Major 2

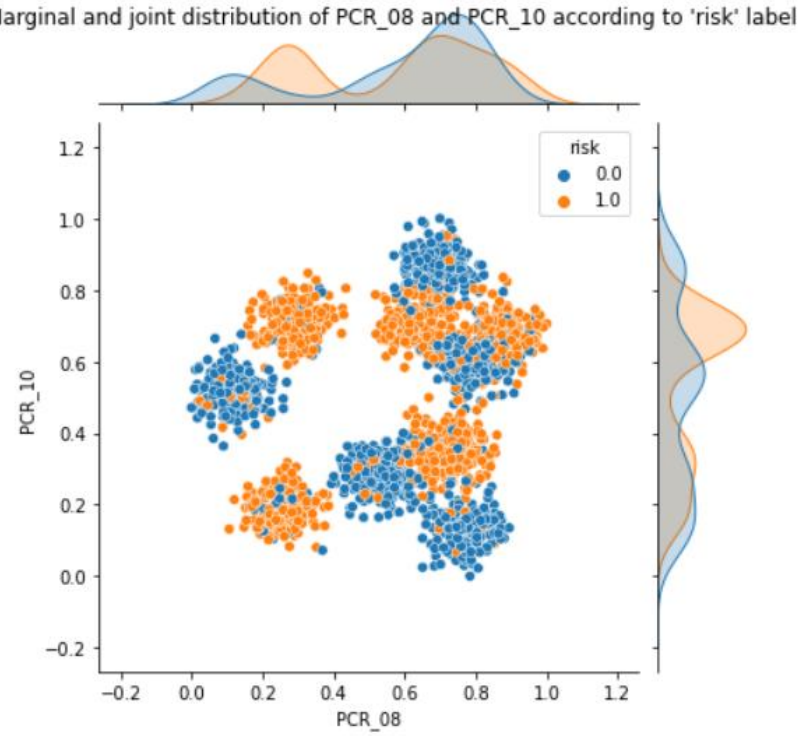
Alon Ziv 208489971

Nitai Kluger 318933785

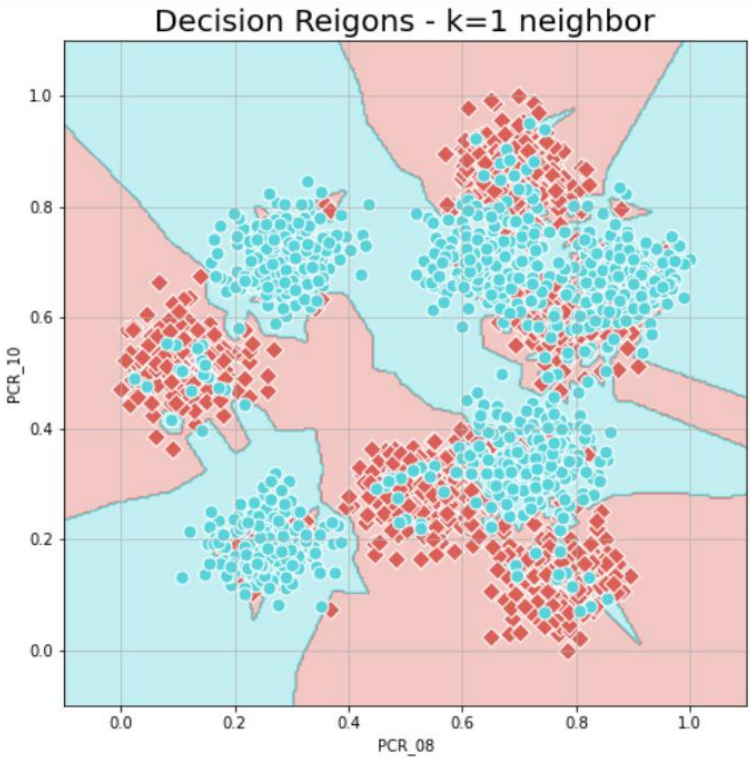
Part 1: Basic model selection with k-Nearest Neighbors

Visualization and basic analysis

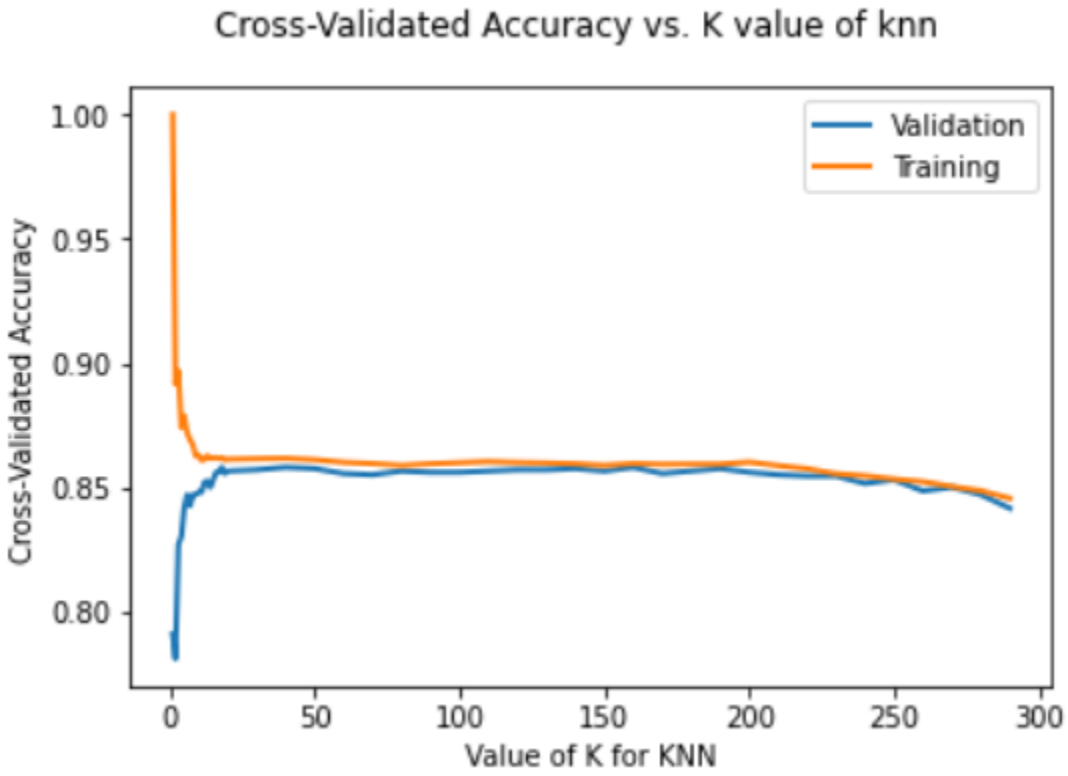
A1:



A2:



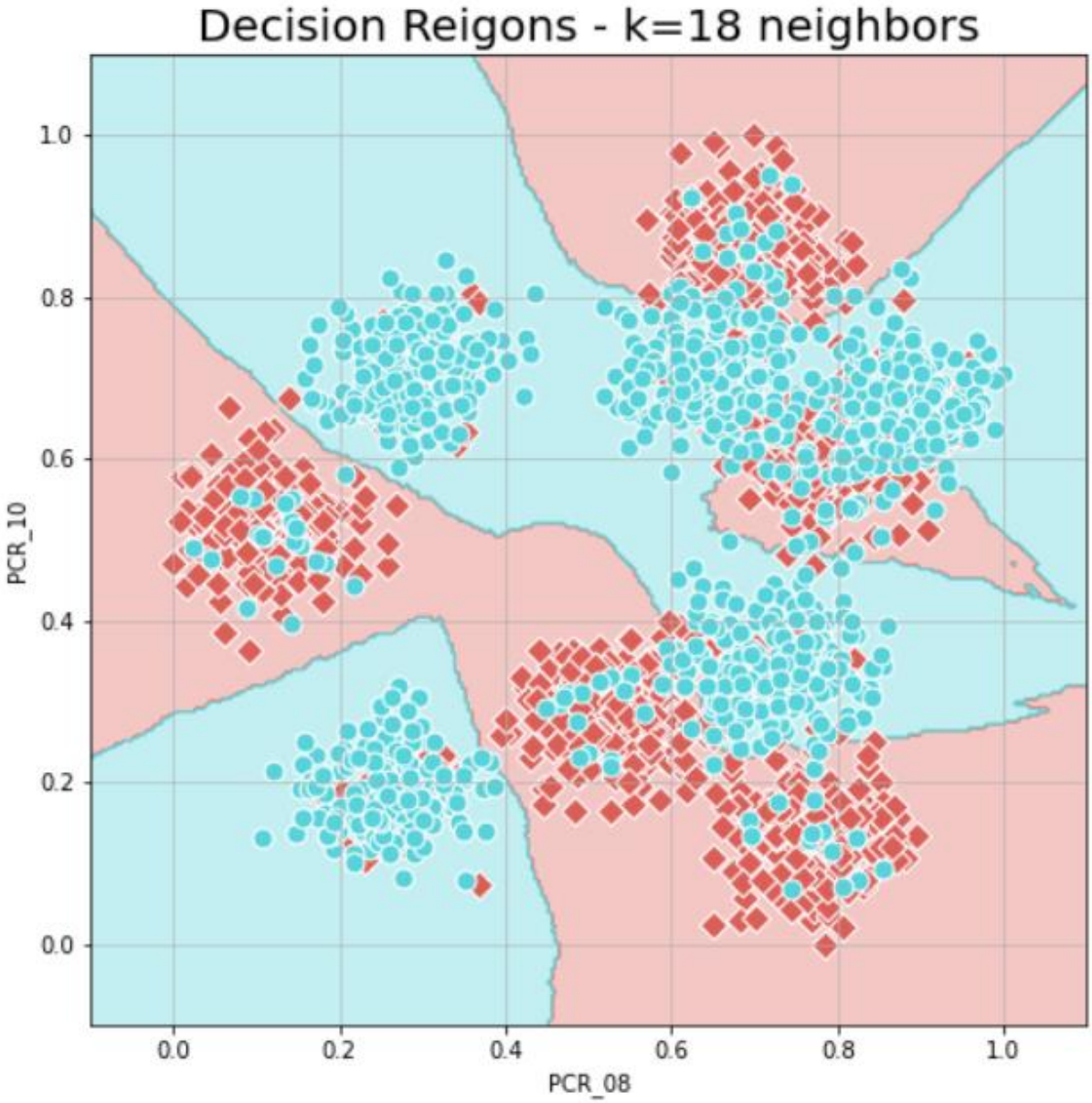
A3: The orange graph is training mean accuracy and blue is validation:



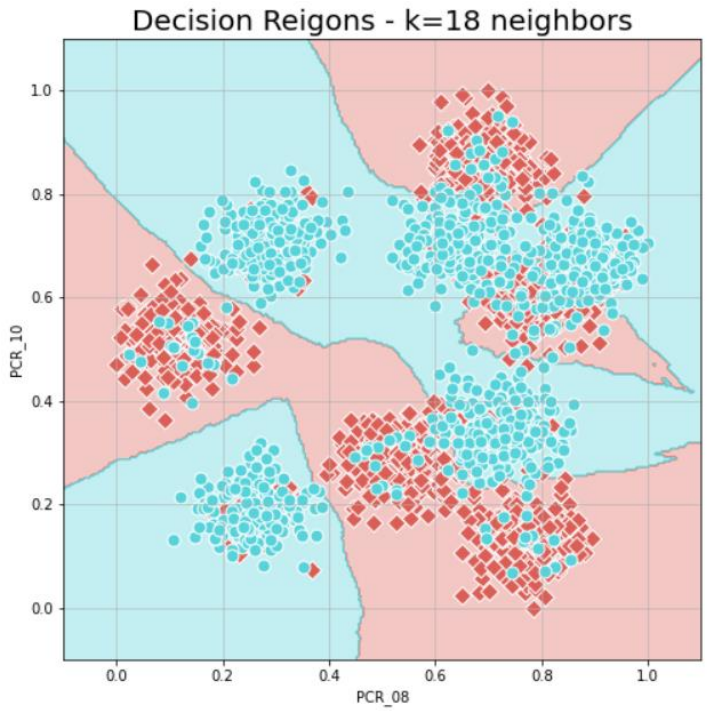
We noticed that for values of k > 30 we get pretty much the same results for both train and validation. An underfitting k would usually be a very large one. A large k is not accurate and relies on neighbors that aren't close. We don’t want to go any further than k = 30. An overfitting value would be a value with very high training accuracy, like k=1 in that example. In addition, the highest test score is for k =18, and it lies in some sort of sweet spot between k=1 and k=30. We chose k=18. Its mean accuracy is 0.855

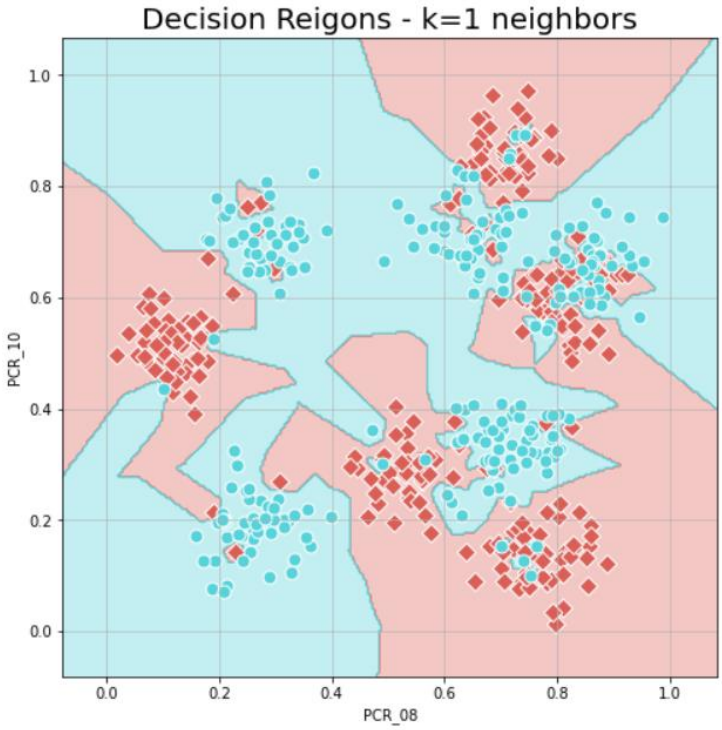
A4:

The test accuracy is: 0.852.


Decision Reigons - k=18 neighbors
1.0
0.8
0.6
0.4
0.2
0.0
PCR_10
0.0
0.2
0.4
0.6
0.8
1.0
PCR_08

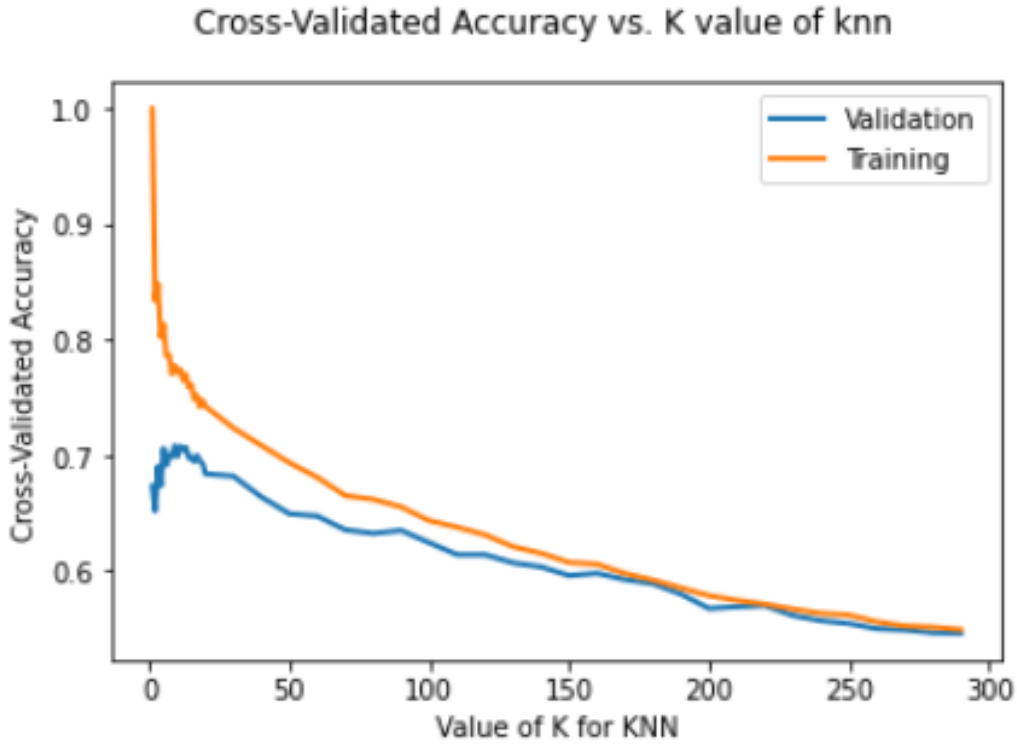
A5: We examined the model of k=1 and k=18 over the test data and got:





With k=1 we receive better training accuracy than with k=18, even though the highest test accuracy was using k=18. We conclude that the high score of k=1 during the training is as a result of overfitting rather then of choosing the optimal k, for its very low test results.

A6:



The operation of a knn model is to infer a label by the k "closest" entries' labels, i.e the k entries with the minimal distance from the given entry to test. It is noticeable that when putting into account **all of the features** and not just the two from Q3, all k values produce worst results.

In KNN models, the closest neighbor is determined by the smallest Euclidian distance:

$$distance = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

When aggregating the other features, their effect on the distance is also calculated. The final vector's location in this n-dimension space is eventually confusing the model rather them pointing it towards the right direction.

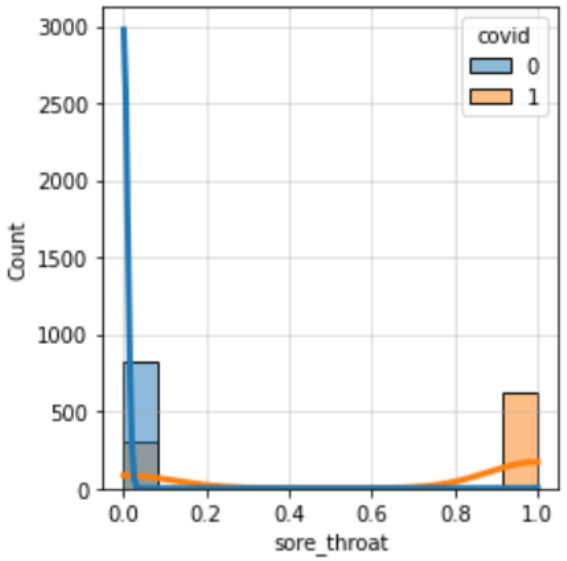
This happens because the other features are not creating the geometric classification of 'risk' clusters. In Q3, the 2 features managed to separate the data to different values of risk.

We conclude that this model is not the best choice for predicting 'risk' with all features.

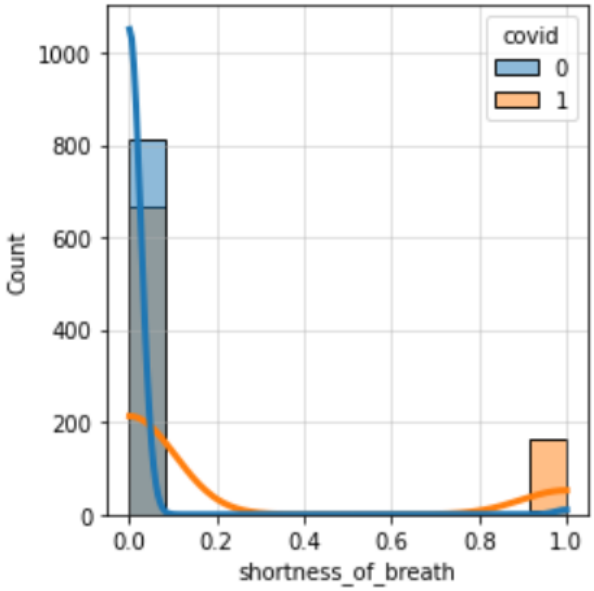
Part 2: Decision trees
Visualization and basic analysis

A7:
The features are **shortness_of_breath, sore_throat, sport_activity, cough**. They are listed in the plots with binary values of 1 and 0 and not 1 and -1 (as we did in the previous major 1) but the meaning is equal.
The histplots of the features with the covid label:

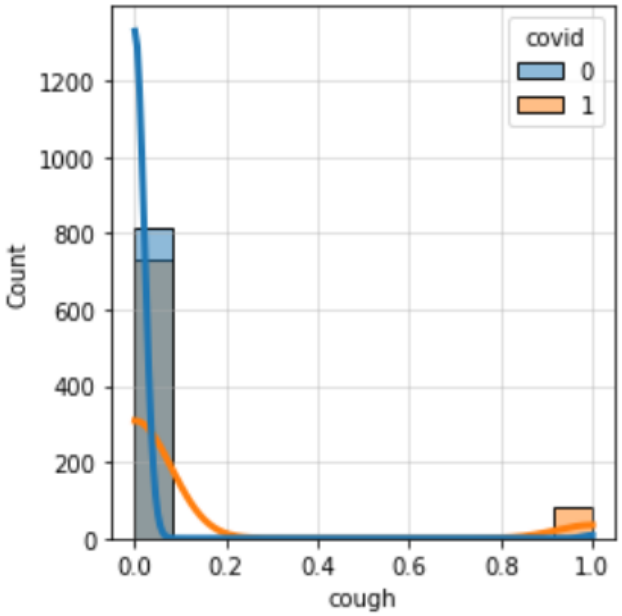
Distribution of sore_throat mapped with covid label



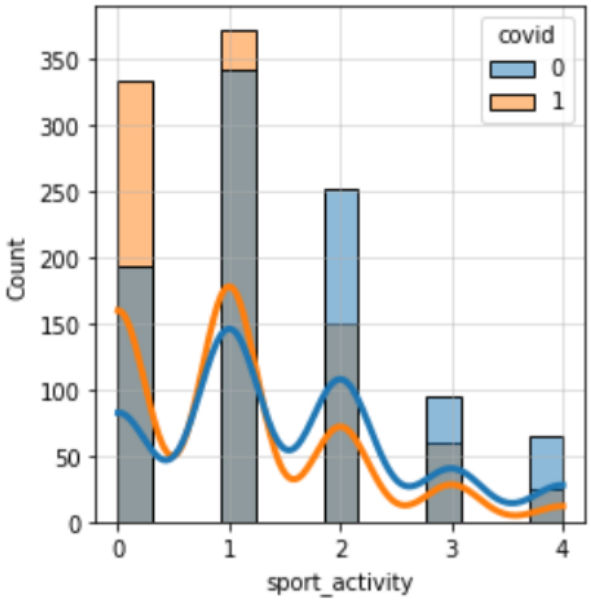
Distribution of shortness_of_breath mapped with covid label



Distribution of cough mapped with covid label



Distribution of sport_activity with covid label



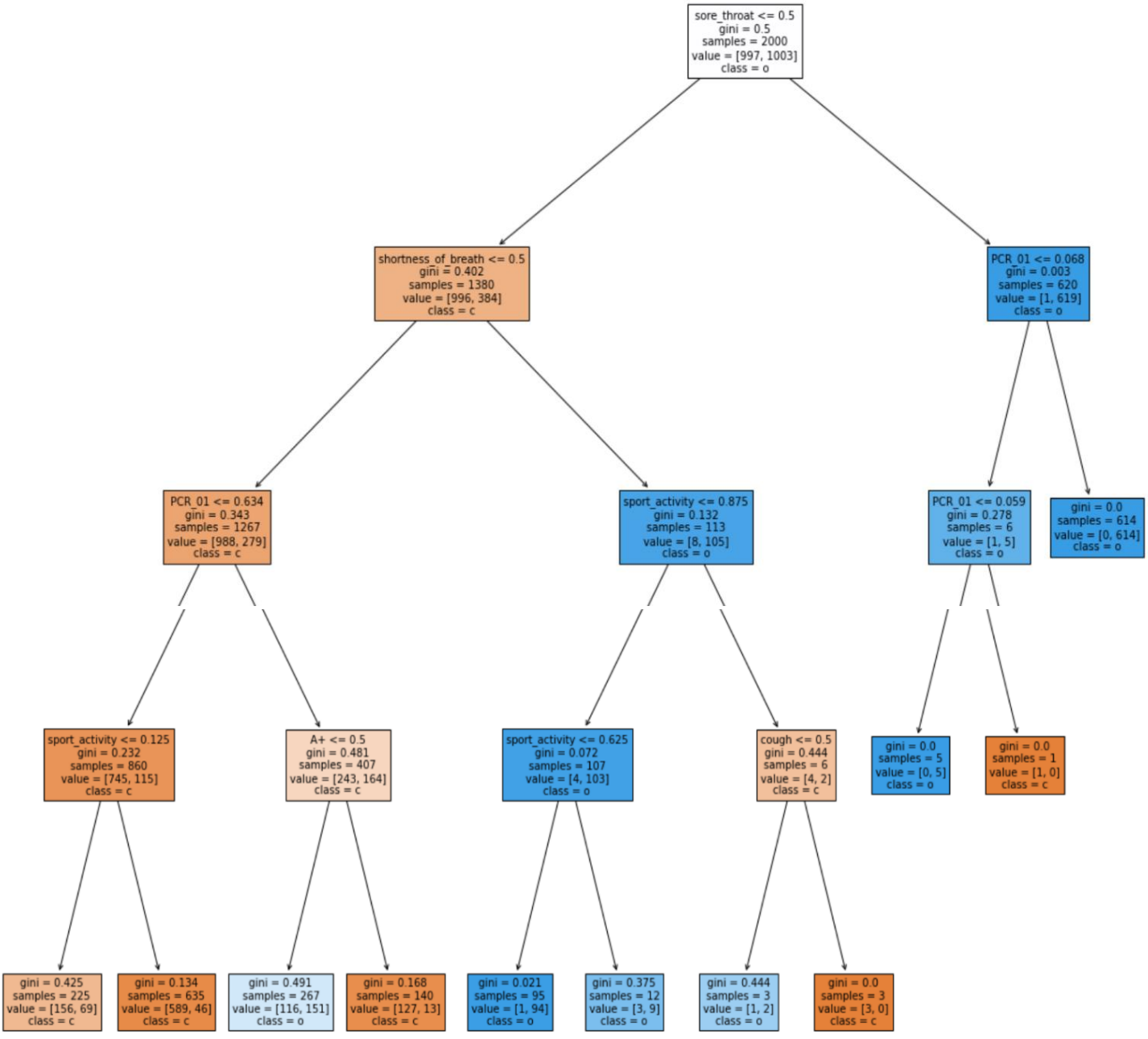
A8: The 10 most correlated features in absolute value from our selected features on major 1 are:

sore_throat	0.666110
shortness_of_breath	0.276242
sport_activity	0.202510
cough	0.189513
PCR_01	0.161217
PCR_04	0.112623
A+	0.090874
sugar_levels	0.080202
fever	0.044107
PCR_05	0.041521

A9:

The training accuracy is 87.55%.

Decision Tree of maximum depth 4,
to predict the covid label



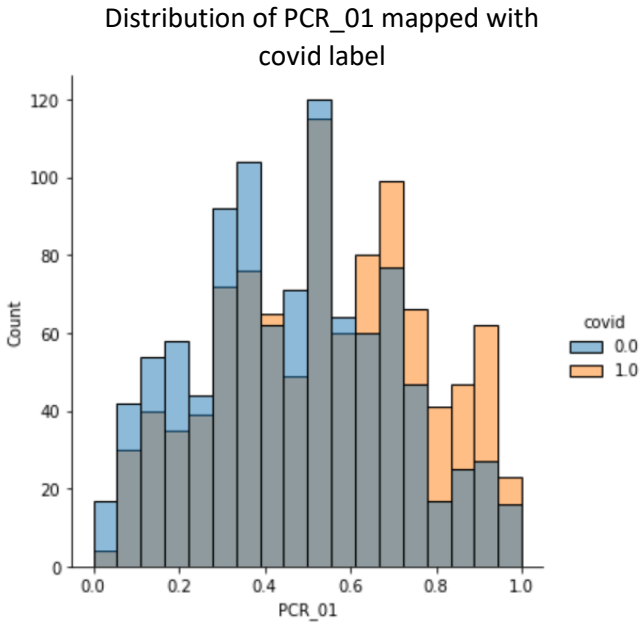
A10:

The importance of the features from Q7 is very much noticeable, as the all 4 correlated features are of the features we chose. Correlation is a parameter for how much two variables are linearly dependent. The closer they are to being linear, the higher the correlation.

The ID3 decision tree also used sore_throat and shortness_of_breath as primary filters, but it also used PCR_01, which we did not mention in Q7. This model searches to divide the dataset with each "step" as best as possible for determining what is the predicted label.

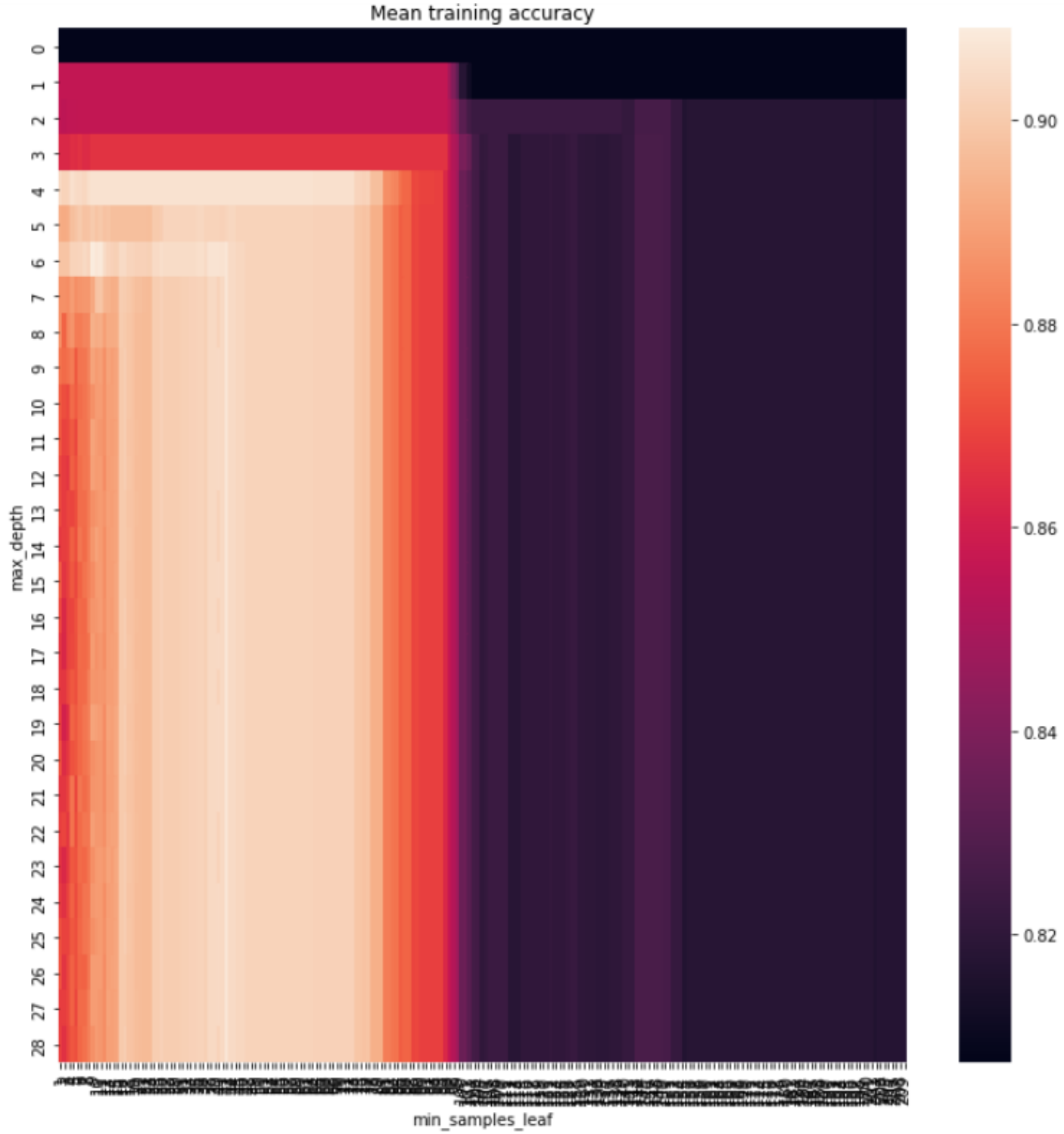
The reason for similarities in the ID3 choice and the most correlated features is that correlated features and especially binary ones can be a great threshold of a label for the tree to use early on in the process of prediction. The reason for differences is that correlation, or linear dependency, doesn't necessarily mean that the distribution of the samples according to that feature **is separable** between the label's 1 and -1. An ID3 model depends on separability (calculated by entropy, not correlation) of the features when selecting the next "rule" for the tree, rather then upon linearity.

Although not as highly correlated, it's noticeable when a feature is separable:

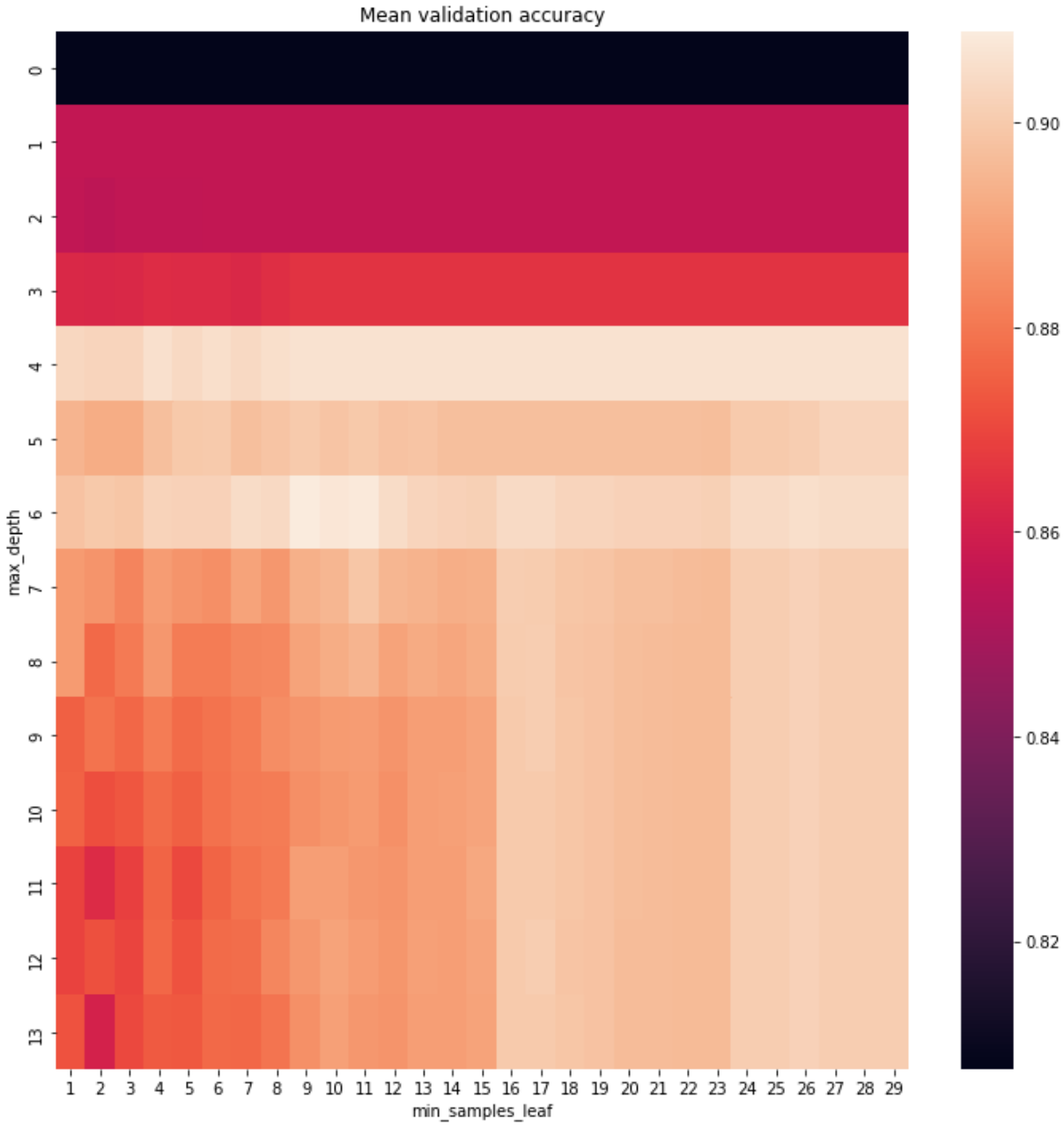


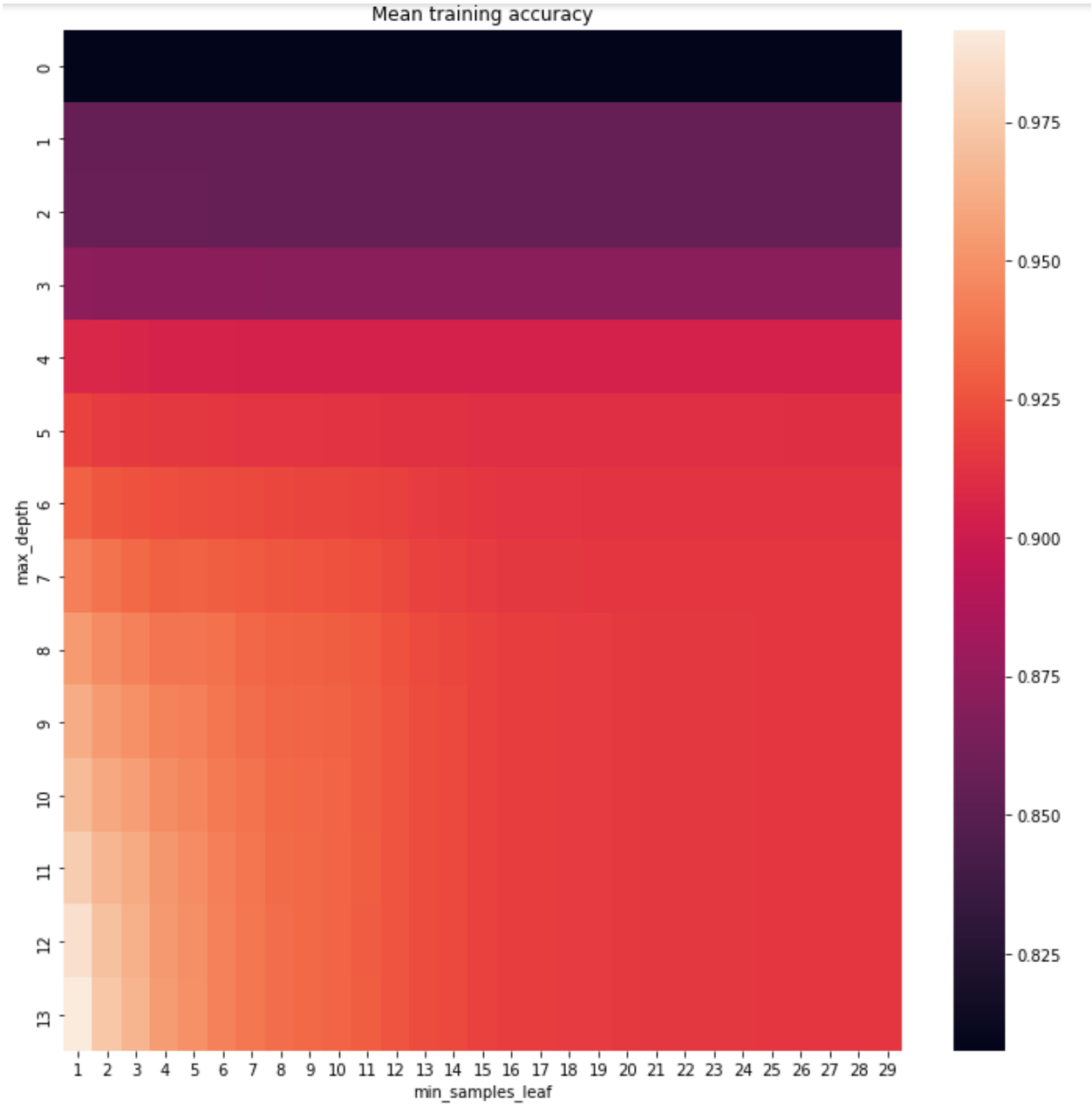
A11:

A. When calculating the range of max_depth, we used the cross_validate with 2 folds and noticed that for any tree with larger depth then 30, the accuracy remains similar- around 85%. We decided no need to go any further then that model complexity for the results aren't that better. When calculating the range of min_samples_leaf, we again used the cross_validate with 2 folds from the range of 1 (best fitting possible for the training set) to the size of the dataset (the worst fitting). We saw that the accuracy remains until min_samples_leaf = 147 similar and equal to 82.45%. To make sure we're not overfitting, we decided the range should be as high as min_samples_leaf = 210, that reaches accuracy of 210.



We noticed a clear difference line around min_samples_leaf = 100; higher min_smaples_leaf than 100 aren't helpful. To get better resolution and more relevant results we redid the process using min_s_leaf= 50, max_depth=15:


Mean validation accuracy
max_depth
0
1
2
3
4
5
6
7
8
9
10
11
12
13
min_samples_leaf
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
0.90
0.88
0.86
0.84
0.82

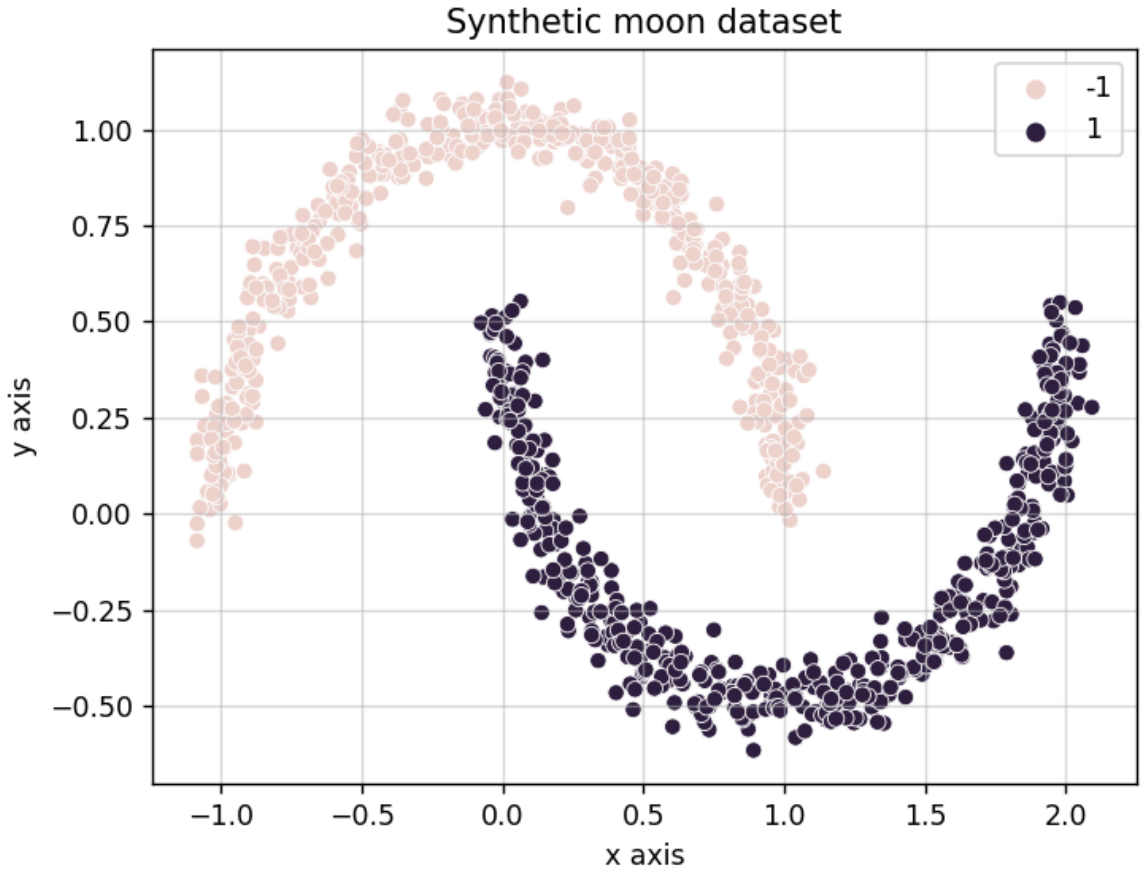


B. And C, above
C. The optimal pair is: max_depth = 7, min_samples_leaf = 9
D. Max_depth = 1, min_samples_leaf = <size_of_dataset>
E. Max_depth = <size_of_dataset> - 1 , min_samples_leaf = 1
F. The combination from D is underfitting because it allows max_depth of 1, which means only a single deviation is allowed (not informative enough). Moreover, the min_samples_leaf is largest possible, meaning a leaf could contain even the entire dataset.
The combination from E is overfitting because it dictates the deepest tree possible, which will lead to the best train accuracy possible with this model. Furthermore, the min_samples_leaf is

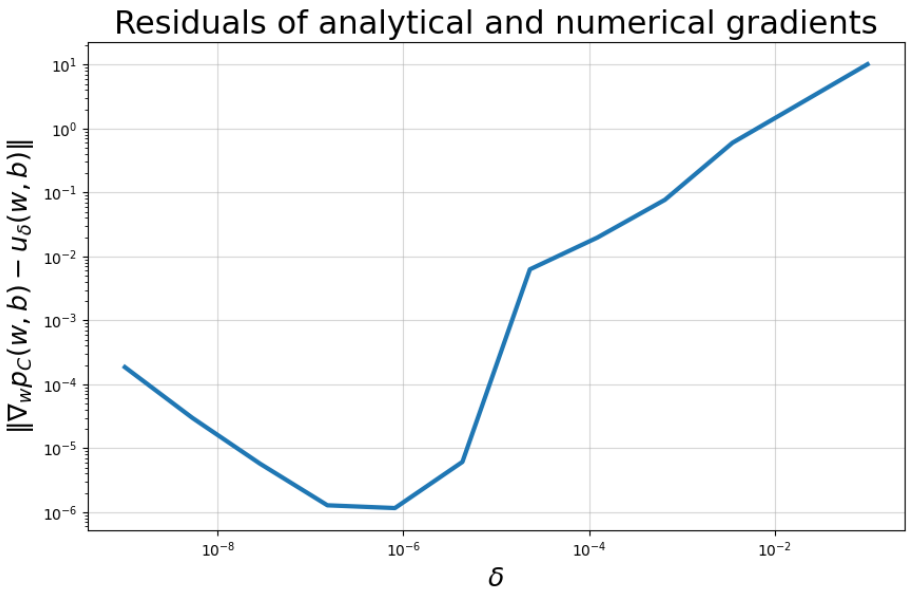
the minimum, which generates a very complicated and specific prediction based on the given dataset.

A12: The test accuracy is 91.6%.

A13. The plot:

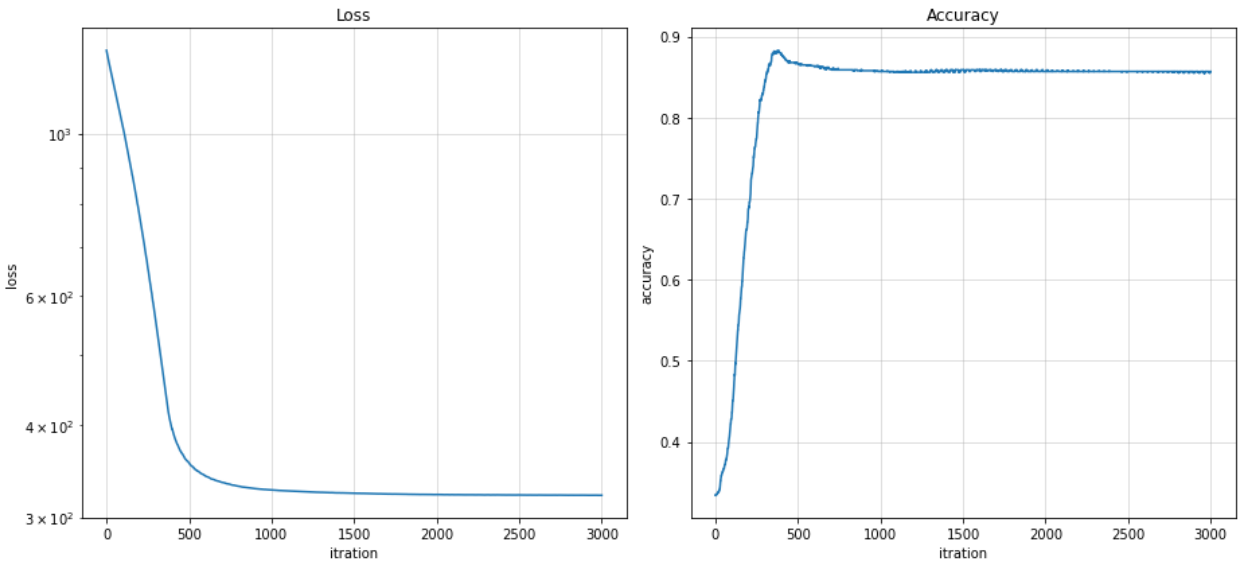


A14. The plot:



As we can see the numerical gradients and the analytic gradients bahave simmilar with a tuned size of delta. If delta is too little numerical errors will cause error in the numerical gradients behaviour because we are dividing with a number close to zero. On the other hand, if delta is too large the numerical gradients arent a good approximation to the analytic gradients since the derivative theroem demands delta should be very small.

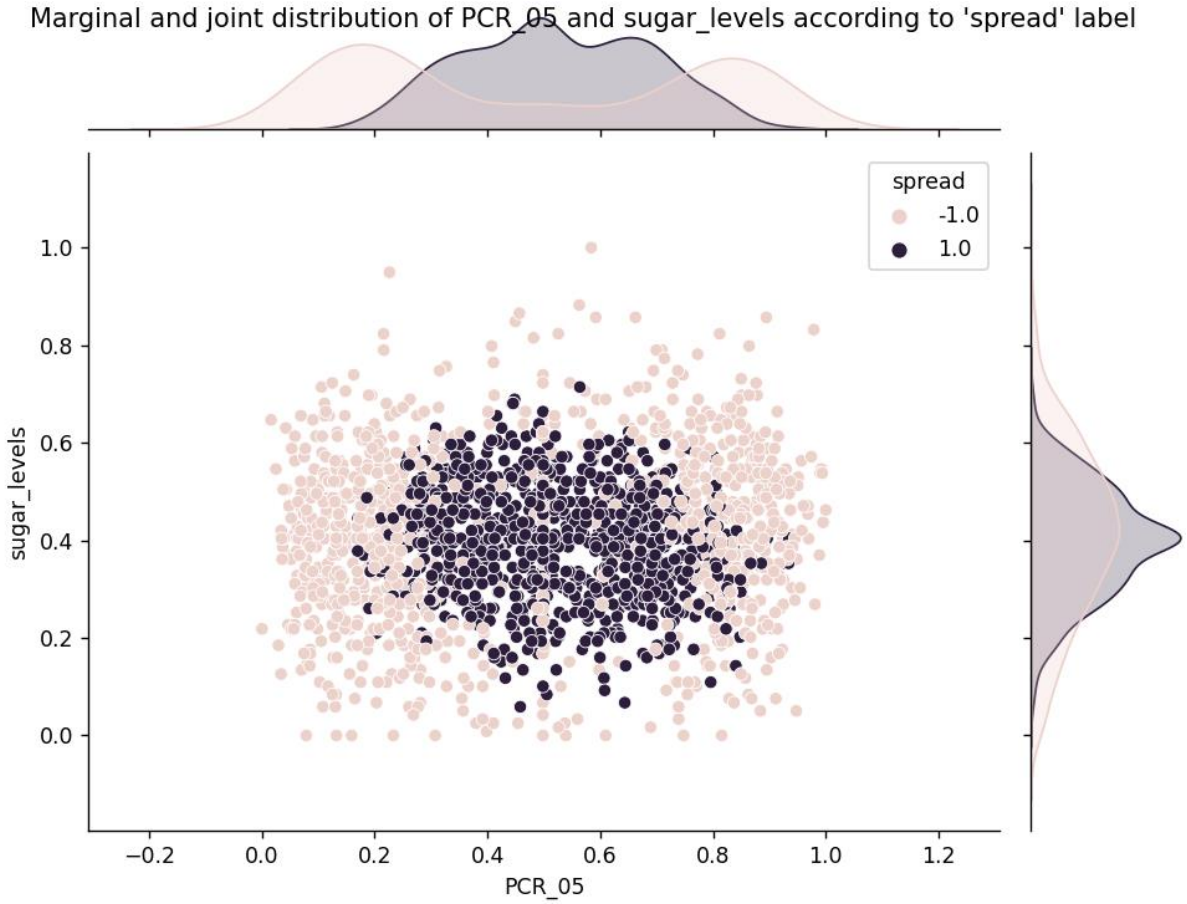
A15. The plot:



The model reached an accuracy peak of 88 % after 400 iterations, and then stayed at a lower stable value of 86 %. Meantime the model reached the loss minimum of 300, only after 1000-

1500 iterations, and stayed stable at that value. The reason why there are not attained at the same step is that the model tries very hard to minimize the loss caused by the outliers of the data. While the metric “accuracy” considers each point equally, the loss metric does not, a point in the outliers raises the loss significantly compared to the other points. Because the target of the model is minimizing the loss, it will pursuit the loss minimum while hurting the accuracy.

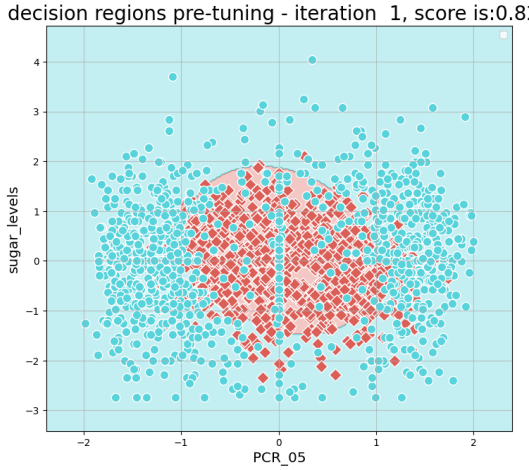
A16. The plot:


Marginal and joint distribution of PCR_05 and sugar_levels according to 'spread' label
spread
-1.0
1.0
sugar_levels
PCR_05
0.0
0.2
0.4
0.6
0.8
1.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

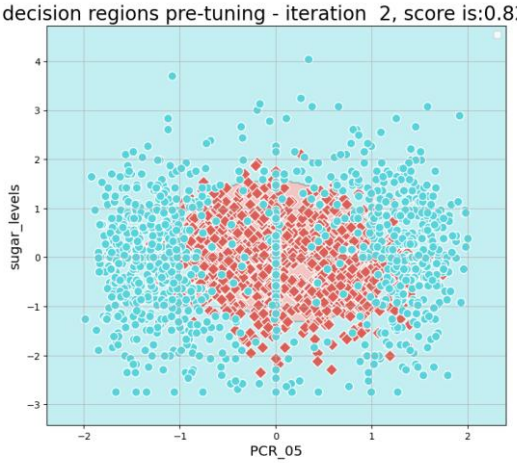
A17. the scores are (0.827, 0.8255, 0.831, 0.8185, 0.845), the mean is 0.829 and the standard deviation is 0.00878.

The decision boundaries of each iteration are described in the following plots:

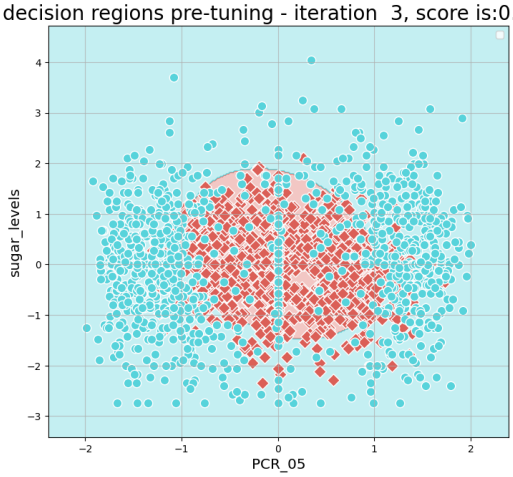
decision regions pre-tuning - iteration 1, score is:0.827



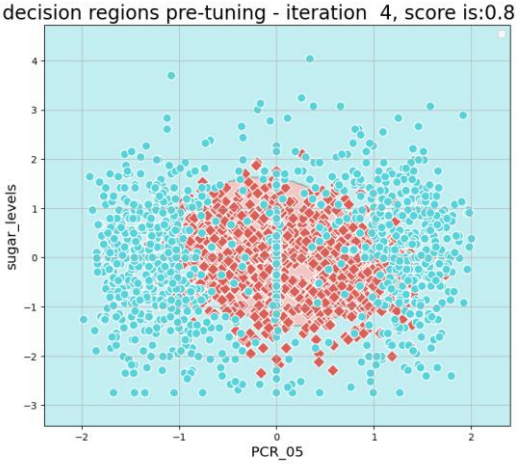
decision regions pre-tuning - iteration 2, score is:0.8255



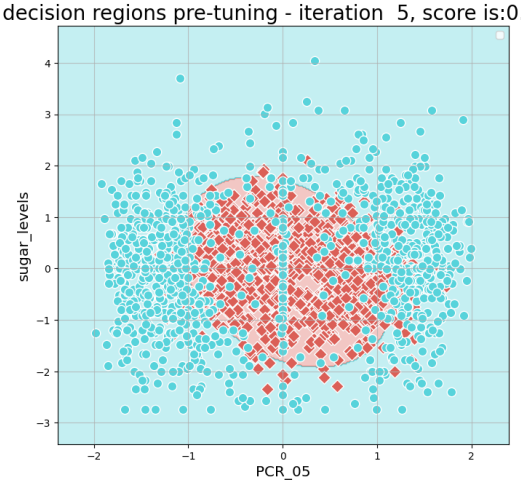
decision regions pre-tuning - iteration 3, score is:0.831



decision regions pre-tuning - iteration 4, score is:0.8185



decision regions pre-tuning - iteration 5, score is:0.845



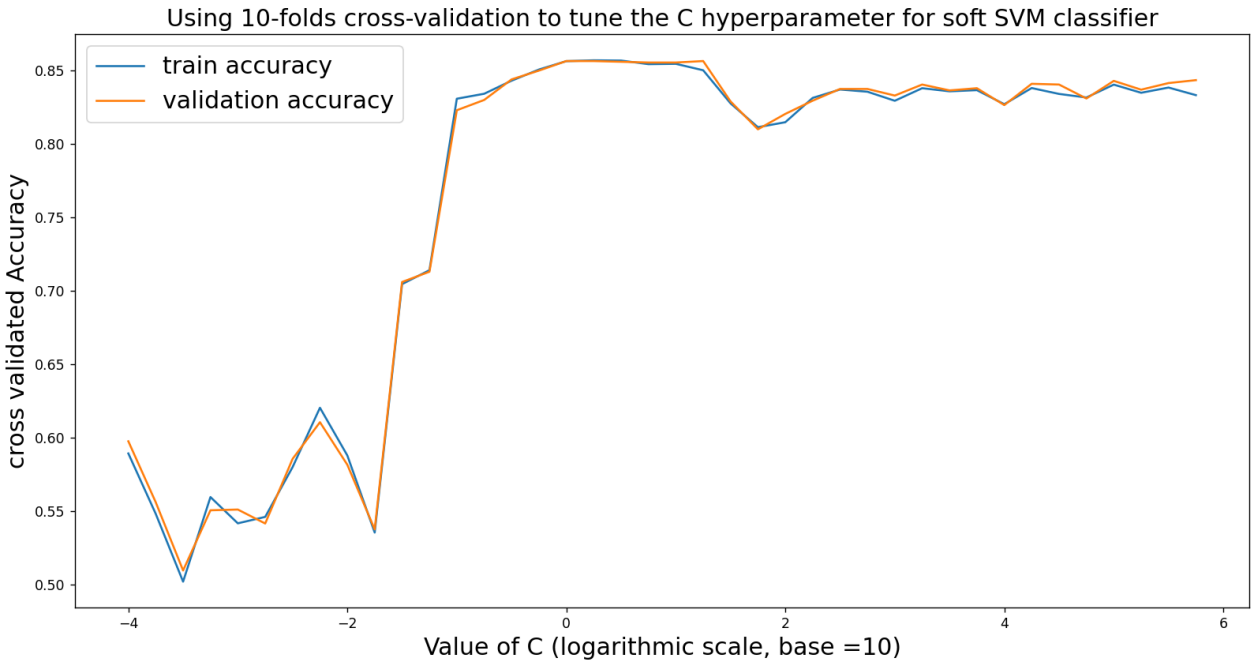
We can see that all the produced models with the current hyperparameter are very similar and have similar accuracies. The reason is that the loss function is convex, so it has unique global minimum that every model tries to converge to with every iteration.

The reason why there are still differences is the stochastic part of the SGD, in which each iteration is determined by a small batch, so it differs in every iteration and between models. Another possible reason for the differences is a relatively large C value and large learning rate, causing large gradient steps that may cause skipping “forward and backward” around the minimum.

A18.

First, we chose a broad range, $(10^{-4}, 10^{6})$.
with that range while using logarithmic sampling, we found that an optimal sub range is :
$(10^{-1}, 10^{2})$

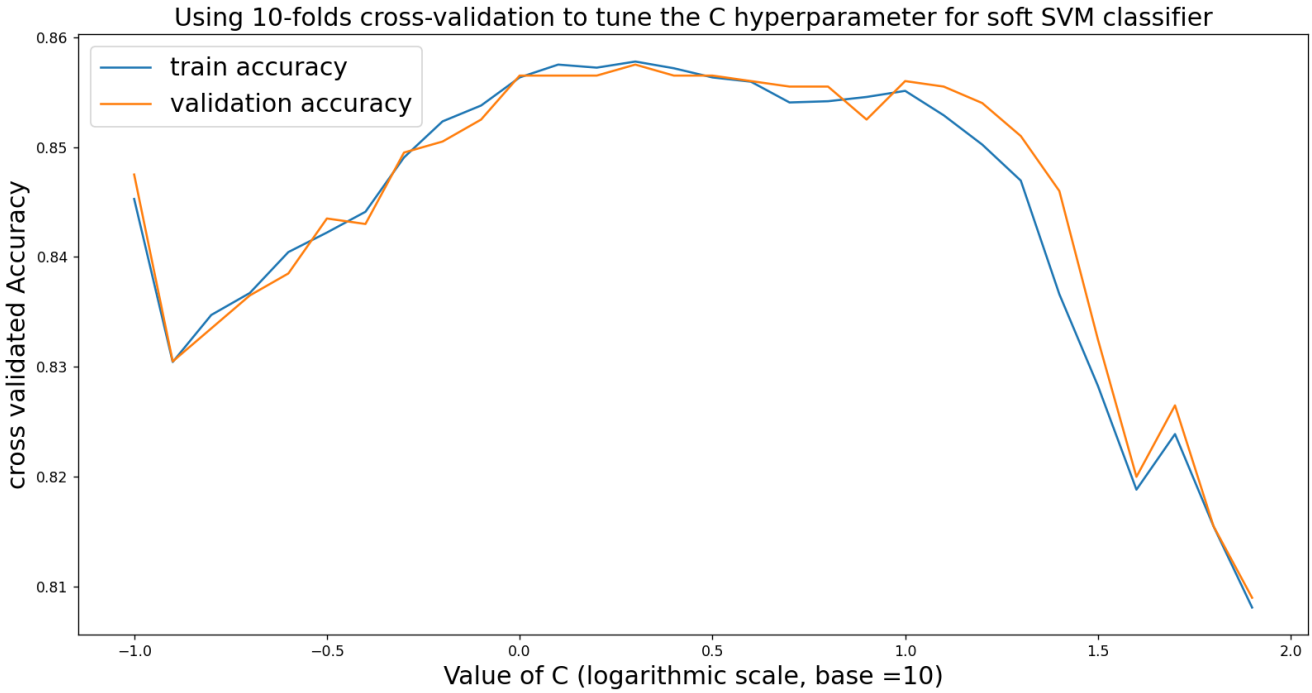
The plot:



We can see while C has a very low value, the scoring a bit below or above 0.5. this happens because the model is underfitting - compared to the norm of w or the regularization, the error in predicting the labels is very low, so the model is determined to minimize as possible the $l_2 \ \ norm\ of\ w$, leading it to classify all the dots as with the same label (in this case, -1).

On the other hand, when C has a very high value, the regularization has a low effect, and the model is more sensitive to outliers and noise in the data. This causes overfitting and poorly generalization to the validation set and therefore the model a lower validation accuracy. Furthermore, the gradient step is dependent on C, so as C grows so as the gradient step grows, thus preventing the model to converge to the minimum.

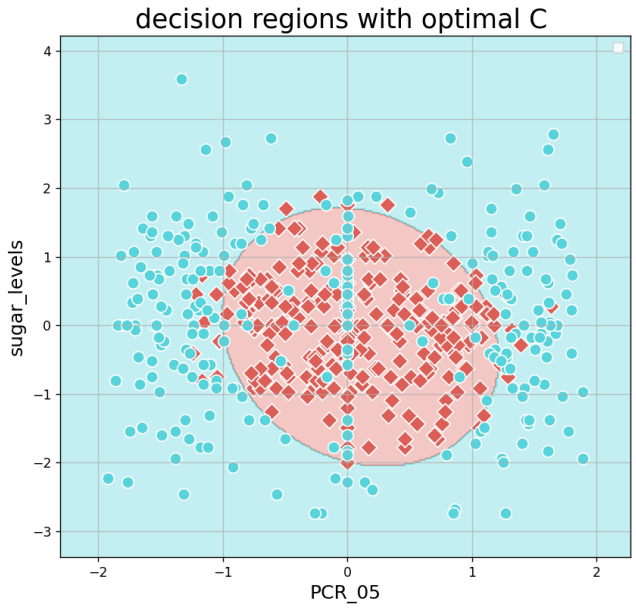
Now, we repeated the process with this range and higher resolution.



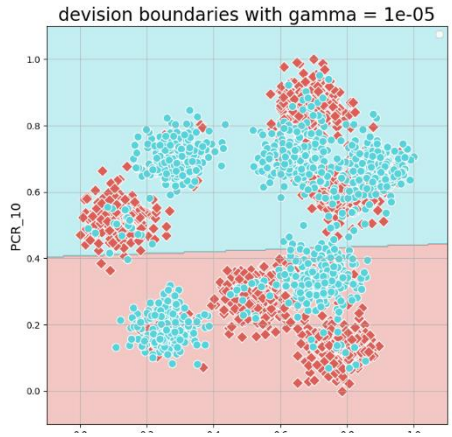
A19. From the above we conclude that:

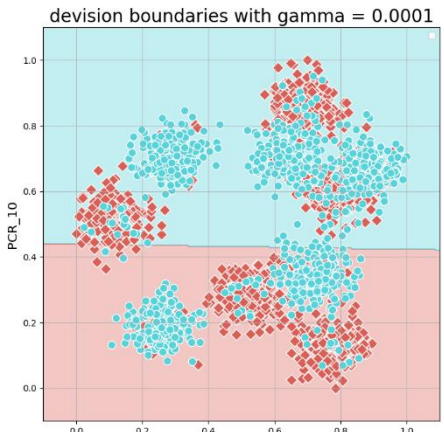
- the best C w.r.t to validation accuracy is: $\boldsymbol{C_{opt} = 2}$ while $\boldsymbol{accuracy_{\text{validation}} = 0.8574}$
- the best C w.r.t to train accuracy is: $\boldsymbol{C_{opt} = 2}$ while $\boldsymbol{accuracy_{train} = 0.8577}$

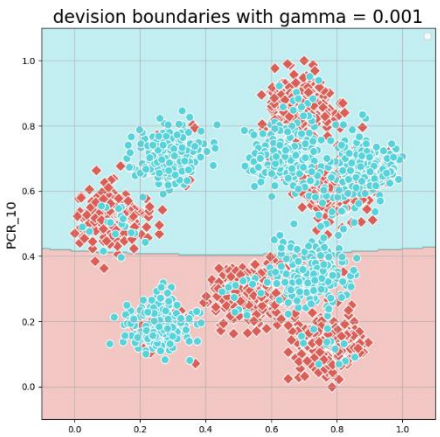
Now, let's see the decision regions:

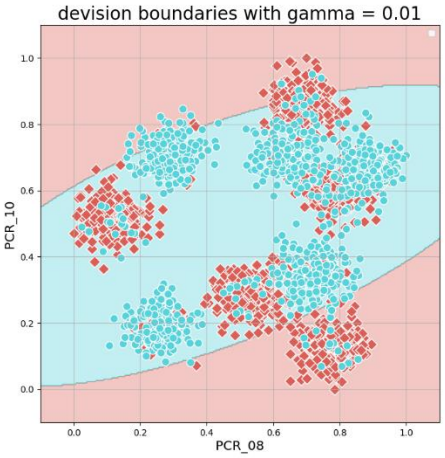


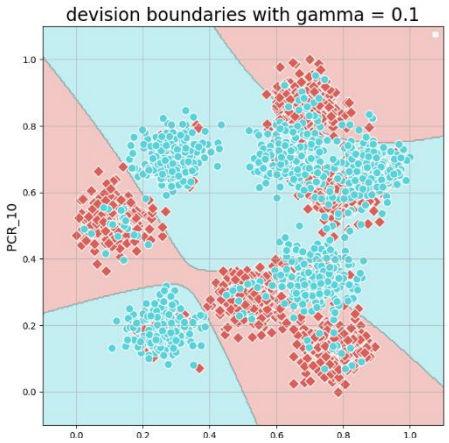
The test accuracy is 0.85.

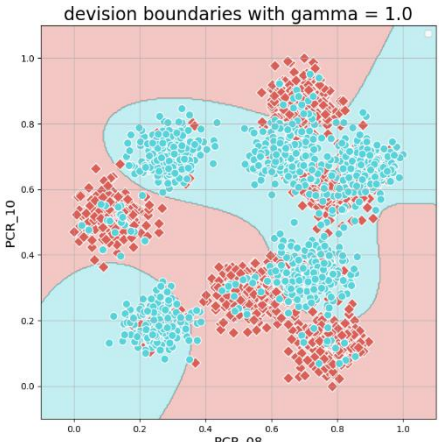

devision boundaries with gamma = 1e-05
PCR_10

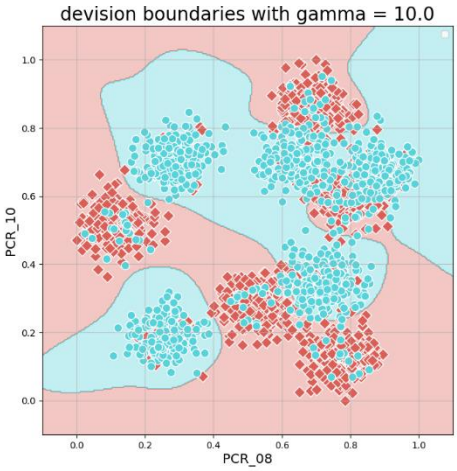

devision boundaries with gamma = 0.0001
PCR_10

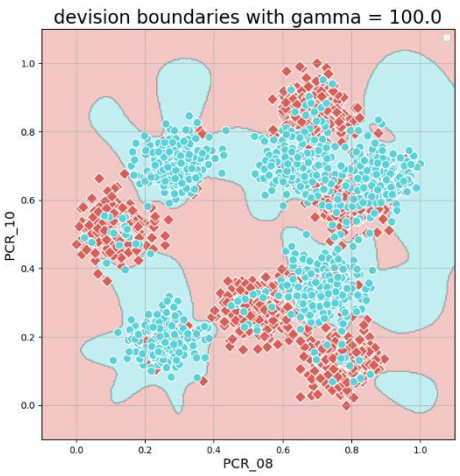

devision boundaries with gamma = 0.001
PCR_10

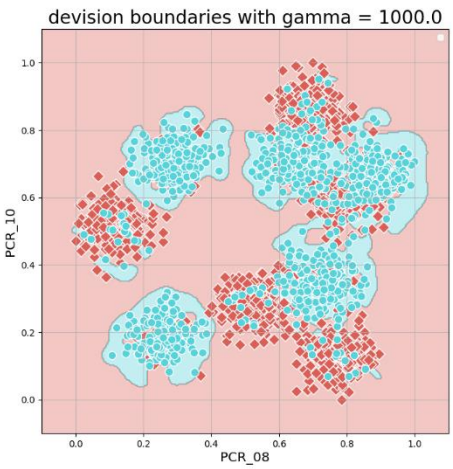

devision boundaries with gamma = 0.01
PCR_10
PCR_08

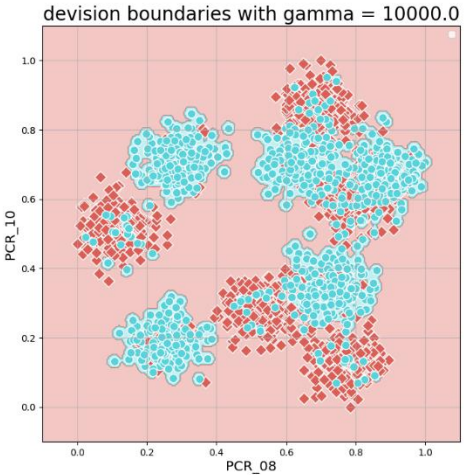

devision boundaries with gamma = 0.1
PCR_10


devision boundaries with gamma = 1.0
PCR_10


devision boundaries with gamma = 10.0
PCR_10
PCR_08


devision boundaries with gamma = 100.0
PCR_10
PCR_08


devision boundaries with gamma = 1000.0
PCR_10
PCR_08


devision boundaries with gamma = 10000.0
PCR_10
PCR_08

A21.

As we seen in class, the rbf clf has $Vcdim = \infty$ and therefore can shatter every possible set. We see that quality in the graph - the soft svc with the rbf kernel and gamma = 10^4 (from now referred as rbf clf) Is extremely overfitting the data, resulting in small decision boundaries surrounding each point.

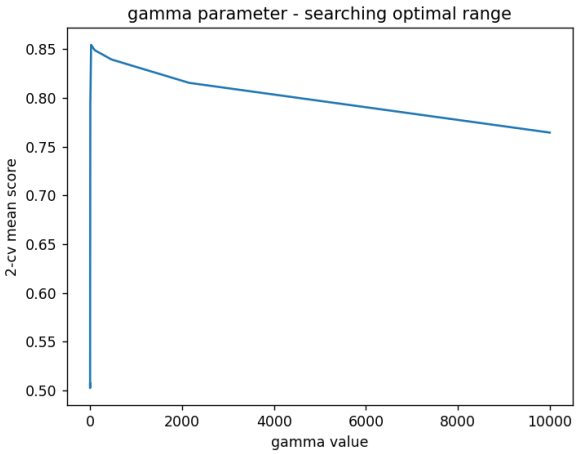
However, this is not the case with KNN, so while the 1-NN model is also overfitting the data, it is deterministic and not iterative and therefore can't overfit beyond a certain point.

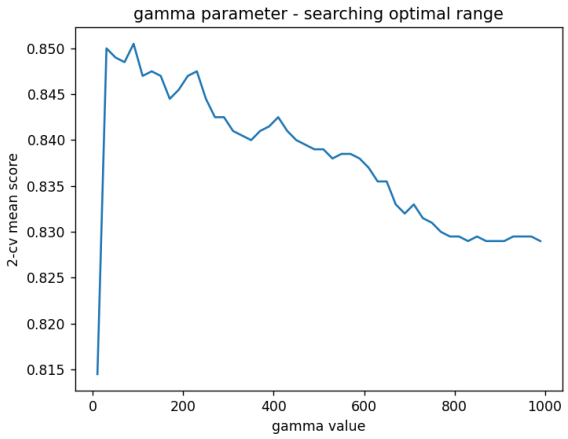
Another difference is that the decision boundaries of the rbf clf are unbalanced, most of the region is red (label -1) because the model limits the boundaries to the specific data. On the other hand, the decision boundaries of the 1- NN are relatively balanced between red and blue regions, Because the model determines the region color by its closest dot, even if just 1 of them appears in a region.

A22. From the above graph we can see that appropriate range for gamma is $(10^1, 10^3)$

Let's validate this assumption using 2-cross validation:

On the left – broad range of gamma, on the right - narrower range respectively.

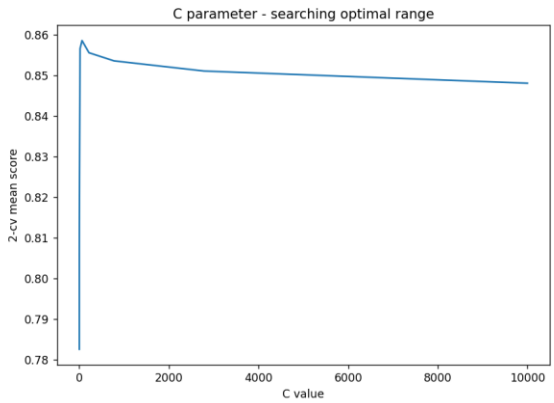


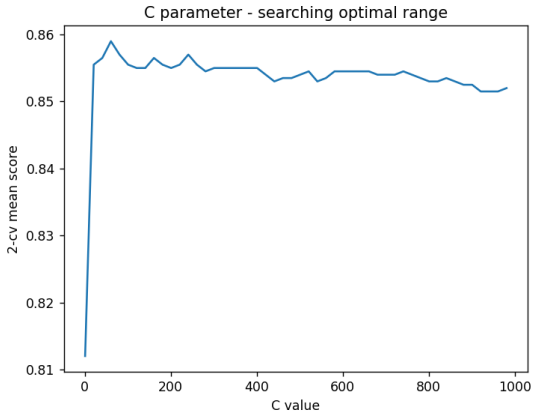


We can see that for $\gamma \in (5,500)$ the score is relatively high (0.83-0.86), and towards high values of gamma there is decrease so we can conclude optimal gamma value is probably within this range. so, we will choose this range

Now, with 2-cross validation let's find an appropriate range for the C parameter:

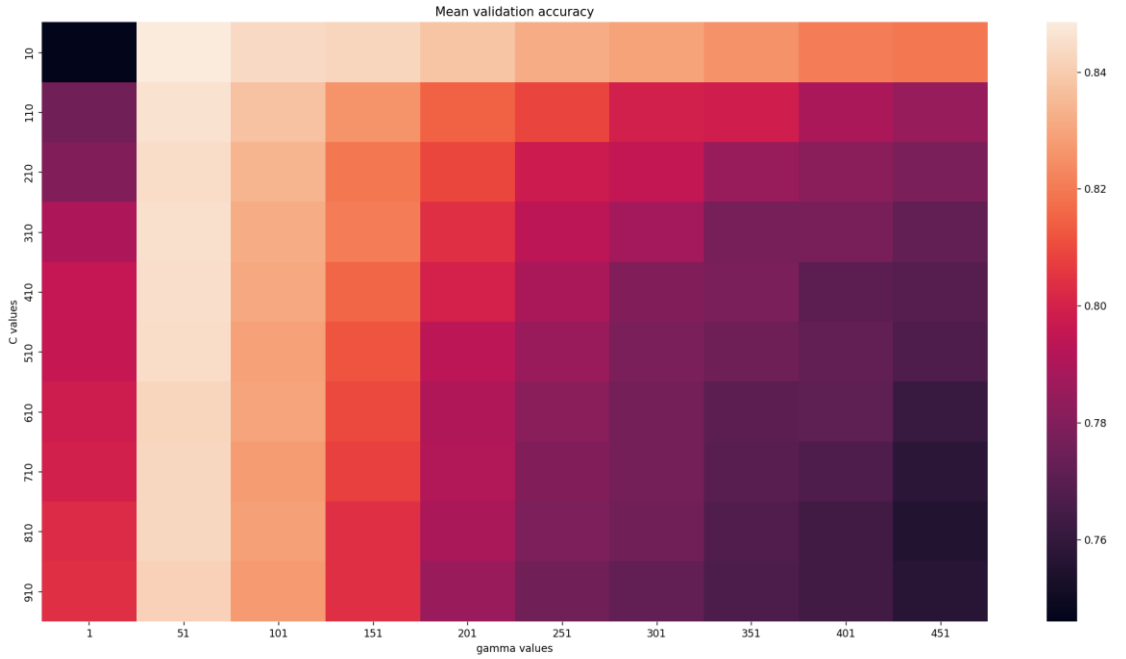
On the left – broad range of C, on the right - narrower range respectively.



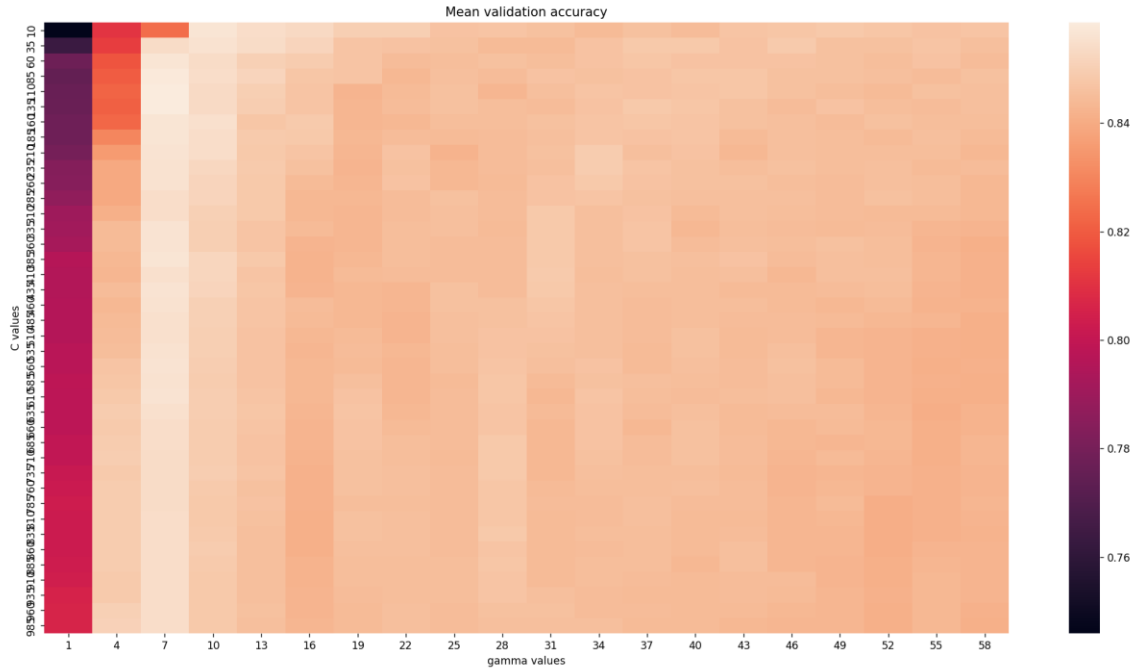


We can see that this range is indeed consistent with relative high score (0.85-0.86), and towards high values of C there is decrease so we can conclude optimal C value is within this range. so, we will choose this range.

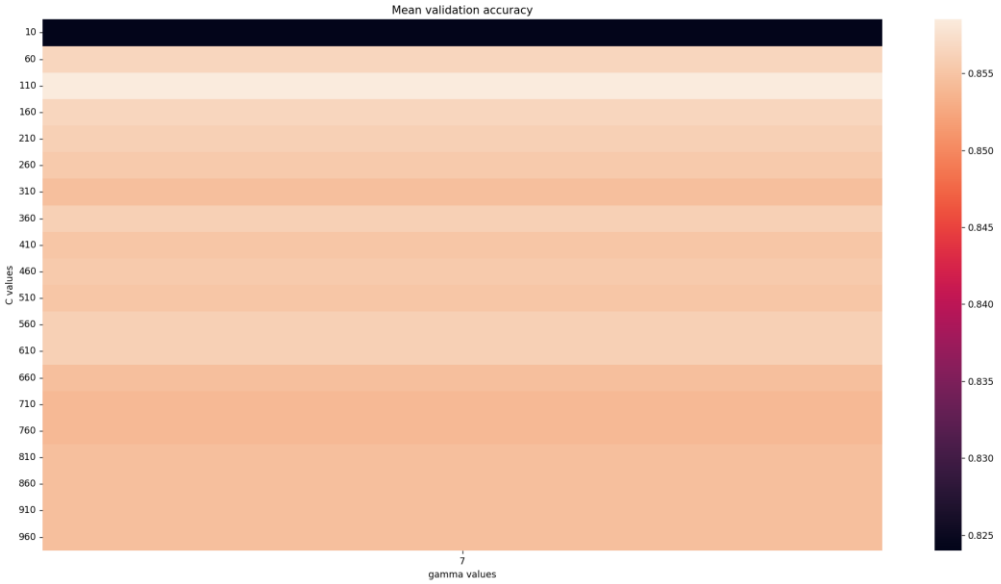
For running-time considerations, we first performed grid search with 2-fold to find the optimal combined sub-ranges for the parameters. First, we performed a broad range grid search with the above ranges and low resolution:



from the above we concluded we can narrow down the range of gamma significantly, to $\gamma \in (1, 60)$, while rising the resolution of the search:

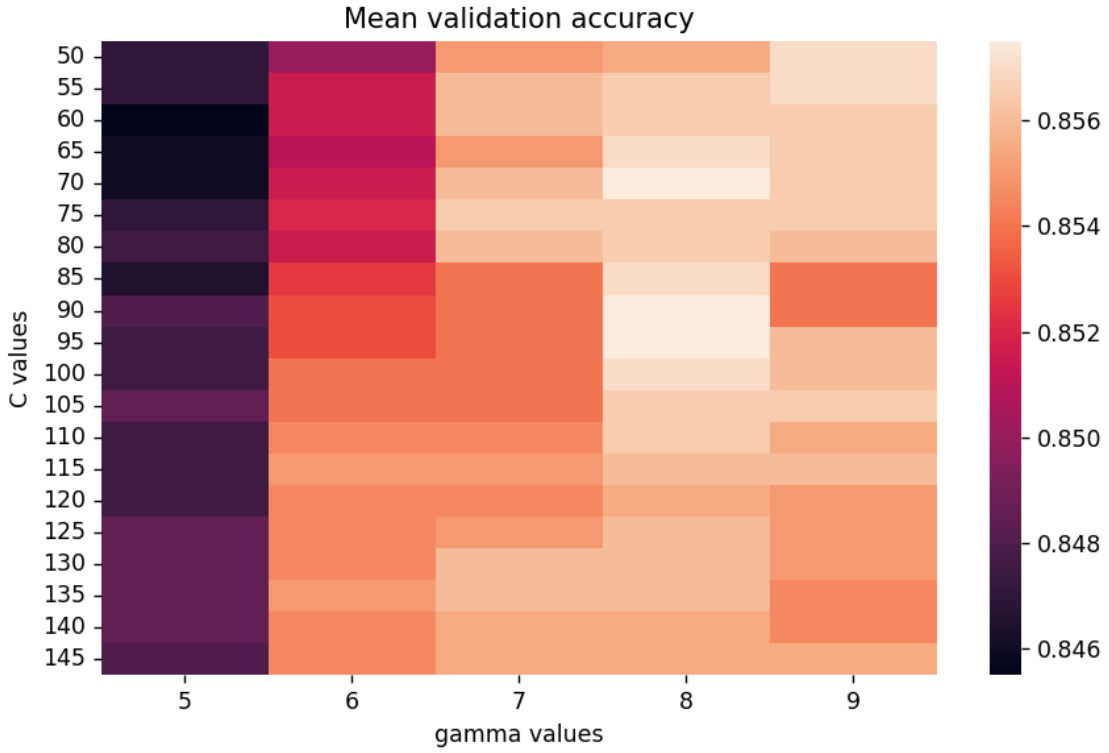


We discovered that the best gamma values for all $C\ values \in (10,10^3)$ is very similar, close to $\gamma = 7$. Now we will fix gamma in search a sub range of C -



We narrow down to $C \in (50,150)$

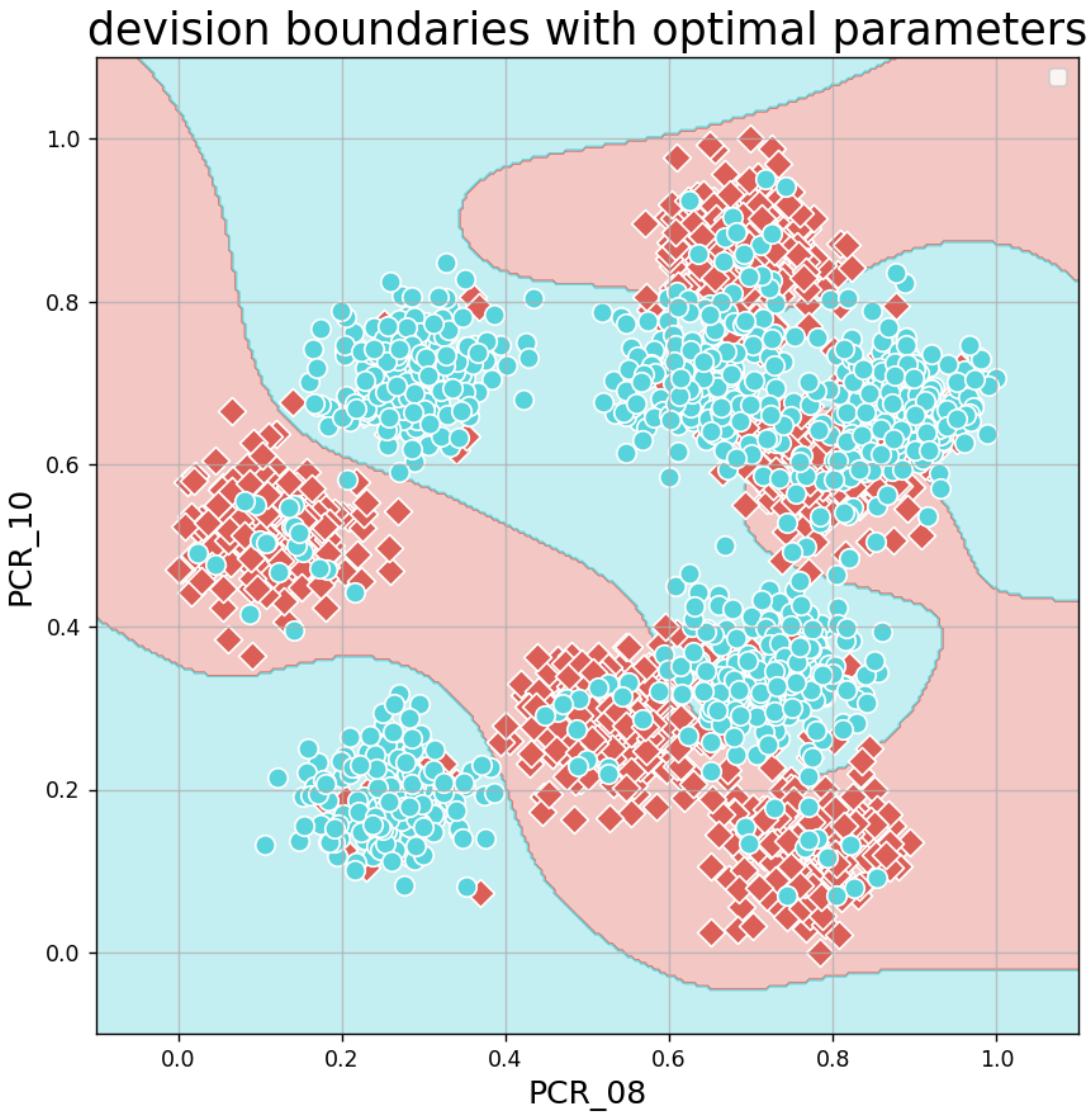
Now we performed a grid search with 8- fold:



Finally, we found that the optimal hyperparameters are:

$$\gamma = 8\,, C = 70$$

A23.



The test accuracy of this model is 0.862.

this model is better in predicting the 'risk' label than the KNN model w.r.t to the test accuracy since the KNN model's is 0.852.