MP3 Report

Course: ECE448-LE1

Team: Yuhang Chen

Jiakai Lin

Wenbo Ye

Date: Apr. 24th, 2023

**Section I: Image Classification**

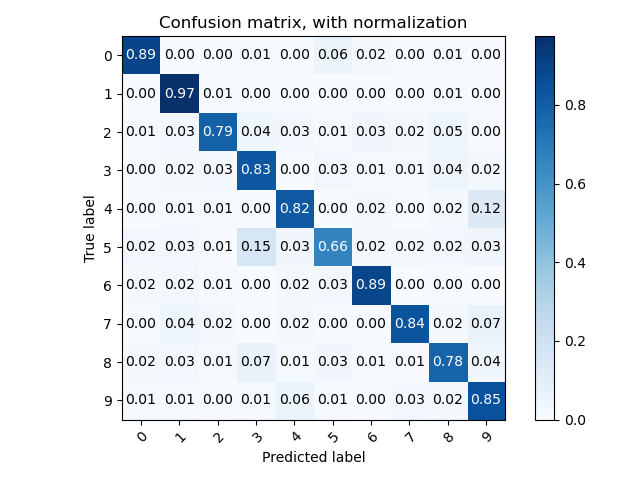
The average classification differs for different values of k.

For k = 1.0, it is 0.836.

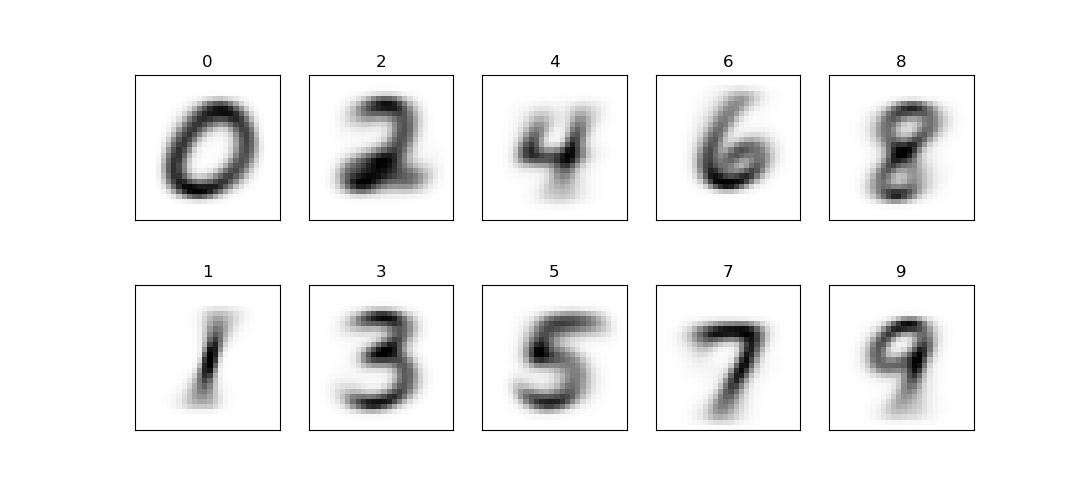
The classification rate for each class:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0.89 | 0.97 | 0.79 | 0.83 | 0.82 | 0.66 | 0.89 | 0.84 | 0.78 | 0.85 |

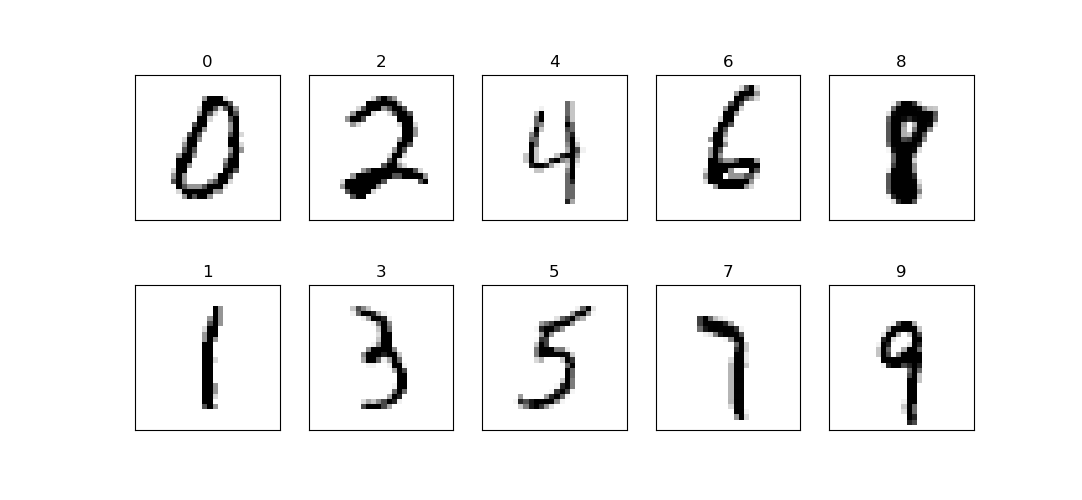
The confusion matrix is:

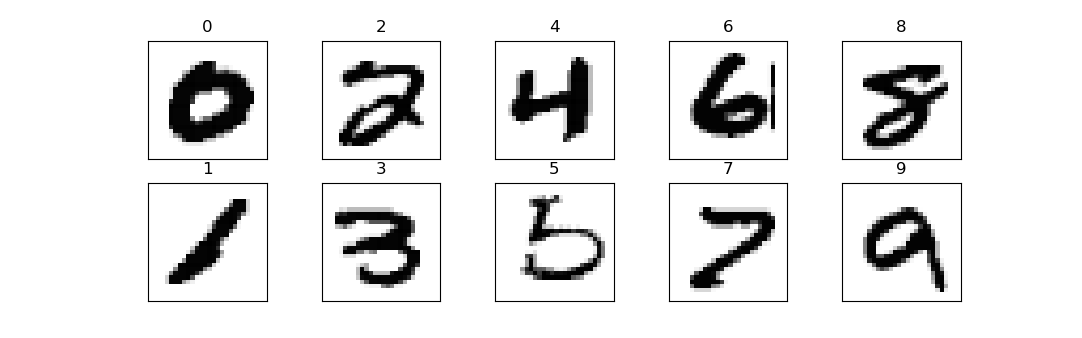


For k = 1, we get the figures after training:



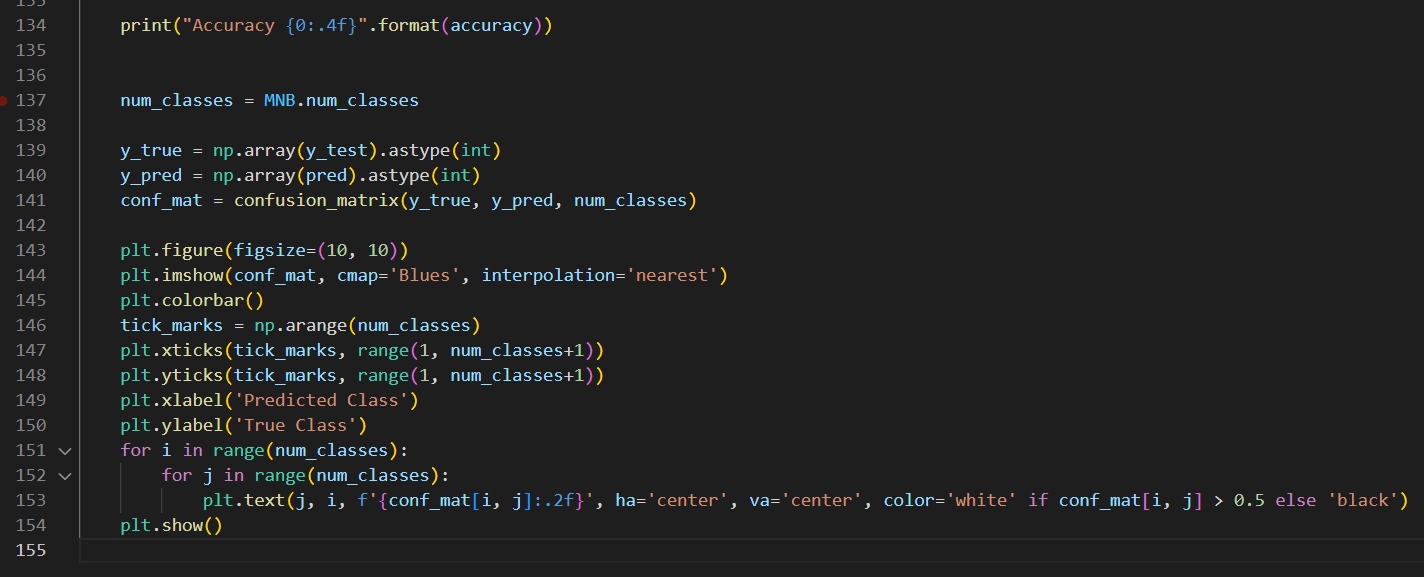
Test examples which have the highest posterior probabilities:



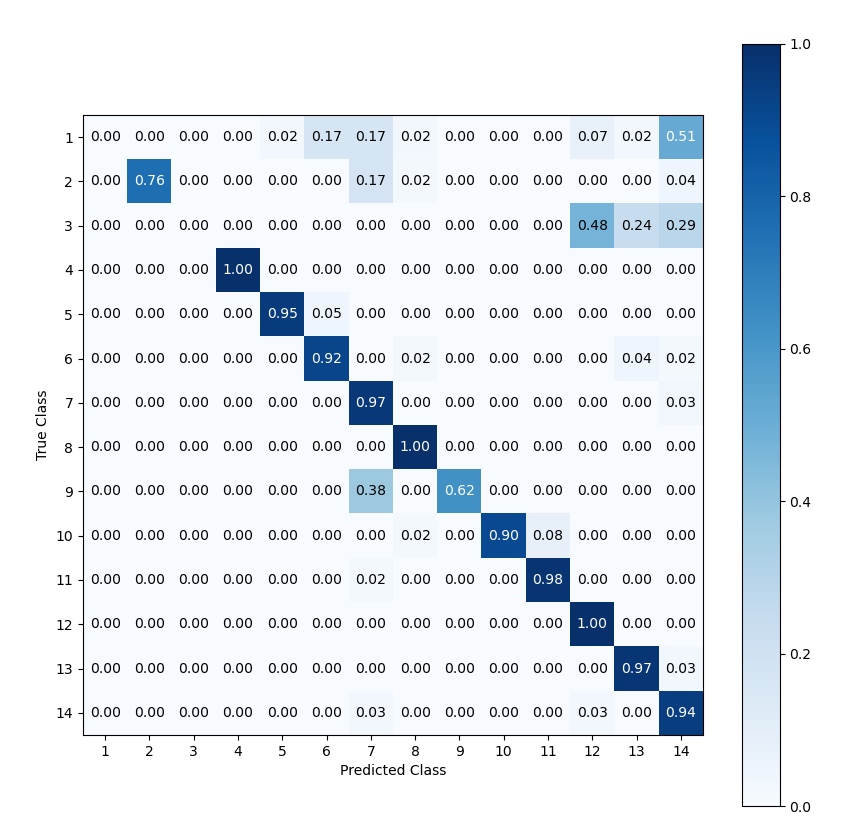
Test examples which have the lowest posterior probabilities: 

**Section II:** **Text Classification**

1. **Confusion matrix**. This is a 14x14 matrix whose entry in row r and column c is the percentage of test text from class r that are classified as class c.

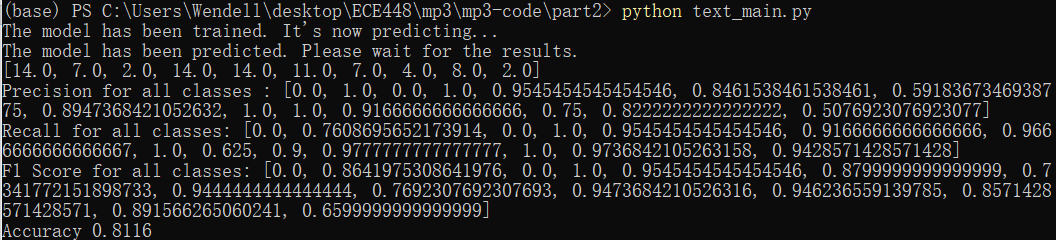


I added some codes at the end of the *text\_main.py* file to generate this diagram.

****

1. **Accuracy, recall, and F1 scores** for each of the classes on the development set.

The output of my program (including the class prior) is shown below.

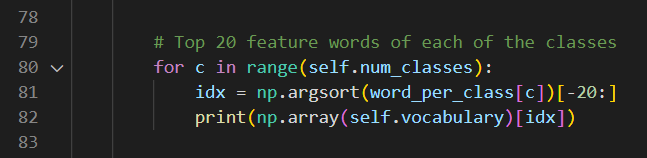
****

**Accuracy: 0.8116**

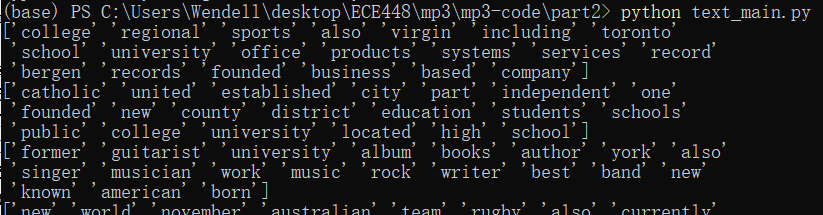
**Recall** for all classes: [0.0, 0.7608695652173914, 0.0, 1.0, 0.9545454545454546, 0.9166666666666666, 0.9666666666666667, 1.0, 0.625, 0.9, 0.9777777777777777, 1.0, 0.9736842105263158, 0.9428571428571428]

**F1 Scores** for all classes: [0.0, 0.8641975308641976, 0.0, 1.0, 0.9545454545454546, 0.8799999999999999, 0.7341772151898733, 0.9444444444444444, 0.7692307692307693, 0.9473684210526316, 0.946236559139785, 0.8571428571428571, 0.891566265060241, 0.6599999999999999]

1. **Top 20 feature words** of each of the classes

****

I added codes at the *fit* function of the *TextClassifier.py* file to generate the results.

****

According to the *classes.txt* file, these top 20 feature words and their corresponding classes are:

**Company:** ['college' 'regional' 'sports' 'also' 'virgin' 'including' 'toronto' 'school' 'university' 'office' 'products' 'systems' 'services' 'record' 'bergen' 'records' 'founded' 'business' 'based' 'company']

**Educational Institution:** ['catholic' 'united' 'established' 'city' 'part' 'independent' 'one' 'founded' 'new' 'county' 'district' 'education' 'students' 'schools' 'public' 'college' 'university' 'located' 'high' 'school']

**Artist:** ['former' 'guitarist' 'university' 'album' 'books' 'author' 'york' 'also' 'singer' 'musician' 'work' 'music' 'rock' 'writer' 'best' 'band' 'new' 'known' 'american' 'born']

**Athlete:** ['new' 'world' 'november' 'australian' 'team' 'rugby' 'also' 'currently' 'hockey' 'american' 'national' 'former' 'footballer' 'professional' 'plays' 'player' 'league' 'played' 'football' 'born']

**Office Holder:** ['national' 'representing' 'american' 'united' 'elected' 'republican' 'representatives' 'since' 'county' 'former' 'served' 'party' 'democratic' 'senate' 'house' 'state' 'politician' 'district' 'member' 'born']

**Mean Of Transportation:** ['company' 'ii' 'american' 'commissioned' 'royal' 'named' 'designed' 'first' 'service' 'launched' 'states' 'world' 'class' 'aircraft' 'united' 'uss' 'ship' 'war' 'built' 'navy']

**Building:** ['added' 'hospital' 'designed' 'states' 'also' 'museum' 'known' 'united' 'street' 'county' 'listed' 'places' 'register' 'national' 'building' 'church' 'located' 'built' 'house' 'historic']

**Natural Place:** ['pass' 'flows' 'ft' 'state' 'east' 'crater' 'creek' 'lies' 'west' 'range' 'tributary' 'near' 'county' 'north' 'km' 'south' 'located' 'mountain' 'lake' 'river']

**Village:** ['southern' 'road' 'township' 'zone' '2010' '1991' 'within' 'people' 'km' 'county' 'state' 'india' 'nepal' 'municipality' 'census' 'located' 'province' 'population' 'district' 'village']

**Animal:** ['moist' 'snails' 'natural' 'forests' 'habitat' 'subtropical' 'endemic' 'mollusk' 'snail' 'tropical' 'described' 'marine' 'known' 'sea' 'gastropod' 'moth' 'genus' 'found' 'species' 'family']

**Plant:** ['perennial' 'bulbophyllum' 'south' 'orchid' 'name' 'grows' 'tree' 'habitat' 'leaves' 'plants' 'common' 'found' 'known' 'flowering' 'endemic' 'native' 'genus' 'plant' 'family' 'species']

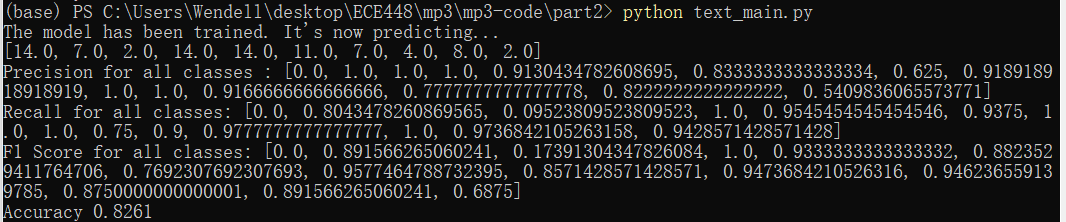
**Album:** ['ep' 'new' 'albums' 'label' 'tracks' 'debut' 'live' 'rock' 'recorded' 'release' 'second' 'music' 'songs' 'american' 'studio' 'first' 'records' 'band' 'released' 'album']

**Film:** ['documentary' 'name' 'novel' 'roles' 'movie' 'first' 'silent' 'films' 'produced' 'also' 'comedy' 'drama' 'based' 'written' 'released' 'stars' 'american' 'starring' 'directed' 'film']

**Written Work:** ['life' 'publication' 'science' 'also' 'peerreviewed' 'books' 'fiction' 'magazine' 'new' 'author' 'american' 'story' 'newspaper' 'series' 'written' 'journal' 'first' 'novel' 'book' 'published']

1. Calculate your accuracy **without including the class prior** into the Naive Bayes equation i.e. Only computing the ML inference of each instance. Report the change in accuracy numbers, if any. Also **state your reasoning** for this observation. Is including **the class prior always beneficial**? Change your class prior to a **uniform distribution**. What is the change in result?

If I ignored the class prior, the output showed as follows.

****

**Accuracy: 0.8261**

The new accuracy was **better** than the previous one.

I think there would be two reasons to account for it. **First**, the values of the prior probabilities are inaccurate, or the distribution of the training set is not reasonable, causing our method to tend to classify texts into certain categories. **Second**, the distribution of the training set and the test set is inconsistent, which means that adding a priori probability may make the algorithm more biased towards the distribution of the training set, resulting in poor classification performance of the test set.

Therefore, including the class prior is **NOT always beneficial**.

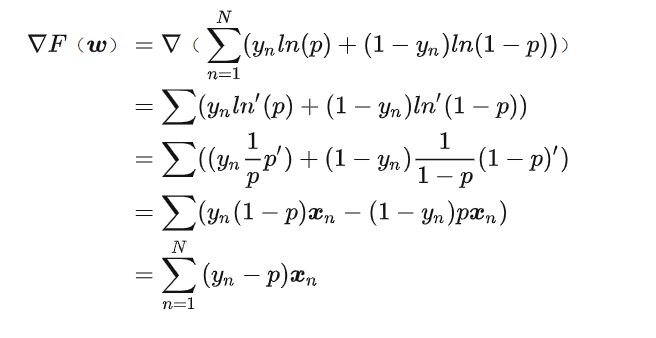
Similarly, if I change the class prior to a uniform distribution, the result will also be 0.8261, as if I simply ignore the prior.

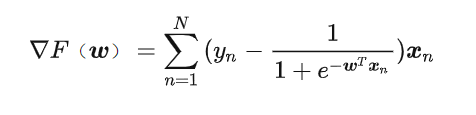
**Section III:**

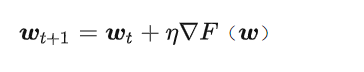
In part 3, we use sigmoid function as the logistic regression function. And we design the Loss function L(w) as the

where,

And we use Gradient Descent Method(GD) to get the parameter w by updating the w in iteration. The gradient is defined as:



Substituting sigmoid function to the above equation, we get:

****By using = 1/iteration\_num as the learning rate, we get the updated w by the iteration:

The testing results are shown in the following figures:

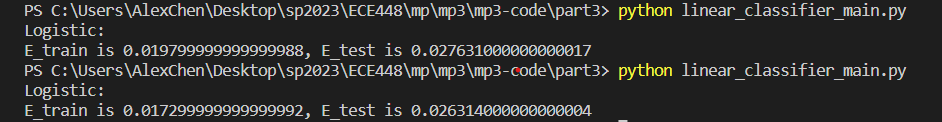
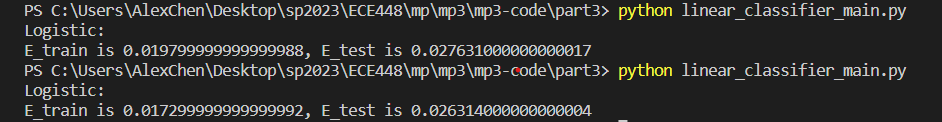


Figure 3.1. The error of Train and Test of Logistic Regression in Test1&Test2

****

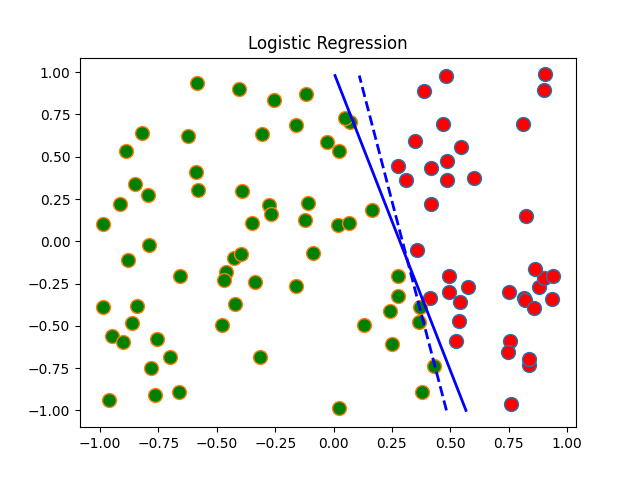


Figure 3.1. The Running Result Plot of Logistic Regression Test 1

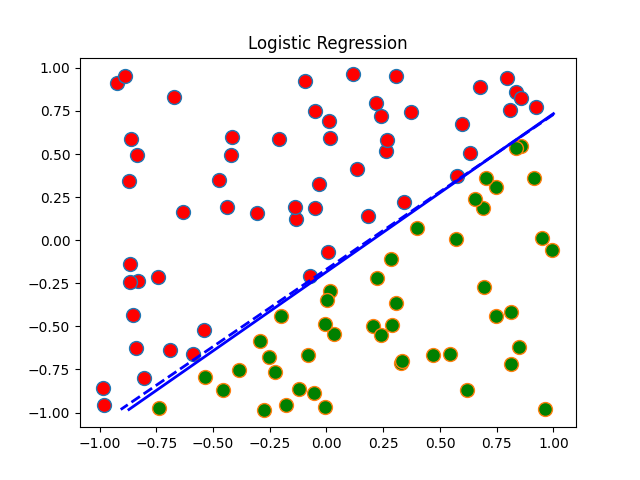


Figure 3.2. The Running Result Plot of Logistic Regression Test 2

**Extra Credit:**

If you have done any work which you think should get extra credit, describe it here

**Statement of Contribution:**

Jiaka mainly implement the algorithm and do the programing for part 1. Jiakai’ code is debugged by himself and tested well, so his code for part 1 is submitted.

The report of Section I is written by Jiakai.

Wenbo mainly implement the algorithm and do the programing for part 2. Jiakai’ code is debugged by himself and tested well, so his code for part 2 is submitted.

The report of Section II is written by Jiakai.

Yuhang mainly implement the algorithm and do the programing for part 3. Yuhang’ code is debugged by himself and tested well, so his code for part 3 is submitted.

The report of Section III is written by Yuhang.

The extra credit part was implemented by all of us.