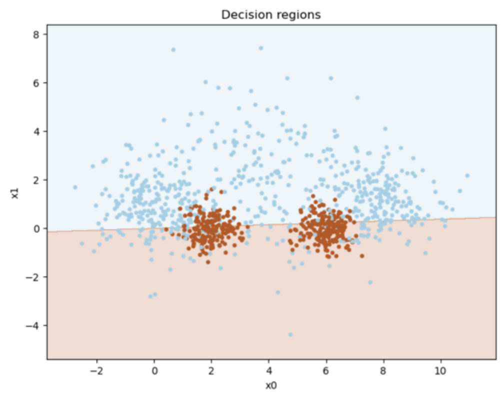
Assignment 2 – Supervised Learning

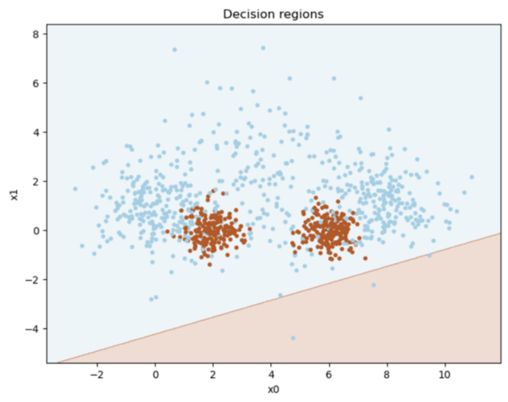
Task Tuning:

The hyperparameters I ended up with was eta = 0.07 and epochs = 200. I found these simply by playing around and testing different values. This gave me an accuracy of around 0.704. The data is very hard to separate linearly, and we won't get a much better result than this. I started by tuning the eta from 0.1 to 0.2 and this made it worse, so I changed it to 0.05. which made it even worse. Then I changed it to 0.075 and it became better and had an accuracy of 0.574.

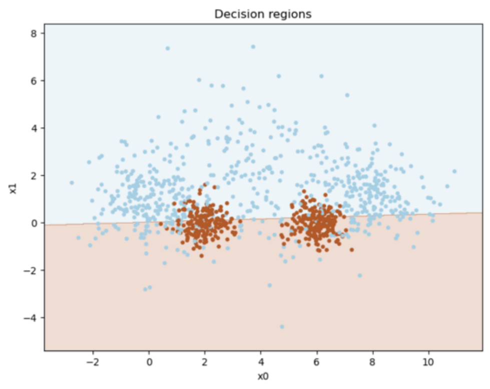
Then I changed epochs to 1000 and it made it better and gave me the accuracy of 0.0704. Then I tried to lower the value and I tried 200 and that gave me the same value. I tried to tune the eta some more, but it gave me the same values. For this dataset I think this is an acceptable result.



Eta = 0.7 Epochs = 200

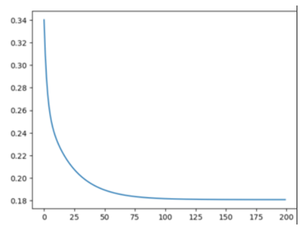


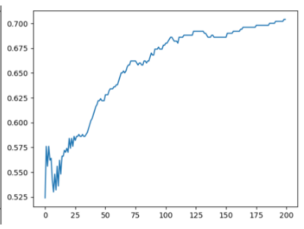
Eta = 0.75 Epochs = 10



Eta = 0.75 Epochs = 1000

Tuning Loss:

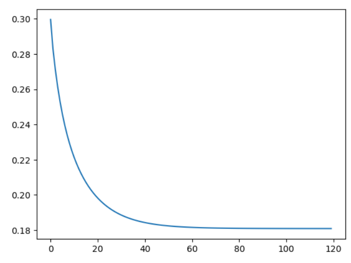

Loss

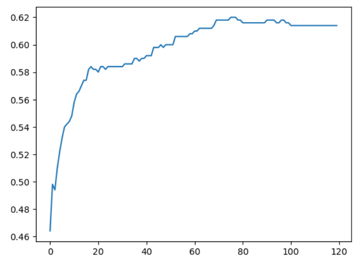

Accuracy

The function is not monotone, and this is as expected.

Scaling:

Eta = 0.07 Epochs = 120


Loss

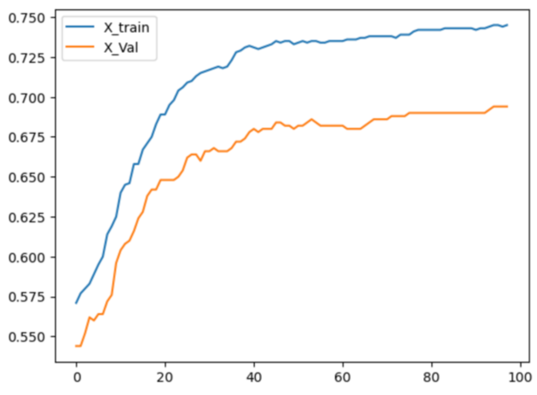

Accuracy

The way I found my hyperparameters was to just to try the ones I tested with earlier and they worked. I saw that the settling time for 200 epochs was longer then needed, so I changed it to 120 and that worked. As we see in the graph the settling time is around 100 epochs.

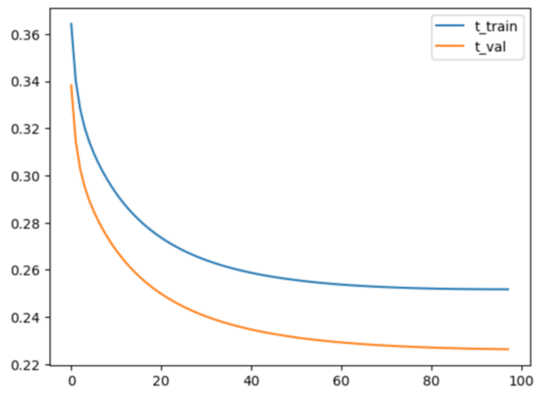
Logistic Regression:

Here I chose eta = 0.1 and tol = 0.00001.

Accuracy

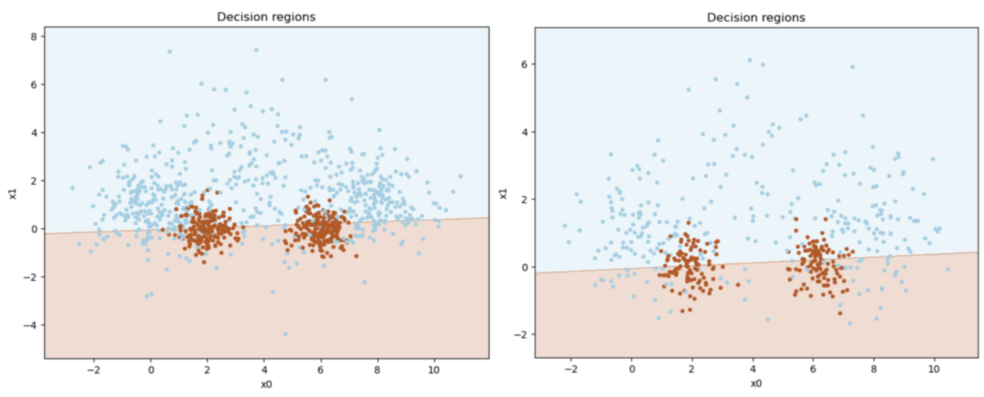


Loss



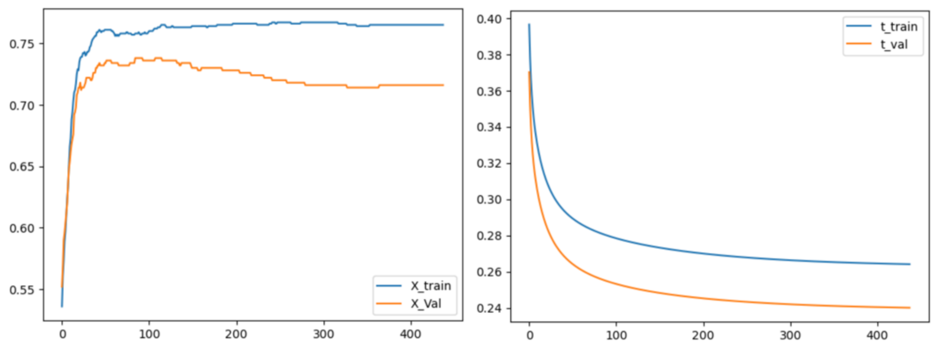
This is without scaling. It runs for 98 epochs. We see that we have very high accuracy, and a high loss. This means that we have made huge errors on very few datapoints for the training set. We have a low accuracy, and a lower loss means that we have made a lot of small errors on a lot of data. If we plot this in datapoints we can see why this is.

Training data Validation data



If we plot the same for the scaled data

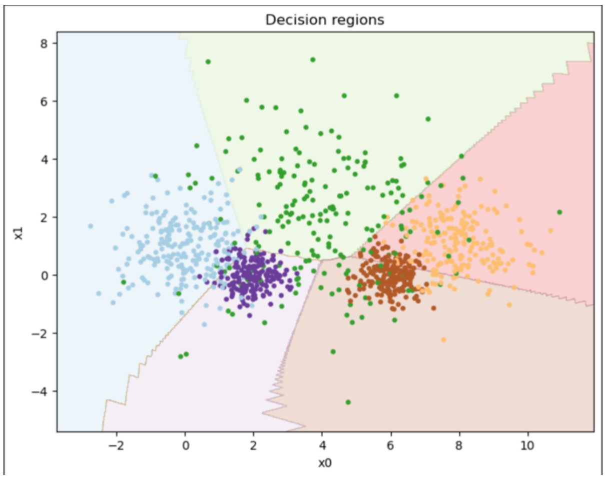
Accuracy Loss



This is with scaling. We can see that for the first 100 epochs, the accuracy goes up. But since we ran the other with very small *'tol'*, it registers when the accuracy goes down again so it doesn't stop running until it's ran about 438 epochs. It's the validation set that goes down in accuracy. This can be because we overfit the dataset, because we don't see a decrease in the training data.

Multi-class classifier:

I ran this for eta = 0.1 and epochs = 1000. These are the regions that I got.

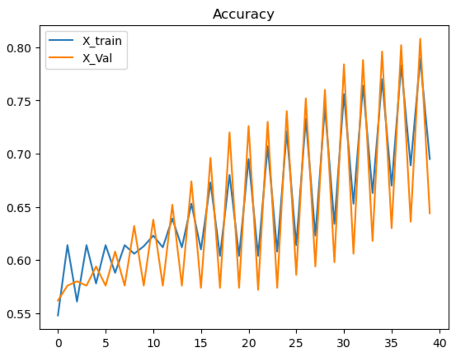


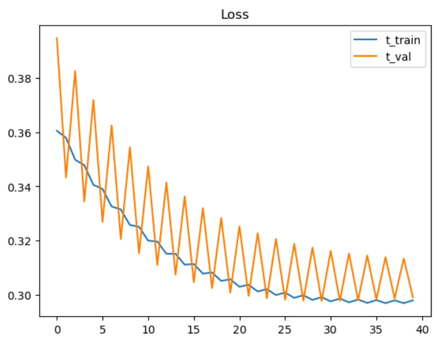
I would say this is a pretty good adaptation for the classes.

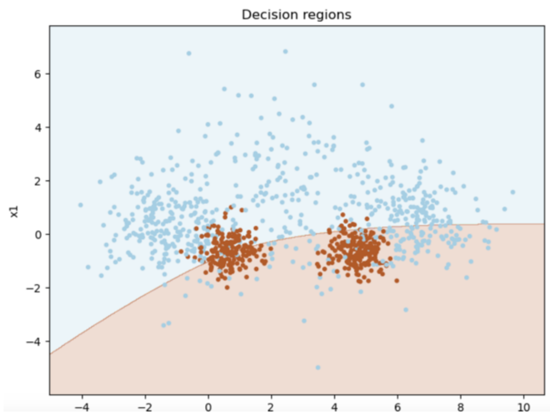
Multi-label classifier:

The best accuracy that I got was about 0.76 for eta = 0.001 and nr of epochs = 37, tol = 0.000098. I let it run for about 400 epochs with the same variables just without the tol. Hidden_dims = 56.

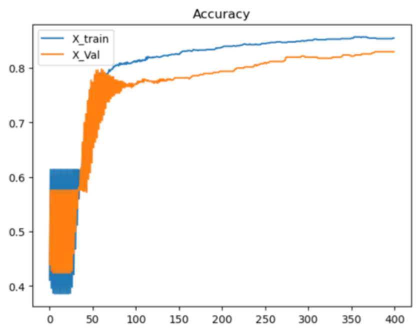
With tol and n_epochs_no_update

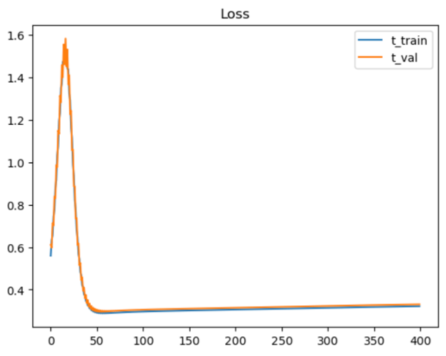


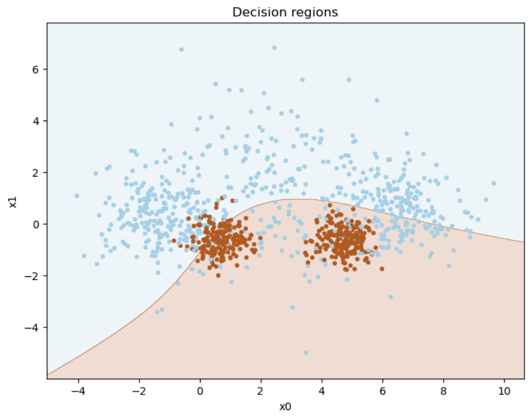




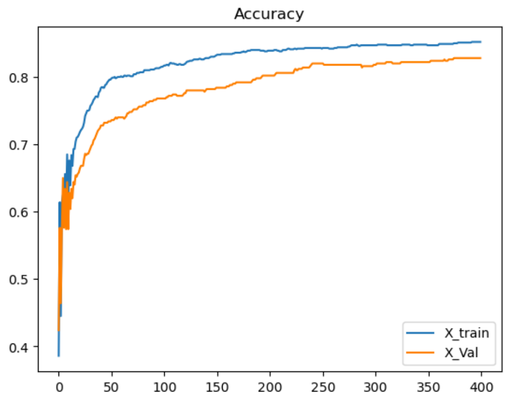
Without tol and n_epochs_no_update

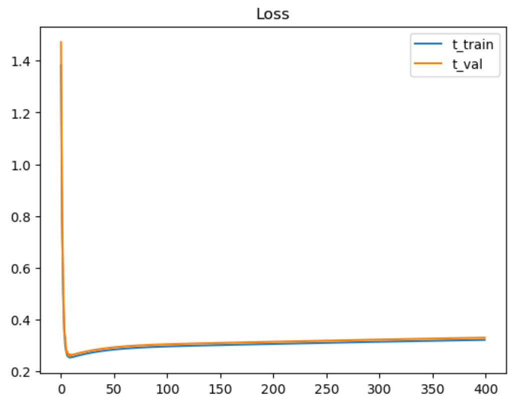


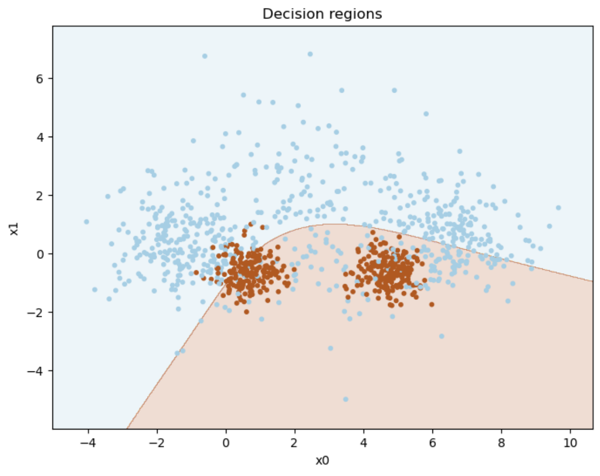




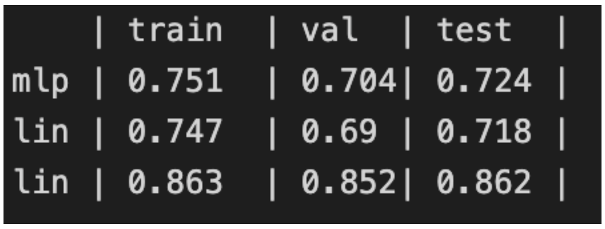
After running the code for 10 times and finding the one with best accuracy the one I got was:







Part 3: Final Testing:



```
    | train  | val  | test  |
mlp | 0.751  | 0.704| 0.724 |
lin | 0.747  | 0.69 | 0.718 |
lin | 0.863  | 0.852| 0.862 |
```

We can see that the one with the best accuracy was the multi-label-binary linear regression. The one with the worst was the logistic classifier.