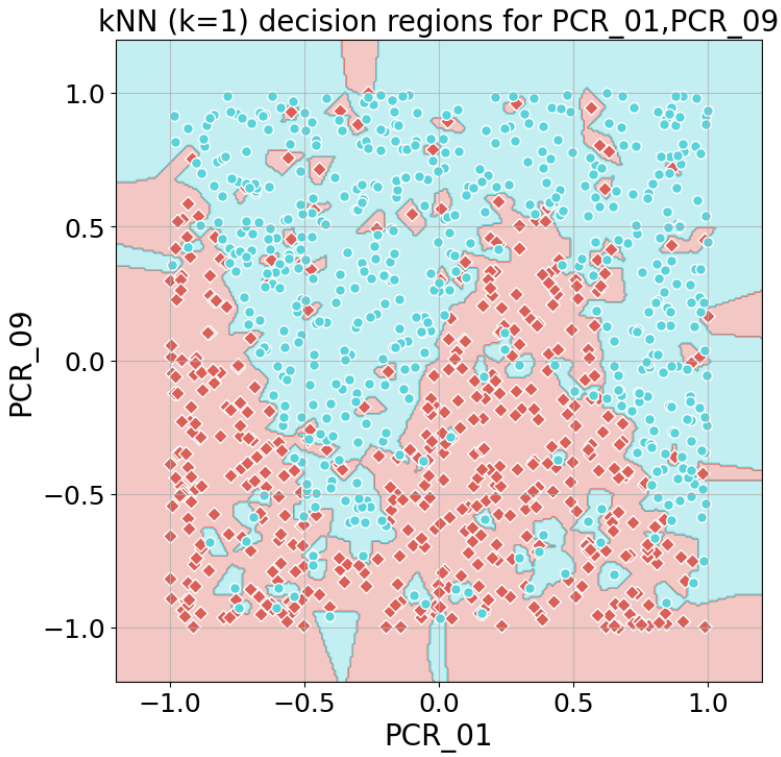
Major 2 Report

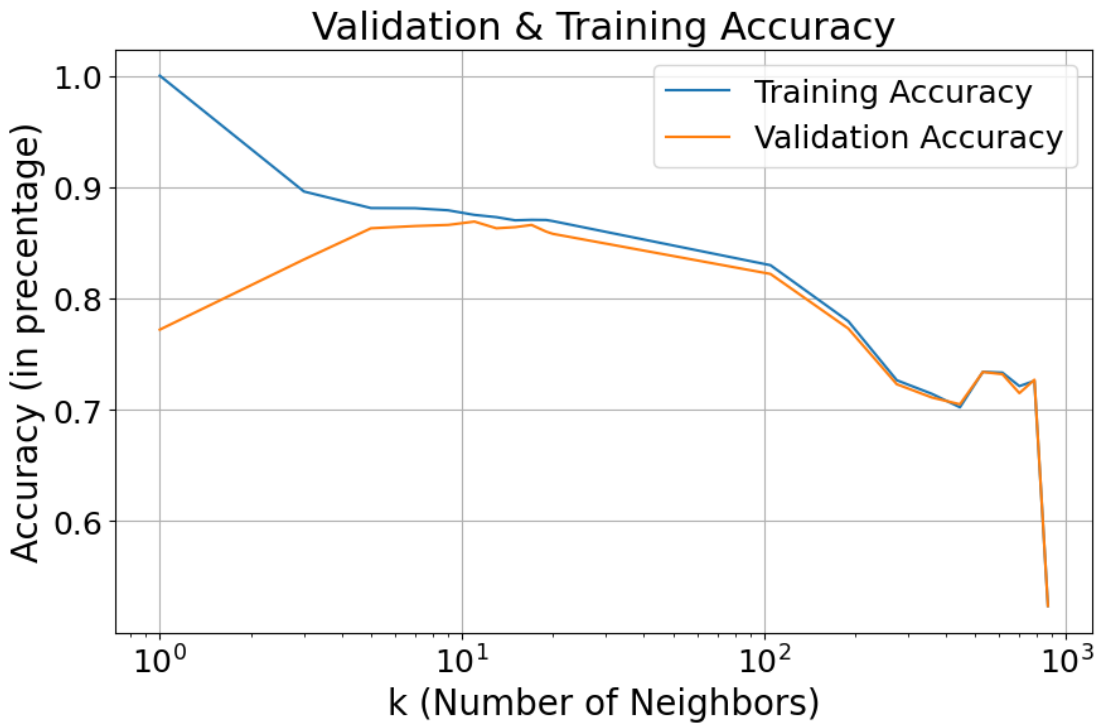
Idan Albo – 206665051, Amit Grosman - 318170214

Part 1 – Basic model selection with k-Nearest Neighbors

Q1.



Q2.



We chose the best k, according to the highest validation accuracy (lowest validation error) as we saw in the lecture. K=11 is the best. Its mean training accuracy is 87.5%, and its validation accuracy is 86.9%.

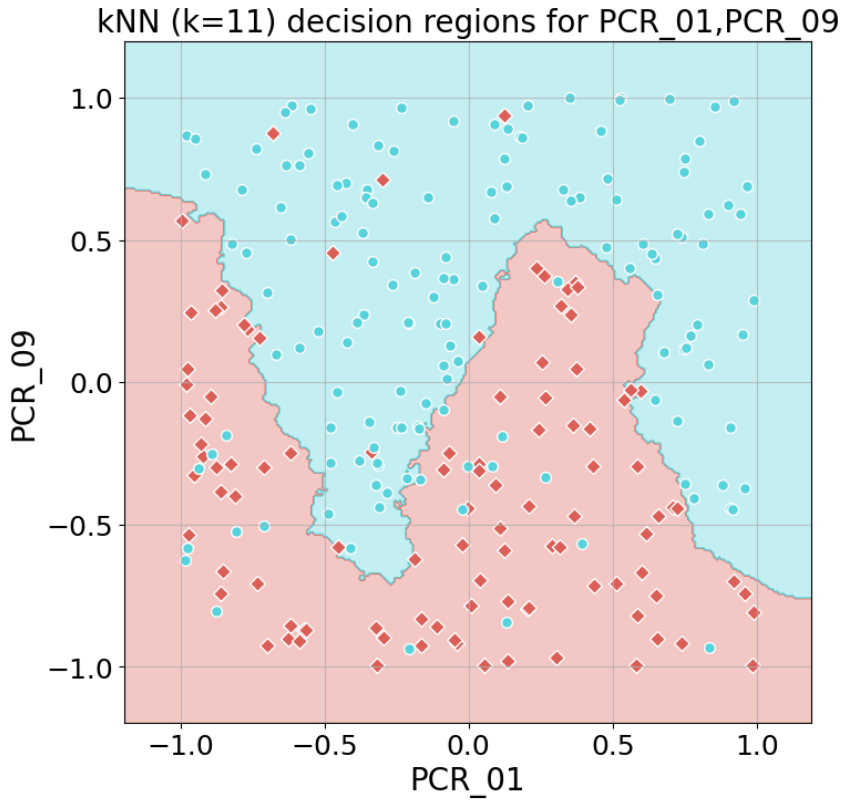
The k values range that cause overfitting are $k \in [1,5]$, because for k in this range we receive very high mean training accuracy, and low accuracy for the validation, meaning that our model fitted the training sample set rather than the actual distribution.
and the k values range that cause underfitting are $k > 100$, because for k in this range we receive low mean training accuracy and low validation accuracy.

Q3.

As we increase k, the decision boundaries become smoother (and to some extent – more generalized) and we expect the model to predict points based on larger surroundings, causing more dependence on the sampled data (specifically, on their general scattering in the hyperplane. When $k \to |S|$ the model will predict every point to the most common label).
Also, the model's variance will decrease because it's prediction is resilient to small changes of a point (to predict / as part of the training set), due to considering larger set of neighbors. This attribute cause h_s to vary less.

As we decrease k, the decision boundaries fit well the training data, which can lead to modeling the true distribution better – meaning lower bias.
Also, the model's variance will increase because the model is sensitive to small changes in points of the training data, due to fitting the decision boundaries more precise the training data.

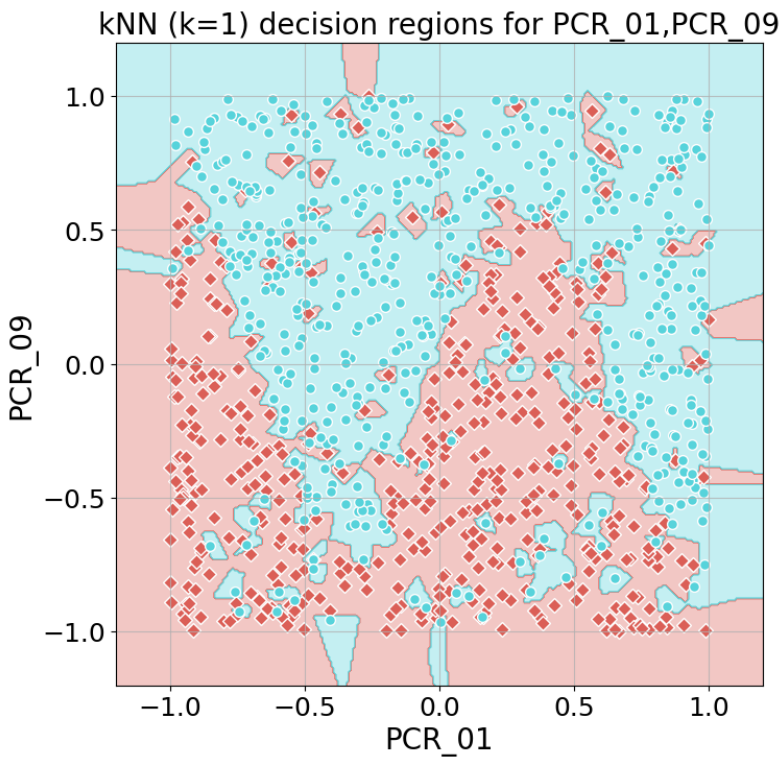
Q4.



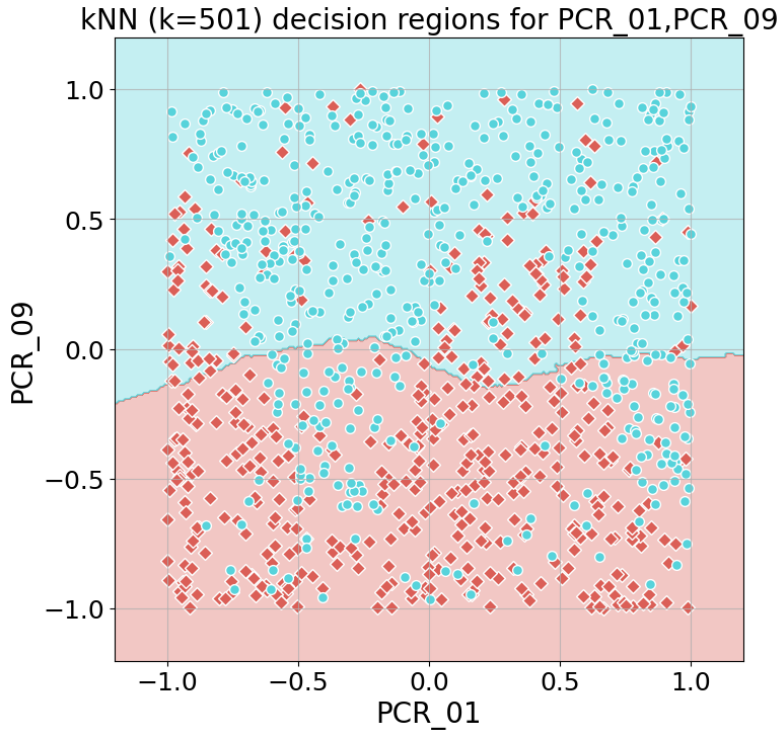
The test accuracy of this model is 88.4%

Q5.

For the kNN model with k=1, we receive the following decision regions:



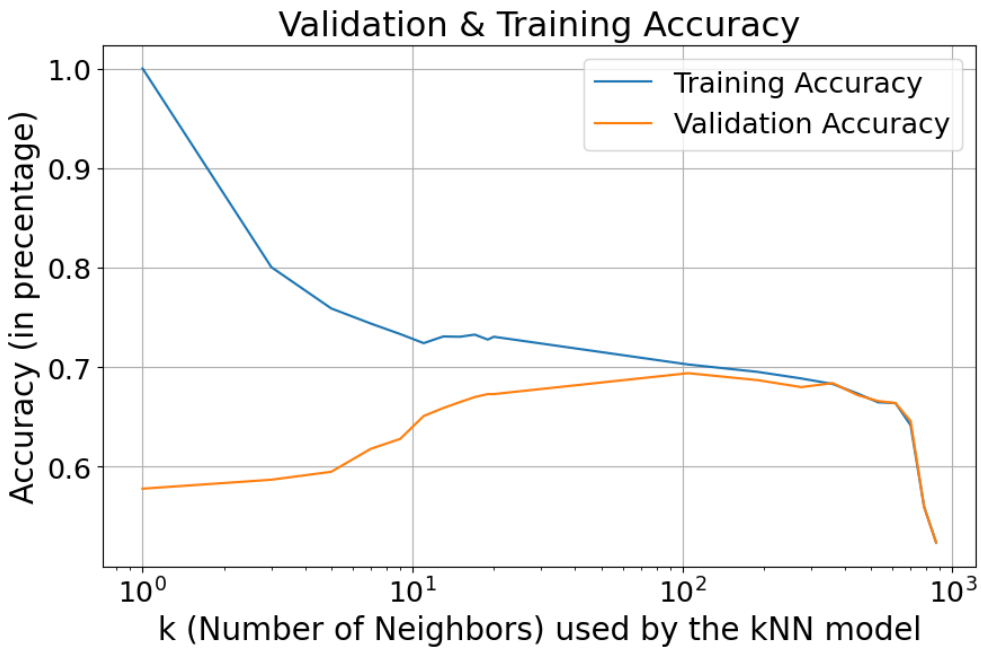
For the kNN model with k=501, we receive the following decision regions:



Comparing to the optimal k (k=11), we can see that for k=1 the boundaries resemble the optimal k boundaries, however it is noise sensitive (and therefore overfitting). As a result, we can see "islands" in the decision regions according to the sample test data.

For k=501, we get a "smooth" boundary for the decision regions, however for such a large neighbors cluster (considering our sample test set size), we are dependent on half of the sample test set. As a result, the model suffers from underfitting.

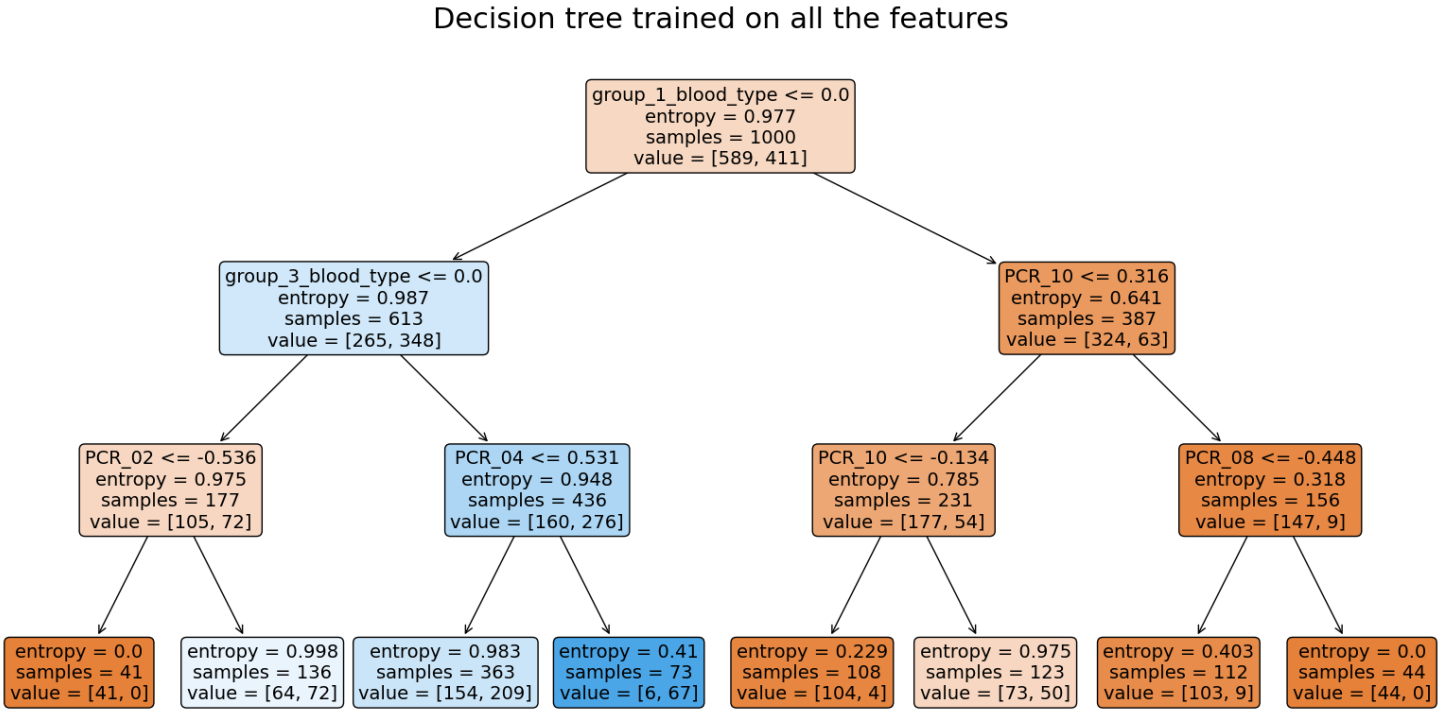
Q6.



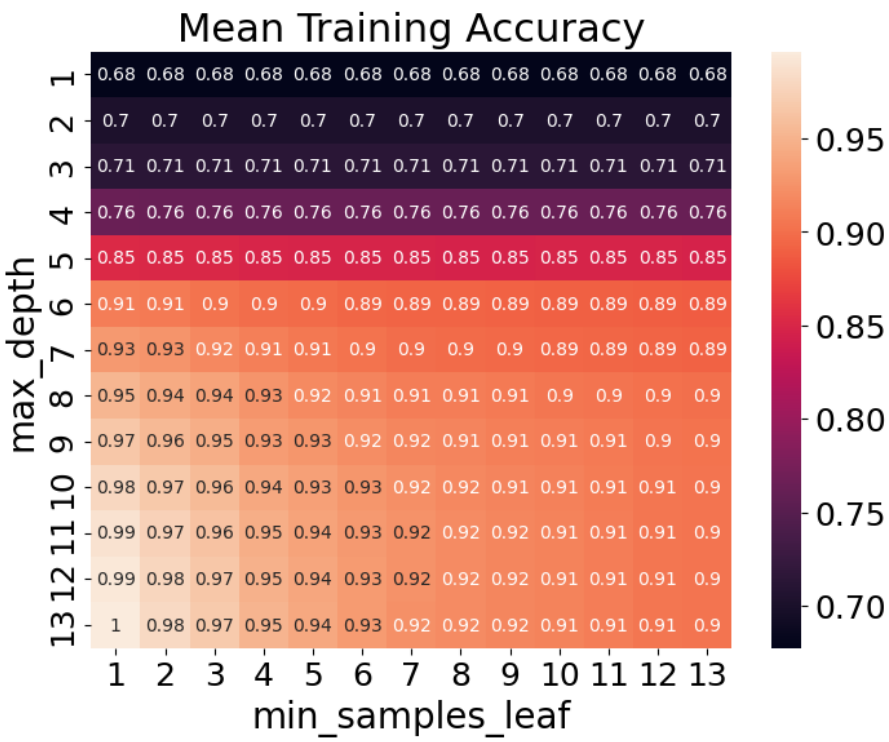
Using all the features from the training dataset (instead of 'PCR_01','PCR_09') we receive mush higher optimal k, and it's easy to notice that the model lost training and validation accuracy significantly.
This can caused by adding many features that does not have clear decision regions (as we saw for some of them in HW1) for the kNN model, with features that has clear decision regions.
By taking them into account together, we considering all features **equally**. Effectively, reducing the importance of features with the distinct decision regions.

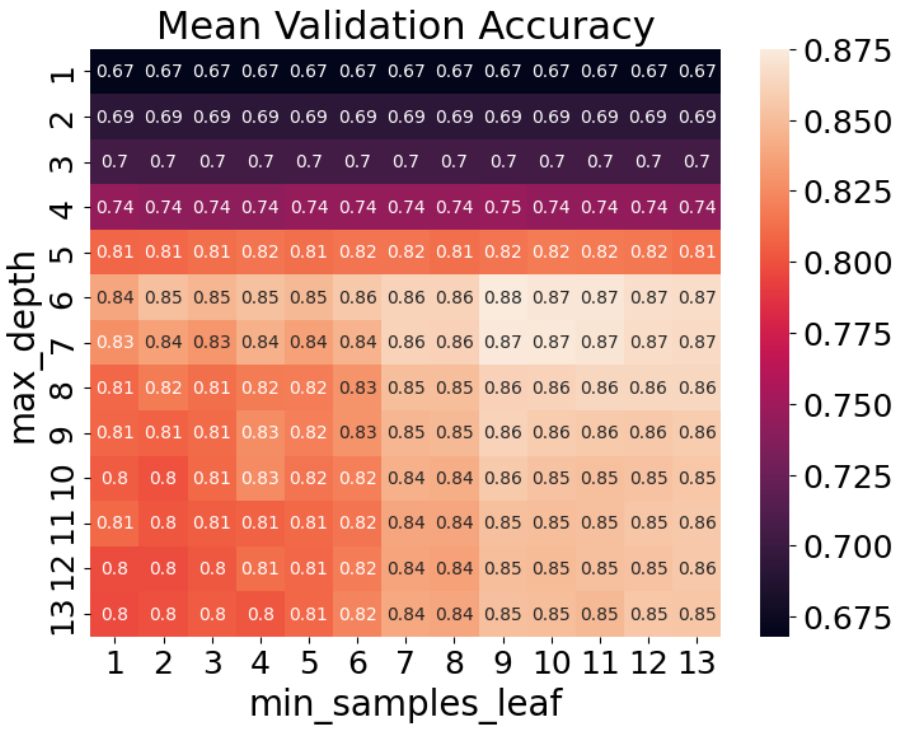
Part 2 – Decision Trees

Q7.



Q8.





c. The optimal hyperparameter combination is max_depth=6, min_samples_leaf=9, which receive mean training accuracy of 90%, and mean validation accuracy of 88% (highest validation accuracy).

d. A "classical" hyperparameter combination that cause underfitting is max_depth=1, and min_samples_leaf=13.

e. A obvious hyperparameter combination that cause overfitting is min_samples_leaf=1, and max_depth=13.

f. For the first combination, we concluded that they result in 'underfitting', because we can see from the heatmaps (both training and validation) that the accuracy is low. Also, considering our knowledge on the decision tree model, we know that setting a low max_depth might result in underfitting.

For the second combination, we conclude that they result in 'overfitting', because we can clearly see from the mean training accuracy heatmap, that this combination led to 100% accuracy for the training set, but only 80% accuracy on the validation set (compared to the training accuracy, this is relatively low), meaning that this hyperparameters combination fit the training dataset well, but perform badly on unseen dataset (validation set in this case).

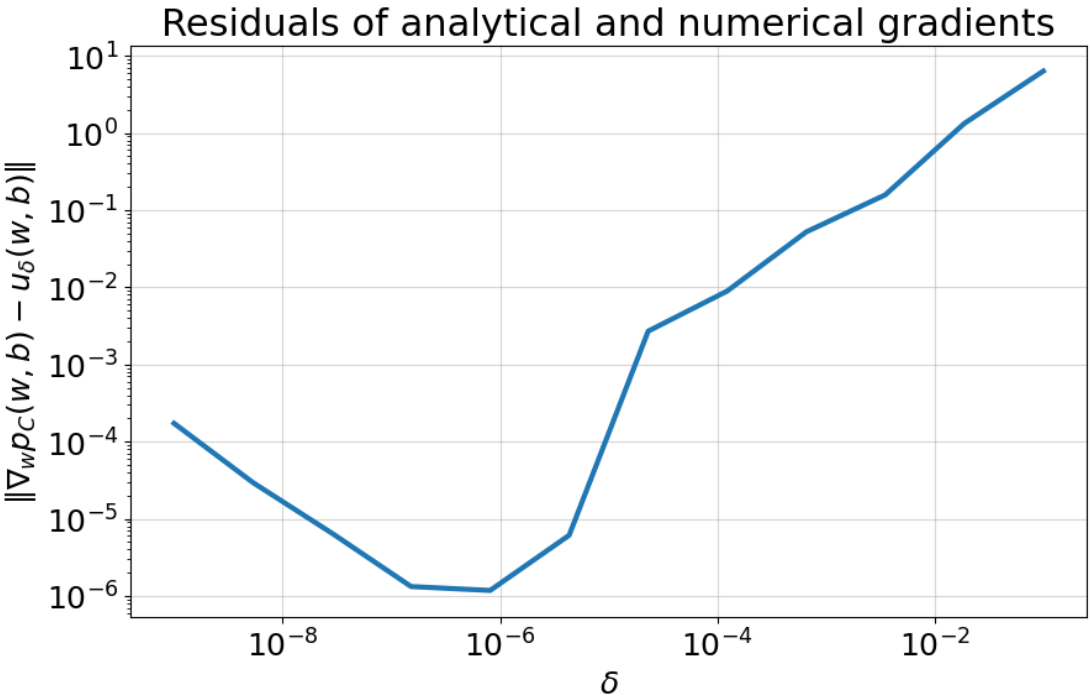
Q9.

The test accuracy of the model with the optimal hyperparameters is 83.6%

Part 3 – Linear SVM and the Polynomial kernel

Q10.

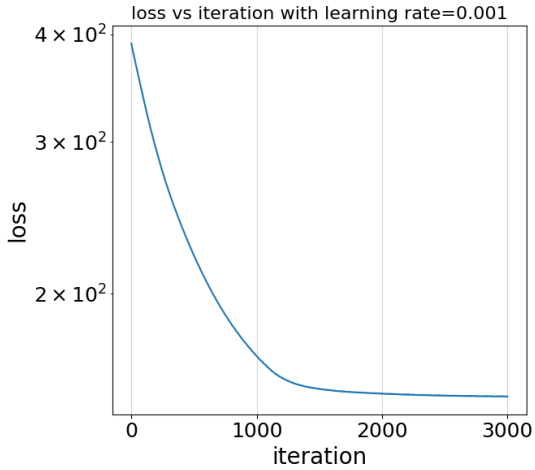
The following is the graph that shows the distance between the numerical sub-gradient and the analytic sub-gradient (with respect to δ):

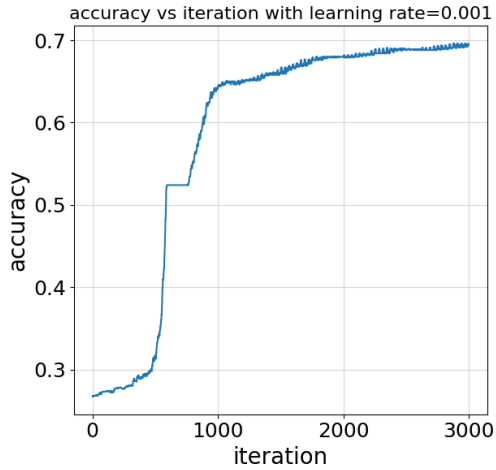


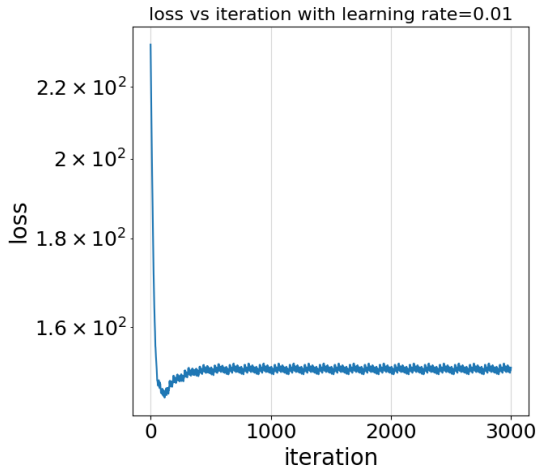
As expected, we can easily see that as we decrease δ to "infinitesimally" small values, we achieve a better approximation to the numerical sub-gradient (meaning, a closer value to the analytic sub-gradient).
That because when decreasing δ, we receive a better approximation for the derivative with respect to w_i (for all i) – that forms the sub-gradient itself.
Also, for very small value of δ (from around $\delta < 10^{-7}$), the calculations are more vulnerable to numerical errors (that depends on the computer computation accuracy abilities)

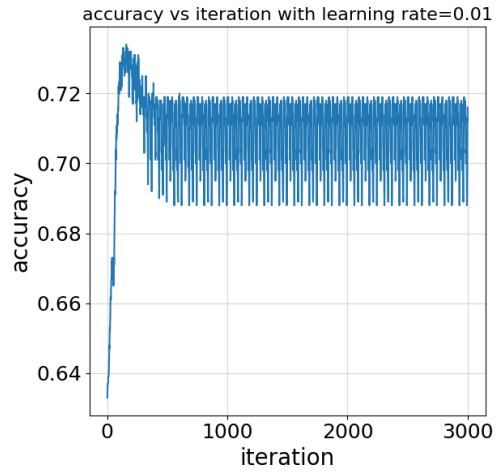
Q11.

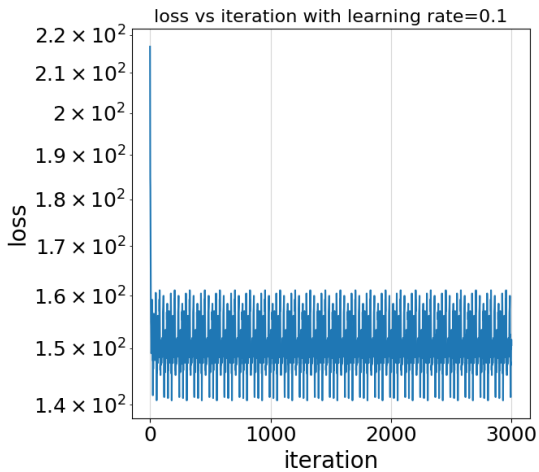
We changed c to 0.2

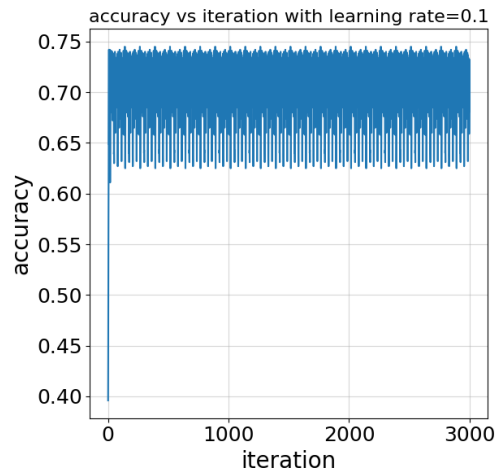


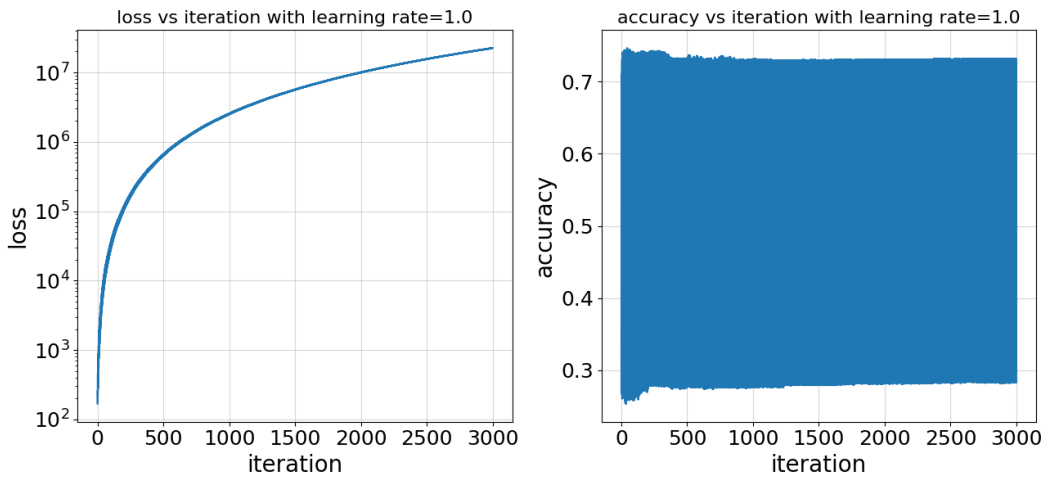






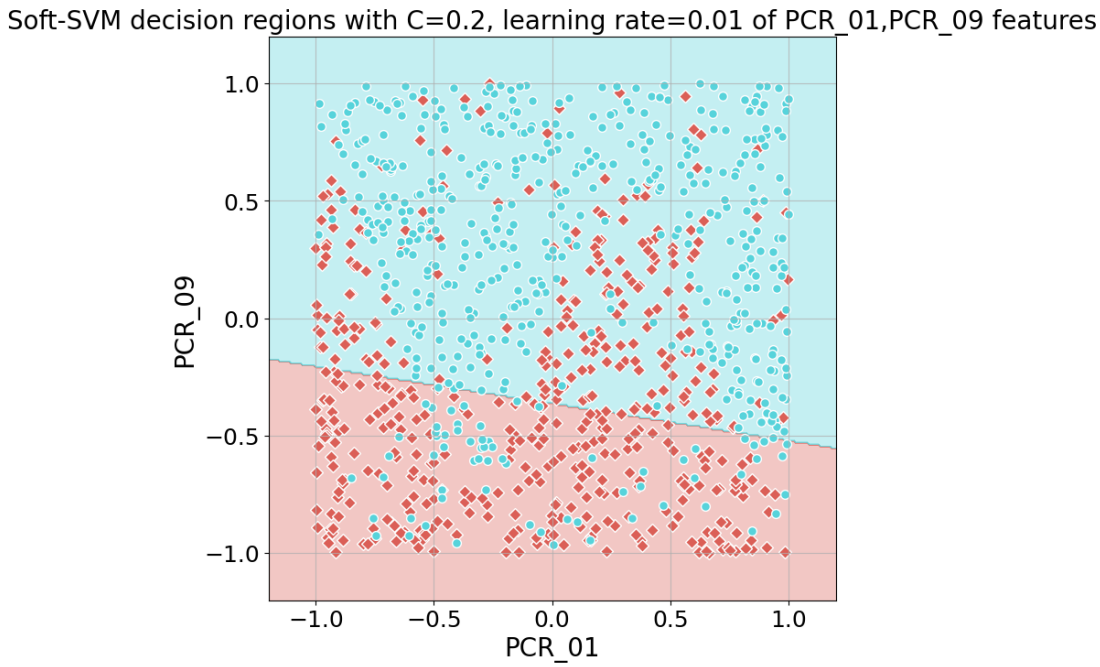






a. We'll choose the learning rate of 0.01, because we can see from the graphs that it achieves the highest accuracy, and the fastest converges of loss (with respect to number of iterations)

b.



c.

```
minimal loss acheieved on step 146 with 148.30316460192753 minimal loss
maximal accuracy acheieved on step 126 with 0.732 maximal accuracy
```

They are not attained at the same step, and that's possible because the loss is calculated as a function of the mislabeled margin distance. Hence, it's possible that the minimal loss is not achieved with the maximal accuracy (As an example, we can think of mislabel a "far" points to ensure higher accuracy, but suffer higher loss), because this is not a 0-1 loss.

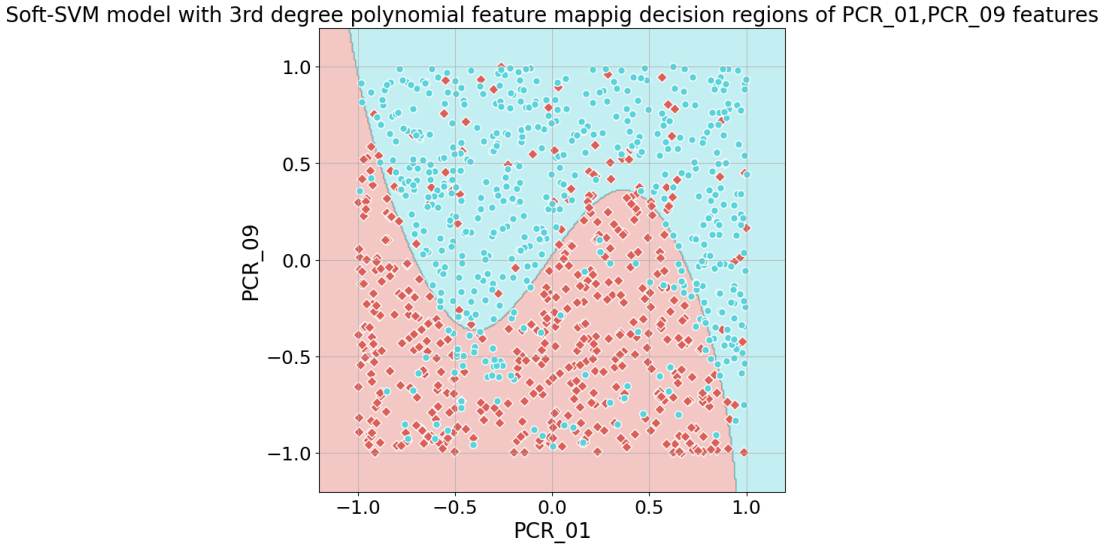
Q12.

C is a hyperparameter of the Soft-SVM model, and by changing it, we possibly change the result of the fitted w vector.
The learning rate (if we only decrease it) will change the convergence rate only, but the model will reach the same minimal loss of the one with the original learning rate (assuming we have enough iterations).

Q13.

a. Training accuracy is 83.6%, and the test accuracy is 82.8%

b.



c. The difference between the decision regions of this model, from the previous model is that this decision region separates the plane (of the PCR_01,PCR_09 features) with a polynomial instead of a linear line. This happened because we used a polynomial feature mapping.
This increases the model accuracy, because the linear model is a special case for this model (and the data is not linear separable)

Q14.

a. The 5 resulting train accuracies are: [0.836, 0.84, 0.837, 0.838, 0.838]
The mean: 0.8378
The standard deviation: 0.0013

b. First, we recall that the SGD algorithm samples a constant batch size of examples in each iteration, to calculate the loss function on them (instead of the regular GD, where we calculate the loss function on all examples) – therefore it's a random variable and we expect to see some variance.
However, we also expect that this variance will not be significant, because we know that the expected value of this loss function value on the samples should behave like the normal GD algorithm.
Also, we know that the Soft-SVM formulation is convex, therefore achieves (one) global minimum – so the model should converge to the same minimal loss value.

Part 4 – The RBF Kernel

Q15.

For simplicity, we'll assume that no two samples hold $i \neq j$ such that $\|x - x_i\| \neq \|x - x_j\|$. Denote $i^* = argmin_{i\in[m],\alpha_i>0}\|x - x_i\|_2^2$.
Since e^x is positive for every $x \in \mathbb{R}$, we can write the following (using the hint):

$$sign\left(\sum_{i\in[m],\alpha_i>0} \alpha_i y_i e^{-\gamma\|x-x_i\|_2^2}\right) = sign\left(\frac{\sum_{i\in[m],\alpha_i>0} \alpha_i y_i e^{-\gamma\|x-x_i\|_2^2}}{e^{-\gamma\|x-x_{i^*}\|_2^2}}\right)$$

$$= sign\left(\alpha_{i^*}y_{i^*} + \sum_{i\in[m]\setminus\{i^*\},\alpha_i>0} \alpha_i y_i e^{-\gamma\left(\|x-x_i\|_2^2-\|x-x_{i^*}\|_2^2\right)}\right)$$

Now, notice that because we assumed that the L_2 distances of x from the samples are different, and because $x_{i^*} = \min_{i\in[m],\alpha_i>0}\|x - x_i\|_2^2$, we conclude that:

$$\forall i \in [m] \setminus \{i^*\}: \ \|x - x_i\|_2^2 - \|x - x_{i^*}\|_2^2 > 0$$

We know that for every $z \in \mathbb{R}$ $\lim_{\gamma\to\infty} e^{-\gamma z} = 0$, so we conclude that:

$$\forall i \in [m] \setminus \{i^*\}, \alpha_i > 0: \ \lim_{\gamma\to\infty} e^{-\gamma\left(\|x-x_i\|_2^2-\|x-x_{i^*}\|_2^2\right)} = 0$$

$$\Rightarrow_{multiplying\ by\ constant} \ \forall i \in [m] \setminus \{i^*\}, \alpha_i > 0: \ \lim_{\gamma\to\infty} \alpha_i y_i\, e^{-\gamma\left(\|x-x_i\|_2^2-\|x-x_{i^*}\|_2^2\right)} = 0$$

$$\Rightarrow_{finite\ sum} \ \lim_{\gamma\to\infty} \sum_{i\in[m]\setminus\{i^*\},\alpha_i>0} \alpha_i y_i e^{-\gamma\left(\|x-x_i\|_2^2-\|x-x_{i^*}\|_2^2\right)} = 0$$

Thus, we conclude that for $\gamma \to \infty$ the following holds:

$$\lim_{\gamma\to infty} \alpha_{i^*}y_{i^*} + \sum_{i\in[m]\setminus\{i^*\},\alpha_i>0} \alpha_i y_i e^{-\gamma\left(\|x-x_i\|_2^2-\|x-x_{i^*}\|_2^2\right)} = \alpha_{i^*}y_{i^*}$$

We also know that the sign function is continuous in $(0, \infty)$ and in $(-\infty, 0)$, and because $y_{i^*} \in \{\pm 1\}$ and $\alpha_{i^*} > 0$, there exists some open ball $B(\alpha_{i^*}y_{i^*}, \epsilon)$ such that it contains in one of the segments above.
From the limit above, there exists some γ_0 such that for all $\gamma \geq \gamma_0$:

$$\left|\alpha_{i^*}y_{i^*} + \sum_{i\in[m]\setminus\{i^*\},\alpha_i>0} \alpha_i y_i e^{-\gamma\left(\|x-x_i\|_2^2-\|x-x_{i^*}\|_2^2\right)}\right| \leq \alpha_{i^*}y_{i^*} + \epsilon$$

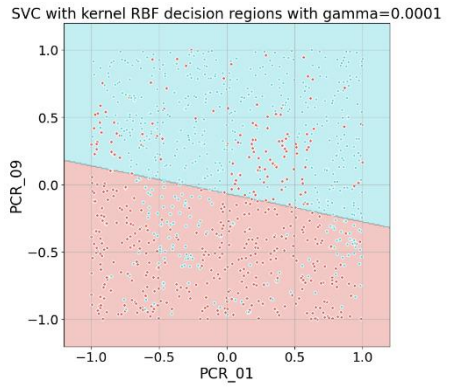
and therefore for such γ_0, we can treat the sign function (for every $\gamma \geq \gamma_0$) as a continuous function. So, for this γ's:

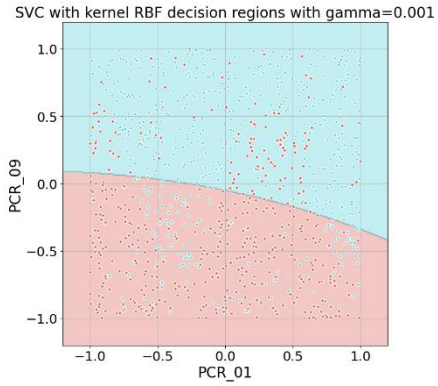
$$sign\left(\alpha_{i^*}y_{i^*} + \sum_{i\in[m]\setminus\{i^*\},\alpha_i>0} \alpha_i y_i e^{-\gamma\left(\|x-x_i\|_2^2-\|x-x_{i^*}\|_2^2\right)}\right)$$
$$= sign(\alpha_{i^*}y_{i^*}) =_{hint} sign(y_{i^*}) =_{y_{i^*}\in\{\pm 1\}} y_{i^*}$$

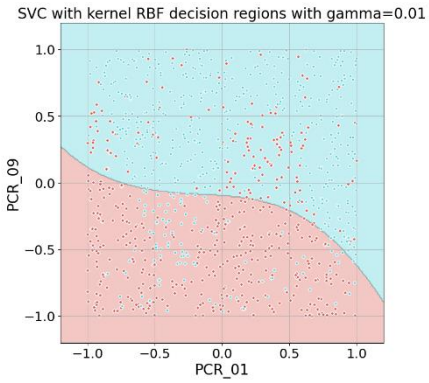
(we looked on the open ball such that $\alpha_{i^*}y_{i^*} + \epsilon$ is in the same segment as $\alpha_{i^*}y_{i^*}$).
Therefore:

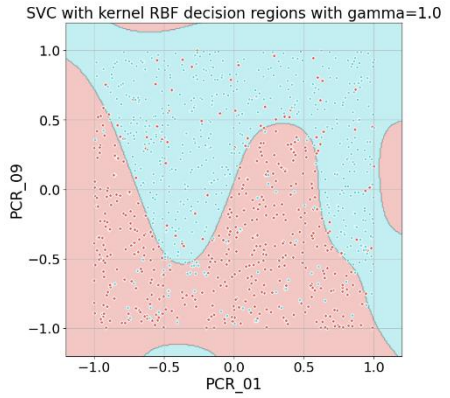
$$\lim_{\gamma\to\infty} sign\left(\sum_{i\in[m],\alpha_i>0} \alpha_i y_i e^{-\gamma\|x-x_i\|_2^2}\right) = y_{i^*}$$

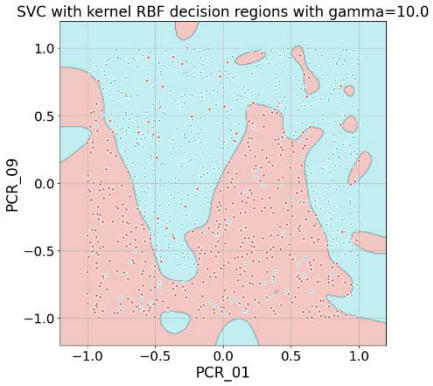
Q16.

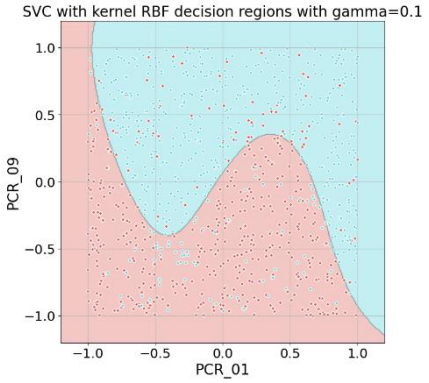

SVC with kernel RBF decision regions with gamma=0.0001
PCR_09
PCR_01

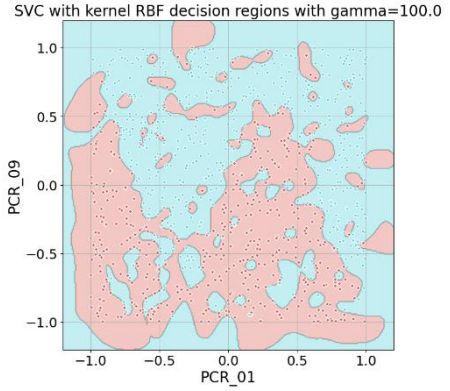

SVC with kernel RBF decision regions with gamma=0.001
PCR_09
PCR_01

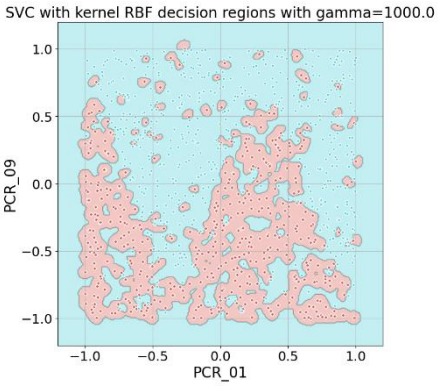

SVC with kernel RBF decision regions with gamma=0.01
PCR_09
PCR_01

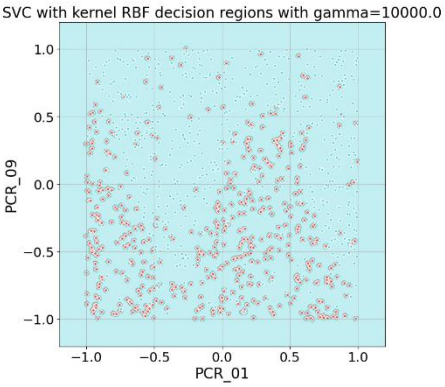

SVC with kernel RBF decision regions with gamma=1.0
PCR_09
PCR_01


SVC with kernel RBF decision regions with gamma=10.0
PCR_09
PCR_01


SVC with kernel RBF decision regions with gamma=0.1
PCR_09
PCR_01


SVC with kernel RBF decision regions with gamma=100.0
PCR_09
PCR_01

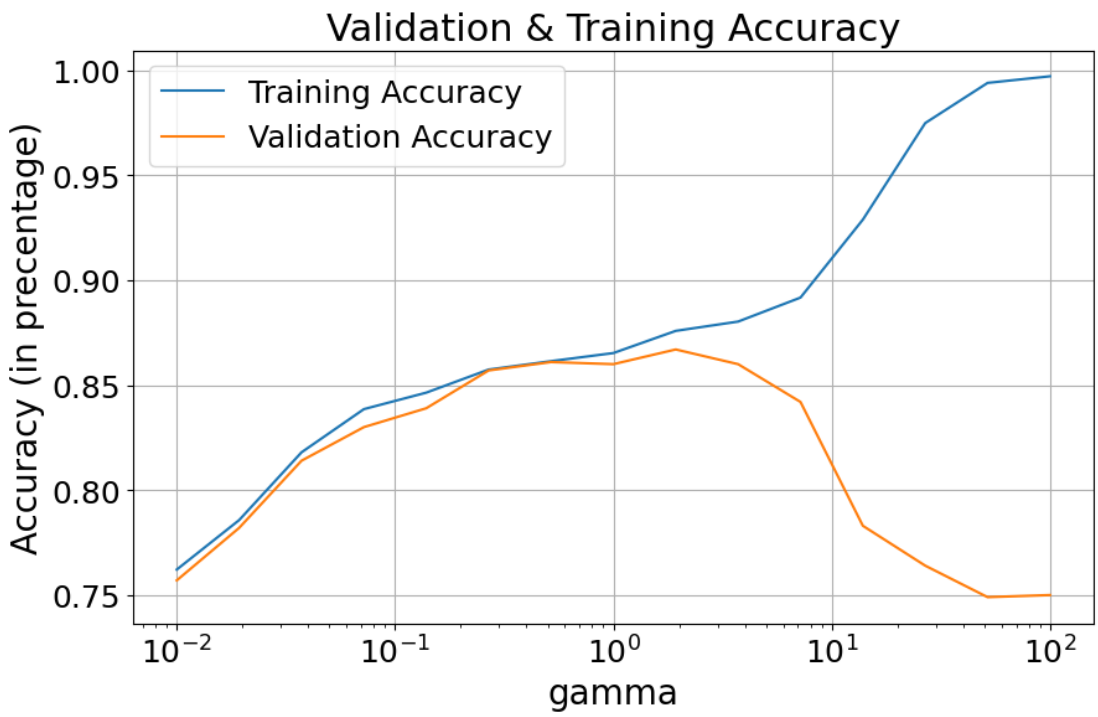

SVC with kernel RBF decision regions with gamma=1000.0
PCR_09
PCR_01


SVC with kernel RBF decision regions with gamma=10000.0
PCR_09
PCR_01

Q17.

The decision regions of the kNN model with k=1 from Q1 is similar to the RBF model with $\gamma = 10^4$ by their major decision boundaries (by rough estimation), and considering "islands" around single points with red label.
However, we notice that in the RBF model, it chooses for the most parts the "default" tagging (+1) – that could results from numerical errors when using large γ (hence, small value for $e^{-\gamma||x_1-x_2||_2^2}$), and the outline of the red decision regions is surrounded by the blue decision region (although the "red" points are closer, unlike kNN model with k=1).

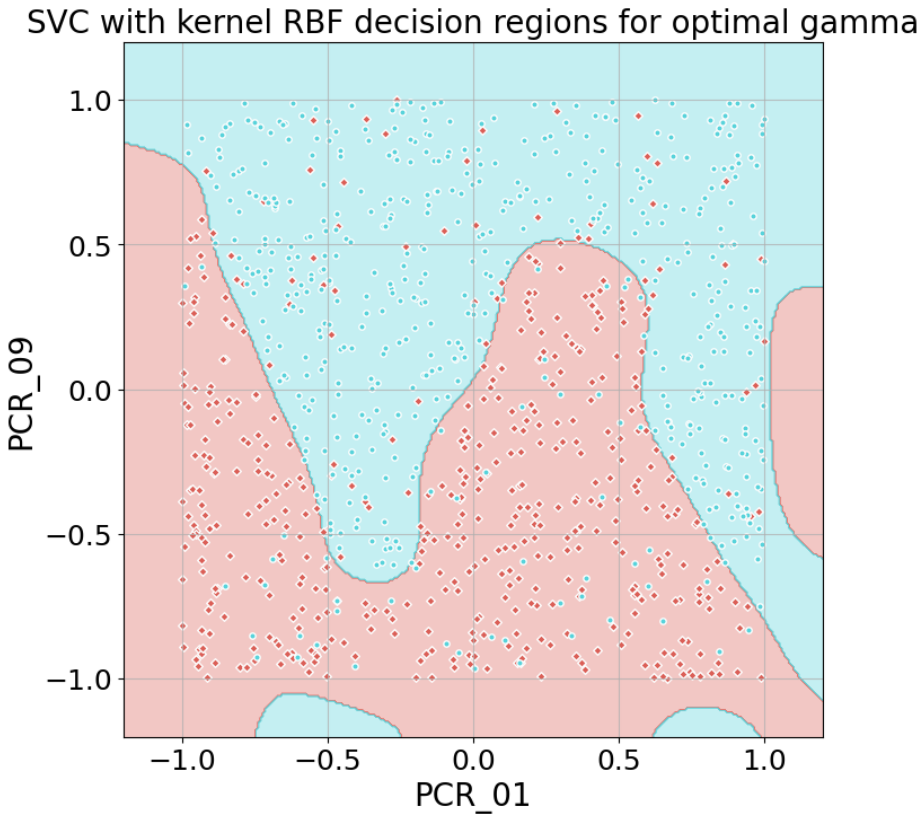
Q18.



We chose the best γ, according to the highest validation accuracy (lowest validation error) as we saw in the lecture. $\gamma = 1.93$ is the best. Its mean training accuracy is 87.5%, and its validation accuracy is 86.7%.

The γ values range that cause overfitting are $\gamma > 8$, because for γ in this range we receive very significant difference between the mean training accuracy and the mean validation accuracy (also, relatively low mean validation accuracy), meaning that our model fitted the training sample set rather than the actual distribution.
and the γ values range that cause underfitting are $\gamma < \frac{1}{2} \cdot 10^{-1}$, because for γ in this range we receive low mean training accuracy and low validation accuracy (also, we saw in the previous question that setting γ in this range resulted in decision regions that underfits).

Q19.



The test accuracy is 88.4%.
The model's decision regions are similar to the decision regions of the kNN model from Q4 (with k=11). We observe that the RBF model has smoother decision boundary than the kNN one.
Also, the kNN model splits the plane into 2 distinct decisions areas (without enclaves – like in the RBF model).
Based on the test accuracy, the kNN model predicts the spread label with 88.4% accuracy, and the RBF model also predicts the spread label with 88.4% accuracy – so it's hard to tell.