Lab exercises project D

David van Balen, Daan van Laar, Joris Gool

5513588, 5518741, 4270126

1

A

Most frequent:

Validation: 14%

Testing: 14%

Naïve bayes:

Validation: 69%

Testing: 55%

Smoothing parameter: 2.00000000

Most frequents correct rate depends entirely on the amount of options and the frequency of said option, 14% could very well be achieved by having 8 options with equal chance, or 2000 options where one pops up 14% of the time. Naïve bayes will do a lot better because it also looks at the properties of said options, it doesn't relate those to eachother so the degree of error is still quite high. For instance measuring holes in numbers will achieve the same amount of holes, height and with for both 6 and 9 making it prone to errrors for those kind of cases.

B

Laplace smoothing makes sure that if a single property is particularly weird that doesn't affect the total chance too badly. This makes sure that if something is very out of the ordinary it will still look at the other properties instead of choosing something somewhat at random.

C

This option tries a large array of smoothing parameters to see which one works best and then uses the optimal smoothing parameter. this will generally lead to better results, which it also does here in comparison to a. (74 compared to 69)

D

There be 784. Log 784 is less than 784, but still accurate enough.

E

Validation data will generally include more extreme cases, whereas a test set is a more realistic representation of what the program will face in practice.

2

A

Iterations	Validating	Testing
1	63	57
2	50	57
3	55	48
4	55	54
5	56	54
6	56	54
10	56	54
50	56	54

B

From 4 iterations onward, the results have stayed the same so the weights with which the training ended were probably (almost) the same. Because this obviously changes depending on the given features and test input, I would use at least 5 iterations when good weights are required.

C

Oh no! We can't draw! :o

3

A

While 'b' looks more like the actual input of a given digit, 'a' represents the pixels that are good tells for what digit the given input is. An edge case will often differ between multiple 6's.

B

It's not as good as a human, but in our simple program we already obtained decent results so a fine tuned perceptron with good features should be able to identify digits pretty good.

4

A

It prevents overshooting and throwing away the information its learnt from the early trainingdata.

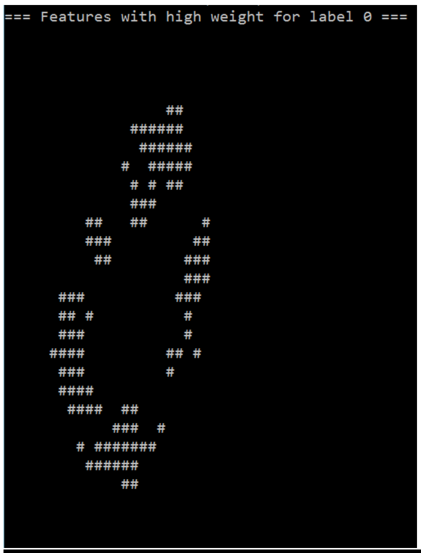
B

It runs the MIRA with multiple values for C, and tests which value performs the best.

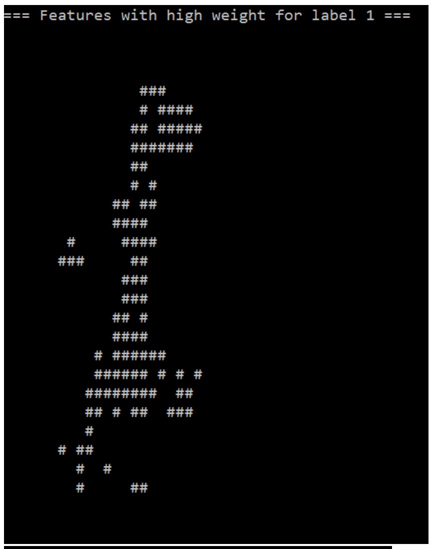
The best C-value was 0.004, with a validating score of 68 and testing score of 62.

C

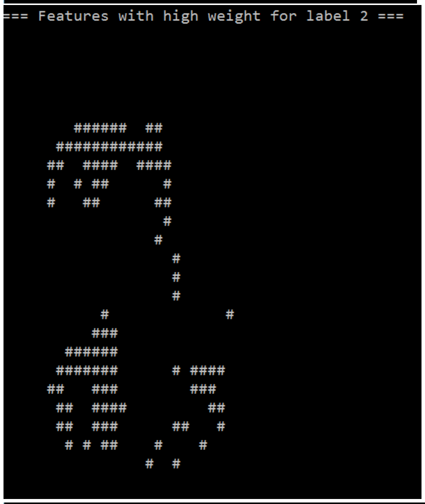
We had to modify line 502 in Dataclassifier.py to accomplish this:



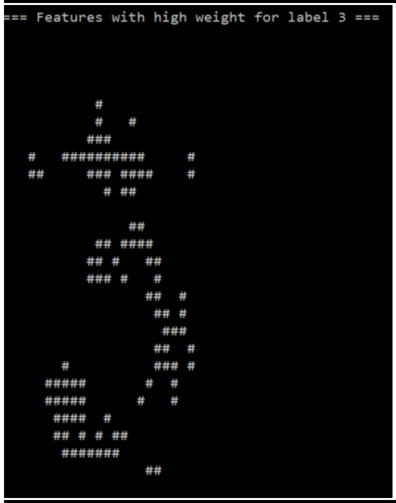
```
=== Features with high weight for label 0 ===



                  ##
              ######
               ######
             #  #####
              # # ##
              ###
         ##   ##      #
         ###         ##
          ##        ###
                    ###
      ###          ###
      ## #          #
      ###           #
     ####         ## #
      ###         #
      ####
       ####  ##
            ###  #
        # #######
         ######
             ##
```



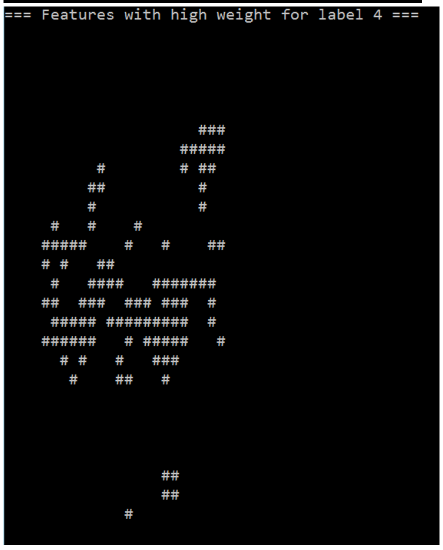
```
=== Features with high weight for label 1 ===



               ###
               # ####
              ## #####
              #######
              ##
              # #
            ## ##
            ####
       #     ####
      ###     ##
             ###
             ###
            ## #
            ####
          # ######
          ###### # # #
         ########  ##
         ## # ##   ###
         #
      # ##
        #  #
        #     ##
```



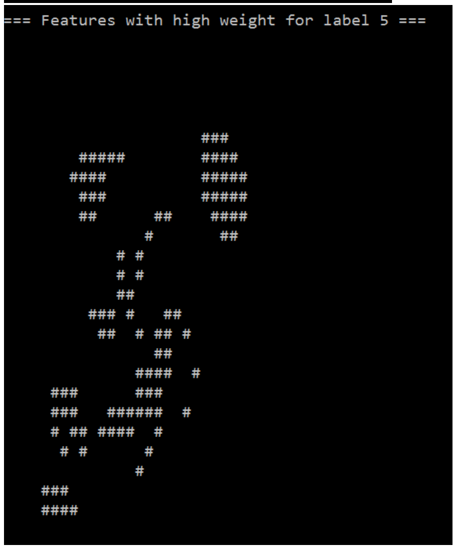
```
=== Features with high weight for label 2 ===




        ######  ##
      ############
     ##  ####  ####
     #  # ##      #
     #   ##      ##
                  #
                 #
                   #
                   #
                   #
           #             #
          ###
       ######
      #######      # ####
     ##   ###        ###
      ##  ####         ##
      ##  ###      ##   #
       # # ##     #    #
                #  #
```



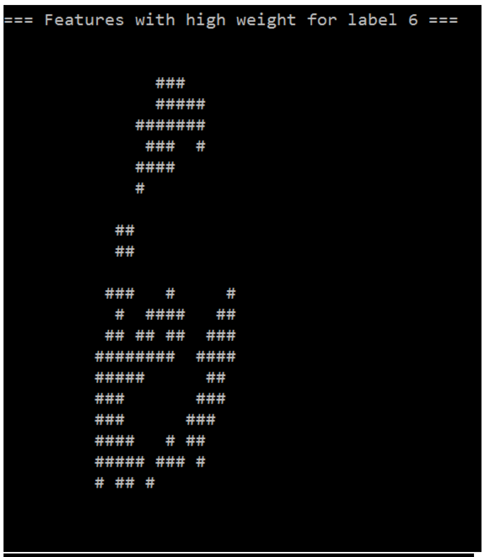
```
=== Features with high weight for label 3 ===



           #
           #   #
          ###
   #   ##########     #
   ##     ### ####    #
            # ##

               ##
           ## ####
          ## #   ##
          ### #   #
                 ##   #
                  ## #
                   ###
                  ##   #
       #          ### #
     #####       #  #
     #####      #   #
      ####  #
      ## # # ##
       #######
                 ##
```



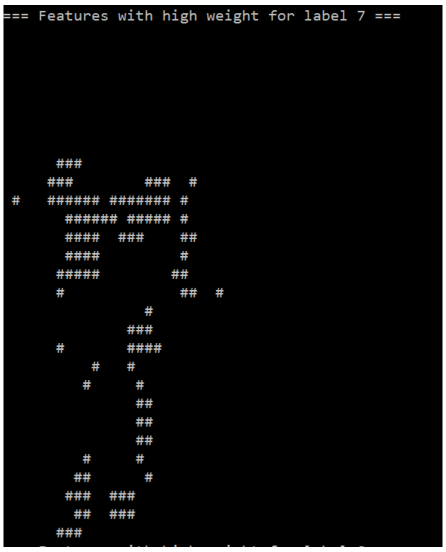
```
=== Features with high weight for label 4 ===




                     ###
                   #####
          #        # ##
         ##          #
         #           #
     #   #    #
    #####    #   #    ##
    # #   ##
     #   ####   #######
    ##  ###  ### ###  #
     ##### #########  #
    ######   # #####   #
      # #   #   ###
       #    ##   #




                 ##
                 ##
             #
```



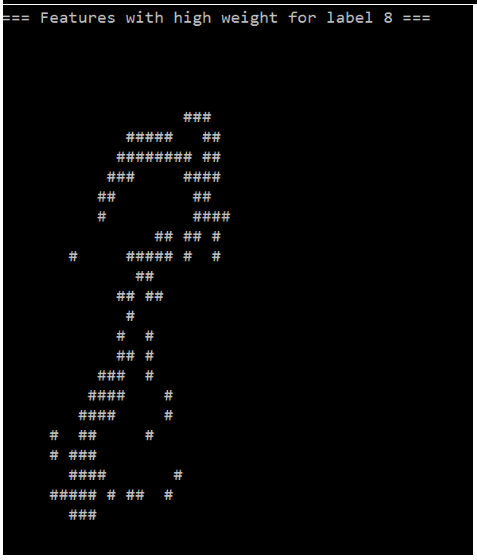
```
=== Features with high weight for label 5 ===




                     ###
        #####        ####
       ####          #####
        ###          #####
        ##      ##    ####
               #       ##
            # #
            # #
            ##
         ### #   ##
          ##  # ## #
                ##
              ####  #
     ###      ###
     ###   ######  #
     # ## ####  #
      # #      #
              #
    ###
    ####
```



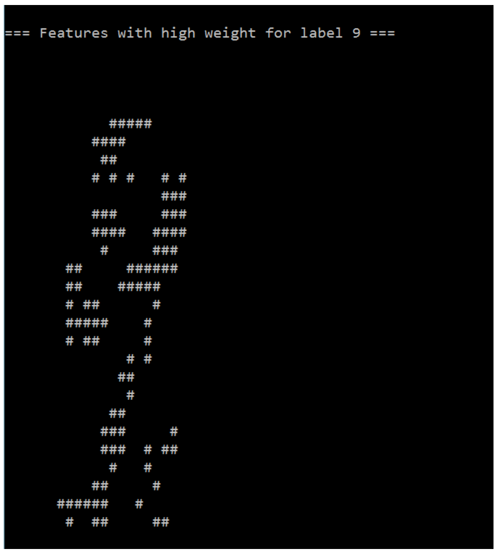
```
=== Features with high weight for label 6 ===


               ###
               #####
             #######
              ###  #
             ####
             #

           ##
           ##

          ###   #     #
           #  ####   ##
          ## ## ##   ###
         ########  ####
         #####      ##
         ###       ###
         ###      ###
         ####   # ##
         ##### ### #
         # ## #
```



```
=== Features with high weight for label 7 ===







      ###
     ###        ###  #
 #   ###### ####### #
       ###### ##### #
       ####  ###    ##
       ####         #
      #####        ##
      #             ##  #
                #
              ###
      #       ####
          #   #
         #     #
               ##
               ##
               ##
         #     #
        ##      #
       ###  ###
        ##  ###
      ###
```



```
=== Features with high weight for label 8 ===




                   ###
             #####   ##
            ######## ##
           ###     ####
          ##        ##
          #         ####
                ## ## #
       #     ##### #  #
              ##
            ## ##
             #
            #  #
            ## #
          ###  #
         ####    #
        ####     #
     #  ##     #
     # ###
       ####       #
     ##### # ##  #
       ###
```



```
=== Features with high weight for label 9 ===




            #####
          ####
           ##
          # # #   # #
                  ###
          ###     ###
          ####   ####
           #     ###
       ##     ######
       ##    #####
       # ##      #
       #####    #
       # ##     #
              # #
             ##
              #
            ##
           ###     #
           ###  # ##
            #   #
          ##     #
      ######   #
       #  ##     ##
```

5

A

Very dependent on what the goal is and how much you need to have right, and the amount of time that may be consumed. Generally I would say no because of the low accuracy.

B

This command show's which features have the highest odds ratio for label 4 over label 2, so the features label 2 is least likely to have and label 4 is most likely to have. Odds are a funny thing we humans use to indicate in how many cases of something happening a certain outcome will present itself. I would interpret the output as the places where if a pixel has a value of higher then 0 the odds of it being a 4 raise substantially.

C

Amount of holes, this one is easy, a lot of digits simply have holes or don't so generally it is a good tool to distinguish between digits, unless someone has particularly bad handwriting. The weightpoint of a digit(where you would have to hold your finger below it so it balances), this is rather unique aswell and for a lot of digits isn't in the middle, it also can't be picked up by the features we have so far(this can easily be calculated with physics formula's). the amount of horizontal blocks(for instance 4×3), this indicates whether a digit has a large straight horizontal part, like a 7, 5 or 4, implementation trivial. Same for vertical block.

D

Just the holes, this was enough to up percentage to 84%, although it does come with a significant performance decrease. It is a, for the extra time it takes, small impact on the end result. Although, again, if accuracy is more important thapyn speed it is still an improvement. The reason for this is as described above that it is a very good tool to distinguish between digits with and without holes.

6

A

88% and 84%

B

StopAgent:

- Did the agent move?

FoodAgent:

- Did the agent either eat a food, or come closer to one?

SuicideAgent:

- Did the agent lose with this move?
- Did the agent come closer to a ghost with this move?

ContestAgent

- Did the agent win with this move?
- Did the agent lose with this move?
- How much did the score change with this move?

C

We implemented exactly the 7 above features, plus the two things for the food feature that we also implemented separately (for a total of 9 features). Aside from the score change, all of the implemented features are Booleans which we implemented as -1 or +1. We did this, because when implementing them as 0 or 1 makes it hard for the perceptron to learn the positive effects of it being 0, because multiplying any weight by 0 obviously results in 0.

The agent aren't included in the perceptron, the perceptron is used to attempt to copy the agents' behaviour. Thus, the agents cannot have any effect on the performance of the classifier. This means that the last question is unanswerable.

7

Q1	Q2	Q3	Q4	Q5	Q6	TOTAL
4/4	1/1	6/6	6/6	4/4	4/4	25/25