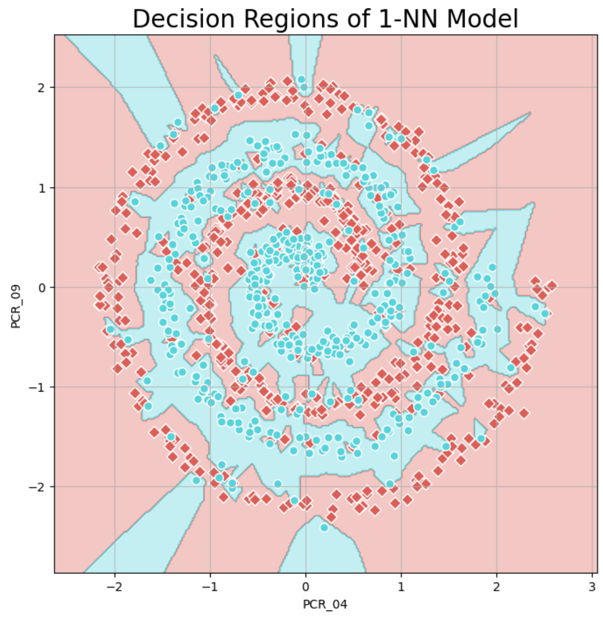
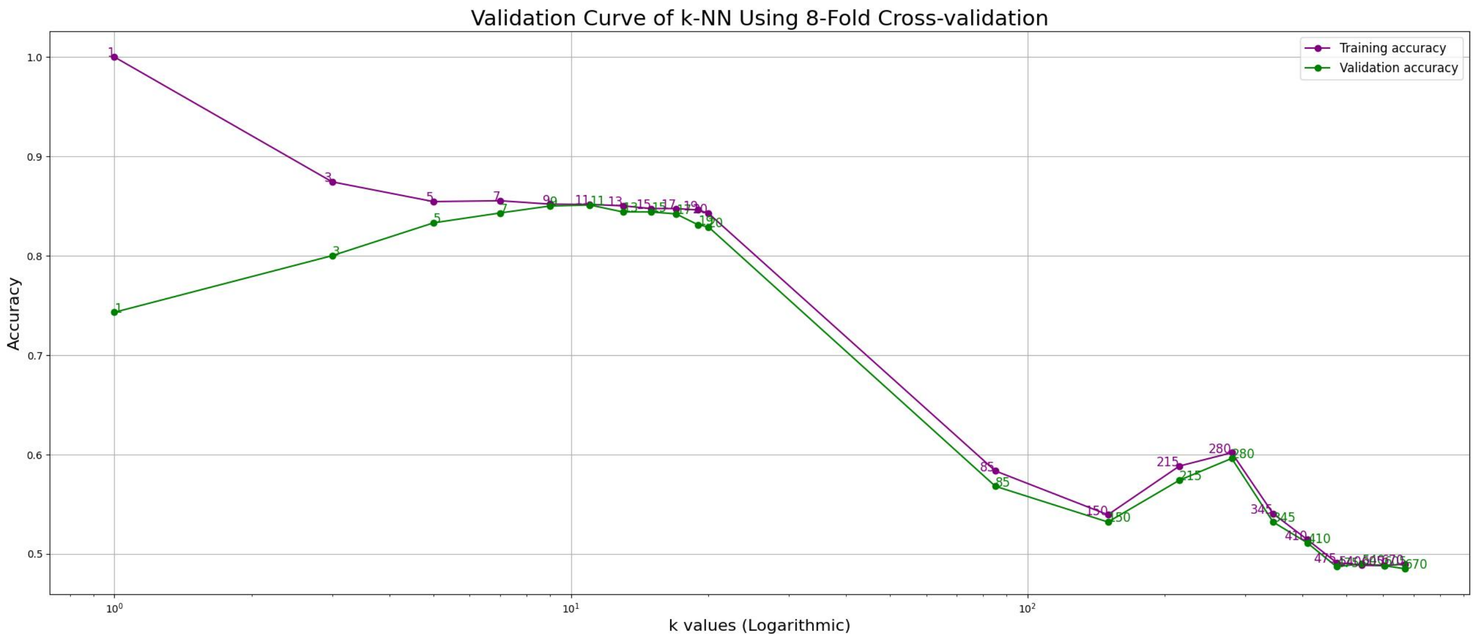
HW2 - Algorithm Implementation and Basic Model Selection

Part 1: Basic model selection with k-Nearest Neighbours

(Q1) The decision regions we got from training 1-NN model on our training set are:



(Q2) The cross-validation with 8 folds gave us the following validation curve:

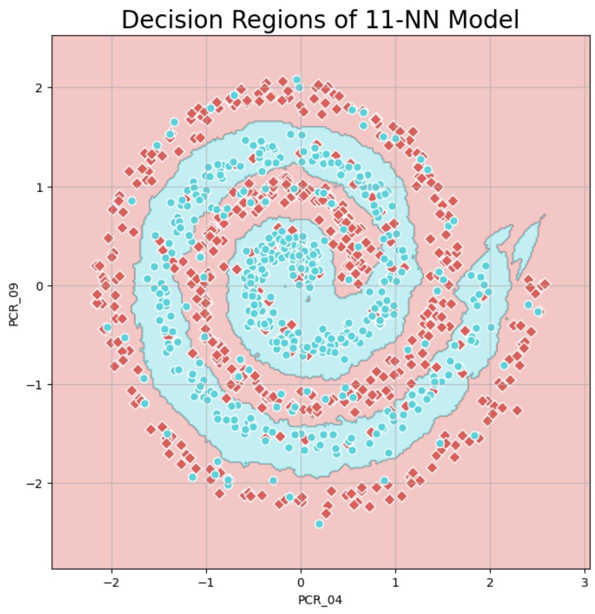


As we learned in class, the overfitting phenomenon occurs when the model "learns too hard" (has no generalization ability, too specific) the training data (overfit) resulting in accurate results for the training data and incorrect results for the test data. This means we see a big difference in accuracy between the training and test (validation) overfitting. In our Validation curve, we get **overfitting in smaller values of k**.

Underfitting occurs when the model does not capture the data well leading to poor

performance on both the training and testing datasets. In our Validation curve, we get **underfitting in high values of k**.
The optimal k is the one that gives us the closest training and validation scores, meaning in the validation curve, the point with the best overlap. **In our case, the best k is k=11, with a training score of 0.851571 and a validation score of 0.851.**

(Q3) The decision regions we got from training 11-NN model on our training set are:



The 11-NN **test accuracy is 0.792. (**11-NN train accuracy is 0.853).

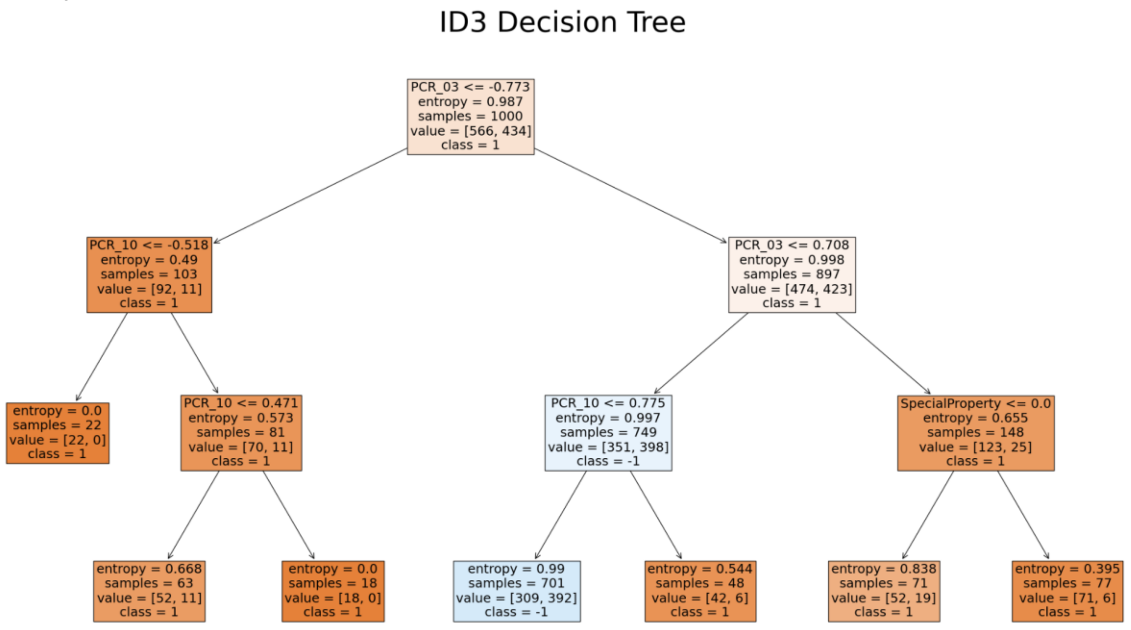
(Q4)

Comparing the boundaries of the 1-NN and 11-NN models, we can see that the 1-NN model boundaries are far more "irregular" (sharp and pointy) compared to the smooth boundaries achieved by the 11-NN model. In addition, the blue and red spots in the 1-NN model are "mixed up" while the 11-NN model has a better separation between the two labels. These differences illustrate the model's ability to deal with outliers in the training set, hence representing the overfitting phenomenon, harming the model's ability to generalize its performance.

Part 2: Decision Trees

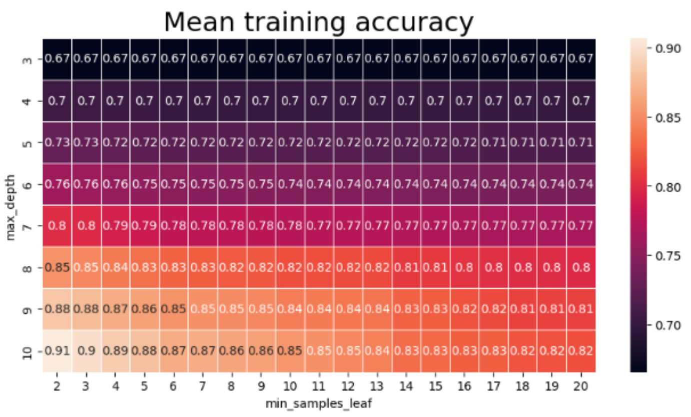
(Q5)

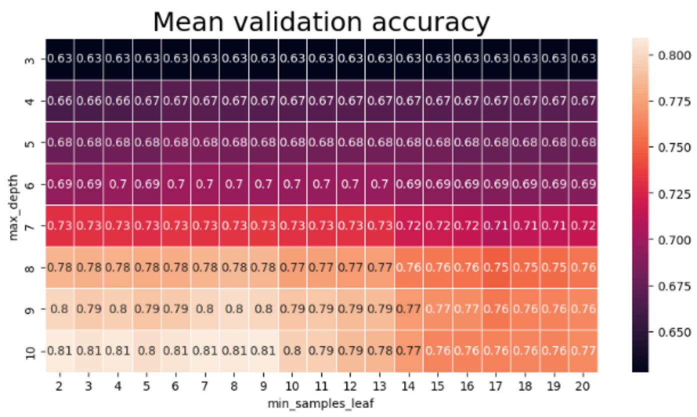
Training the model with ID3 and max_depth=3 resulted in the following tree, with training accuracy of 0.649:


ID3 Decision Tree
PCR_03 <= -0.773 entropy = 0.987 samples = 1000 value = [566, 434] class = 1
PCR_10 <= -0.518 entropy = 0.49 samples = 103 value = [92, 11] class = 1
PCR_03 <= 0.708 entropy = 0.998 samples = 897 value = [474, 423] class = 1
entropy = 0.0 samples = 22 value = [22, 0] class = 1
PCR_10 <= 0.471 entropy = 0.573 samples = 81 value = [70, 11] class = 1
PCR_10 <= 0.775 entropy = 0.997 samples = 749 value = [351, 398] class = -1
SpecialProperty <= 0.0 entropy = 0.655 samples = 148 value = [123, 25] class = 1
entropy = 0.668 samples = 63 value = [52, 11] class = 1
entropy = 0.0 samples = 18 value = [18, 0] class = 1
entropy = 0.99 samples = 701 value = [309, 392] class = -1
entropy = 0.544 samples = 48 value = [42, 6] class = 1
entropy = 0.838 samples = 71 value = [52, 19] class = 1
entropy = 0.395 samples = 77 value = [71, 6] class = 1

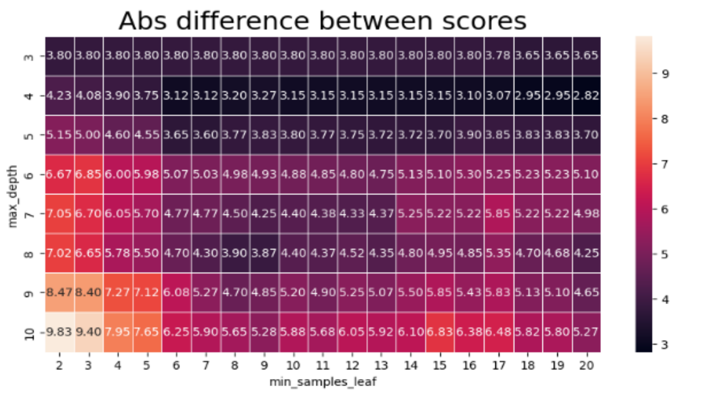
(Q6)

To find the best hyperparameter combination we used heatmaps, where in each iteration we narrowed and specified the grid area to find the best combination of max_depth and min_samples_leaf.





To validate our choice, we used a heatmap of the absolute value of the difference between the 2 scores multiplied by 100,



C. From the 3 plots, we can conclude that the hyperparameter combination of **min_sample_leaf=9 and max_depth=10.** We can see the difference is the lowest of the best validation accuracy.

D. A hyperparameter combination that causes **underfitting: min_samples_leaf=20 and max_depth=4.**
Underfitting will happen when the decision is very generalised, meaning is made by the "majority" of the examples, which happens min_samples_leaf parameter is large and max_depth is small, as seen also in the heatmaps.

E. A hyperparameter combination that causes **overfitting: min_samples_leaf=2 and max_depth=10.**
Overfitting will happen when the decision is specific to the training data and not generalised enough resulting in lower accuracy of the test data. This happens when the values of min_samples_leaf are small with large values of max_depth.

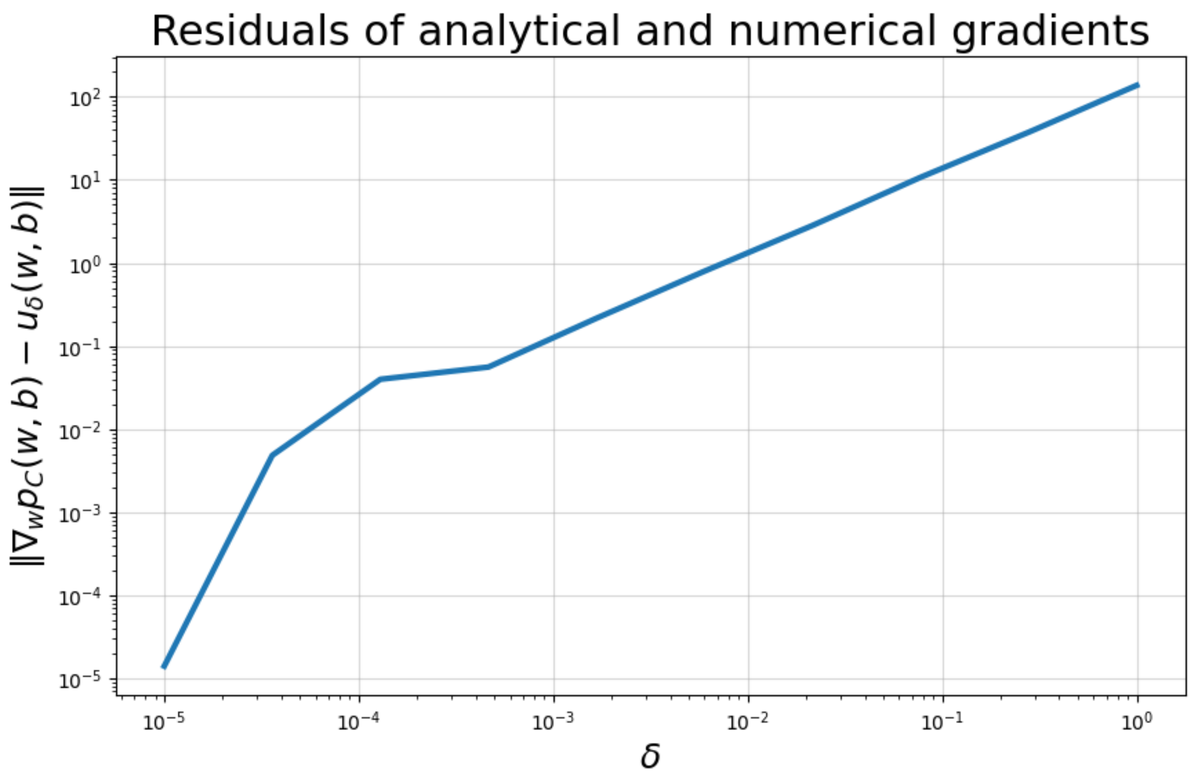
(Q7)

We used 19 values for the min_samples_leaf hyperparameter and 8 values for the max_depth hyperparameter. In total, we had **152 combinations to evaluate.**
A third hyperparameter would multiply the total amount by the number of values we would choose for it.
With m being the number of hyperparameters to evaluate, we can express the total combinations number as: $\Pi(x_i), i = (1, m)$. With the simplifying assumption of $x_i = x_j$ we get a total number of x^m which means the total amount of combination is exponential in the number of hyperparameters.

(Q8)

Training a decision tree with the hyperparameters: min_sample_leaf=9 and max_depth=10, gave us a test accuracy of 0.844.

Part 3: Linear SVM and the Polynomial kernel



(Q9)

The graph shows the difference between the analytical and numerical gradients as a function of delta. The analytical gradient should be quite accurate, so the main factor is the effect delta has on the numerical gradient. The numerical gradient is calculated by the equation: $\frac{f(x_0+\delta)-f(x_0)}{\delta}$.

When delta approaches zero, this equation is exactly the definition of the derivative (in our case, generalised to the gradient) of f(x) in x_0. Which means: **For smaller delta values, we expect to see a small difference between the numerical and analytical gradient, and for greater deltas we expect a bigger difference, as seen in the graph.**

(Q10)

Regarding the **Loss** as a function of step: The graph matches our expectations. The SGD algorithm minimizes the loss function at every step, so we were expecting a decreasing graph. Additionally, it is likely that in the first steps, we will observe a sharp decline in the loss compared to the last steps where we were expecting a plato. This is because the closer the algorithm gets to the minimum, the lower the gradient is, therefore the smaller the step is.

Regarding the **Accuracy**: because the problem is not separable we got a weird behaviour.

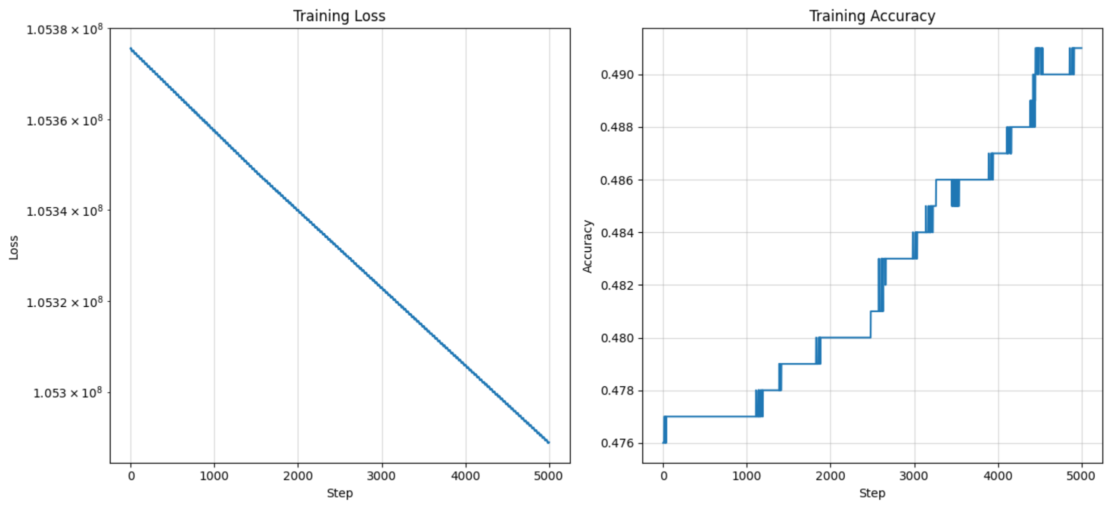
(Q11)

We would choose lr = **1.e-09.**

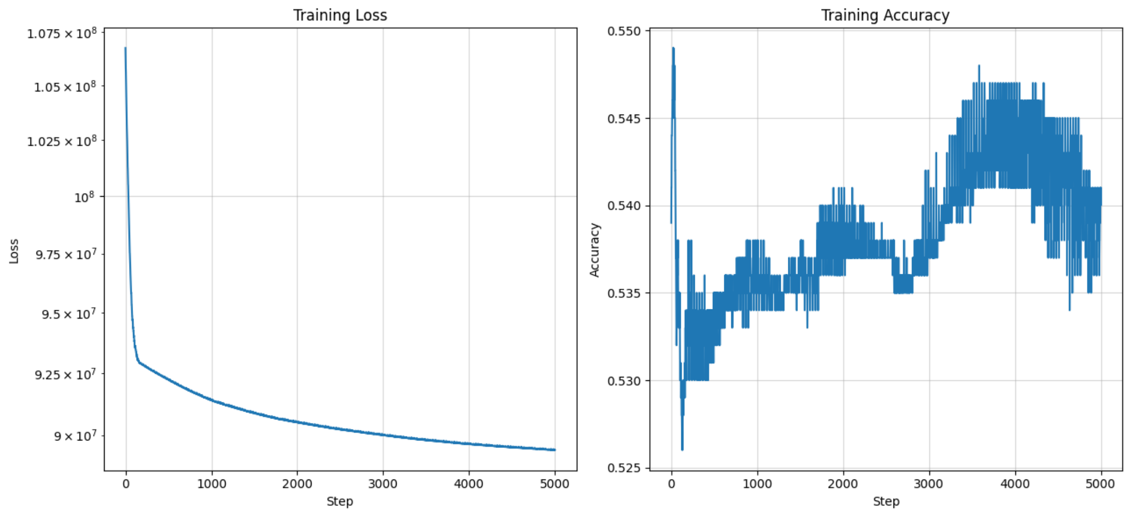
In the given steps number, the smaller learning rate (1.e-11) does not reach the same

accuracy and loss as (1.e-09), suggesting this lr (and therefore the step size as well) is too small.
The larger learning rates produce a very large step which makes it difficult for the model to converge to a minimum. In the graphs this phenomenon looks like a very noisy and fluctuated lost and accuray.

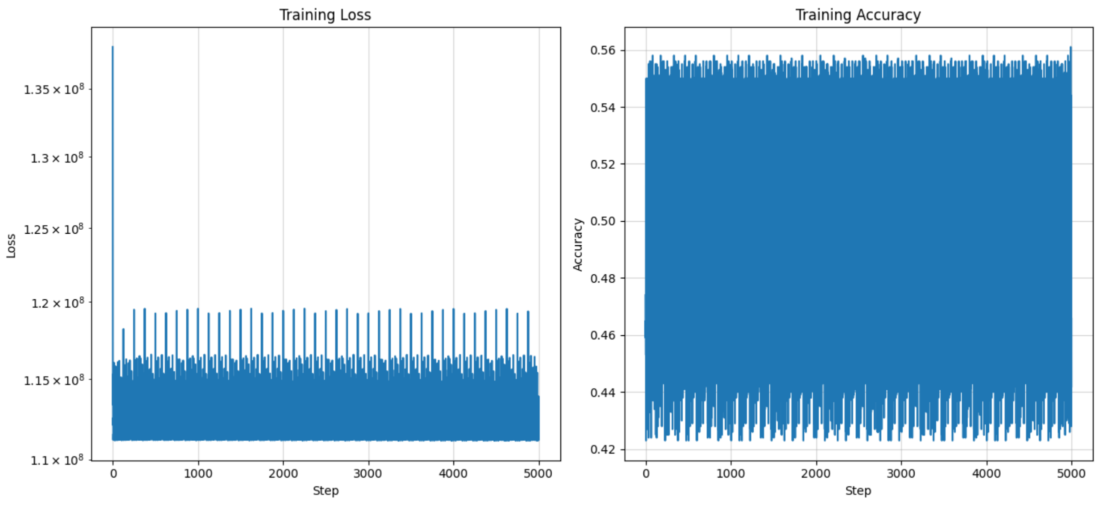
Learning rate = 1.e-11



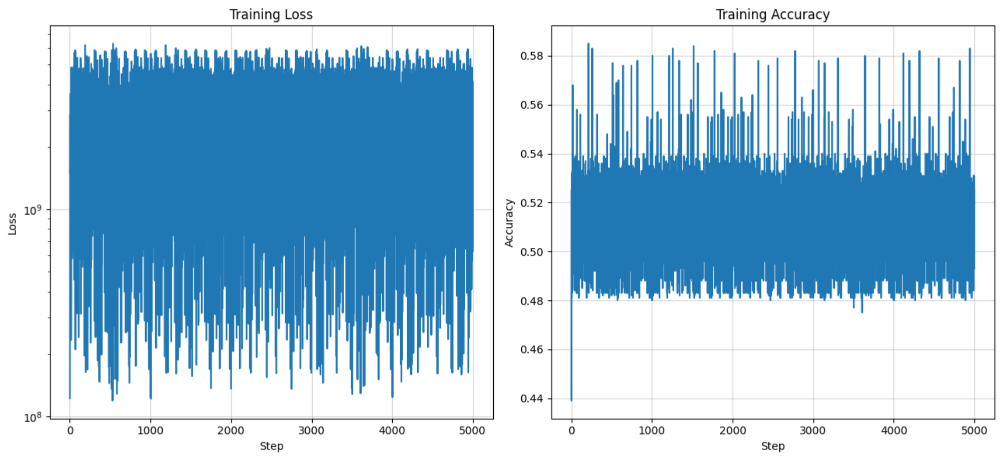
Learning rate = 1.e-09



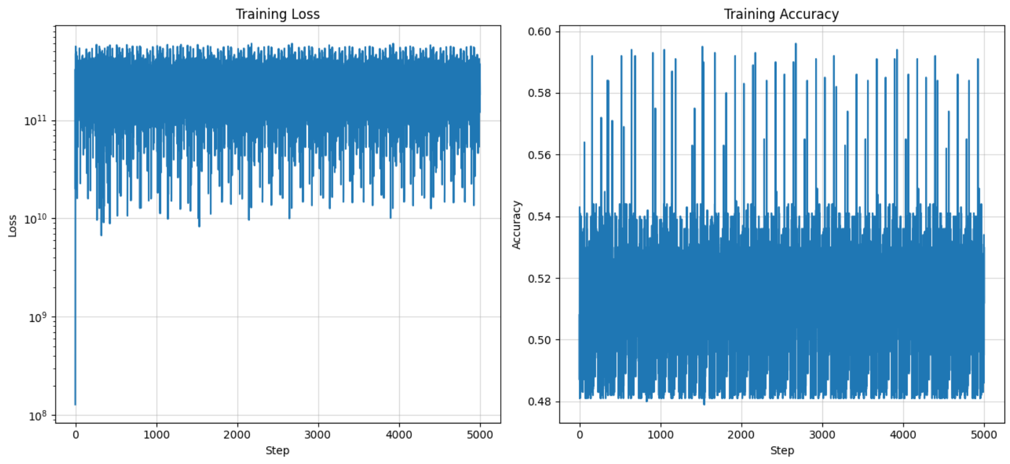
Learning rate = 1.e-07



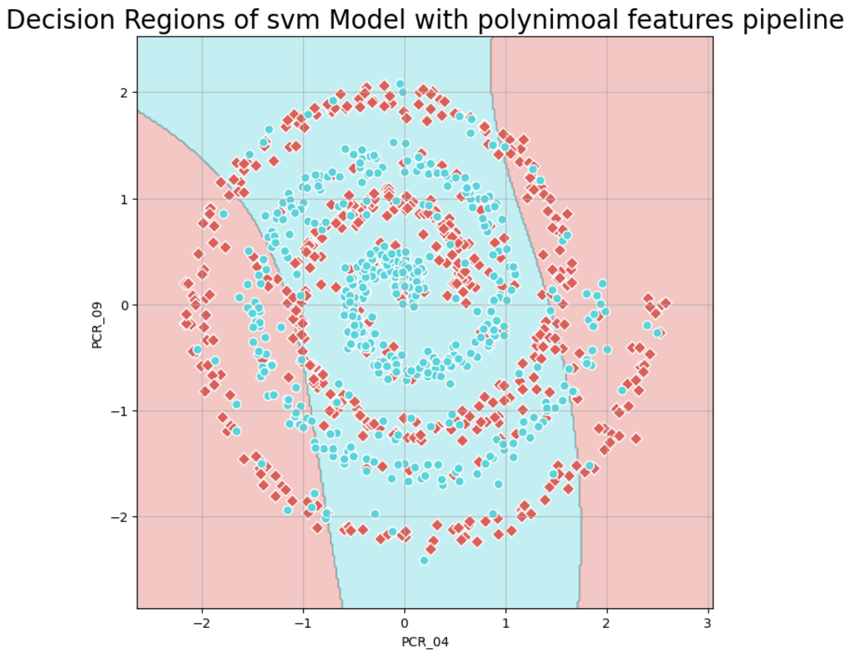
Learning rate = 1.e-05



Learning rate = 1.e-03



(Q12)



a.

b. The training accuracy is 0.605

The test accuracy is 0.6

Part 4: The RBF kernel

(Q13)

a. Given α s.t $\forall i \in [l, m], \alpha_i = 1$, write the prediction rule $h(x)$,

$$h(\boldsymbol{x}) = \operatorname{sign}\left(f(x)\right) = \operatorname{sign}\left(\sum_{i=1}^{m} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x})\right) = \operatorname{sign}\left(\sum_{i=1}^{m} y_i K(\boldsymbol{x}_i, \boldsymbol{x})\right)$$
$$= \operatorname{sign}\left(\sum_{i=1}^{m} y_i \exp\left(-\gamma \left|\left|x_i - x\right|\right|_2\right)\right)$$

b. $h(\boldsymbol{x}) = \operatorname{sign}(\sum_{i=1}^{m} y_i K(\boldsymbol{x}_i, \boldsymbol{x})) =$

$$\operatorname{sign}\left(\sum_{j\in\{i\in[m]:y_i=1\}}^{m} \overbrace{y_i}^{1} K(\boldsymbol{x}_i, \boldsymbol{x}) + \sum_{j\in\{i\in[m]:y_i=-1\}}^{m} \overbrace{y_i}^{-1} K(\boldsymbol{x}_i, \boldsymbol{x})\right)$$
$$= \operatorname{sign}\left(\sum_{j\in\{i\in[m]:y_i=1\}}^{m} y_i K(\boldsymbol{x}_i, \boldsymbol{x}) - \sum_{j\in\{i\in[m]:y_i=-1\}}^{m} y_i K(\boldsymbol{x}_i, \boldsymbol{x})\right)$$

c. There is $s > 1$ s.t $\forall i \neq j \in [m]$: $\left|\left|x_i - x_j\right|\right|_2 > 3\delta$

$\forall x \in D$ there is $(x_i, y_i) \in s$ s.t: $y = y_i$ and $\left|\left|x_i - x_j\right|\right|_2 < \delta - 1$

Given $(x, y = 1)$ prove: $\sum_{j\in\{i\in[m]:\, y_i=1\}} K(x, x_j) > \exp\{-\gamma(\delta - 1)\}$

Assign x_l as the clean in s that for which exists $y_l = y = 1$,

$$\sum_{j\in\{i\in[m]:\, y_i=1\}} K(x, x_j) \underset{*}{\geq} K(x, x_l) = \exp\left\{-\gamma ||x - x_l||_2\right\} \underset{**}{>} \exp\{-\gamma(\delta - 1)\}$$

* K is non-negative, sum of parts, inequality greater than or equal to the sum of its parts.

** Increasing the absolute value of a negative evaluator $\Rightarrow$ decrease the exponent.

d. Prove: $\sum_{j\in\{i\in[m]:\, y_i=-1\}} K(x, x_j) \leq \frac{m}{2} \exp\{-\gamma(2\delta - 1)\}$

$$\sum_{j\in\{i\in[m]:\, y_i=-1\}} K(x, x_j) = \sum_{j\in\{i\in[m]:\, y_i=-1\}} \exp\left\{-\gamma \left|\left|x_j - x_l\right|\right|_2\right\}$$

Given $\forall j \in \{i \in [m] : \; y_i = -1\} :\; \left|\left|x - x_j\right|\right|_2 > 3\delta$

$$\left|\left|x_j - x_l\right|\right|_2 = \left|\left|x_j - x + x - x_l\right|\right|_2 \underset{\text{א"ש המשולש}}{\geq} \left|\left|x_j - x\right|\right|_2 + ||x - x_l||_2 > 3\delta$$

$$\Rightarrow \underbrace{\left|\left|x_j - x\right|\right|_2}_{==\left|\left|x - x_j\right|\right|_2} > 3\delta - \underbrace{\left|\left|x_j - x_l\right|\right|_2}_{\leq \delta - 1} > 3\delta - (\delta - 1) = 2\delta + 1 > 2\delta - 1$$

$$\sum_{j\in\{i\in[m]:\, y_i=-1\}} K(x, x_j) \underset{*}{\leq} \sum_{j\in\{i\in[m]:\, y_i=-1\}} \exp\{-\gamma(2\delta - 1)\} \underset{**}{=} \frac{m}{2} \exp\{-\gamma(2\delta - 1)\}$$

* Increasing the absolute value of a negative evaluator

** the parts of the sum are not Linear dependent in j

e. Given $\gamma = \ln\left(\frac{m}{2}\right)$, show that $h(x) = 1$.

$$h(x) = \operatorname{sign}\left(\sum_{j\in\{i\in[m]:y_i=1\}}^{m} K(\boldsymbol{x},\boldsymbol{x_j}) - \sum_{j\in\{i\in[m]:y_i=-1\}}^{m} K(\boldsymbol{x},\boldsymbol{x_j})\right)$$

let's look at the sums,

$$\underbrace{\sum_{j\in\{i\in[m]:y_i=1\}}^{m} K(\boldsymbol{x},\boldsymbol{x_j})}_{>\exp(-\gamma(\delta-1))} - \underbrace{\sum_{j\in\{i\in[m]:y_i=-1\}}^{m} K(\boldsymbol{x},\boldsymbol{x_j})}_{\leq\frac{m}{2}\exp\{-\gamma(2\delta-1)\}} > \exp(-\gamma(\delta-1)) - \frac{m}{2}\exp\{-\gamma(2\delta-1)\}$$

$$= \exp\left(-\ln\left(\frac{m}{2}\right)(\delta-1)\right) - \underbrace{\frac{m}{2}\exp\left\{-\ln\left(\frac{m}{2}\right)(2\delta-1)\right\}}_{=\left(\exp\left\{-ln\left(\frac{m}{2}\right)\cdot(\delta-1)\right\}\right)^2}$$

$$= \exp\left(-\ln\left(\frac{m}{2}\right)(\delta-1)\right) - \left(\exp\left\{-ln\left(\frac{m}{2}\right)\cdot(\delta-1)\right\}\right)^2 =$$

$$= \underbrace{\exp\left(-\ln\left(\frac{m}{2}\right)(\delta-1)\right)}_{positive} \cdot \underbrace{\left(1 - \underbrace{\exp\left\{\underbrace{-\underbrace{ln\left(\frac{m}{2}\right)}_{positive}\cdot\underbrace{(\delta-1)}_{positive}}_{negative}\right\}}_{<1}\right)}_{positive} > 0$$

Hence, $h(x) = \operatorname{sign}(\textit{positive expression}) = 1$

f. **Prove the General Prediction Rule:** Based on the previous proofs, show that the prediction rule $h(x)$ is always correct under the given assumptions.

- $h(x)$ is always correct $\Leftrightarrow$ $\forall(x,y)\in D: h(x)=y$
- we've already shown: $\forall(x,y=1): h(x)=1$
- we need to prove: $\forall(x,y=-1): h(x)=-1$

for $(x,y=-1)$:

1. Assign x_l as the clean in s that for which exists $y_l = y = 1$,

$$\sum_{j\in\{i\in[m]:\,y_i=-1\}} K(x,x_j) \underset{*}{\geq} K(x,x_l) = \exp\left\{-\gamma\|x-x_l\|_2\right\} \underset{**}{>} \exp\{-\gamma(\delta-1)\}$$

* K is non-negative, sum of parts, inequality greater than or equal to the sum of its parts.

** Increasing the absolute value of a negative evaluator ⇒ decrease the exponent.

2. $\sum_{j\in\{i\in[m]:\,y_i=-1\}} K(x,x_j) = \sum_{j\in\{i\in[m]:\,y_i=-1\}} \exp\left\{-\gamma\left\|x_j-x_l\right\|_2\right\}$

Given $\forall j\in\{i\in[m]:\ y_i=-1\}:\ \left\|x-x_j\right\|_2 > 3\delta$

$$\left\|x_j-x_l\right\|_2 = \left\|x_j-x+x-x_l\right\|_2 \overset{\geq}{\text{א"ש המשולש}} \left\|x_j-x\right\|_2 + \|x-x_l\|_2 > 3\delta$$

$$\Rightarrow \underbrace{\left\|x_j-x\right\|_2}_{==\left\|x-x_j\right\|_2} > 3\delta - \underbrace{\left\|x_j-x_l\right\|_2}_{\leq\delta-1} > 3\delta-(\delta-1) = 2\delta+1 > 2\delta-1$$

$$\sum_{j\in\{i\in[m]:\,y_i=-1\}} K(x,x_j) \underset{*}{\leq} \sum_{j\in\{i\in[m]:\,y_i=-1\}} \exp\{-\gamma(2\delta-1)\} \underset{**}{=} \frac{m}{2}\exp\{-\gamma(2\delta-1)\}$$

* Increasing the absolute value of a negative evaluator

** the parts of the sum are not Linear dependent in j

3. $h(x) = \text{sign}\left(\sum_{j\in\{i\in[m]:y_i=-1\}}^{m} K(\boldsymbol{x},\boldsymbol{x_j}) - \sum_{j\in\{i\in[m]:y_i=1\}}^{m} K(\boldsymbol{x},\boldsymbol{x_j})\right)$

   let's look at the sums,

$$\underbrace{\sum_{j\in\{i\in[m]:y_i=-1\}}^{m} K(\boldsymbol{x},\boldsymbol{x_j})}_{\le\frac{m}{2}\exp\{-\gamma(2\delta-1)\}} - \underbrace{\sum_{j\in\{i\in[m]:y_i=1\}}^{m} K(\boldsymbol{x},\boldsymbol{x_j})}_{>\exp(-\gamma(\delta-1))} \underset{by\ subsection}{<} 0$$

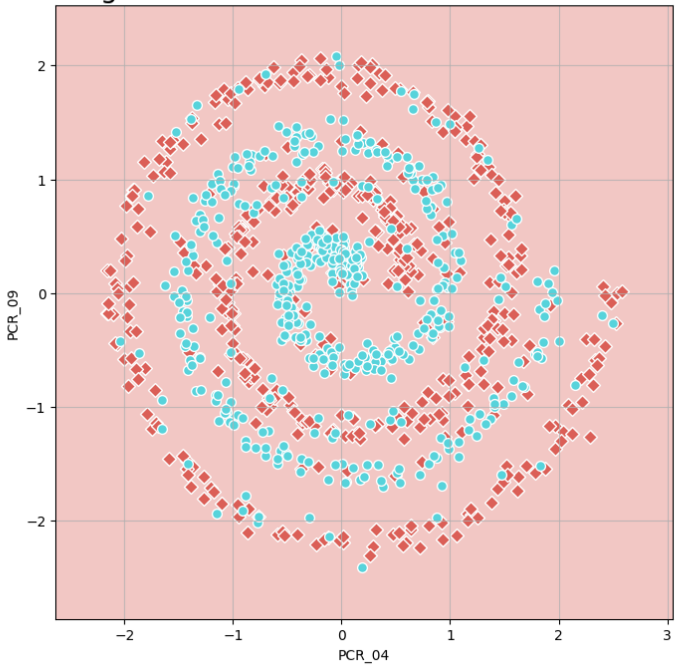
   Hence, $h(x) = \text{sign}(\textit{negative expression}) = 1$

$\Rightarrow$ **$h(x)$ is always correct**

(Q14)

Training an SVM with an RBF kernel with c=1 and gamma=1e-7 gave us the following decision regions:

Decision Regions of SVM wih RBF kernel with Gamma = 1e-7

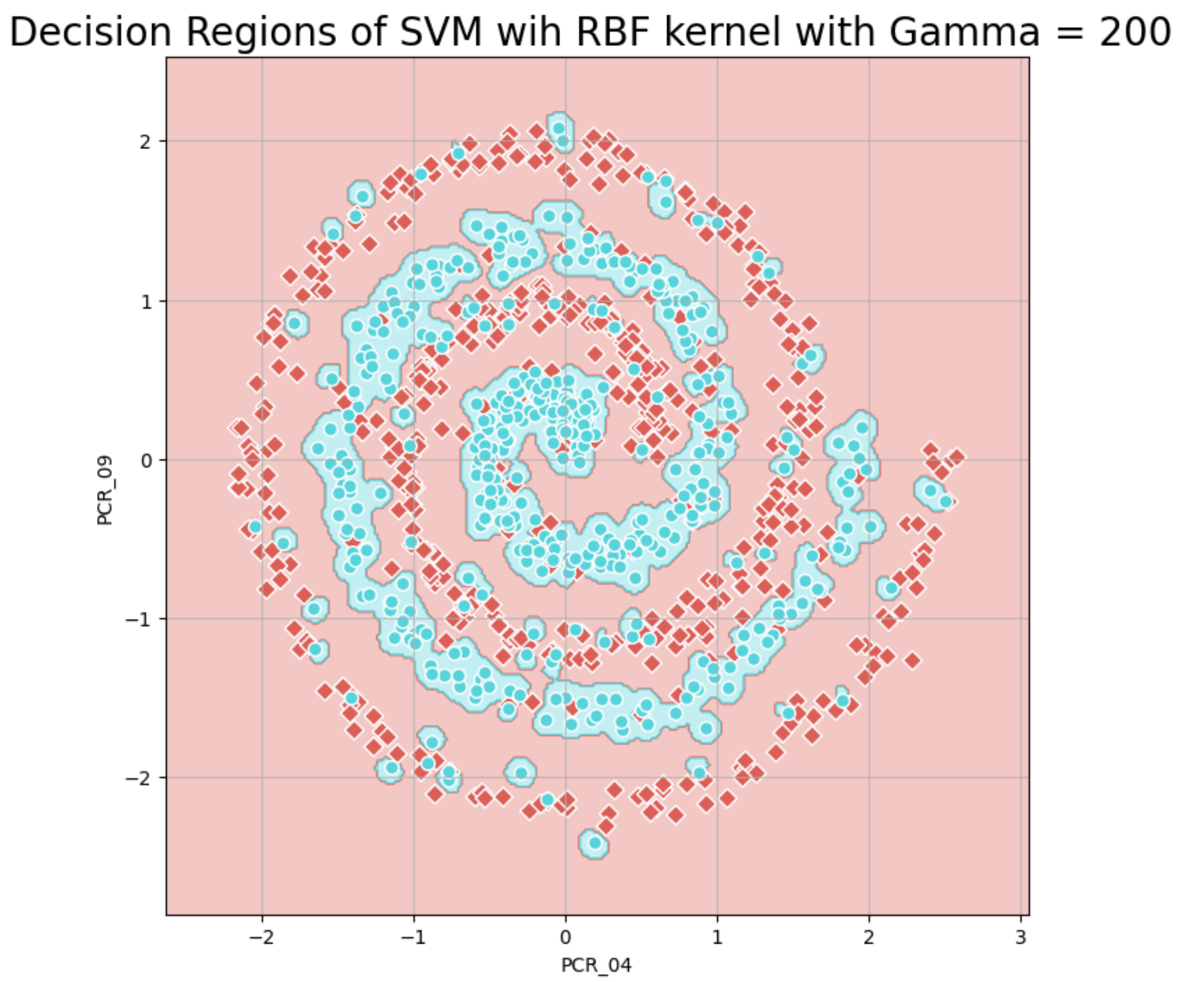


The training accuracy is 0.512 and the test accuracy is 0.452.

Based on these accuracies, which are low and indicate poor performance of the model on both the train and test datasets, we can conclude that the mode doesn't capture the data well, hence **we would classify the model as underfitting**.

(Q15)

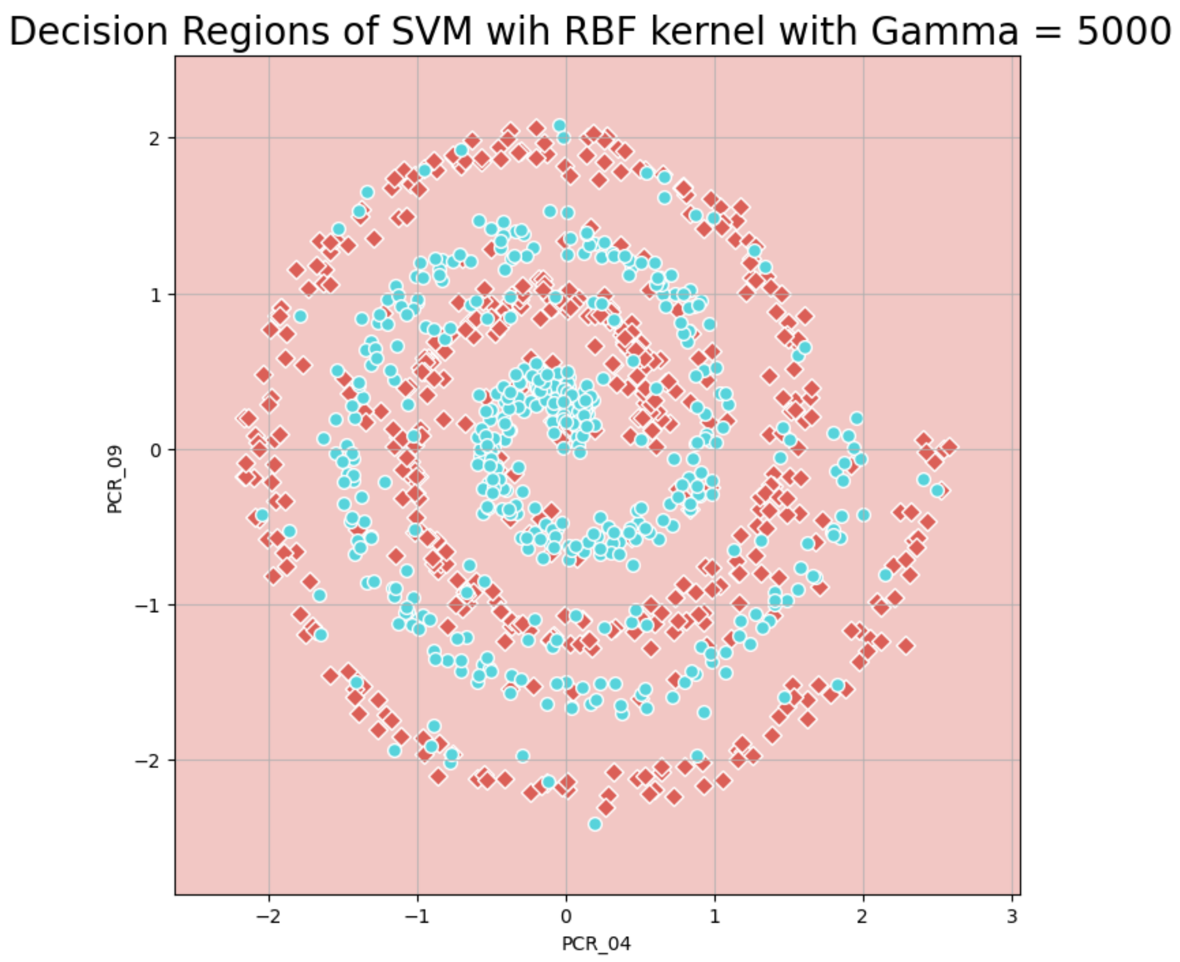
Training an SVM with an RBF kernel with c=1 and gamma=200 gave us the following decision regions:



The training accuracy is 0.953 and the test accuracy is 0.728.
The decision boundaries are similar but we can see that the11-NN's boundaries are more generalized, we have only 2 blue areas, but the SVM's boundaries are very specific and we have many more blue areas. The reason for the difference could be because of not very small values of the second norma produce overly complex SVM's decision boundary which is sensitive to individual points.

(Q16)

Training an SVM with an RBF kernel with c=1 and gamma=5000 gave us the following decision regions:



The training accuracy is 0.999 and the test accuracy is 0.5
Based on these accuracies, which are very high for the train set and low for the test set, that indicates great performance only on the train set. We can conclude that the mode doesn’t capture very well the training data and doesn’t generalize well, hence **we would classify the model as overfitting**.