In [233]:

```
# Author: Alfredo Mejia
# Date  : 11/15/20
# Class : CS 4395.001
# Assign: Homework 8
# Descrp: This notebook follows the instructions given in homework 8 description. It will
 follow the seven steps accurately.
#         This notebook will use the Federalist papers to try and use it as data to predic
t and test the authors of each paper.
#         It will use the TF-IDF vectorizer and train on the Bernoulli Naive Bayes Model,
 Logistic Regression, and a Neural
#         Network model. The results are shown below.
```

In [234]:

```
# Note I will not be using the print command because Jupyter Notebook does it automaticall
y
```

In [235]:

```
import pandas as pd
```

In [236]:

```
# Step 1a: Read in the csv file and convert author column to categorical data
dataframe = pd.read_csv("federalist.csv")
dataframe['author'] = dataframe.author.astype('category')
```

In [237]:

```
# Step 1b: Display the first few rows
dataframe.head(10)
```

Out[237]:

	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...
5	HAMILTON	FEDERALIST No. 6 Concerning Dangers from Disse...
6	HAMILTON	FEDERALIST. No. 7 The Same Subject Continued (...
7	HAMILTON	FEDERALIST No. 8 The Consequences of Hostiliti...
8	HAMILTON	FEDERALIST No. 9 The Union as a Safeguard Agai...
9	MADISON	FEDERALIST No. 10 The Same Subject Continued (...

In [238]:

```
# Step 1c: Display the counts by author
dataframe['author'].value_counts()
```

Out[238]:

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

In [239]:

```
# Step 2a: Divide into train and test, with 80% in train (One way of doing it)
from sklearn.model_selection import train_test_split
train, test = train_test_split(dataframe, test_size=0.2, random_state=1234)
```

In [240]:

```
# Step 2b: Display shape of train
train.shape
```

Out[240]:

```
(66, 2)
```

In [241]:

```
# Step 2c: Display shape of test
test.shape
```

Out[241]:

```
(17, 2)
```

In [242]:

```
# Step 2d: Divide into train and test, with 80% in train (Better way of doing it) and prin
t shape of x_train
x_train, x_test, y_train, y_test = train_test_split(dataframe['text'], dataframe['author'
], test_size=0.2, random_state=1234)
x_train.shape
```

Out[242]:

```
(66,)
```

In [243]:

```
# Step 2e: Display shape of y_train
y_train.shape
```

Out[243]:

```
(66,)
```

In [244]:

```
# Step 2f: Display shape of x_test
x_test.shape
```

Out[244]:

```
(17,)
```

In [245]:

```
# Step 2g: Display shape of y_test
y_test.shape
```

Out[245]:

```
(17,)
```

In [246]:

```
# Step 2h: Convert text labels to numbers
y_train = y_train.cat.codes
y_test = y_test.cat.codes
```

In [247]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [248]:

```
# Step 3a: Remove stop words and fit training data to TF-IDF vectorization
tfidf_vectorizer = TfidfVectorizer(lowercase=True, stop_words='english')
tfidf_vectorizer.fit(x_train)
```

Out[248]:

```
TfidfVectorizer(stop_words='english')
```

In [249]:

```
# Step 3b: Transform train and test data
x_train = tfidf_vectorizer.transform(x_train)
x_test = tfidf_vectorizer.transform(x_test)
```

In [250]:

```
# Step 3c: Display shape of train
x_train.shape
```

Out[250]:

```
(66, 7727)
```

In [251]:

```
# Step 3d: Display shape of test
x_test.shape
```

Out[251]:

```
(17, 7727)
```

In [252]:

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score
```

In [253]:

```
# Step 4: Train and test on Naive Bayes Model
naive_bayes = BernoulliNB()
naive_bayes.fit(x_train, y_train)
pred = naive_bayes.predict(x_test)
accuracy_score(y_test, pred)
```

Out[253]:

```
0.5882352941176471
```

In [254]:

```
# Step 5a: Redo the splitting for clarity
x_train, x_test, y_train, y_test = train_test_split(dataframe['text'], dataframe['author'
], test_size=0.2, random_state=1234)
y_train = y_train.cat.codes
y_test = y_test.cat.codes
```

In [255]:

```
# Step 5b: Redo TF-IDF vectorization with new parameters
tfidf_vectorizer = TfidfVectorizer(lowercase=True, stop_words='english', max_features=1000
, ngram_range=(1,2))
tfidf_vectorizer.fit(x_train)
x_train = tfidf_vectorizer.transform(x_train)
x_test = tfidf_vectorizer.transform(x_test)
```

In [256]:

```
# Step 5c: Redo the training and testing on Naive Bayes Model
naive_bayes = BernoulliNB()
naive_bayes.fit(x_train, y_train)
pred = naive_bayes.predict(x_test)
accuracy_score(y_test, pred)
```

Out[256]:

```
0.8823529411764706
```

In [257]:

```
from sklearn.linear_model import LogisticRegression
```

In [258]:

```
# Step 6a: Train and test Logistic Regression Model with no parameters
logistic_regression = LogisticRegression()
logistic_regression.fit(x_train, y_train)
pred = logistic_regression.predict(x_test)
logistic_regression.score(x_test, y_test)
```

Out[258]:

```
0.5882352941176471
```

In [259]:

```
# Step 6b: Train and test Logistic Regression Model with parameters
logistic_regression = LogisticRegression(max_iter=1000, solver='liblinear', penalty='l2',
dual=True, class_weight='balanced')
logistic_regression.fit(x_train, y_train)
pred = logistic_regression.predict(x_test)
logistic_regression.score(x_test, y_test)
```

Out[259]:

```
0.7058823529411765
```

In [260]:

```
from sklearn.neural_network import MLPClassifier
```

In [261]:

```
# Step 7: Train and test on a neural network with different topologies
# Topology 1
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)
accuracy_score(y_test, pred)
```

Out[261]:

```
0.6470588235294118
```

In [262]:

```
# Topology 2
classifier = MLPClassifier(max_iter=500,hidden_layer_sizes=(500, 250))
classifier.fit(x_train, y_train)
pred = classifier.predict(x_test)
accuracy_score(y_test, pred)
```

Out[262]:

```
0.8235294117647058
```

In [265]:

```
# Topology 3
classifier = MLPClassifier(learning_rate='invscaling', alpha=0.01, solver='lbfgs', hidden_
layer_sizes=(20,10))
classifier.fit(x_train, y_train)
pred = classifier.predict(x_test)
accuracy_score(y_test, pred)
```

Out[265]:

```
0.8235294117647058
```

In [266]:

```
# Topology 4
classifier = MLPClassifier()
classifier.fit(x_train, y_train)
pred = classifier.predict(x_test)
accuracy_score(y_test, pred)
```

```
D:\System\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perc
eptron.py:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[266]:

```
0.7647058823529411
```

In [ ]:

```
# Note: The warning shown above is simply stating the model hasn't converged and that is p
erfectly fine.
```

In [267]:

```
# Topology 5
classifier = MLPClassifier(hidden_layer_sizes=(100, 500))
classifier.fit(x_train, y_train)
pred = classifier.predict(x_test)
accuracy_score(y_test, pred)
```

Out[267]:

```
0.8823529411764706
```

In [269]:

```
# Topology 6
classifier = MLPClassifier(learning_rate_init = 0.01, learning_rate='adaptive')
classifier.fit(x_train, y_train)
pred = classifier.predict(x_test)
accuracy_score(y_test, pred)
```

Out[269]:

```
0.8823529411764706
```

In [ ]:

```
# The best result that I could get is around 88%.
```