NeuroComputation
Part 2 Adaline classifications

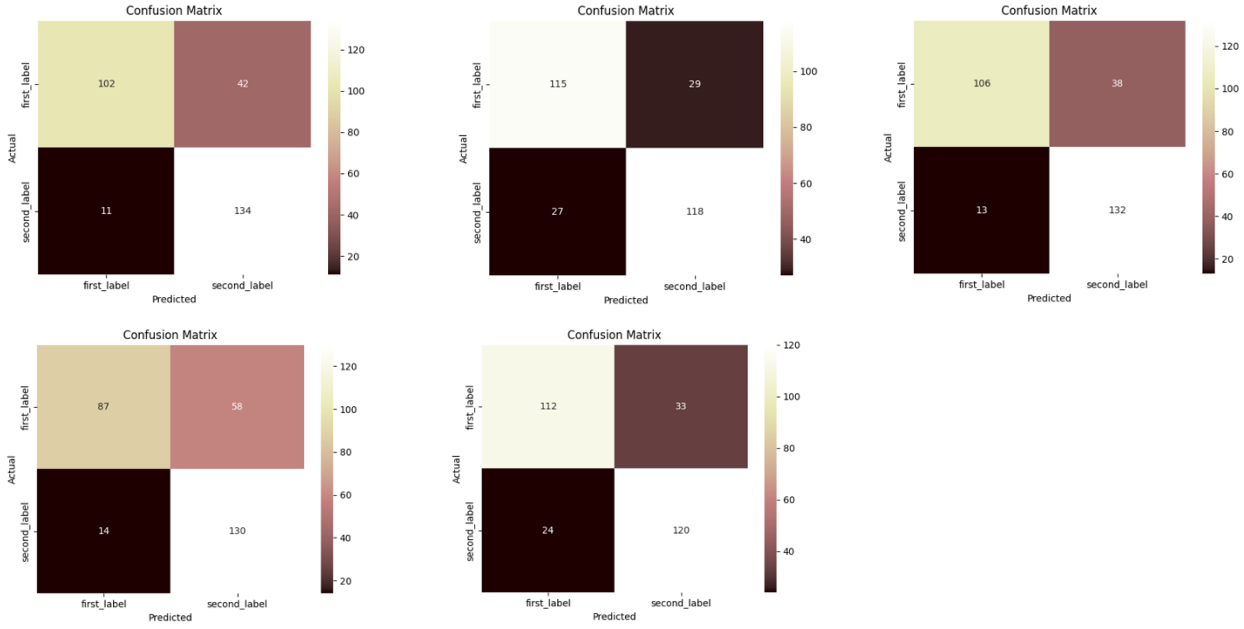
Roni Pick – 206794075
Or Kazu Cohen – 206585515

Results:

In the following confusing matrixes, we represent for each classification the 5 folds of the data. Our learning rate was 0.001, and the number of epochs was 600.

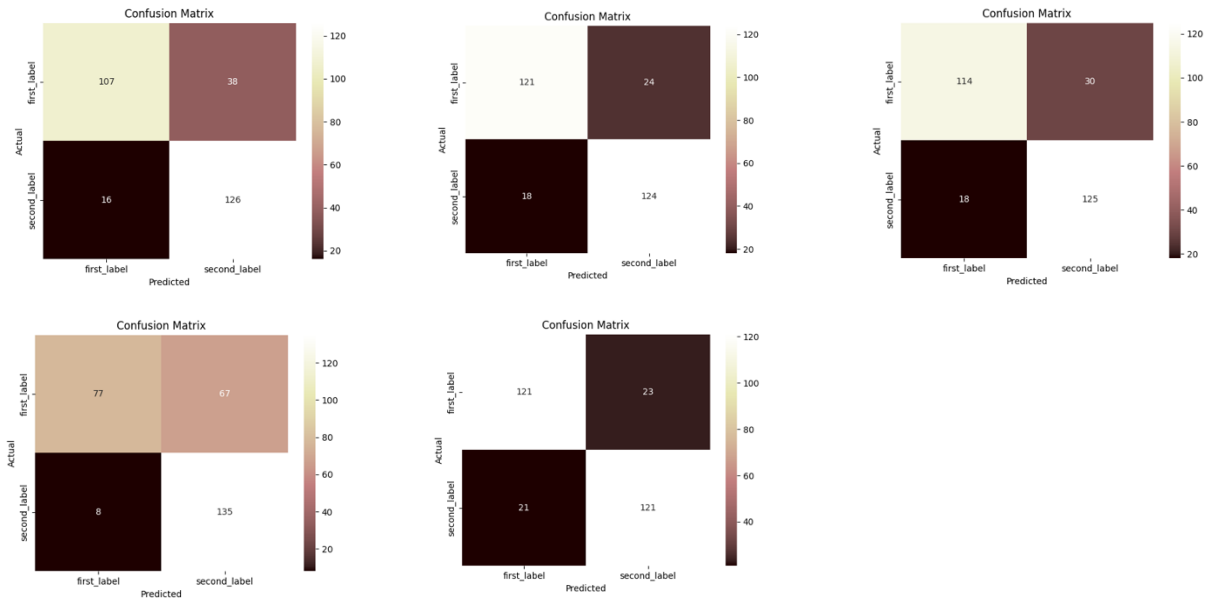
“mem” versus “bet” classification:

In these confusing matrixes, first_lable represent "mem" and second_lable represent "bet"
Mean accuracy: 80%
Std deviation: 0.03
As we can see, there is a lot of miss classification between these two, the algorithm finds it more difficult to recognize the "bet" letter and he classifies it as "mem".
The reason is that those letters look very similar to each other.



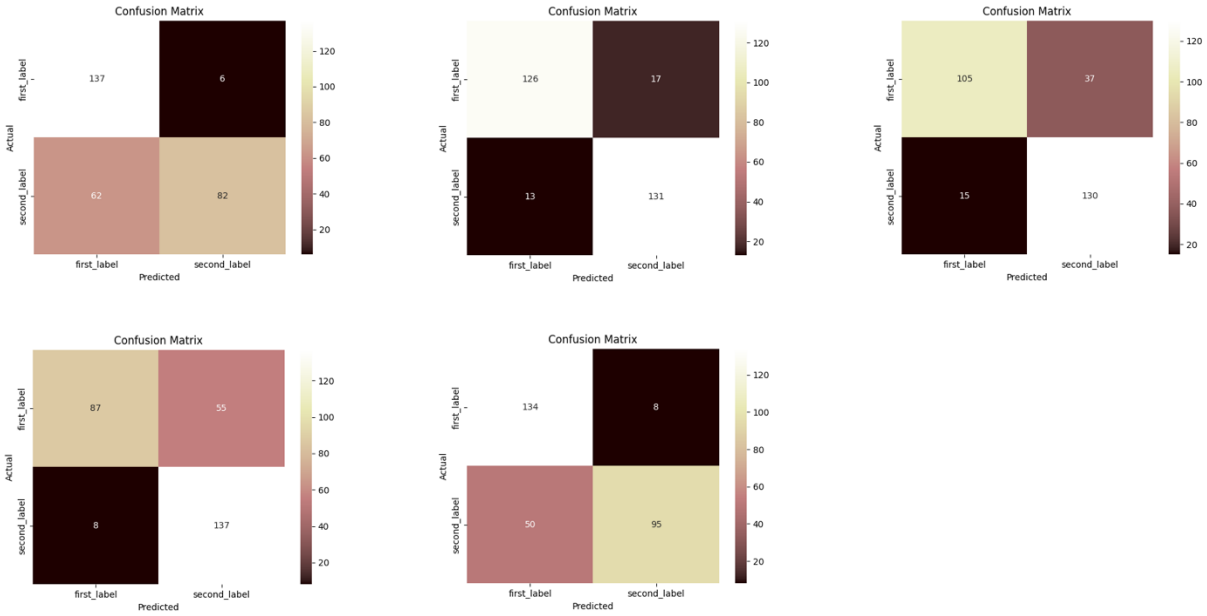
“lamed” versus “bet” classification:

In these confusing matrixes, first_lable represent "lamed" and second_lable represent "bet"
Mean accuracy: 82%
Std deviation: 0.04
As we can see, there is less miss classification between these two then the previews two.
Here the algorithm finds it more difficult to recognize the "lammed" letter and he classifies it as "bet".



“lamed” versus “mem” classification:

In these confusing matrixes, first_lable represent "mem" and second_lable represent "lamed"
Mean accuracy: 83%
Std deviation: 0.01
As we can see, here is our best result by the mining of minimum miss classification between the letters. Here the algorithm finds it more difficult to recognize the "lammed" letter and he classifies it as "mem".



Conclusion:

We can learn from the results that the miss classification of our algorithm is bigger in "lammed" then in "mem" and "bet", and the highest error rate between two latters are between "bet" and "mem".

Additionally:

1. In order to implement the Adaline algorithm, we got help from this website: https://notebook.community/stellaxux/machine-learning-in-python/ch2/Implement%20Adaline%20algorithm
2. Our code has 4 parts – main.py, Adaline.py, create_dataset.py, pre_proccesing.py:
    - main.py – creating the datasets array, sending it to create_dataset.py and then sending to pre_proccesing.py to train and test the model.
    - create_dataset.py – receiving the data from the main, checking validation and add each label to the correct label array (by 1="bet", 2="lamed", 3="mem"), and add the data to the correct latter array.
    - Adaline.py – all the Adaline's algorithm functions.
    - pre_proccesing.py – the training of the model, we sent the data from the main, each time we sent 2 letters as required.