Chaithra Kopparam Cheluvaiah
SUID 326926205
ckoppara@syr.edu

HW 05: Use Decision Tree to Solve a Mystery in History

Section 1: Data preparation

About the dataset

The Federalist Papers were a series of eighty-five essays urging the citizens of New York to ratify the new United States Constitution. Written by Alexander Hamilton, James Madison, and John Jay. The essays originally appeared anonymously in New York newspapers in 1787 and 1788 under the pen name "Publius". The Federalist Papers are considered one of the most important sources for interpreting and understanding the original intent of the Constitution.

In this dataset, all 85 papers are titled based on their author, Hamilton, Madison, or Jay. Some essays have dual authorship. Additionally, there are 11 essays with historically disputed authorship.

Additionally, a Term Document Matrix is included which has normalized functional word counts per essay.

Classification Problem

While the authorship of 73 of The Federalist essays is fairly certain, the identities of those who wrote the twelve remaining essays are disputed by some scholars. We are trying to solve this mystery using the decision tree algorithm.

Data Pre-Processing

```
papers = pd.read_csv('data-fedPapers85.csv')
papers.shape
```

```
(85, 72)
```

```
papers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85 entries, 0 to 84
Data columns (total 72 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   author     85 non-null     object
 1   filename   85 non-null     object
 2   a          85 non-null     float64
 3   all        85 non-null     float64
 4   also       85 non-null     float64
 5   an         85 non-null     float64
 6   and        85 non-null     float64
 7   any        85 non-null     float64
 8   are        85 non-null     float64
 9   as         85 non-null     float64
 10  at         85 non-null     float64
 11  be         85 non-null     float64
```

Data Preparation first required loading the .csv file as "papers". Next, the file was generally assessed using papers.info().

Chaithra Kopparam Cheluvaiah
SUID 326926205
ckoppara@syr.edu

Next, we start preparing our data frames for the experiment. First, we established and created a disputed authors data frame that will be the prediction target for the decision tree model. We then created a not disputed authors data frame by excluding all rows where the author is considered disputed. This not disputed data frame was then used to create our training and test datasets.

```
# removing disputed essays from the training and testing data
temp_df = papers[papers['author']!='dispt']
temp_df.head()
```

We then removed the first two columns (author and filename) in order to leave the data frame as just function words and feature values.

```
X = temp_df.iloc[:,2:] # independent variables - removing author and filename
Y = temp_df['author'] # dependent variables
```

These data sets were split using the train_test_split() function where 40% of the data was split into the test file and 60% was split into the training file.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=0, test_size=0.4)
# default 40% is test and 60% is training data
```

```
X_train.shape
```

```
(44, 70)
```

```
X_test.shape
```

```
(30, 70)
```

So, 44 essays are used for training the decision tree classifier and 30 essays for testing the predictions.

Chaithra Kopparam Cheluvaiah
SUID 326926205
ckoppara@syr.edu

```
Y_train.unique()
```

```
array(['Hamilton', 'Madison', 'HM', 'Jay'], dtype=object)
```

```
Y_test.unique()
```

```
array(['Hamilton', 'Jay', 'Madison', 'HM'], dtype=object)
```

Ensuring training and testing essays have the essays of all the authors – Hamilton, Madison, Jay, and Hamilton & Madison.

Section 2: Build and tune decision tree models

We started by creating a general decision tree model using API DecisionTreeClassifier(). We have used ‘entropy’ to measure the purity of the split and to obtain the information gained in every split. We had used ‘Gini impurity’ as well however, there was no significant improvement in the accuracy.

```
# training the model
clf = DecisionTreeClassifier(random_state=0, criterion='entropy')
clf.fit(X_train,Y_train)
```

Chaithra Kopparam Cheluvaiah
SUID 326926205
ckoppara@syr.edu

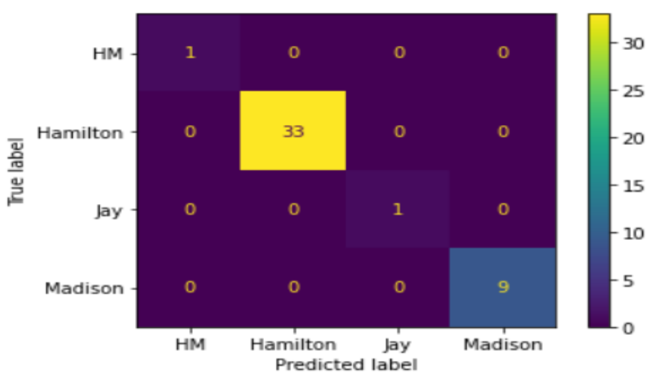
Model Evaluation

From the below classification report on training and testing datasets, We can notice that the accuracy of the model on training data is 100% and on testing data is 80%. We can suspect that our model is overfitting. Also, we have only 74 essays (excluding disputed) to train and test our model. This can be one of the reasons for overfitting. However, We can perform a few post pruning techniques to resolve the problem of overfitting.

```
ConfusionMatrixDisplay.from_estimator(clf,X_train,Y_trai
```

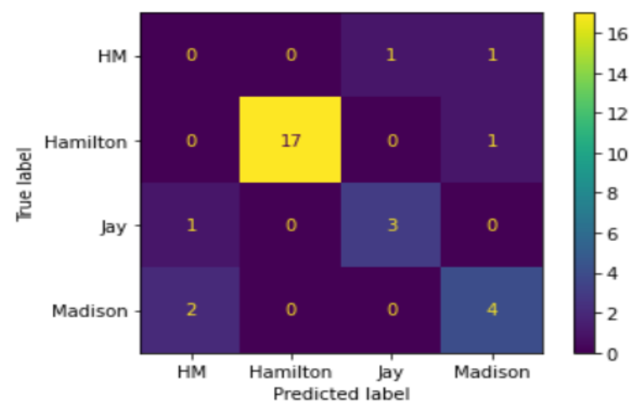
	precision	recall	f1-score	support
HM	1.00	1.00	1.00	1
Hamilton	1.00	1.00	1.00	33
Jay	1.00	1.00	1.00	1
Madison	1.00	1.00	1.00	9
accuracy			1.00	44
macro avg	1.00	1.00	1.00	44
weighted avg	1.00	1.00	1.00	44

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixD
```




```
print(classification_report(Y_test, pred))
```

	precision	recall	f1-score	support
HM	0.00	0.00	0.00	2
Hamilton	1.00	0.94	0.97	18
Jay	0.75	0.75	0.75	4
Madison	0.67	0.67	0.67	6
accuracy			0.80	30
macro avg	0.60	0.59	0.60	30
weighted avg	0.83	0.80	0.82	30



Chaithra Kopparam Cheluvaiah
SUID 326926205
ckoppara@syr.edu

Performing Post Pruning to avoid overfitting

Pruning technique is parameterized by the cost complexity parameter, ’ccp_alpha’. Greater values of ccp_alpha increase the number of nodes pruned. It is necessary to choose the right ‘ccp_alpha’ to cut down the branches of the decision tree.

Based on different ‘ccp_alpha’ values found from the training data, accuracy was plotted for training and testing data sets.

```
path = clf.cost_complexity_pruning_path(X_train, Y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
ccp_alphas
```

-> code showing different ccp_alpha values

```
array([0.        , 0.04545455, 0.17100961, 0.81127812])
```

For different ccp_alpha values, we evaluated the performance of the model on training and testing data. ccp_alpha value between 0.2 - 0.8 resulted in better testing accuracy and training accuracy.

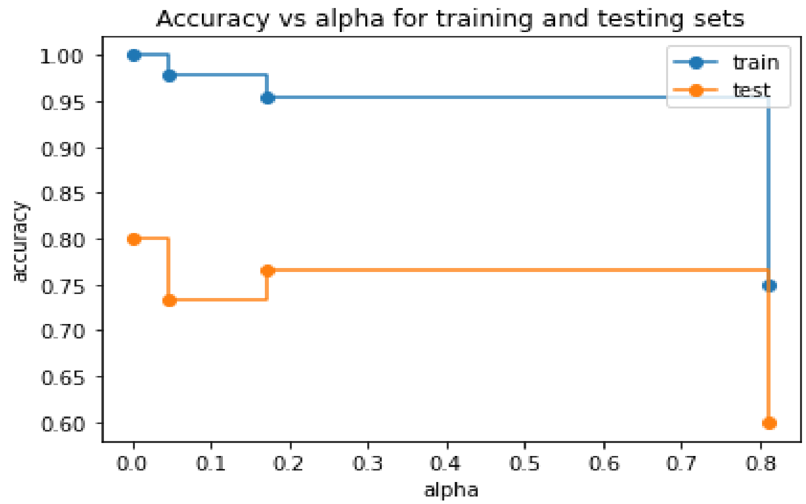
```
from sklearn.metrics import accuracy_score

ccp_alpha = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]

for val in ccp_alpha:
    clf = DecisionTreeClassifier(random_state=0,criterion='entropy',ccp_alpha=val)
    clf.fit(X_train, Y_train)
    accuracy_training = accuracy_score(Y_train, clf.predict(X_train))
    accuracy_testing = accuracy_score(Y_test, clf.predict(X_test))
    print(f'Training Accuracy(ccp_alpha={val}):{accuracy_training},   Testing Accuracy(max_depth={val}):{accuracy_testing}')
```

```
Training Accuracy(ccp_alpha=0.0):1.0,   Testing Accuracy(max_depth=0.0):0.8
Training Accuracy(ccp_alpha=0.1):0.9772727272727273,   Testing Accuracy(max_depth=0.1):0.7333333333333333
Training Accuracy(ccp_alpha=0.2):0.9545454545454546,   Testing Accuracy(max_depth=0.2):0.7666666666666667
Training Accuracy(ccp_alpha=0.3):0.9545454545454546,   Testing Accuracy(max_depth=0.3):0.7666666666666667
Training Accuracy(ccp_alpha=0.4):0.9545454545454546,   Testing Accuracy(max_depth=0.4):0.7666666666666667
Training Accuracy(ccp_alpha=0.5):0.9545454545454546,   Testing Accuracy(max_depth=0.5):0.7666666666666667
Training Accuracy(ccp_alpha=0.6):0.9545454545454546,   Testing Accuracy(max_depth=0.6):0.7666666666666667
Training Accuracy(ccp_alpha=0.7):0.9545454545454546,   Testing Accuracy(max_depth=0.7):0.7666666666666667
Training Accuracy(ccp_alpha=0.8):0.9545454545454546,   Testing Accuracy(max_depth=0.8):0.7666666666666667
Training Accuracy(ccp_alpha=0.9):0.75,   Testing Accuracy(max_depth=0.9):0.6
Training Accuracy(ccp_alpha=1.0):0.75,   Testing Accuracy(max_depth=1.0):0.6
```

Chaithra Kopparam Cheluvaiah
SUID 326926205
ckoppara@syr.edu



From the above plot, cost complexity value between 0.2 to 0.8 seems to be stable with accuracy on training and testing data sets. We decided to proceed with a minimum cost complexity parameter of 0.4 for the model.

Hyperparameter Tuning

Here, we are going to be tuning based on ‘max_depth’. We will try with max depth starting from 1 to 10 and depending on the final ‘accuracy’ score we are going to choose the value of max_depth.

But, surprisingly there was no increase or decrease in the accuracy of the model for any max_depth values. The accuracy remained constant for both training and testing data. The accuracy of training data is 0.95 and testing data is 0.77. So, any value between 1-10 can be used for building the classifier.

The below snapshot shows the code of hyperparameter tuning.

Chaithra Kopparam Cheluvaiah
SUID 326926205
ckoppara@syr.edu

```
max_depth = [1,2,3,4,5,6,7,8,9,10]

for val in max_depth:
    clf = DecisionTreeClassifier(random_state=0,criterion='entropy',ccp_alpha=0.4,max_depth=val)
    clf.fit(X_train, Y_train)
    accuracy_training = accuracy_score(Y_train, clf.predict(X_train))
    accuracy_testing = accuracy_score(Y_test, clf.predict(X_test))
    print(f'Training Accuracy(max_depth={val}):{accuracy_training},   Testing Accuracy(max_depth={val}):{accuracy_testing}')
```

```
Training Accuracy(max_depth=1):0.9545454545454546,   Testing Accuracy(max_depth=1):0.7666666666666667
Training Accuracy(max_depth=2):0.9545454545454546,   Testing Accuracy(max_depth=2):0.7666666666666667
Training Accuracy(max_depth=3):0.9545454545454546,   Testing Accuracy(max_depth=3):0.7666666666666667
Training Accuracy(max_depth=4):0.9545454545454546,   Testing Accuracy(max_depth=4):0.7666666666666667
Training Accuracy(max_depth=5):0.9545454545454546,   Testing Accuracy(max_depth=5):0.7666666666666667
Training Accuracy(max_depth=6):0.9545454545454546,   Testing Accuracy(max_depth=6):0.7666666666666667
Training Accuracy(max_depth=7):0.9545454545454546,   Testing Accuracy(max_depth=7):0.7666666666666667
Training Accuracy(max_depth=8):0.9545454545454546,   Testing Accuracy(max_depth=8):0.7666666666666667
Training Accuracy(max_depth=9):0.9545454545454546,   Testing Accuracy(max_depth=9):0.7666666666666667
Training Accuracy(max_depth=10):0.9545454545454546,   Testing Accuracy(max_depth=10):0.7666666666666667
```

For, our final model (after applying the pruning technique and hyperparameter tuning), we decide to go with minimum cost complexity of 0.4 and the maximum depth of the tree as 10. We trained the model again on the same training dataset.

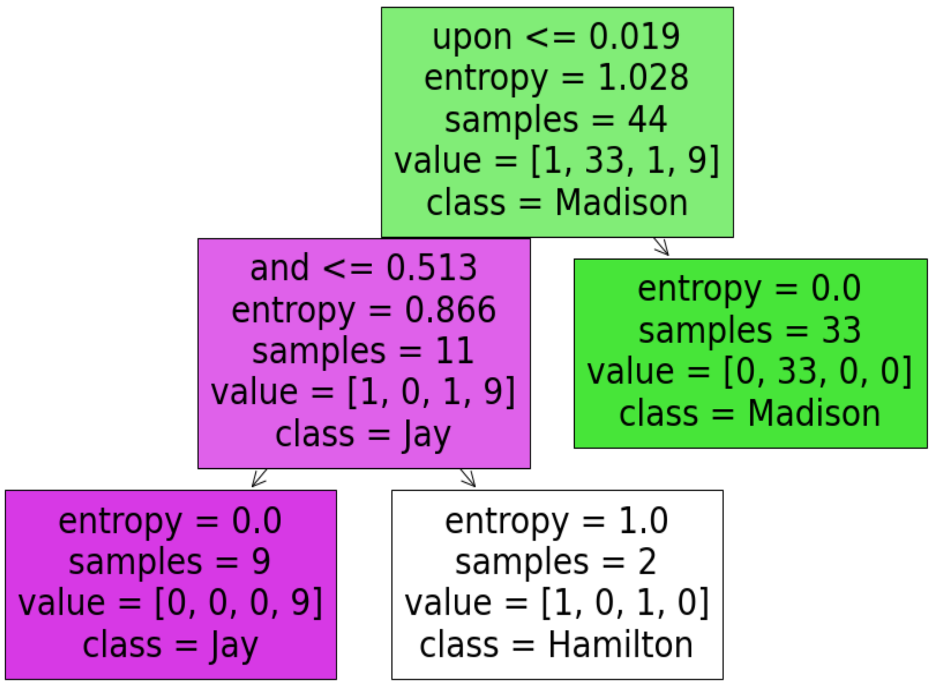
```
clf = DecisionTreeClassifier(random_state=0,criterion='entropy',ccp_alpha=0.1,max_depth=10)
clf.fit(X_train, Y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.1, criterion='entropy', max_depth=10,
                       random_state=0)
```

We then used tree.plot_tree() to visualize the tree and tree.export_text() to obtain all the rules in the string format. We then inspected our first general tree.

- Feature **‘upon’** is having highest information gain of 0.02 and is considered to be the first feature in classifying the authors.

Chaithra Kopparam Cheluvaiah
SUID 326926205
ckoppara@syr.edu



Textual Representaion of the Decision Tree

```
|--- upon <= 0.02
|   |--- and <= 0.51
|   |   |--- class: Madison
|   |--- and >  0.51
|   |   |--- class: HM
|--- upon >  0.02
|   |--- class: Hamilton
```

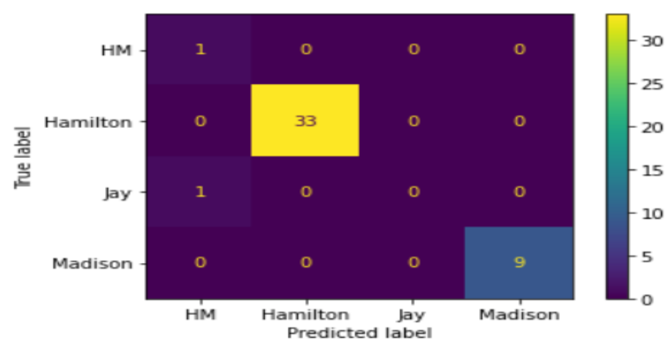
Training data accuracy

```
ConfusionMatrixDisplay.from_estimator(clf,X_train,Y_train)
```

```
              precision    recall  f1-score   support

          HM       0.50      1.00      0.67         1
    Hamilton       1.00      1.00      1.00        33
         Jay       0.00      0.00      0.00         1
     Madison       1.00      1.00      1.00         9

    accuracy                           0.98        44
   macro avg       0.62      0.75      0.67        44
weighted avg       0.97      0.98      0.97        44
```



Chaithra Kopparam Cheluvaiah
SUID 326926205
ckoppara@syr.edu

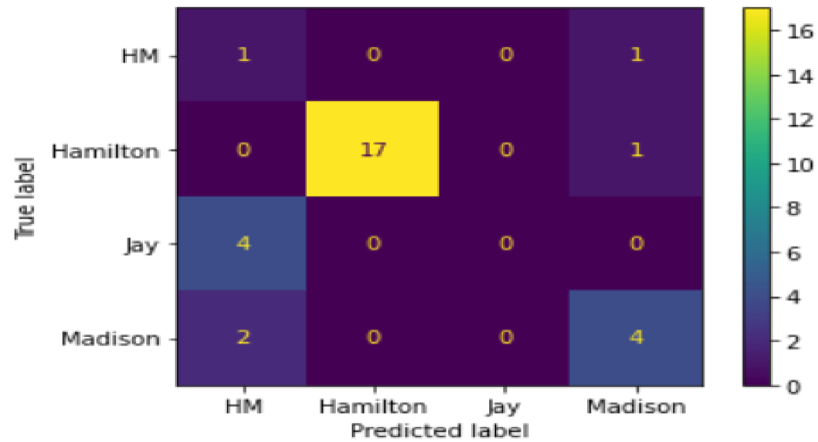
Testing data accuracy

```
print(classification_report(Y_test, pred))
```

```
              precision    recall  f1-score   support

          HM       0.14      0.50      0.22         2
    Hamilton       1.00      0.94      0.97        18
         Jay       0.00      0.00      0.00         4
     Madison       0.67      0.67      0.67         6

    accuracy                           0.73        30
   macro avg       0.45      0.53      0.47        30
weighted avg       0.74      0.73      0.73        30
```



Section 3: Prediction

Federalist papers remain very important documents related to the history of the US constitution. Understanding who is the author of 11 disputed papers is still a mystery. While all the models that we have trained show a stronger likelihood that the author was Madison. However, the results are still inconclusive. Due to the small dataset, models can be overfitting resulting in inaccurate predictions. There are many essays still being classified as HM indicating Hamilton and Madison are helping each other in writing the federal papers.

According to the prediction by the decision tree classifier, the disputed papers are likely written by Madison.

```
dispt_pred=clf.predict(distputed_train)
dispt_pred
```

```
array(['Madison', 'Madison', 'Madison', 'Madison', 'Madison', 'Madison',
       'Madison', 'Madison', 'Madison', 'Madison', 'Madison'],
      dtype=object)
```