**Assignment 3 – Report**

Group members: Caiwei He(caiweih2), Chengrui Zhu(cz8), Kelong Wu(kwu18)

**Part 1**

1. According to the assignment requirement, we implemented part 1-1, including setups for images and labels, calculating (Pij | class) and Pclass, smoothing by adding constant (from 0.1 to 10) to the numerator and constant \* 2 to the denominator, and testing using MAP which calculates the largest (log Pclass + log P(f1,1 | class) + log P(f1,2 | class) + ... + log P(f28,28 | class)).

We tried different smoothing constants, and found out that in all, the overall performance decreased with the increase of the constant value, that is, the stronger the smoothing, the worse the performance. It makes sense because the stronger the smoothing, the more distortion the figure has. So we chose 0.1 as the constant value.

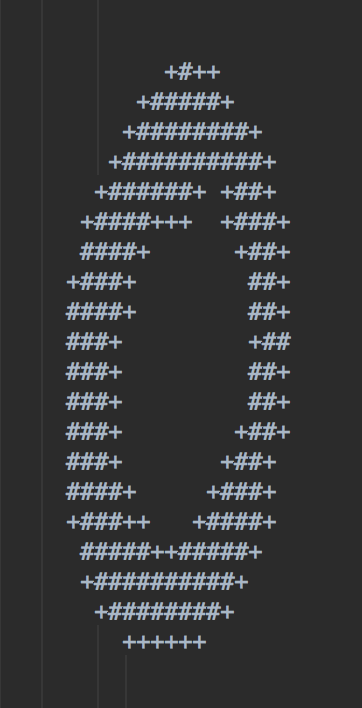
Classification rate:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0.93 | 0.87 | 0.84 | 0.71 | 0.75 | 0.7 | 0.80 | 0.86 | 0.76 | 0.60 |

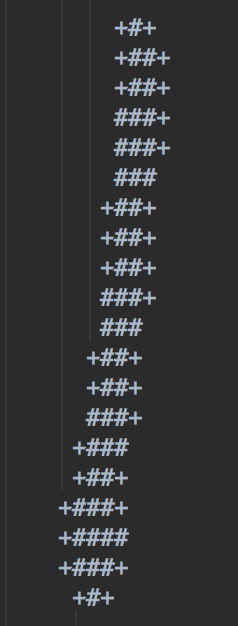
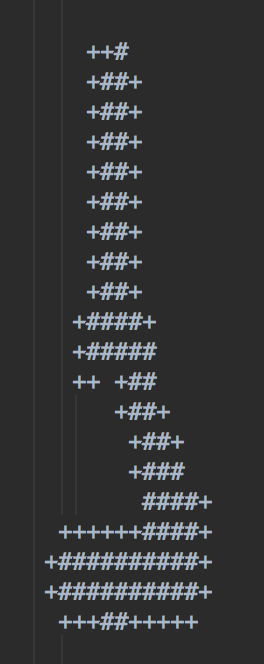
Confusion matrix:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | 0.93 | 0 | 0.010 | 0 | 0 | 0.022 | 0.011 | 0 | 0.012 | 0.007 |
| 1 | 0 | 0.87 | 0.031 | 0.009 | 0 | 0.011 | 0.046 | 0.056 | 0.012 | 0.007 |
| 2 | 0.012 | 0.008 | 0.84 | 0 | 0.009 | 0.011 | 0.046 | 0.044 | 0.037 | 0 |
| 3 | 0 | 0 | 0.042 | 0.71 | 0 | 0.133 | 0 | 0 | 0.017 | 0.022 |
| 4 | 0.012 | 0 | 0.021 | 0 | 0.75 | 0.033 | 0.046 | 0.033 | 0.037 | 0.075 |
| 5 | 0.061 | 0.017 | 0 | 0.027 | 0.009 | 0.7 | 0.069 | 0 | 0.098 | 0.015 |
| 6 | 0.037 | 0.008 | 0.063 | 0.018 | 0.038 | 0.011 | 0.80 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0.010 | 0.062 | 0.009 | 0.011 | 0 | 0.86 | 0.012 | 0.015 |
| 8 | 0.049 | 0 | 0.052 | 0.009 | 0.019 | 0.022 | 0.023 | 0.033 | 0.76 | 0.007 |
| 9 | 0 | 0 | 0 | 0.053 | 0.170 | 0.067 | 0 | 0.156 | 0.122 | 0.60 |

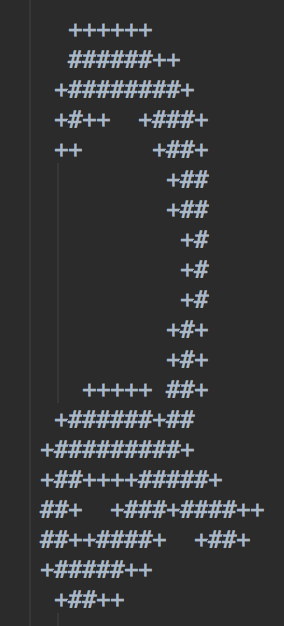
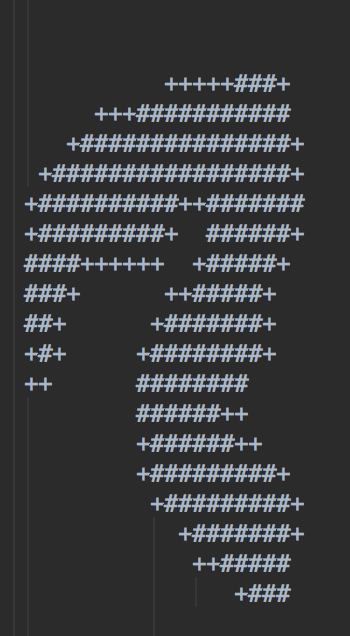
Highest and lowest posterior probabilities in test examples:

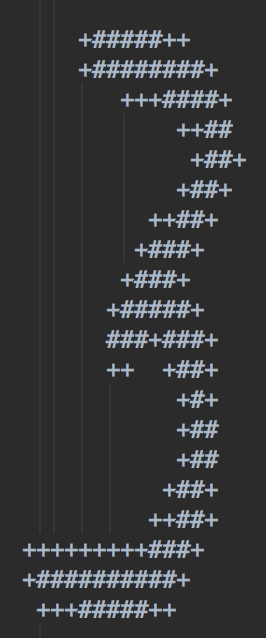
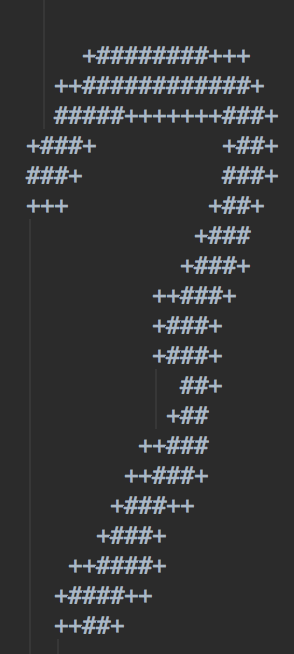
No.723 highest 0 No.610 lowest 0

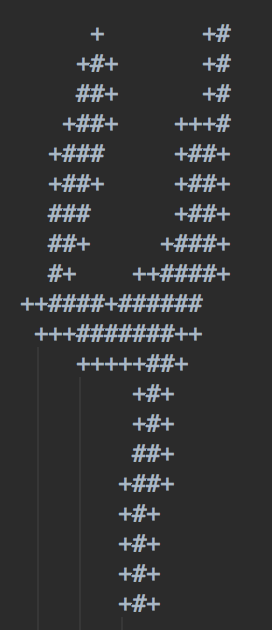
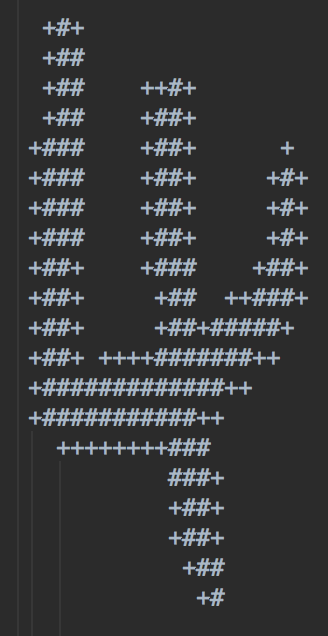
No.633 highest 1 No.527 lowest 1

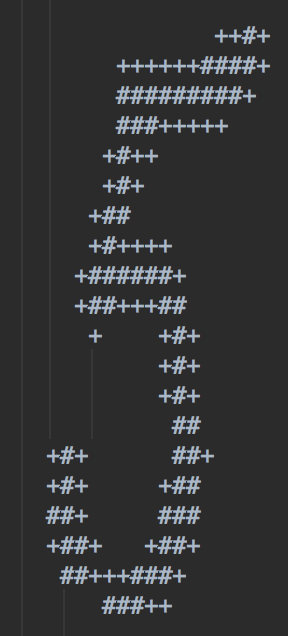
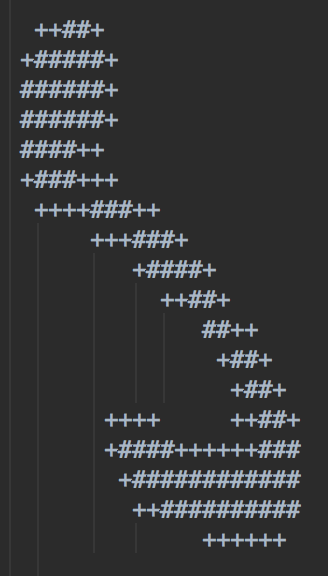
No.795 highest 2 No.790 lowest 2

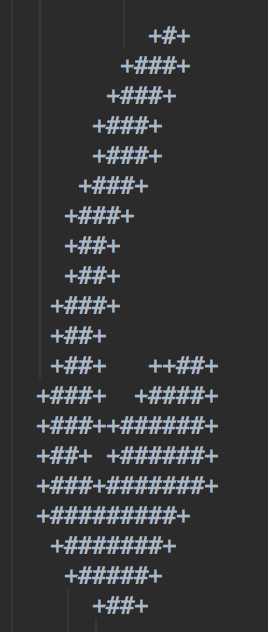
No.205 highest 3 No.681 lowest 3

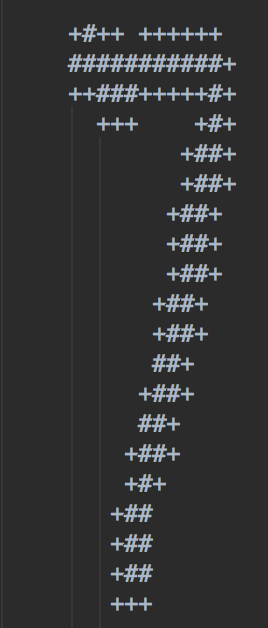
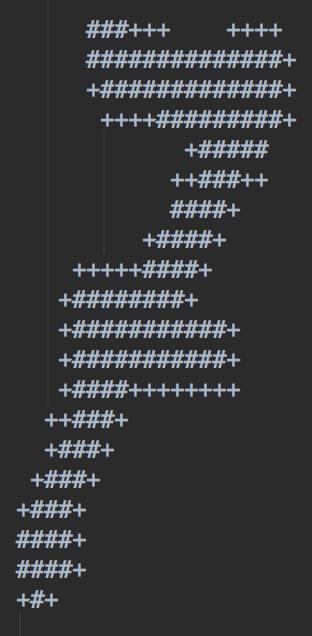
No.111 highest 4 No.253 lowest 4

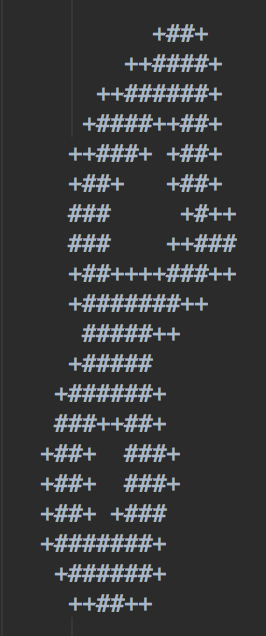
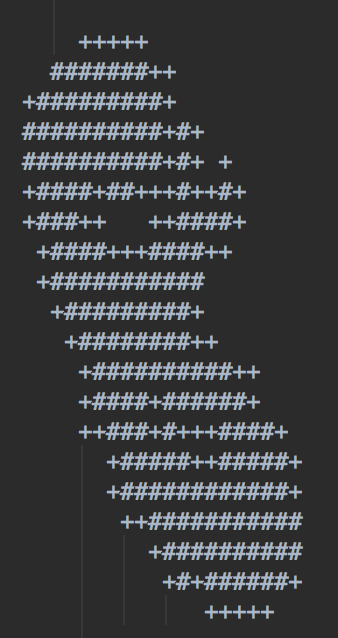
No.471 highest 5 No.737 lowest 5

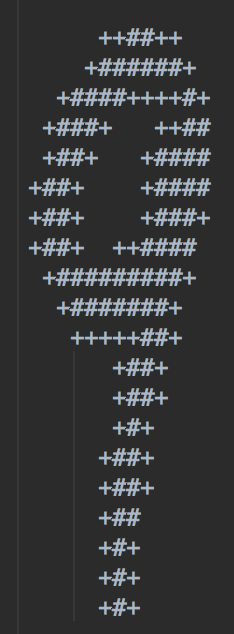
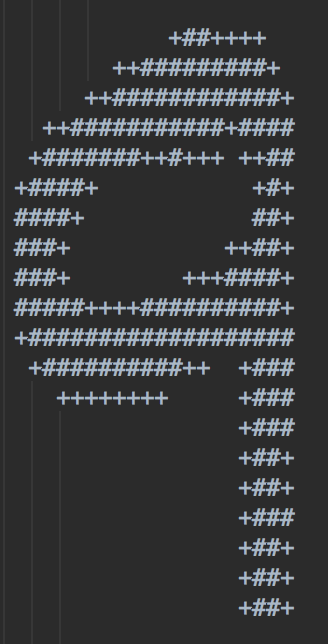
No.632 highest 6 No.444 lowest 6

No.784 highest 7 No.119 lowest 7

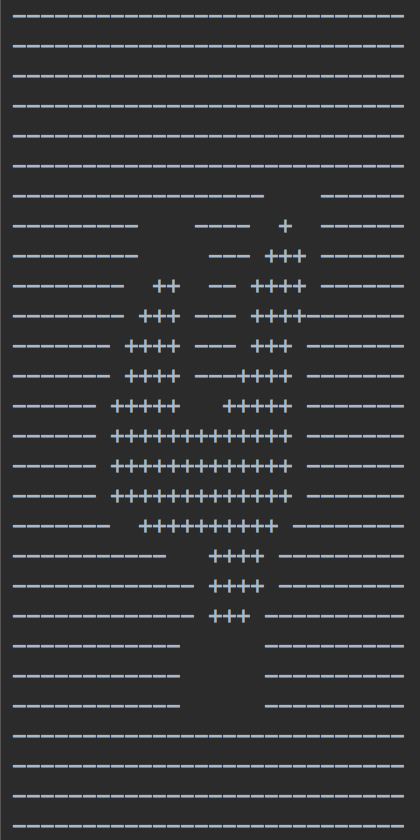
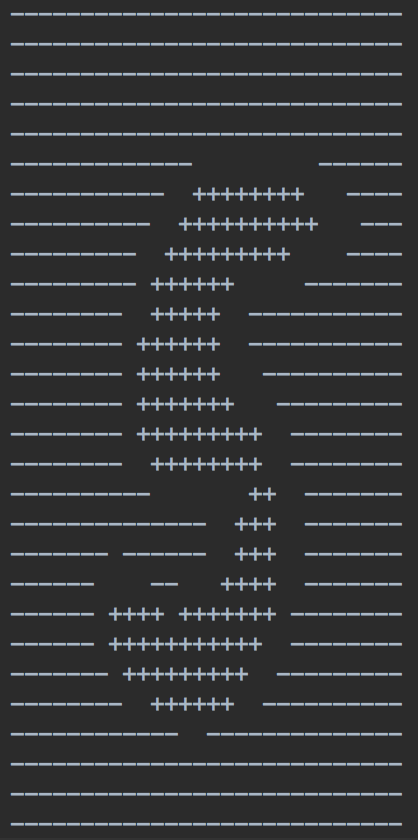
 

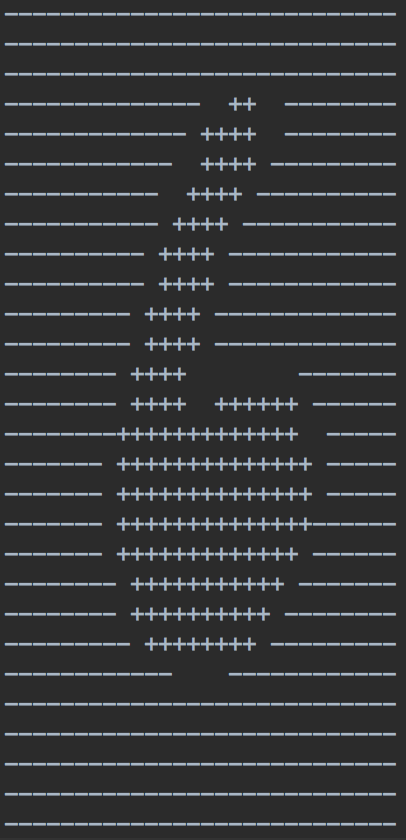
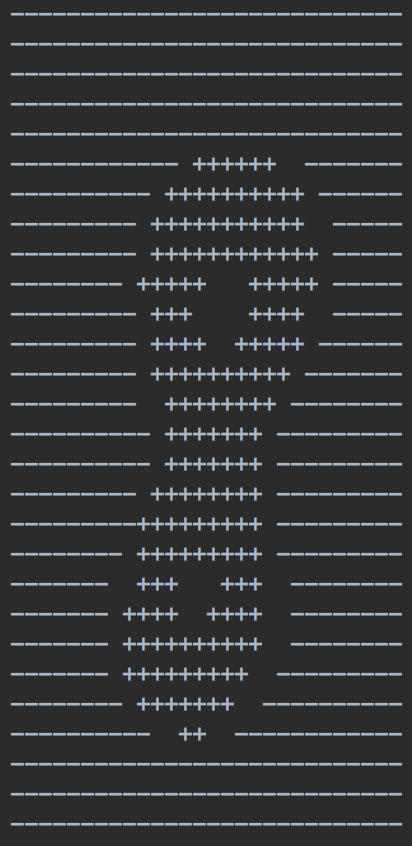
No.560 highest 8 No.101 lowest 8

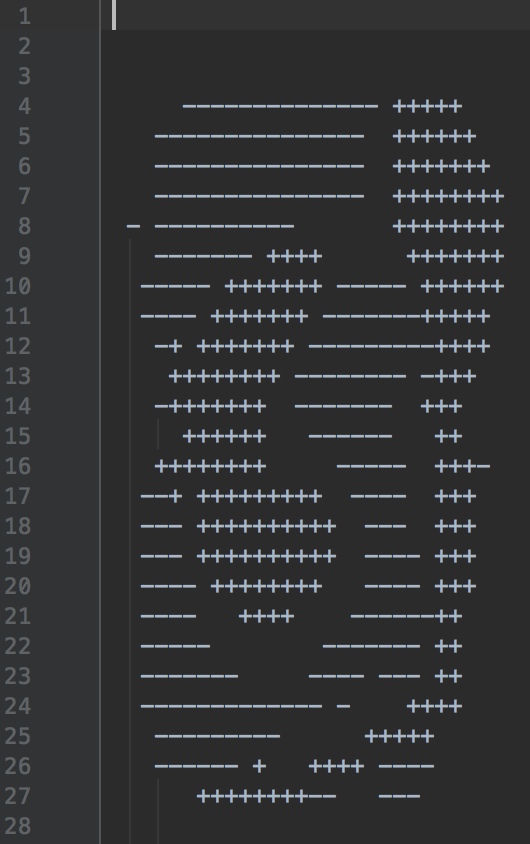
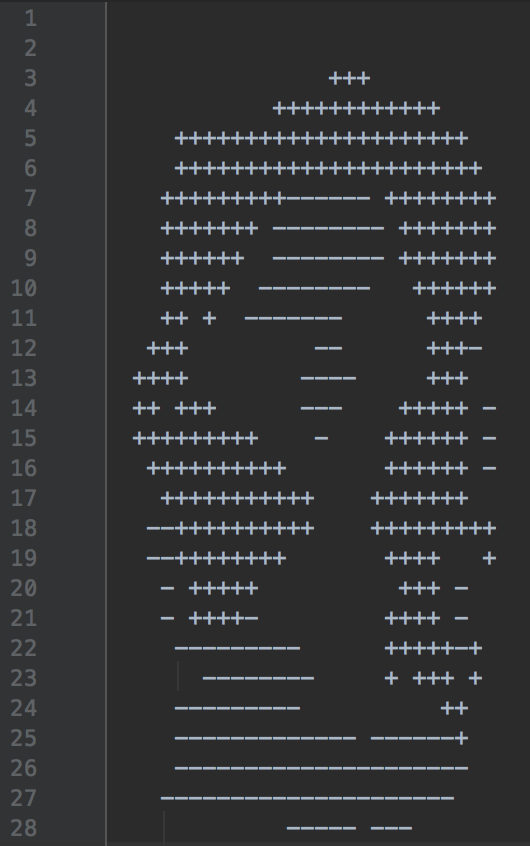
No.745 highest 9 No.801 lowest 9

According to the confusion matrix, the four pairs are (5, 3) (4, 9) (7, 9) (8, 9). For each digit, we used ‘+’ to denote features with probabilities larger than 0.5, ‘ ’ for features with probabilities between 0.3 and 0.5, and ‘-’ for features with probabilities below 0.3. And for odds ratio, we used '+' to denote features with log odds larger than 0.2, ' ' for features with log odds between +-0.2, and '-' for features with log odds below -0.2.

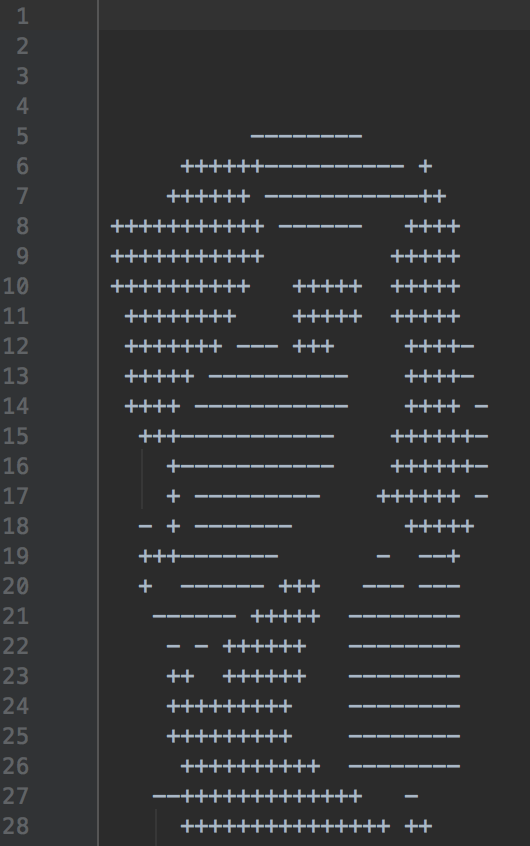
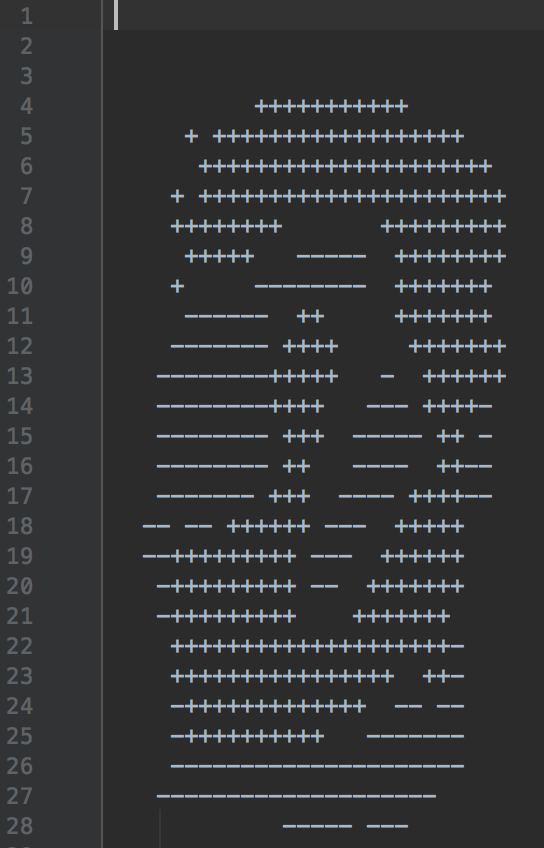
  

3-9

odds ratio 5/3 odds ratio 4/9

odds ratio 7/9 odds ratio 8/9

**Part 2**

1.

(1) **Implementation**

In this part, we use Naïve Bayes classifier to solve this basic binary classification problem. Based on MAP, we have the equation,

Thus, to maximize p(class|W), we need to first estimate the prior distribution p(class) and these binary variables distribution using ML estimation.

To estimate, we need to implement Laplace smoothing method to avoid 0 appearing. Here we set Laplace smoothing constant as 0.5 and 1. Since this is a binary problem, ‘%’ and ‘ ’ take half and half. These constants make sense. Then we have

and

To test the result, we compute p(yes|W) and p(no|W) seperately and choose the bigger one.

The complete code is in ‘2.1/classification.py’.

(2) **Result**

confusion matrix

|  |  |  |
| --- | --- | --- |
|  | yes | no |
| yes | 0.98 | 0.02 |
| no | 0.06 | 0.94 |

2.

(1) **Implementation**

In this part, we need to classifier 5 digits. It is similar to 2.1 when we use Naïve Bayes classification based on Bernoulli distribution. The only difference is that there are 5 classes in 2.2. Thus, we still have

Since the prior distribution is an average distribution, we can overlook it when we compute log likelihood.

Since it is still a binary problem, we set Laplace smoothing constant as 0.5 and 1. Thus,

and

To test the result, we compute five p(class|W) seperately and choose the largest one.

The complete code is in ‘2.2/digit.py’.

(2) **Result**

Confusion Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 0.875 | 0 | 0 | 0 | 0.125 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0.375 | 0 | 0.625 | 0 |
| 5 | 0.125 | 0 | 0.125 | 0 | 0.75 |

Total accuracy is 85%.