# Solutions

November 22, 2022

## 1 Preliminaries

Load the “Federalist Papers” dataset from ILIAS. This dataset contains 85 Federalists Papers written by three different authors, namely Hamilton, Madison and Jay. We have also papers written by Hamilton and Madison together, and texts without a clear attribution (either Hamilton or Madison).

[1]:
```
import pandas as pd

fed_data = pd.read_csv('federalist-papersNew2.csv', index_col=0) # Index is the␣
 ↪first column
fed_data.head(5)
```

[1]:
```
   000  1  10  100  104  105  109  11  114  115  …  young  your  yourself  \
1    0  2   0    0    0    0    0   0    0    0  …      0    10         0
2    0  0   0    0    0    0    0   0    0    0  …      0     0         0
3    0  0   0    0    0    0    0   0    0    0  …      0     0         0
4    0  0   0    0    0    0    0   0    0    0  …      0     0         0
5    0  0   0    0    0    0    0   0    0    0  …      1     3         0

   yourselves  zaleucus  zeal  zealand  zealous  zelden    AUTHOR
1           0         0     3        0        0       0  Hamilton
2           0         0     0        0        0       0       Jay
3           0         0     0        0        0       0       Jay
4           0         0     0        0        0       0       Jay
5           1         0     0        0        0       0       Jay

[5 rows x 11501 columns]
```

Now we define our train set, made up by the 65 papers certainly written by Hamilton and Madison.

[2]:
```
train = fed_data[fed_data['AUTHOR'].isin({'Hamilton', 'Madison'})] # Train set
train.AUTHOR.value_counts() # Get number of papers for each author
```

[2]:
```
Hamilton    51
Madison     14
Name: AUTHOR, dtype: int64
```

The test set instead consists of the 12 disputed papers, written by Hamilton or Madison.

[3]:
```python
test = fed_data[fed_data['AUTHOR'].isin({'Hamilton OR Madison'})] # Test set
print(test.index.values) # Get the 12 disputed papers
```

```
[49 50 51 52 53 54 55 56 57 58 62 63]
```

## 2 Practical assignment: Apply the Naive Bayes classifier to solve the authorship attribution problem related to the twelve disputed Federalist Papers (written by "Hamilton OR Madison"). You can use the 65 papers written by "Hamilton" (51) and "Madison" (14) to train your classifier and the disputed papers to evaluate your system. As features, you can use the following words: {"to", "upon", "would"}.

To solve the attribution problem, we can use the Naive Bayes classifier. This simple model estimates the probability of each paper to belong to a certain author. The "naive" assumption behind this model is the independence between the feature variables. To implement the Naive Bayes classifier, we will assume that the probability $P(x_i \mid y)$ distributes over a Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Before proceeding, we need to extract the features (x_train) and the predicting variable (y_train) from our train and test dataset. For the ground truth (y_test), we consider all disputed papers as written by Madison.

[4]:
```python
x_train = train[['to', 'upon', 'would']] # Features for train
y_train = train['AUTHOR'] # Predicting variable for train
x_test = test[['to', 'upon', 'would']] # Features for test
y_test = test['AUTHOR'] # Ground truth
y_test = y_test.replace('Hamilton OR Madison', 'Madison') # Consider disputed␣
  ↪papers as written by Madison
```

Then, we can train our Gaussian Naive Bayes classifier from the scikit-learn library.

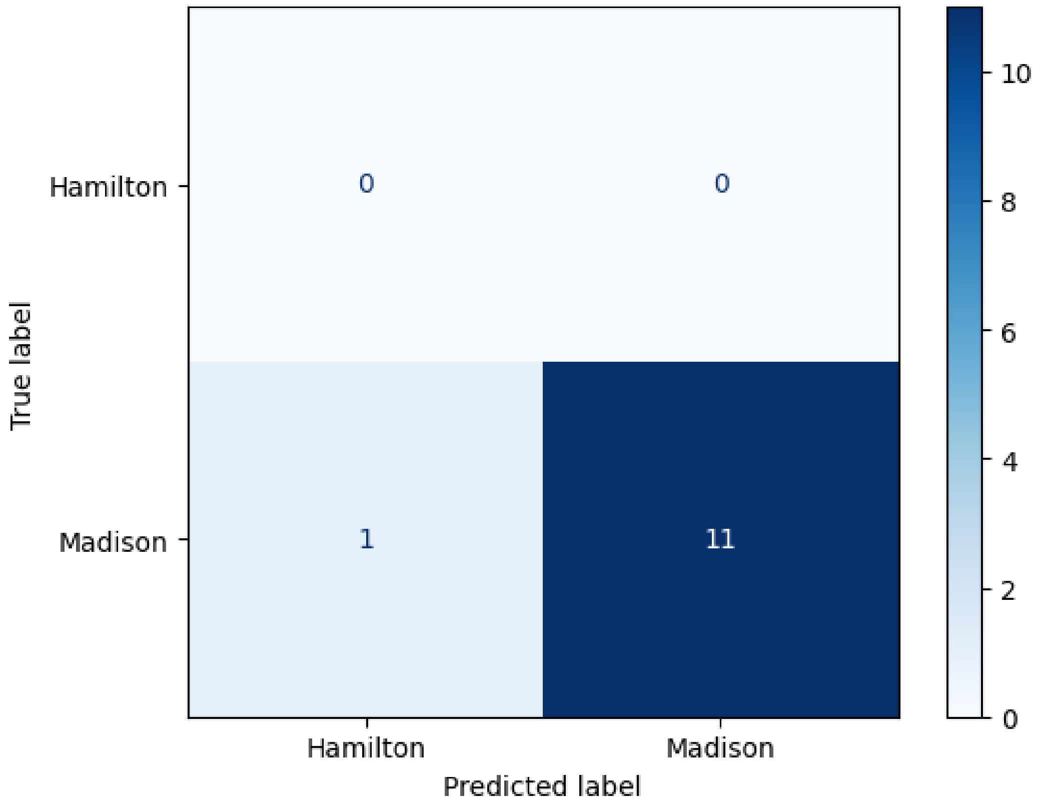
[5]:
```python
from sklearn.naive_bayes import GaussianNB # Import NB

model = GaussianNB() # Init Naive Bayes classifier
model.fit(x_train, y_train) # Fit NB on train
y_pred = model.predict(x_test) # Predict on test set
```

To visualize our predictions, we can plot a confusion matrix. In this matrix, we will have the prediction on the x-axis and the ground truth on the y-axis.

[6]:
```python
import matplotlib.pyplot as plt # Lib to plot
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay #␣
  ↪Confusion matrix
```

```
cm = confusion_matrix(y_test, y_pred, labels=model.classes_) # Define a␣
↪confusion matrix using model classes (authors) as labels
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.
↪classes_) # Plot the matrix
disp.plot(cmap=plt.cm.Blues) # Plot in scale of blues for better visualization
plt.show()
```



As we can see, the model predicted correctly 11 papers out of 12 (1 only assigned to Hamilton). Therefore, the accuracy of our classifier is 91.6%.