 shape of train and test
print(X_train.shape)
print(X_test.shape)
```

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

In [75]:
```
# remove stopwords and perform tf-idf vectorization
stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords)

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

# print shape of train and test
print(X_train.shape)
print(X_test.shape)
```

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

In [76]:
```
# Bernoulli Naive Bayes model
naive_bayes = BernoulliNB()
naive_bayes.fit(X_train, y_train)
```

Out[76]:
```
BernoulliNB()
```

In [77]:
```
# accuracy on test
pred = naive_bayes.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.5882352941176471
```

In [78]:
```
# second Bernoulli Naive Bayes model
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)
```

In [79]:
```
# max_features = 1000, add bigrams as feature
vectorizer2 = TfidfVectorizer(stop_words=stopwords, max_features=1000, ngram_range=(1,2))
X_train = vectorizer2.fit_transform(X_train)
X_test = vectorizer2.transform(X_test)
```

In [80]:
```
# try new model with updated parameters
naive_bayes = BernoulliNB()
naive_bayes.fit(X_train, y_train)
```

Out[80]:
```
BernoulliNB()
```

In [81]:
```
# new accuracy on test
pred = naive_bayes.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.9411764705882353
```

In [82]:
```
# logistic regression
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)

vectorizer = TfidfVectorizer(binary=True)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

classifier = LogisticRegression(C = 0.1, solver='lbfgs', class_weight='balanced')
classifier.fit(X_train, y_train)

# accuracy on test
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.7647058823529411
```

In [83]:
```
# neural networks topology 1
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True)
X = vectorizer.fit_transform(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)

classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(15, 2), random_state=1)
classifier.fit(X_train, y_train)
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.7647058823529411
```

In [84]:
```
# neural networks topology 2
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True)
X = vectorizer.fit_transform(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)

classifier = MLPClassifier(alpha=1e-05, hidden_layer_sizes=(15,), random_state=1,
              solver='lbfgs')
classifier.fit(X_train, y_train)
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.7647058823529411
```

In [88]:
```
# neural networks topology 3
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True)
X = vectorizer.fit_transform(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)

classifier = MLPClassifier(hidden_layer_sizes=(8,8,8), activation='relu', solver='adam', max_iter=1000)

classifier.fit(X_train, y_train)
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.5882352941176471
```

multi-layer perceptron using backpropagation gives best accuracy