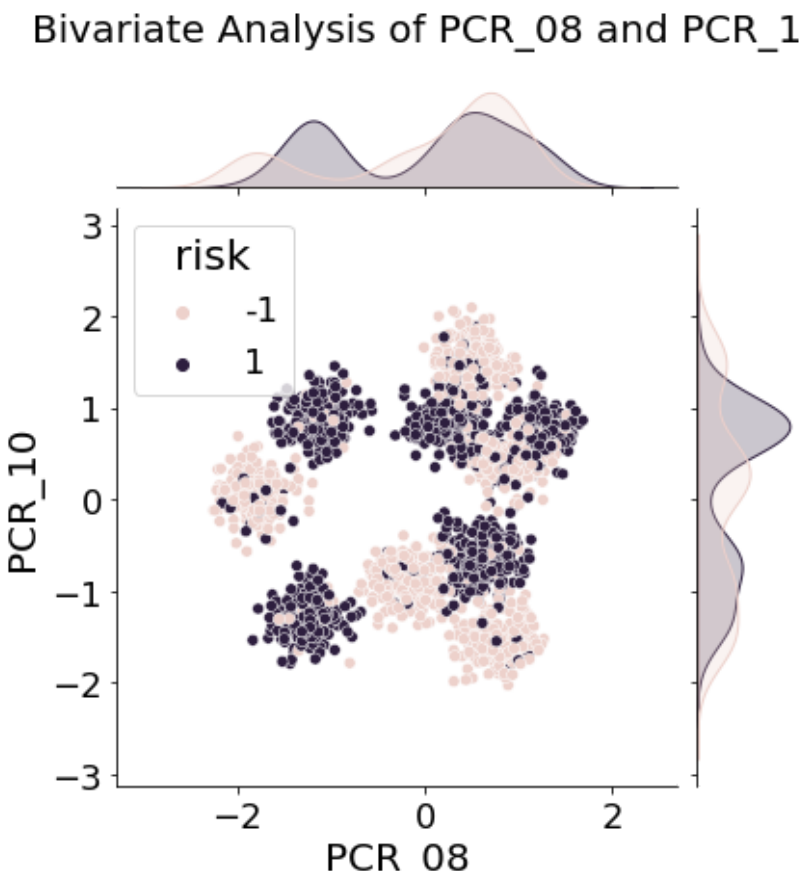
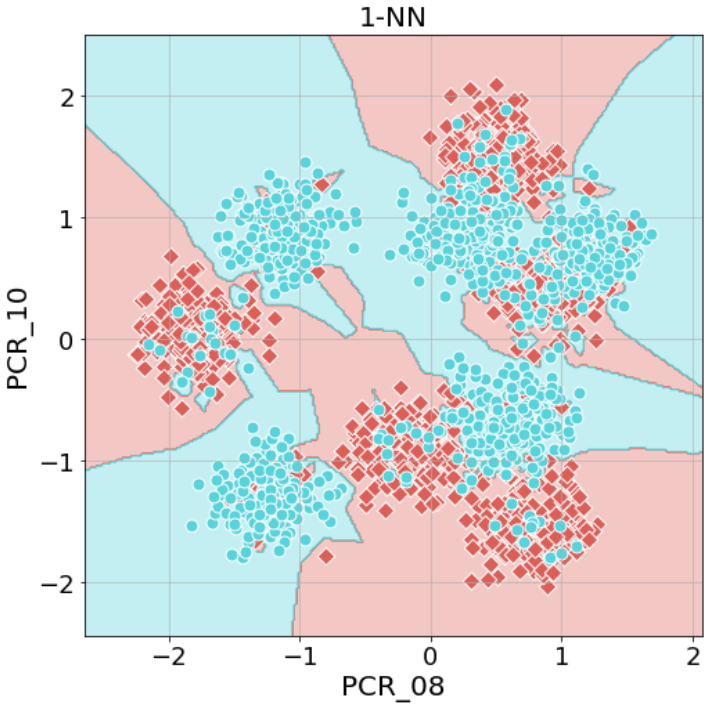
Part 1: Basic model selection with k-Nearest Neighbors

Visualization and basic analysis

Q1)

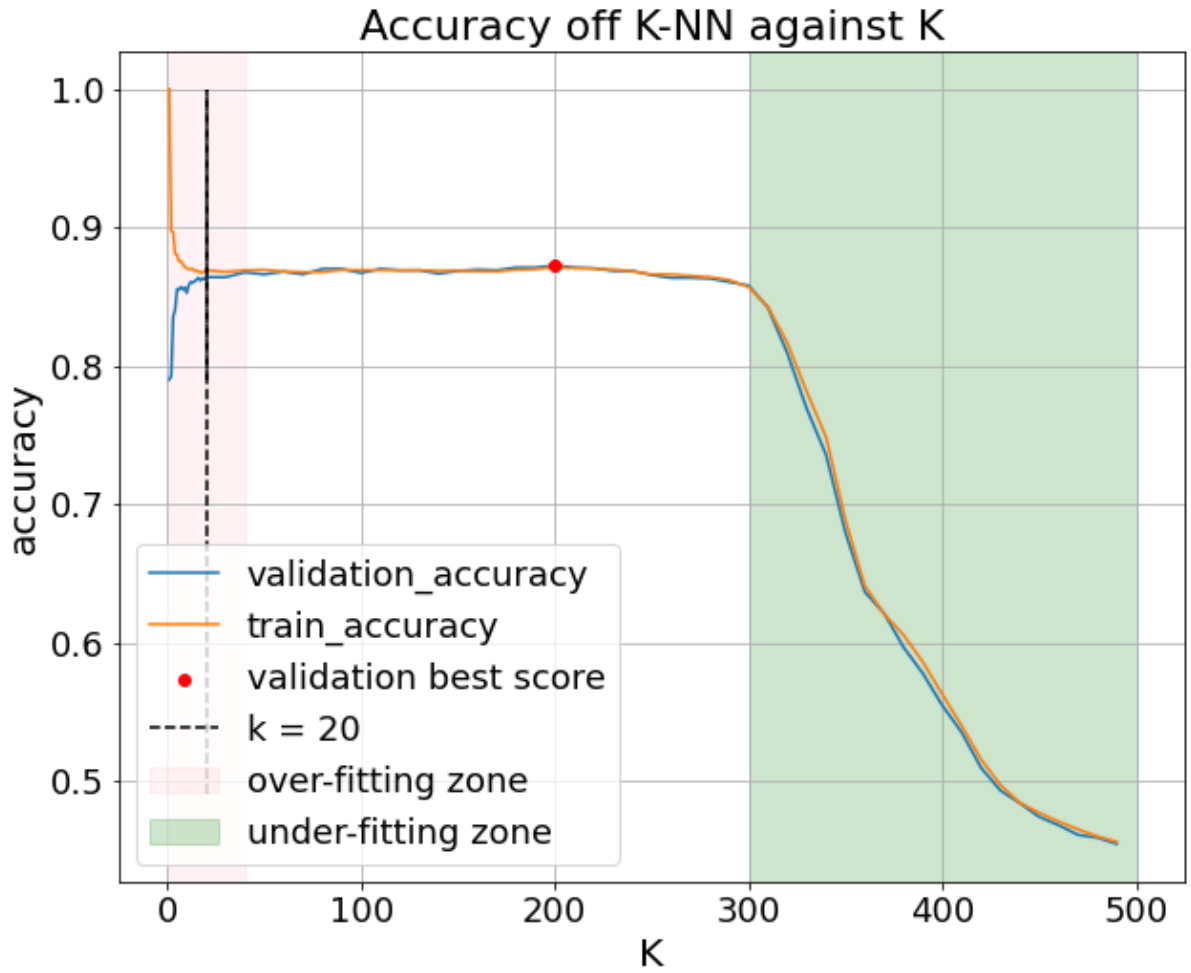


Q2)



Model selection

Q3)



The best k value is 200 which yields a train accuracy of 0.8711 and a validation accuracy of 0.8725

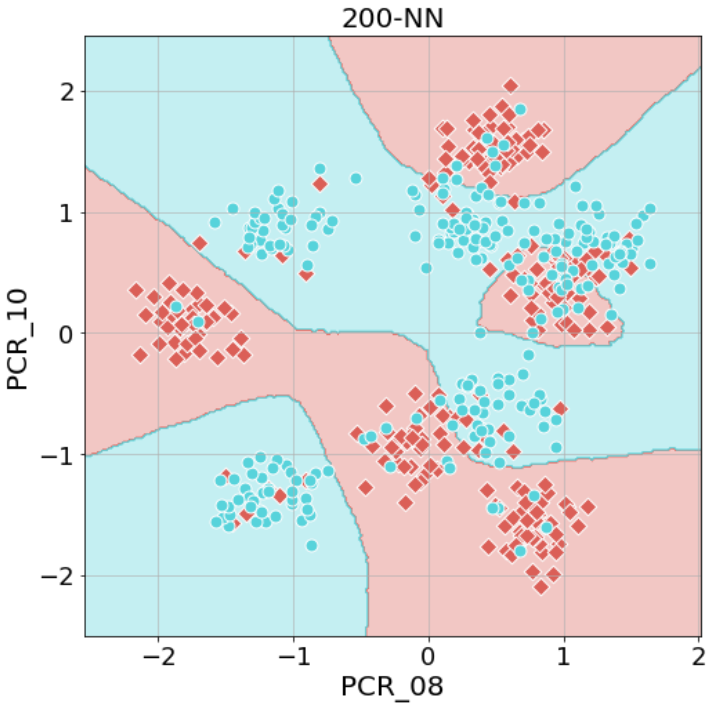
We can notice that these two values are very close to each other, hinting that the generalization capabilities of the model with this parameter are very good.

As we can see from the plot, our model is overfitting when k is in the range between 1 and 40, and it is underfitting when k is greater than 300 (the performance of the classifier starts to drop after k=200, but until k=300 the performance are not too poor, after this value we can really assist to a heavy under-fitting). It should be remarked that these distinctions are not so sharp and the real boundaries are more gradual.

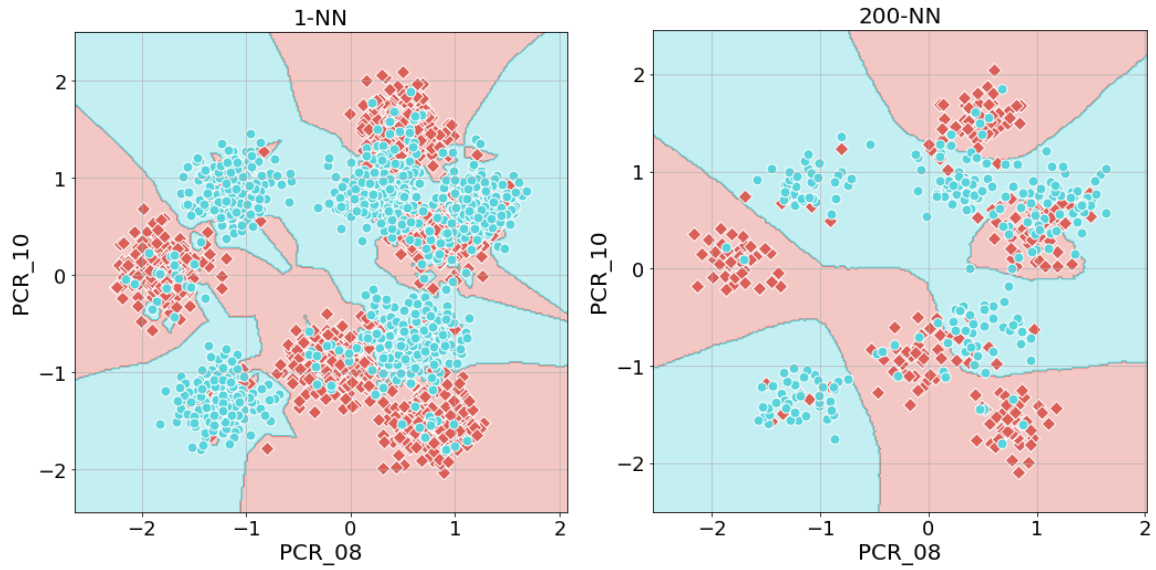
In detail, in the over-fitting zone we can see that the model is performing really well on the training data, but it's performing poorly in the validation data. In fact, with low values of k, the decision boundaries are very indented and they are highly sensitive to the training data, thus having a low bias, but a high variance.

Whereas, in the under-fitting zone we can see that the model is performing poorlier both in the training data and in the validation data. In fact, with high values of k, the decision boundaries are over-smooth and they are too little sensitive to the training data, thus having a high bias, but a low variance.

Q4)
The accuracy obtained with the best hyper-parameter (k=200) is 0.826

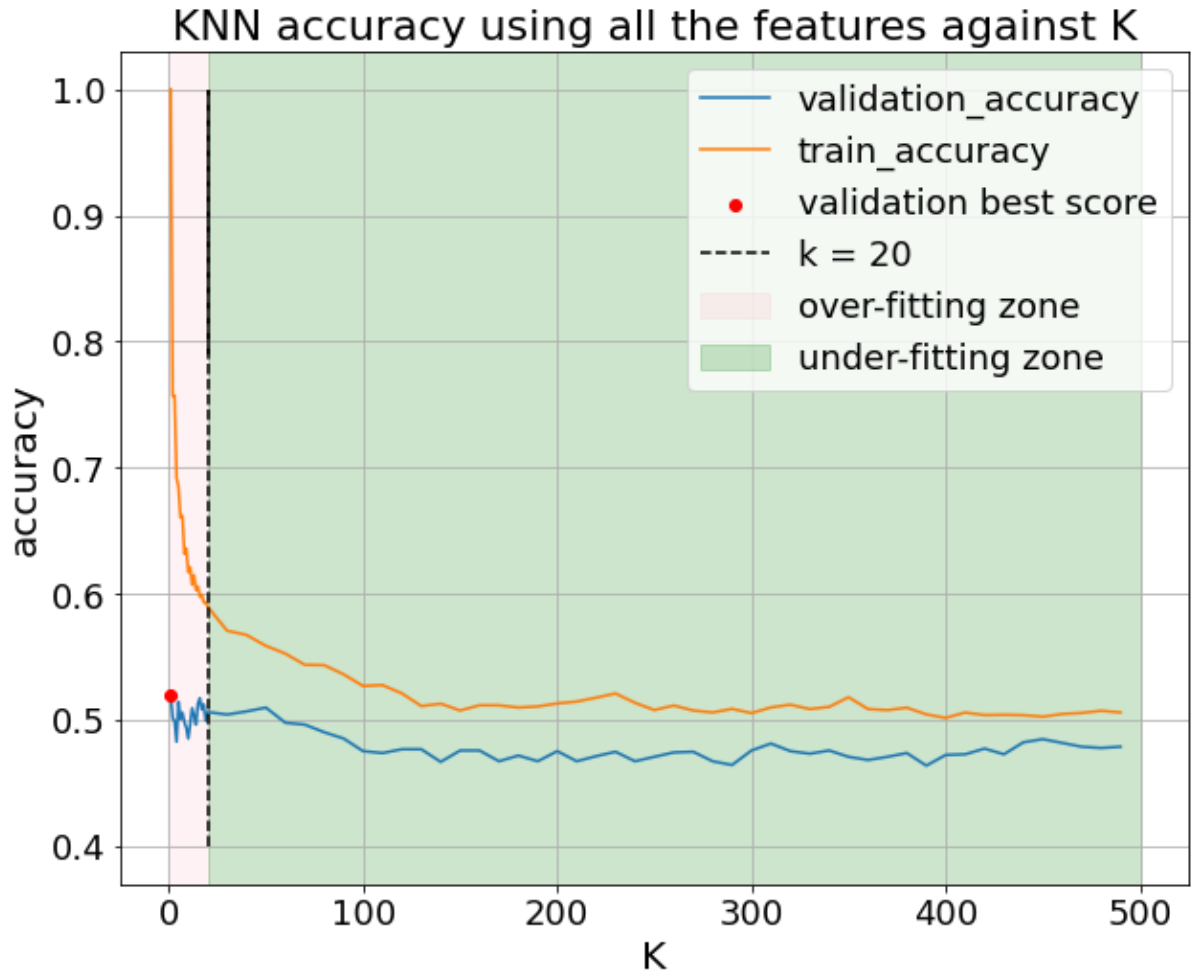


Q5)



By comparing the two plots (when k=1 and when k=200), we can confirm our previous claim on the influence of k over the decision boundaries. Indeed, with k=1 we can see that the boundaries are indented and are highly influenced on the training data, gaining a low bias and a high variance.
Differently, when k=200 we can see that the decision boundaries are smooth (but not over-smooth) and well defined, managing to capture the localization of the clusters of data, despite the presence of noise. In this way, it is possible to say that with this model we have reached a fair tradeoff between the bias and the variance, as it is confirmed on the accuracy of the validation data which is very close to the accuracy obtained in the training data.

Q6)



In this case the best k is 1 which gives a training accuracy of 1.0 and a validation accuracy of 0.5195

We can affirm that when k is in the range between 1 and 20, the model is overfitting as the difference between the train accuracy and the validation accuracy is very big, hinting that the bias is low and the variance is high. Whereas with bigger values of k, the model seems to under-fit as both the training accuracy and the validation accuracy are low, giving a high bias and a low variance. All in all, the model does not seem to provide substantial benefits to a random guess, as the validation accuracy tends to stay around 0.5 and with certain values of k its validation accuracy drops even further.

By comparing this latter plot with the plot which showed the results obtained using only PCR_08 and PCR_10, we can see that in the former there was a rather big range of values of k which reached a “sweet spot” between the training accuracy and the validation accuracy. Whereas in the latter, there does not seem to exist a value of k which yields desirable results, as it either over-fits (with low values of k) or it under-fits (with high values of k).

This phenomenon can be explained by the high sensitivity of KNearestNeighbor classifiers to the selected features. In fact, by introducing non-relevant features, the classifier does not manage to endorse the most-relevant features, but rather it tries to give the same weight to all of them.

Furthermore, given that we have introduced a lot more features, the model needs many more samples in order to learn the latent decision function. This phenomenon is also known as the “curse of dimensionality”, where in high dimensional spaces the samples needed to learn grow very fast and the data-points are very sparse, which is a further factor which undermines the performance of KNearestNeighbors classifier.

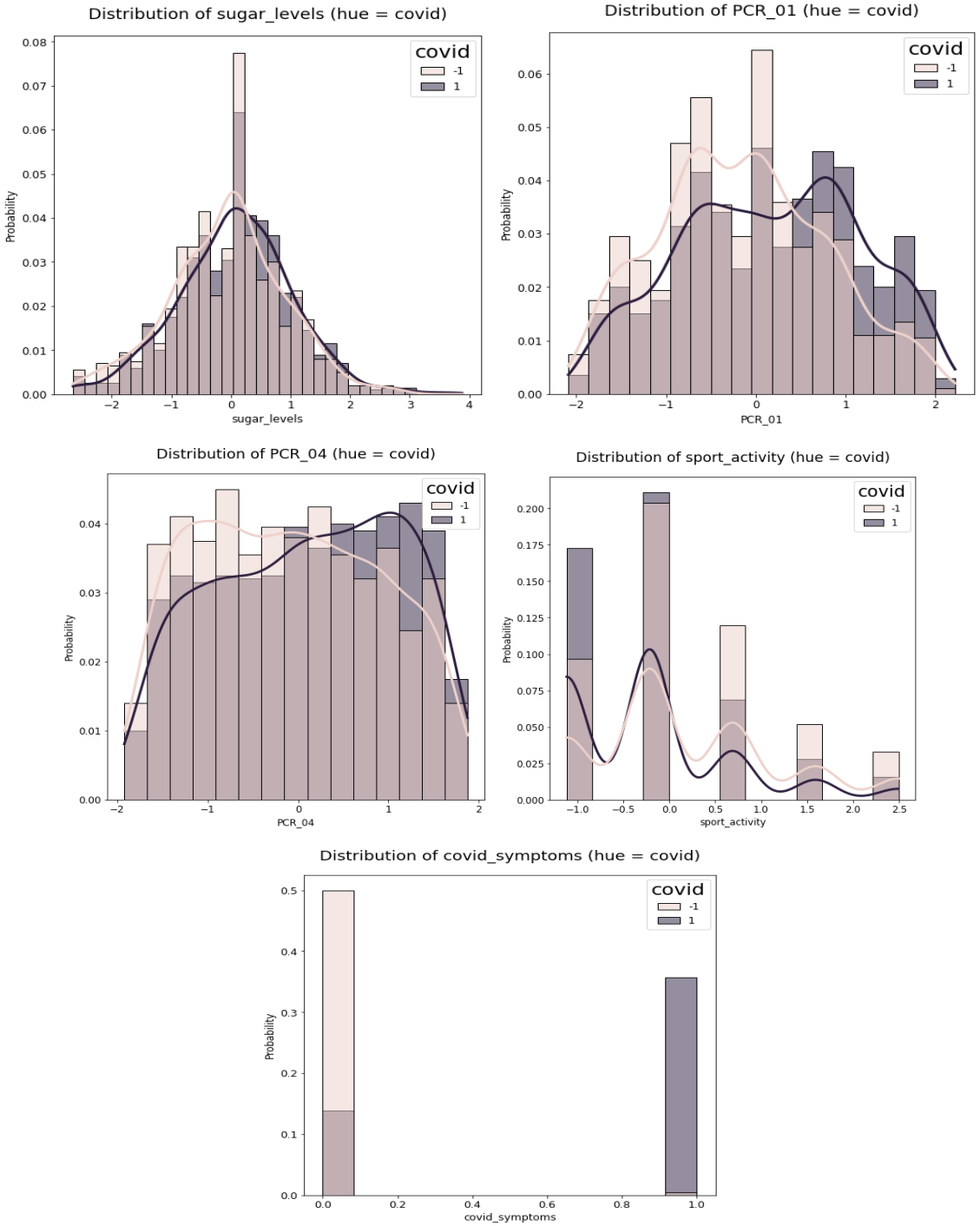
Part 2: Decision trees

Visualization and basic analysis

Q7)

In the first assignment the 5 most useful features to predict the covid label that we have identified were: suger_levels, PCR_01, PCR_04, sport_activity and covid_symptoms

Reminder: covid_symptoms is a binary feature that is one if the patient has at least one of the following symptoms: sore_throat, cough and shortness_of_breath.



By looking at the plots, we can see that for each feature we can localize a threshold that well separates positive samples from negative samples. For instance almost every patient that has 1 to the covid_symptoms feature is labeled as covid.

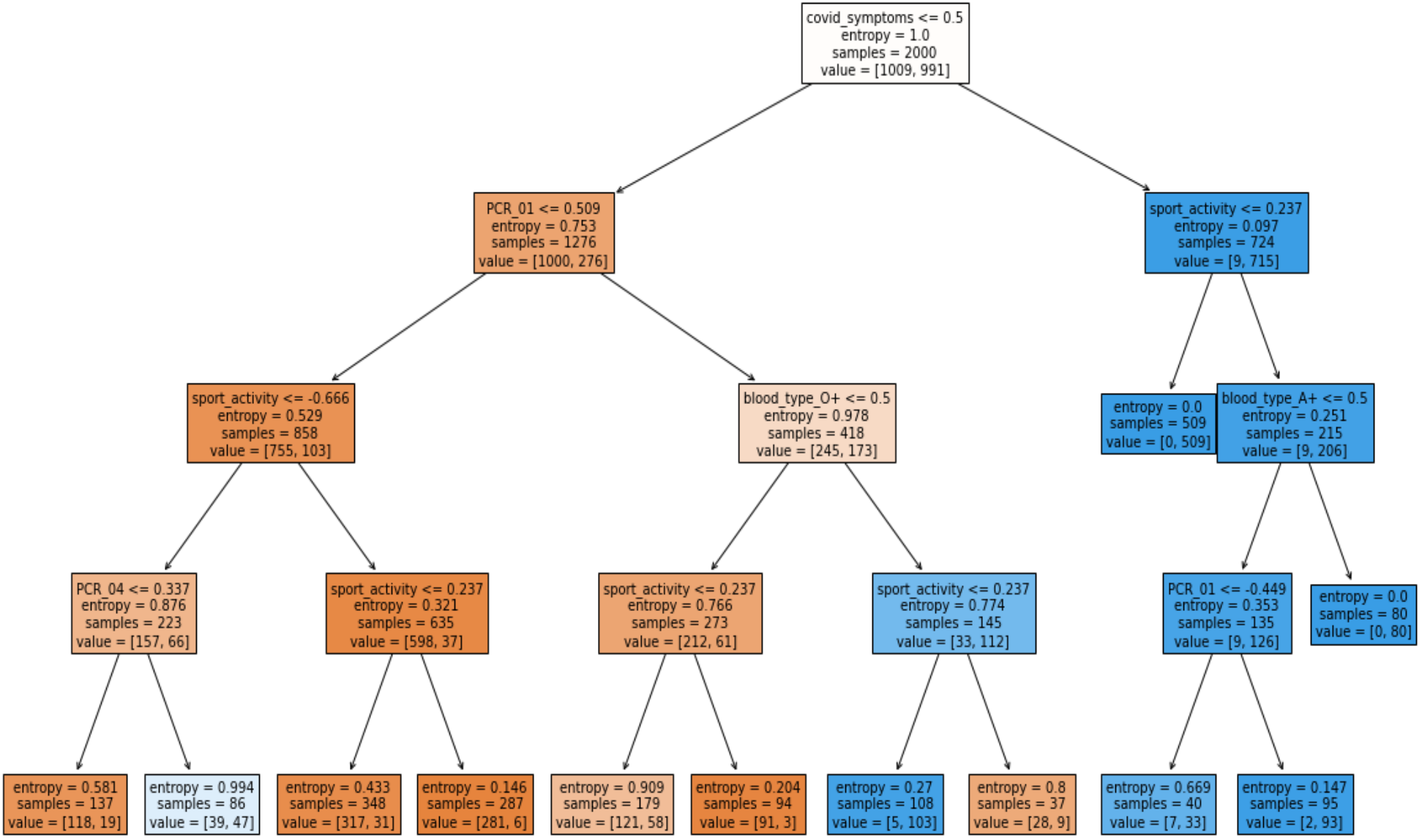
Moreover for patients with sport_activity > 0 there are bigger chances to be labeled as false to covid.

Q8)

	covid
covid_symptoms	0.741340
sport_activity	0.209082
PCR_01	0.169038
PCR_04	0.101333
sugar_levels	0.075169
blood_type_A+	0.068121
blood_type_O+	0.066160
blood_type_A-	0.055900
fever	0.035015
household_income	0.022060

Q9) After training our decision tree with our entire dataset we get a training score of 0.9105 which is very significantly high.
Here we can see the plot of our tree.

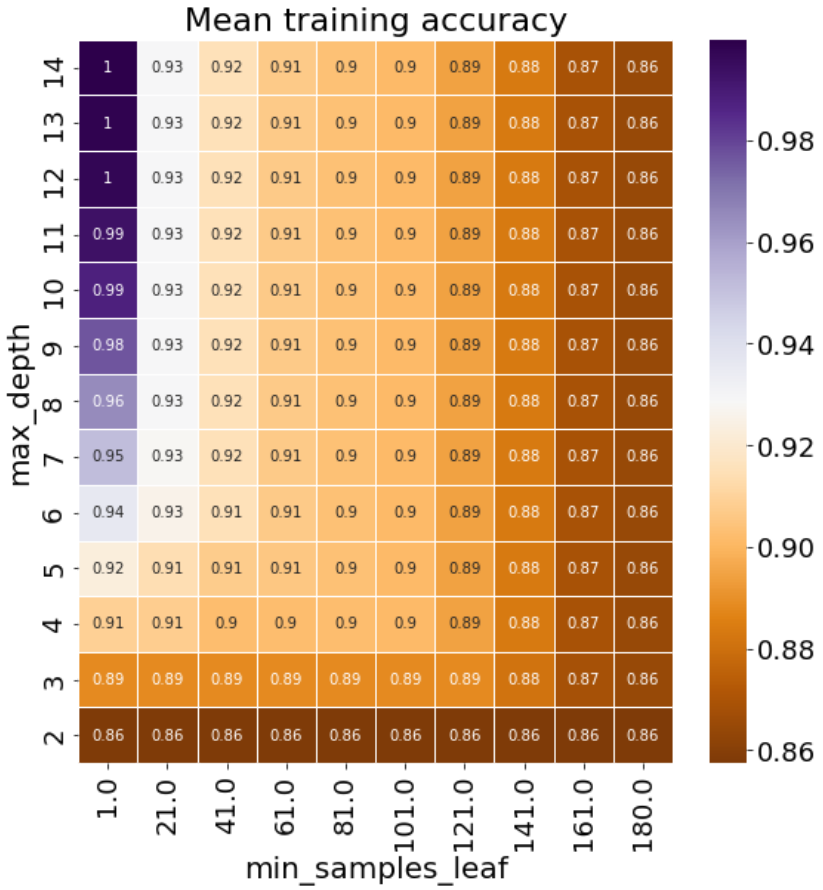
Decision tree (ID3) with maximal depth = 4

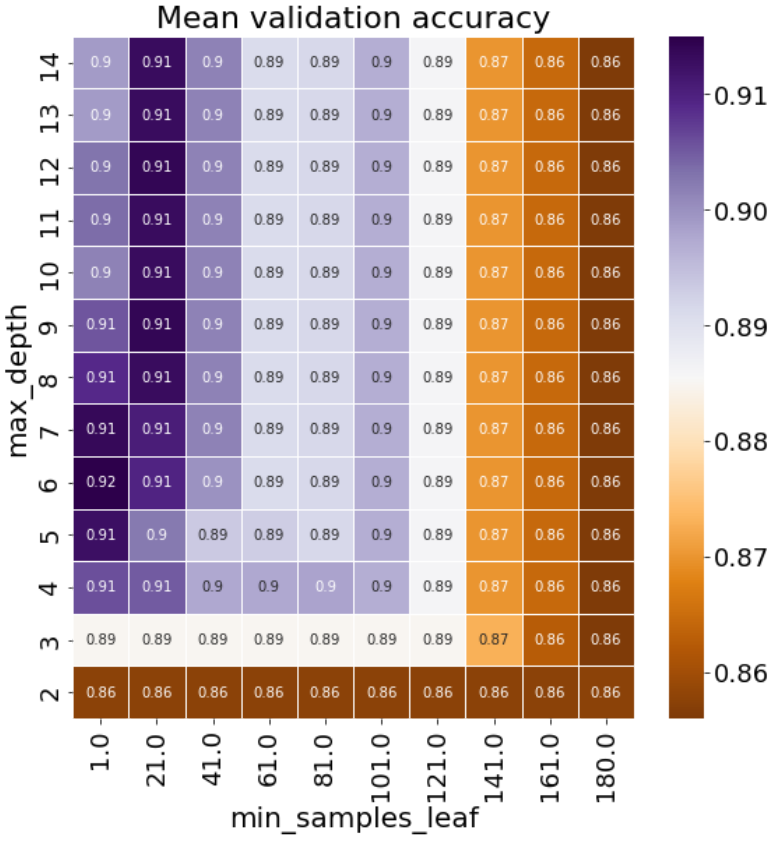


Q10)
The named features in Q7 are the most correlated with covid's label.
As we can see, the features from Q7 are all used as splitting conditions in our decision tree. Moreover, we can see that blood_type_O+ and blood_type_A+ are also used in the decision tree, which is not very surprising because these two features are also in the top 10 most correlated features. Indeed, ID3 at each split tries to maximize the Information Gain, which measures the amount of entropy, a measure of uncertainty in classification, that is lost as a consequence of the split. Therefore, when a feature has a high correlation with the label, it is also likely that it will lower the entropy in a node.
The first interesting pattern that we can appreciate is that every patient that has a value of one in correspondence of covid symptoms and that it also has a sport activity that is under 0.237 is labeled as positive to covid. Indeed, there is a pure node at only depth 2 that supports the above claim.
Furthermore, from this plot we can see that by using as the root node "covid_symptoms" a clear pattern arises. Indeed, in case when covid_symptoms is 1, all of the following branches are heavily skewed towards classifying the samples as True (to Covid), whereas when covid_symptoms is 0, all of the following branches are heavily skewed towards classifying the samples as False. This pattern gives a confirmation that this feature has a high correlation with covid, as it is confirmed by the Pearson's correlation index which amounts to 0.741340. This observation is once more confirmed by the plot shown in Q7, where covid seems to be almost separable.
It should be noticed that the tree might also have over-fitted the training data. Indeed we can see a leaf with only 80 samples and zero entropy, corresponding to: covid_symptoms=1, sport_activity > 0.237 and blood_type=A+. With such a low entropy and so little number of samples it is highly likely that this may signal the presence of over-fitting.

Model selection

Q11)c)





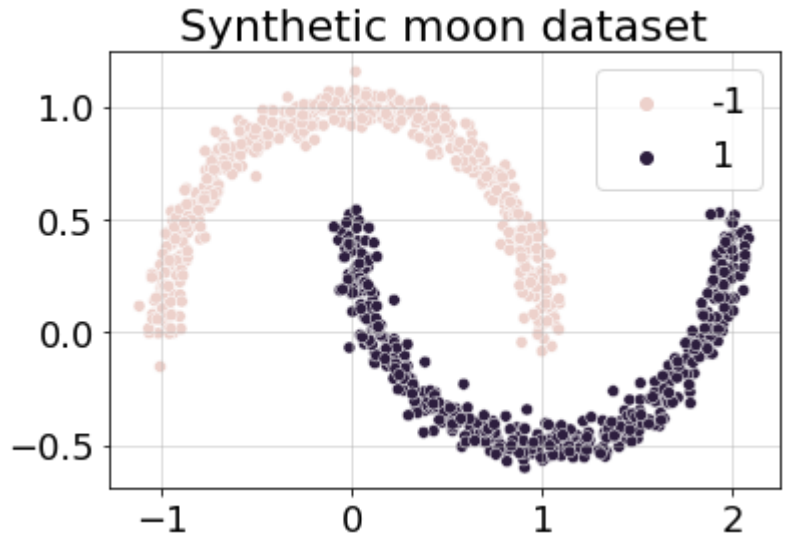
In our tests the best combination is obtained with a maximum depth of 6 and a minimum number of samples per leaf equal to 1, delivering a mean validation accuracy of 91,5%.
d)
Our model underfits whenever the number of minimum samples per leaf is greater than 141, regardless of the maximum depth.
e)
Our model overfits whenever the number of minimum samples per leaf is less than or equal to 21 and at the same time the maximum depth is greater than or equal to 8.
f)
The model underfits for the said values since it is not allowed to split some nodes in which the entropy, and consequently the impurity of the nodes, is still high. For this reason the nodes which require an additional split or even more than one will be subjected to an acute uncertainty that will lead to a poor performance. Clearly, in this way the model will be subjected to a smaller variance and a higher bias, as all of the underfitting models do.
On the other hand, the model overfits in the above range given that it will recognize the noise present in the data as part of a significant pattern, striving to minimize as much as possible any kind of node, not conceiving the possibility that the impurity could be given by outliers. With this configuration, the model will minimize the bias at the cost of variance, as all of the overfitting models do.

Q12)
The decision tree with the best parameters that we have found delivers a testing accuracy of 93%

Part 3: Linear SVM and the Polynomial kernel

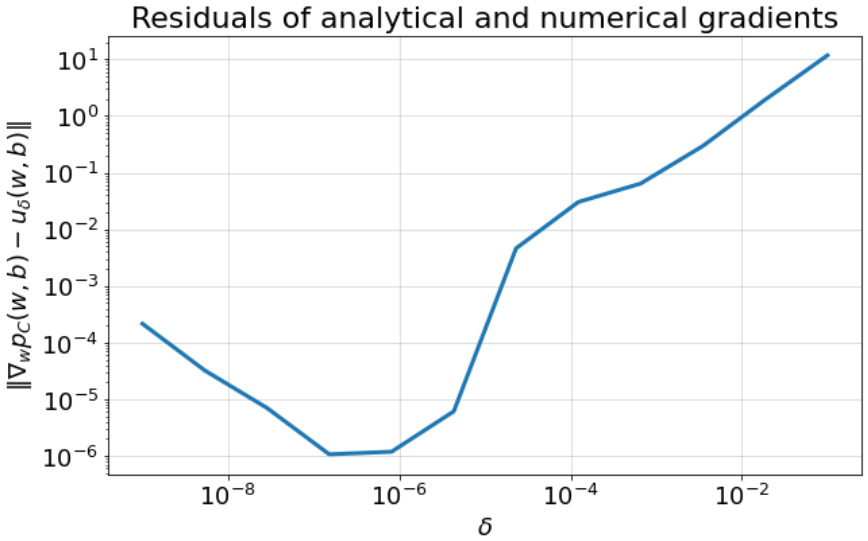
Loading a synthetic dataset for basic experiments

Q13)



Verifying your implementation: Numerical vs. analytical gradients

Q14)



The behavior of this function can be explained in the following way.
With big values of delta the numerical approximation is too coarse, as delta is too big and it does not manage to approximate well the analytical subgradient (the secants that are used in the numerical gradient do not approximate well the tangent that it is used for the analytical derivative, as the second point with which we compute the numerical derivative is too far away to well approximate dx), and therefore the difference is relatively big. This phenomenon becomes less and less prominent as delta shrinks.
Then we reach a certain value of delta where the difference reaches its minimum. At this point the computer has reached the maximum of its precision and this threshold is also known as machine epsilon.

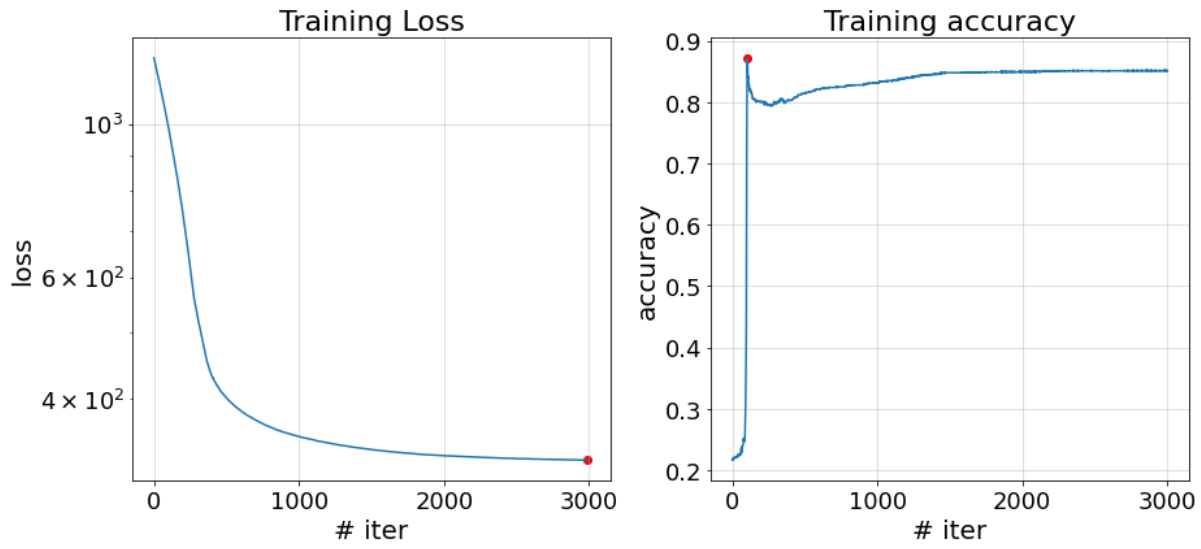
Once past this point, the computer loses precision and the difference tends to diverge as delta shrinks further.

Solving Soft SVM problems using Stochastic Gradient Descent (SGD)

Q15)

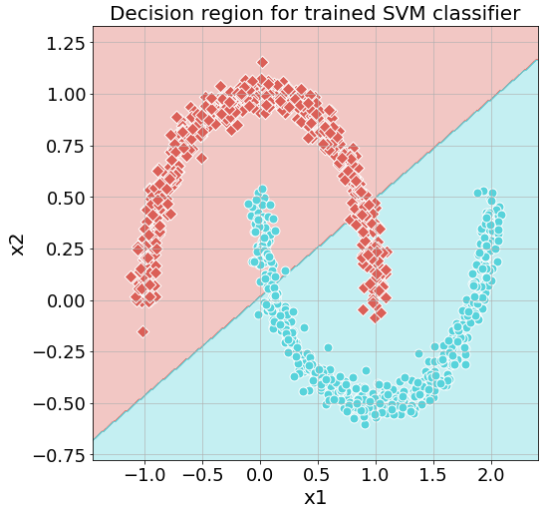
The best loss was obtained on the 2990-th iteration and is equal to: 326.023

The best accuracy was obtained on the 244-th iteration and is equal to: 0.885



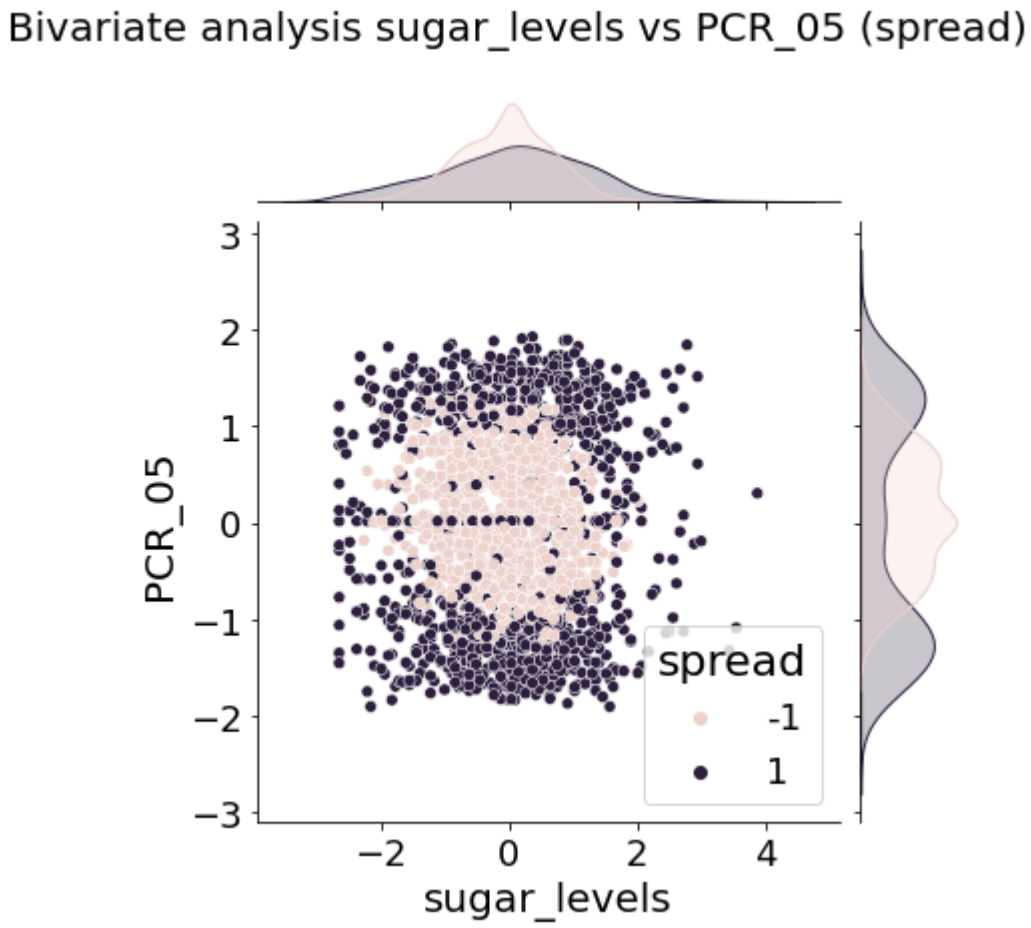
As we can see we do not achieve the best loss and the best accuracy in the same index. This can be explained by the fact that the loss heavily penalizes outliers as the confidence of misclassifications grows, whereas the accuracy is not influenced by this factor. Indeed it is possible to have models with two different losses with the same accuracy. Therefore it is possible that the model that achieved the best accuracy had a big confidence in its misclassification and this could have led to a bigger loss.

Furthermore, even though this phenomenon has less influence, the loss that we used is positive also for right classifications with a small confidence whereas the accuracy is not influenced by this factor. This is yet another trait that sets apart the loss from the accuracy.



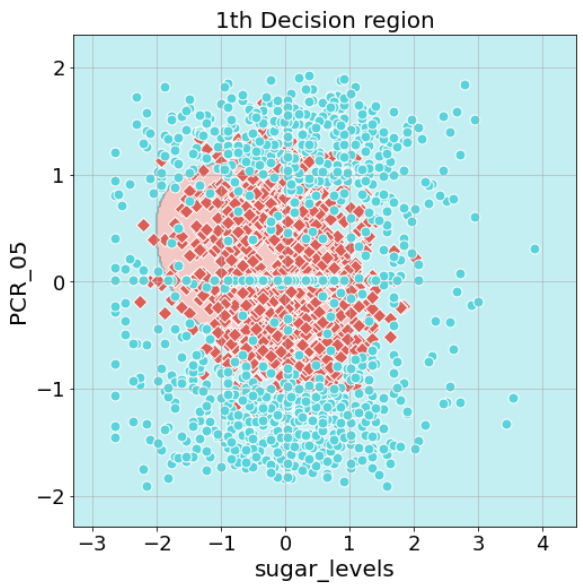
Using a feature mapping

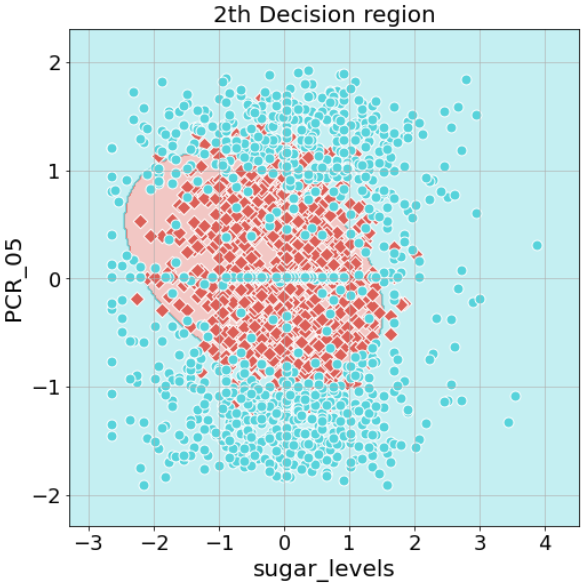
Q16)

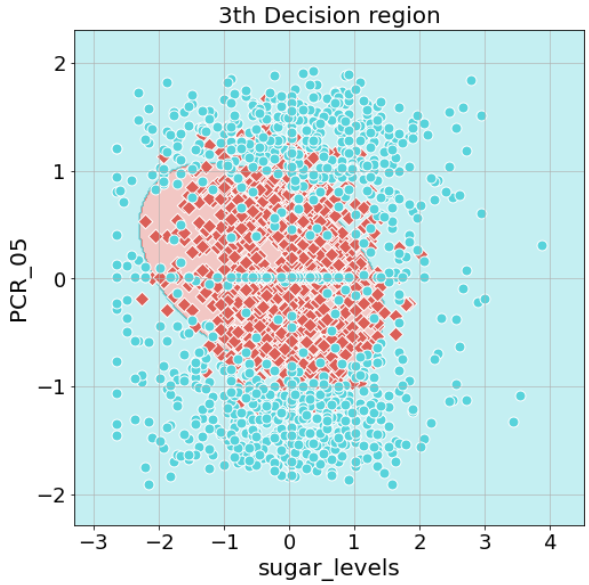


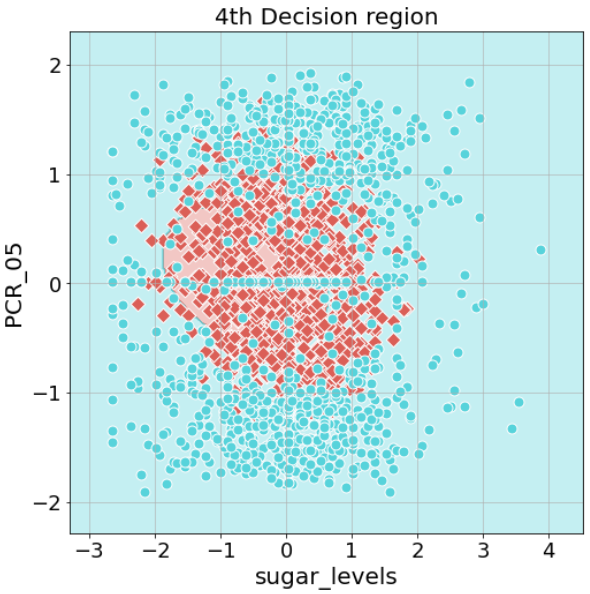
As we have already pointed out in hw1, we have a “circular interaction” between these two features.

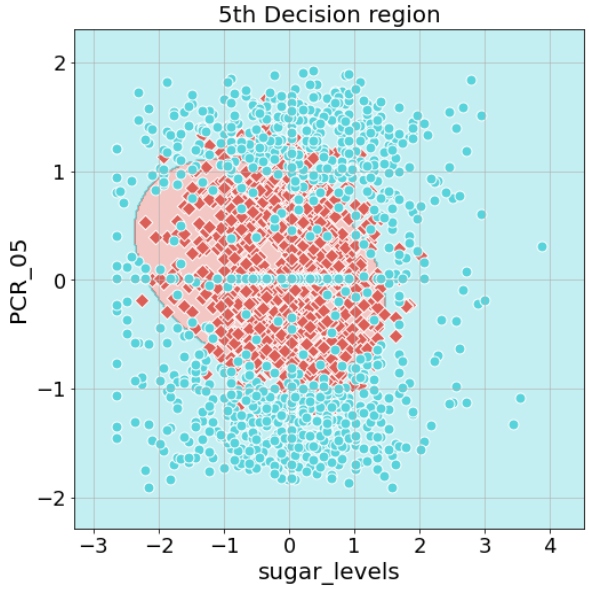
Q17)
for test 1 we got an accuracy of 0.81
For test 2 we got an accuracy of 0.83
For test 3 we got an accuracy of 0.83
For test 4 we got an accuracy of 0.67
For test 5 we got an accuracy of 0.84


1th Decision region
PCR_05
sugar_levels


2th Decision region
PCR_05
sugar_levels

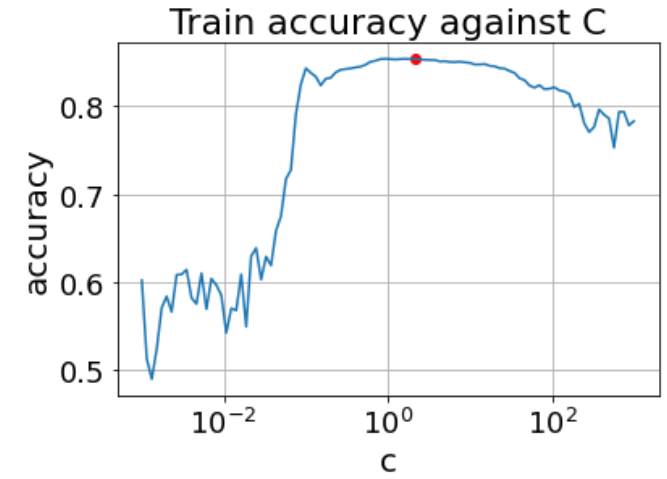

3th Decision region
PCR_05
sugar_levels

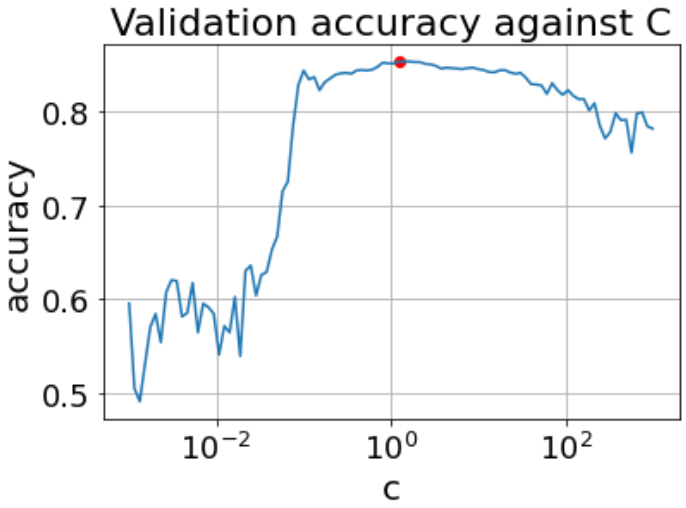

4th Decision region
PCR_05
sugar_levels


5th Decision region
PCR_05
sugar_levels

The average accuracy is 0.69 and the standard deviation is 0.09937
Our variance is rather low. This variance could be derived by the fact that the learning rate is fixed and the weights and the bias are initialized randomly. For this reason, the parameters which are initialized closer to the optimum would reach a better performance.
Moreover there are some additional factors that bring such a variance in our results.
First, despite not being the most influential factor, we have constrained the number of iteration to 10.000 and if we initialize our problem too far from the minimum, potentially we could not give the algorithm enough iterations to converge. Moreover, we are using the algorithm Mini-Batch stochastic gradient descent with a batch size of 32. This factor brings some variance in the optimization, as in this way the descent region is only the expected value of the gradient and this estimate gets noisier and noisier as the size of the batch gets smaller. For this reason, by increasing the batch size we should expect less variance.
Moreover, this algorithm at first goes rather fast towards the minimum, but towards the end it starts to struggle to improve the results, yielding the variance that we see. For all of these reasons we need to expect some differences in the decision regions and in the accuracies.

Q18)
We have explored the performance of SVM with a second-degree polynomial feature mapping using as a range for C: $[10^{-3}, 10^{3}]$ using 100 values sampled in this range.





As we can see, for low values of C the accuracy (of both the training and the validation set) is small and it is just a bit better than a random guess. This can be explained by the fact that the objective function puts a lot of weight towards minimizing the regularization term, trying to minimize the variance at the expense of increasing the bias (we know that when C goes to zero, the optimal weights go also towards zero, thus minimizing the variance, as this optimal value is the same for all possible data-sets, and maximizing the bias). For very low values of C this tradeoff is too much unbalanced towards minimizing the variance, and therefore we can see that the model underfits. This phenomenon is promoted also by the low complexity of the model (unlike more complex models like RBF, which generally tend to overfit more easily).

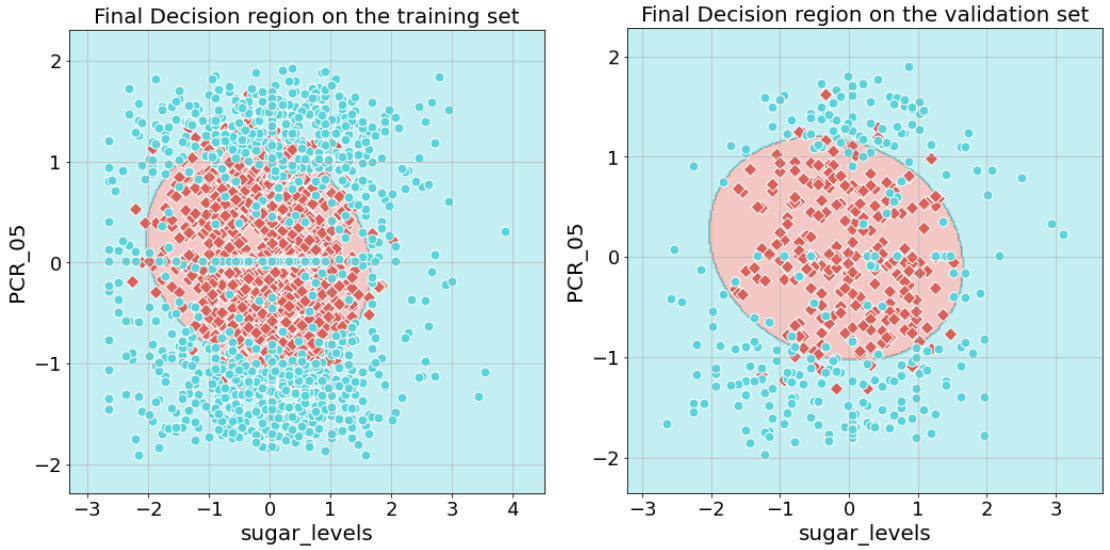
After a certain threshold, this trade-off between the bias and the variance gets more balanced and the accuracy increases sharply. This signals that finally the model is putting the right focus on lowering the bias (at a little expense of the variance). This behavior is also seen in the validation accuracy, showing that the model has a good generalization capability. After reaching the optimal C, we can observe that our accuracies are very stable and staging around 0.85 on both sets, which mean that for C between 1 and 70 we've reach a smooth spot.

At the end we can see that both the validation and the training accuracy are going down. We suspect that this behavior might be derived by the fact that as C goes to infinity, our problem looks more and more similar to Hard-SVM. Given that we have a certain amount of noise, and given that our model class is not too expressive (the VC dimension here is

relatively low), the model has a hard time separating data and therefore we expect the model to perform worse both when it tries to generalize (as it can be seen in the validation plot) and both when it is tested with the data that it has already encountered (as it can be seen in the training plot).

In particular, we have found that the best C for the training set is 1.072 with a corresponding accuracy of 0.85405. Whereas in the case of the validation set, the best C found is 1.630 with an accuracy of 0.8540.

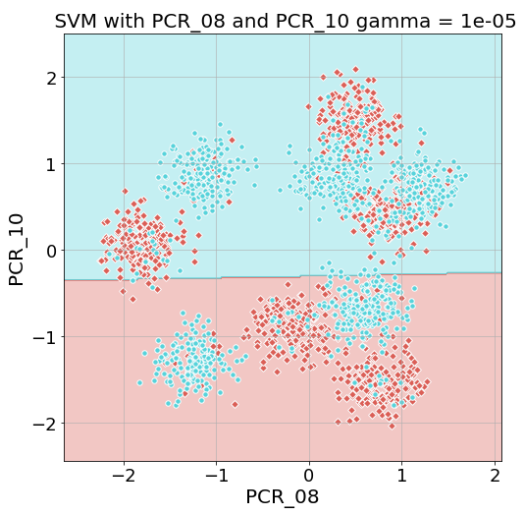
Q19)

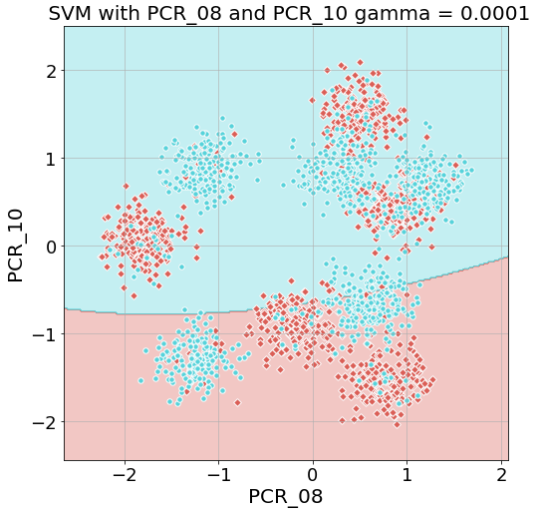


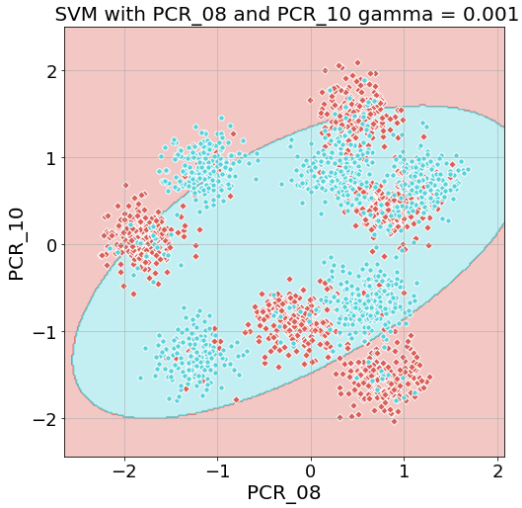
Test accuracy: 0.85

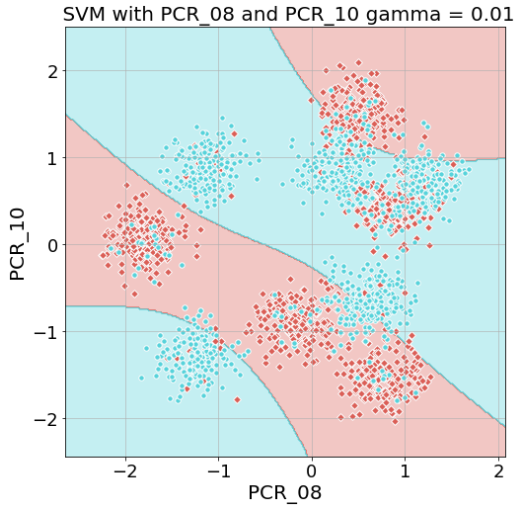
Part 4: The RBF kernel

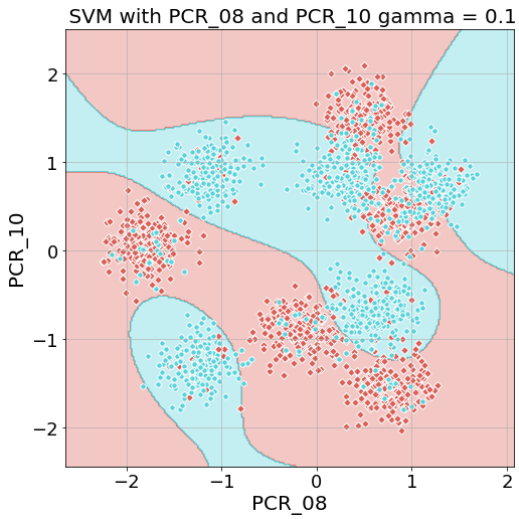
Q20)

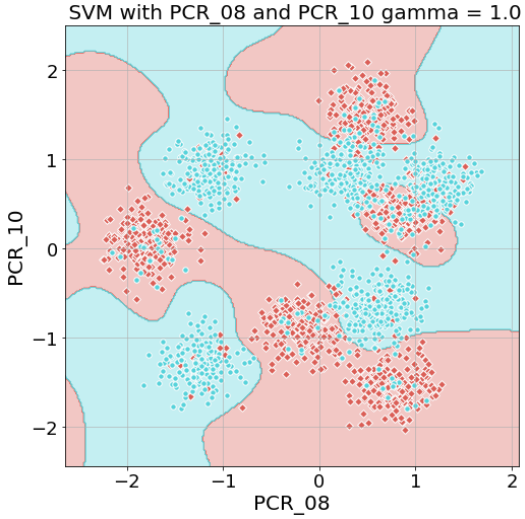


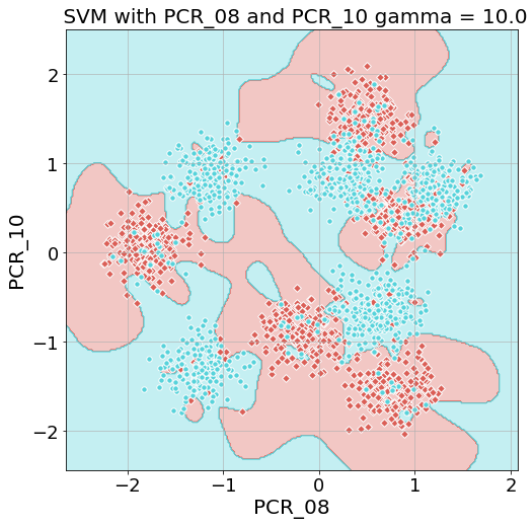


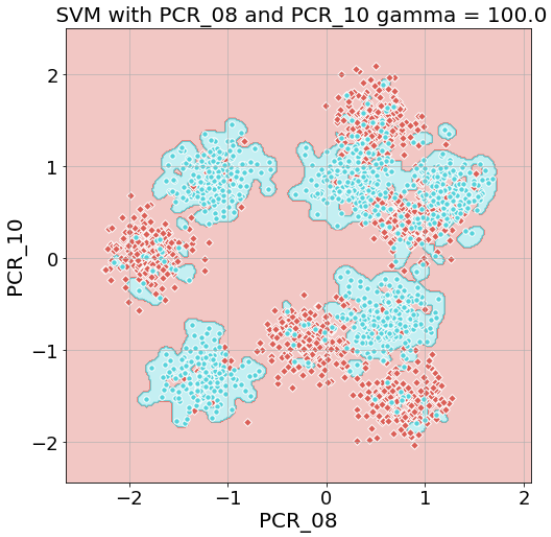


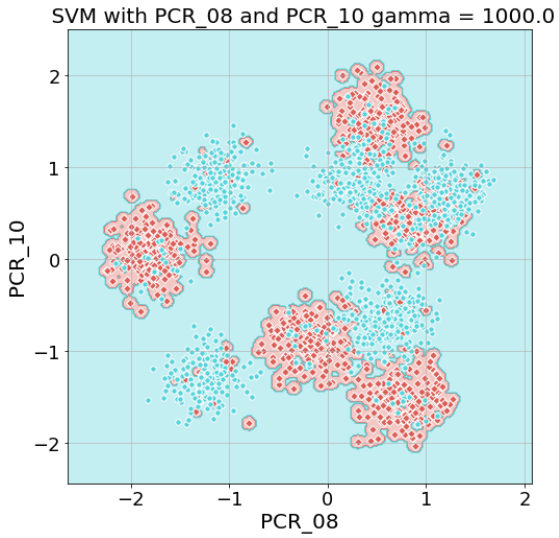


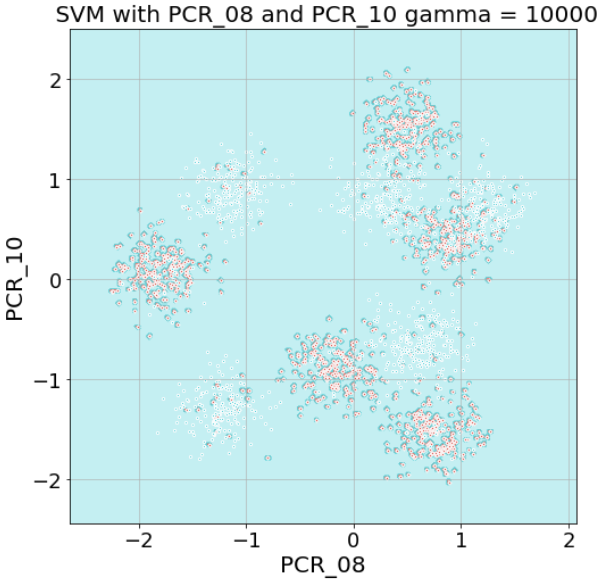




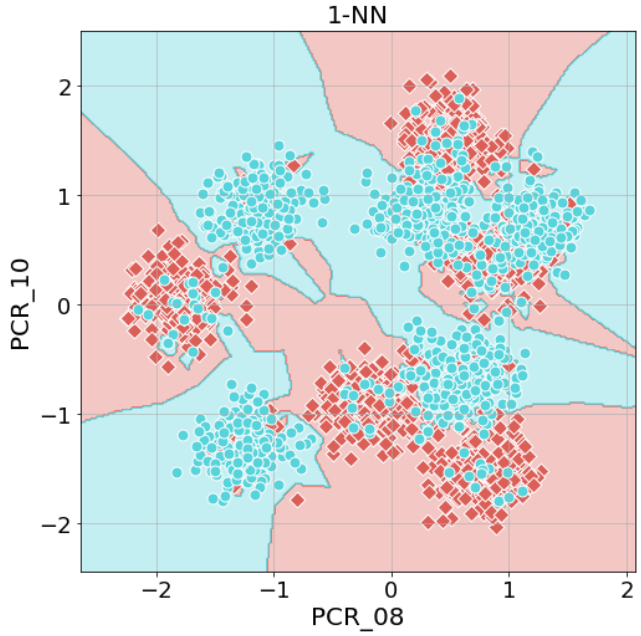

SVM with PCR_08 and PCR_10 gamma = 10.0
PCR_10
PCR_08
2
1
0
−1
−2
−2
−1
0
1
2


SVM with PCR_08 and PCR_10 gamma = 100.0
PCR_10
PCR_08
2
1
0
−1
−2
−2
−1
0
1
2


SVM with PCR_08 and PCR_10 gamma = 1000.0
PCR_10
PCR_08
2
1
0
−1
−2
−2
−1
0
1
2

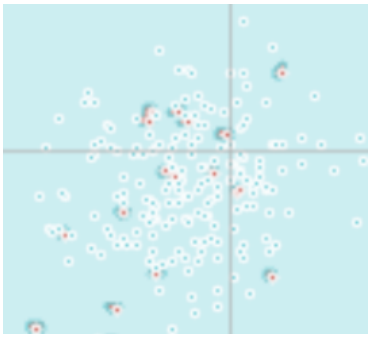

SVM with PCR_08 and PCR_10 gamma = 10000
PCR_10
PCR_08
2
1
0
−1
−2
−2
−1
0
1
2

Q21) As we remember the 1-NN model from Q2 is like below.



The two model are similar in the sense that the model overfits the data, as we can see for each cluster even the noisy data points are classified correctly.
For instance in the 1-NN we can observe the following overfitted points:

And so for the SVM classifier with RBF kernel (γ = 10000)



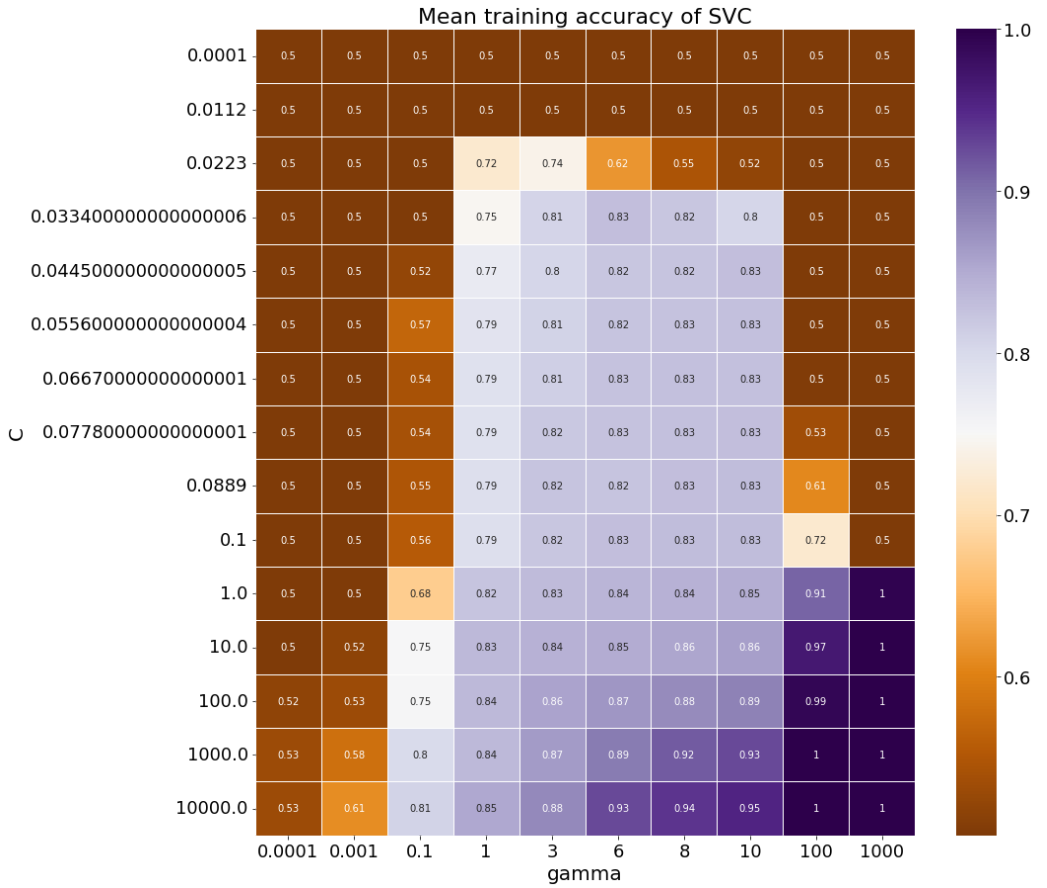
Both of these models tend to overfit the data, yielding a small bias and a high variance. Nonetheless, it is possible to spot some differences. Indeed, RBF behaves similarly to a weighted KNN where the weight is controlled by gamma. More explicitly, as gamma grows to infinity, RBF tends to become more and more similar to 1-NN, whereas as gamma goes to zero RBF tends to become more and more similar to a K-NN classifier whose K is very big. In this case, given that the gamma is big, but not too much, it is somehow similar to a K-NN classifier whose K is rather small, but bigger than 1. For this reason, we can treat the RBF classifier as a more regularized model which has a lower variance and a slightly bigger bias. The effects of this are shown in the outliers. Indeed, as we can see, the influence of the outliers is rather small and does not seem to have a high impact if we move slightly further away from it. Differently, 1-NN does not make a difference in these terms, and marks a bigger area that surrounds the outlier with the label of this latter.
All in all, despite the fact that both of them overfit, it is possible to state that RBF with this gamma delivers a better result in terms of the expected error. Moreover, in general, RBF has the advantage that it does not need to store all of the samples needed for the classification, as KNN requires, but rather it only requires to store the supporting vectors which in general they can be much less than the total number of samples, constituting another advantage that leans us towards preferring RBF.
The overfitting that can be observed on both models is due to the fact that the VC-dimension of 1-NN is infinity, and when gamma tends to infinity, the VC-dimension of the RBF kernel also tends to infinity. Such a VC_dimenstion means that we can't bound the number of samples needed to have a bound on the error. This implies that theoretically we can always overfit, regardless of the number of the samples. For this reason the regularization term is really important when using one of the above models.

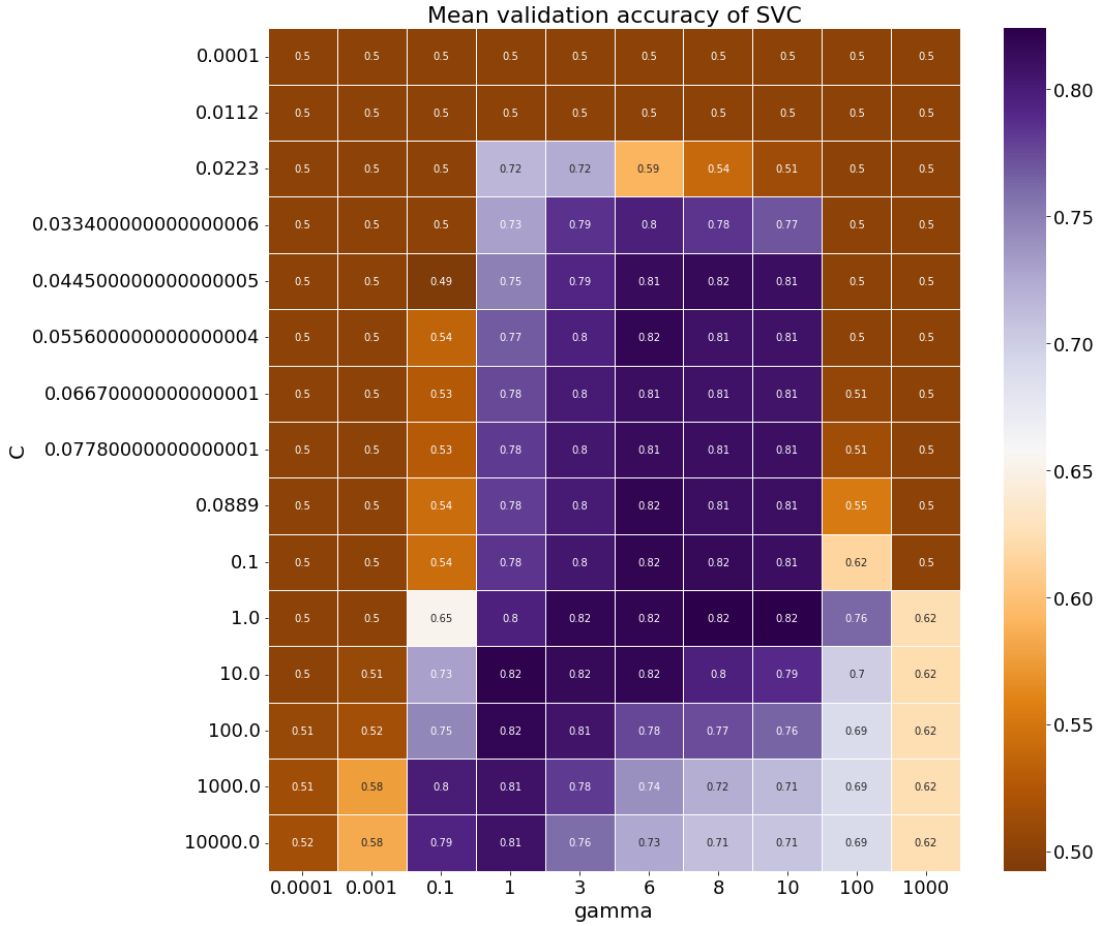
Q22)
After performing Grid Search we obtained the following parameters:

- -C = 0.02
- -gamma = 8.055

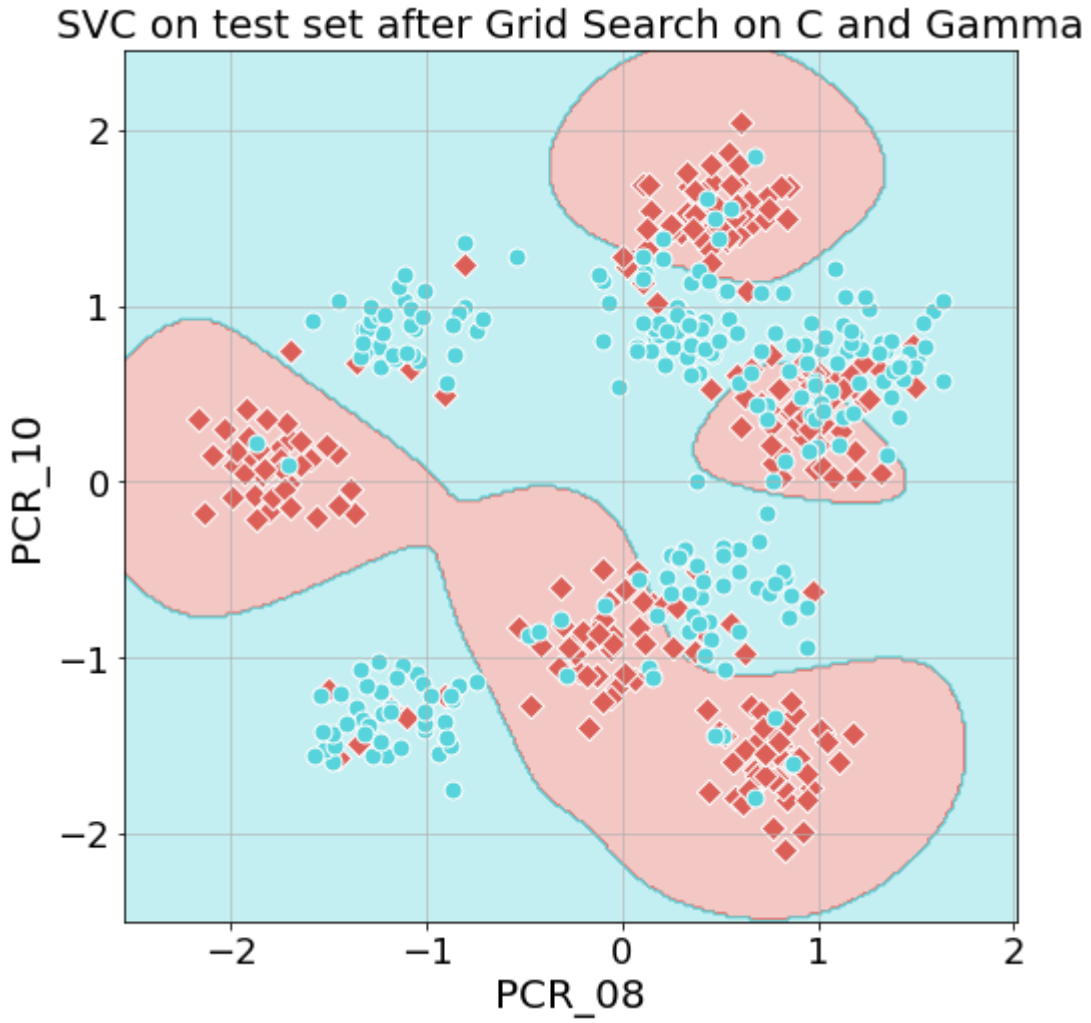
In order to find this combination we decided to look at the results we got in Q20 and we saw that for gamma under 0.1 our model was underfitted and for gamma over 10 our model was overfitted to the training set. So we decided to start (with cv = 2) to look for gamma in range [0, 20]. And after that reducing the interval until we obtained relevant results.
As we can see on the decision boundaries, our model is neither underfitted (he is able to recognize clusters) nor overfitted (noisy data don’t affect classification in a cluster).

As we can see on the heatmaps as C and gamma are increasing, the model overfits (we even obtain a 100% accuracy with extreme values on the training set while we obtained 62% on the validation set) since, as we have seen in previous questions, the higher gamma it gets the more our model will fit the training samples. And by increasing C we reduce the model estimation error, i.e we increase overfitting.
We can also observe that for low values of C and gamma our model underfits, since on both sets we obtain about 50% accuracy. This is due to the fact that when C is low there is less “weight” on the approximation error and more on the regularization. And when gamma is low on the RBF kernel the influence of each point reaches very far positions from the original point, making it more similar to a K-NN with a large K.
The smoothest spot is for gamma between 3 and 10 and for C between 0.0334 and 10


Mean training accuracy of SVC
C
0.0001 0.0112 0.0223 0.033400000000000006 0.044500000000000005 0.055600000000000004 0.06670000000000001 0.07780000000000001 0.0889 0.1 1.0 10.0 100.0 1000.0 10000.0
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
0.5 0.5 0.5 0.72 0.74 0.62 0.55 0.52 0.5 0.5
0.5 0.5 0.5 0.75 0.81 0.83 0.82 0.8 0.5 0.5
0.5 0.5 0.52 0.77 0.8 0.82 0.82 0.83 0.5 0.5
0.5 0.5 0.57 0.79 0.81 0.82 0.83 0.83 0.5 0.5
0.5 0.5 0.54 0.79 0.81 0.83 0.83 0.83 0.5 0.5
0.5 0.5 0.54 0.79 0.82 0.83 0.83 0.83 0.53 0.5
0.5 0.5 0.55 0.79 0.82 0.82 0.83 0.83 0.61 0.5
0.5 0.5 0.56 0.79 0.82 0.83 0.83 0.83 0.72 0.5
0.5 0.5 0.68 0.82 0.83 0.84 0.84 0.85 0.91 1
0.5 0.52 0.75 0.83 0.84 0.85 0.86 0.86 0.97 1
0.52 0.53 0.75 0.84 0.86 0.87 0.88 0.89 0.99 1
0.53 0.58 0.8 0.84 0.87 0.89 0.92 0.93 1 1
0.53 0.61 0.81 0.85 0.88 0.93 0.94 0.95 1 1
0.0001 0.001 0.1 1 3 6 8 10 100 1000
gamma
1.0 0.9 0.8 0.7 0.6


Mean validation accuracy of SVC
C
0.0001 0.0112 0.0223 0.033400000000000006 0.044500000000000005 0.055600000000000004 0.06670000000000001 0.07780000000000001 0.0889 0.1 1.0 10.0 100.0 1000.0 10000.0
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
0.5 0.5 0.5 0.72 0.72 0.59 0.54 0.51 0.5 0.5
0.5 0.5 0.5 0.73 0.79 0.8 0.78 0.77 0.5 0.5
0.5 0.5 0.49 0.75 0.79 0.81 0.82 0.81 0.5 0.5
0.5 0.5 0.54 0.77 0.8 0.82 0.81 0.81 0.5 0.5
0.5 0.5 0.53 0.78 0.8 0.81 0.81 0.81 0.51 0.5
0.5 0.5 0.53 0.78 0.8 0.81 0.81 0.81 0.51 0.5
0.5 0.5 0.54 0.78 0.8 0.82 0.81 0.81 0.55 0.5
0.5 0.5 0.54 0.78 0.8 0.82 0.82 0.81 0.62 0.5
0.5 0.5 0.65 0.8 0.82 0.82 0.82 0.82 0.76 0.62
0.5 0.51 0.73 0.82 0.82 0.82 0.8 0.79 0.7 0.62
0.51 0.52 0.75 0.82 0.81 0.78 0.77 0.76 0.69 0.62
0.51 0.58 0.8 0.81 0.78 0.74 0.72 0.71 0.69 0.62
0.52 0.58 0.79 0.81 0.76 0.73 0.71 0.71 0.69 0.62
0.0001 0.001 0.1 1 3 6 8 10 100 1000
gamma
0.80 0.75 0.70 0.65 0.60 0.55 0.50

Q23)



With the optimal parameters we obtained an accuracy of 0.826.

As we can remember we obtained an accuracy of 0.782 with the 1-NN model and an accuracy of 0.826 with the 200-NN model (When 200 was the optimal K obtained)
So both models are even in terms of accuracy.