# Introduction to Machine Learning (02360766)

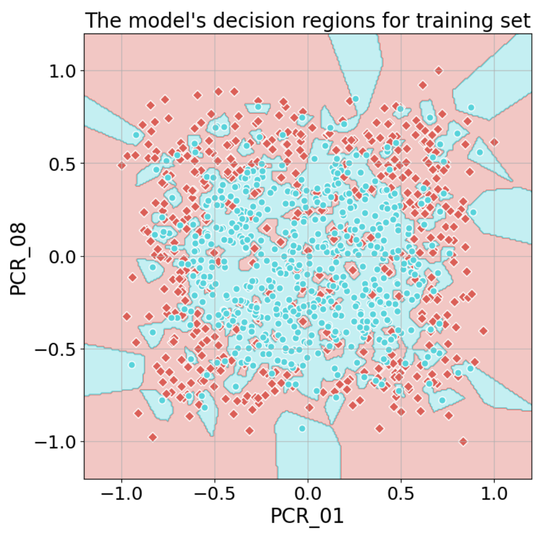
Homework 2

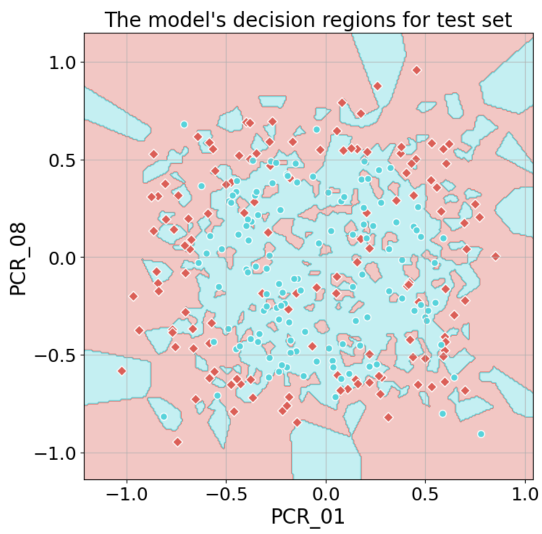
Submitted by: Gal Porat, Eyal Amdur

ID: 318849270, 315116095

## Part 1: Basic model selection with k-Nearest Neighbors

**(Q1) Training set score: 1.0** **Test set score: 0.7**



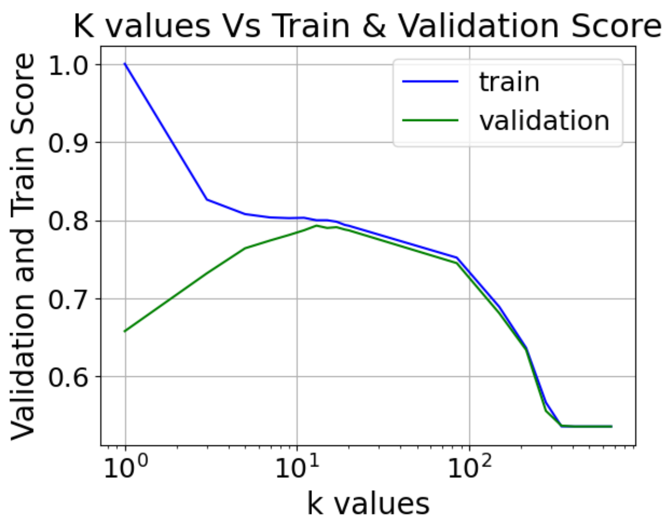


**(Q2)**

From the attached plot we can see that the K that provides the maximal validation is K=13

With:

```
K: 13 Validation score: 0.793 Train score: 0.7998571428571429
```

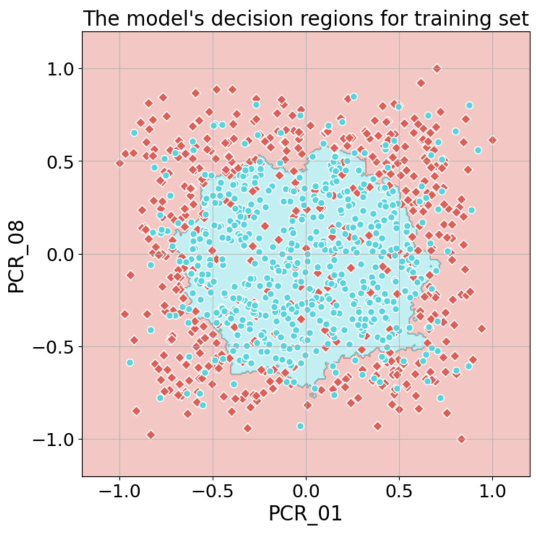


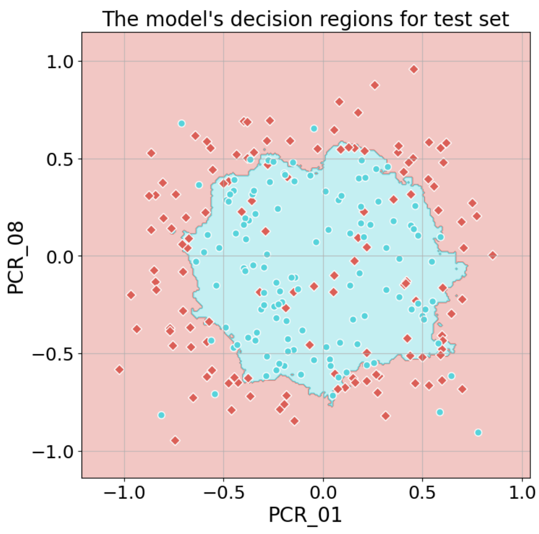
Using low values of k, such as k = 1 leads to overfitting, as shown in the plot where the training score is perfect at 1. This happens because, with k = 1, each sample is considered as its own nearest neighbour, which gives us perfect fit to the training data without allowing the model to generalize effectively.

On the other hand, using high values of k, such as k = 670 results in underfitting due to the considerably low training score. A model with k = 670 considers a large amount of samples as its neighbours, which diminishes the affect of the closest neighbours, ultimately resulting in a failure to capture the patterns in the data.

**(Q3)** For K=13 (Best k from Q2) the results are:

**Training set score: 0.8** **Test set score: 0.776**



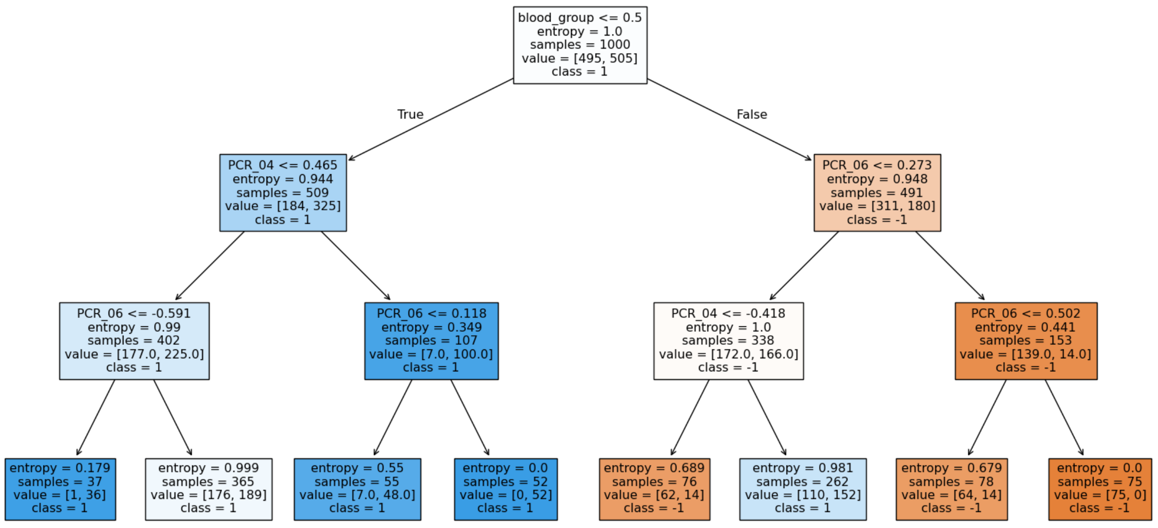


**(Q4)** In Question 1, the test score was 0.7 when k was 1, whereas it increased to 0.776 when k is 13. As mentioned before, using k = 1 results in overfitting, which consequently leads to a lower score on the test set.

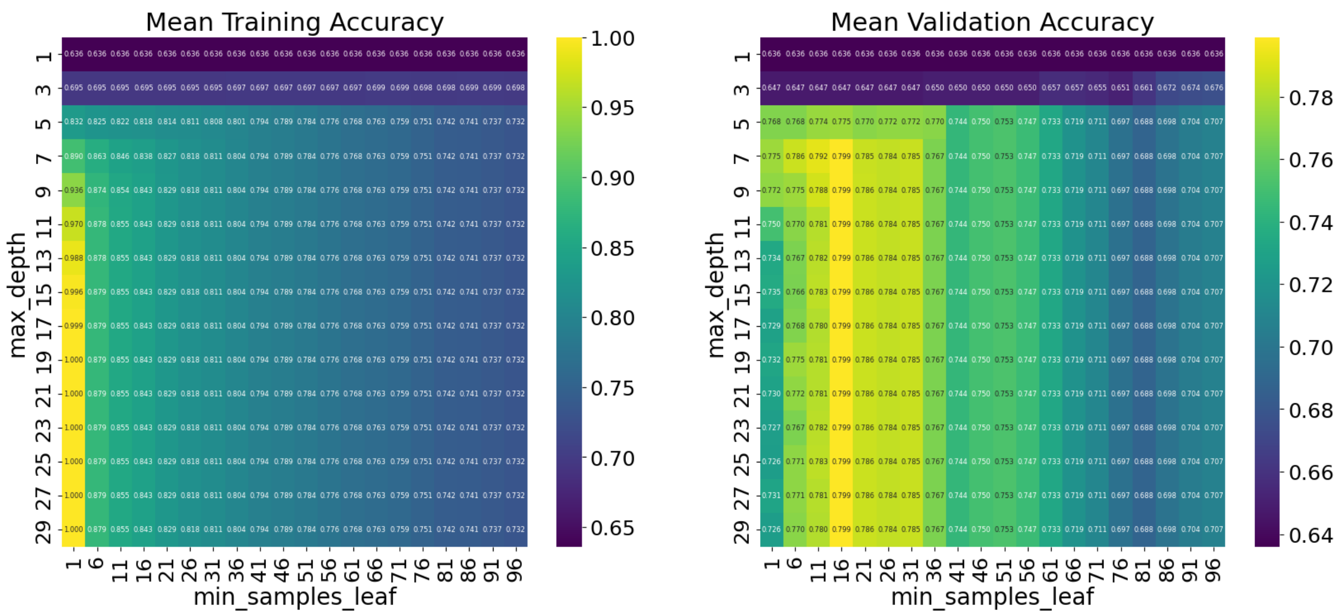
## Part 2: Decision trees

**(Q5)**

```
Training set score:  0.678
```



**(Q6)**



**c)** The optimal combination according to the heatmap is:

min_samples_leaf: 16

max_depth: 7

**d)** A hyperparameter combination that causes underfitting is:

min_samples_leaf: 96

max_depth: 1

**e)** A hyperparameter combination that causes overfitting is:

min_samples_leaf: 1

max_depth: 29

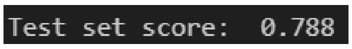
**f)** The hyperparameter in question d leads to underfitting because the tree is restricted to a single split. As a result, it cannot effectively separate non-separable data, such as the data in our case.

The hyperparameter in question e causes overfitting because the model is allowed to grow a tree with leaves containing just one sample. While this results in highly accurate predictions on the training set, the model becomes too tailored to the training data. Consequently, it performs poorly on the validation set, as it fails to generalize well to unseen data.

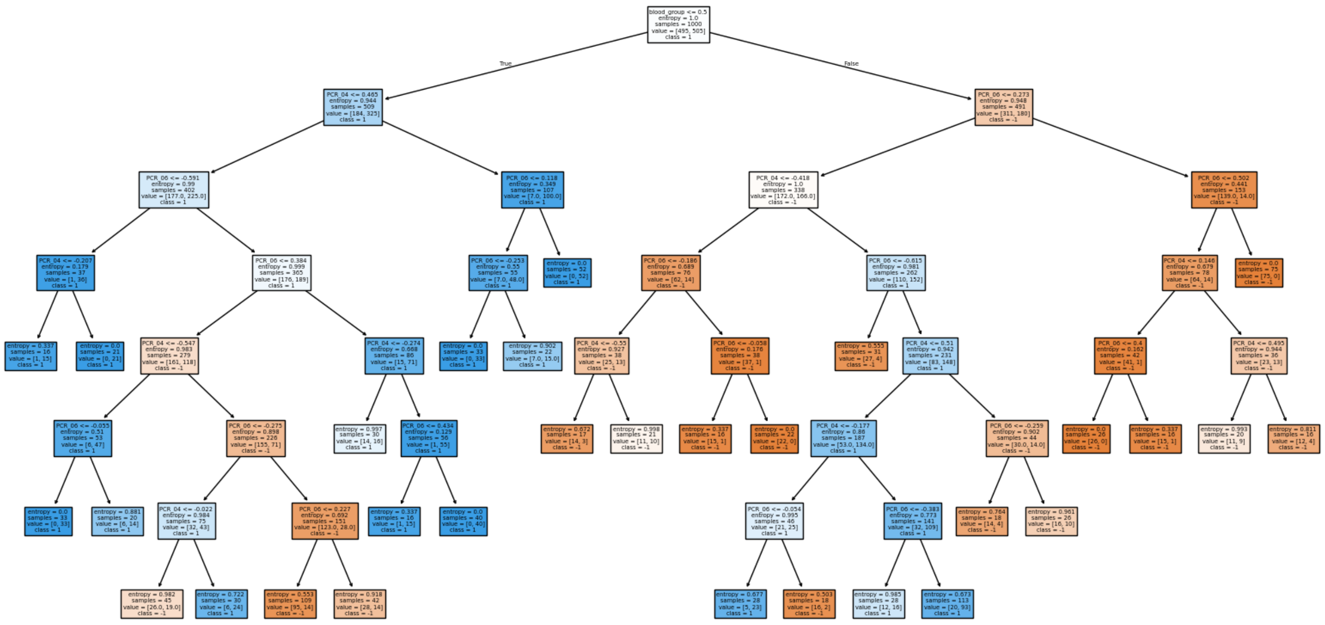
**(Q7)** Since our grid consisted of 15 potential values for max_depth and 20 values for min_samples_leaf, we evaluated a total of 15×20=300 hyperparameter combinations.

If we had introduced a third hyperparameter, the total number of combinations would increase by multiplying the current total by the number of options for the new parameter. Generally, if each hyperparameter has n possible values and there are k hyperparameters, the total number of combinations is n^k.

While having more hyperparameters to tune can improve model performance by exploring a wider range of configurations, it significantly increases computational cost. Therefore, it’s essential to balance the potential performance gains with the computational expense.

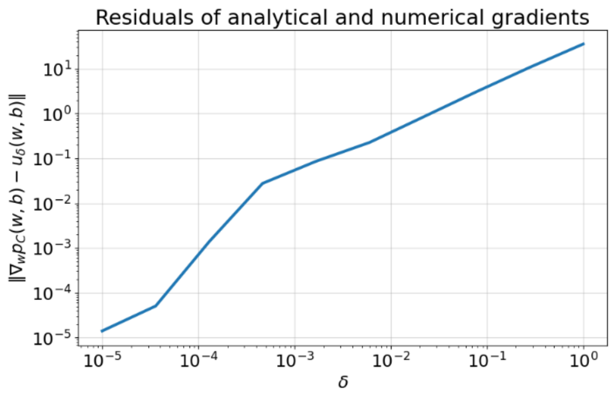
**(Q8)** 

```
Test set score:  0.788
```



## Part 3: Linear SVM and the Polynomial kernel

**(Q9)**



As can be seen in the graph, the smaller the delta, the smaller the difference between the analytical and numerical derivatives. This is because a smaller delta represents a slope at a point that is closer to the computed point, thus providing a value that is closer to the actual slope at that point.

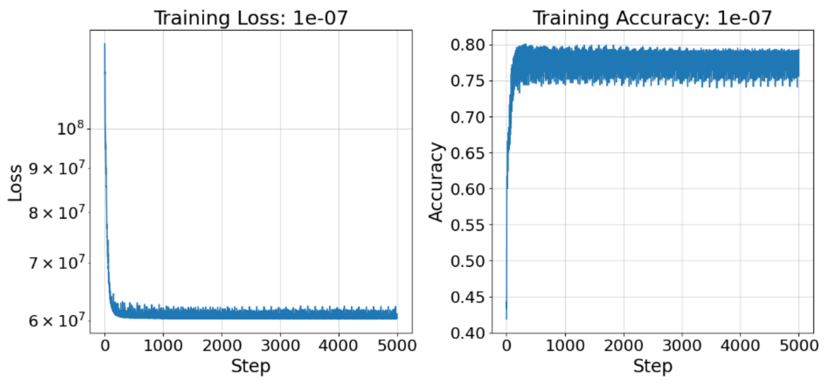
**(Q10)**

The relationship between the error and accuracy graphs is not as expected. As the number of steps increases, we would anticipate a decrease in error and an increase in accuracy. While the error does decrease, the accuracy varies without a clear trend. This difference arises due to the high value of the regularization parameter C, which emphasizes the loss function and leads to the HardSVM problem.

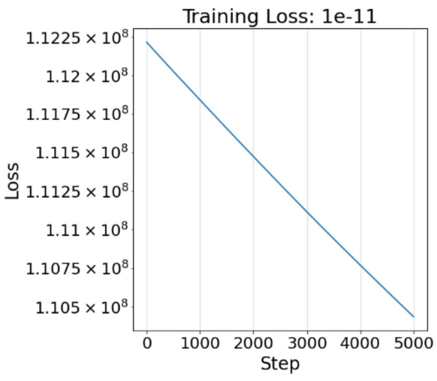
**(Q11)**

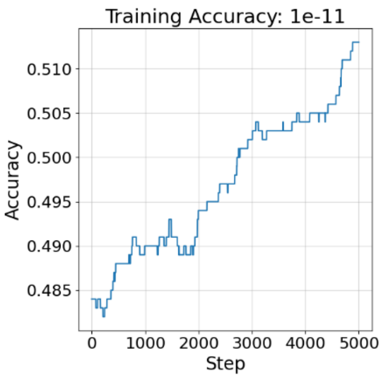
We would choose the learning rate of 1e-07.

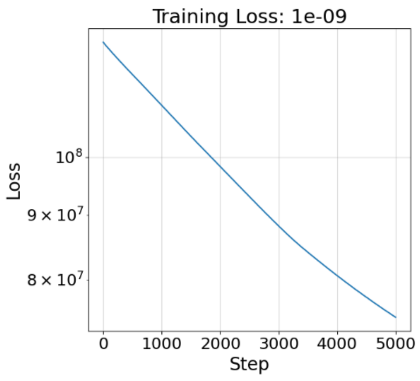
As shown in the graph below, for this rate, the error decreases rapidly with a steep slope, and accordingly, the accuracy measure rises to a high score, ranging between 0.74 and 0.80.
In comparison to other measures, we can see that the upper bounds of the accuracy measures are approximately equal to the rate we chose, but the lower bounds are much lower.
Additionally, these measures are unstable, making it difficult to clearly understand their scores for each step.
Our chosen learning rate of 1e-07 graphs:

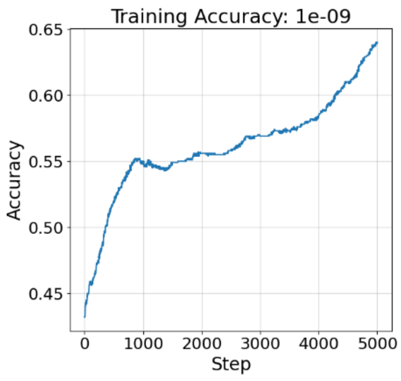


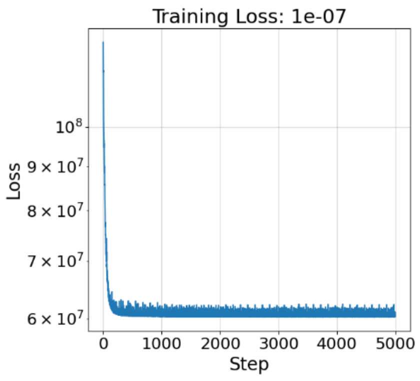
**All the learning rates graphs:**

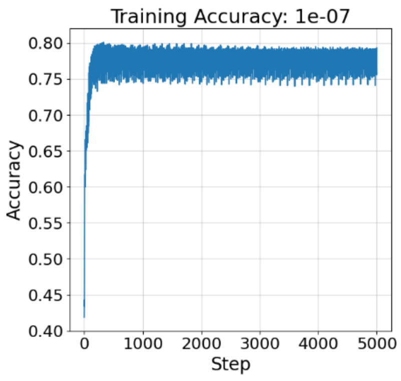


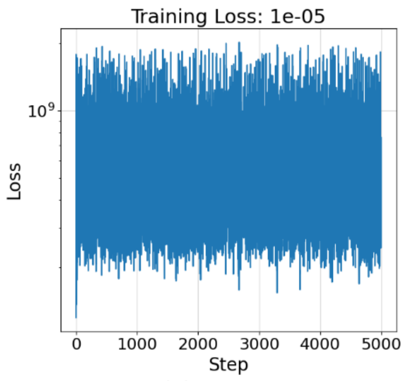


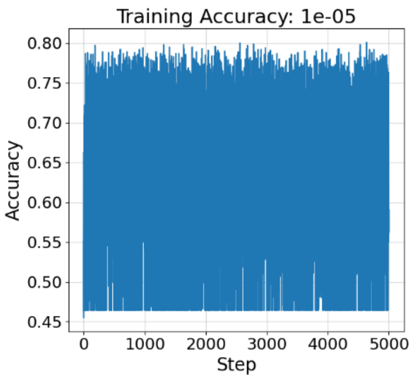


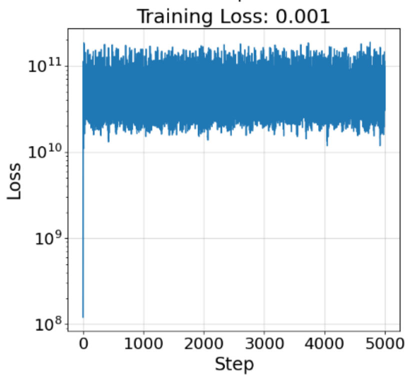


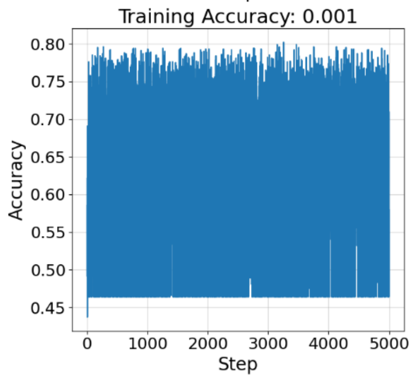


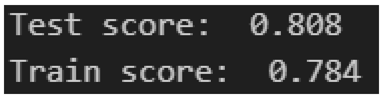






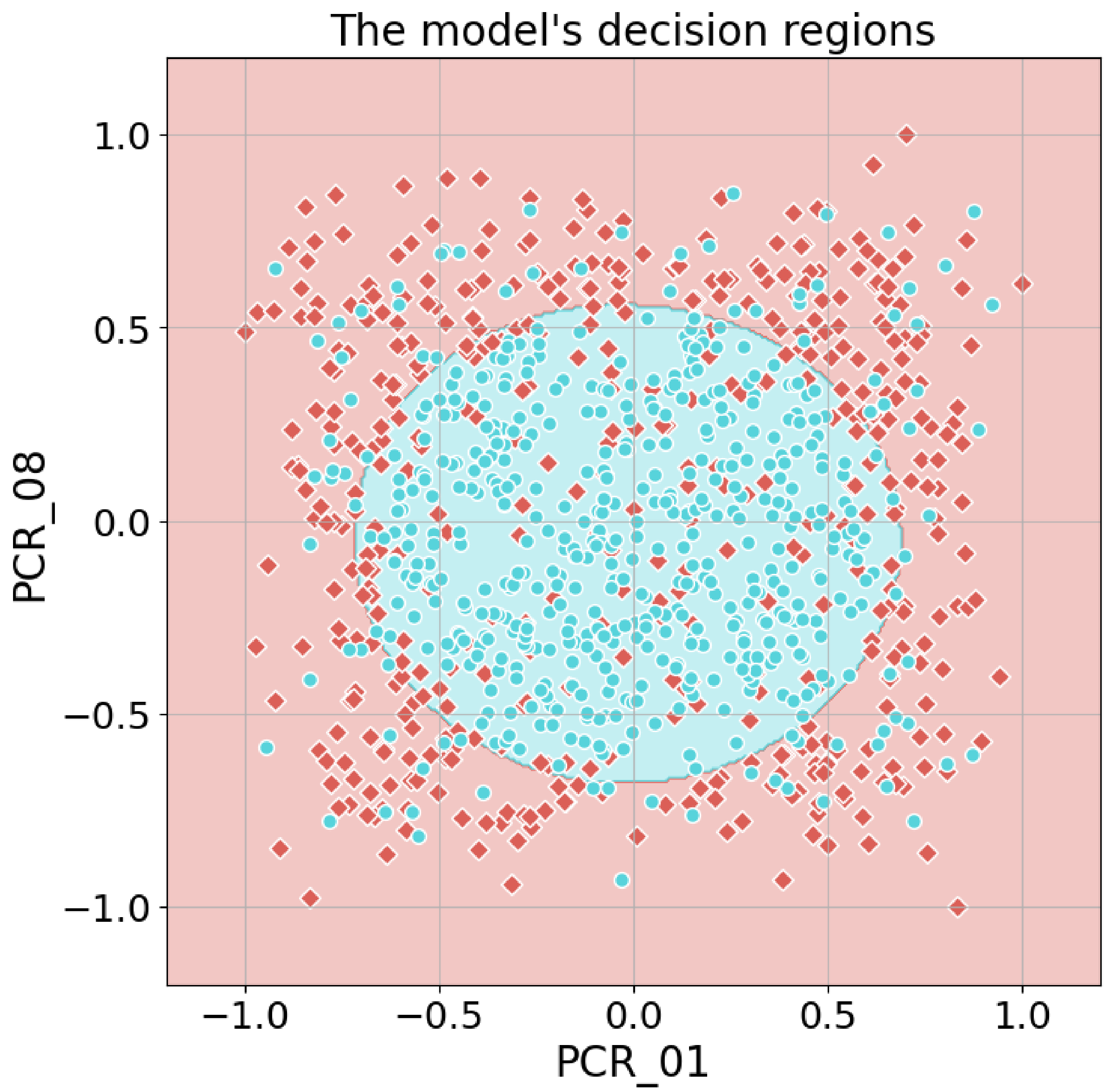




**(Q12)** Our model accuracy is: 

```
Test score:  0.808
Train score:  0.784
```

And the decision model is:



## Part 4: The RBF kernel

**(Q13)**
**Section 1**

a. We know that by definition exists: $\left(\emptyset(S)\right)_i = \begin{cases} 1\ , & i \in S \\ 0\ , & o.w \end{cases}$.
Therefore exists:

$$\langle \emptyset(S_i), \emptyset(S_j) \rangle = \emptyset(S_i)^T \cdot \emptyset(S_j) =$$

$$= \left(\mathbb{I}\{1 \in S_i\} \quad \dots \quad \mathbb{I}\{d-1 \in S_i\} \quad \mathbb{I}\{d \in S_i\}\right) \begin{pmatrix} \mathbb{I}\{1 \in S_j\} \\ \dots \\ \mathbb{I}\{d-1 \in S_j\} \\ \mathbb{I}\{d \in S_j\} \end{pmatrix} =$$

$$\sum_{k \in [d]} \mathbb{I}\{k \in S_i\} \cdot \mathbb{I}\{k \in S_j\} = |S_i \cap S_j|$$

Where as the last equality comes from the fact that the size of the intersection of $S_i\ and\ S_j$ equals the amount of words which belons to both sets.
All in all , we showed there exists a feature mapping $\emptyset(S): \Omega \to \mathbb{R}^d$ as defined earlier, such that $\langle \emptyset(S_i), \emptyset(S_j) \rangle = |S_i \cap S_j|$.
Therefore $K_\cap(S_i, S_j) = |S_i \cap S_j|$ is a well-defined kernel.

b. From (a) we showed that $K_\cap(S_i, S_j) = |S_i \cap S_j|$ is a well-defined kernel.
And we know that kernels built by recursively combining one or more of the rules that were defined in the instructions. Therefore we'll use the rule: $k(x, x') = e^{k_1(x,x')}$.
We know exists:

$$K_{spam}(S_i, S_j) = e^{|S_i \cap S_j|} = e^{K_\cap(S_i, S_j)}$$

And because $K_\cap(S_i, S_j)$ is a well-defined kernel, we proved that $K_{spam}(S_i, S_j)$ is a well-defined kernel.

**Section 2:**

c. We can see that $||\emptyset(S)||_1 = \sum_{k \in [d]} \mathbb{I}\{k \in S\} = |S|$.
Therefore exists:

$$f(S_i) = e^{-||\emptyset(S_i)||_1} = e^{-|S_i|}$$

So we can use the second rule from the instructions: $k(x, x') = f(x)k_1(x, x')f(x')$.
We know exists:

$$e^{-|S_i \cup S_j|} = e^{-|S_i| - |S_j| + |S_i \cap S_j|} = e^{-|S_i|} e^{|S_i \cap S_j|} e^{-|S_j|} = f(S_i) K_{spam}(S_i, S_j) f(S_j)$$

And because $K_{spam}(S_i, S_j)$ is a well-defined kernel, we proved that $e^{-|S_i \cup S_j|}$ is a well-defined kernel.

d. Given $k_1(x,x'), k_2(x,x')$ kernels, we know that exists: $\emptyset_1(x): \Omega \rightarrow \mathbb{R}^{n_1}, \emptyset_2(x): \Omega \rightarrow \mathbb{R}^{n_2}$ s.t :

$$k_1(x,x') = \langle \emptyset_1(x), \emptyset_1(x') \rangle$$
$$k_2(x,x') = \langle \emptyset_2(x), \emptyset_2(x') \rangle$$

We'll show that $k(x,x') = k_1(x,x') \cdot k_2(x,x')$ is a valid kernel.

First we'll define: $\psi : \Omega \rightarrow \mathbb{R}^{n_1+n_2}$ such as:

$$\psi(x) = \begin{pmatrix} \emptyset_1^1(x) \cdot \emptyset_2^1(x) \\ \vdots \\ \emptyset_1^1(x) \cdot \emptyset_2^{n_2}(x) \\ \vdots \\ \emptyset_1^{n_1}(x) \cdot \emptyset_2^1(x) \\ \vdots \\ \emptyset_1^{n_1}(x) \cdot \emptyset_2^{n_2}(x) \end{pmatrix}$$

And we know that:

$$k_1(x,x') \cdot k_2(x,x') = \langle \emptyset_1(x), \emptyset_1(x') \rangle \langle \emptyset_2(x), \emptyset_2(x') \rangle =$$
$$\sum_{i=1}^{n_1} \emptyset_1^i(x) \cdot \emptyset_1^i(x') \cdot \sum_{i=1}^{n_2} \emptyset_2^i(x) \cdot \emptyset_2^i(x') = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \emptyset_1^i(x) \cdot \emptyset_1^i(x') \cdot \emptyset_2^j(x) \cdot \emptyset_2^j(x')$$

And now we'll see that:

$$\langle \psi(x), \psi(x') \rangle =$$

$$= \begin{pmatrix} \emptyset_1^1(x) \cdot \emptyset_2^1(x) & \dots & \emptyset_1^1(x) \cdot \emptyset_2^{n_2}(x) & \dots & \emptyset_1^{n_1}(x) \cdot \emptyset_2^1(x) & \dots & \emptyset_1^{n_1}(x) \cdot \emptyset_2^{n_2}(x) \end{pmatrix} \begin{pmatrix} \emptyset_1^1(x) \cdot \emptyset_2^1(x) \\ \vdots \\ \emptyset_1^1(x) \cdot \emptyset_2^{n_2}(x) \\ \vdots \\ \emptyset_1^{n_1}(x) \cdot \emptyset_2^1(x) \\ \vdots \\ \emptyset_1^{n_1}(x) \cdot \emptyset_2^{n_2}(x) \end{pmatrix}$$

$$= \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \emptyset_1^i(x) \cdot \emptyset_1^i(x') \cdot \emptyset_2^j(x) \cdot \emptyset_2^j(x') = k_1(x,x') \cdot k_2(x,x') = k(x,x')$$

All in all we showed that $k(x,x') = \langle \psi(x), \psi(x') \rangle$ which means its a well-defined kernel.

e. According to (b) we know that $K_{spam}(S_i, S_j) = e^{|S_i \cap S_j|}$ is a well-defined kernel.
According to (c) we know that $e^{-|S_i \cup S_j|}$ is a well-defined kernel.
According to (d) we know that multiplication of 2 well-defined kernels is also well-defined kernel.
Therefore exists:

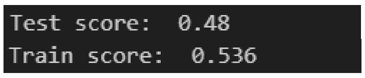
$$K_{spam\ pro\ max}(S_i, S_j) = e^{|S_i \cap S_j| - |S_i \cup S_j|} = e^{|S_i \cap S_j|} e^{-|S_i \cup S_j|} = K_{spam}(S_i, S_j) \cdot e^{-|S_i \cup S_j|}$$

Which proves that $K_{spam\ pro\ max}(S_i, S_j)$ is a well-defined kernel.

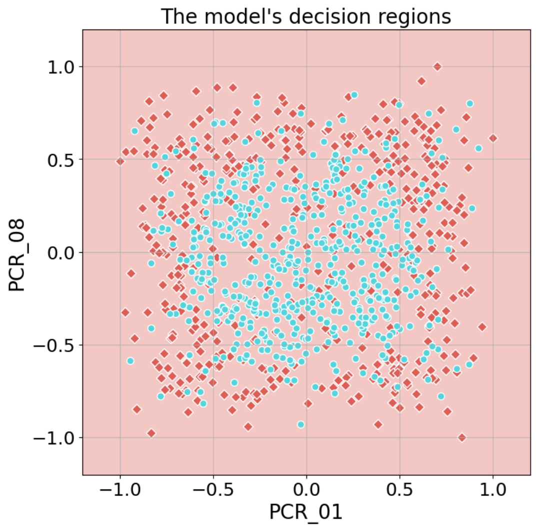
f. $K_{spam\ pro\ max}(S_i, S_j)$ is better because it balances the similarity by looking at both the overlap and the total size of the sets, not just the intersection. This makes it less biased toward smaller sets and gives a clearer sense of how much two emails really have in common. Overall, it's a more fair and reliable way to measure similarity for classification.

g. Yes, we suggest this kernel: $K_{better}(S_i, S_j) = e^{\frac{|S_i \cap S_j|}{\sqrt{|S_i| \cdot |S_j|}}}$.
$K_{better}(S_i, S_j)$ is better than $K_{spam\ pro\ max}(S_i, S_j)$ because it balances the overlap by considering the sizes of both sets more evenly. This helps avoid cases where one set is much bigger and skews the similarity score, making it a fairer way to compare emails.

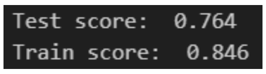
**(Q14)**



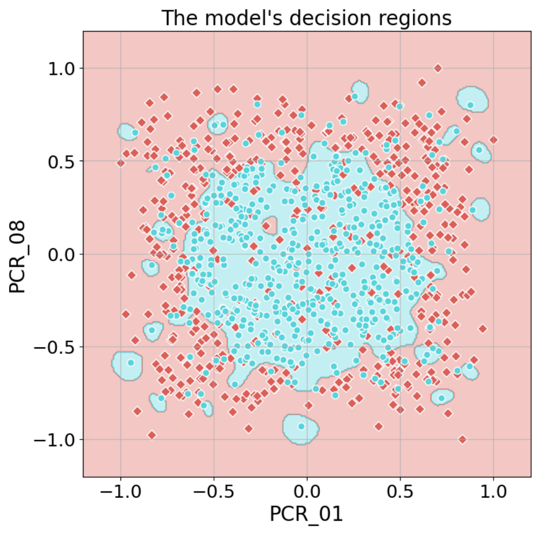
```
Test score:  0.48
Train score:  0.536
```



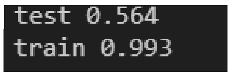
Based on the graph included in the report, it’s clear that the model is underfitting. This is because of the low results, especially on the training set, suggesting that the model is too simple and unable to capture the underlying patterns in the data.

**(Q15)** 

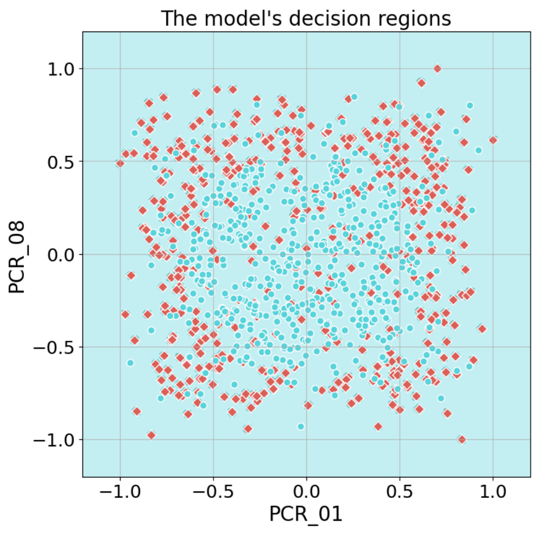
```
Test score:  0.764
Train score:  0.846
```



It can be observed that this graph is similar to the one produced by the KNN model with an optimal k value of 13 from question 3, though there are few differences.
Firstly, the test score of the KNN model (0.776) is higher than that of the SVM model with RBF (0.764).
However, the score of the SVM model with RBF is very close to that of the optimal KNN, as the chosen gamma is likely optimal or close to it.
This slight difference in results may arise from the fundamental differences in how the two models classify data, with KNN relying on local data points and SVM with RBF kernel modelling a global decision boundary.
As we saw in question 14, with a gamma that is too small, we will experience underfitting, whereas in question 16, with a gamma that is too large, we will encounter overfitting.

**(Q16)** 

```
test 0.564
train 0.993
```



Based on the graph included in the report, it's clear that the model is overfitting. The low performance on the test set, in contrast to the very high accuracy on the training set, suggests that the model has learned to fit the training data too closely and is not generalizing well to new data.