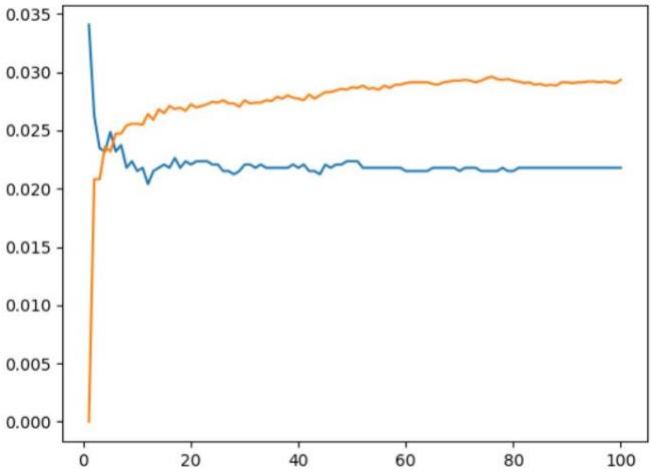
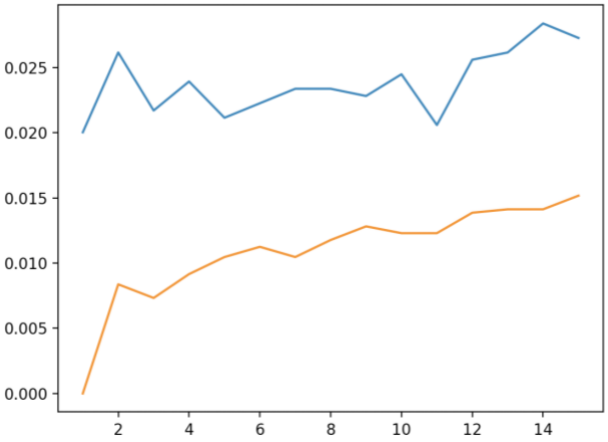
Zachary Panzarino
903305160
HW 2

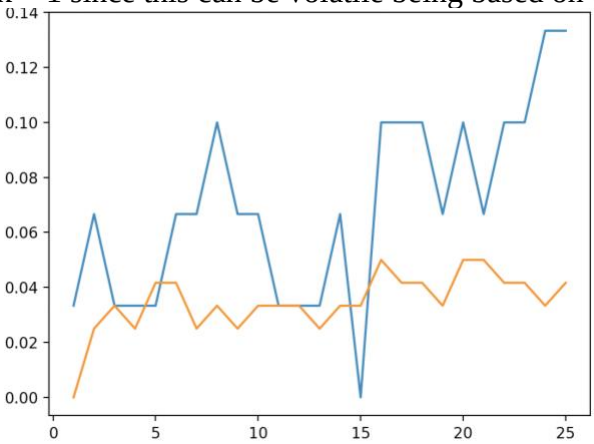
1. k-Nearest Neighbors
    a. Testing data is represented in blue. Training data is represented in orange.



For HTRU, k = 12 since it minimizes the test loss but is more efficient than higher k values.

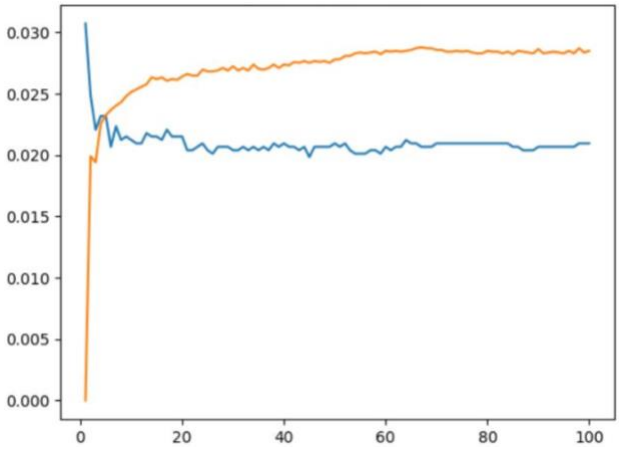


For Optdigits, k = 11 since it minimizes the test loss and is more stable then when k = 1 since this can be volatile being based on only 1 neighbor.

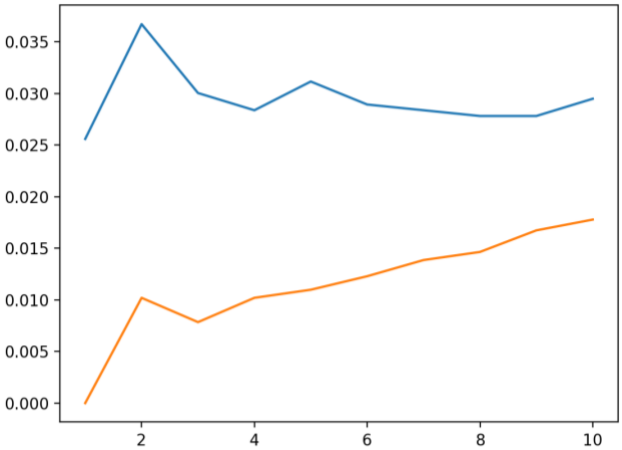


For Iris, k = 15 since this clearly minimizes the loss more than any other potential value of k.

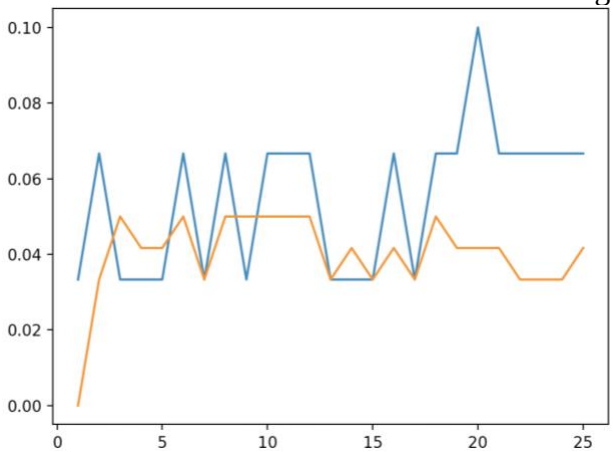
b. *Manhattan*



For HRTU, k = 25 since it minimizes the test loss but is more efficient than higher k values.

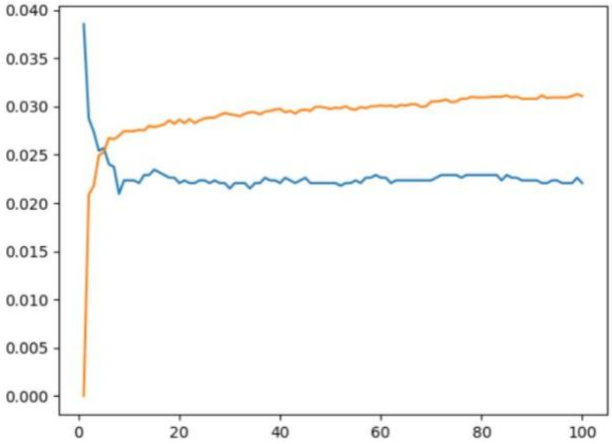


For Optdigits, k = 9 since it has a lower loss than most of the other possible k values and is more stable than k = 1 due to a higher number of neighbors.

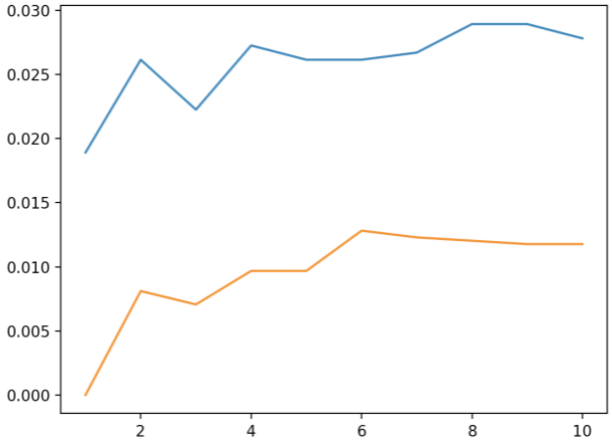


For Iris, k = 9 since it is a local minimum for loss. There are also more points with a similar loss, but this seems to be the best balance of a high enough k but not so high that it comes with a reduction in computational efficiency.

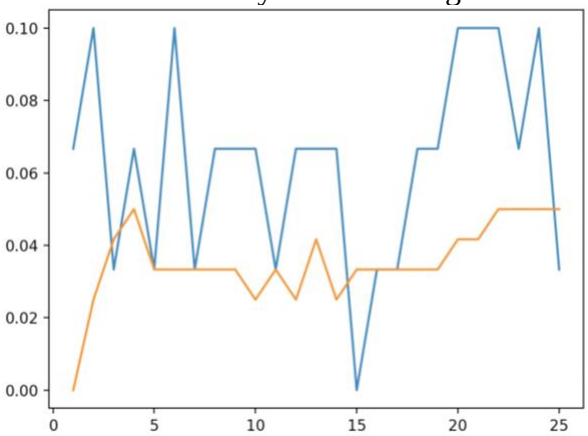
Mahalanobis



For HRTU, k = 10 since it minimizes test loss but is more efficient than higher k values.



For Optdigits, k = 3 because it is a local minimum for test loss and is more stable than k = 1 which only utilizes 1 neighbor.

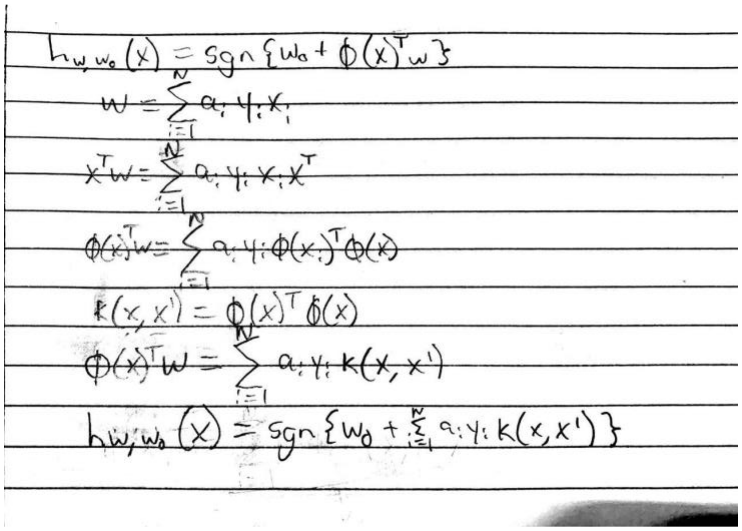


For Iris, k = 15 since it minimizes the loss significantly compared to any other potential value of k.

Performance:

For HRTU the Manhattan function had the lowest loss at k = 25. This was very similar to the loss of the other functions. Therefore, it might be more practical to use a lower k value with a different function that results in better efficiency.
For Optdigits, the Euclid function had the lowest loss at k = 11. This is a nice medium value of k that could work well in practice, although a lower k might have some performance increase.
For Iris, the Euclid function had the lowest loss at k = 15. This had the optimal performance time and should be used in practice.

2. Logistic Regression
    a. The logistic regression converged on iteration 110 with score 0.97654. The KNN converged at about 0.98.
    b. The logistic regression for optdigits converged on iteration 3376 with score 0.95993. The logistic regression for iris converged on iteration 8816 with score 0.9. Both performed worse than KNN.
    c. As shown through the two classes in HTRU, the logistic regression performs optimally for binary data. The KNN function for HTRU took significantly longer to run than the logistic regression, while only slightly more accurate. The others showed a larger loss difference with improved runtime.
3. Support Vector Machines
    a. The given expression can be rewritten to only use the kernelized x as shown below:

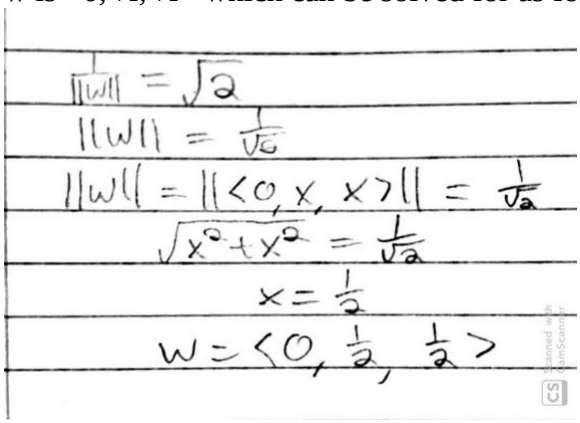


This needs to be done in order to show that individual x's are not important since only pairs of x's need to be used. Therefore, rather than using each value alone a kernel can be used. In the case that the given alpha value is 0, then the vector does not need to be computed (and it is not a support vector).

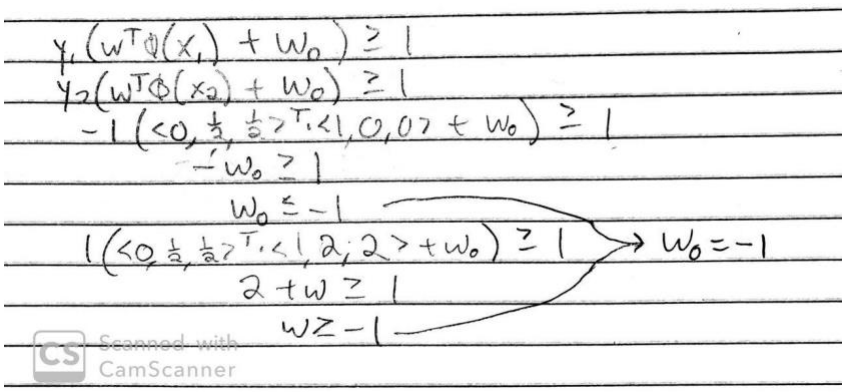
    b. For klin, the dimension is D. For kpoly, the dimension is $\binom{D+M-1}{M}$. For kexp, the dimension is infinite.
    c. Since there are only two points, the decision boundry must be perpendicular to the line connecting the points after transformation (1, 0, 0) and (1, 2, 2). Since the optimal vector w is perpendicular to the decision boundary, it must be parallel to this line. Therefore, a vector that is parallel to w is <0, 2, 2> which is the direction vector between these two points.

d. To find the margin, first find the distance between the two points: | <1, 0, 0> - <1, 2, 2> | which comes out to be $2\sqrt{2}$. Since the margin is half of this distance, the margin is $\sqrt{2}$.

e. w is <0, ½, ½> which can be solved for as follows:



f. w0 is -1, and it can be solved for as follows:



g. The discriminant function is as follows: