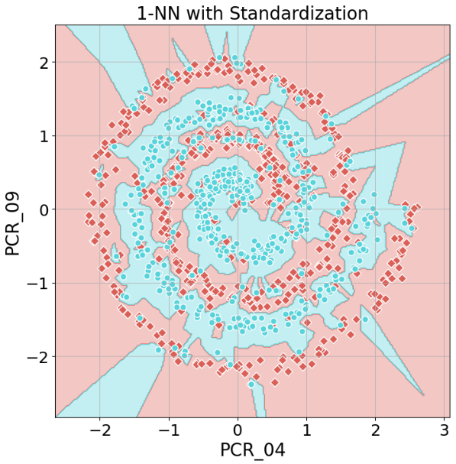
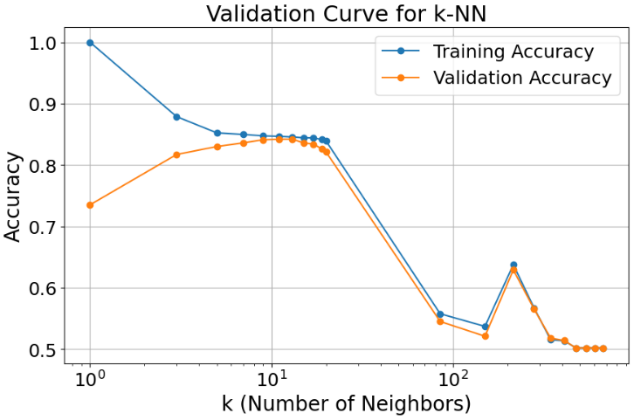
**Part 1**

**A1:**

**A2:**

Best k: **11**

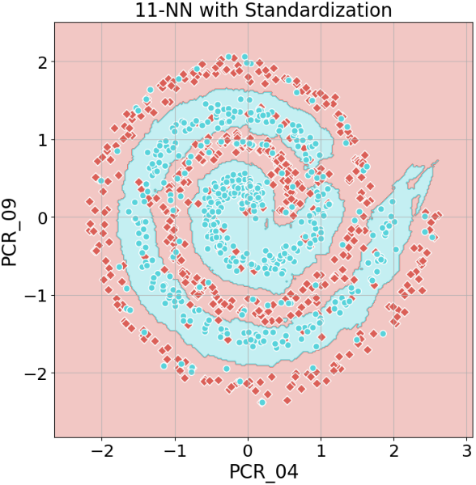
Average Training Accuracy: **0.8466**

Average Validation Accuracy: **0.8420**

Values that cause underfitting: **85, 150, 215, 280, 345, 410, 475, 540, 605, 670**

Values that cause overfitting: **1, 3**

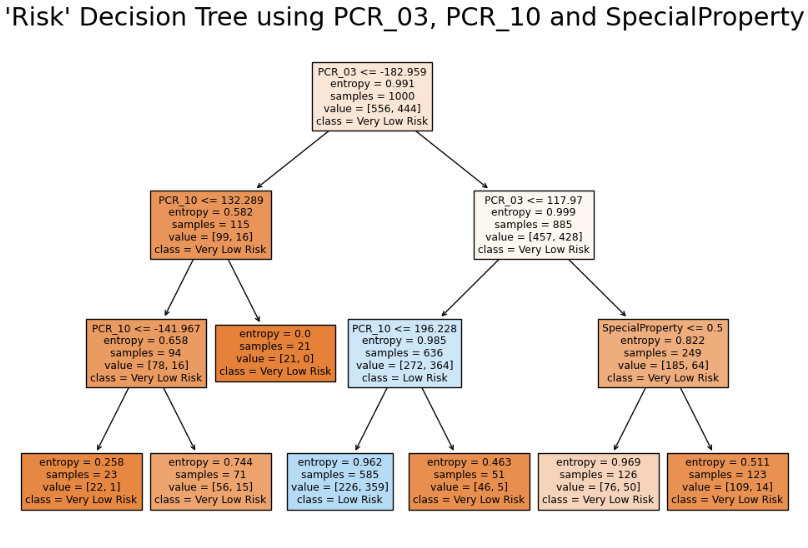
**Underfitting** occurs when the model is too simple to capture the structure of the data – resulting in poor performance on the training data and on the testing data.

****On the other hand, **overfitting** occurs when the model fits the training data too well, catching outliers and noise that doesn’t generalize the data well enough.

**A3:** the Test Accuracy is **0.5**

**A4:**

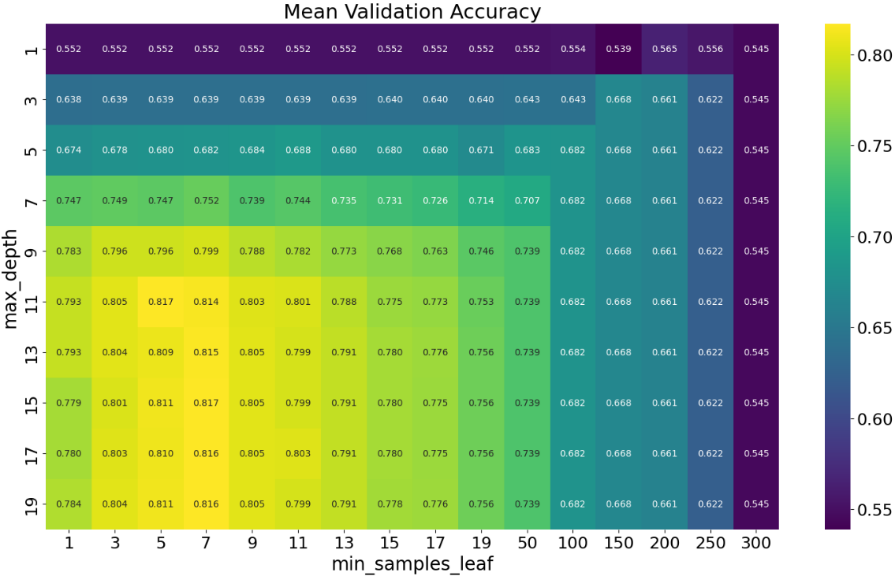
the 1-NN boundary is highly complex and “jittery” which makes it very flexible and sensitive to noise and outliers in the training data, in comparison, the 11-NN boundary is way smoother and more generalized which makes it less likely to follow noise and outliers in the training data despite the observed pooper performance(0.5 Accuracy compared to 1-NN’s 0.72 Accuracy).

**Part 2**

**A5:**

**Training Accuracy: 0.69**

**A chart of a number of colors

Description automatically generated with medium confidenceA6:**

**c. Optimal:** (max\_depth,min\_samples\_leaf) = (11,5)

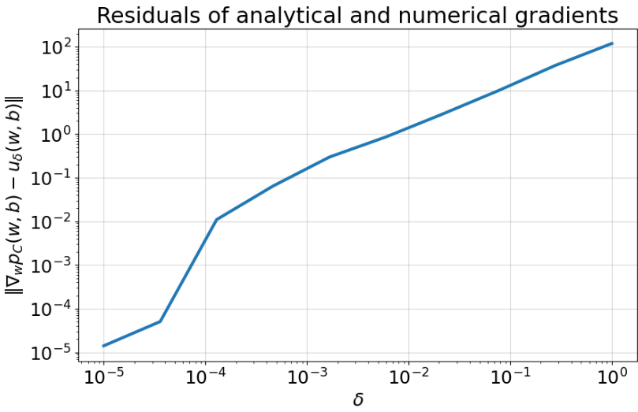
**d+f. Underfitting:** (max\_depth,min\_samples\_leaf) = (1,1), because both validation and training accuracies are low

**e+f. Overfitting:** (max\_depth,min\_samples\_leaf) = (19,1) because it has the highest training\validation difference (0.216 diff)

**A7:** the number hyperparameter combinations in my grid is: 10\*16 = 160, if I wished to add a third hyperparameter the number of combinations would be 160\*(amount­\_of\_values) which significantly increases the computational cost and time required for a grid search, which in result would increase the model complexity

**A8:** Test Accuracy using (max\_depth = 11,min\_samples\_leaf = 5): **0.8520**

**Part 3**

**A9:** that is too large will make the numeric derivative be further away from the real derivative therefore the norms between the analytical gradient and the numeric gradient will only increase

**A10:**

C is extremely large therefore the clsasifier will try it’s best to missclassify as little as possible(somewhat similar to hardSVM).

looking at the data plot we can see that the data inseperable linearly which is why we won’t be able to get high accuracy(around 53% at best). Another thing is that we use hinge-loss which penalizes missclassifications by their distance from the margin(which also affects the accuracy of the classifier).

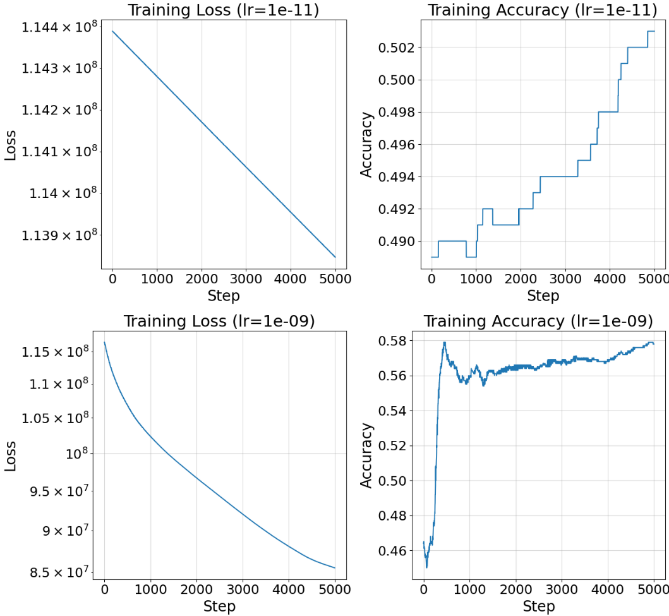
lr(Learning Rate = step size) is extremely small and as a result it takes a while to find the best accuracy(in our case ~3500 steps)

**Train Accuracy:**

* initial increase – as expected because the model learns and becomes better at predicting.
* Fluctuations - might be caused by overfitting in a case where the model learned the noise of the training data

**Train Loss:**

* Initial decrease - the loss starts at a very high values and initially it decreases rapidly – as expected because the model is quickly learning and reducing errors on the training data.
* Plateauing – it gets to a point where further training doesn’t significantly reduce the training error

**A11:**

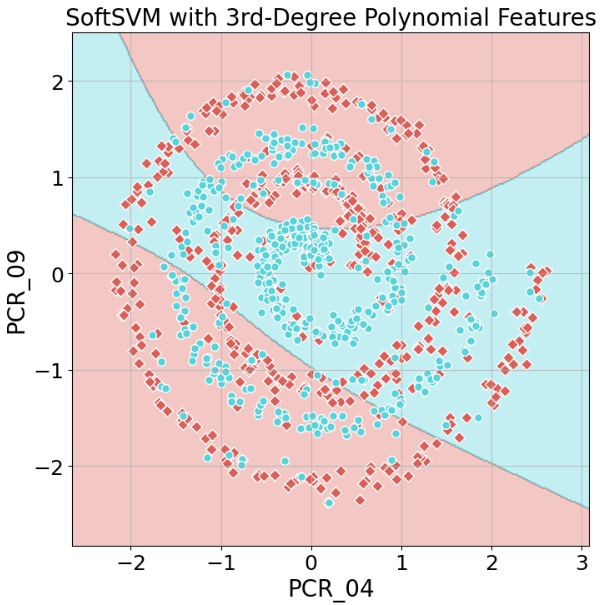
I didn’t include the 1e-03 plot here but it is similar to the 1e-05 plot;

The learning rate I’d choose is **1e-09** because it seems to have the best Training Loss & Accuracy plots as they are not as jittery as the 1e-05&1e-07 plots(too big of a step resulting in the model not being able to converge to a minimum as good as the models with the smaller step)

Also, the training accuracy of the 1e-09 is better than of the 1e-11 because it takes way less steps to get to the maximum accuracy

**A graph of steps and steps

Description automatically generated**

****

**A12:**

Training Accuracy: 0.59

Test Accuracy: 0.57

**Part 4**

**A13:**

**A14:**

**A15:**

**A16:**