Exam 1

Tylman Michael
CSE 546 Machine Learning

2/24/2023

1 Problem 1

The image for this plot is in Figure 1. The reason why these samples can be correctly classified without any regularization is because the SVM will look to maximize the margin which results in 0 items missed without any care for the fact that these 2 points are outliers. However, once we start to use regularization, it lowers the impact of outliers on our decision boundary, which *could* result in the decision boundary moving to favor a cleaner margin between the two easier clusters.

1.1 a)

Solution b can be better than solution a if those two points really are outliers. If they are a poor indication of the general behavior of the phenemenon being studied, then they should not be allowed to have such a strong impact on the learned decision boundary of our model. We want regularization to make our model less sensitive to outliers.

1.2 b)

Solution a can be better than solution b if those two points are not outliers, but rather our training set overall is a poor indicator of the behavior of the phenomenon. If class A actually was bimodal in behavior, with 2 possible clusters with a probability of 13/15 for the larger, and 2/15 for the smaller, then our data is a perfect model of the phenomenon and every point should be considered. Interstingly enough, an example of something bimodally distributed is DNA Methylation, which I worked on previously before coming to UofL. This can be seen in figure 2 of the paper located Here if you're interested in learning more.

2 Problem 2

The sample which cannot be classified correctly is shown in Figure 2 as the darker pink star. I drew this such that 2 of the 3 nearest neighbors are of the correct class, such that it will work with k=3, but then 3 of the nearest 5 neighbors are of the wrong class such that it will not classify correctly with k=5. I then padded the 2 correct and the 3 incorrect neighbors with enough of the same class such that they would have enough neighbors close by that they will entirely fill the all of the top 5 closest neighbors with members of the same class to ensure they would be classified as correct.

Then, to not muddy the water with my work, I placed the remaining 9 members of each class in a little grid off on their own such that they would be trivial to see that they are going to be correctly labelled.

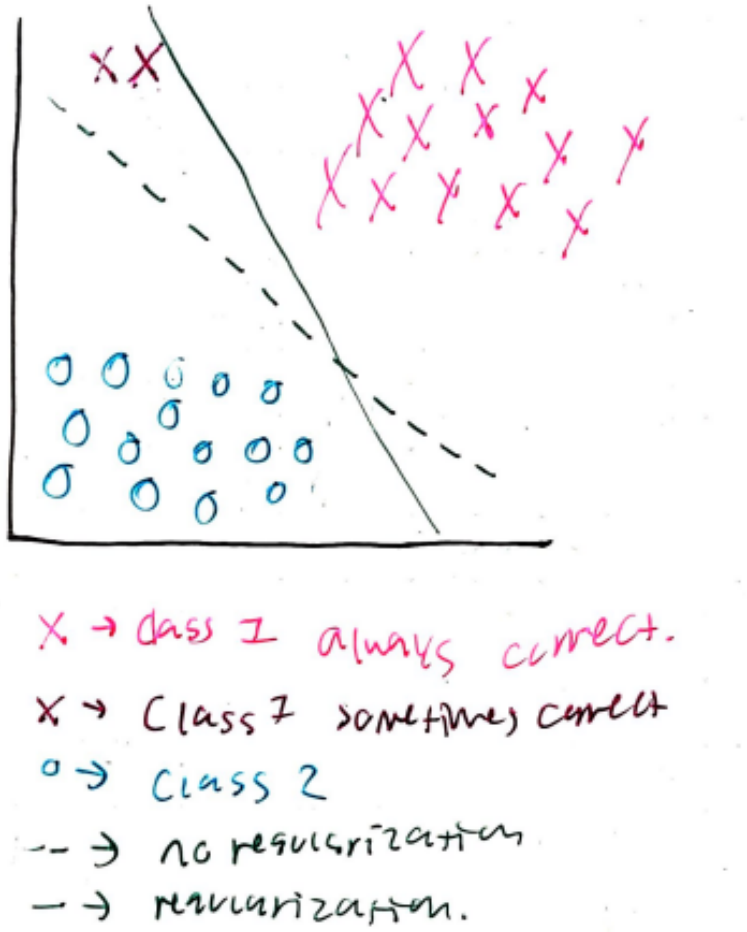


Figure 1: Plot for Problem 1

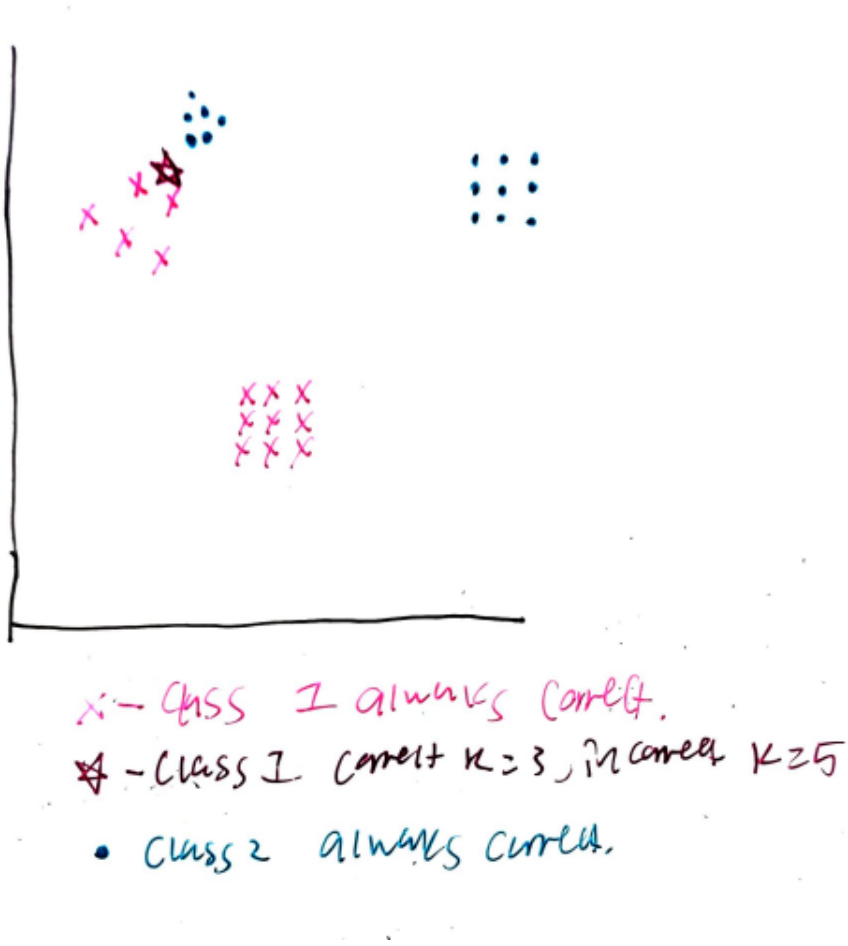


Figure 2: Plot for Problem 2

3 Problem 3

3.1 a)

I would employ the Naive Bayes classifier for this data primarily due to the following two reasons.

1. The amount of blue dots that are floating above the orange cluster, along with the small shared barrier of orange and blue markers to the left of the orange cluster, would be regions where a KNN classifier would likely struggle, and no clear linear boundary exists that can outperform a classifier that considers the floating points above the orange cluster.

2. The data seems to vaguely be normally distributed around 2 means, with the orange dataset having an extremely small variance compared to that of the blue data.

Since this dataset seems to meet the assumptions taken by a Naive Bayes classifier, and has evidence of areas where other models would likely struggle, I think the Naive Bayes classifier would perform very well on this dataset. It may be possible that a decision tree-based approach (with a random forest or gradient boosted model) could perform as well as a Naive Bayes classifier, but these trees can be longer to train and require more memory if we were to grow this dataset.

I don't think that the supposed performance gained (if any!) would be enough to overcome how well the Naive Bayes classifier fits this application, and I suspect they would have poorer generalization. For example, if we were to have unknown samples below the orange cluster, to the right of the little outcropping of blue dots, I think they would more likely belong to the blue class. In this case, the Naive Bayes classifier would accurately assess these new samples, but the decision trees would likely not.

3.2 b)

I'd definitely choose a decision-tree based approach for this dataset. The KNN models would struggle on the sparse blue dots that are on the edges around the more tightly-packed orange dot. The blue dots

	DT	rf	Ridge
Index	7	699	21
mean_fit_time	0.07518452405929565	0.48302948474884033	0.006589710712432861
std_fit_time	0.008200230273480508	0.01895513907270191	0.0030282122144871353
mean_score_time	0.0007671713829040527	0.01826488971710205	0.0007261037826538086
std_score_time	2.2898348274780204e-05	0.002575886247859588	8.508545817910581e-05
params	{'max_depth': 8}	{'max_depth': 18, 'max_features': 'sqrt', 'n_e...	{'alpha': 0.21}
split0_test_score	0.6355198018628058	0.7501321764065413	0.6052436952899654
split1_test_score	0.6510997100662925	0.7505541081197071	0.6010076828873177
split2_test_score	0.6411094700172677	0.7456836444199912	0.5946907422545507
split3_test_score	0.6664654418191578	0.7608058326506076	0.6185872571494901
mean_test_score	0.648548605941381	0.7517939403992118	0.604882344395331
std_test_score	0.011753846382596016	0.0055418535381469	0.008758381186980357
rank_test_score	1	1	1
split0_train_score	0.7349418999993165	0.9471680408214387	0.6056854495887198
split1_train_score	0.7476653419297844	0.9481788560026908	0.6072523252854863
split2_train_score	0.7417372631414741	0.9471798870778553	0.6092210751647267
split3_train_score	0.7401602000260372	0.9486331625352517	0.6012482786323003
mean_train_score	0.741126176274153	0.9477899866093091	0.6058517821678083
std_train_score	0.004536311766915701	0.0006366323602614115	0.0029382634178992096
best_final_test	0.6443869545715294	0.7464982261423195	0.6043793992130073
best_cv_test	0.648548605941381	0.7517939403992118	0.604882344395331
best_cv_train	0.741126176274153	0.9477899866093091	0.6058517821678083

Table 1: Best Results and MetaData

don't have any strongly defined mathematical distribution that makes it seem like we could leverage a Naive Bayes classifier, nor do the orange dots appear to be normally distributed. There is no one line of separation, unless we were to use a nonlinear projection about the centroid of the orange class, probably with a radial basis function (which we haven't covered).

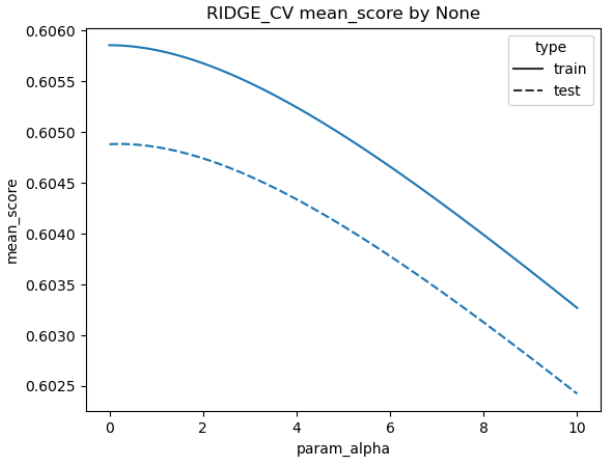
With all of that in mind, the orange class can be cleanly separated into a box which surrounds it on all sides that also does not include any blue dots. This leads me to believe that a decision-tree based approach would likely work very well. I'd have to say to go with a standard random forest approach over a gradient boosted approach purely because this seems to be a rather simple dataset that we could get great performance on with a random forest with faster training time than with a gradient-boosted classifier.

4 Problem 4

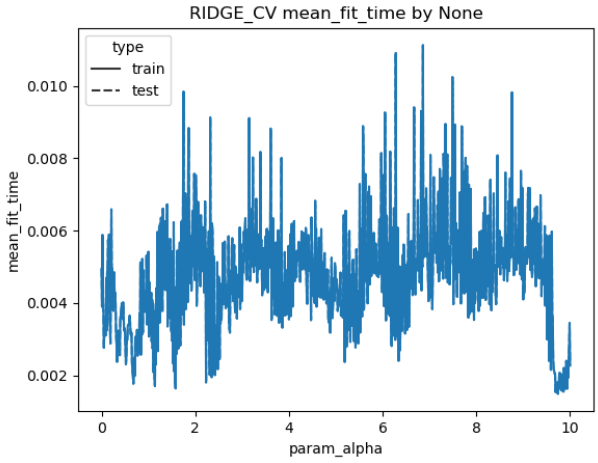
For every model in problems 4 and 5, I chose to use a combination of a scaler and a normalizer to force all of the data onto a unit sphere around the origin. This might have been a poor choice for some models, but due to time constraints I decided it would be better to do the data normalization that I was the most comfortable with understanding conceptually since the focus of this exam is on the models themselves, and not the preprocessing.

I will list a single table with the best results (with the param lists cut) for every model for ease of reference in Table 1. The most important rows to recognize are the last few, where best_final_test is the accuracy of the best model on the test set, best_cv_test and best_cv_train are the average test accuracies of the best model across the CV splits for test and train respectively. A couple of those values are available earlier in the table as well, but I included them at the all together for ease of use.

Additionally, I did do a 4-fold cross-validation for all questions in problem 4 simply because my code assumed that it would be working on cross-validation folds, and it was easier to just make my machine work 4x as hard instead of fixing my code to allow for no cross-validation.



(a) Ridge Regression Mean score



(b) Ridge Regression Average Train time.

Figure 3: Ridge Performance

4.1 Ridge Regression

For the Ridge Regression classifier, I did not have a discrete that I could use to group my data and label it with my pre-existing software. So, I kluged a patch in order to make my plotting structure work without a grouping variable. I will certainly find a better solution for it in the future, but due to the time constraints we will have to work with it for now.

The results of my cross validation for the Ridge Regressor are shown in Figure 3. I have included the time complexity required as well as the mean performance. As we can see, the alpha parameter initially gave some boost to accuracy, but then caused a steady drop in accuracy as it increased. However, the effect of this is overly illustrated on the plot by the automatic choice of y-boundaries. If we look closely, we can see that alpha actually did not have much of an appreciable effect on the performance of the model.

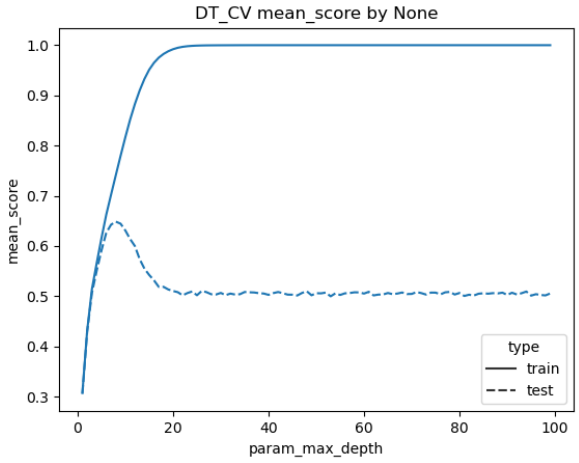
The top and bottom of our graph are a mere .35 percent apart from one another. Admittedly, we can see the difference between the training and the test curve decrease as we alpha increases, so it is lowering overfitting. But, it's really not accomplishing much.

We can see that the best result for our ridge regressor occurs when alpha is .021, but as we stated earlier it does not make much of a difference. However, we can say that there is little to no overfitting occuring, considering the fact that the mean test score and mean train score are within each other's standard deviation. This woudl be a great plus in favor of the ridge regressor, if it had performed better.

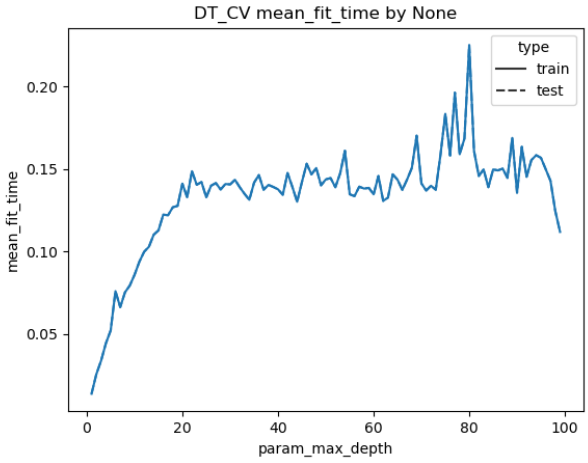
4.2 Decision Tree

The Decision Tree model (DT) suffered from a similar problem as the Ridge Regressor, so it also must deal with the kluged plotting problem. I also only varied the maximum depth pruning parameter, since the assignment only requested a parameter (not plural parameters), and there is no singular "pruning" parameter in sklearn. I have plotted the results of my training in Figure 4

We can see here that the DT model saw great improvement up until a very clear maximum where the model began to classically overfit around a maximum depth of 8. At a max depth of 8, we got our best result as shown in Table 1, which also showed only mild beginnings of overfitting. The overfitting continued to get worse from this point until about max depth of 20 where it seems like our model became fully overfit and saw no significant change for the rest of the experiment.



(a) DT Mean score



(b) DT Average Train time.

Figure 4: DT Performance

4.3 Random Forest

The Random Forest model (RF) did manage to have multiple items for me group by for my plotting software, so that's nice. As such, there will be four plots for the RF in Figure 5. Originally, I had my n_estimators search go from 100 to 700 in steps of 100, but I discoverd that the model was already very far overfit by the very first datapoint. Additionally, I had my max depth set to go from 10 to 100 in steps of 10, but the performance plateau'd already at 20. So, I re-ran this experiment with estimators ranging from 1 to 20, and max_depth ranging from 1 to 30. This did increase the total amount of time spent training, but I was able to work concurrently with the writeup and the training so that it wasn't too bad.

Once I did so, I found that the number of estimators capped out performance very quickly, reaching near peak performance at about 5 estimators, but slowly increasing until 20. I think if I had more time, I would have expanded my search past 25, but with my previous mistakes I had boxed myself into a corner.

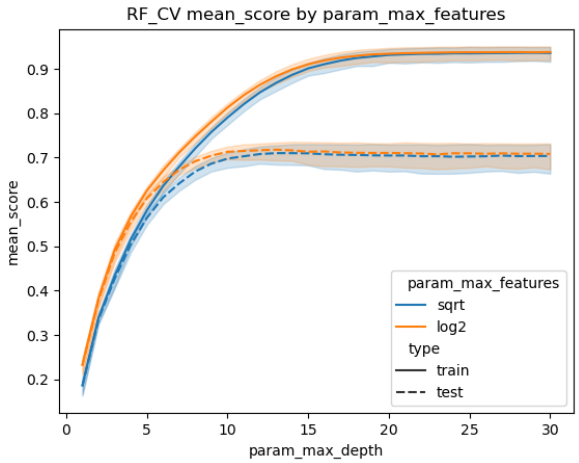
Regardless, to explicitly stte the best parameters I had for the RF classifier, they are: "'max_depth': 18, 'max_features': 'sqrt', 'n_estimators': 20" Across every split, the log2 feature selection outperformed the sqrt feature selection. We can additionally see overfitting begin to occur before a maximum depth of 10, as the training and testing curves split from one another.

Overall, the random forest performed well with moderate overfitting at the best selected model with a training accuracy of .947 and a testing accuracy of .75. However, it did generalize very well when applied to the test set, where it only lost .005 performance when going from the best average validation score to the test dataset.

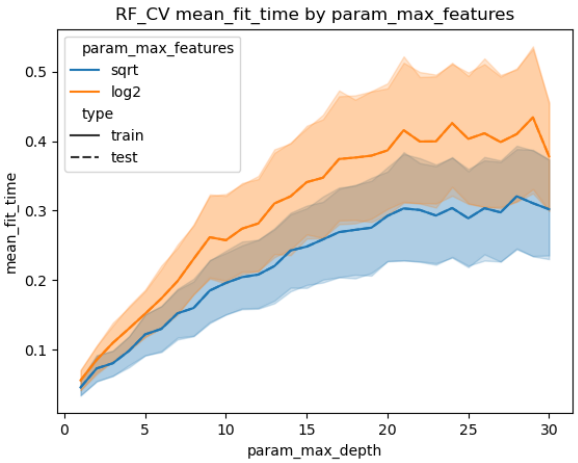
4.4 Comparison

The random forest classifier comes out ahead, which is honestly to be expected when we consider our options. If our dataset had any real complexity to it, a linear model likely might not be able to perform well. Additionally, it makes sense that a random forest of many decision trees will outperform a single decision tree for complex data. This also aligns with a comment you made in class in regards to the random forest classifier being one of the best classifiers we will cover in this class with an extremely wide array of possible applications.

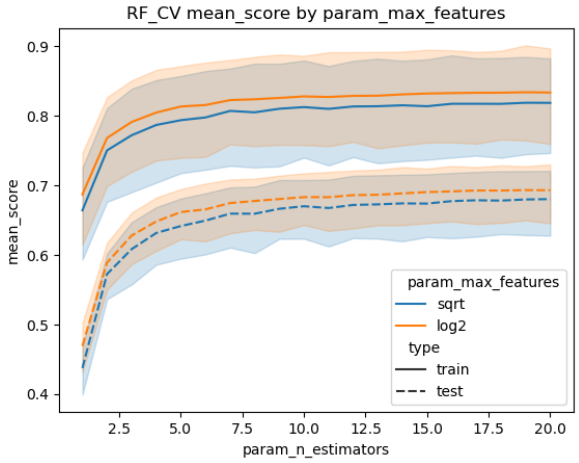
To make the feature importances more easily compared, I normalized the feature importances for each model. For the Ridge regressor, I normalized the absolute value of the feature coefficient. The results of this can be seen in Figure ??. We can see that there's a consensus that Median Income is a strong driver of price, which makes a lot of sense. Essentially, rich people are very likely to spend more money on their homes. Something that surprised me was how much the Ridge regressor gravitated



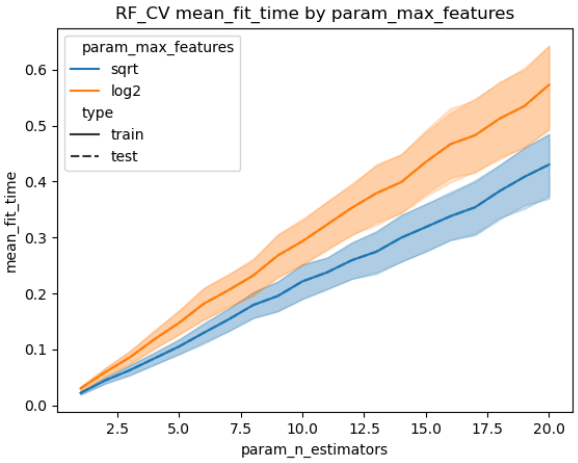
(a) RF Performance by Max Depth



(b) RF Average Fit Time by Max Depth



(c) RF Performance by Estimators



(d) RF Average Fit Time by Estimators

Figure 5: RF Performance

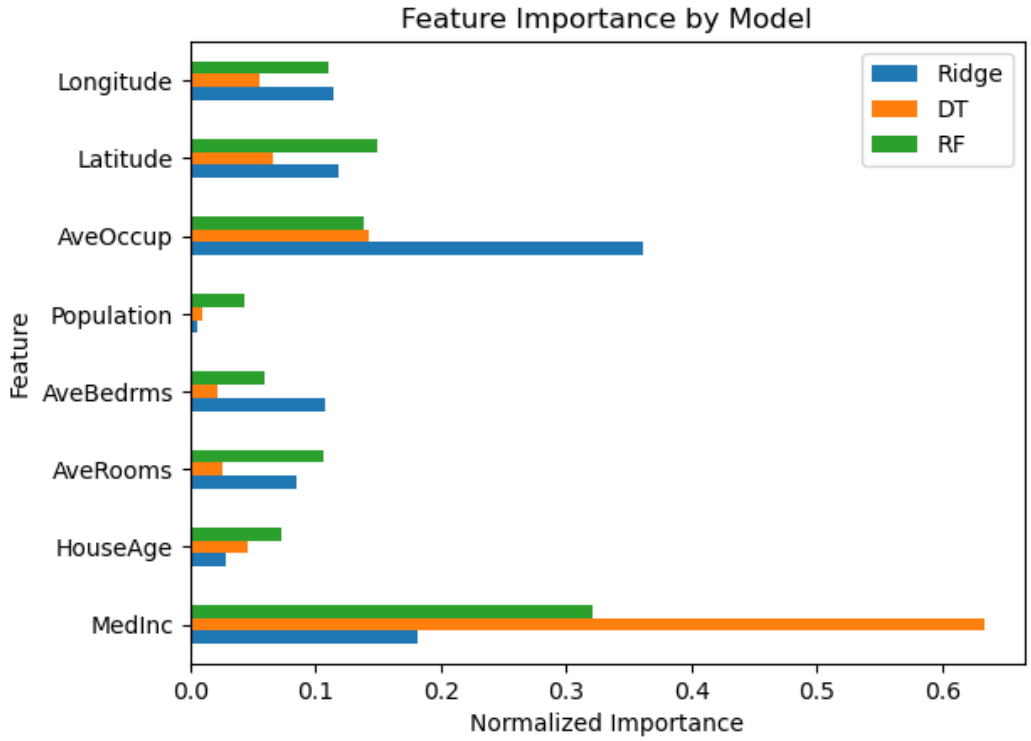


Figure 6: Feature Importances

towards the average occupancy feature. I suppose that would be bounded into simpler discrete values likely in the range of 1 to 7 that can be simply measured with my normalization.

I think that the median income provides the most clear information that can be leveraged by a simple model, which is why this feature so strongly dominates in the single decision tree model. However, I do see how the other features could provide more complex information that a more advanced model can leverage which is why it doesn't dominate as much in the Random Forest model. Additionally, the Decision Tree is much more intentional about building it's one tree than the Random Forest is about building the multitude of trees it has. Therefore, the Random Forest model can afford to spend some energy on considering features that might not be as easily leveragable.

5 Question 5

5.1 a)

I used the same functionality as the previous questions to generate my results, which can be looked at in the accompanied notebook. I will note, however, that my notebook will produce some additional plots if a model has multiple values that I mark as being and x value, which are accuracy contour plots. I've decided not to try and include these plots in my analysis, but you can feel free to look at them if you'd like to see exactly how two options interact with respect to their combined effect on accuracy.

5.2 b)

I am including my result for question 5 in it's own table in Table 2, rather than lump it in with the others of question 4. This is less space efficient, but better for readability.

Looking at the best result for the Gradient Boosted trees classifier (GB), we can see that this model is performing quite well. With a mean test score of .93 and mean train score of 1, this is a very promising result right out of the gate. Looking at the behavior of parameters as shown in Figure 7, it appears that learning rate introduces a chaotic effect when it is set too high, with the maximum depth providing some stabilization.

	gb
Index	138
mean_fit_time	3.0721352100372314
std_fit_time	0.17360566320392115
mean_score_time	0.001826167106628418
std_score_time	7.43228573555367e-05
param_learning_rate	0.21947368421052632
param_max_depth	3
param_n_estimators	19
params	{'learning_rate': 0.21947368421052632, 'max_de...
split0_test_score	0.9333333333333333
split1_test_score	0.947075208913649
split2_test_score	0.8997214484679665
split3_test_score	0.9415041782729805
mean_test_score	0.9304085422469823
std_test_score	0.018378948855280845
rank_test_score	1
split0_train_score	1.0
split1_train_score	1.0
split2_train_score	1.0
split3_train_score	1.0
mean_train_score	1.0
std_train_score	0.0
best_final_test	0.9166666666666666
best_cv_test	0.9304085422469823
best_cv_train	1.0

Table 2: Best Results and MetaData Question 5

Additionally, it appears that the higher maximum depth cleanly separates the performance across all factors, with the order of performance nearly always being in the order of 5, 4, and then 3 with 5 being the best performance for the vast majority of the parameters we attempted. However, it appears that the order actually inverts when we approach the optimal performace point for each model. So, despite 4 and 5 beating 3 for most of the plot, the best performing model actually was a model with a maximum depth of 3. The optimal parameters for this model are shown in Table 2 directly unlike the models in Table 1. This is because I designed my table generation to include all options which are shared by all models. Since we are showing only 1 model in Table 2, this means that all options are shown.

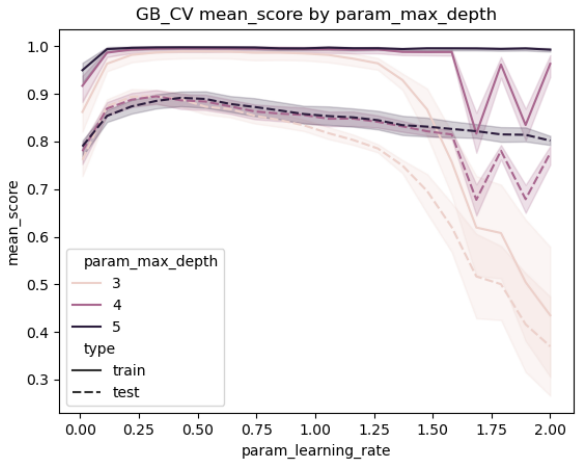
In terms of generalization, this model does very well. We are performing at a test accuracy of .916 with an average validation accuracy of .93. We did achieve a training accuracy of 1, which can be cause for concern regarding overfitting and generalization, but it appears that these concerns are unwarranted given the validation and test accuracies. I will admit, when I originally did this experiment with a much larger gridsearch, I found some models which performed moderately better. I chose to go with this analysis, since I was able to much better plot the performance with smoother continuity than when I had to skip the number of estimators in steps of 5 or 10.

5.3 c)

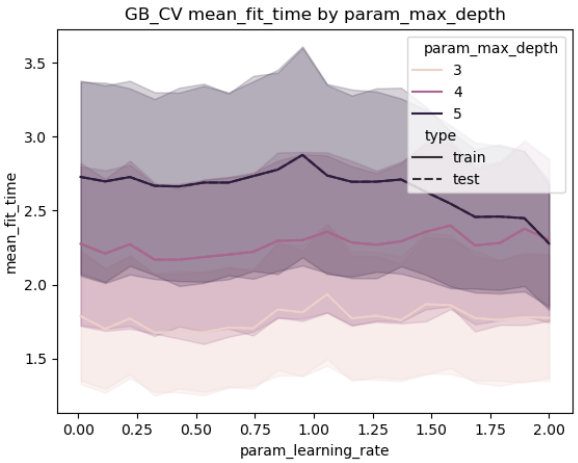
Looking at the relevant features for the GB classifier in Figure 8, we can see that we have our work cut out for us. This mapping of relevance likely doesn't immediately evoke many connections to someone looking at it. However, if we look closely we can see a method to the madness.

When drawing a number, humans will likely make "errors" in the direction that we were already going to move our pencil, and we are usually more likely to overflow in the direction of our writing. Most people naturally write from top left to bottom right when making connected movements. For example, think about how you would draw a 2, 3, or 5. Most people start in the top left corner of the number, and move to the bottom right. You might think that the 5 is an exception because many people draw the lower part of the 5 followed by the horizontal line. But this is what I mean by "connected movements", since draing the bottom part of the 5 goes from top to bottom, and the hat goes from left to right.

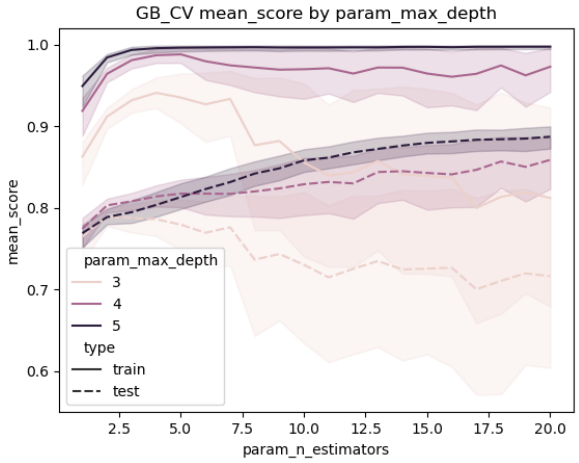
Following these ideas of how people draw images and make errors, we can see that the image begins to make sense. The bottom right quadrant of the image appears to be more important overall, which is where we'd expect to see a lot of unique errors relating to how the number is drawn. Thinking critically, we can estimate that the numbers 2, 3, 5, 6, and 9 are far more likely to have items overflowing to the bottom right than the other numbers.



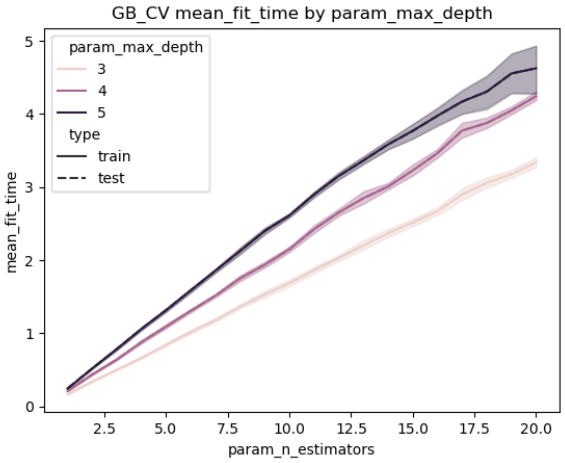
(a) GB Performance by Learning Rate



(b) GB Average Fit Time by Learning Rate



(c) GB Performance by Estimators



(d) GB Average Fit Time by Estimators

Figure 7: GB Performance

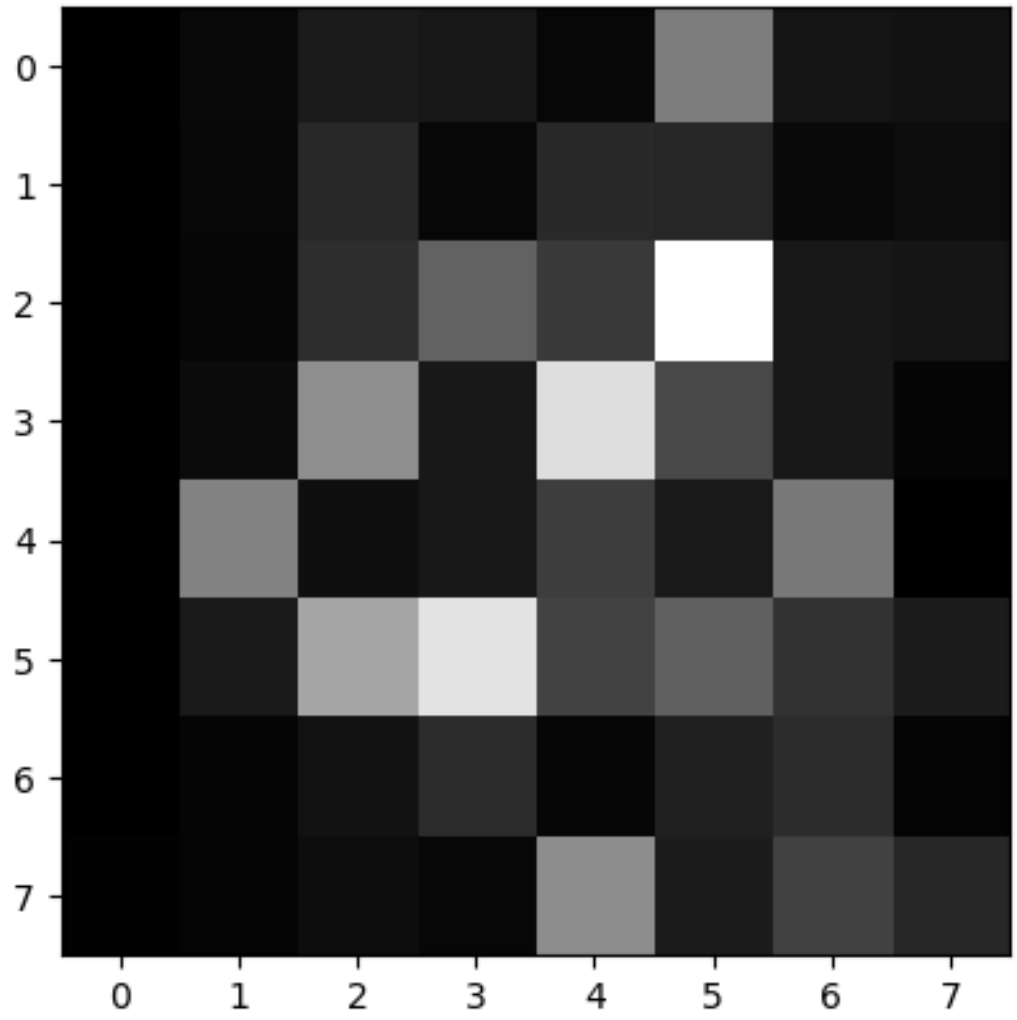


Figure 8: Feature Importances Image

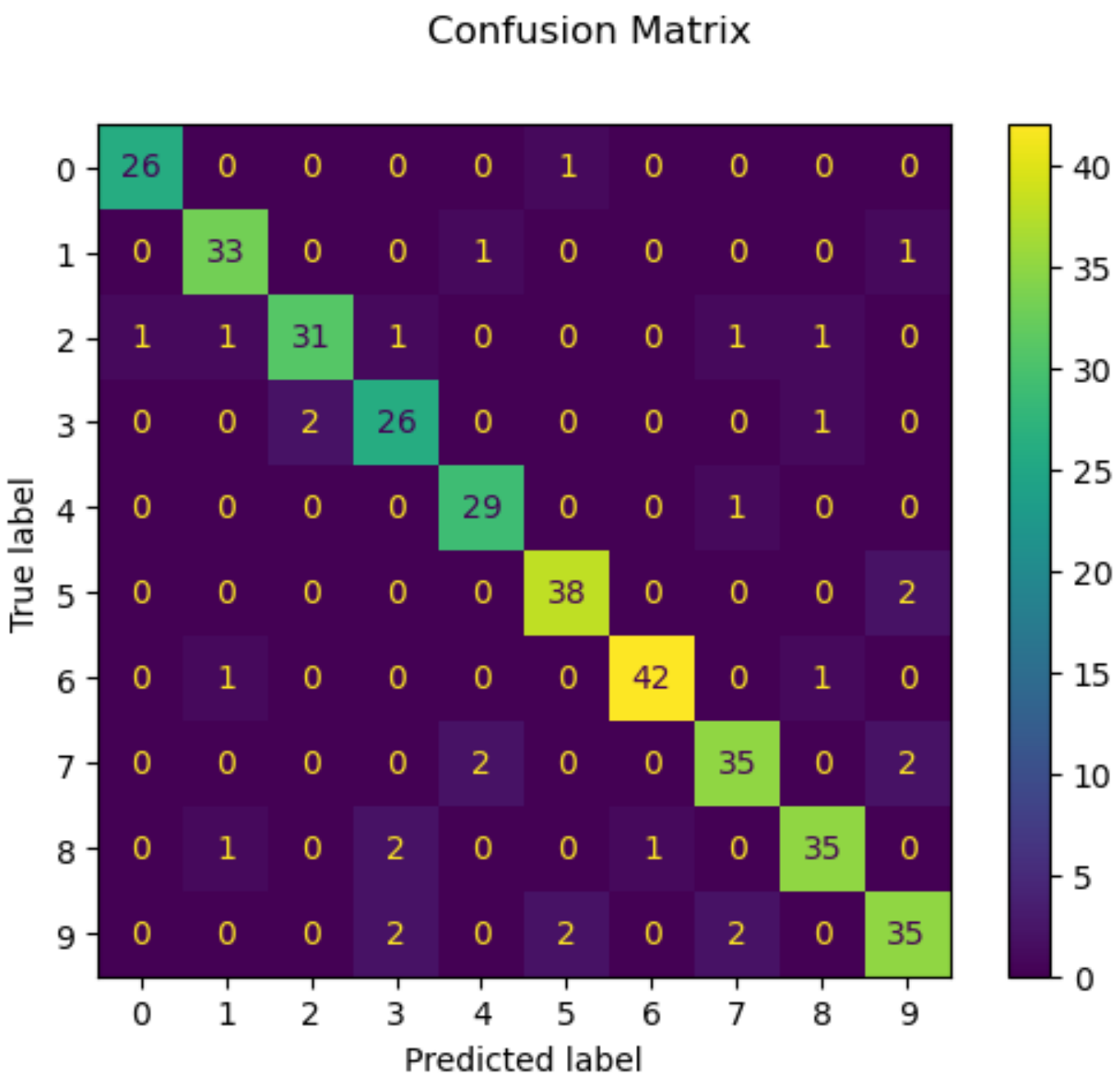


Figure 9: Confusion Matrix

Looking at the pixel at (0,5), we can consider the fact that this would only realistically be activated by a 1, 5, 6 or 7. In addition, if we look at the highly activated pixels in the center of the image, we can see that these would be activated differently by 0, 4, 5, and 8 than the other numbers. With all of this in mind, we now have a vague idea of different regions of activation that could be connected to what number is being written.

5.4 d)

To begin identifying missed samples, I used a Confusion Matrix to get an idea of how many samples we missed and how common the errors were. We can see this in Figure 9. We can see that we performed quite well, with no combination having more than 2 samples missclassified in the same way. Moving on from this matrix since it wasn't directly requested in the exam, and I am tired, we arrive at the plot of one missed sample from each class, and the table that contains the neighbor information.

Looking at Figure 10, we can see some examples of missed samples sorted by their true value. Starting with 0, we can see that the left side and the bottom of the 0 are more activated than the upper

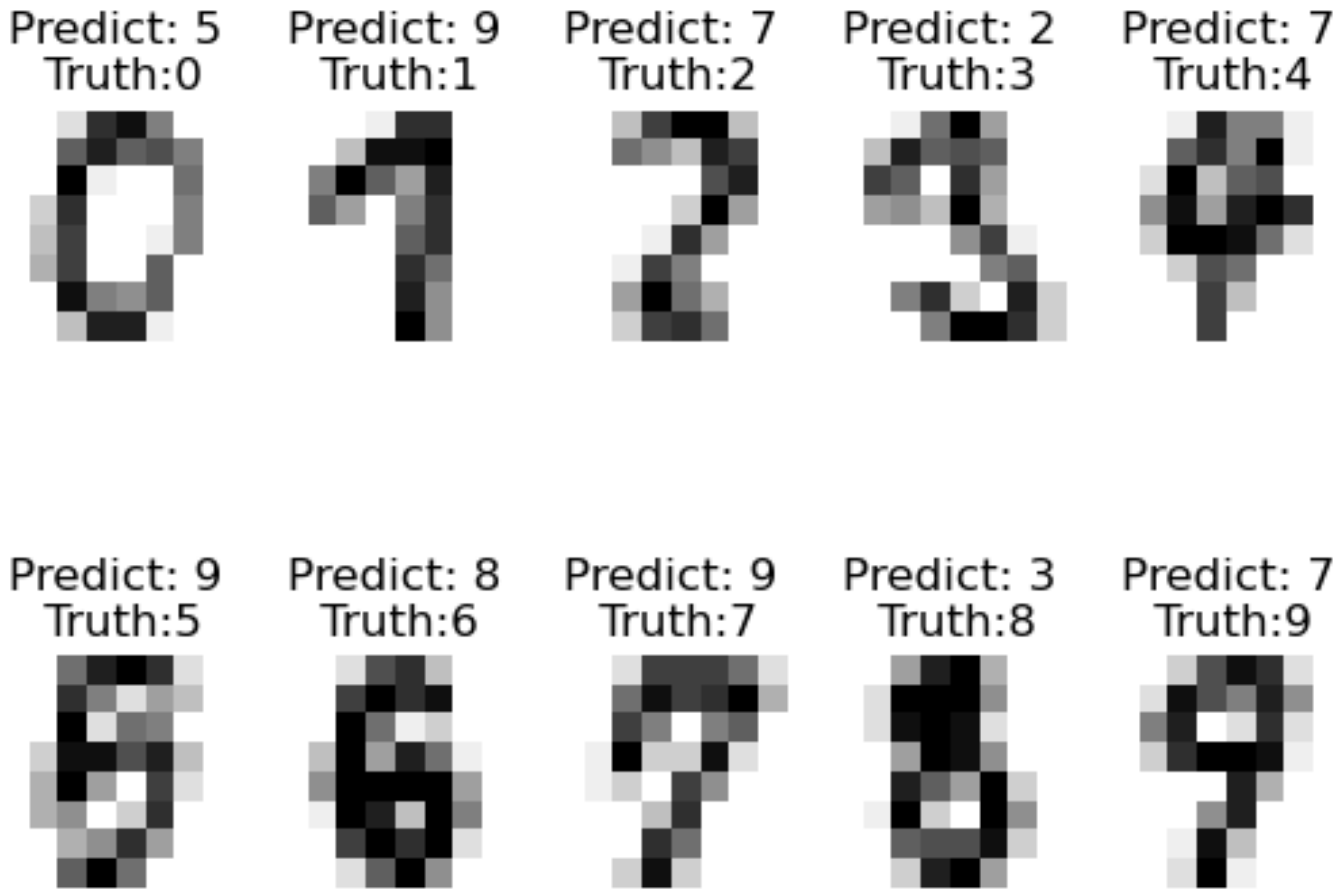


Figure 10: Missed Examples

truth	pred	idx	Neigbor#1_True Class	Neigbor#1_Distance	Neigbor#2_True Class	Neigbor#2_Distance	Neigbor#3_True Class	Neigbor#3_Distance	Neighbor_average_Distance
0	5	307	0	0.5900	0	0.6000	0	0.6000	0.5977
1	9	213	1	0.5900	1	0.6500	1	0.6600	0.6301
2	7	66	3	0.7300	7	0.7900	7	0.8000	0.7728
3	2	185	3	0.4300	9	0.6900	3	0.7000	0.6083
4	7	315	4	0.6700	7	0.7800	7	0.7900	0.7459
5	9	84	5	0.8400	5	0.8600	5	0.8600	0.8547
6	8	47	6	0.6300	6	0.7100	6	0.7500	0.6971
7	9	21	7	0.3500	7	0.4300	7	0.5100	0.4286
8	3	122	3	0.8500	6	0.8600	3	0.9200	0.8767
9	7	115	9	0.5300	9	0.5400	9	0.6200	0.5629

Table 3: Neighbors of missed samples

right side of the 0, which is likely why it was predicted as a 5. Looking at the top 3 neighbors, we see that actually all 3 of them are the correct class. I think this is evidence of a weakness in the model.

For number 1, we can see that this 1 had a very long upper hat that was primarily spread out on a diagonal. This can lead the result to look like a 9 with a small loop. Overall, I'd say this should have been more likely predicted as a 7 in my opinion, but I can see how a 9 would arise. Much like the 0, this also had the top 3 neighbors labelled correctly. Another evidence of a weakness in the model.

For number 2, This was missclassified as a 7. I can totally see why. I could be convinced that was a 7 with a mistake myself. Regardless, it definitely doesn't look like a 2 very clearly to me. The neighbors seemed to agree, with the top neighbor appearing as 3, and the next 2 appearing as a 7. I'd say that this is a good example of a bad sample that's understandable to miss.

For number 3, I think that this is a terrible 3. This does not look like much that's understandable, and personally I count it as a win since it shows errors exactly how I predicted people would make mistakes in section 5a. Looking at the neighbors, we can see it did have a pretty close neighbor that was correct, but that it also shared a 9. Even though the neighbors seem to be doing alright, I personally would say that this sample is uncharacteristic of most 3's and is an outlier.

For number 4, this looks awful. I wouldn't even consider this to be a number. Both the predicted value and the nearest neighbors agree that this looks like a 7, but I'd say that this is mostly because it

vaguely has a higher concentation at the top right going down to the bottom middle, which a 7 is the most likely to have.

For number 5, this sample looks quite noisy. We can see the 5 in there, but almost the whole grid is activated in some noticeable way. Admittedly, all of the top neighbors are correct, but they have the second largest average distance of any of the numbers, which supports that this is an outlier due to noise.

For number 6, I think that the model mistook it for an 8 because of the raw amount of activation. Given my normalization, these strong amount of activation would have been squashed which I can understand how it would lead to a missclassification of this sample. However, all of the nearest neighbors are correct, so I'd say that this one is likely a mistake due to my choice of normalization.

For number 7, I think this one is an examplee of the weakness of this model. The number 7 and number 9 have shown up the most consistently of any of the numbers so far, and it appears that the model struggled the most with number 7 and 9. As such, it makes sense that the model would struggle with confusing 7's and 9's (which we will see again at 9). This number most closely clusters with it's neighbors, and visually can be understood as a 7 with only moderate effort. I think this is an example of a weakness in the model.

For number 8, I can understand the mistake because there's no middle loop for the 8. That's a very important feature of the number that this sample doesn't have. You can see the problem in the neighbors where none of them are correct, and they have the largest average distance of any number.

Finally, for number 9, there's honestly no excuse for the model to get this wrong. This is clearly a clean 9, and all of the neighbors are correct. This is a strong evidence for the consistent problem of our model confusing 7's and 9's. I hate to end on such a strong negative point, but this is a problem that would have to be addressed if we were trying to push this model into production somewhere for some reason. Likely, the problem is that this 9 is too far to the right, where we can see in the feature importances image our model is largely ignoring.