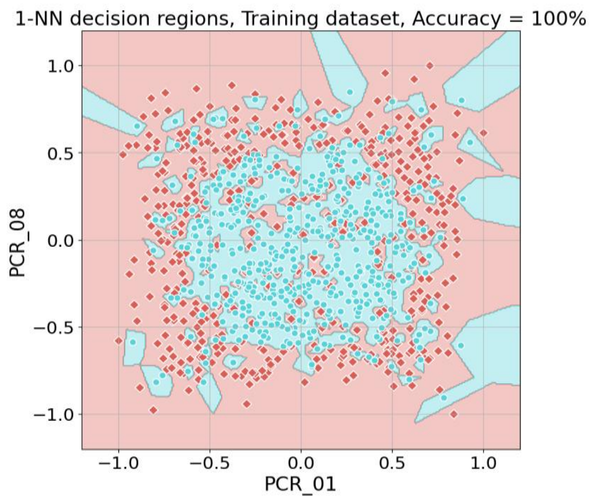
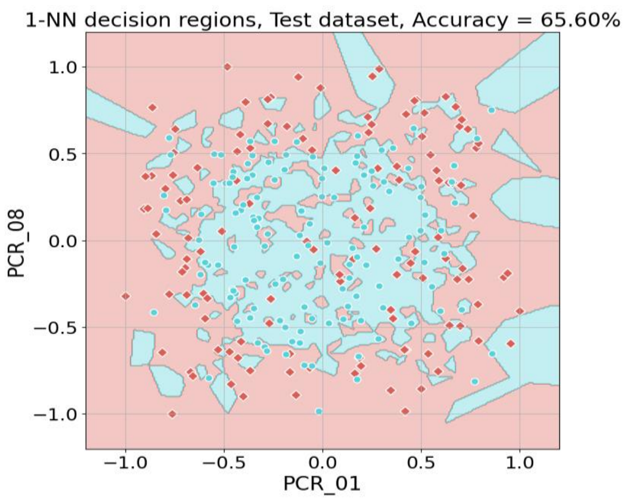
<u>**236766 WET HW2- REPORT-208081588, 209462415**</u>

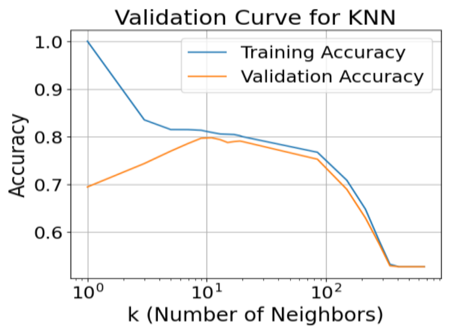
Q1:
Train a k-NN model using k = 1 on your training set and use the visualize_clf method to visualize the resulted decision regions (with appropriate title including accuracy and labels) for train and test sets.



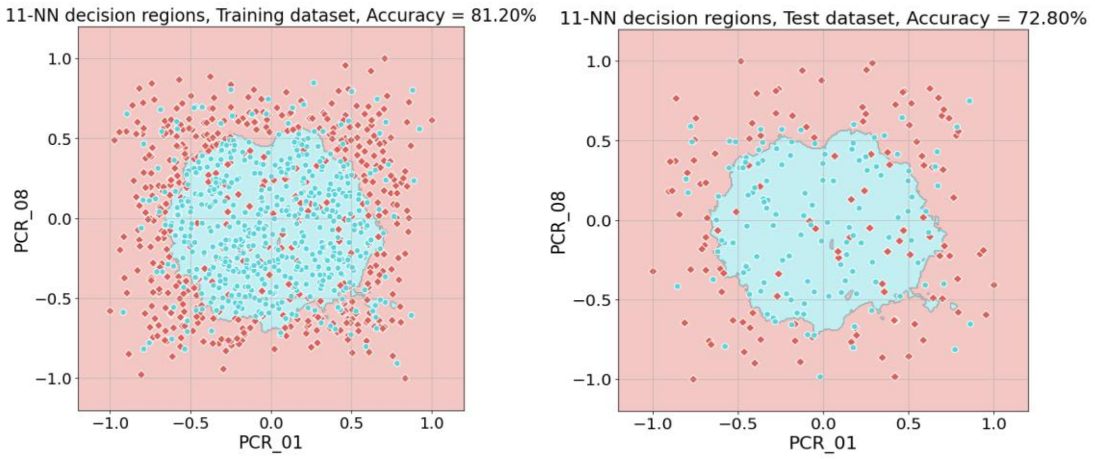


Q2:
Using the outputs of cross_validate, plot a validation curve, i.e., the (mean) training and validation accuracies (y-axis) as functions of the k values (x-axis). Make the x-axis logarithmic (using plt.semilogx) and attach the plot (with the 2 curves) to your report.
Answer: Which k is the best? What are its average training and validation accuracies (estimated by cross-val.)? Which k values cause overfitting and underfitting and why?

A:
We believe that k=11 is the best, because it achieves the highest validation accuracy while not overly under or over fitting (as we can see in the graph, the training accuracy is relatively high and the difference between the training and validation accuracy is relatively low).
The average training accuracy for k=11 is: 80.91%.
The average validation accuracy for k=11 is: 79.80%.
As seen in the graph, low k values (k<= 5) cause high overfitting since the difference between the training and validation accuracies is high (>= 10%). Also, high k values cause underfitting because we see the training accuracy dropping, even reaching scores below 60%.



Q3:
Use the optimal k value you found and retrain a k-NN model on all the training samples.
In your report: plot the decision regions of this final model (using visualize_clf)
and write its test accuracy (computed on the separate test split) of this model.



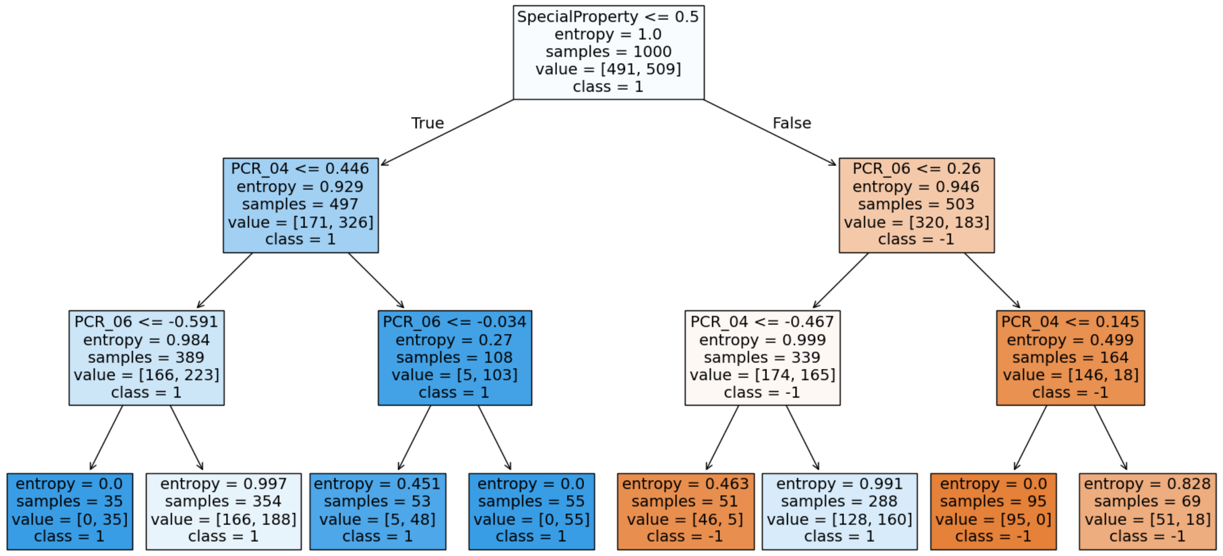
The test accuracy for k=11 is: 72.80%.

Q4:
Compare the boundaries of the two models you have trained in (Q1) and (Q3).
Discuss the results and the exhibited behaviors (2-3 sentences).

As can be seen in the plots in Q1 and Q3, the 1-NN model greatly overfits the training data and is influenced heavily by single isolated points when picking decision regions.
On the other hand, the 11-NN model considers more neighbors when picking decision regions which results in more uniformal regions and better generalization for new data. As a result, the 11-NN model achieves greater accuracy on the test dataset.

Q5:
Train a model with ID3 and max_depth=3 (not including the root level; use the entire training set, i.e., all the features after preprocessing from all the training samples).
What is the training accuracy? Visualize the trained tree using plot_tree (provide feature and class names; use filled=True) and attach the plot to your report. The plot should be readable!

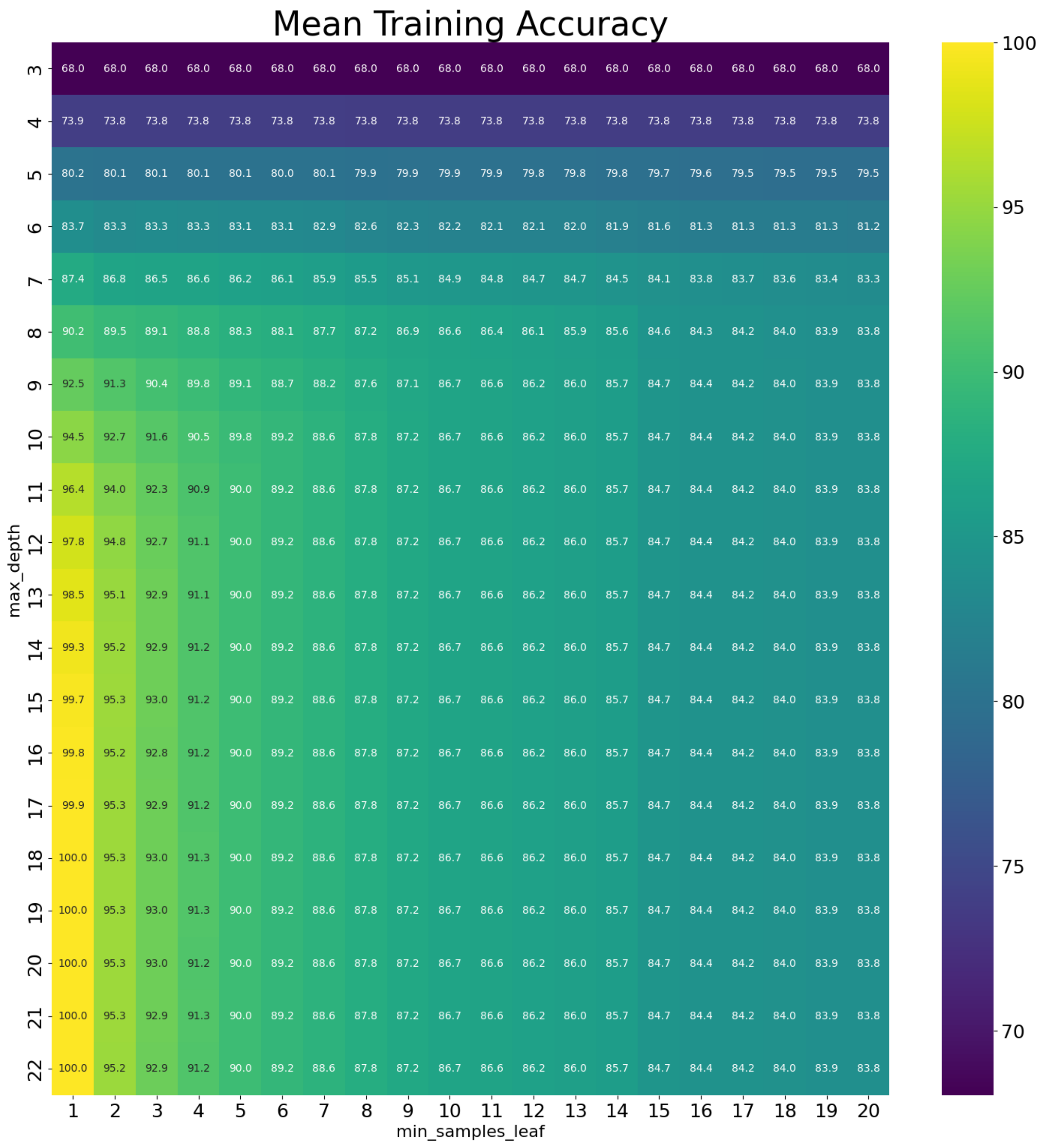
A:
The training accuracy of ID3 model with max_depth=3 is: 67.80%.
Plot below.



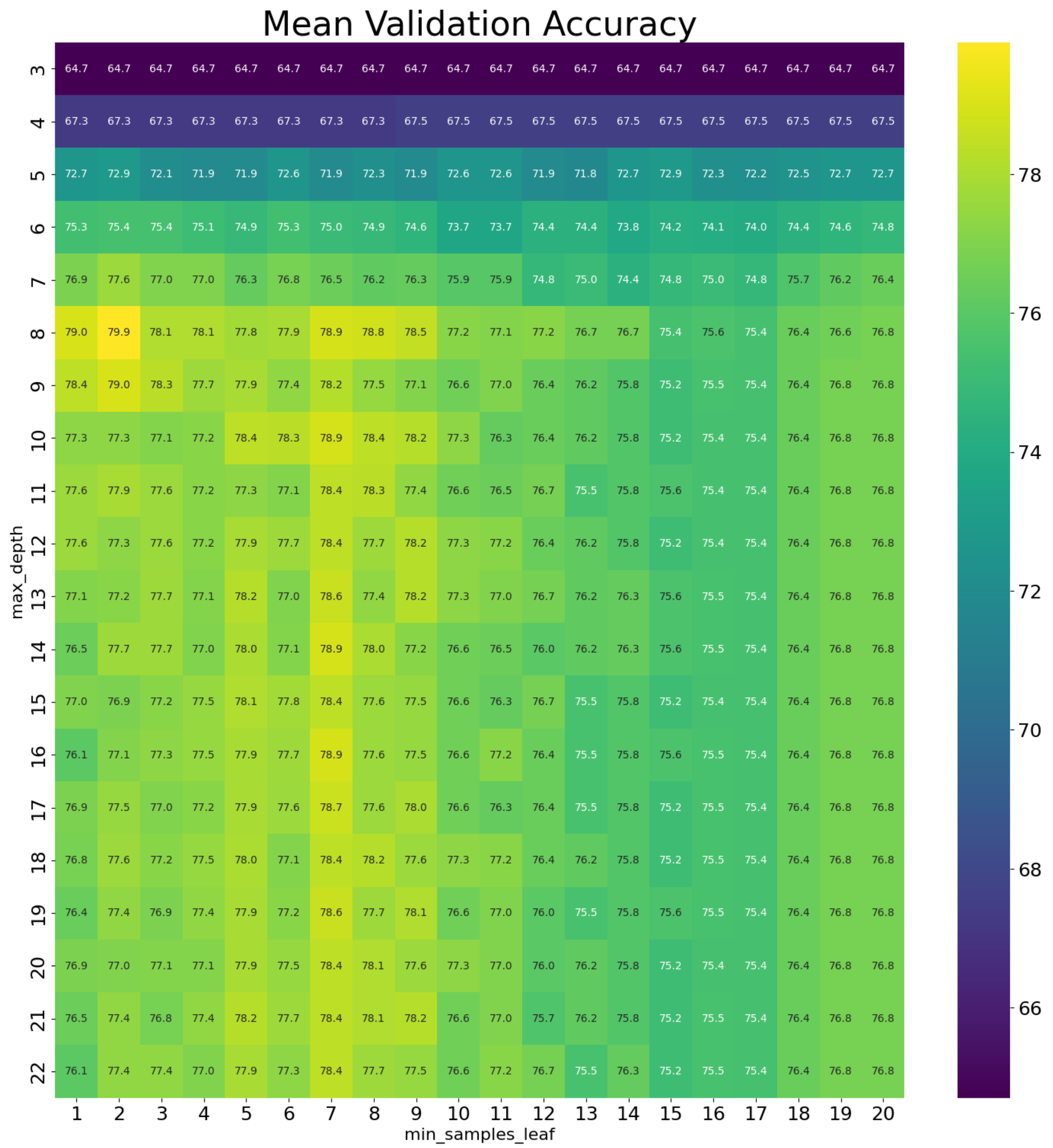
Q6:
Using 5-fold cross-validation, tune the two hyperparameters by performing a grid search (see GridSearchCV). Find the combination yielding the best validation error for predicting the risk class.
c. Add the 2 plots to your report and specify which hyperparameter combination is optimal.
d. Write a hyperparameter combination that causes underfitting.
e. Write a hyperparameter combination that causes overfitting.
f. Add a short discussion regarding why each specific hyperparameter-combination from sub-questions 'd' and 'e' resulted in under/over-fitting.

A:
The optimal hyperparameter combination, in respect to validation accuracy, is max_depth=8 and min_samples_leaf=2. This is shown in the plots below.
A combination that causes underfitting is max_depth=3 and min_samples_leaf=20.
A combination that causes overfitting is max_depth=22 and min_samples_leaf=1.

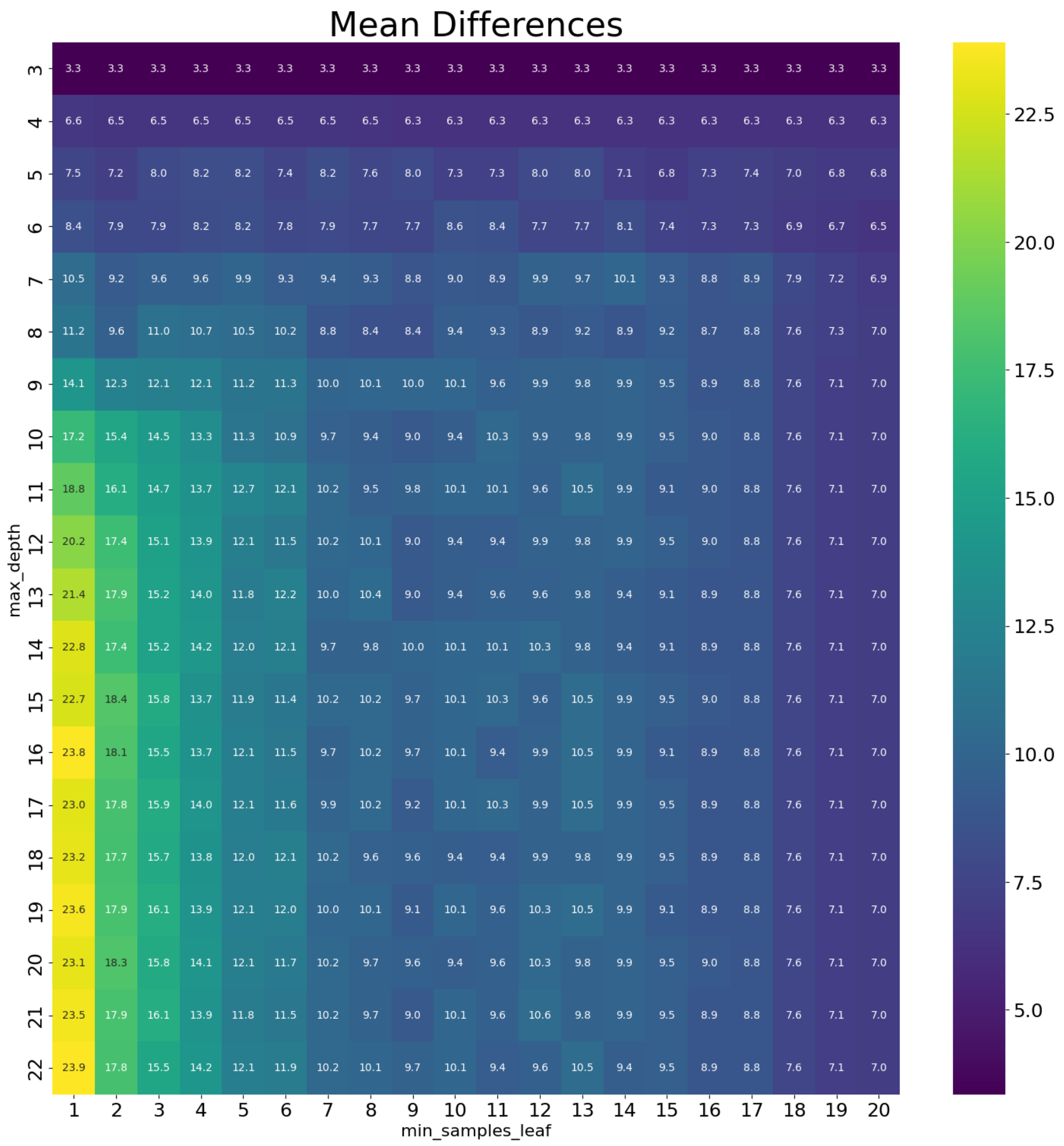
The above combinations cause under/over-fitting because as max_depth grows we can get very specific decision areas up to a single sample while low max_depth is not enough for meaningful sorting of the dataset. Similarly if min_samples_leaf is small, then we allow the tree to reach decisions based on a specific sample, and if it is large we generalize too much - losing the ability to finely sort the samples.
Therefore, a combination of low max_depth and with large min_samples_leaf results in a rough partition into decision areas that has low accuracy even on the training data (underfitting), while the opposite combination results in fine decision areas that match the training samples with great accuracy while not achieving higher validation accuracy (overfitting).



max_depth \ min_samples_leaf	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0	68.0
4	73.9	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8	73.8
5	80.2	80.1	80.1	80.1	80.1	80.0	80.1	79.9	79.9	79.9	79.9	79.8	79.8	79.8	79.7	79.6	79.5	79.5	79.5	79.5
6	83.7	83.3	83.3	83.3	83.1	83.1	82.9	82.6	82.3	82.2	82.1	82.1	82.0	81.9	81.6	81.3	81.3	81.3	81.3	81.2
7	87.4	86.8	86.5	86.6	86.2	86.1	85.9	85.5	85.1	84.9	84.8	84.7	84.7	84.5	84.1	83.8	83.7	83.6	83.4	83.3
8	90.2	89.5	89.1	88.8	88.3	88.1	87.7	87.2	86.9	86.6	86.4	86.1	85.9	85.6	84.6	84.3	84.2	84.0	83.9	83.8
9	92.5	91.3	90.4	89.8	89.1	88.7	88.2	87.6	87.1	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
10	94.5	92.7	91.6	90.5	89.8	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
11	96.4	94.0	92.3	90.9	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
12	97.8	94.8	92.7	91.1	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
13	98.5	95.1	92.9	91.1	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
14	99.3	95.2	92.9	91.2	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
15	99.7	95.3	93.0	91.2	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
16	99.8	95.2	92.8	91.2	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
17	99.9	95.3	92.9	91.2	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
18	100.0	95.3	93.0	91.3	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
19	100.0	95.3	93.0	91.3	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
20	100.0	95.3	93.0	91.2	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
21	100.0	95.3	92.9	91.3	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8
22	100.0	95.2	92.9	91.2	90.0	89.2	88.6	87.8	87.2	86.7	86.6	86.2	86.0	85.7	84.7	84.4	84.2	84.0	83.9	83.8



max_depth	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7	64.7
4	67.3	67.3	67.3	67.3	67.3	67.3	67.3	67.3	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5	67.5
5	72.7	72.9	72.1	71.9	71.9	72.6	71.9	72.3	71.9	72.6	72.6	71.9	71.8	72.7	72.9	72.3	72.2	72.5	72.7	72.7
6	75.3	75.4	75.4	75.1	74.9	75.3	75.0	74.9	74.6	73.7	73.7	74.4	74.4	73.8	74.2	74.1	74.0	74.4	74.6	74.8
7	76.9	77.6	77.0	77.0	76.3	76.8	76.5	76.2	76.3	75.9	75.9	74.8	75.0	74.4	74.8	75.0	74.8	75.7	76.2	76.4
8	79.0	79.9	78.1	78.1	77.8	77.9	78.9	78.8	78.5	77.2	77.1	77.2	76.7	76.7	75.4	75.6	75.4	76.4	76.6	76.8
9	78.4	79.0	78.3	77.7	77.9	77.4	78.2	77.5	77.1	76.6	77.0	76.4	76.2	75.8	75.2	75.5	75.4	76.4	76.8	76.8
10	77.3	77.3	77.1	77.2	78.4	78.3	78.9	78.4	78.2	77.3	76.3	76.4	76.2	75.8	75.2	75.4	75.4	76.4	76.8	76.8
11	77.6	77.9	77.6	77.2	77.3	77.1	78.4	78.3	77.4	76.6	76.5	76.7	75.5	75.8	75.6	75.4	75.4	76.4	76.8	76.8
12	77.6	77.3	77.6	77.2	77.9	77.7	78.4	77.7	78.2	77.3	77.2	76.4	76.2	75.8	75.2	75.4	75.4	76.4	76.8	76.8
13	77.1	77.2	77.7	77.1	78.2	77.0	78.6	77.4	78.2	77.3	77.0	76.7	76.2	76.3	75.6	75.5	75.4	76.4	76.8	76.8
14	76.5	77.7	77.7	77.0	78.0	77.1	78.9	78.0	77.2	76.6	76.5	76.0	76.2	76.3	75.6	75.5	75.4	76.4	76.8	76.8
15	77.0	76.9	77.2	77.5	78.1	77.8	78.4	77.6	77.5	76.6	76.3	76.7	75.5	75.8	75.2	75.4	75.4	76.4	76.8	76.8
16	76.1	77.1	77.3	77.5	77.9	77.7	78.9	77.6	77.5	76.6	77.2	76.4	75.5	75.8	75.6	75.5	75.4	76.4	76.8	76.8
17	76.9	77.5	77.0	77.2	77.9	77.6	78.7	77.6	78.0	76.6	76.3	76.4	75.5	75.8	75.2	75.5	75.4	76.4	76.8	76.8
18	76.8	77.6	77.2	77.5	78.0	77.1	78.4	78.2	77.6	77.3	77.2	76.4	76.2	75.8	75.2	75.5	75.4	76.4	76.8	76.8
19	76.4	77.4	76.9	77.4	77.9	77.2	78.6	77.7	78.1	76.6	77.0	76.0	75.5	75.8	75.6	75.5	75.4	76.4	76.8	76.8
20	76.9	77.0	77.1	77.1	77.9	77.5	78.4	78.1	77.6	77.3	77.0	76.0	76.2	75.8	75.2	75.4	75.4	76.4	76.8	76.8
21	76.5	77.4	76.8	77.4	78.2	77.7	78.4	78.1	78.2	76.6	77.0	75.7	76.2	75.8	75.2	75.5	75.4	76.4	76.8	76.8
22	76.1	77.4	77.4	77.0	77.9	77.3	78.4	77.7	77.5	76.6	77.2	76.7	75.5	76.3	75.2	75.5	75.4	76.4	76.8	76.8



max_depth \ min_samples_leaf	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3
4	6.6	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3
5	7.5	7.2	8.0	8.2	8.2	7.4	8.2	7.6	8.0	7.3	7.3	8.0	8.0	7.1	6.8	7.3	7.4	7.0	6.8	6.8
6	8.4	7.9	7.9	8.2	8.2	7.8	7.9	7.7	7.7	8.6	8.4	7.7	7.7	8.1	7.4	7.3	7.3	6.9	6.7	6.5
7	10.5	9.2	9.6	9.6	9.9	9.3	9.4	9.3	8.8	9.0	8.9	9.9	9.7	10.1	9.3	8.8	8.9	7.9	7.2	6.9
8	11.2	9.6	11.0	10.7	10.5	10.2	8.8	8.4	8.4	9.4	9.3	8.9	9.2	8.9	9.2	8.7	8.8	7.6	7.3	7.0
9	14.1	12.3	12.1	12.1	11.2	11.3	10.0	10.1	10.0	10.1	9.6	9.9	9.8	9.9	9.5	8.9	8.8	7.6	7.1	7.0
10	17.2	15.4	14.5	13.3	11.3	10.9	9.7	9.4	9.0	9.4	10.3	9.9	9.8	9.9	9.5	9.0	8.8	7.6	7.1	7.0
11	18.8	16.1	14.7	13.7	12.7	12.1	10.2	9.5	9.8	10.1	10.1	9.6	10.5	9.9	9.1	9.0	8.8	7.6	7.1	7.0
12	20.2	17.4	15.1	13.9	12.1	11.5	10.2	10.1	9.0	9.4	9.4	9.9	9.8	9.9	9.5	9.0	8.8	7.6	7.1	7.0
13	21.4	17.9	15.2	14.0	11.8	12.2	10.0	10.4	9.0	9.4	9.6	9.6	9.8	9.4	9.1	8.9	8.8	7.6	7.1	7.0
14	22.8	17.4	15.2	14.2	12.0	12.1	9.7	9.8	10.0	10.1	10.1	10.3	9.8	9.4	9.1	8.9	8.8	7.6	7.1	7.0
15	22.7	18.4	15.8	13.7	11.9	11.4	10.2	10.2	9.7	10.1	10.3	9.6	10.5	9.9	9.5	9.0	8.8	7.6	7.1	7.0
16	23.8	18.1	15.5	13.7	12.1	11.5	9.7	10.2	9.7	10.1	9.4	9.9	10.5	9.9	9.1	8.9	8.8	7.6	7.1	7.0
17	23.0	17.8	15.9	14.0	12.1	11.6	9.9	10.2	9.2	10.1	10.3	9.9	10.5	9.9	9.5	8.9	8.8	7.6	7.1	7.0
18	23.2	17.7	15.7	13.8	12.0	12.1	10.2	9.6	9.6	9.4	9.4	9.9	9.8	9.9	9.5	8.9	8.8	7.6	7.1	7.0
19	23.6	17.9	16.1	13.9	12.1	12.0	10.0	10.1	9.1	10.1	9.6	10.3	10.5	9.9	9.1	8.9	8.8	7.6	7.1	7.0
20	23.1	18.3	15.8	14.1	12.1	11.7	10.2	9.7	9.6	9.4	9.6	10.3	9.8	9.9	9.5	9.0	8.8	7.6	7.1	7.0
21	23.5	17.9	16.1	13.9	11.8	11.5	10.2	9.7	9.0	10.1	9.6	10.6	9.8	9.9	9.5	8.9	8.8	7.6	7.1	7.0
22	23.9	17.8	15.5	14.2	12.1	11.9	10.2	10.1	9.7	10.1	9.4	9.6	10.5	9.4	9.5	8.9	8.8	7.6	7.1	7.0

Q7:
Write the number of hyperparameter combinations that were evaluated in your grid search. Had you wished to tune a third hyperparameter, how would that affect the number of combinations? Shortly discuss how searching over additional hyperparameters affects the total number of possible combinations.

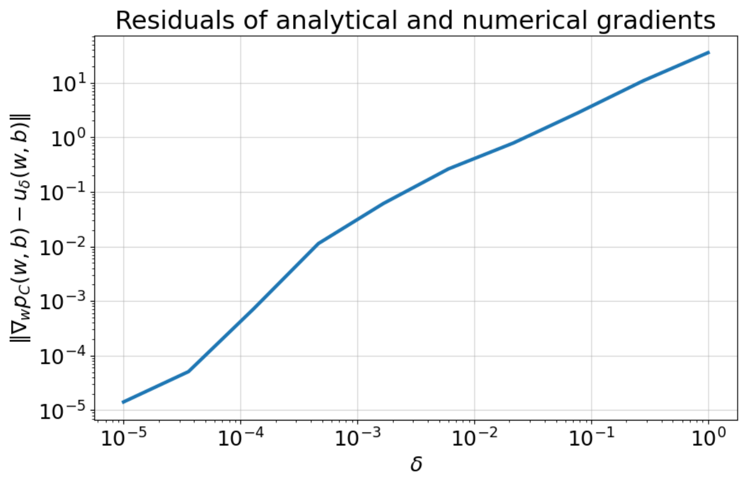
A:
We evaluated 400 combinations in our grid search.
If we wished to tune a third hyperparameters, the number of combinations would be multiplied by the number of possible values for the third hyperparameter.
Each additional hyperparameter with additional possible values would require searching over the previous combinations combined with each possible value of the new hyperparameter. Namely, if we have hyperparameters $\lambda_1, \ldots, \lambda_k$ each with a number of possible values $n_1, \ldots, n_k$, we would have $\prod_{i=1}^{k} n_i$ combinations to search over - this can increase significantly with each additional hyperparameter.

Q8:
Use the optimal hyperparameter combination you found and retrain a decision tree on all the training samples. In your report write the test accuracy of this model.

A:
The test accuracy of optimal decision tree is: 75.60%.

Q9:
Using PCR_01, PCR_08, generate a plot that compares the numerical gradients to the analytic gradients. Do this by running the following command:
Attach the plot to your report. Briefly discuss and justify the demonstrated behavior.

A:



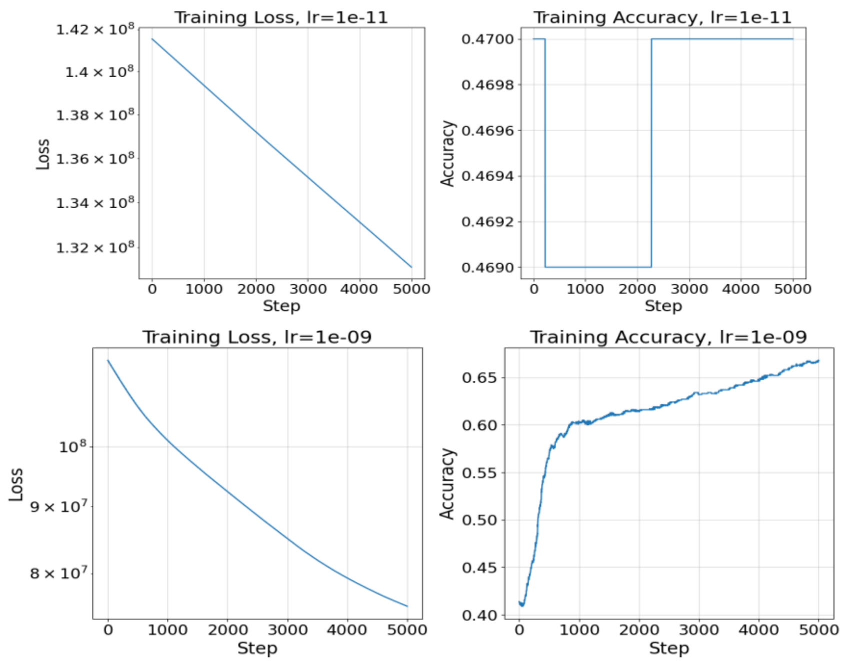
As we can see in the plot, the analytic gradients and the numerical gradients converge to the same values (the difference approaches zero as delta decreases), this is expected since the analytic gradient in each coordinate is the limit of the numerical gradient as delta approaches 0.

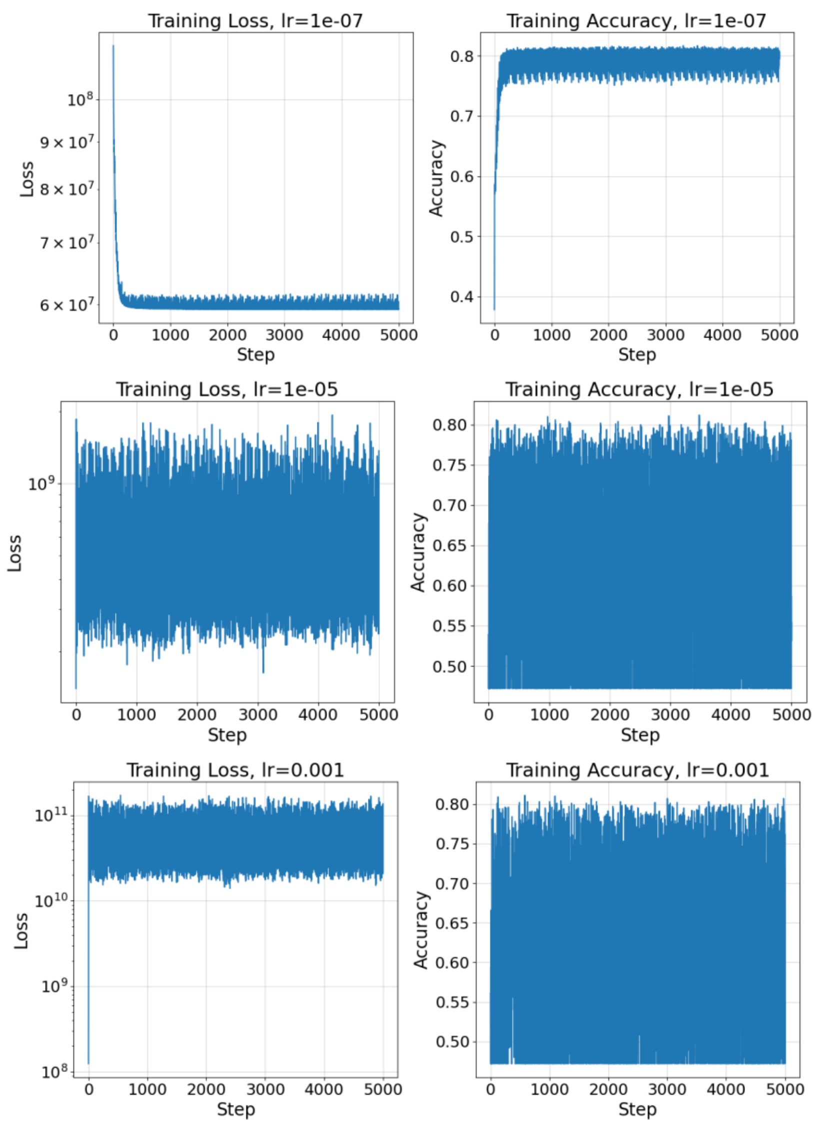
Q10:
We trained a SoftSVM with C = 1e11, lr = 2e-14 and obtained the following curves:
Discuss the behavior of the loss and accuracy during the training of the model and any interactions you observe between the loss and the accuracy.
Does this interaction match your expectations?
If it does – explain why. If it does not – try to settle it.

A:
The training loss decreases as expected from GD (because the objective is convex and we use small lr, it should converge to a minima). On the other hand, the accuracy fluctuates without converging. This does not meet our expectations.
We believe this happens because C is very large, which causes the model to try to avoid misclassification of points, especially if they are far from the decision boundary. This goal does not correlate with achieving higher accuracy, therefore compromising the accuracy of the model in favor of low penalties.

Q11:
Add the plots to the report and explain which learning rate you would choose and why.

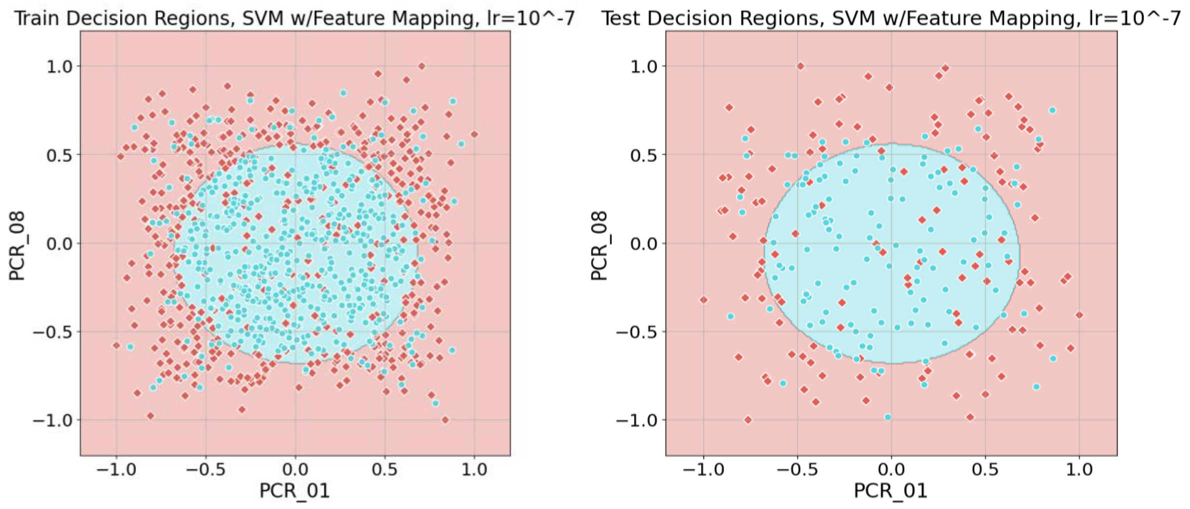
A:





We would choose learning rate = 10^-7 because it reaches the best accuracy.
We can see that when the learning rate is bigger than 10^-7, the loss and accuracy both fluctuate significantly even when approaching 5000 steps, meaning we have a great chance of getting a model with low accuracy even after 5000 steps.
On the other hand, when the learning rate is smaller than 10^-7, we need more than 5000 steps to reach the same accuracy and loss as in 10^-7.
So although with learning rate = 10^-7, the accuracy and loss don't converge and even fluctuates, the fluctuation is insignificant and after 1000 steps we can see the accuracy doesn't drop below 0.75, which is higher than the best accuracy reached in 5000 steps with learning rate = 10^-9.

Q12:
In your report: a. Plot the trained model's decision regions. b. Write the respective training and test accuracies of the model.



The model's train accuracy is: 80.0%.
The model's test accuracy is: 73.6%.

Q13:
a. Using the vector dot product show that $< \phi(S_i), \phi(S_j) >= |S_i \cap S_j|$ and explain why it's a well-defined kernel.
b. Now prove that K_{spam} is a well-defined kernel.
c. First, let's refresh our memory on Set theory. Given two finite groups S_i, S_j the size of the union is $|S_i \cup S_j| = |S_i| + |S_j| - |S_i \cap S_j|$.
Define the function $f(S_i) = e^{-||\phi(S_i)||_1}$ and prove that $e^{-|S_i \cup S_j|}$ is a kernel.
d. Prove that if $k_1(x, x')$ and $k_2(x, x')$ are kernels then $k(x, x') = k_1(x, x') \cdot k_2(x, x')$ is a kernel.
e. Prove that $K_{spam\ pro\ max}$ is a well-defined kernel.
f. Give a general explanation (no more than 3 sentences) of why the suggested kernel is a better choice.
g. Can you suggest an even better kernel? (It isn't supposed to be well-defined).

A:

a. The vector dot product of $\phi(S_i)$ and $\phi(S_j)$ results in a sum where the k'th term is 1 if the k'th coordinate of <u>both</u> $\phi(S_i)$ and $\phi(S_j)$ is 1, and 0 otherwise.
   Additionally, a word represented by $k \in [d]$ is in $S_i \cap S_j$ if and only if it is in S_i <u>and</u> in S_j, which by definition means $(\phi(S_i))_k = (\phi(S_j))_k = 1$.
   Therefore the k'th term in the sum given by the dot product is 1 if and only if the word represented by k is in $S_i \cap S_j$ which gives the required equation.
   By definition, the kernel $K(S_i, S_j) = |S_i \cap S_j| =< \phi(S_i), \phi(S_j) >= \phi(S_i)^T \phi(S_j)$ is well-defined.

b. We know that if $K_1(x,x')$ is a well-defined kernel then $e^{K_1(x,x')}$ is a well-defined kernel, which gives that $K_{spam}(S_i,S_j) = e^{|S_i \cap S_j|}$ is a well-defined kernel.

c. First, notice that $\|\phi(S)\|_1 = \sum_i \left|\left(\phi(S)\right)_i\right| = \sum_{i:\, i\in S} 1 = |S|$. Therefore we have:

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

Which gives that $e^{-|S_i \cup S_j|}$ is a well-defined kernel because it is in the form $f(x)K(x,x')f(x')$ when K is a well-defined kernel (according to b).

d. Since k_1, k_2 are well defined kernels, we have:

$$k_1(x,x') = \phi_1(x)^T\phi_1(x'), \qquad k_2(x,x') = \phi_2(x)^T\phi_2(x')$$

$$\begin{aligned} k_1(x,x')\cdot k_2(x,x') &= \phi_1(x)^T\phi_1(x')\phi_2(x)^T\phi_2(x') \\ &= \sum_{i=1}^{n_1}\left(\phi_1(x)\right)_i\left(\phi_1(x')\right)_i\sum_{j=1}^{n_2}\left(\phi_2(x)\right)_i\left(\phi_2(x')\right)_i \\ &= \sum_{i,j}\left(\phi_1(x)\right)_i\left(\phi_1(x')\right)_i\left(\phi_2(x)\right)_j\left(\phi_2(x')\right)_j \\ &= \sum_{i,j}\left(\phi_1(x)\right)_i\left(\phi_2(x)\right)_j\left(\phi_1(x')\right)_i\left(\phi_2(x')\right)_j \end{aligned}$$

Define a feature map

$$\psi(x) = \left(\left(\phi_1(x)\right)_1\left(\phi_2(x)\right)_1, \dots, \left(\phi_1(x)\right)_1\left(\phi_2(x)\right)_{n_2}, \left(\phi_1(x)\right)_2\left(\phi_2(x)\right)_1, \dots \right.$$
$$\left. \dots, \left(\phi_1(x)\right)_2\left(\phi_2(x)\right)_{n_2}, \dots, \left(\phi_1(x)\right)_{n1}\left(\phi_2(x)\right)_1), \dots, \left(\phi_1(x)\right)_{n1}\left(\phi_2(x)\right)_{n_2}\right)$$

Then $k_1(x,x')\cdot k_2(x,x') = \psi(x)^T\psi(x')$ – a well-defined kernel.

e. Since $e^{|S_i \cap S_j|}$ and $e^{-|S_i \cup S_j|}$ are well-defined kernels, we get that:

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

is a well-defined kernel according to d.

f. Notice that :

$$\begin{aligned} K_{spam\ pro\ max}(S_i,S_j) &= e^{|S_i \cap S_j|-|S_i \cup S_j|} = e^{|S_i \cap S_j|-|S_i|-|S_j|+|S_i \cap S_j|} \\ &= e^{-\left((|S_i|-|S_i \cap S_j|)+(|S_j|-|S_i \cap S_j|)\right)} \end{aligned}$$

Meaning the kernel will get higher values to sets S_i, S_j where each has few distinct words – which is natural because we want emails that have similar word sets to get higher values. This is better than the previous kernel, which only considers the size of the intersection regardless of the distinct elements in each set which can greatly differentiate two emails (imagine sets S_i, S_j with $S_i \subset S_j$ but $|S_i| \ll |S_j|$).

g. Define:

$$\forall i \in [d], \quad W(i) = 1 - \frac{|\{S \in \Omega \colon \mathrm{i} \in S\}|}{|\Omega|} \in [0,1]$$

$$\psi\colon \Omega \to [0,1]^d, \quad \left(\psi(S)\right)_i = \begin{cases} W(i), & i \in S \\ 0, & o.w \end{cases}$$

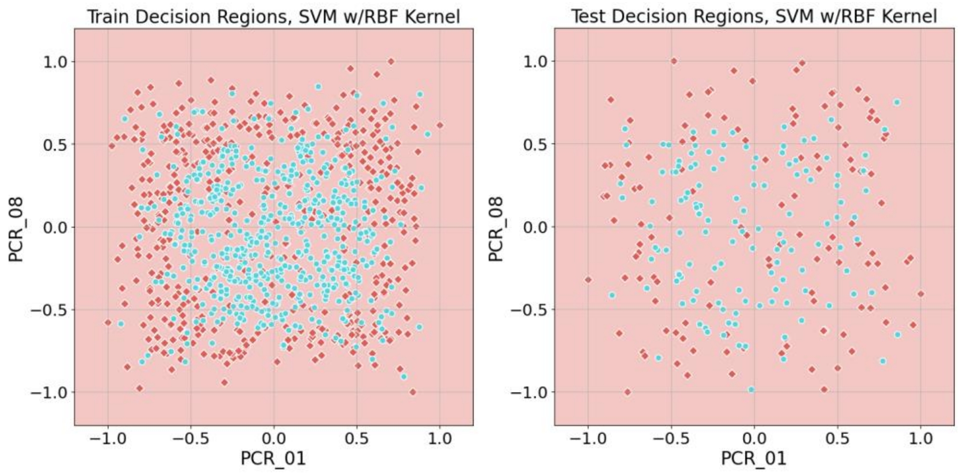
$$f(S) = \frac{1}{|S|}\sum_{i=1}^{d}\left(\psi(S)\right)_i = \frac{1}{|S|}\|\psi(S)\|_1 \in [0,1]$$

$$K(S_i,S_j) = f(S_i)\cdot K_{spam\ pro\ max}(S_i,S_j)\cdot f(S_j)$$

Explanation:
For each word in the email space, we define a weight function W that gives higher values to rare words (with low frequency in the email dataset).
Then, for each email in the dataset, we define a function f that calculate the average weight of words in the email – giving higher values to emails with many rare words and lower values for email with many frequent words.
Finally, we define the kernel to be $K(S_i, S_j) = f(S_i) \cdot K_{spam\ pro\ max}(S_i, S_j) \cdot f(S_j)$.
This means, that emails are evaluated based not only on the value of the previous kernel, but also based on how many rare words they include.
In particular, two emails that share many rare words will get higher values in our kernel than two emails that share many frequent words, which will get even higher values than two emails that don't share many words and include mostly frequent words.
This addresses the fact that most emails probably contain common words such as "the", "is", "a", "are", etc. and we do not want them to have the same weight as other words which may hold more information regarding the purpose of the email.
We believe that by also considering the rarity of words in emails, this kernel will provide more accurate classifications.

Q14:
Use sklearn.svm.SVC to train an SVM with an RBF kernel on the two features and the spread variable, with C=1 and γ =1e-7. Plot the model's decision regions and include this plot in your report. Additionally, provide the training and test accuracies of the model. Based on these accuracies, would you classify the model as underfitting, overfitting, or as a reasonable fit?

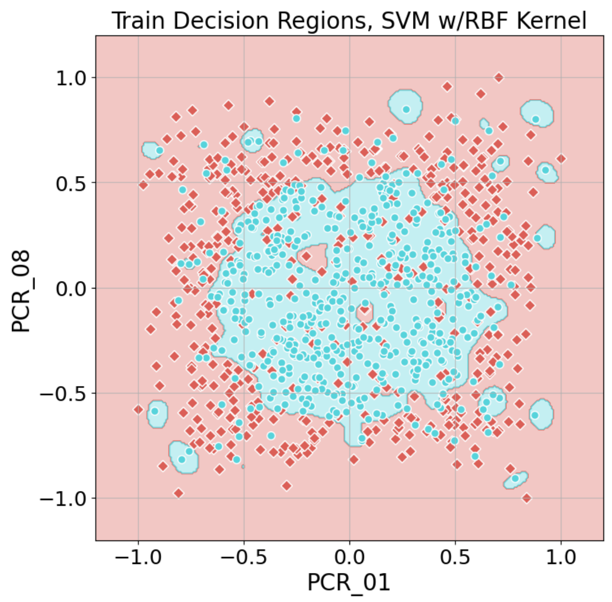
A:

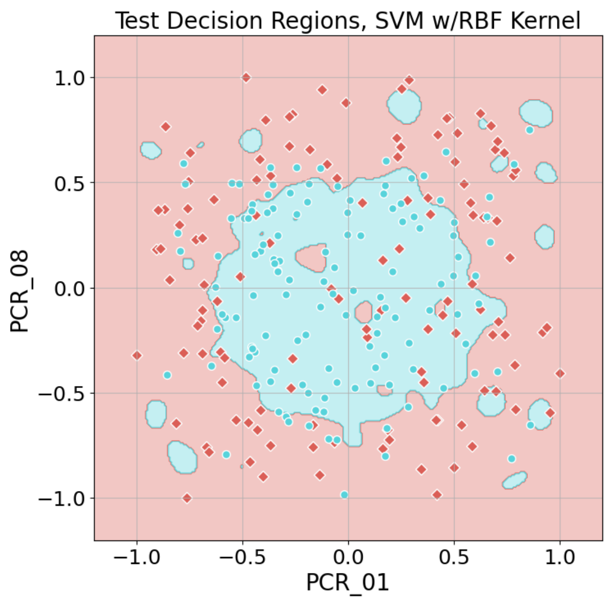


The model's train accuracy is: 52.8%.
The model's test accuracy is: 51.2%.
Based on the accuracies, we would classify the model as underfitting because the train accuracy is low.

Q15:
Use sklearn.svm.SVC to train an SVM with an RBF kernel on the two features and the spread variable, with C=1 and γ =200. Plot the model's decision regions and include this plot in your report. Additionally, provide the training and test accuracies of the model. Compare the decision regions of this model with the k-NN model with k optimal from (Q3). We expected the two models to be very similar, but we should still see significant differences. Try to explain the differences (there isn't a single correct answer).

A:



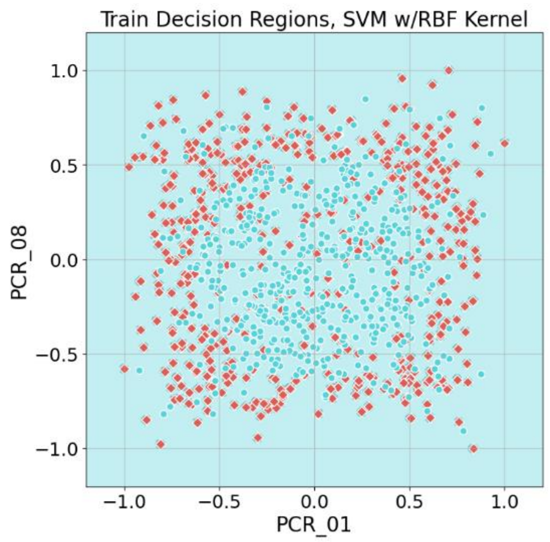


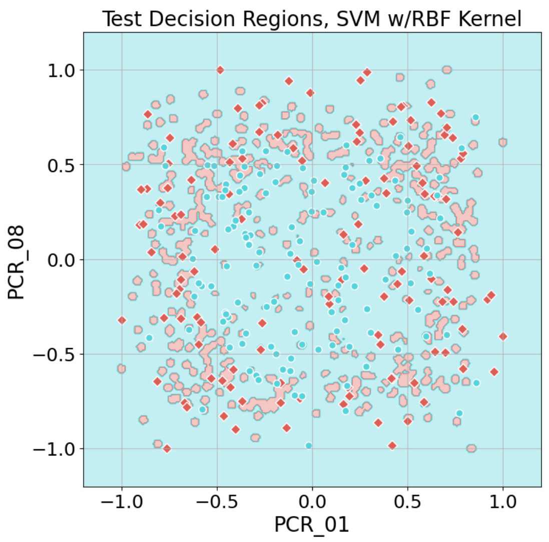
The model's train accuracy is: 85.1%.
The model's test accuracy is: 72.8%.

As we can see in the plots, the decision regions in the rbf model are quite different than the decision regions in the 11-NN model. We believe this has to do with the fact that a large gamma makes the rbf kernel heavily consider nearby samples (regarding the numeric value of their distance) when classifying new data, resulting in decision areas more similar to 1-NN. Meaning isolated samples define a decision area with similar labels, as no other points are close enough to influence the decision. This does not happen in 11-NN since the model considers the nearest neighbors regardless of the value of their distance – only that it's closer than other samples.

Q16:
Use sklearn.svm.SVC to train an SVM with an RBF kernel on the two features and the spread variable, with C=1 and γ = 5000. Plot the model's decision regions and include this plot in your report. Additionally, provide the training and test accuracies of the model. Based on these accuracies, would you classify the model as underfitting, overfitting, or as a reasonable fit?

A:





The model's train accuracy is: 99.3%.
The model's test accuracy is: 57.6%

Based on the accuracies, we would classify the model as overfitting because the train accuracy is high but the test accuracy is low.