MIDTERM CSE-546-50-4248: 10/19/2024

MICHAEL ENCINAS

Problem #1 (10 points)

Create a 2-dimensional data set with 20 samples that has the following properties:

- Samples should belong to 2 classes (10 samples per class)
- All samples can be classified correctly using Linear SVM with no regularization.
- Two samples from class 1 can be misclassified using Linear SVM with regularization.

Generate a scatter plot of your data. Use a different color/symbol for each class. **Indicate the 2 samples that cannot be classified correctly in (b). Explain why these points are classified correctly in (a) and incorrectly in (b).**

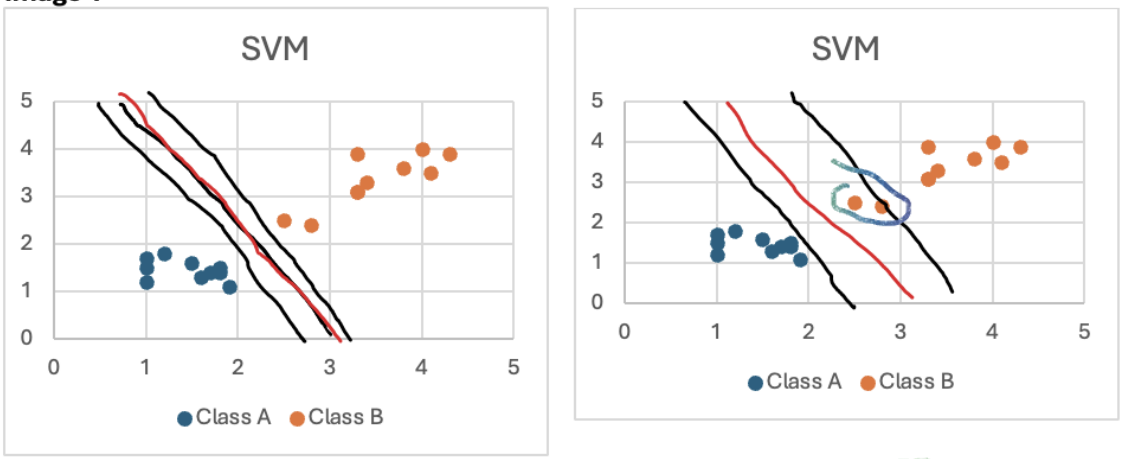
a) Provide a justification why the solution in (b) can be better than the one in (a)

b) Provide a justification why the solution in (a) can be better than the one in (b)

I created 20 samples, 10 in each class manually. Below are my samples and the class they belong to. I went ahead and instead of drawing my scatterplots in word.Image 1 shows my visualization of Linear SVM with no regularization and Linear SVM with regularization.

class_A =[1.2, 1.8], [1.5, 1.6], [1.8, 1.4], [1, 1.5], [1.6, 1.3],[1.9, 1.1], [1.0, 1.7], [1.7, 1.4], [1.0, 1.2], [1.8, 1.5]
class_B = [3.8, 3.6], [3.3, 3.1], [4.1, 3.5], [3.3, 3.9], [2.8, 2.4], [4.3, 3.9], [2.5, 2.5], [3.4, 3.3], [3.3, 3.1], [4, 4]

Image 1



No regularization

Regularization.

Misclassified Examples

2 two samples that cannot be classified correctly in (b)(Linear SVM with regularization) is (2.5,2.5) and (2.8, 2.4). The reason for this is that in (a)(Linear SVM with no Regularization) the Linear SVM tries to draw a line in the middle and classify all points correctly. In B the regularization is adjusted where it can have a bigger margin in which it could classify points wrong.

Solution (b) can be better than (a) because if you have new data points that tend to be more outliers or not all grouped up close to majority of points for one class than your classifier can have more of

a range to work with in order to plot a new point correctly in which if it's (a) it might be to strict and classify a new datapoint wrong.

Solution (a) could be better than (b) because it could be more accurate in predicting classes correctly since its separated more away from boundary which means it will have a less change of misclassifying compared to (b).

Problem #2 (10 points)

Create a 2-dimensional data set with 20 samples that has the following properties

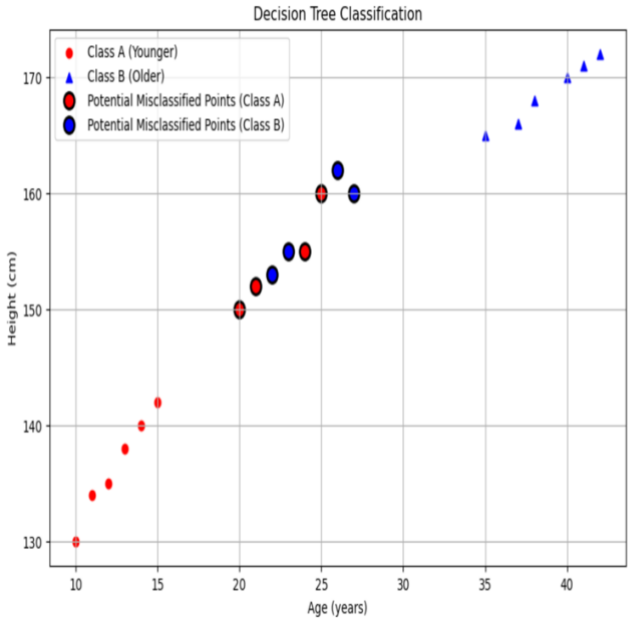
a) Samples should belong to 2 classes (10 samples per class)
b) NOT All samples can be correctly classified using a decision tree classifier with ***depth <= 3***
c) All samples can be correctly classified using a decision tree classifier with ***depth > 3***

If it is not possible to generate such data, explain why. Otherwise, generate a scatter plot of your data using a different color/symbol for each class. **Indicate the samples that can/cannot be classified correctly for each case.**

For each case, display your decision tree indicating the **feature/threshold used at each non-leaf node and the number of samples at each leaf node.**

Note: This data should be generated manually and you do not need to run any code on it. Also, the feature/threshold should be estimated by visual inspection of the data and not computed based on information gain.

I will be doing Age as X and Height as Y. (Age, Height) for each dataset.
Image was created to visualize these datapoints I created with classes established and the datapoints that cannot be classified correctly. Image 2.

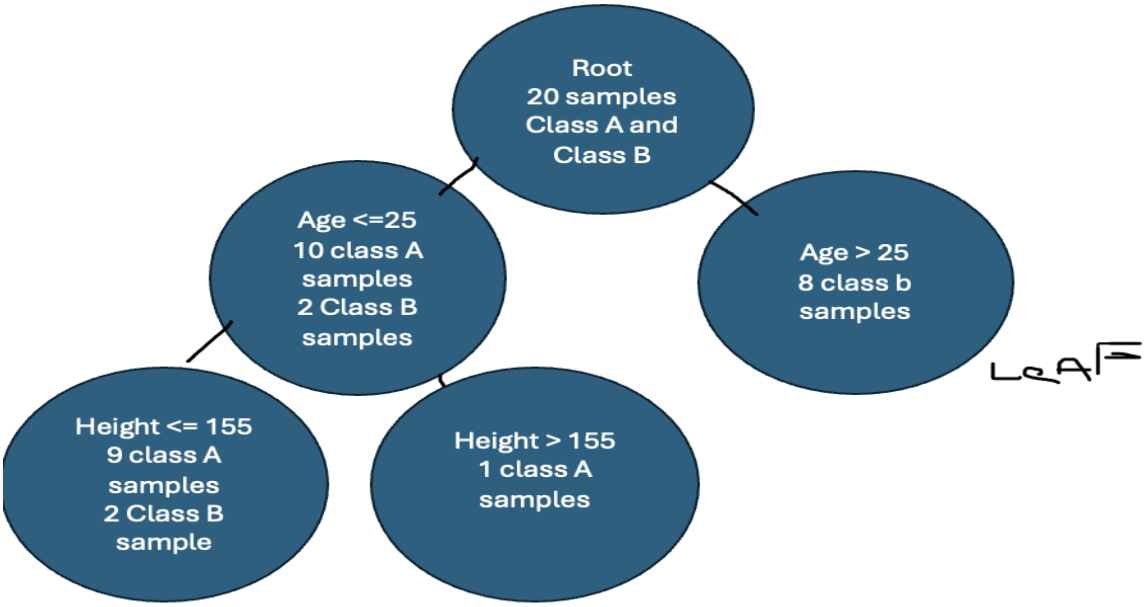


Class A (Younger People) **Image 2**

(10, 130)
(12, 135)
(14, 140)
(13, 138)
(11, 134)
(15, 142)
(Harder to classify):
(20, 150)
(21, 152)
(25, 160)
(24, 155)

Class B (Older People)
(35, 165)
(40, 170)
(38, 168)
(42, 172)
(37, 166)
(41, 171)
(Harder to classify):
(22, 153)
(23, 155)
(26, 162)
(27, 160)

Depth <= 3
1st split is going to be = 25 in age, so left child node is less than or equal to 25 and right child node is greater than 25.
2nd split is making left child split on height = 155 cm . Can't go further now because I will still not be able to classify all classes correctly with just three levels of depth. Even if I used age is less than or equal to 22 in age for far left bottom node it will not complete everything

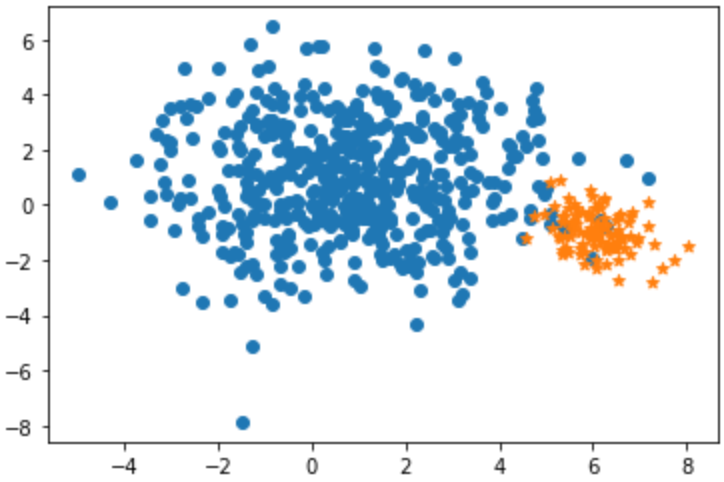


Continue on from last but use depth greater than 3.
Then my split is at age is 24. Next will be Height on right child node at height being less than 153.
This will give me depth of 4 and geive me pure answers.

Depth > 3

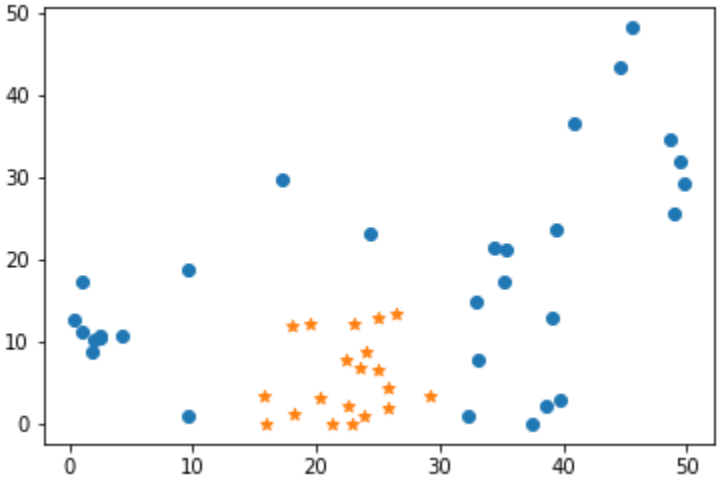
Problem #3 (6 points)

1. Given the following 2-Dim data set with 2 classes. Out of all classifiers that we have covered in class, which one is the best candidate for this data? **Justify your choice**.



For Problem 3 I would have said KNN but it's too close to blue, especially blue in under orange in that big orange group. I would say Kernal SVM would be the best candidate for this data. I need to have a curve boundary since linear boundaries would not work here do to overlaps or not clear space margins. RBF Kernal would allow for me to have a curved boundary. Kernal SVM also allows me to change the dimensionality so I can reach all orange and separate blue class. I have a lot of parameters to adjust like gamma and c that could give me smaller margins to work with.

2. Given the following 2-Dim data set with 2 classes. Out of all classifiers that we have covered in class, which one is the best and simplest candidate for this data? **Justify your choice**.



I would use Decision tree for this dataset as the simplest candidate for this data. Since orange is cluster together in a small group it would be easier to do a decision tree to be able to find both classes here. We can have a branch from the tree going out to blue group and then orange group would have its own branch off. I do not see any close blue datapoints next to orange. I do not think any Logistic regression, KNN, etc. would be able to satisfy the unique classifications of the datapoints on this scatter plot.

Problem #4 (9 points)
List three different strategies that can be used to reduce overfitting. Describe each strategy and explain when it can be used and why it can reduce overfitting.

If you have overfitting, then that tells me that my model is to complex so I would try to simplify my model. I must make sure it's not to simply where is underfitting but where I can adjust and get good generalization. I would adjust my variation of inputs since my data could be too large or too complex while making sure my data points are not to similar. For example, In KNN if I'm training my model and I have overfitting at K = 1 then I adjust my K to go up to 2-5 to get my model to be not to complex and more open to multiple neighbors.

Another strategy to reduce overfitting is regularization. Regularization is when we adjust parameters for a model to make it not so complex. Regularization also makes sure that not one feature or specific noisy datapoints are contributed too much to a model. For example, alpha would be a way to reduce overfitting. We have L1(Lasso) and L2(Ridge) which are to regression types. The larger the alpha it forces coefficients to move towards zero, decreases training set performance and may improve generalization.

Another strategy to reduce overfitting is making your depth less on decision trees. Depth is the depth going down from root that gives you're the ability to continue to make splits at specific feature constraints that you place down for child nodes. The more depth you go down the more options you have to obtain pure leaves in which will increase your training accuracy. If you limit the depth, limit the maximum number of leaves and require a minimum number of points in a node to keep splitting it then you will be able to reduce overfitting in decision tree models.

Problem #5 (65 points)
For this problem, you need to use the built-in sklearn ***digits*** dataset. You can load this data using

```
from sklearn.datasets import load_digits
digits = load_digits()
```

(data will be stored in ***digits.data*** and desired output will be stored in **digits.target**)

Divide the data into training and test sets using `train_test_split` and `random_state=0`

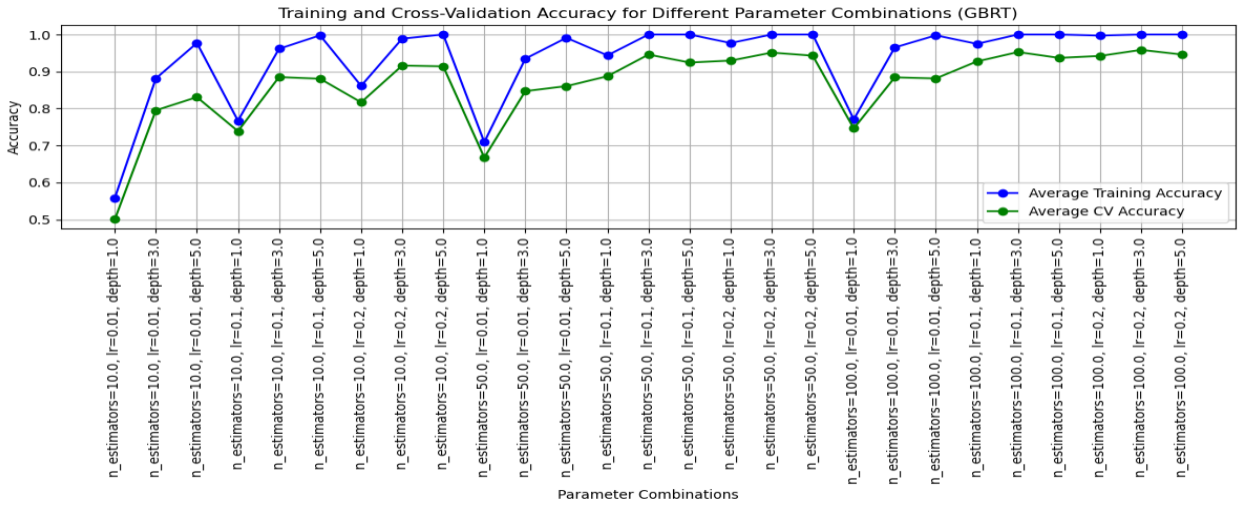
1. Train a ***Gradient Boosted Regression Trees*** classifier and optimize its performance on this data. Using *4-fold cross validation*, design your experiment to learn the best values for the following parameters using *4-fold cross validation*: ***n_estimators, learning_rates***, and ***max_depth***
    a. Analyze the results of the classifier using its optimal parameters and comment on its generalization capability by comparing the accuracy on the training, validation, and test data.
    b. Identify the number of misclassified test samples from each class. Explain why some classes are easier to classifier than others.
    c. **Visualize** and **explain** the relevant features identified by the classifier

Create a white 8x8 image that represents the original 64 features. Map each identified relevant feature to this 2D image and display it using a color or a grey scale that reflects its importance (*as illustrated in the lecture*)

Gradient Boosted Regression Trees

The parameters I chose to run is N_estimators (10,50, 100), Learning_rates (.01, .1, .2) and Max_depths (1,3,5) for my GBT. 4 K-Fold was used for CV accuracy scores which are obtained by average cv scores. Displayed will be average training scores, average cv scores, and time taken with its parameter. Image 1 and Image 2 will visualize this.

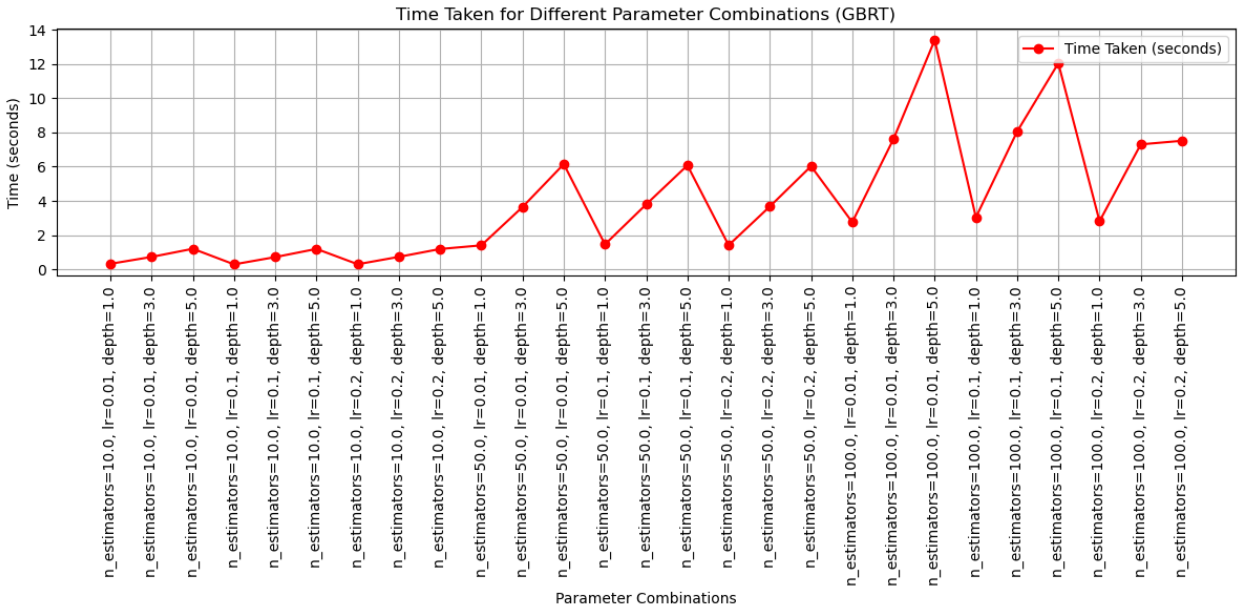
Image 1



I have underfitting when depth is 1 if learning rate is low at .01, I tend to get better generalization scores when n_estimator is increased. Learning rate seems to place a big factor in my scores when increased. When depth is increased, I get better training scores but that also leads to overfitting since my training scores are way higher than my cv accuracy scores.

My model here tends to overfit for many of my parameters here. Just looking at my results I will not use N_estimator equals to 100. At N_estimator = 50 I get overfitting on almost all my data except when N_estimator (50), LR (.1) Depth(1) with scores of average training 0.943578, average cv training 0.887176 with time taken at 1.463538. I will take this score to test my model since N_estimator = 10 parameter is either overfitting or underfitting to the extremes.

Image 2

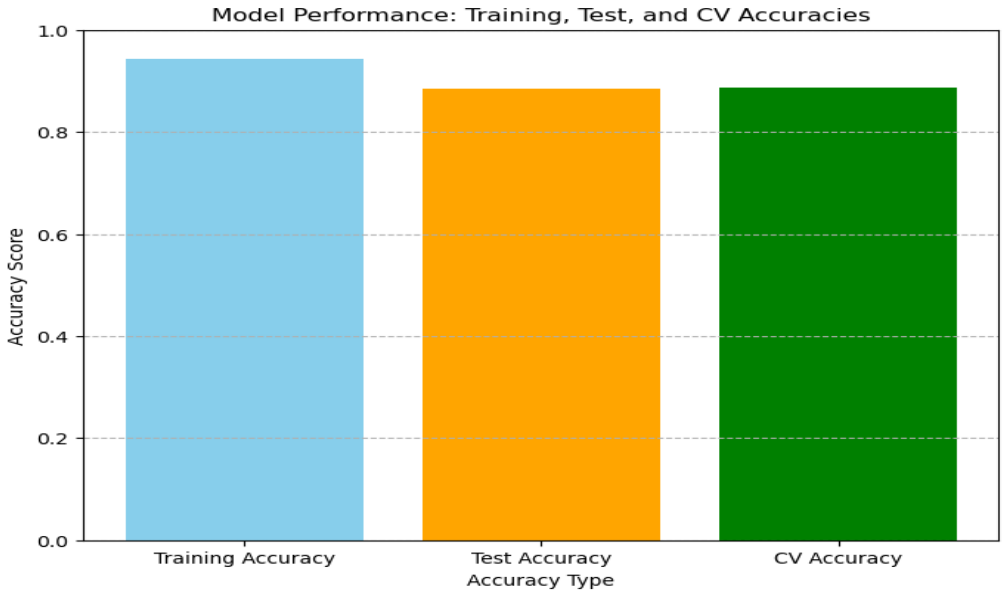


In Image 2, when N_estimator is low (10) than time taken is a lot faster, however, when N_estimator (50,100) is increased time is slower, however, when depth is low at 1 then time between three estimator slower than each other but not in comparison to when depth is increased to 3 or 5.

Testing

I used the parameters of Depth 1, N_estimator 50 and learning rate of .1 to train my GBT model fully with Train data. I then tested my test data on my model that was trained to see what its training accuracy and testing accuracy was.

Image 3 is output of my results.

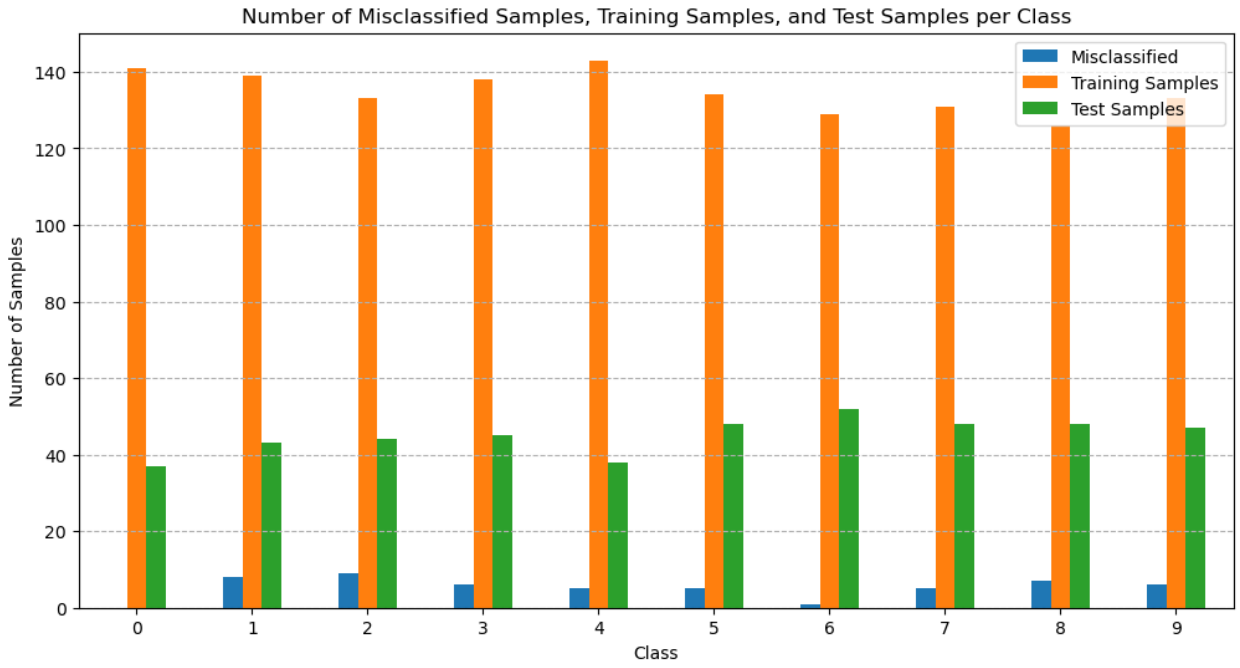


With the chosen parameters I selected to train and test my model I get the same accuracy scores for my training which was approximately 94 % and a test accuracy score of 88 %. My

CV score on this parameter was 88%. So, my model stayed consistent when I trained it fully and used it on unseen new data that was tested. CV scores and testing scores were the same percentage and so was my training score with its previous training score. GBT appears to do well in this dataset on CV accuracy results that tend to show real results when testing on new unseen data.

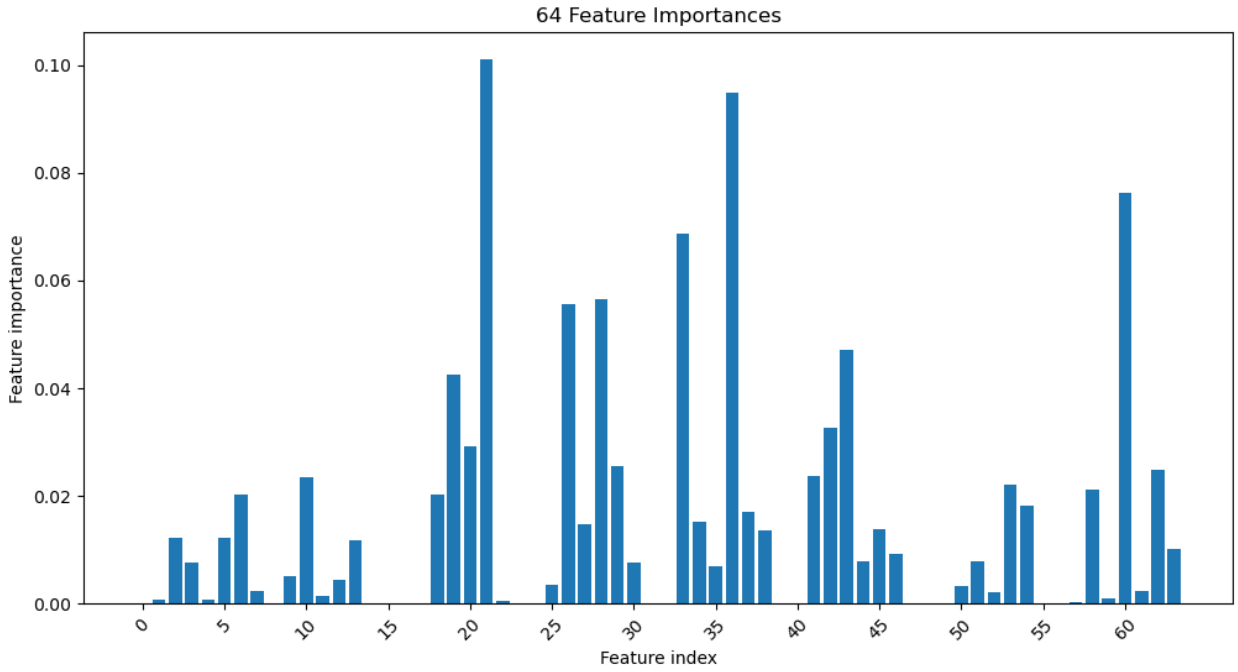
B) I outputted my misclassified test samples from each class. I wanted to see if training samples and testing samples were equal just to make sure no bias was involved that can cause misclassification and based on Image 4, everything is close to each other and not bias. I did not have to many test samples classified wrong in comparison to the amount of data I was given. I would say Class 0 and Class 6 are easy to identify just based on their shape being very distinguishable. 1 and 2 might be harder to distinguish because it could look like 7 or 3. 6 might look like 8 or 9.

Image 4



I then look at feature importance from my trained model. Image 5.

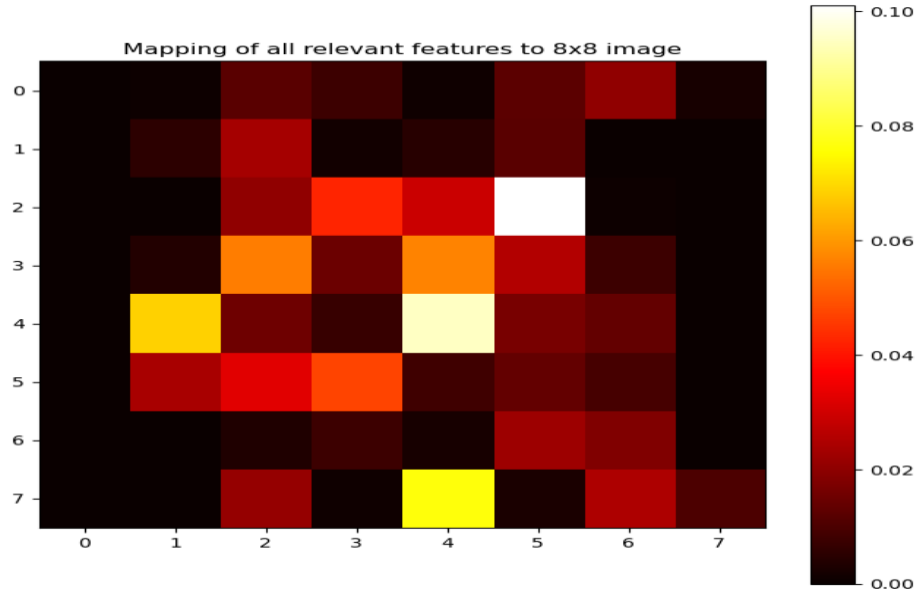
Image 5



Feature importance tends to be more towards the middle on this index which Makes a lot more sense because when I print out the heat map. The important features tend to group up in the middle which makes sense because numbers can end or have signature marks in the middle of its image. 64 features are just the image of the number.

Image 6 below shows how the most importance tends to be in the middle or at marks where the numbers end.

Image 6



2. *Train a* **Support Vector Machines classifier** and optimize its performance on this data.

Using *4-fold cross validation*, design your experiment to learn the best values for the following parameters

i. Data normalization: ***no preproce***ssing vs. ***StandardScaler***
ii. Kernel: **Linear** vs. **RBF**
iii. The regularization parameter **C**: 4 different values.

a. Analyze the results of the classifier using its optimal parameters and comment on its generalization capability by comparing the accuracy on the training, validation, and test data.

b. Identify the number of misclassified test samples from each class.

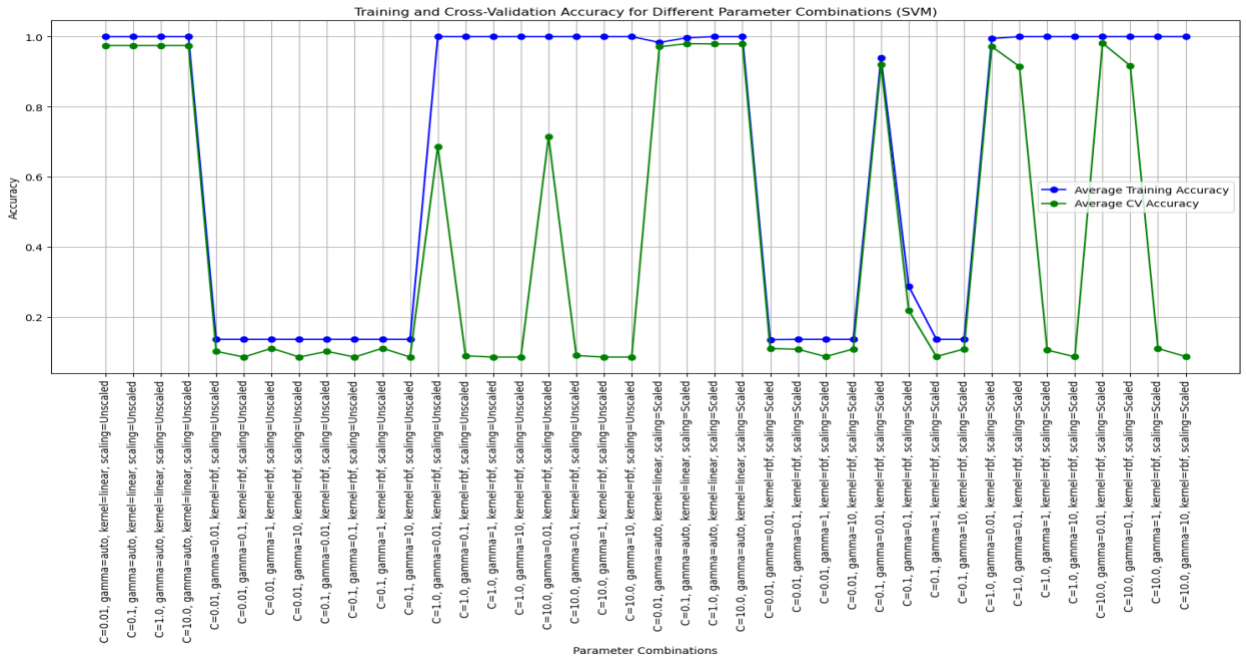
Support Vector Machine

I will be using 4 k-fold validation when I'm training my model to find the best parameters. The parameters I am choosing to run are the C values of .01, .1, 1, 10, Gamma values of .01, .1, 1, and 10 with the kernels of Linear and RBF. I will first show unscaled data and then I will show standard scaler.

Image 7 will output all unique parameters with its average training accuracy scores and cross validation scores.

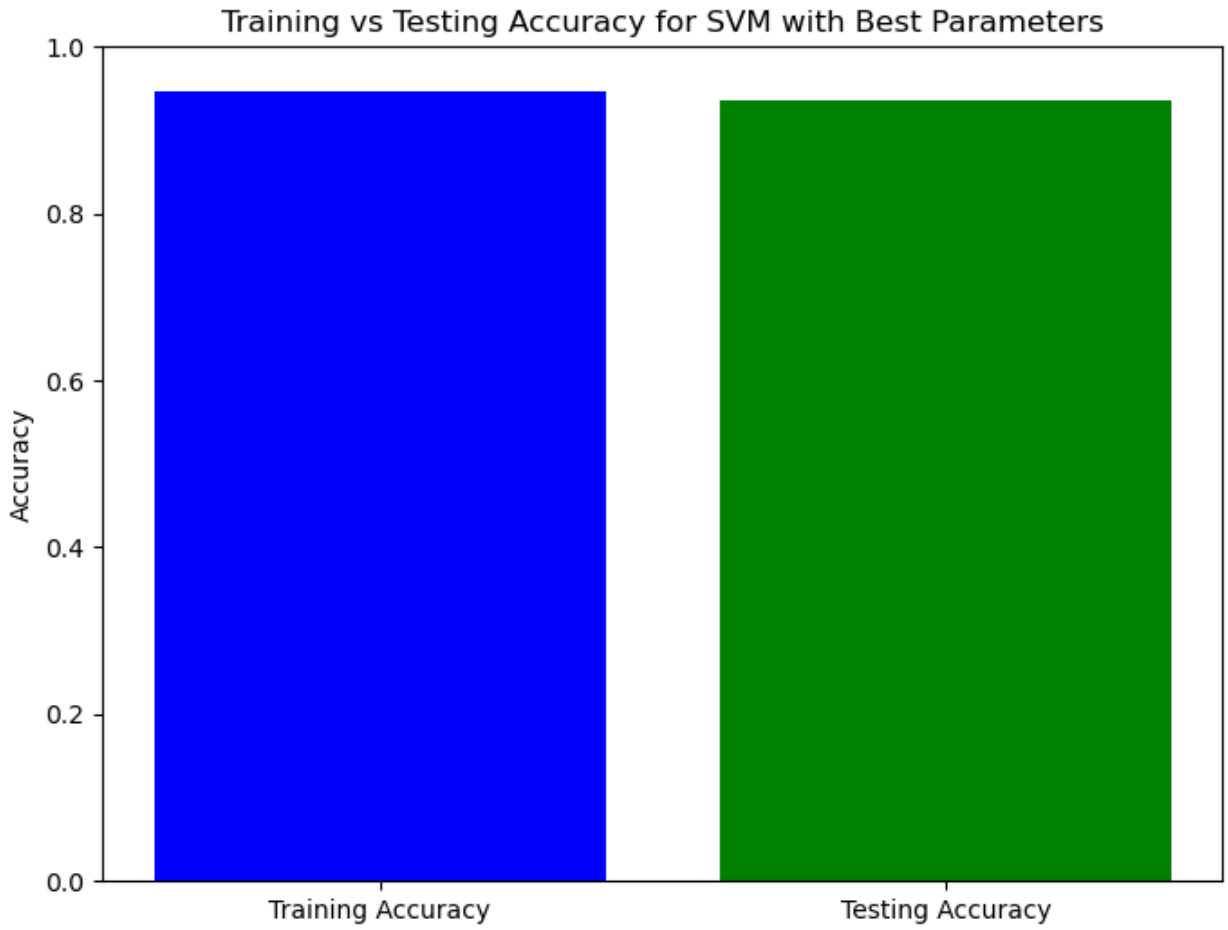
In image 7 I get overfitting on all unscaled linear kernel. I get overfitting when kernel is switched to RBF for unscaled and C is set to .01, .1 and 1. When I up c to 1 with gamma at .01 for RBF Unscaled then I see my scores for training and cv increases. However, it's now overfitting. This overfitting continues on for the rest of my entire unscaled data. When I scale my data for linear kernel I continue to get overfitting. High training scores and High cv accuracy scores. When I introduce RBF Kernel for scaled, I get underfitting when C = .01. I get the best training accuracy and cv accuracy generalization when I used scaled data with the parameters of c .1, gamma .01 and RBF kernel. Training accuracy 0.940115 CV accuracy 0.920537. After this parameter the rest of my data overfits. I will choose this parameter to test my model.

Image 7



Testing

Image 8



I got training accuracy scores of .9465 and testing accuracy score of .9356 with my model being trained fully on my selected best parameter. I have good generalization scores and SVM is faster than GBRT.

I checked to see what samples I misclassified with my data points, and I got a printout of

Class 0: 0 misclassified samples

Class 1: 1 misclassified samples

Class 2: 3 misclassified samples

Class 3: 8 misclassified samples

Class 4: 1 misclassified samples

Class 5: 2 misclassified samples

Class 6: 2 misclassified samples

Class 7: 1 misclassified samples

Class 8: 6 misclassified samples

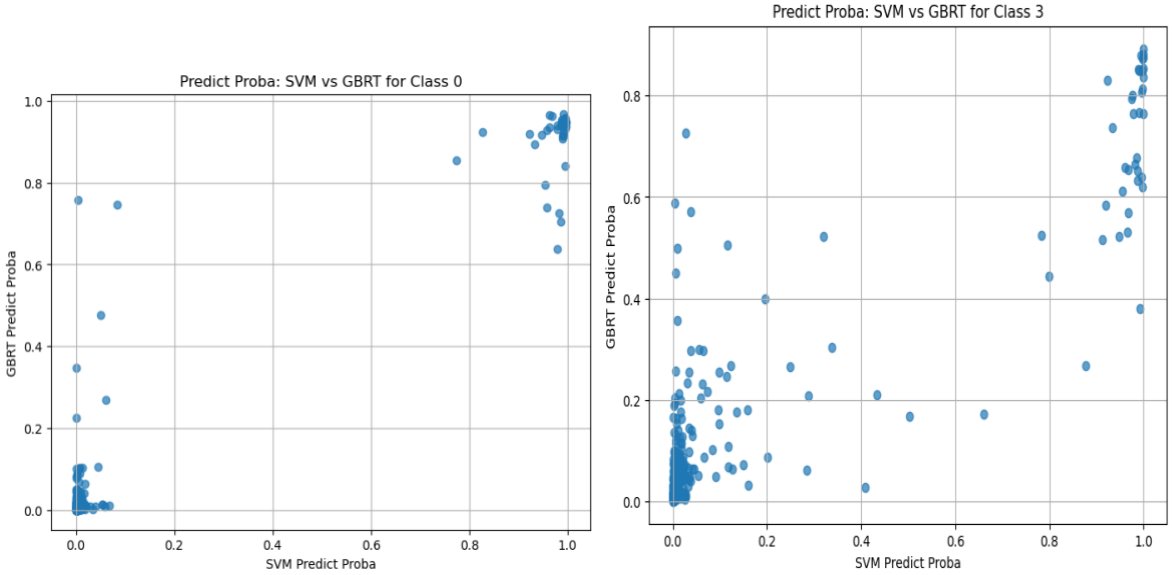
Class 9: 5 misclassified samples

When looking at my misclassified samples, I noticed that 3, 6 and 5 have the largest missed datapoints. I wonder if because of the loop of 3 not being complete when compared to 8 it gets confused and switches them up. Same thing goes for 9 which is 8 if you close it off. I believe it's possible it gets those small features wrong. I still have a high percentage of correctly guessed predictions, but I would believe that's why it might miss some samples.

3. Analyze the correlation between the output of the 2 classifiers by displaying the *predict_proba* of SVM vs. *predict_proba* of GBRT (using test data). Using these scatter plots (only for classes "0", "3", and "7"), identify (if available) the following 3 groups
   c. G-1: Samples that are easy to classify correctly by the SVM, but hard to classify by GBRT
   d. G-2: Samples that are easy to classify correctly by the GBRT, but hard to classify by SVM
   e. G-3: Samples that are hard to classify correctly by both methods

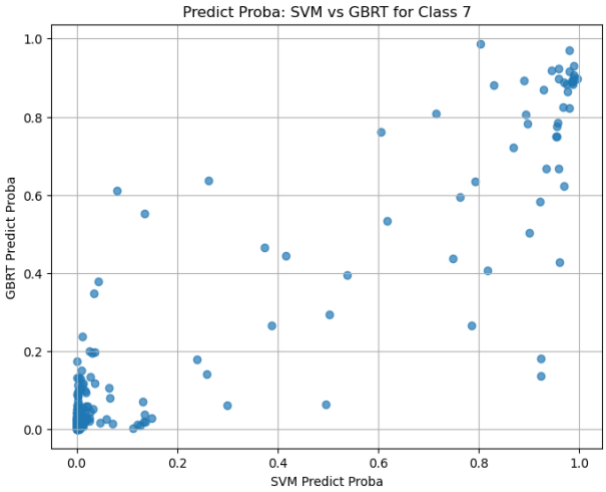
For each group, display few samples (as images) and identify any common features among them.

Image 9



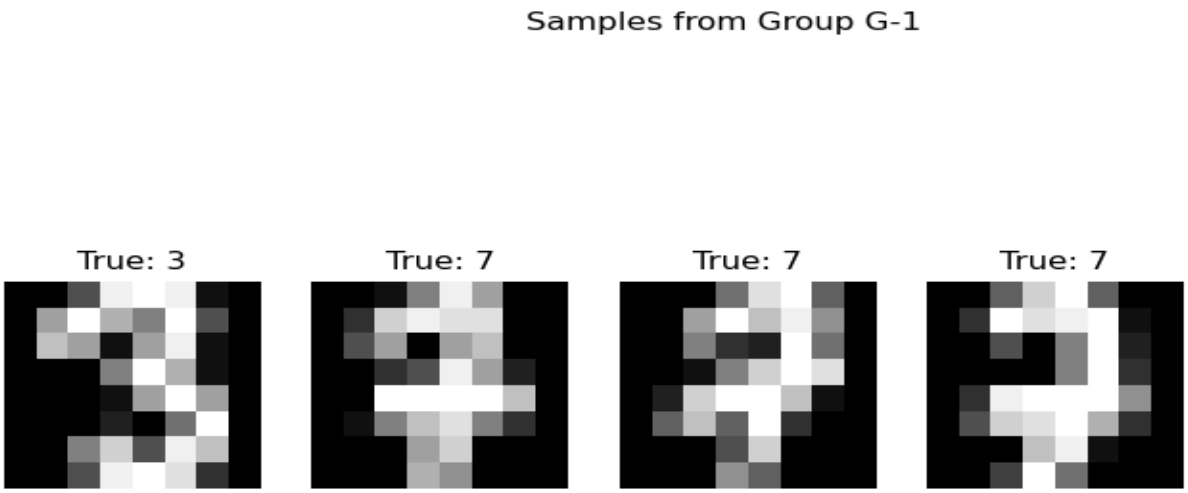
Class 0 it appears get the same wrong class 0 predictions and the same right class 0 predictions. SVM does appear to not predict as correctly compared to GBRT wen SVM has prediction (Confidence) value of .1 in which GBRT is better to sale up at .2 to .7. GBRT appears to be more confident when SVM prediction confidence is lower. Might be more robust to handle data.

For Class 3 SVM prediction confidence when its confidence is low below .2 it appears GBRT is at .2 to .6 which shows that's its more confident in his predictions when SVM is predicting confidence is low. However, when it comes to have good confidence at .9 and above for SVM, it appears GBRT stays at around .5 to .9. So, when SVM is confidence on its prediction probability it's a lot more sure about it then compared to GBRT.



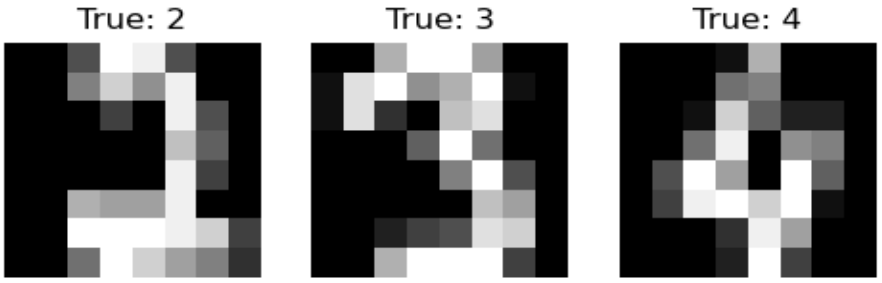
For class 7 SWM and GBRT tend to go back and forth in comparison when one is more confident than another. I don't see any huge differences when looking at this data since both of them have a wide scale at Class 7.

a. G-1: Samples that are easy to classify correctly by the SVM, but hard to classify by GBRT
b. G-2: Samples that are easy to classify correctly by the GBRT, but hard to classify by SVM
c. G-3: Samples that are hard to classify correctly by both methods



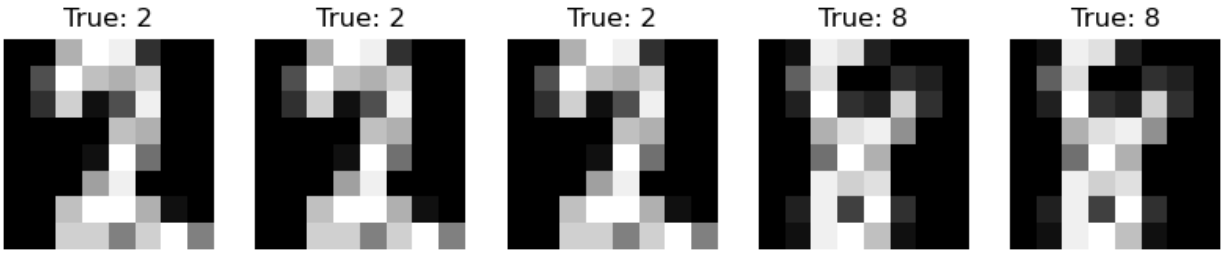
SVM can distinguish images that are similar in appearance which based on these images are 3 and 7 compared to GBRT which is having a hard time with the middle split between 3. This might be because of the way SVM is able to learn and operate more complex designs compared to a tree.

Samples from Group G-2



GBRT can distinguish better these numbers than SVM might be because GBRT is more robust and simpler compared to GBRT. The numbers above have wide gaps of dark which might confuse SVM. Simple images appear to benefit GBRT a lot better than SVM. However, from Class 0, SVM appears to handle more complex images better.

Samples from Group G-3



Both models are unable to identify Class 2 and Class 8 because the pixels from these images are hard to read or distinguish between what the correct number is. The pattern is blurry, and the numbers don't have full clarity between what sets it apart from other numbers. Even the black space between the top of 8 does not show it fully connect.

Link: https://youtu.be/z4XoTeGXP6w