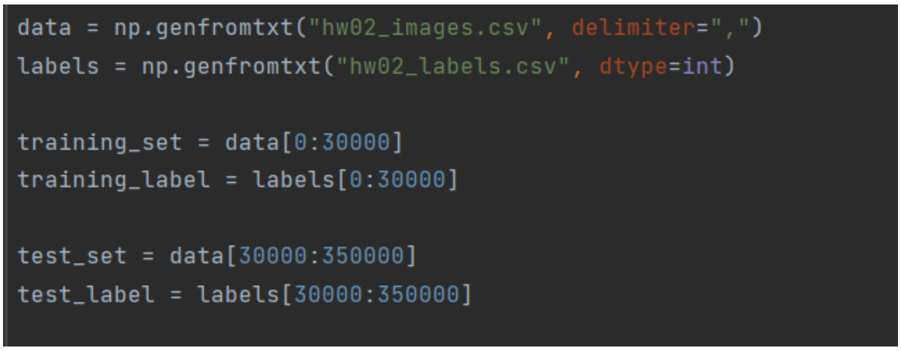
Eren Barış Bostancı
68770

ENGR 421 – HW2

Data Importing:

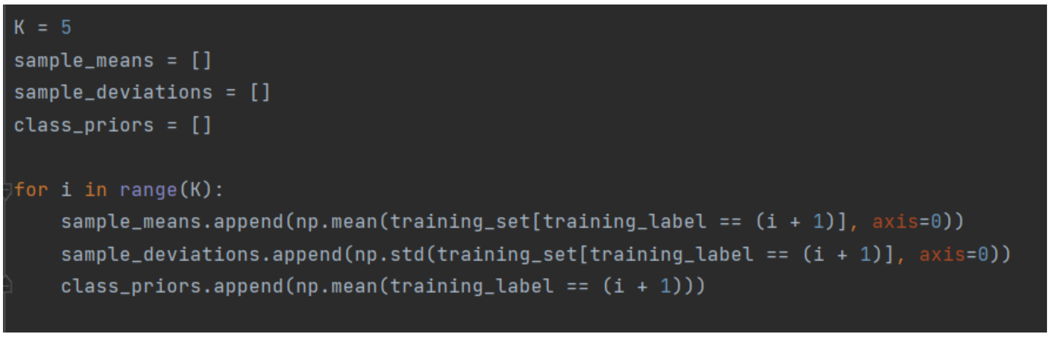
First, I used np.genfromtxt method from numpy library to import the given csv files into arrays. For training set I take the first 30000 entry and for testing set I take the last 5000 entry.



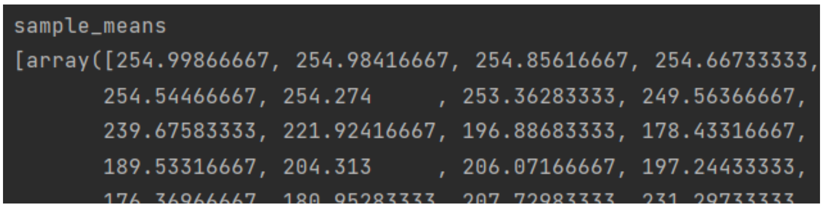
```
data = np.genfromtxt("hw02_images.csv", delimiter=",")
labels = np.genfromtxt("hw02_labels.csv", dtype=int)

training_set = data[0:30000]
training_label = labels[0:30000]

test_set = data[30000:350000]
test_label = labels[30000:350000]
```

Parameter Estimation:

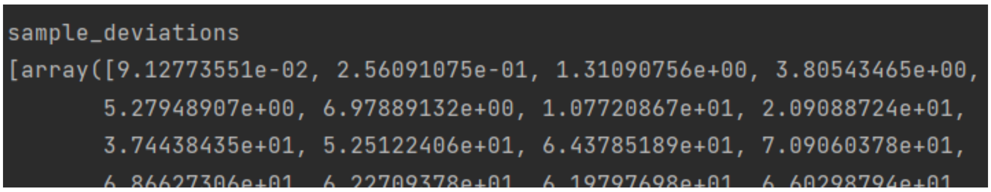
I estimated the parameters, which are means, standard deviation and priors for all the three classes.



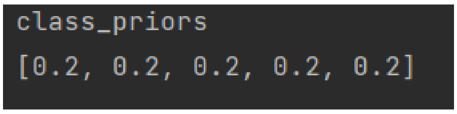
```
K = 5
sample_means = []
sample_deviations = []
class_priors = []

for i in range(K):
    sample_means.append(np.mean(training_set[training_label == (i + 1)], axis=0))
    sample_deviations.append(np.std(training_set[training_label == (i + 1)], axis=0))
    class_priors.append(np.mean(training_label == (i + 1)))
```



```
sample_means
[array([254.99866667, 254.98416667, 254.85616667, 254.66733333,
        254.54466667, 254.274     , 253.36283333, 249.56366667,
        239.67583333, 221.92416667, 196.88683333, 178.43316667,
        189.53316667, 204.313     , 206.07166667, 197.24433333,
        176.36966667, 180.95283333, 207.72983333, 231.29733333
```



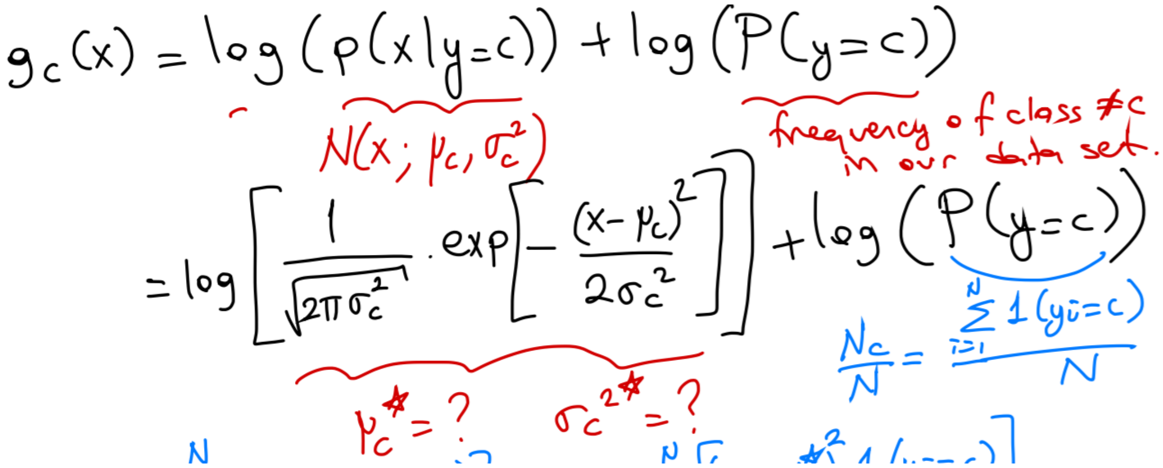
```
sample_deviations
[array([9.12773551e-02, 2.56091075e-01, 1.31090756e+00, 3.80543465e+00,
        5.27948907e+00, 6.97889132e+00, 1.07720867e+01, 2.09088724e+01,
        3.74438435e+01, 5.25122406e+01, 6.43785189e+01, 7.09060378e+01,
        6.86627306e+01, 6.22709378e+01, 6.19797698e+01, 6.60298794e+01
```



```
class_priors
[0.2, 0.2, 0.2, 0.2, 0.2]
```

Confusion Matrix

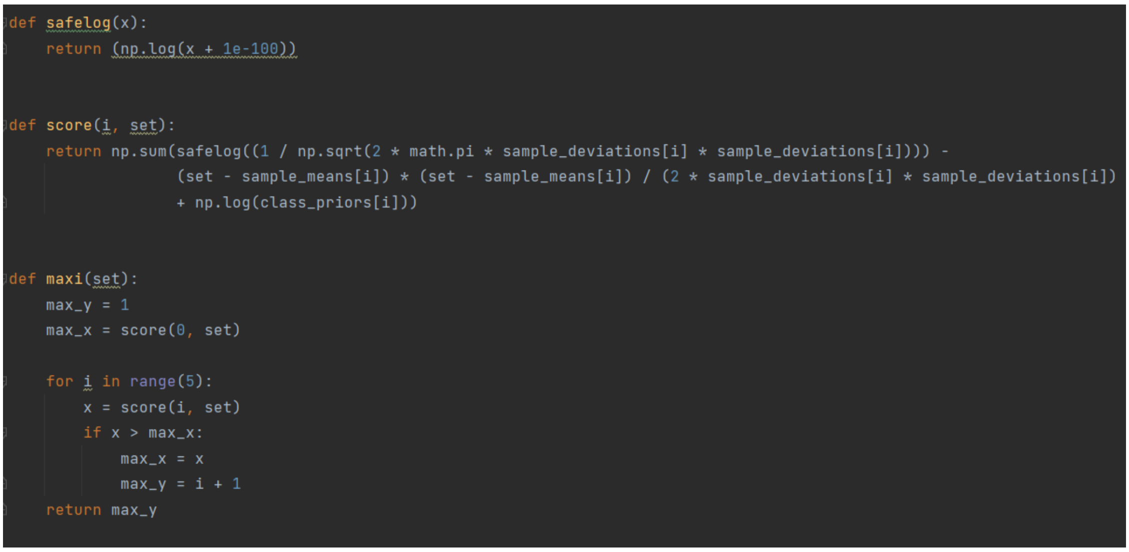
After calculating sample means, standard deviation and priors, I used them to calculate the score values. I used the Parametric Classification rule to calculate the confusion matrixes as we did in lecture.



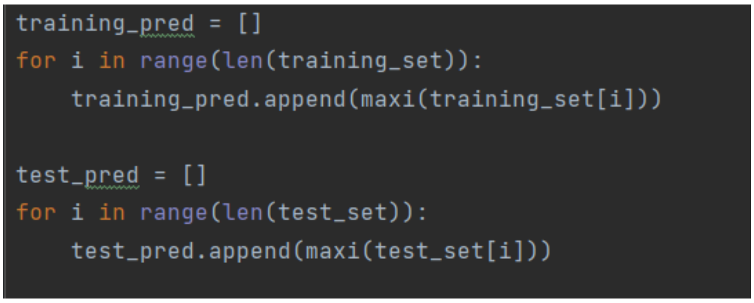
$$g_c(x) = \log(\underbrace{p(x|y=c)}_{\mathcal{N}(x;\,\mu_c,\,\sigma_c^2)}) + \log(\underbrace{P(y=c)}_{\text{frequency of class \#c in our data set.}})$$

$$= \log\underbrace{\left[\frac{1}{\sqrt{2\pi\sigma_c^2}}\cdot\exp\left[-\frac{(x-\mu_c)^2}{2\sigma_c^2}\right]\right]}_{\mu_c^{*} = ? \qquad \sigma_c^{2*} = ?} + \log(\underbrace{P(y=c)}_{\frac{N_c}{N} = \frac{\sum_{i=1}^{N} 1(y_i=c)}{N}})$$

I write score function according to this equation and used maxi method to prepare the prediction results. I separate the exp part in the log and make simplification.

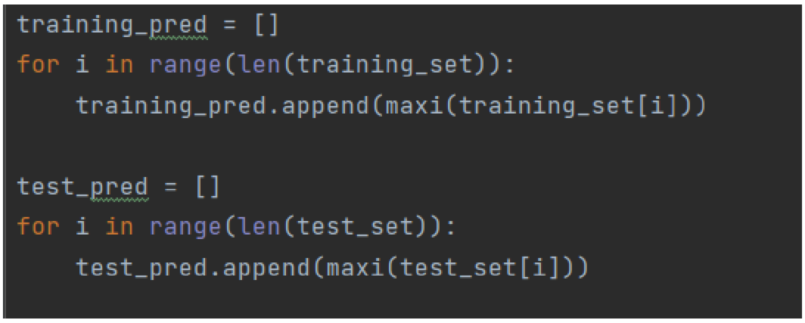


```
def safelog(x):
    return (np.log(x + 1e-100))


def score(i, set):
    return np.sum(safelog((1 / np.sqrt(2 * math.pi * sample_deviations[i] * sample_deviations[i]))) -
                  (set - sample_means[i]) * (set - sample_means[i]) / (2 * sample_deviations[i] * sample_deviations[i])
                  + np.log(class_priors[i]))


def maxi(set):
    max_y = 1
    max_x = score(0, set)

    for i in range(5):
        x = score(i, set)
        if x > max_x:
            max_x = x
            max_y = i + 1
    return max_y
```



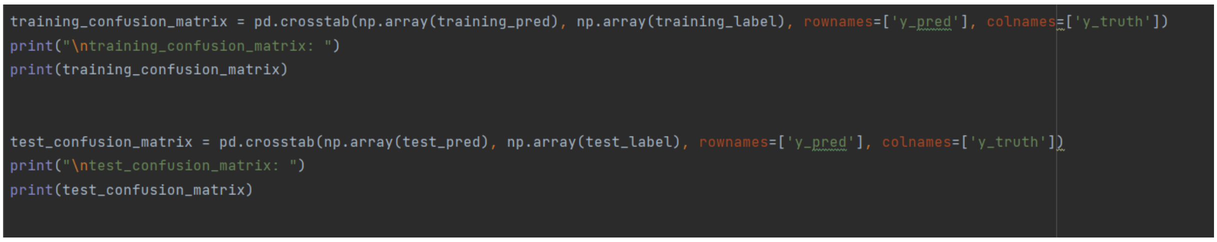
```
training_pred = []
for i in range(len(training_set)):
    training_pred.append(maxi(training_set[i]))

test_pred = []
for i in range(len(test_set)):
    test_pred.append(maxi(test_set[i]))
```

For prepare the confusion matrixes, I calculate the score of each entry and take the max score to predict the class.

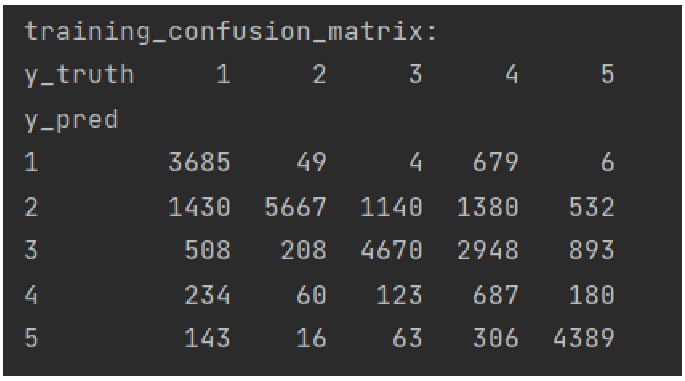


```
training_pred = []
for i in range(len(training_set)):
    training_pred.append(maxi(training_set[i]))

test_pred = []
for i in range(len(test_set)):
    test_pred.append(maxi(test_set[i]))
```

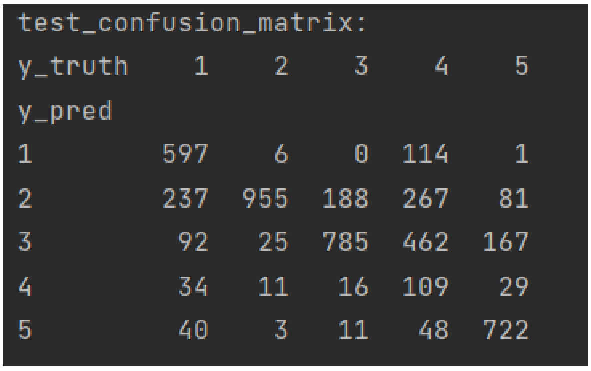
Then set the table and printed it.



```
training_confusion_matrix = pd.crosstab(np.array(training_pred), np.array(training_label), rownames=['y_pred'], colnames=['y_truth'])
print("\ntraining_confusion_matrix: ")
print(training_confusion_matrix)


test_confusion_matrix = pd.crosstab(np.array(test_pred), np.array(test_label), rownames=['y_pred'], colnames=['y_truth'])
print("\ntest_confusion_matrix: ")
print(test_confusion_matrix)
```



```
training_confusion_matrix:
y_truth     1     2     3     4     5
y_pred
1        3685    49     4   679     6
2        1430  5667  1140  1380   532
3         508   208  4670  2948   893
4         234    60   123   687   180
5         143    16    63   306  4389
```



```
test_confusion_matrix:
y_truth    1    2    3    4    5
y_pred
1        597    6    0  114    1
2        237  955  188  267   81
3         92   25  785  462  167
4         34   11   16  109   29
5         40    3   11   48  722
```