Applied Machine Learning

Using a Decision Tree to Solve a Mystery in History

In this project, we are going to use the decision tree algorithm to solve the disputed essay problem.

Report template:

Section 1: Data preparation

You will need to separate the original data set to training and testing data for classification experiments. Describe the contents of your training and test data. What steps did you take?

Section 2: Build and tune decision tree models

First, build a DT model using the default settings, then tune the parameters to see if a better model can be generated. Compare these models using appropriate evaluation measures. Describe and compare the patterns learned in these models.

Section 3: Prediction

After building the classification model, apply it to the disputed papers to find out the authorship. Does the DT model reach the same conclusion the clustering algorithms did? Explain any differences.

Provide your code in a separate script.

<u>Report</u>

The given dataset is about the authors that have written essays in the past for New York newspapers in 1787. I need to find the author for 11 essays around which there is dispute i.e., whether the author is Hamilton or Madison . To solve this disputed essay problem, I have used decision tree algorithm.

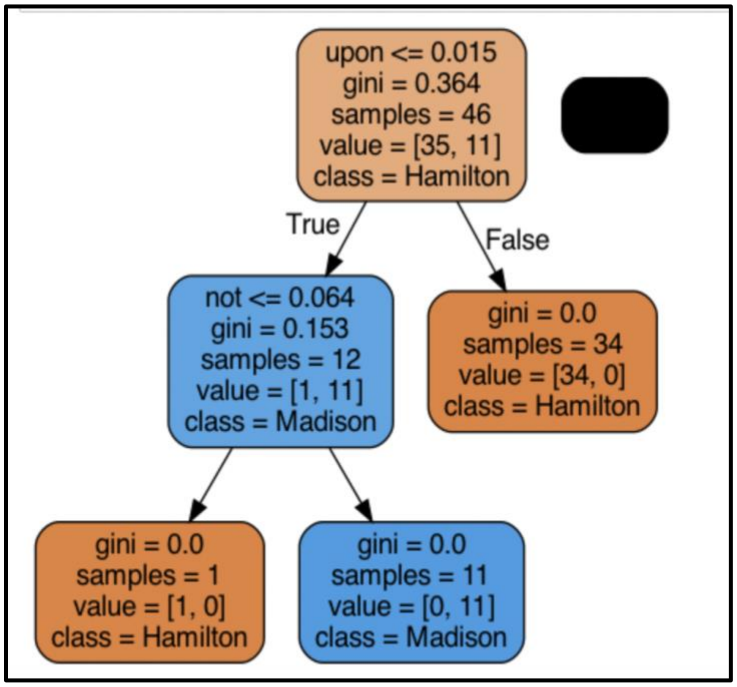
Decision tree algorithm is a supervised machine learning model in which we predict the target value based on set of decision rules.

<u>Data Preparation</u>

In order to prepare data for Decision tree algorithm I checked for null values 1st so that they can be imputed but, in this data, set we don’t have any Null values present. We have **Hamilton** and **Madison** that are considered to be the main authors of disputed essays. So, I removed the entries where the author is either **Jay** or **HM** or **dispt** as they are not the contenders for this. The total essays that we have now are 66 i.e., **51 from Hamilton**, **15 from Madison**. Now, the important step was to separate the independent and the dependent features. All the **independent features** I stored separately in the **X** variable i.e., all the words and dropped the filename as it was not required. I stored my **target variable** i.e., authors in **y** variable. To evaluate the performance of the model I performed the **train-test split**. I split my data into **70-30** ratio that means **70% training data** and **30% test data**. I stored all my independent features in **X_train** and **X_test** variable. All my target values will be in **y_train** and **y_test** as per the split. There will be different number of records in both train and test variables.

<u>Decision Tree Model</u>

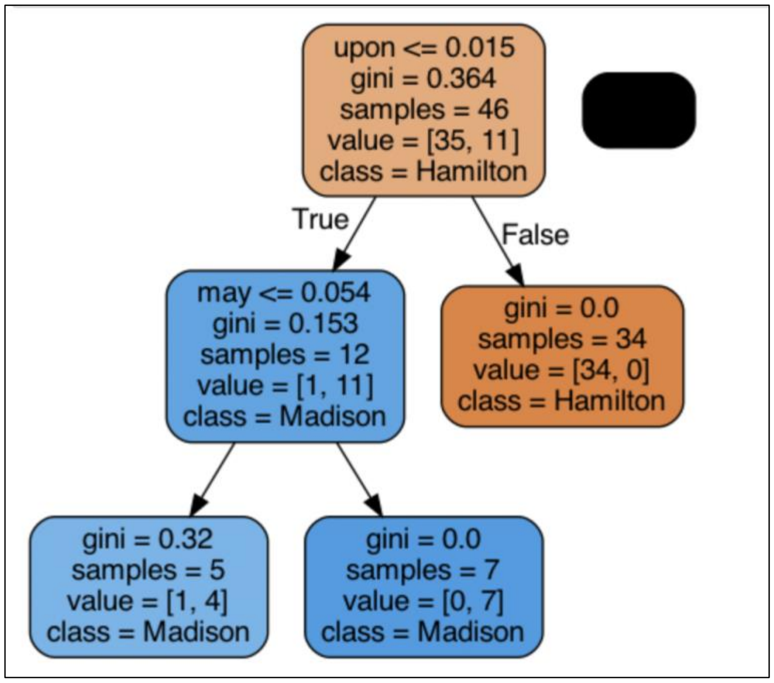
From sklearn I imported the necessary library i.e., **Decision Tree Classifier**. I used **graphviz** and **pydotplus** libraries to create decision tree. The 1st node in decision tree is called the **root node**, the nodes in the middle are called **internal nodes** and the last node is referred as the **leaf node**.



In decision tree we always split on the attribute that has the **maximum information gain** by doing this we get to the node with complete homogeneity i.e., gini index=0 and all the points belong to one particular class. **Gini Index** is defined as the degree of randomly selecting a data point being classified incorrectly. From the decision tree above, in the root node there are **46 samples or essays** where the presence of the word **upon** is less than **0.015%**, **gini impurity is 0.364**. 35 essays belong to Hamilton and 11 belong to Madison before the split. If the percentage of upon **is greater than or equal to 0.015%** then we have **34** essays that belong to **Hamilton** as the gini index is 0. But if the condition is **True** then we proceed to the internal node where we have the next split based on presence of word '**not**' being (**0.064%).** We have **12 essays** for that. If the presence of word '**not**' is **less than or equal to 0.064%** then we have **1** essay that belongs to **class Hamilton** in the leaf node. But if the percentage is greater than **0.064%** then we have 11 essays that belong to Madison. By checking the model performance of my train and test it gives me an **accuracy of about 100% on Train set and 95% on Test set**. This actually shows that model has performed well on the Train data and not so well on the Test data which has led to **overfitting**. We need to carry hyperparameter tuning to avoid overfitting.

<u>Hyperparameter Tuning Decision Tree</u>

In Hyperparameter tuning we pass certain parameters to the learning algorithm to control the training of the model. The choice of hyper parameters has a lot of bearing on the final model that is produced. In Decision tree we use the **GridSearchCV method** to define parameters like **Maximum depth of the tree, Minimum Samples at leaf node and criterion** on which the algorithm would function like **gini or entropy.** Maximum depth of the tree means how long we want the tree to be i.e., the number of nodes we want. Minimum Samples at leaf node means the number of samples required to leaf node. Hyperparameter tuning gives me the best model from the 4 fold Cross validation i.e., running over 200 possibilities. The best model has max depth of 2 and minimum samples at leaf node are 10. **Train Accuracy of the model is 97% and Test Accuracy of the model is 95%. We get the below tree after hyperparameter tuning.**



Prediction

In this last phase I actually created a classification report to get clarity on the essays to reach a conclusion to whom do they belong to. The report shows the main classification metrics **Precision, Recall and F1-score** as per authors. **Precision-** It is defined as the probability i.e., a predicted '**Yes**' is actually a '**Yes**'. **Recall-**It is the probability to find out all the actual '**Yes**' cases and out of that how we could get right. F1- is defined as the harmonic mean of Precision and Recall. In this model our major focus is to see the Precision for each author i.e., to check how many times we predicted the author to be Hamilton or Madison and actually it was Hamilton or Madison. For **Hamilton** we have a **Precision of 94%** that means model has, both in Actual and Predicted scenario model has confirmed Hamilton was the author .

For Madison we have Precision to be **100% means that model predicted the author to be Madison when in actual scenario was also Madison**. From this we can actually conclude that for our **11 disputed essay's we will have Madison as the author** since we get a **Precision of 100%** whereas in case of Hamilton even though we have high Precision i.e., 94% we have a chance of making an error of 6% in that case. All the essays written by **Hamilton** have presence of word '**upon**' greater than 0.015% and all the essays written by Madison have presence of word '**may**' to be **0.054%** in both greater than and less than cases that mean disputed essays will also have this presence.. The result is similar to what we got in our Clustering algorithm.