NeuroComputation
Part 3 Adaline classifications

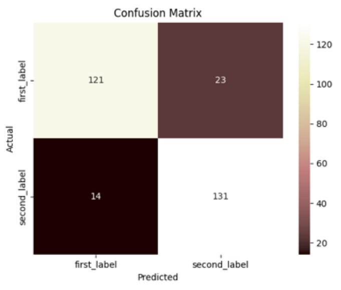
Roni Pick – 206794075
Or Kazu Cohen – 206585515

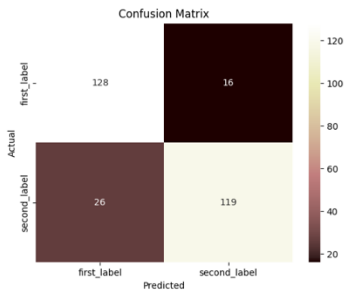
Results:

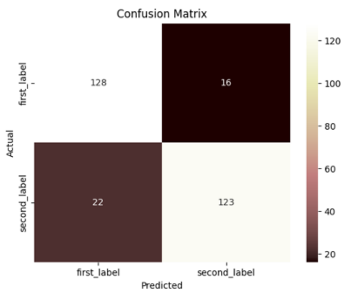
In the following confusing matrixes, we represent for each classification the 5 folds of the data. We will also show the differences between the Adaline results vs the Feed Forward results.

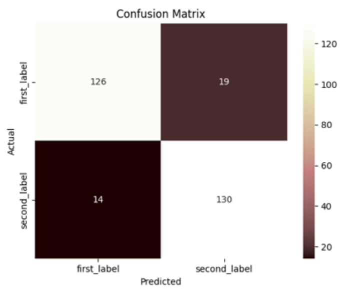
“mem” versus “bet” classification:

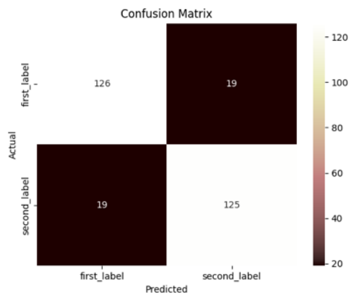
In these confusing matrixes, first_lable represent "mem" and second_lable represent "bet"
Mean accuracy: 87.06%
Std deviation: 0.03
We can see here an improvement from the Adaline algorithm where the mean accuracy was 80% – improvement of 7.03%. However, the Std deviation remains the same – 0.03.





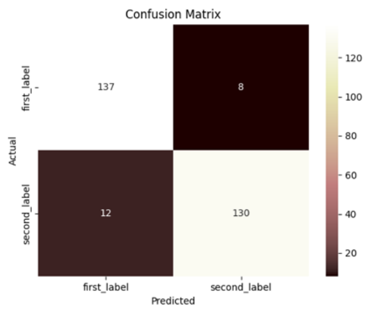


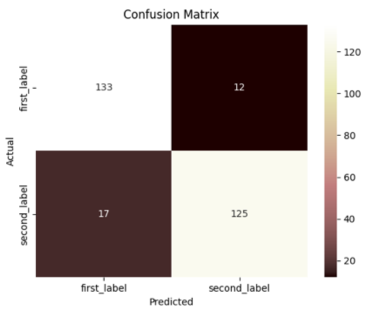


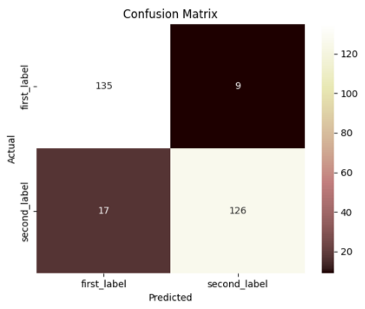


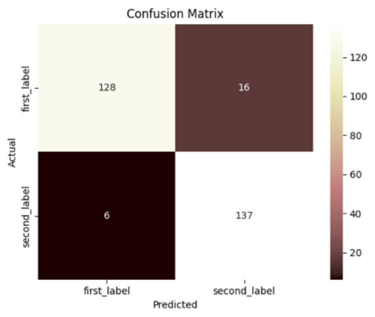
“lamed” versus “bet” classification:

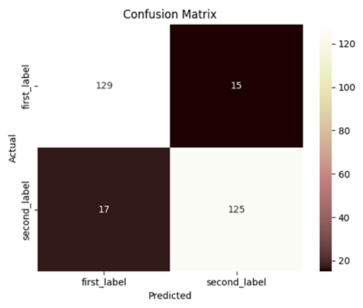
In these confusing matrixes, first_lable represent "lamed" and second_lable represent "bet"
Mean accuracy: 91.42%
Std deviation: 0.02
We can see here an improvement from the Adaline algorithm where the mean accuracy was 82% – improvement of 9.42%, and in the Std deviation that was 0.04 – improvement of 0.02.





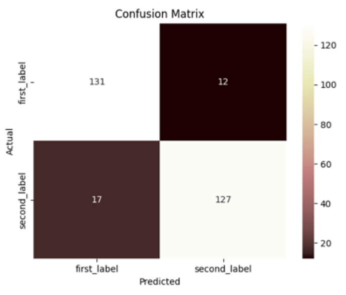


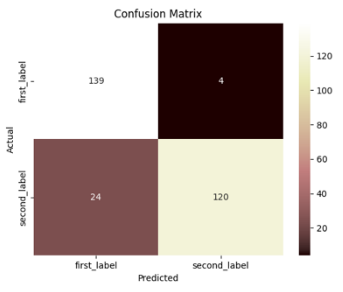


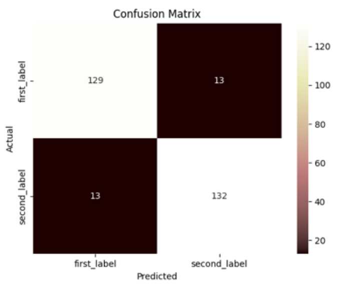


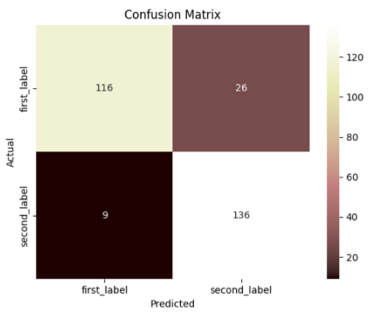
“lamed” versus “mem” classification:

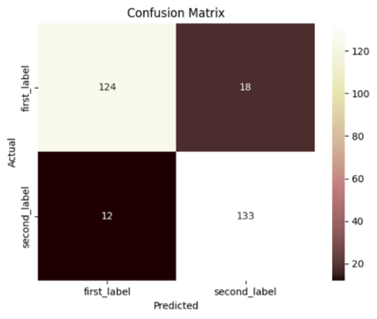
In these confusing matrixes, first_lable represent "mem" and second_lable represent "lamed"
Mean accuracy: 89.06%
Std deviation: 0.01
We can see here an improvement from the Adaline algorithm where the mean accuracy was 83% – improvement of 6.06%, however the Std deviation remains the same – 0.01.











Explanation:

In this part we were asked to replace the Adaline with a feed forward neural network, using our previous part code.
In the implementation we decided to use 3 layers:

1. A dense layer with 128 units and "relu" activation function and the input dimension is set to "input dim":
   we decided on those number of neurons and levels because it's not too small to potentially lose important information, and not too large to risk overfitting the data.
   The input dimension set to input dim to match the dimensionality of the input data.

2. A dense layer with 64 units and "relu" activation function:
   In this case, 64 neurons were chosen, which is a moderately sized hidden layer.
   It is not too small or too large and allowing the network to capture a reasonable level of complexity without risking overfitting.

   In both (1) and (2) we chose the "relu" activation function helps introduce non-linearity into the model, allowing the network to learn more complex representations and enabling faster training.

3. A dense layer with "num classes" units and "softmax" activation function where "num classes" represents the number of output classes in the problem, and each neuron in the output layer corresponds to a specific class.
   We used the "softmax" activation function to produce class probabilities.

We have compiled the model with "adam" optimizer, "categorical_crossentropy" loss function, and use "accuracy" as the metric for evaluation.

Conclusion:

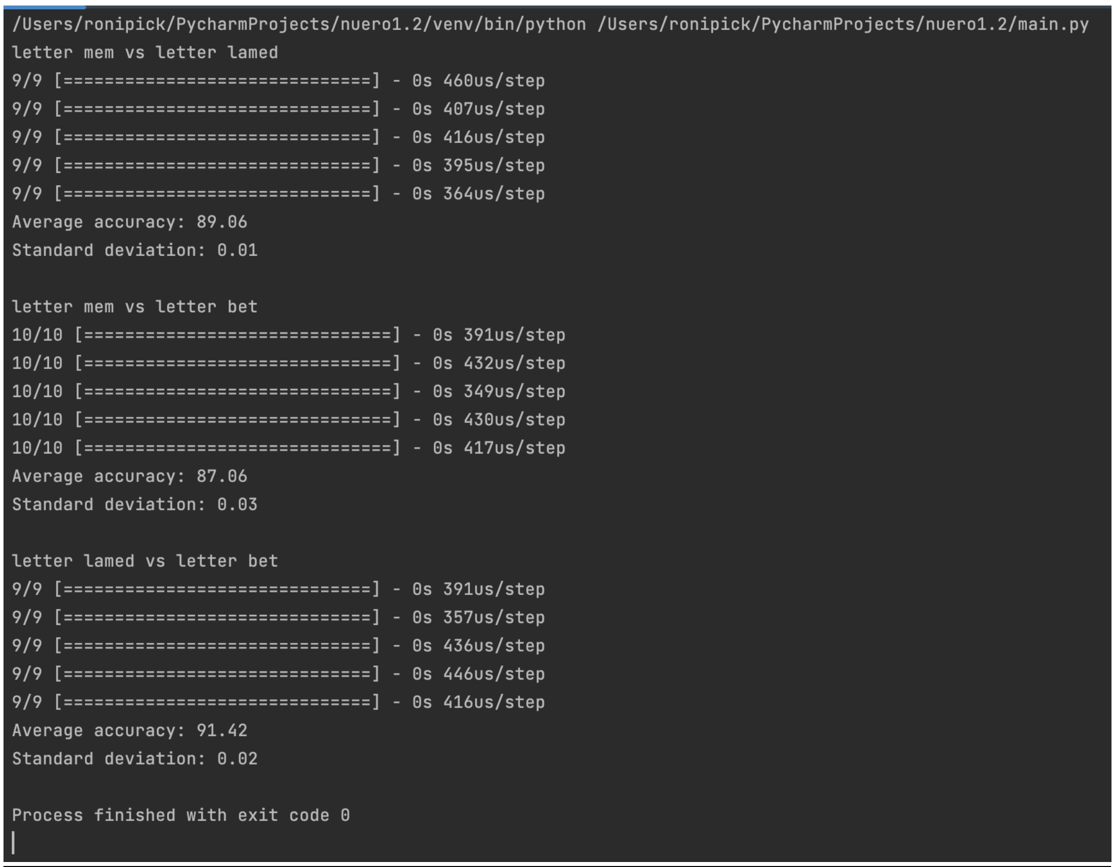
We can learn from the results that the feed forward algorithm produces better results thanks to the fact that we add more neuron layers that helps the model to learn more features about the data.
we can still see that the miss classification in our algorithm between two letters are between "bet" and "mem" with the highest error rate as with the Adaline algorithm.

Additionally:

Our code has 4 parts – main.py, feedForward.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.
- feedForward.py – the Feed Forward's algorithm.
- pre_proccesing.py – the training of the model, we sent the data from the main, each time we sent 2 letters as required.

Terminal results attachment:



```
/Users/ronipick/PycharmProjects/nuero1.2/venv/bin/python /Users/ronipick/PycharmProjects/nuero1.2/main.py
letter mem vs letter lamed
9/9 [==============================] - 0s 460us/step
9/9 [==============================] - 0s 407us/step
9/9 [==============================] - 0s 416us/step
9/9 [==============================] - 0s 395us/step
9/9 [==============================] - 0s 364us/step
Average accuracy: 89.06
Standard deviation: 0.01

letter mem vs letter bet
10/10 [==============================] - 0s 391us/step
10/10 [==============================] - 0s 432us/step
10/10 [==============================] - 0s 349us/step
10/10 [==============================] - 0s 430us/step
10/10 [==============================] - 0s 417us/step
Average accuracy: 87.06
Standard deviation: 0.03

letter lamed vs letter bet
9/9 [==============================] - 0s 391us/step
9/9 [==============================] - 0s 357us/step
9/9 [==============================] - 0s 436us/step
9/9 [==============================] - 0s 446us/step
9/9 [==============================] - 0s 416us/step
Average accuracy: 91.42
Standard deviation: 0.02

Process finished with exit code 0
```