Intro

I will be applying Kernal SVM and MLP classification to my MNIST dataset. I will go into further detail later into my fashion-MNIST dataset. I will be using k-fold cross validation on my training data while trying to find the best optimal parameters. I also will be rescaling my data to see if I am able to get better training accuracy and cv accuracy by trying these three methods to my scaling: No preprocessing, Standard Scaling and MinMax Scaling. During this process i wil use different parameters in order to make a final decision on what parameters I will run on my test dataset.

Once I get my final parameters for my Kernal SVM and MLP then I will train my models and test my model on the new test dataset. I will be checking training accuracy, testing accuracy, train/test time and make a final analysis on which model generalized better. I will be explaining during the entire process where overfitting or underfitting are occurring.

I then will analyze the correlation between the output of the 2 classifiers which is introduced by predict_proba of SVM and MLP (Using Test Data).

Mini-Fashion-MNIST

This dataset has 10000 samples in the training dataset and 5000 samples in the testing dataset. Each sample consist of 28x28 pixel images which has a pixel intensity between (0-255). The output target variables are going to be 0:T-Shift/top, 1:Trouser, 2:Pullover, 3:Dress, 4:Coat. The data is balanced at 2000 samples per class in the training dataset.

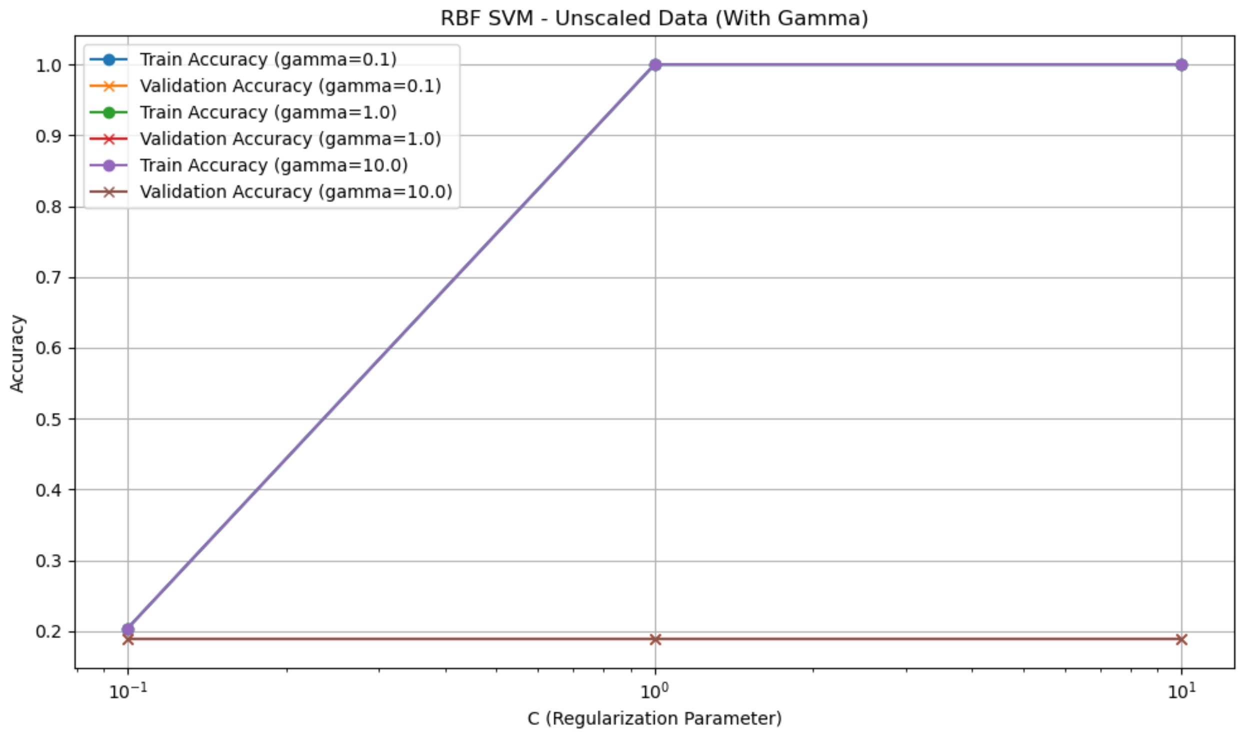
Kernal SVM

I will be varying my C (.01, .1, 1, 10), gamma (.01, .1, 1, 10) and kernal ('Poly', 'RBF'). At first it will be unscaled data and then I will implement StandardScaler and MinMaxScaler. K-Fold is going to be split at 4 (4 K-Fold).

C = .1, 1, 10 - gamma = .1, 10,10 RBF Kernal Unscaled
C = .01, .1, 1, 10 - gamma = .01, .1, 1.0 Poly Kernal Unscaled

Figure 1 will show visualization of RBF Unscaled with the different parameters above and Figure 2 will show visualization of Poly Unscaled with different parameters as well.

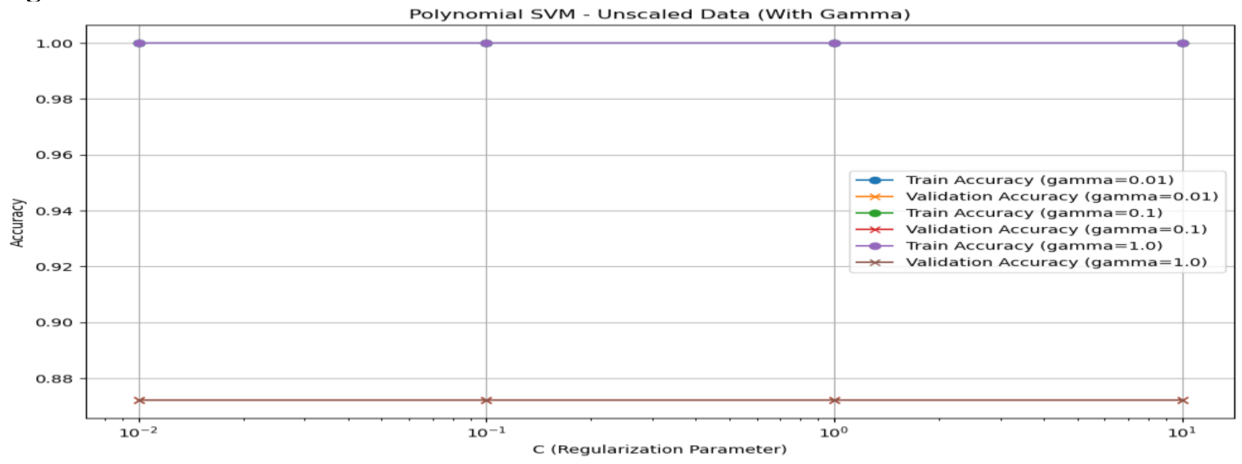
Figure 1



For Kernal SVM with RBF as kernal (unscaled) my game of .1 and 1 were underfitting with train and cv accuracy that was underfitting at around .20 percent to .18 consistently. When I gave my gamma at 10, I get overfitting when C is 1.0 and 10. I do not have good generalization

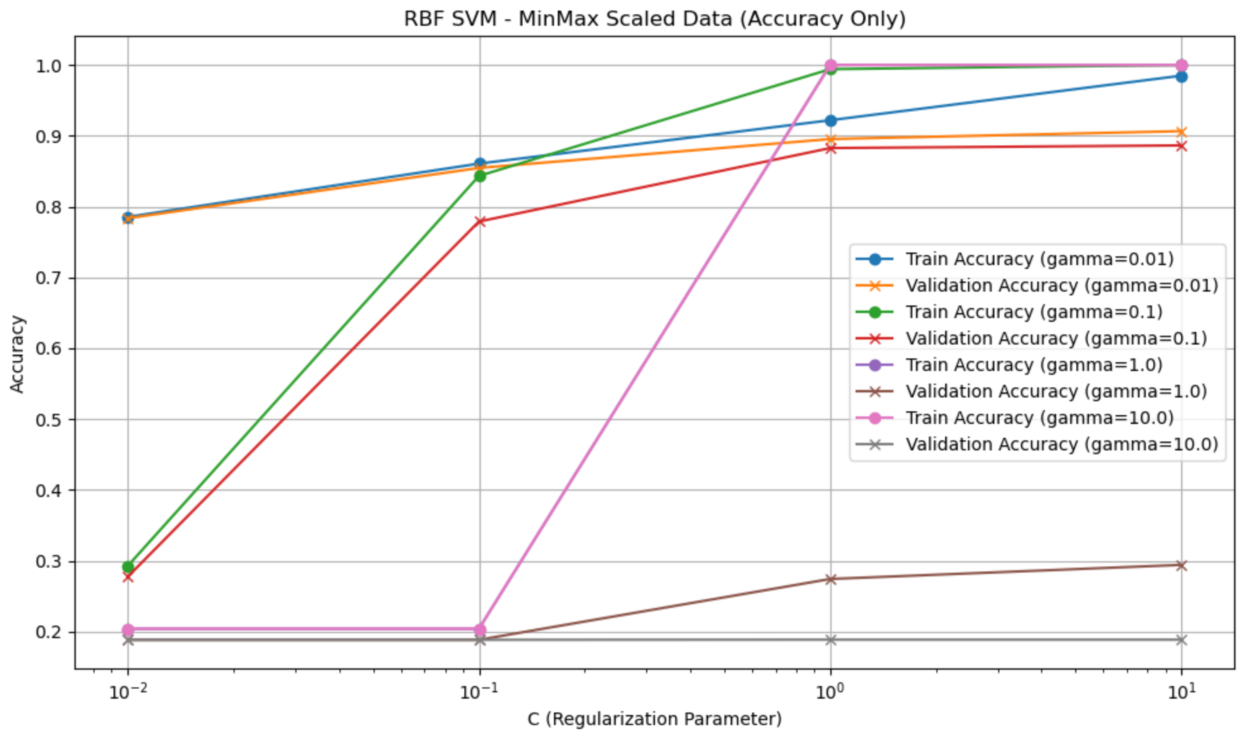
When observing Figure 2 below I have overfitting on all parameters. With training scores of 1.0 and cv average accuracy scores of .8721.

Figure 2



Now I will be trying the different scale systems of MinMax and StandardScaler.

Figure 3



In figure 3 I get a lot better generalization scores when gamma is .01 with scores that are in the high 70s for training and cv accuracy. The gamma with .01 improves when C is increased from .01 to .1 and then 1. When C is at 10 for Gamma at .01 its overfitting. Underfittting occurs at .01 for C when gamma is .01 as well. My accuracy scores just by looking at them appeared to dramastically improve since scaling my data to MinMax. For example for Gamma = .01and C is 1 I get high generation score of .9219 for training accuary and a cv average score of .8953. So far when gamma is low at .01 my scores appear to do a lot better than the other gamma parameters. When my games are 1 and 10 they tend to underfit a lot more unless gamma at .1 has a high c of 1 or 10 then its overfitting with high values

Figure 4

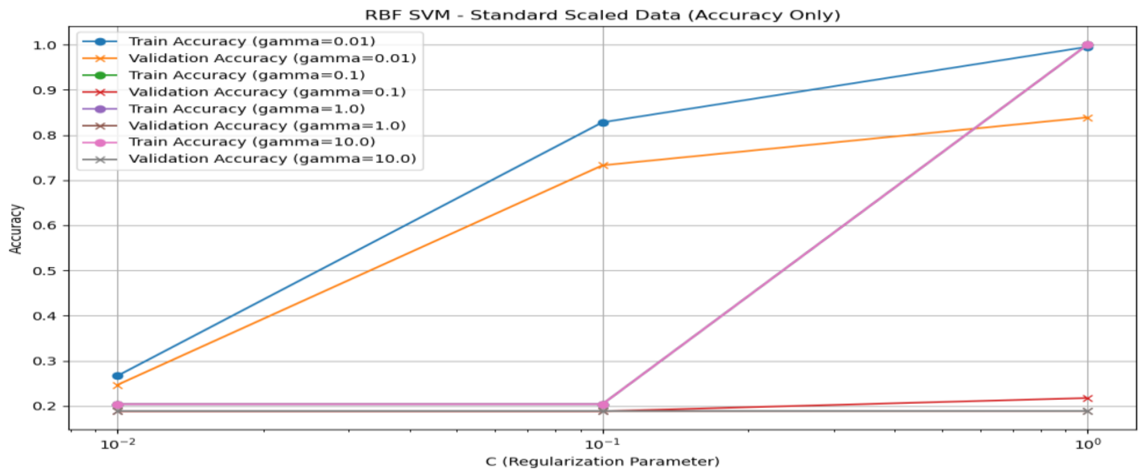
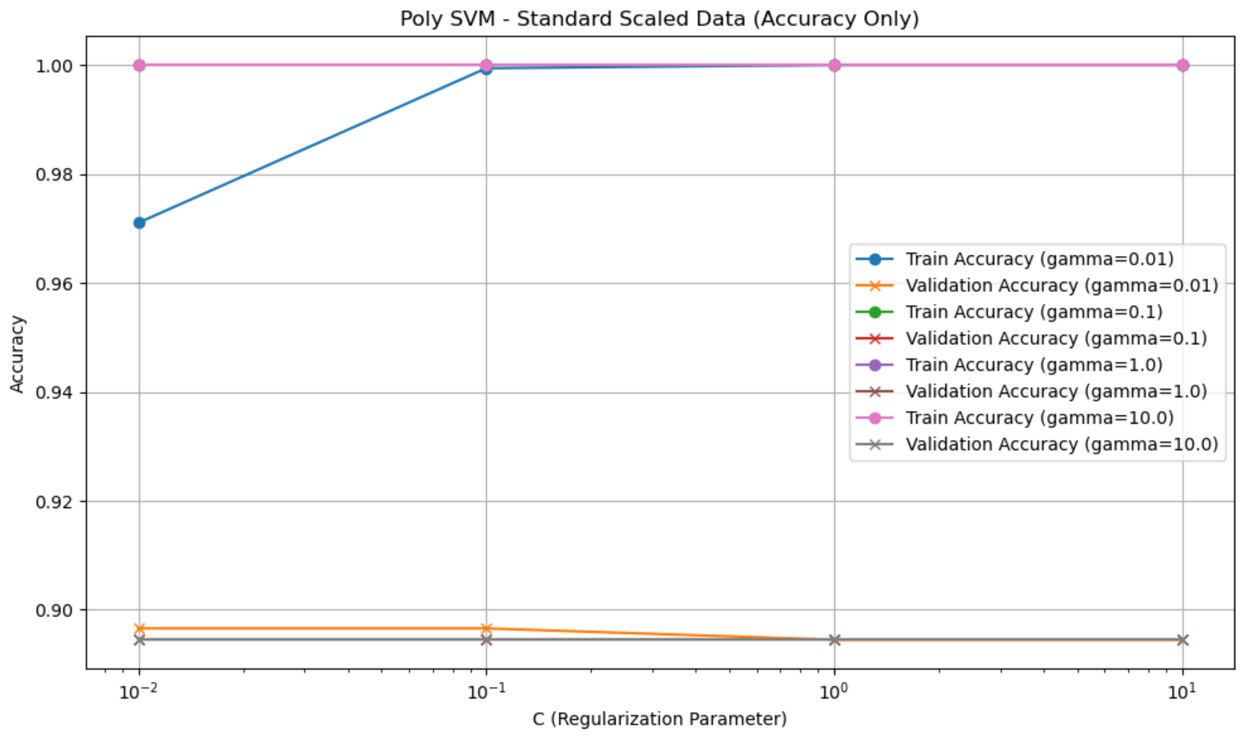


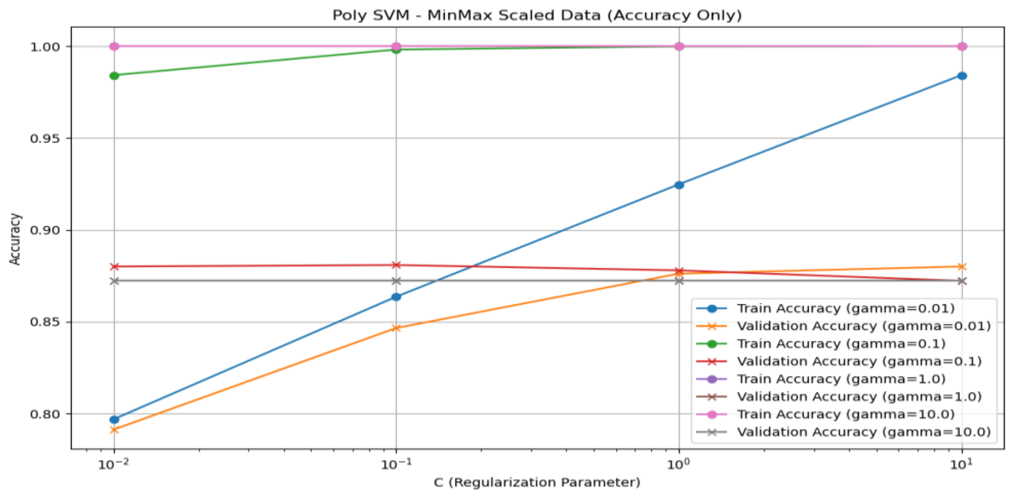
Figure 4 shows that with standard scaler is unfitting when C = .01. At C = .1 , gamma at 10 is still underfitting with an average training accuracy of .8281 and average cv accuracy of .7328. The other gammas at .01 and 1.0 both underfit at C = .1. At C = 1 we get underfitting at all gammas except when gamma is 10 then we get overfitting with a much higher training score compared to cv score.

Figure 5



In Figure 5 When I scale my poly svm my data is improved however at all gammas and c with standard scaler then my data Is overfitting.

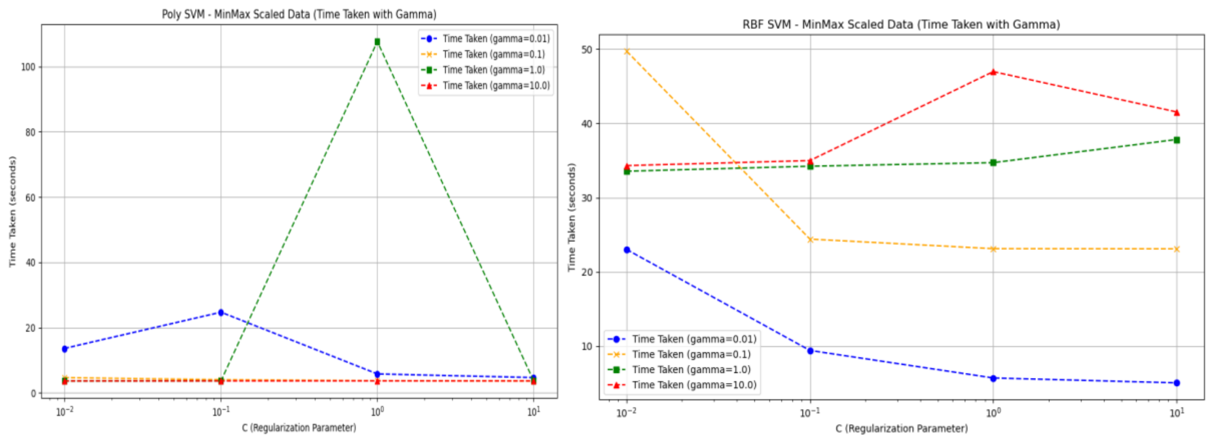
Figure 6



In Figure 6, at gamma 10 and .1 then I get overfitting on all my generalization scores. When gamma is .01 I get underfitting at C at .01, however at .1 C I get a little better generalization scores at with a training accuracy of .863 and an average cv accuracy of .8464. The next best training scores and cv scores are gamma at .01 with c at 1 with scores of .92 and cv score of .87. The rest of my data is overfitting for the other parameters.

Based on my results I am going to compare time with my minmax RBF (gamma .01, c 1) to my Minmax poly SVM (Gamma .01, C 1).

Figure 7



Based on the times above in figure 7, I will take Poly SVM MinMax with Gamma at .01, C 1 since Poly appears to be a lot more faster.

MLP Multilayer Perceptrons

Below are the parameters that I will be using to see what my best parameters are for MLP. I will check at first unscaled. Then I will check MinMax scaling and Standardscaler.

hidden_layer_sizes_values = [(10,), (10, 10), (30,), (30, 30)]
alpha_values = [0.01, 1, 10]
solvers = ['adam', 'lbfgs']

Figure 8

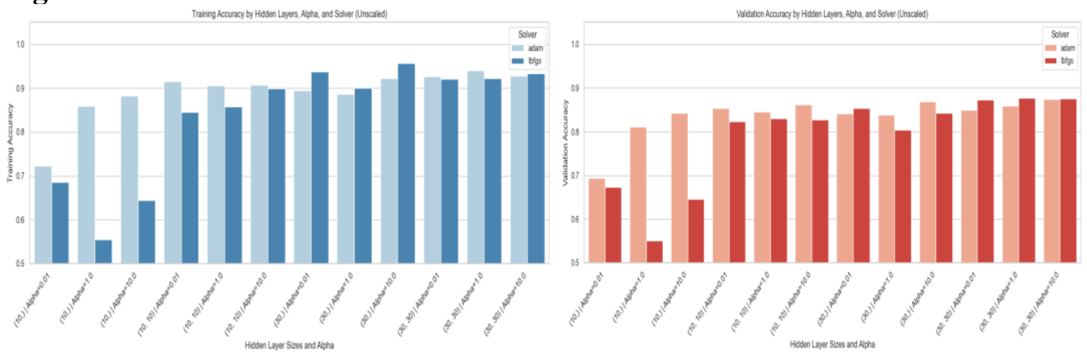
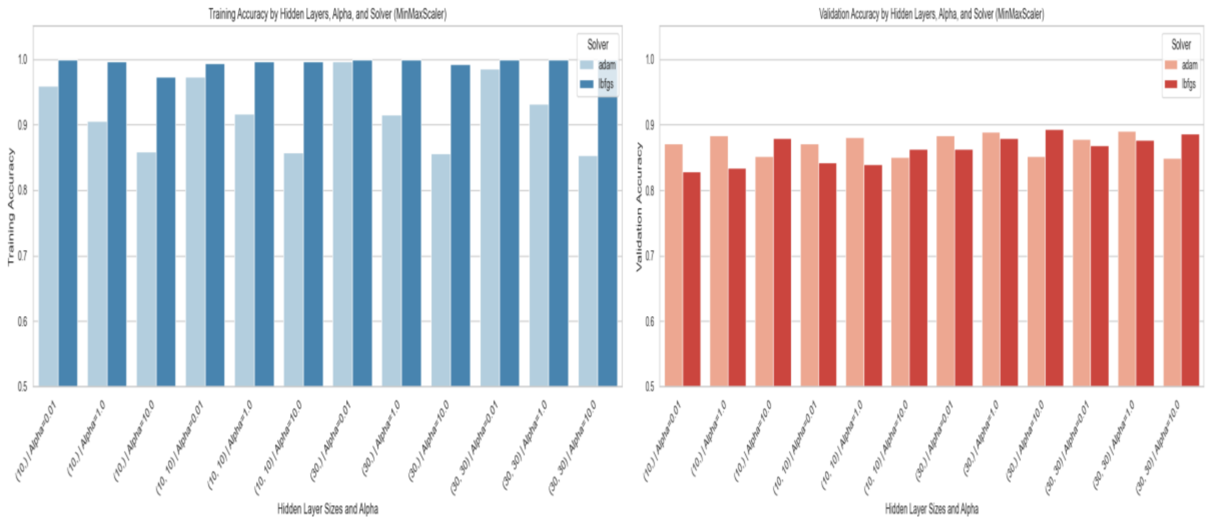


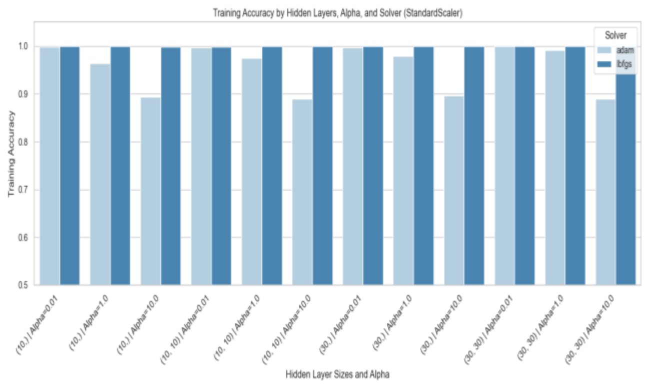
Figure 8 for unscaled data when hidden layer is just one layer at 10 then I have underfitting at alpha equal to .01with both solvers. However, at alpha equal to 1.0, I have good generalization scores at alpha equal to 1.0 with adam solver which is shown with 85% training accuracy and 82% CV training accuracy. As you can see in Figure 8 as my layers are increased by number and layer then my training accuracy rise and so does my cv accuracy. Adam solver appears to perform a lot better in my model at all stages compared to LBFGS which seems to do well as layers are increased. I do have slight overfitting in the majority of my model since my training accuracy scores are higher than my CV scores which appears to be at least 5 % to 7 % more. My best unscaled parameters appears to be Hidden layer at 30,30 with LBFGS solver and alpha at 1.0. Training is .9222 with cv accuracy of .87772. Times taken will be compared later on.

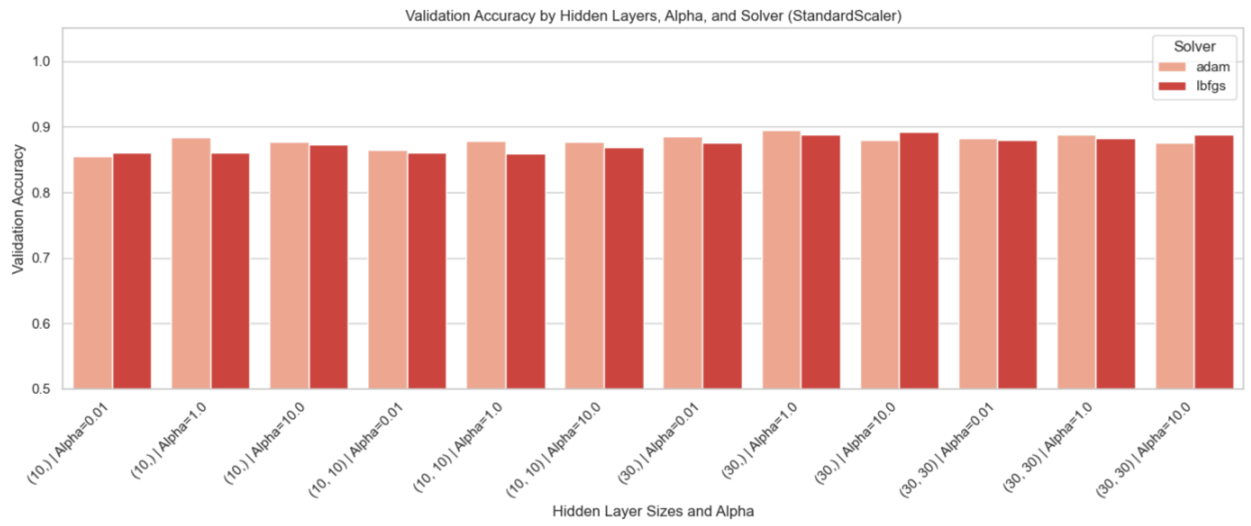
Figure 9



In figure 9 I now implemented MinMax scaler. I can tell just by looking at my chart that my training accuracy and cv accuracy scores have all improved with the new scaling. I have overfitting on Layer 10 besides when my alpha is 1.0 and solver is adam. I get a score of .9061 for training accuracy and a cv accuracy score of .8834. I have overfitting when hiden layers are increase to 10, 10, however at 10,10 with alpha equal to 1 I get an accuracy training score of .9169 and an average cv accuracy score of .8812. The rest of my data appears to overfit except for a couple other parameters. There is one parameter out of this group I think performed well with an average training accuracy score of .93142 and average cv accuracy score of .890488 (30,30 hidden layers, alpha 1.0, adam solver, Time taken to run 10.754).

Figure 10





I tend to have overfitting on almost all of my data for the standard scaler. There is a couple of parameters for this model that generalize well but unscaled and minmax outperformed this specific model with this scaling.

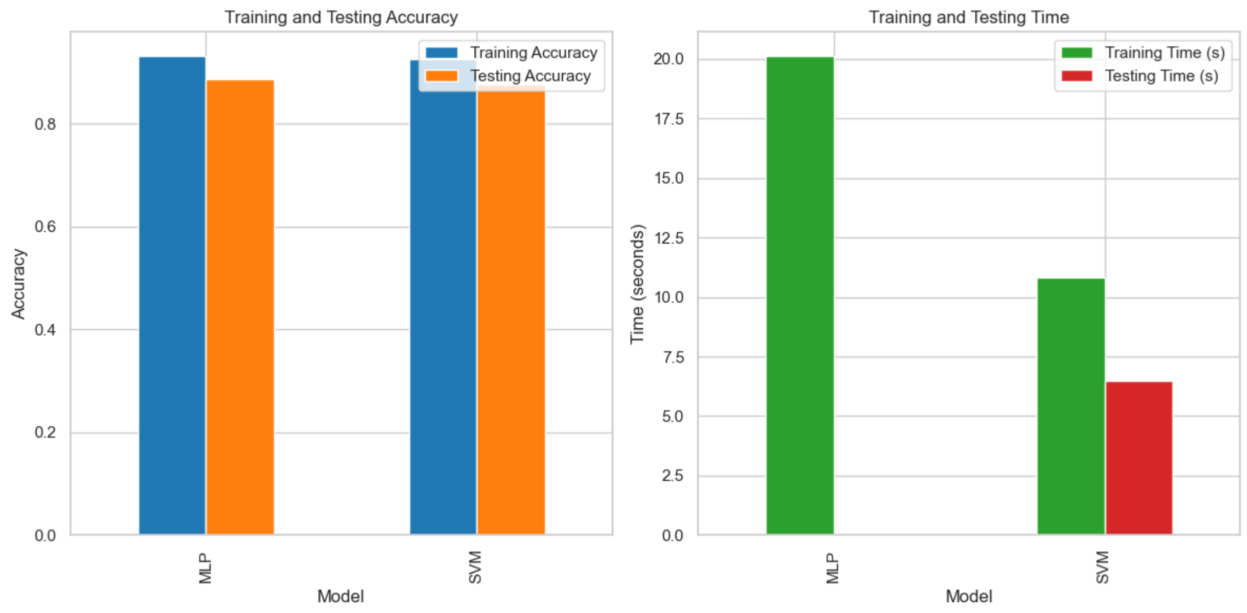
Based on my results overall I chose for my MLP model is minmax scaler with 30,30 hidden layers, alpha 1.0, adam solver. Time was good and accuracy scores were high.

Testing my models on test dataset

I chose my final parameters which are listed below. I will now be training my model with the train dataset and then testing my model on the test dataset. I will be comparing training accuracy, test accuracy and train/test time.
mlp = MLPClassifier(hidden_layer_sizes=(30, 30), alpha=1.0, solver='adam', max_iter=2000, random_state=42)
svm_poly = SVC(kernel='poly', C=1.0, gamma=0.01, random_state=42)

Figure 11 will show the comparison between the two models in which I will make a final determination on which model performed better.

Figure 11



In figure 11 both models performed really well on generalization. Training accuracy and test accuracy are both high with MLP having a score of .9317 for training accuracy and a test score of .8858. While SVM had .9257 for training accuracy and .8762 for test score. MLP is a lot slower to train on the datasert but is a lot faster on test while SVM is average on both train and testing speed. However in total time they are similar with overrall times being close to each other.

Based on the scores I willl select MLP has the best model with the parameters being Minmax scaling, Hidden Layers 30,30, Alpha = 1 and Solver = adam.

Final Analysis

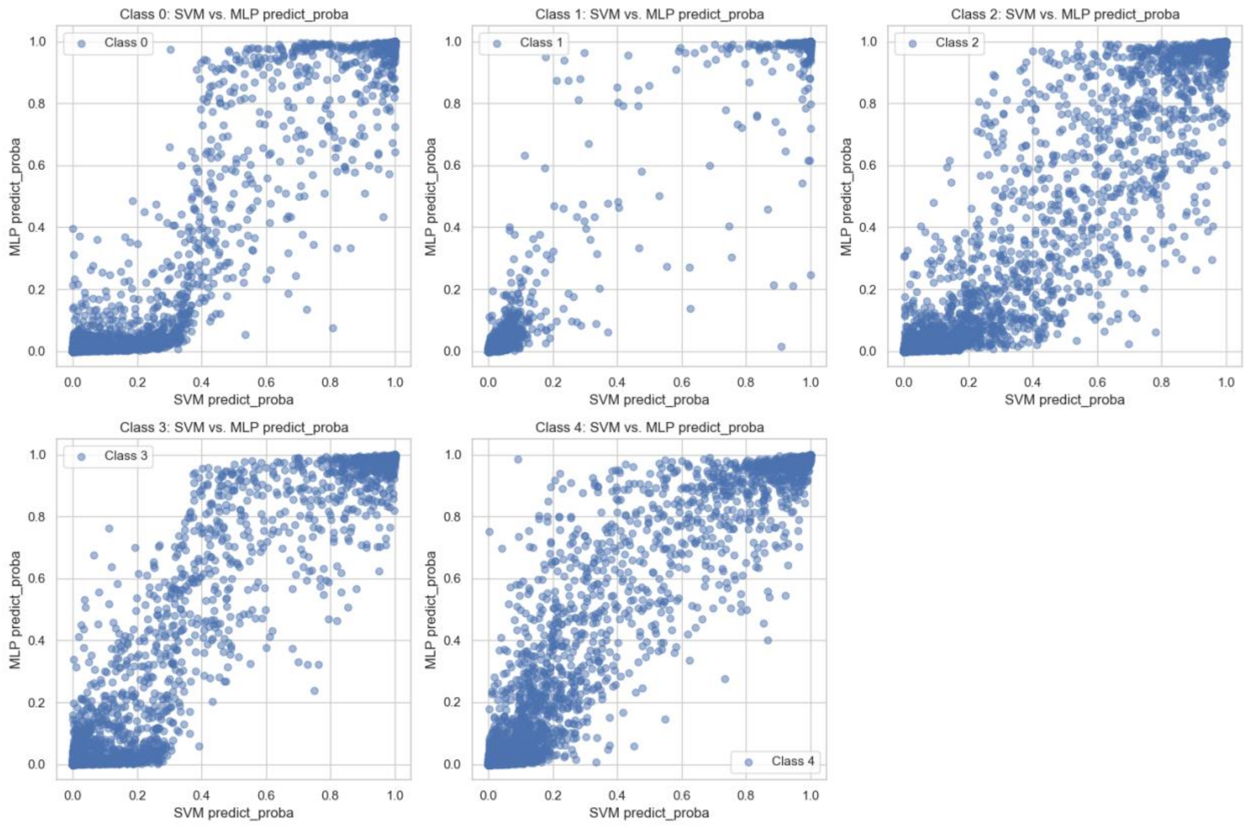
Now I will be analyzing the correlation between the output of the 2 classifiers. This will be achieved by using predict_proba on both SVM and MLP. I will be showing 3 groups.

From Professors Project Description

- G-1: Samples that are easy to classify correctly by the SVM, but hard to classify by MLP
- G-2: Samples that are easy to classify correctly by the MLP, but hard to classify by SVM
- G-3: Samples that are hard to classify correctly by both methods

For reference: The mapping of all 0-4 integers to class labels is:
0: T-shirt/top
1: Trouser
2: Pullover
3: Dress
4: Coat

Figure 12

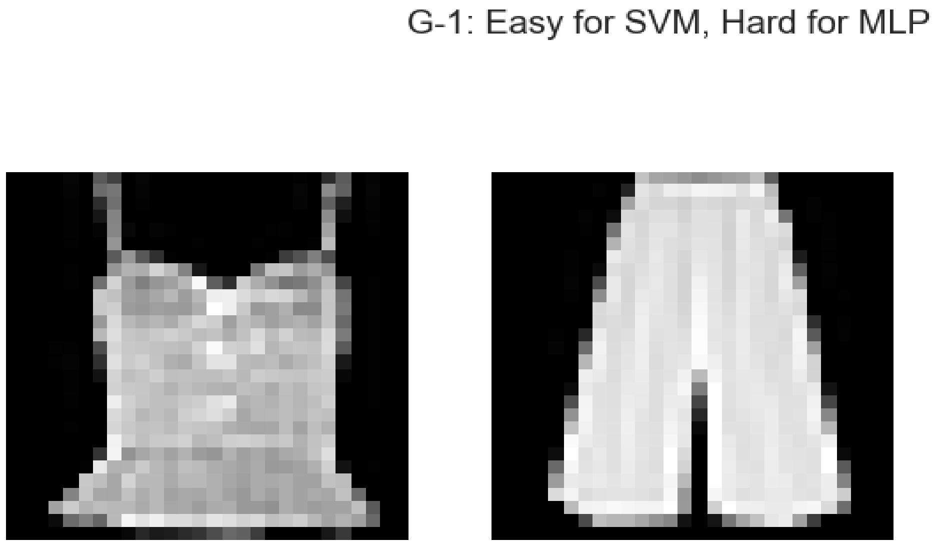


SVM and MLP appear to have similar results in probability for predicting the correct classes in Class 0 and Class 3. I believe the reason this chart for those classes looks so similar is because t-shirt/top and dress might have similar appearance on images. Sames goes for Class 2 and Class 4 since its pull over and coat. Both SVM and MLP have similar issues. I will be looking deeper at the G groups to see where my misclassifications are with each model and looking at their similarties and differences.

G-1: Samples that are easy to classify correctly by the SVM, but hard to classify by MLP
G-2: Samples that are easy to classify correctly by the MLP, but hard to classify by SVM
G-3: Samples that are hard to classify correctly by both methods

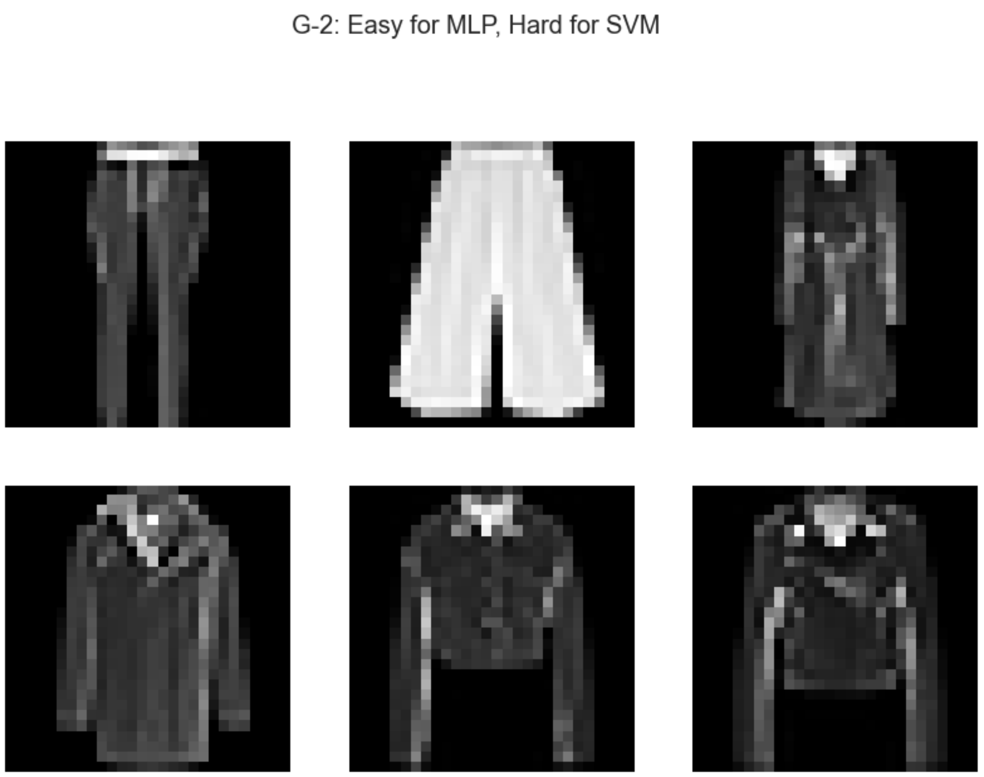
For reference: The mapping of all 0-4 integers to class labels is:
0: T-shirt/top
1: Trouser
2: Pullover
3: Dress
4: Coat

Figure 13



In figure 13 It appears SVM can distinguish between tops and trouser where MLP cant. Maybe since SVM can handle more complexity with its kernal that's it has a easy time to identify harder images compared to MLP.

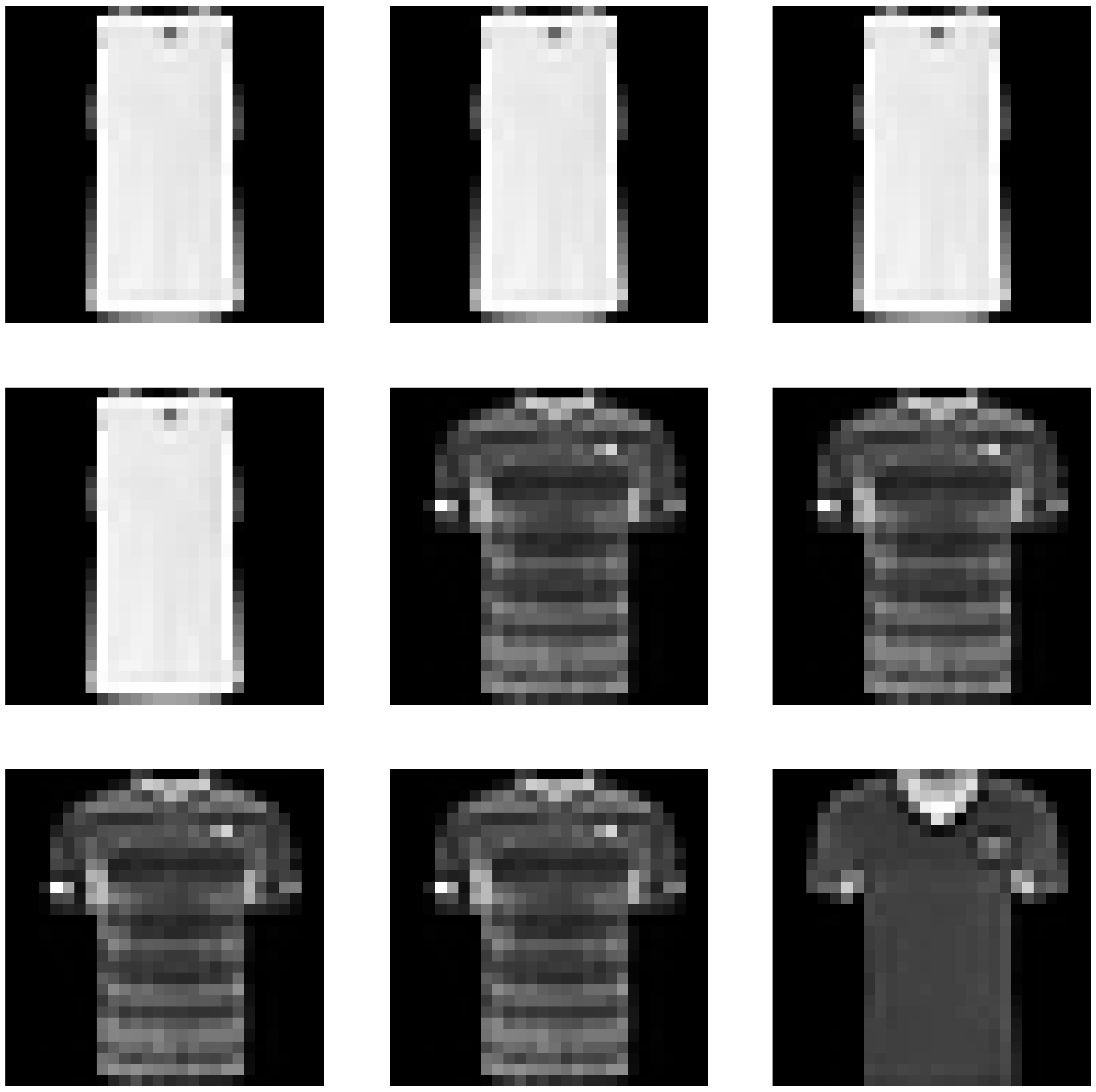
Figure 14



In Figure 14, MLP is able to identify pants, dresses, pullovers, coats and trouser a lot better than SVM. The images above look very similar which MLP can pick up but SVM cant. The images here are easier to identify to MLP because neural networks are able to distinguish features easier compared to SVM.

Figure 15

G-3: Hard for Both



Both models struggle here because the images are simple but look the same like a rectangle. Because it’s a super simple image for shirts it would appear it would struggle to identify that since shirts could look like pullovers, coats or even dresses it might not be able to identify the correct class. If the images don’t have any distinct differences than MLP and SVM can struggle it seems.

Link : https://youtu.be/_s2jiYkIxig