1. Implementation

a) I'll try to describe at a high level the step I took in order to get to a decision tree:

A. Using the class ***DataSetLoader***, I am loading the training data as an array having the following form:

```
[
    ['Wil van der Aalst', '-'],
    ['Scott Aaronson', '+'],
    …
]
```

B. I use then the ***DataSetFeaturesEnricher*** to create the features for each row. At the end I get an array having the following form:

```
[
    [False, False, False, False, True, False, '+'],
    [False, True, False, True, False, True, '-'],
    …
]
```

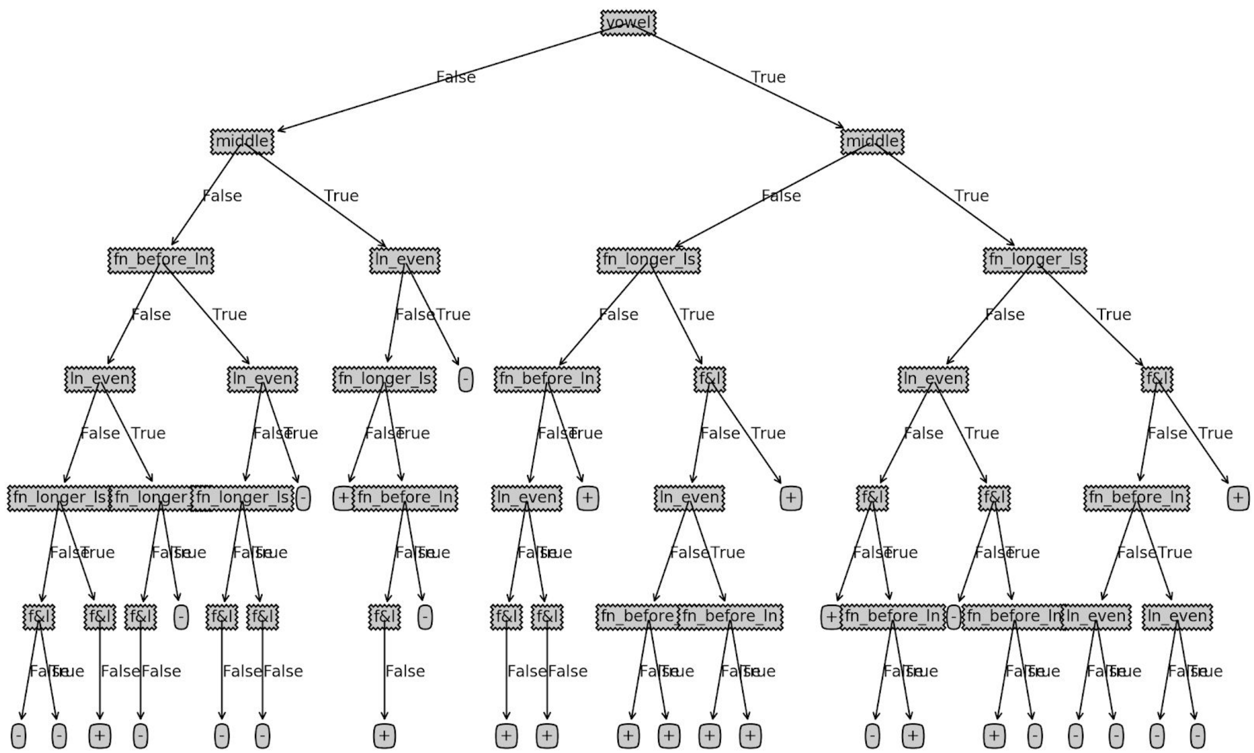
The first 6 columns are the features values, while the last is the label for that entry.

C. The array from point B is sent to the ***DecisionTree*** class where using the ID3 algorithm I return the tree having the form:

```
{
    'vowel': {
        False: {
            'middle': {
                False: '-',
                True: {... }
                }
            },
        True: {
            'middle': {
                False: '+',
                True: {
                    'ln_even': {
                        False: {... },
                        True: {... }
                    }
                }
            }
        }
}
```

}

D. I then render my tree using the ***mathplotlib*** module. I get the following image after I train a decision tree using the 6 features listed in the ***feature_description.txt*** file:

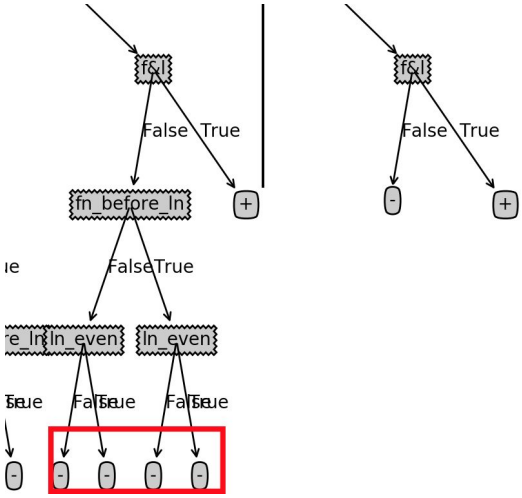


Legend:

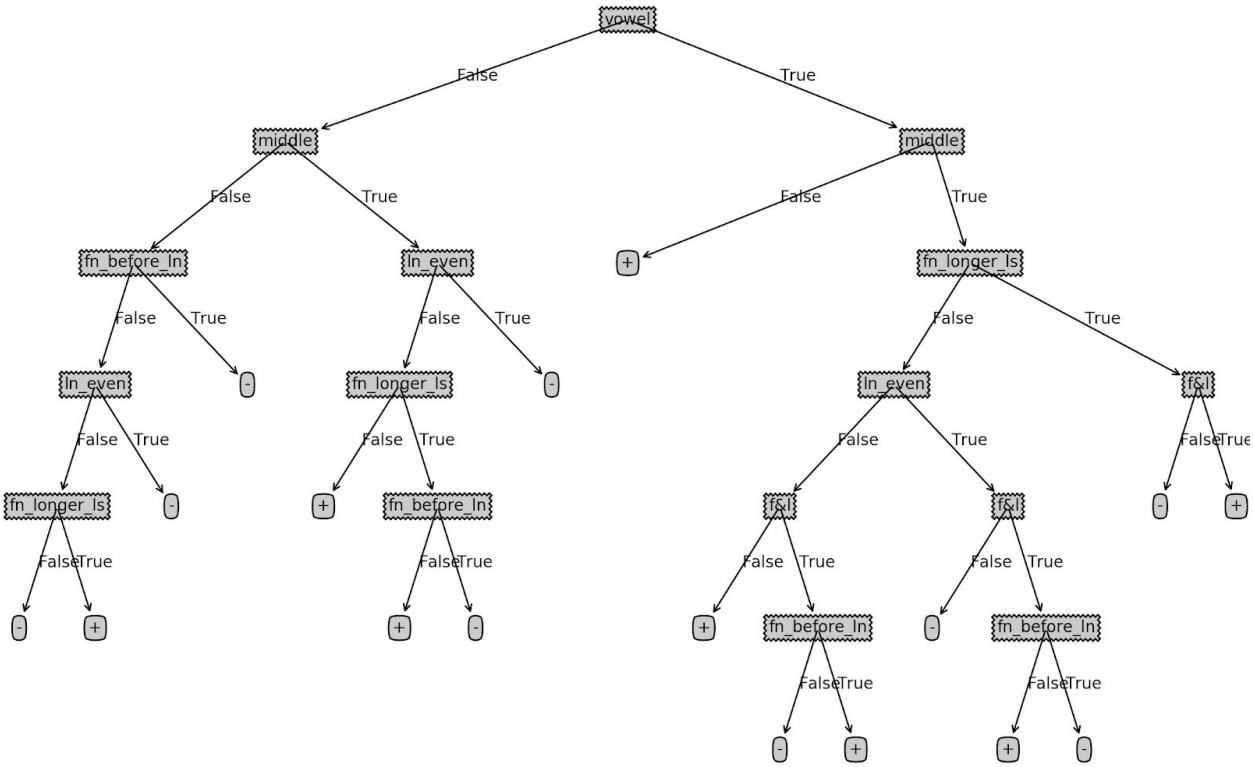
1. vowel - Is the second letter of their first name a vowel (a,e,i,o,u)?
2. middle - Do they have a middle name?
3. fn_even - Is the number of letters in their last name even?
4. fn_longer_ls - Is their first name longer than their last name?
5. fn_before_ln - Does their first name come alphabetically before their last name?
6. f&l - Does their first name start and end with the same letter?

E. In the previous picture you can notice that some features leaves are returning the same label on both branches (True/False). I created a class called ***TreePruner*** where I try to fix this.

For example the 4 leaves from below are becoming only one (having label ‘-’) under the `f&l` label.



After the pruning operation I get to this final tree:



F. After I get this tree, I send it to a **Classifier** class that will return labels for any new entry. Each new entry should be an array of 6 features.

b) Suggest at least 4 other features you could have extracted from this dataset.

1. The number of vowels in the full name is odd;
2. Last letter of the last name is vowel;
3. Does the full name contains more than 2 names;
4. Does the full name contains less than 15 characters.

c) Report the error of your decision tree on the **Dataset/training.data** file.
The error rate that I got on the training data was: **7.19%**

d) Report the error of your decision tree on the ***Dataset/test.data*** file.
The error rate that I got on the test data was: **8.11%**

e) Report the maximum depth of your decision tree.
The maximum depth of my decision tree is 6 (equal with the number of features used).

2. Limiting depth

a) Using the steps described in the homework description I used 10 features (6 feature listed in the ***feature_descriptions.txt*** file and the 4 features that I listed inside the **Implementation** section (b)) to train the tree using the k-fold cross validation method. I got to the following values:

Depth	Error rate when **training00.data** was used as test set	Error rate when **training01.data** was used as test set	Error rate when **training02.data** was used as test set	Error rate when **training03.data** was used as test set	Average error rate	Standard deviation
1	13.51%	18.92%	18.92%	18.02%	17.34%	2.59
2	14.41%	11.71%	10.81%	10.81%	11.94%	1.7
3	14.41%	9.91%	9.01%	10.81%	11.04%	2.37
4	14.41%	8.11%	7.21%	6.31%	9.01%	3.67
5	**9.01%**	**6.31%**	**8.11%**	**8.11%**	**7.88%**	**1.13**
10	10.81%	10.81%	13.51%	7.21%	10.59%	2.59

We can see that a tree having depth 5 has the lowet error rate (7.88%) while also has the lowest standard deviation. That means that the values used for calculating the average are not spread.
b) Using the depth with the greatest cross-validation accuracy from your experiments: train your decision tree on the Dataset/training.data file. Report the accuracy of your decision tree on the Dataset/test.data file.

I tested a decision tree using all the training data, but limiting the depth of the tree to 5. The error rate that I got on the testing data using the tree with depth 5 was: **5.41%**

c) Discuss the performance of the depth limited tree as compared to the full decision tree. Do you think limiting depth is a good idea? Why?

When we limited the depth to 5 (the initial tree had the depth 6), the error rate on the test data went from 8.11% down to 5.41%. I think the initial tree was overfitted on the training data. By cutting the depth with one level, we removed a feature that was not bringing a lot of ordering in the data (that feature had the lowest entropy gain) but was making at the same time the tree overfitted. Not using all the features, make the tree more generic.

3. Extra validation

For validation my tree, I used also the decision tree classifier from the SciKit project (http://scikit-learn.org/stable/modules/tree.html). The SciKit classifier uses a optimized version of CART. The tree that I got is similar with the one created by my scipt:


second_letter_of_their_first_name_a_vowel ≤ 0.5
gini = 0.469
samples = 445
value = [278, 167]
class = +
True
False
has_middle_name ≤ 0.5
gini = 0.245
samples = 126
value = [18, 108]
class = -
has_middle_name ≤ 0.5
gini = 0.301
samples = 319
value = [260, 59]
class = +
first_name_come_alphabetically_before_their_last_name ≤ 0.5
gini = 0.13
samples = 100
value = [7, 93]
class = -
is_the_number_of_last_name_letters_even ≤ 0.5
gini = 0.488
samples = 26
value = [11, 15]
class = -
first_name_longer_that_last_name ≤ 0.5
gini = 0.158
samples = 266
value = [243, 23]
class = +
first_name_longer_that_last_name ≤ 0.5
gini = 0.436
samples = 53
value = [17, 36]
class = -
is_the_number_of_last_name_letters_even ≤ 0.5
gini = 0.245
samples = 35
value = [5, 30]
class = -
is_the_number_of_last_name_letters_even ≤ 0.5
gini = 0.06
samples = 65
value = [2, 63]
class = -
first_name_longer_that_last_name ≤ 0.5
gini = 0.337
samples = 14
value = [11, 3]
class = +
gini = 0.0
samples = 12
value = [0, 12]
class = -
first_name_come_alphabetically_before_their_last_name ≤ 0.5
gini = 0.072
samples = 187
value = [180, 7]
class = +
is_the_number_of_last_name_letters_even ≤ 0.5
gini = 0.323
samples = 79
value = [63, 16]
class = +
is_the_number_of_last_name_letters_even ≤ 0.5
gini = 0.5
samples = 28
value = [14, 14]
class = -
first_name_starts_and_ends_with_same_letter ≤ 0.5
gini = 0.211
samples = 25
value = [3, 22]
class = -
first_name_longer_that_last_name ≤ 0.5
gini = 0.332
samples = 19
value = [4, 15]
class = -
first_name_longer_that_last_name ≤ 0.5
gini = 0.117
samples = 16
value = [1, 15]
class = -
first_name_longer_that_last_name ≤ 0.5
gini = 0.105
samples = 36
value = [2, 34]
class = -
gini = 0.0
samples = 29
value = [0, 29]
class = -
gini = 0.0
samples = 7
value = [7, 0]
class = +
first_name_come_alphabetically_before_their_last_name ≤ 0.5
gini = 0.49
samples = 7
value = [4, 3]
class = +
is_the_number_of_last_name_letters_even ≤ 0.5
gini = 0.137
samples = 95
value = [88, 7]
class = +
gini = 0.0
samples = 92
value = [92, 0]
class = +
first_name_come_alphabetically_before_their_last_name ≤ 0.5
gini = 0.255
samples = 40
value = [34, 6]
class = +
first_name_come_alphabetically_before_their_last_name ≤ 0.5
gini = 0.381
samples = 39
value = [29, 10]
class = +
first_name_starts_and_ends_with_same_letter ≤ 0.5
gini = 0.231
samples = 15
value = [13, 2]
class = +
first_name_starts_and_ends_with_same_letter ≤ 0.5
gini = 0.142
samples = 13
value = [1, 12]
class = -
first_name_come_alphabetically_before_their_last_name ≤ 0.5
gini = 0.153
samples = 24
value = [2, 22]
class = -
gini = 0.0
samples = 1
value = [1, 0]
class = +
first_name_starts_and_ends_with_same_letter ≤ 0.5
gini = 0.219
samples = 16
value = [2, 14]
class = -
gini = 0.444
samples = 3
value = [2, 1]
class = +
gini = 0.165
samples = 11
value = [1, 10]
class = -
gini = 0.0
samples = 5
value = [0, 5]
class = -
gini = 0.071
samples = 27
value = [1, 26]
class = -
gini = 0.198
samples = 9
value = [1, 8]
class = -
gini = 0.32
samples = 5
value = [4, 1]
class = +
gini = 0.0
samples = 2
value = [0, 2]
class = -
gini = 0.111
samples = 51
value = [48, 3]
class = +
gini = 0.165
samples = 44
value = [40, 4]
class = +
gini = 0.33
samples = 24
value = [19, 5]
class = +
first_name_starts_and_ends_with_same_letter ≤ 0.5
gini = 0.117
samples = 16
value = [15, 1]
class = +
first_name_starts_and_ends_with_same_letter ≤ 0.5
gini = 0.278
samples = 24
value = [20, 4]
class = +
first_name_starts_and_ends_with_same_letter ≤ 0.5
gini = 0.48
samples = 15
value = [9, 6]
class = +
gini = 0.0
samples = 9
value = [9, 0]
class = +
first_name_come_alphabetically_before_their_last_name ≤ 0.5
gini = 0.444
samples = 6
value = [4, 2]
class = +
gini = 0.0
samples = 11
value = [0, 11]
class = -
first_name_come_alphabetically_before_their_last_name ≤ 0.5
gini = 0.5
samples = 2
value = [1, 1]
class = +
is_the_number_of_last_name_letters_even ≤ 0.5
gini = 0.142
samples = 13
value = [1, 12]
class = -
is_the_number_of_last_name_letters_even ≤ 0.5
gini = 0.165
samples = 11
value = [1, 10]
class = -
gini = 0.231
samples = 15
value = [2, 13]
class = -
gini = 0.0
samples = 1
value = [0, 1]
class = -
gini = 0.124
samples = 15
value = [14, 1]
class = +
gini = 0.0
samples = 1
value = [1, 0]
class = +
gini = 0.287
samples = 23
value = [19, 4]
class = +
gini = 0.0
samples = 1
value = [1, 0]
class = +
gini = 0.497
samples = 13
value = [7, 6]
class = +
gini = 0.0
samples = 2
value = [2, 0]
class = +
gini = 0.0
samples = 2
value = [0, 2]
class = -
gini = 0.0
samples = 4
value = [4, 0]
class = +
gini = 0.0
samples = 1
value = [1, 0]
class = +
gini = 0.0
samples = 1
value = [0, 1]
class = -
gini = 0.0
samples = 7
value = [0, 7]
class = -
gini = 0.278
samples = 6
value = [1, 5]
class = -
gini = 0.278
samples = 6
value = [1, 5]
class = -
gini = 0.0
samples = 5
value = [0, 5]
class = -