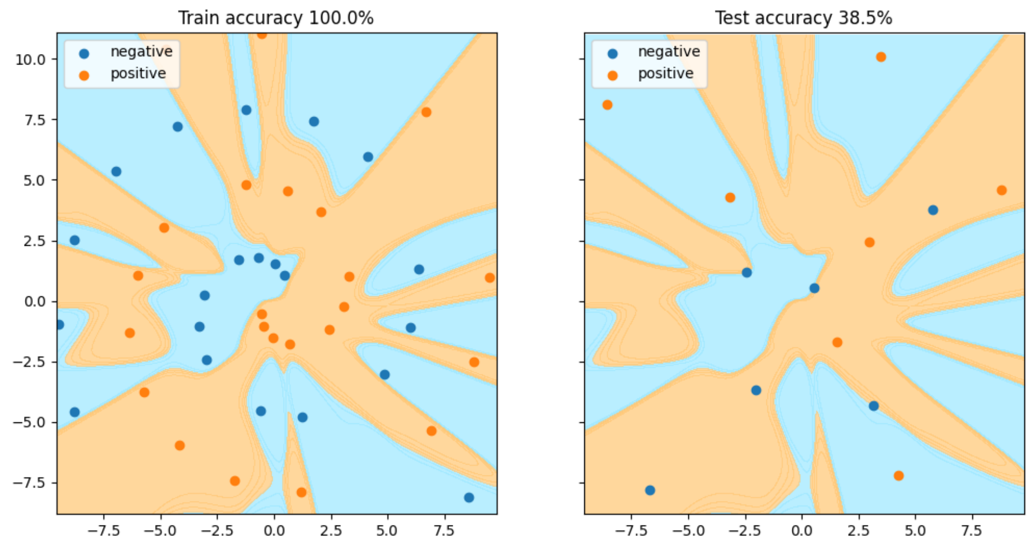
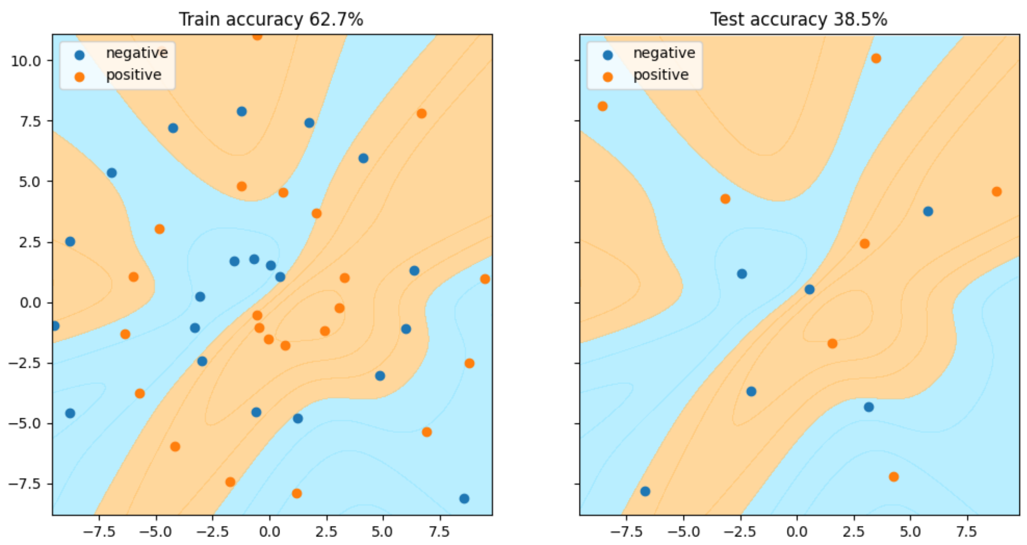
2a.

I was able to pass the 'test_custom_transform' case. I did this by figuring out parametric equations for each spiral of the form $x(t) = \alpha t cos(t), y(t) = \beta t sin(t)$. I learned that for the first spiral, $\alpha = \beta = 1$, and for the second spiral, $\alpha = \beta = -1$. I then wrote a function that would plot 1000 points on these spirals between t=0 and t=20. My data transform function called this function to create lists of points representing each spiral. For each element of the data set, I then computed two new features: the minimum distance of that point to the points constituting the first spiral, and the minimum distance of that point to the points constituting the second spiral. Obviously, points belonging to the first spiral would be very close to it, and points belonging to the second spiral would be very close to that one. These new features allowed a logistic regression model to easily learn which points belonged to which spiral.

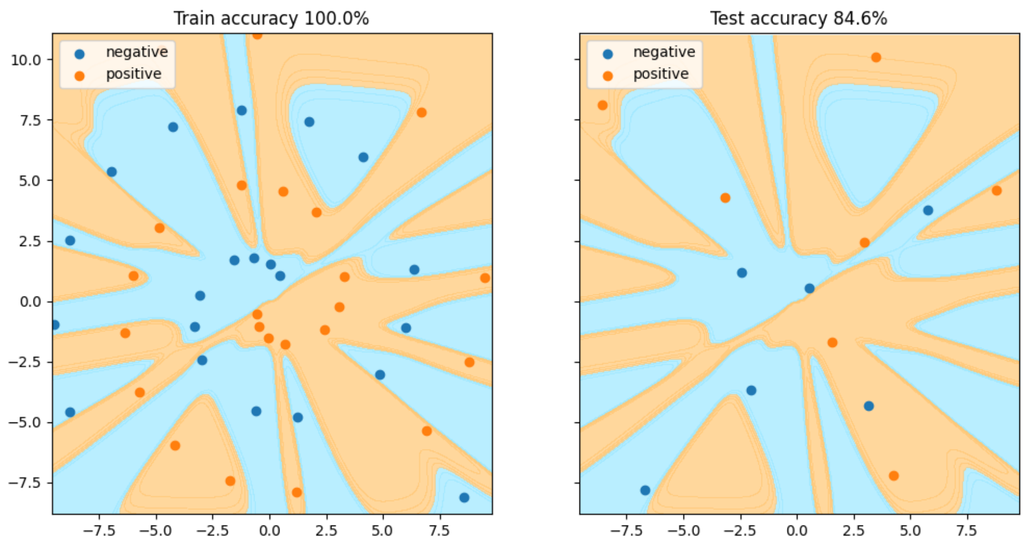
2b.





The difference between these two decision boundaries is that one used a much lower learning_rate (0.01 instead of the default 1). Changing the hyperparameters influenced the decision boundaries in the following ways: decreasing the learning rate tended to lead to simply boundaries, and adding more nodes and layers tended to lead to more complicated boundaries. Increasing alpha tended to lead to better accuracy (I could not find much accuracy improvement with most of the other things I tried to vary).

1c.



For this model, I used the standard hyperparameters, only with alpha set to 1 instead of 0. It worked pretty well – I found that changing this parameter had the greatest effect on accuracy of everything I tried (increasing the number of nodes increased accuracy a little bit but not nearly so much and using an l1 penalty led to a really underfit model although that's maybe because alpha was set too large).