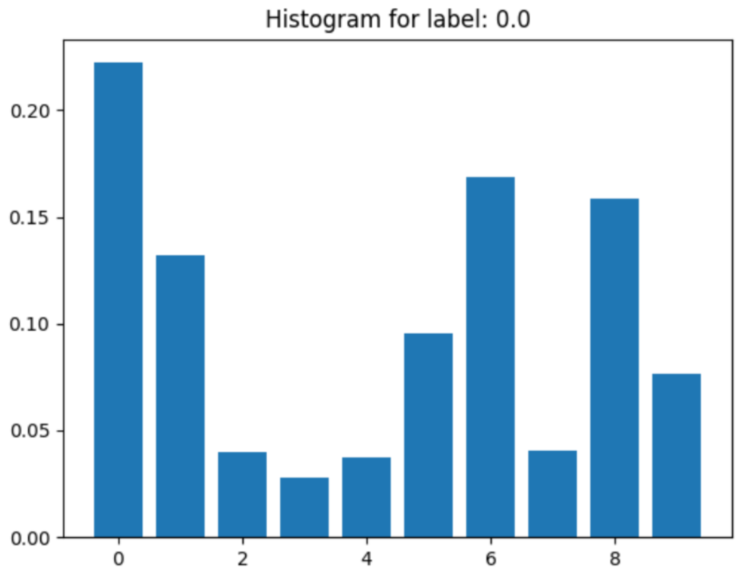
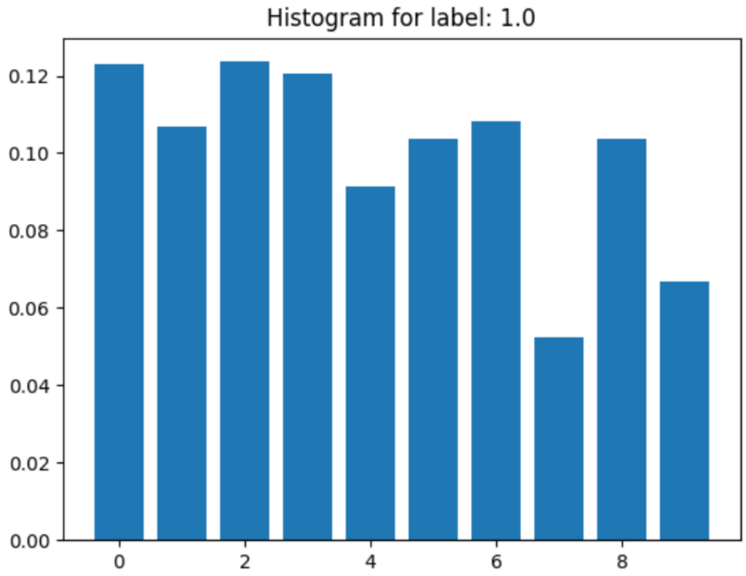
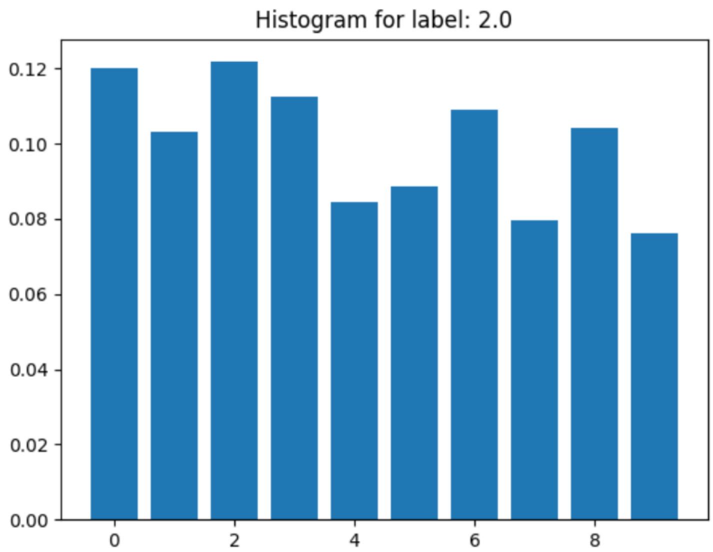
***Note:**

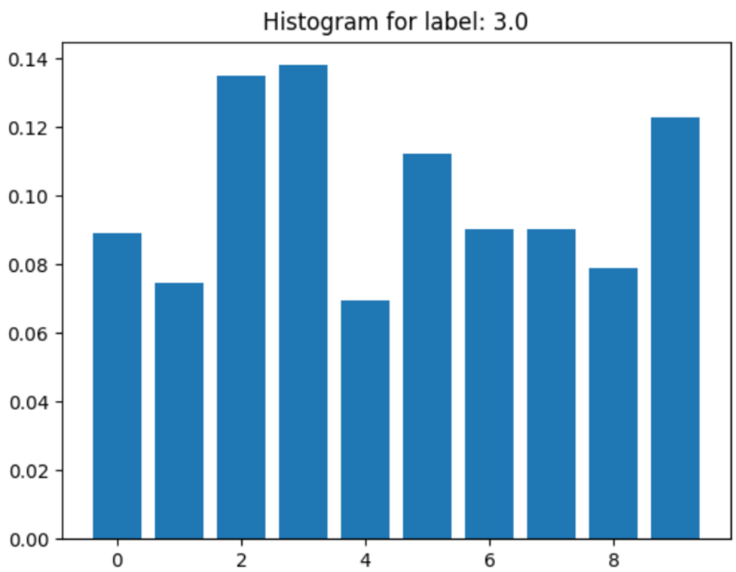
- **I used the *seaborn* module to visualize the confusion matrices. Run “pip install seaborn” to install dependencies**
- **After running main.py, you’ll be prompted with the option to use the existing files on disk (which will not be submitted so the program will automatically go to the other case the first time it’s run). After the files are saved to disk, you have the option to load these files for a much faster runtime. These instructions are also printed in the terminal when run.**

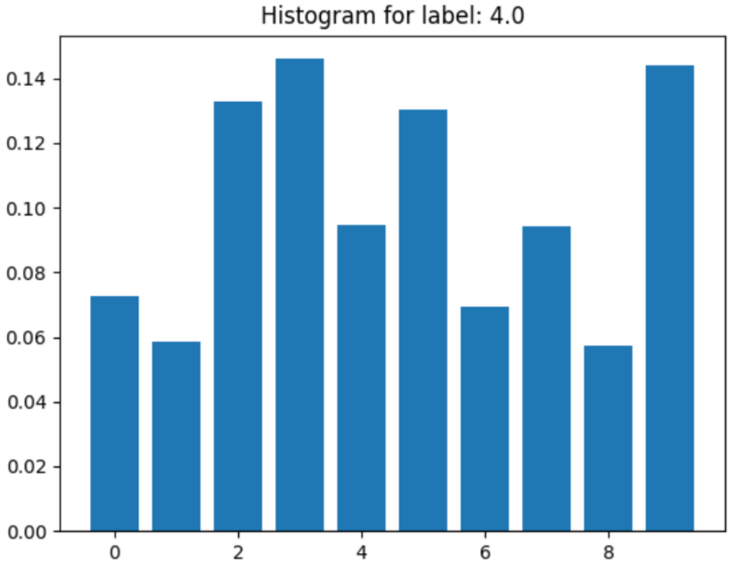
4.
The histogram with label 0.0 has dramatic differences in bin sizes, and I would expect it to be the easiest histogram to separate because of this. Specifically, the first bin is a clear outlier and our object recognition would surely be able to choose this bin as the correct class. However, the other histograms appear to have a maximum of ~0.04 deviation from the median group. The fairly uniform distribution indicates that our object recognition would have a tough time separating the classes (confusing the system).

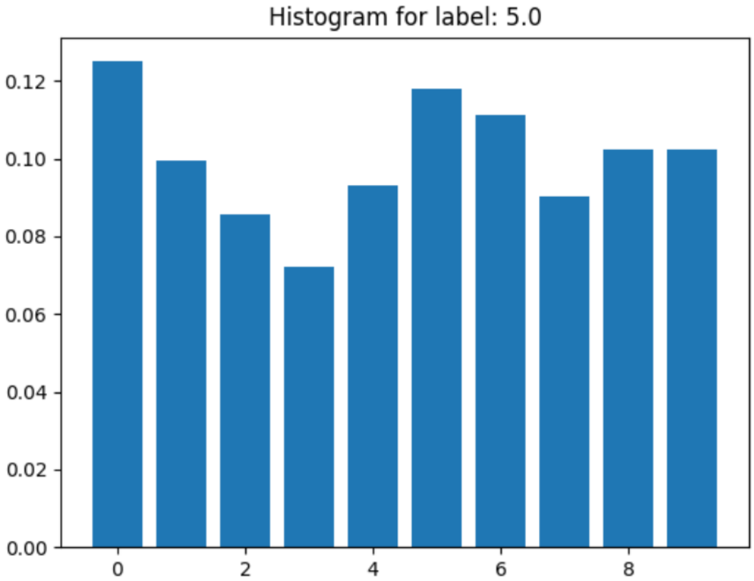


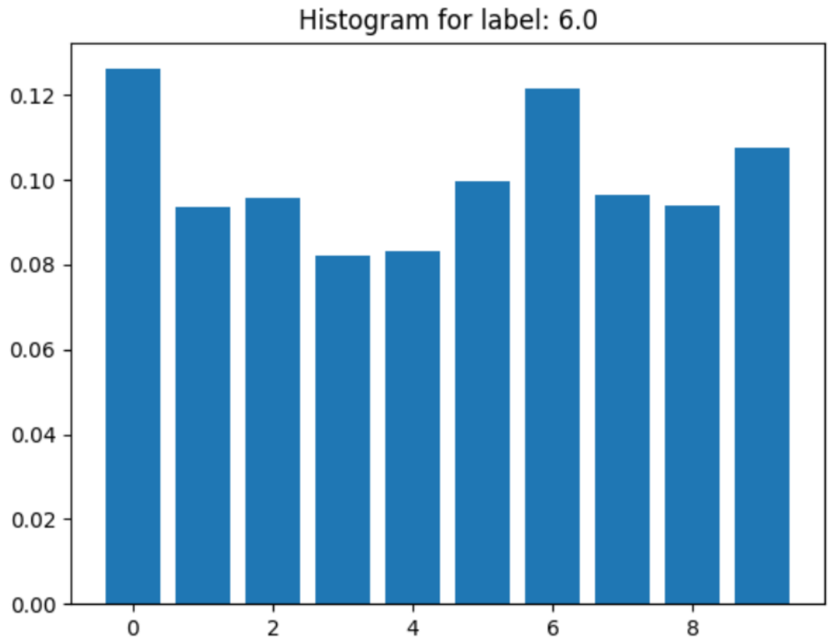

Histogram for label: 1.0
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8

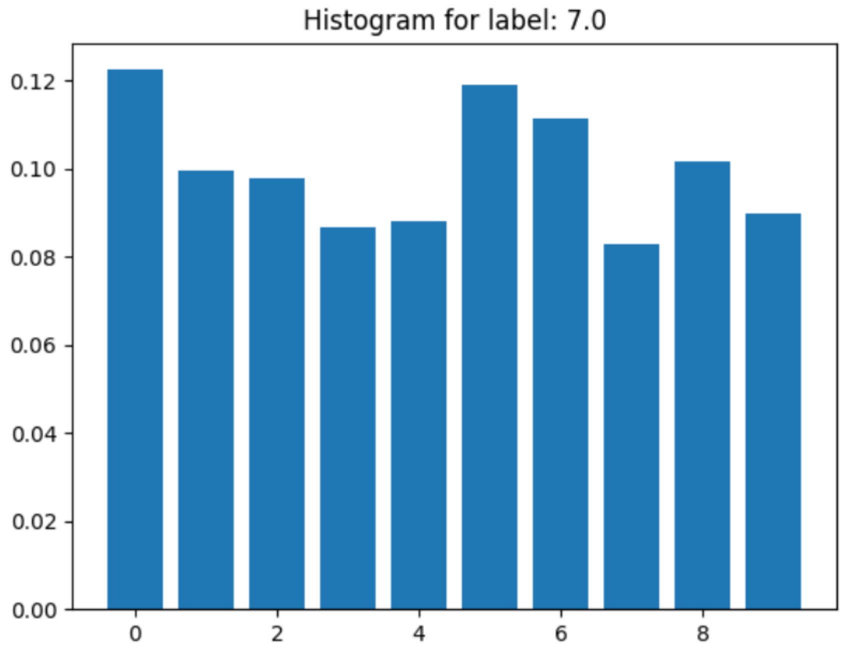

Histogram for label: 2.0
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8

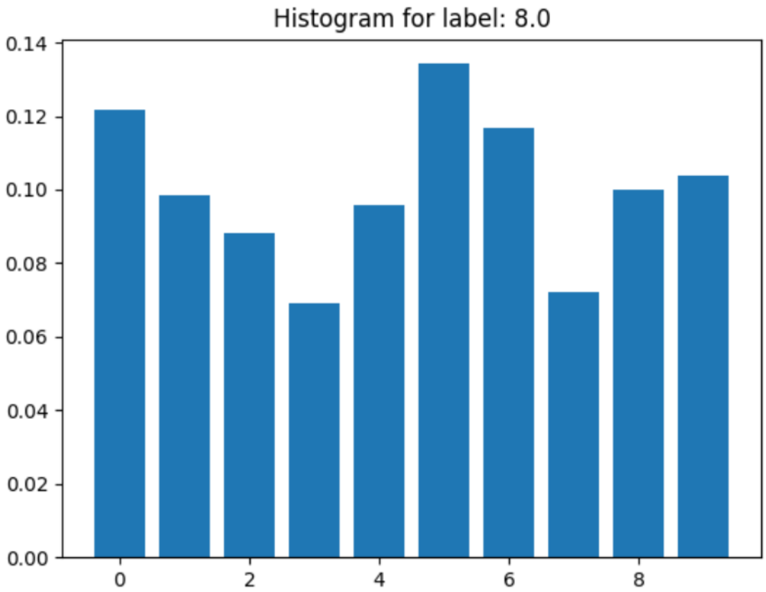

Histogram for label: 3.0
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8

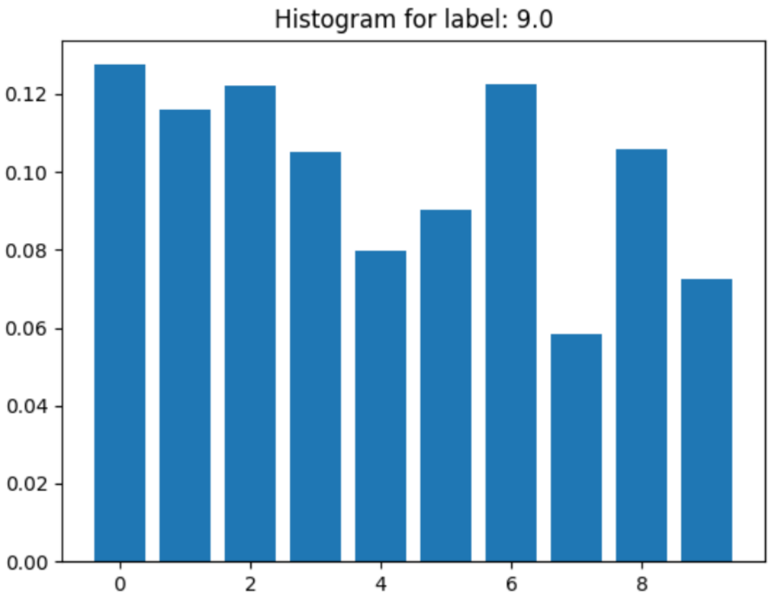

Histogram for label: 4.0
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8

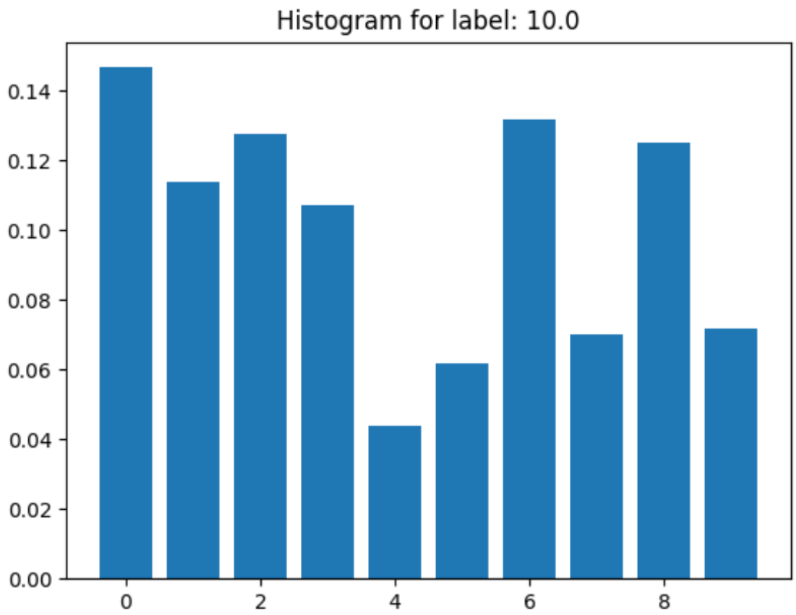

Histogram for label: 5.0
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8

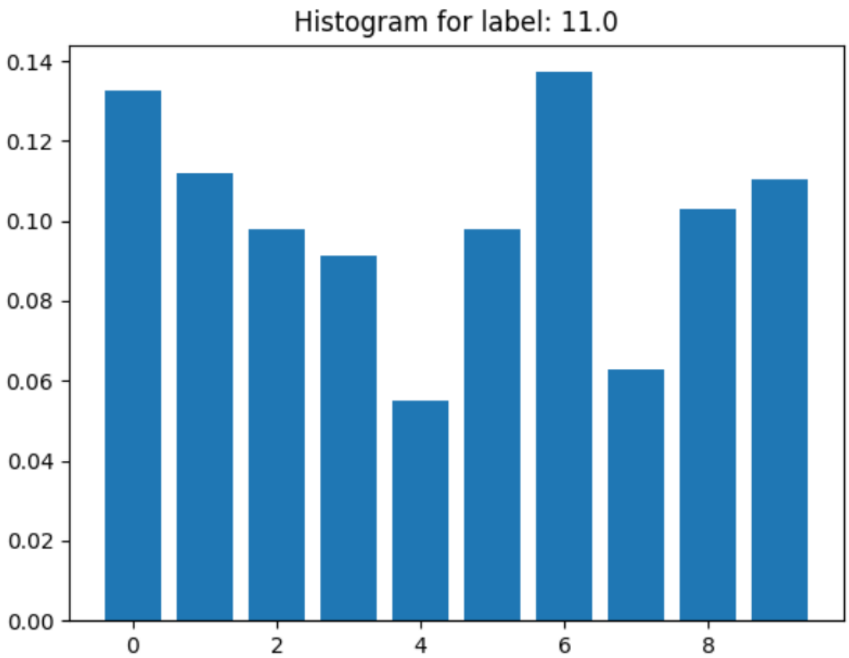

Histogram for label: 6.0
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8

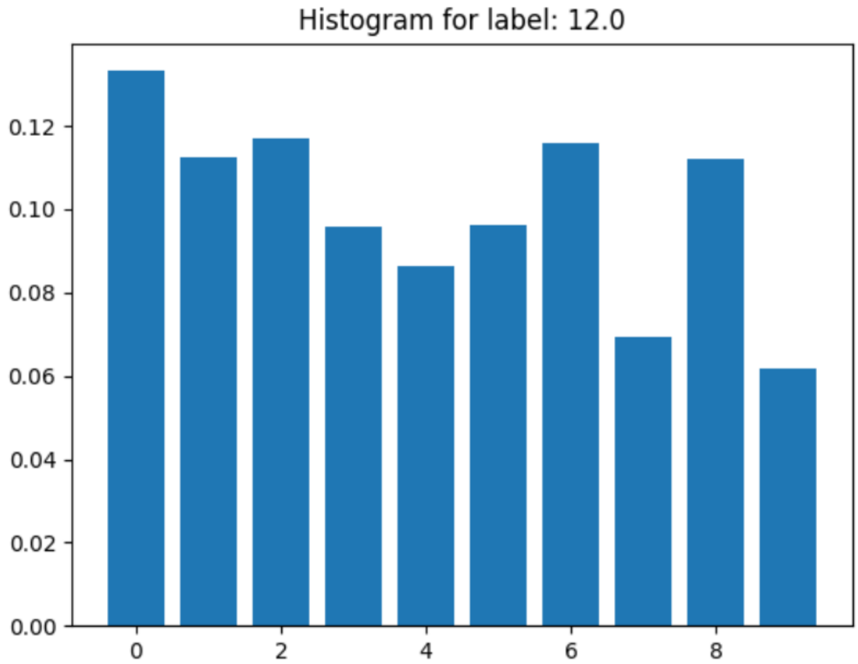

Histogram for label: 7.0
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8

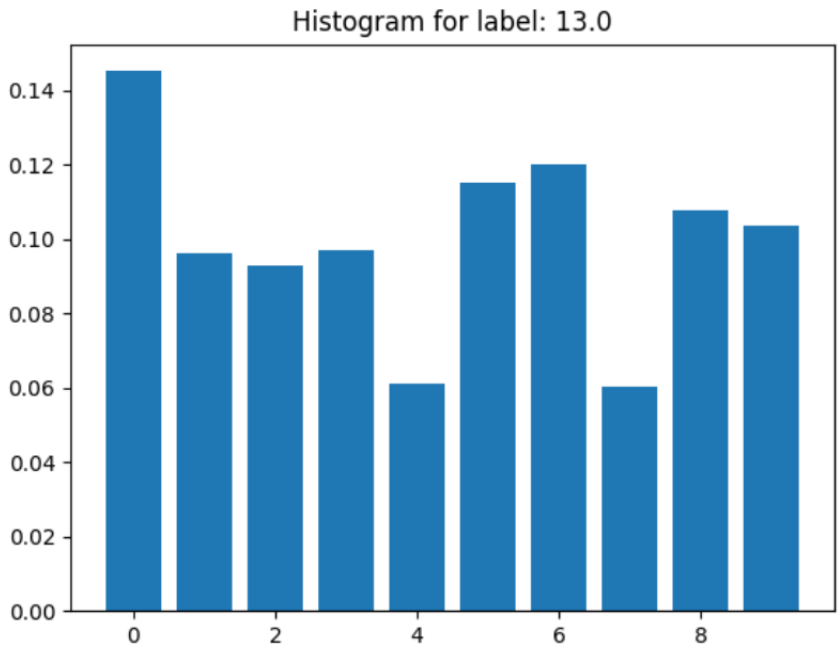

Histogram for label: 8.0
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8

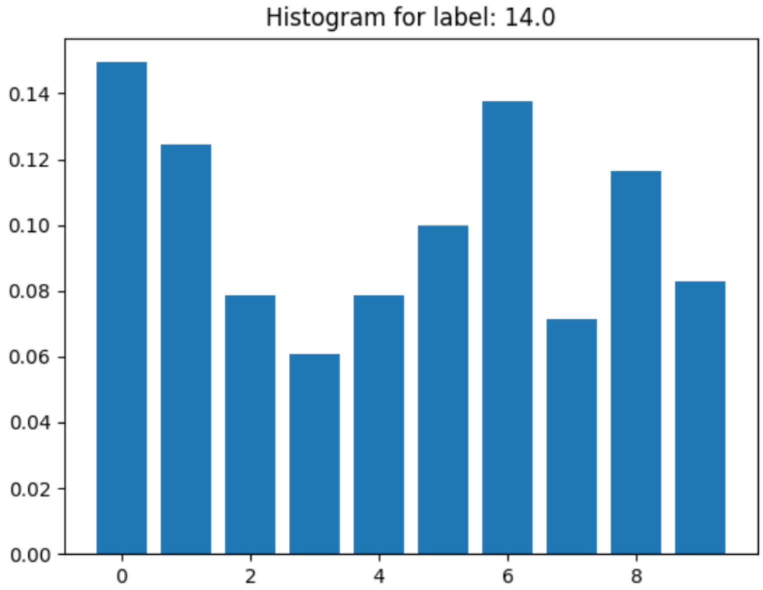

Histogram for label: 9.0
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8


Histogram for label: 10.0
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8


Histogram for label: 11.0
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8


Histogram for label: 12.0
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8


Histogram for label: 13.0
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8

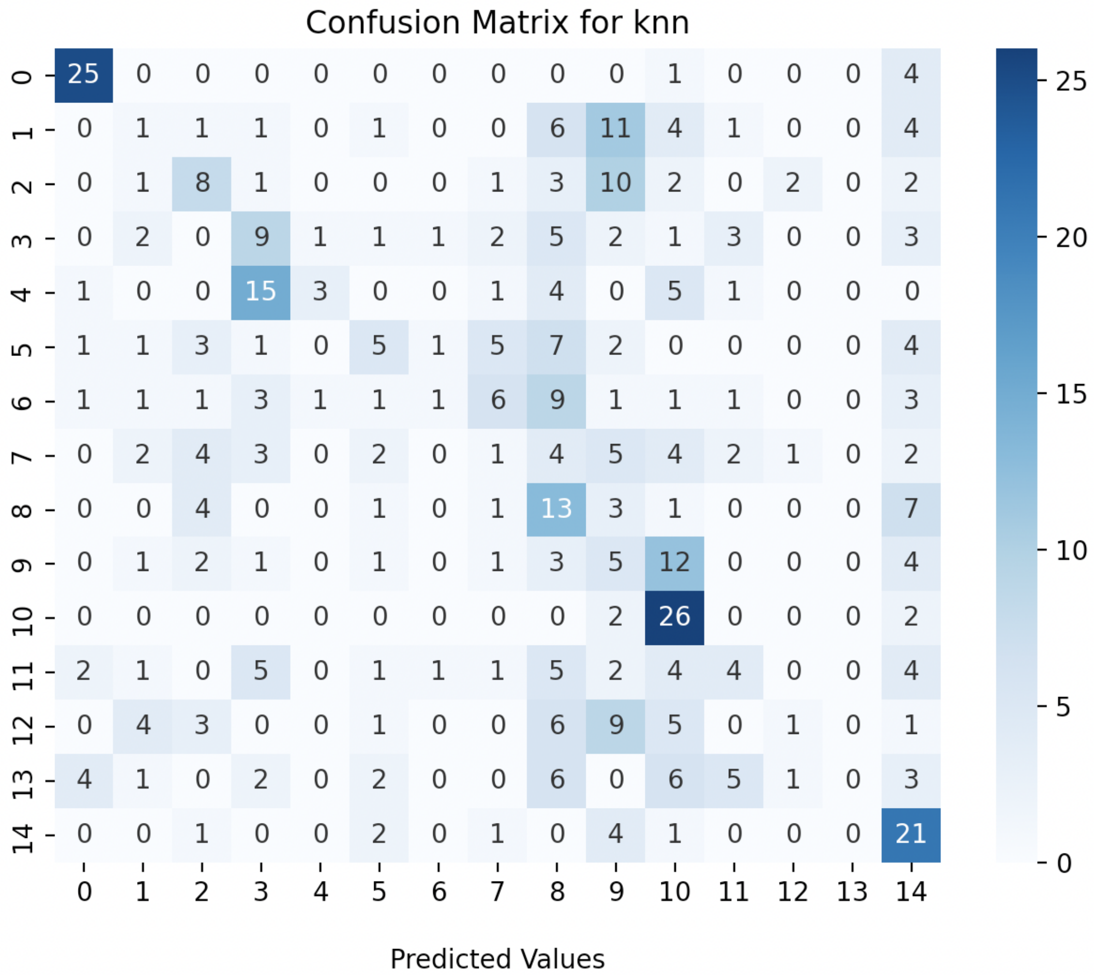


To measure performance of nearest neighbour classifier report classification accuracy and plot confusion matrix (which should be of size 15x15). You will be required to hand in both of these measures as part of your PDF writeup. Experiment with the size of k, how does this effect the performance of the classifier?

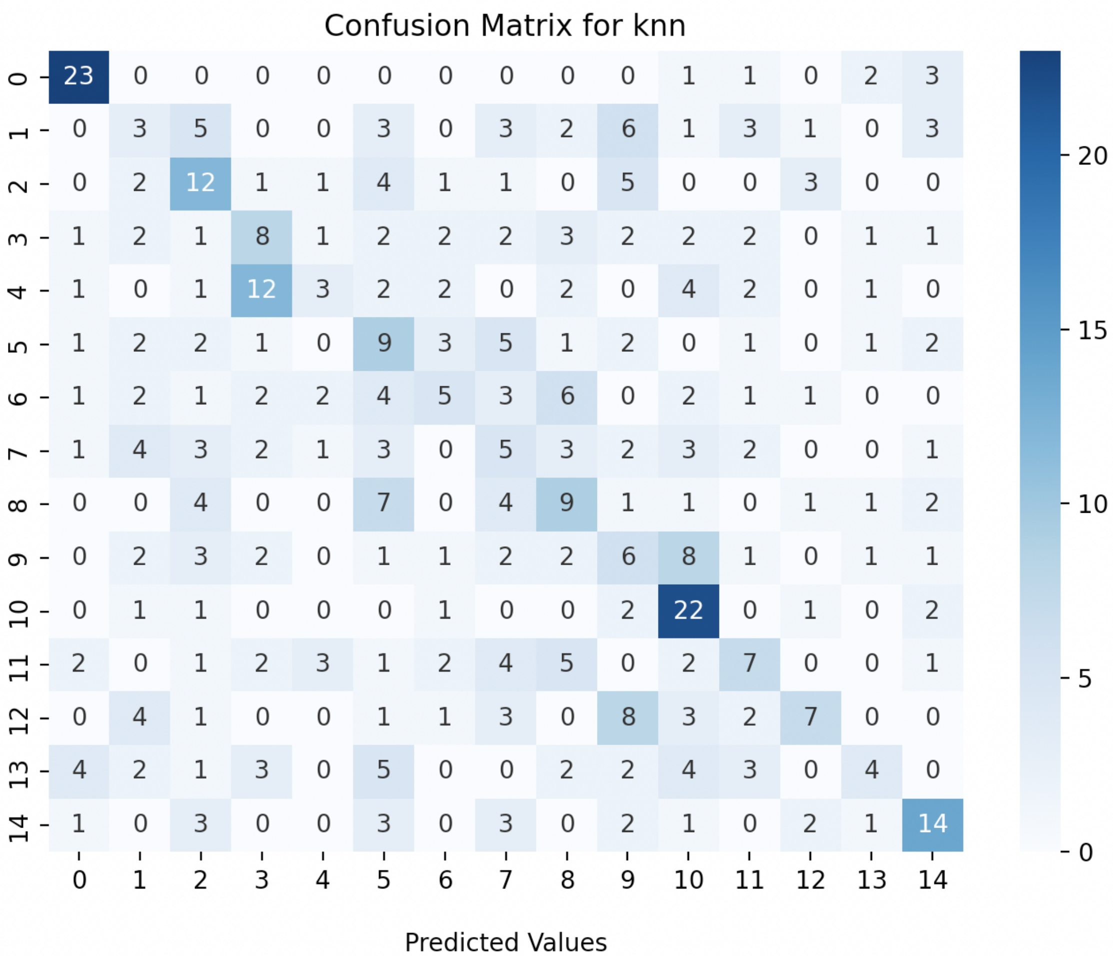
5.
knn_accuracy_score: 0.305
svm_accuracy_score: 0.298 (C=1.0(default))
As the size of k increases, the clusters begin taking on values further from what they should be. Meaning, the more data points we choose to calculate the center, the more it skews the value. However, if we choose too few number of nearest_neighbours (k) < 5, we'll see the center value become less accurate as well.

6.
C is the regularization parameter. The strength of the regularization is inversely proportional to C and must be strictly positive. We can see from the confusion matrices that the C parameter has a sweet-spot, where if set too small or large, accuracy falls proportionally. We can see that a value of 10.0 provided the best results, with the majority of the classes chosen diagonally through the confusion matrix.
In conclusion, both k and C have "sweet-spots" giving us the most accurate results.

K = 100

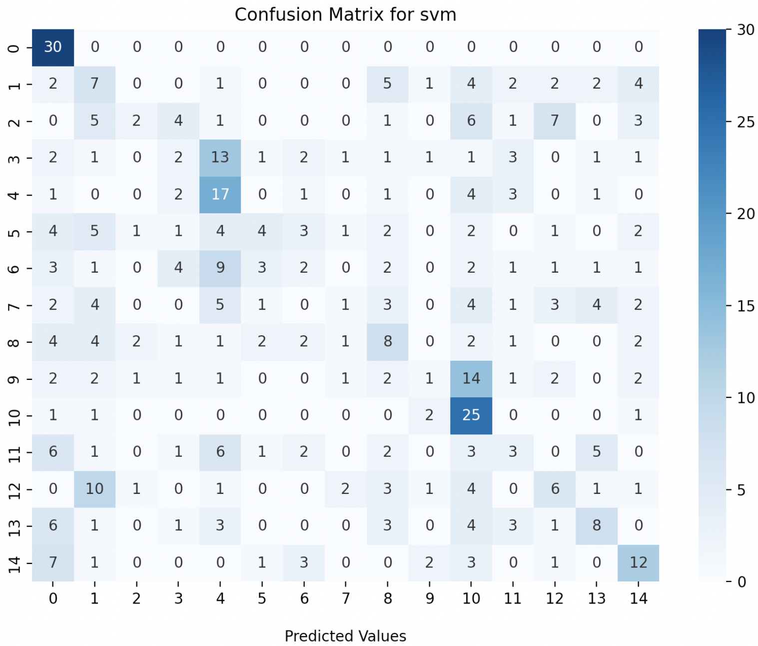


K = 10



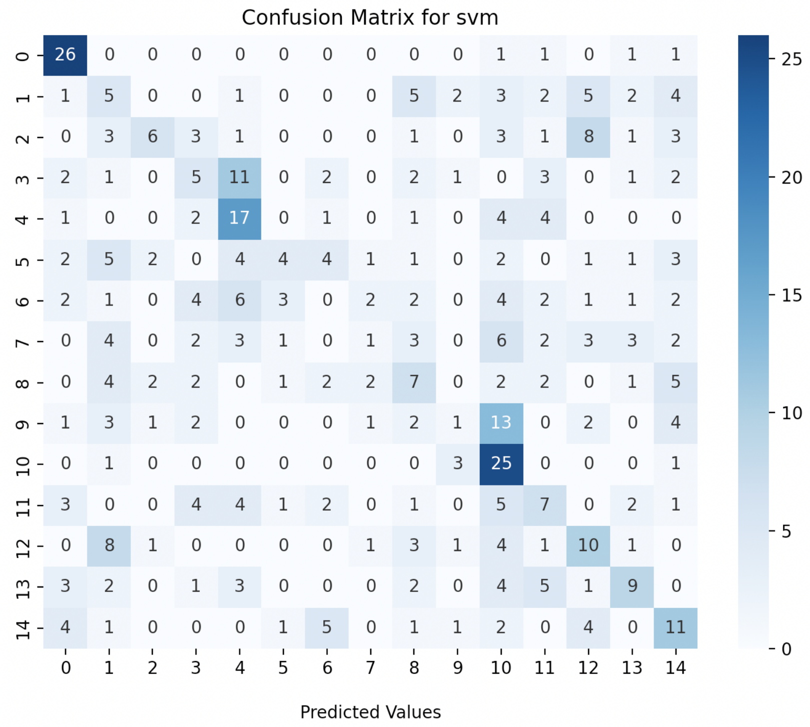
C = 1.0 (default)

Svm_accuracy = 0.28444444444444444



C = 10.0

svm_accuracy_score: 0.298



C = 1000.0

Svm_accuracy = 0.24444444444444444