# LAB 3 WRITEUP

## Documentation:

**Running  Train Decision Tree -**

The decision Tree train starts from the main and then goes to the function Decision_tree(dataframe, depth=0, max_depth=3, used_attributes=None) which carries out the decision tree for x depth. I chose a max depth of 3 to reduce Overfitting, a value that's not too high - also fewer splits so easier to look at. After Decision Tree is done it returns a model file with the content needed to run Predict Decision Tree

**Running  Predict Decision Tree -**

**Note:** Both predictions are distinctly ran based on contents of .model file, in my main I have functionality that checks for numbers in the .model file, if there's numbers it's a model for An Ada boost prediction, if it's just strings then Decision Tree.

- If you ever want to see the .model data you can also debug or remove a .close in main to see "hypotx.txt" which is a text file to see the .model content( I first programmed this without knowing that .model files would be used so needed to adapt).

Predict Decision Tree starts from the predict function, which goes through a created dictionary list that links branches together. This linkage is used to determine the endpoint for a sentence once it goes down a certain branch, to either derive en or dl based on which attributes it gets true or false along the way.

**Running  Train AdaBoost -**

Training AdaBoost starts from the Ada boost function: Ada_Boost(dataframe, K, hypothesisOut). K represents the amount of weak learners Ada Boost will use. I decided to go with 7 because I didn't want to go over double integer values to avoid complex data. I also wanted a good balance that wasn't too complex. The value of 7 allows for a somewhat even split, but also a "favor" factor with the additional 1 to a 6(even split) I believe this can help separate really good and not too good learners, with one also hopefully being a decider moreover to one side. Ada_Boost will run with Decision_tree_stump - this function must run to get the weak learners, this data is then put into Ada_Boost which updates weights - In order to see finished results the depth of iterations on ADABOOST must be completed.  After this, weights, favored attributes from weak learner and label determine are parsed in an array for model file for prediction.

**Running  Predict AdaBoost -**

Using the model file created by train ADABoost, the function predict_ADA(examples, features, hypothesis) takes in the model file, which has a list that corresponds to the attributes explored by each weak learner, they're language and the weights. Each sentence is then checked to see if it contains that attribute, and then assigns weight accordingly( multiply. + for ENG. - for DUTCH)  to that sentence's value. A value must be + or negative too then determine en or dl.

## **Results:**

- when using more than 20 features, I found accuracy mostly remained the same so stayed at 20

- When using larger cycles for ADABoost I didn't get too much of a difference compared to smaller values, anything below 7 cycles was sometimes less efficient but above 4 cycles and above generally gave same accuracy

- Depths on the Decision Tree were the wildest variable, The Decision Tree seemed to do decent on limited data when I did predictions on depths of 1 and 2. But after adding more robust features and trying depths that extended more and even explored all attributes on each branch side did better, a depth of 3 was good enough for the test cases as the features split on would give branches with all the same labels often.

## **Features:**

- I chose the features of very distinct adjectives and "play on" words both languages used themselves. After that I picked some words that had a lot of the lower alphabet letters for Dutch, as I found letters from the l - z range with j's and other "jah"/""zuh" sounding words alongside I - Z in Dutch were distinct.

## **BEST.MODEL:**

After running several test cases, I determined my Decision Tree was usually faster, so my best.model represents my prediction of decision tree model data. For the best model I trained around 50,000 lines of English and Dutch with my decision tree, giving me a robust and high classifying tree.

## *Examples to use/Numerical Results of Examples:*

If you run **myexamples.txt** as the parameter for examples in **train** for ada  or dt, you should get the following results below. **run examples.dat** right after for **predict** for ada or dt  - so these results would be based on the choices made from the trainer's process  on myexamples.txt

**(correct and wrong are not part of print)**

results for predict  decision tree: 90% accuracy

```
nl     correct
en    correct
nl     wrong
nl     correct
en    correct
en   correct
nl    correct
en   correct
nl    correct
en  correct
```

results for predict ada boost: 90 % accuracy

```
nl        correct
en        correct
nl        wrong
nl        correct
en        correct
en        correct
nl        correct
en        correct
nl        correct
en        correct
```

- Other tests should work as well if trained than predicted.

## Packages Used:

- pandas - dataframe for storage
- sys - cli arguments
- random - when an even number of   both label types on an ending condition branch choose randomly.

- math - log functions
- pickle - parse .model files