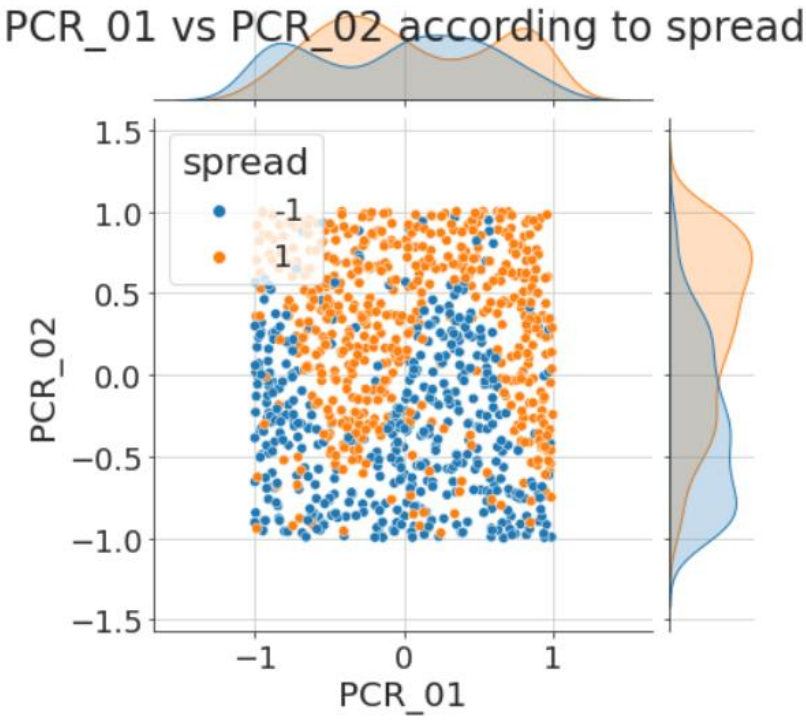
Major HW2 – Final Report

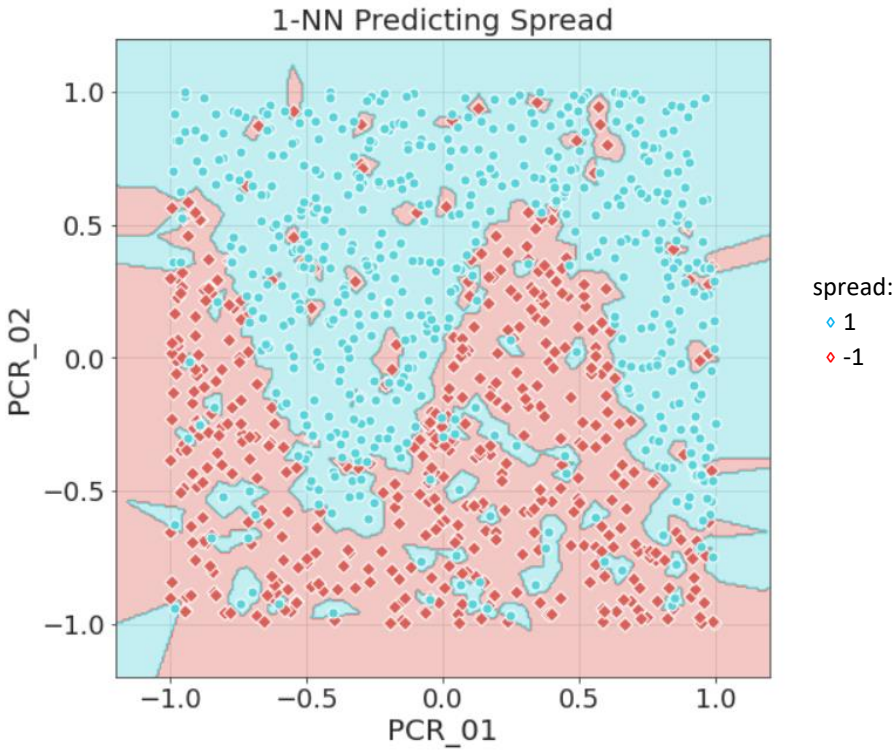
(Q1)

The following is a jointplot of PCR_01 and PCR_02 according to spread:



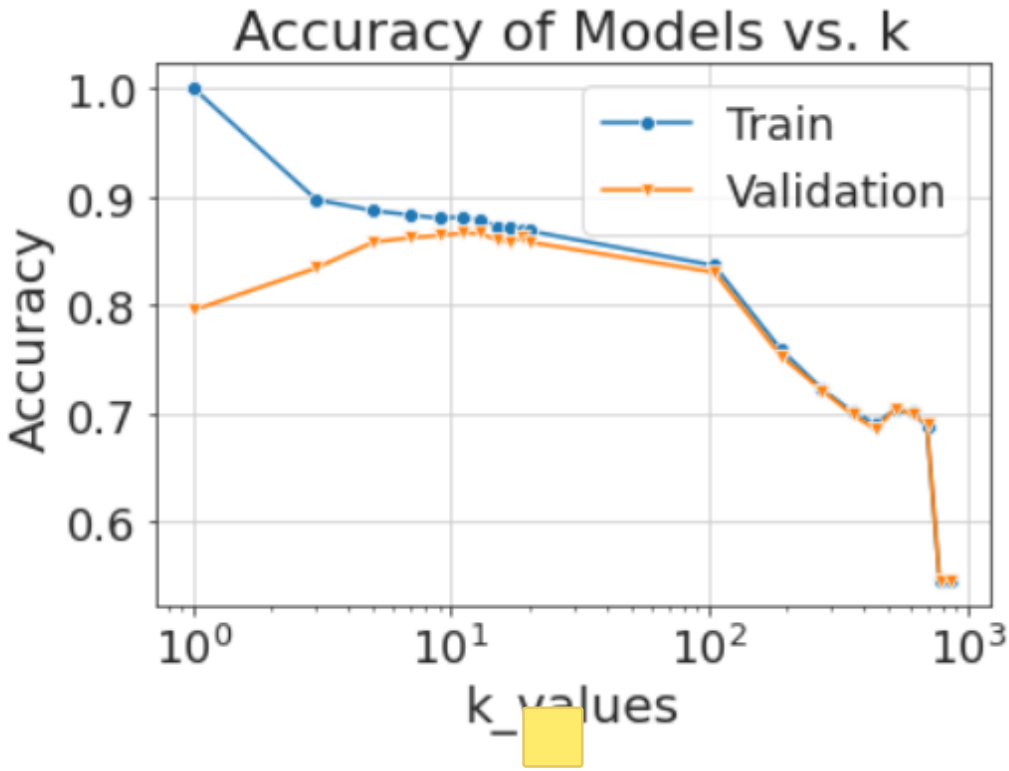
(Q2)

The decision regions of a 1-NN classifier:



(Q3)

The following is a plot of the accuracy of the models vs. the value of k (where k is on a logarithmic axis):



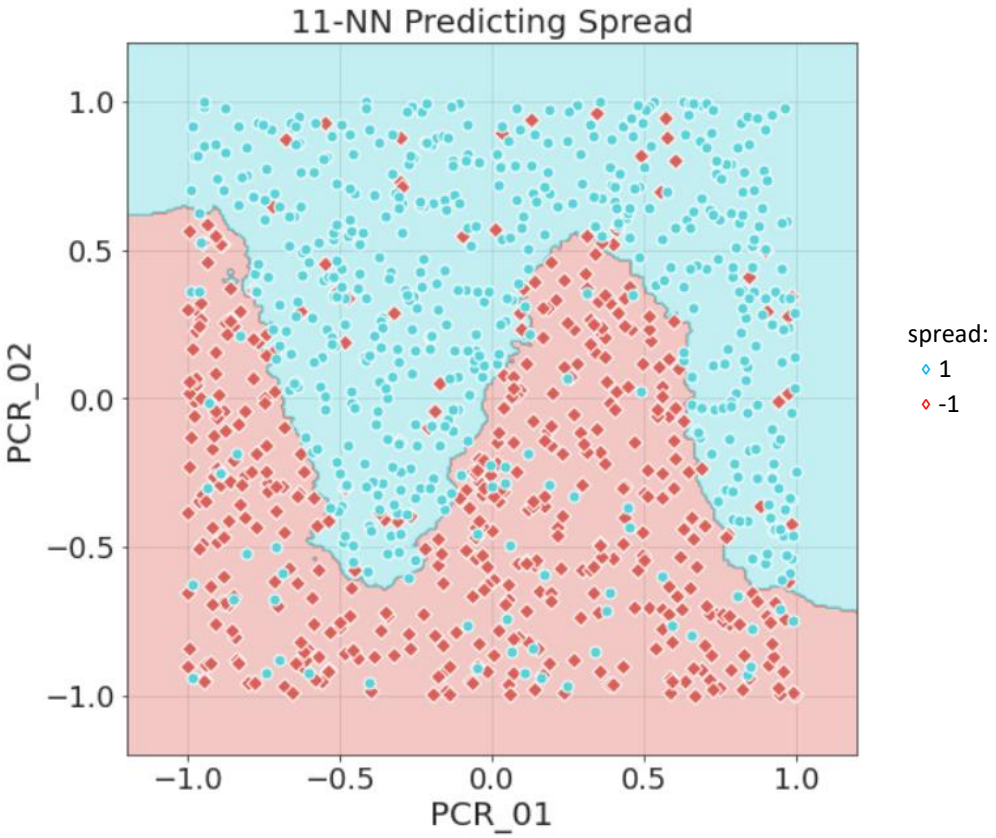
The best k value is 11 with a mean validation accuracy of 0.866 and a mean train accuracy of 0.881.

Low k in kNN values cause overfitting – since few neighbors are taken into account, the influence of each neighbor is high, and therefore outliers in the training set can create a region of wrong classification around them. This can be seen in the plot – the train accuracy is high but the generalization is not as good (validation accuracy is lower).

High k values cause underfitting – as k gets large, samples that are far away are taken into account equally as neighbors, even though their relevance is low, and so, the more important close neighbors have less weight in the classification than they should. This results in low accuracy (both training and validation). It is interesting to note that training and validation accuracies are virtually the same for large k's – this is reasonable since an underfit model barely refers to the specific training set it is trained on (so the behavior on the validation set is equally as bad).

(Q4)

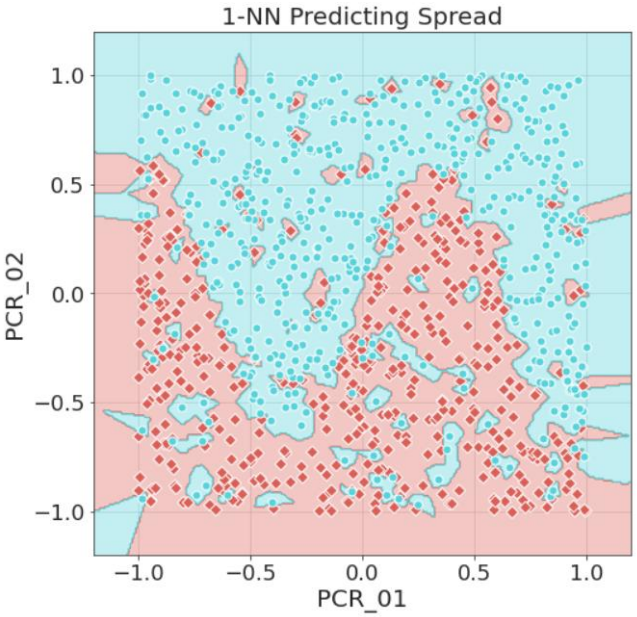
The following plot shows the decision boundaries for the optimal k=11 we found:

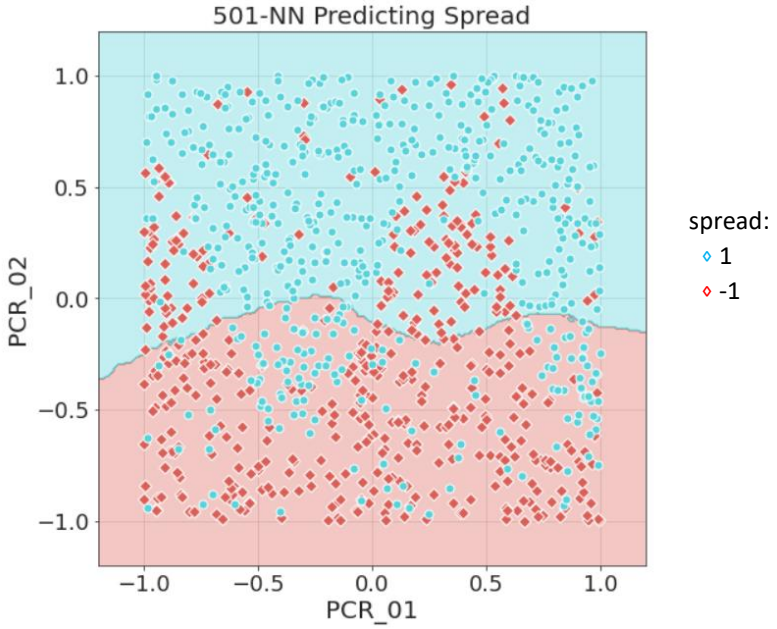


The accuracy of the model on the test set is 0.892.

(Q5)

We trained two additional models with k=1, 501:





The decision boundaries of these models behave as we explained:

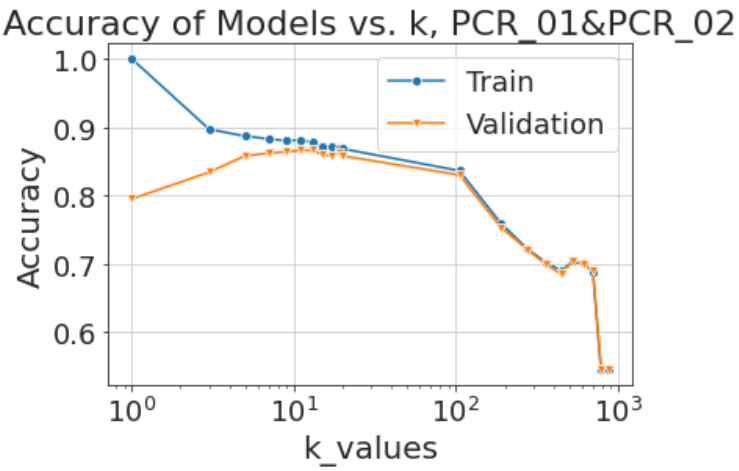
For k=1, the lowest possible value for k, every point creates a decision region around it. Therefore, all outliers create decision regions of wrong classification, and the generalization ability is low – an overfitting situation.

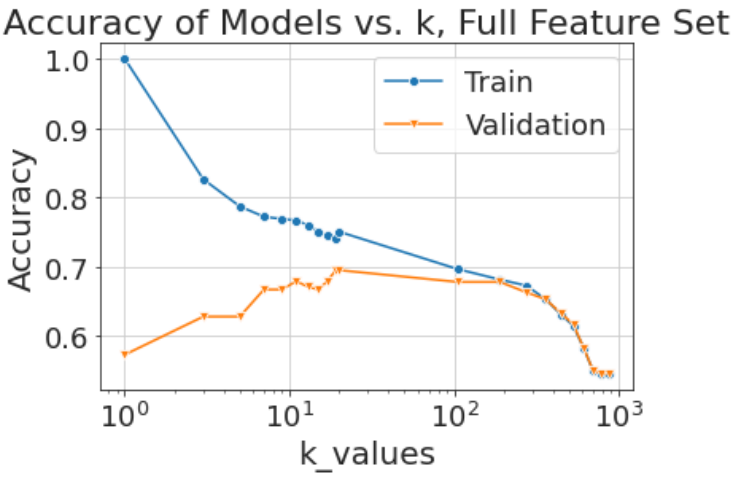
However, k=501 is a very large value for this dataset: for comparison, the whole training set has 1000 samples, so each point takes into account as neighbors roughly half of the samples! This is a classic underfitting situation which leads to low accuracy (both training and test). For example, the large mass of spread=-1 (red) values around $PCR_{01}, PCR_{02} \in [0,0.5]$ is wrongly classified as 1 (blue) since it is surrounded by mostly blue points, in the upper half of the plane, and since k is large, we take into account more of the surrounding points than the points of the mass itself.

Both of these contrast to the decision boundaries of the optimal k=11 we found; the boundaries for k=11 seem on one hand suitable to the training set and on the other hand not overfit to each and every sample.

(Q6)

For convenience, we show side by side the validation curves of the new kNN model (with the whole feature set) and the old plot from (Q3):



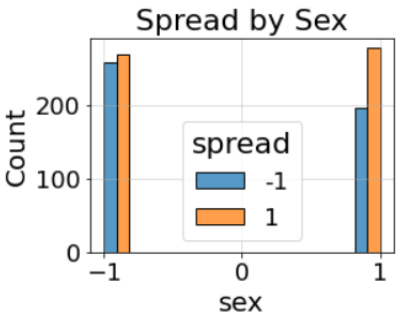


First, it should be noted that the overall trend of the train and validation plots is similar for both graphs – since for both, low k values lead to overfitting (high training accuracy and low validation accuracy), and high k values lead to underfitting and plummeting of both train & validation accuracies.

However, as we can see, the new models are significantly worse than the PCR_01 x PCR_02 models: the training and validation accuracies are worse for all values of k (except perhaps the extreme large k) and the best validation accuracy in the new models is approximately 0.7, compared to 0.866 previously.

The reason behind this, is probably that most features in the dataset are not appropriate for a k-NN model that predicts *spread*. The choice of a kNN model assumes that proximity (via Euclidean distance) between two samples, predicts equality in regard to the target label. In our dataset, this is

clearly not the case for all features. For example, the *sex* feature has almost no correlation to the *spread* label:



Given two random samples with *sex=-1,* the chances that their *spread* label is different is almost 50%. For *sex=1* the chances are slightly lower but still very significant. This means that proximity (actually – equality) in regard to the *sex* feature does not predict equality in regard to the *spread* label.

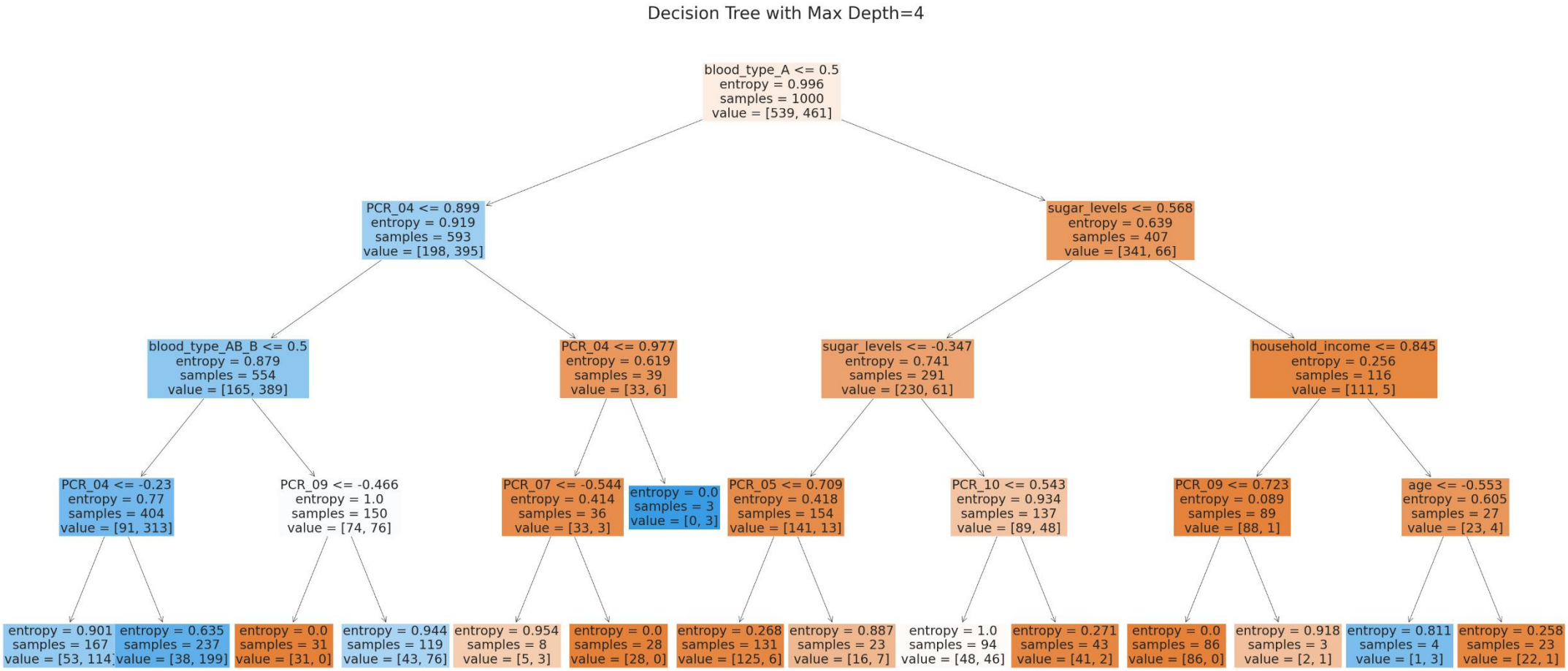
A similar observation can be made for many of the other features as well.

These “bad” features affect the distance calculated between each two samples just as much as the important, correlative features, but their distance does not correlate with the target label. Therefore, when the kNN model classifies a point, the neighbors chosen will not accurately reflect similarity to the given point with regard to the target label, and so the altogether prediction is harmed.

This explains why it is very important to carefully choose appropriate features when using any kind of learning algorithm, and specifically kNN.

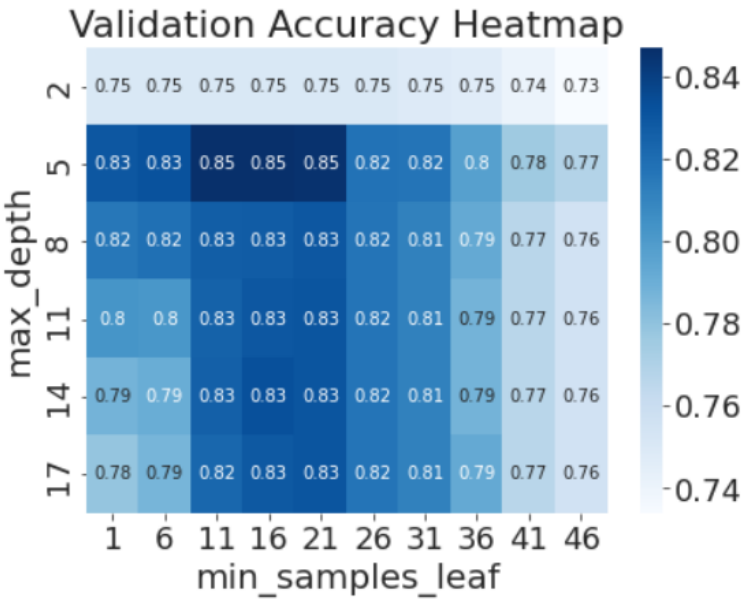
(Q7)

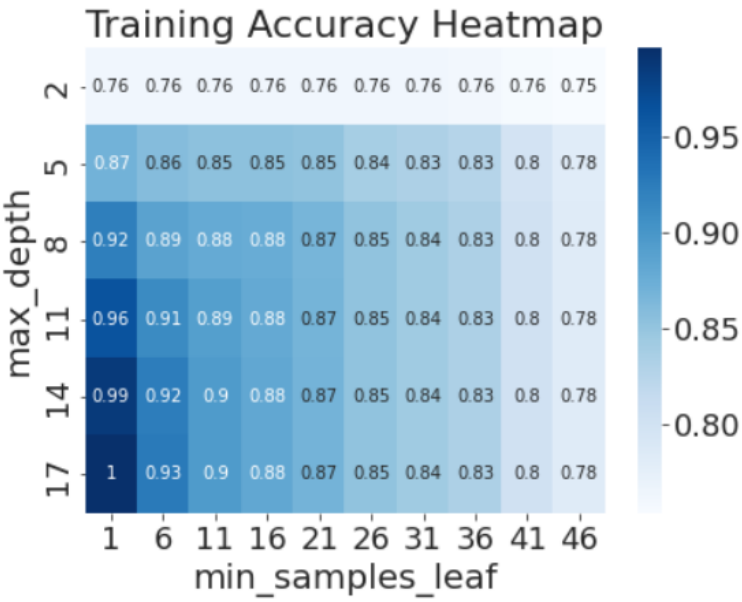
The training accuracy we received for the ID3 Decision Tree model with max_depth=4 was 0.799.



(Q8)

After searching for appropriate parameter ranges, we choose *min_samples_leaf* ∈ [1,50] and *max_depth* ∈ [2,20]. We obtained the following heatmaps for the mean validation and training accuracies:





The optimal combination for validation accuracy is *max_depth*=5 and *min_samples_leaf* of 11-21.

An example hyperparameter combination that caused **underfitting** is *max_depth*=2 and *min_samples_leaf*=46.

An example hyperparameter combination that caused **overfitting** is *max_depth*=17 and *min_samples_leaf*=1.

The reasons for this behavior:

Low values of *max_depth* lead to underfitting since few decisions are made regarding the data – more useful splits still remain to execute on the data. Similarly, high values of *min_samples_leaf* also cause underfitting since they stop the splitting process, even if the entropy of the node is high. This can be seen in the heatmaps – for these combinations we have low training and validation accuracies.

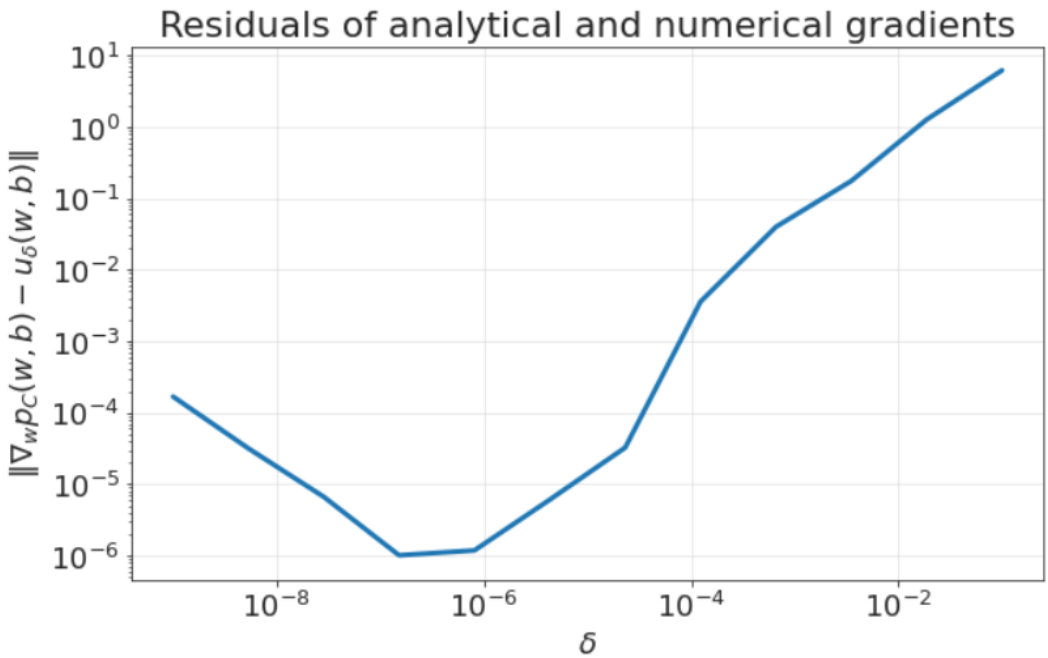
On the other hand, high values of *max_depth* and low values of *min_samples_leaf* lead to overfitting since they allow extensive splitting in a way that fits small amounts of samples, even if they are outliers, therefore highly fitting the training set and causing low generalization. We can see that for these values, training accuracy is very high (and even reaches 1) while validation accuracy suffers a loss.

(Q9)

We trained a decision tree with *max_depth*=5 and *min_samples_leaf*=16 and obtained a test accuracy of 0.828.

(Q10)

The following plot describes the residuals of the numerical and analytic subgradient:

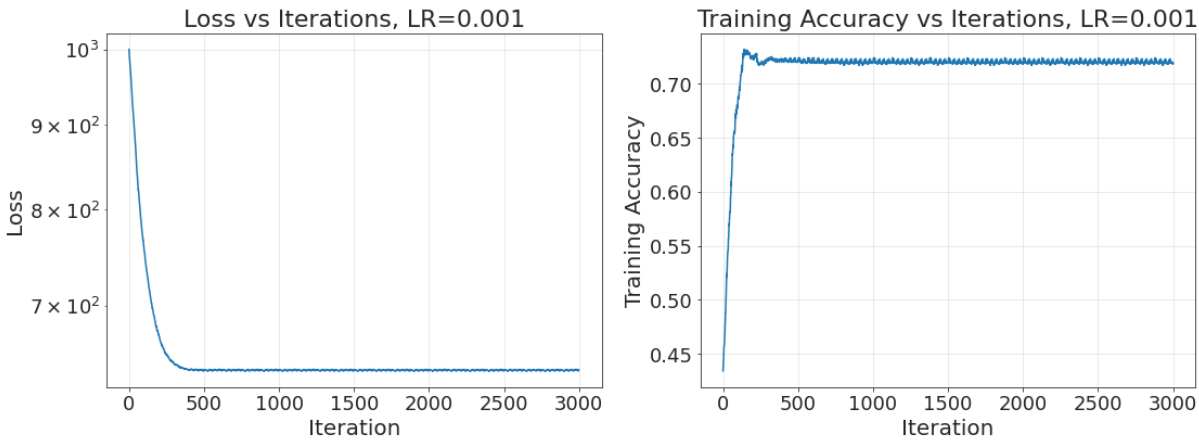


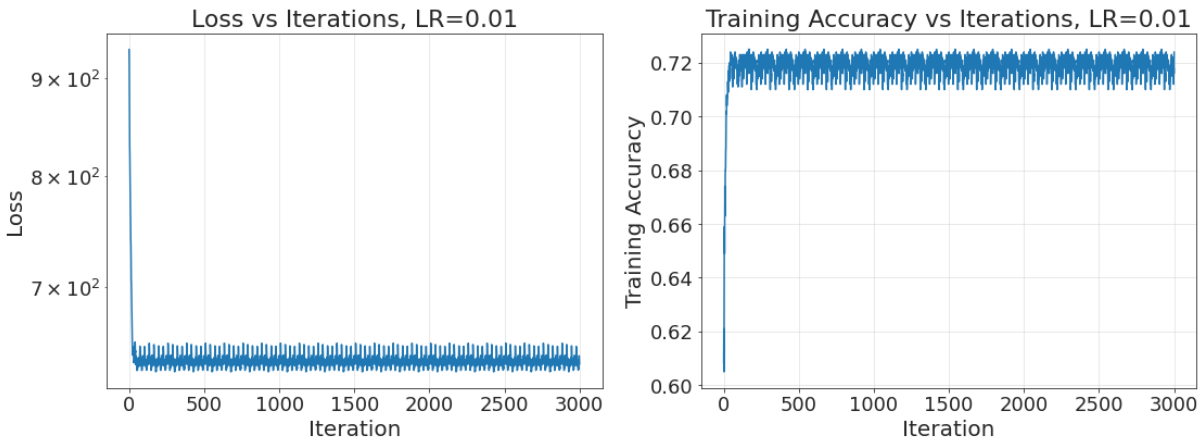
As we can see, as δ gets smaller, the residuals converge to zero – this is because the numerical subgradient converges to the analytic formula as δ goes to zero, as we expect. Note that for very small values of δ ($\leq\approx 10^{-7}$) the residuals grow; most likely this is due to computational limitations that lead to numerical errors when dividing by infinitesimal numbers.

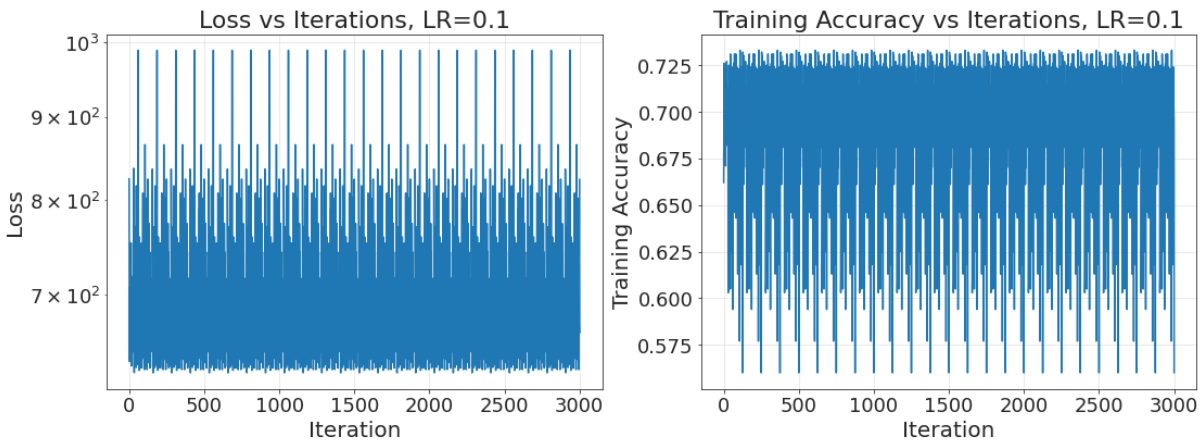
(Q11)

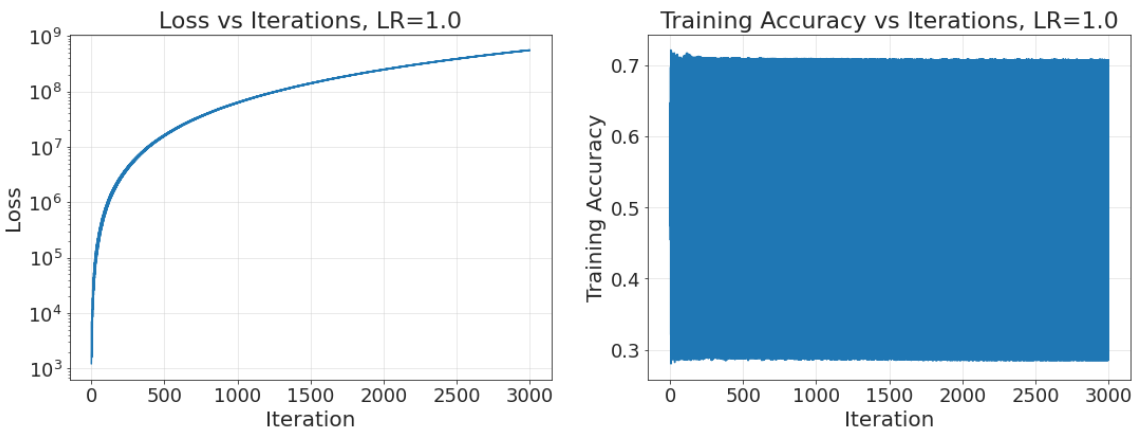
We trained an SVM model with a regularization parameter C of 1. We increased C from 0.1 to 1 because the initial value of 0.1 gave a minimum point of (w, b) that was an underfit model: the resulting classifier was a majority-rule classifier and had low accuracy (~0.55).

We received the following plots for learning rates 0.001, 0.01, 0.1, 0.1, 1: (on the next page)


Loss vs Iterations, LR=0.001
Loss
Iteration
Training Accuracy vs Iterations, LR=0.001
Training Accuracy
Iteration


Loss vs Iterations, LR=0.01
Loss
Iteration
Training Accuracy vs Iterations, LR=0.01
Training Accuracy
Iteration

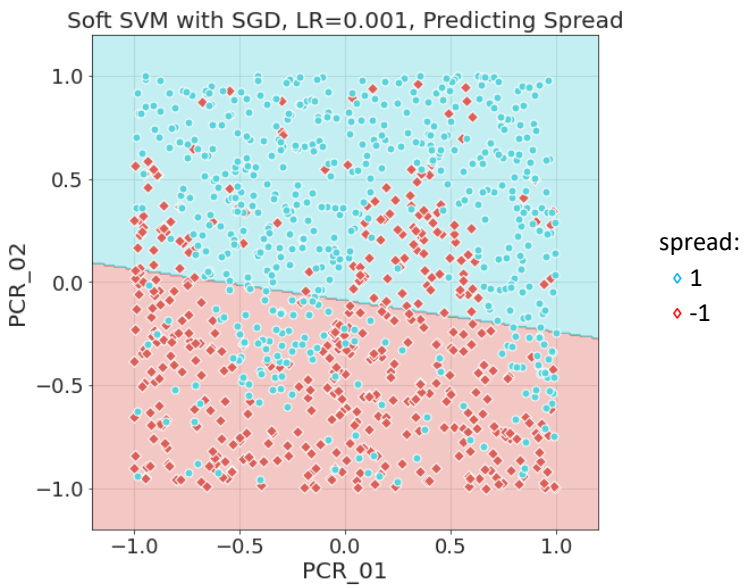

Loss vs Iterations, LR=0.1
Loss
Iteration
Training Accuracy vs Iterations, LR=0.1
Training Accuracy
Iteration


Loss vs Iterations, LR=1.0
Loss
Iteration
Training Accuracy vs Iterations, LR=1.0
Training Accuracy
Iteration

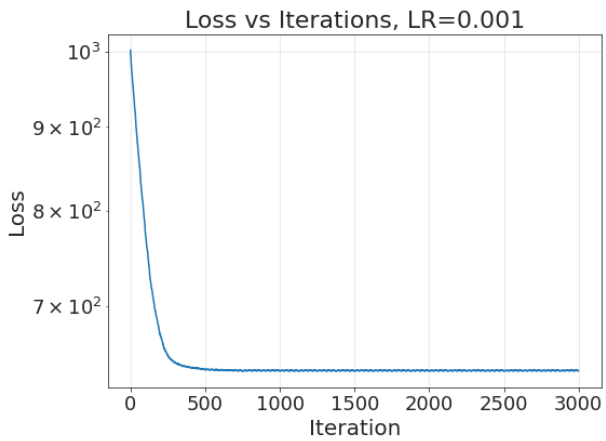
a. Based on the above plots, the learning rate we would choose is 0.001:

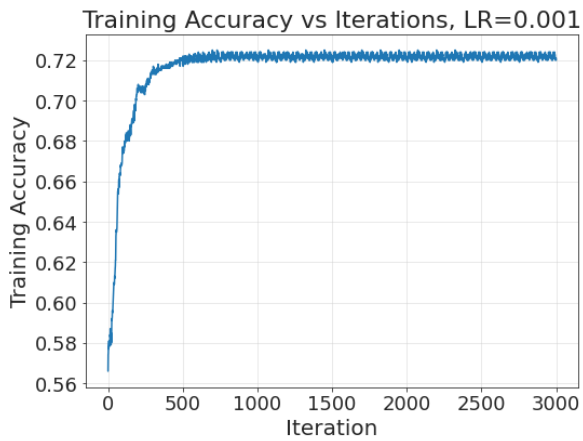
- First, we can notice that the two rates 0.001, 0.01 converge to approximately the same loss value ($\approx 6.5e2$) and accuracy (≈ 0.72). In contrast, for learning rate 0.1 the loss oscillates wildly – probably the step size is too large, which doesn't allow w, b to reach the minimum. For learning rate 1 the loss completely diverges: w, b get farther away from the minimum with each iteration.
- LR=0.001 converges in ≈ 300 iterations and LR=0.01 in very few (appears $O(1)$). Their behavior is similar, but for LR=0.01 we have larger oscillations around the minimum. It is still enticing to choose LR=0.01 because of the very fast convergence, but we preferred to allow ourselves a few hundred iterations to reach a more precise minimum.

b. The following plot shows the decision regions for LR=0.001:



c. The loss and accuracy plots for the model with 0.001 (the model with the decision regions above)



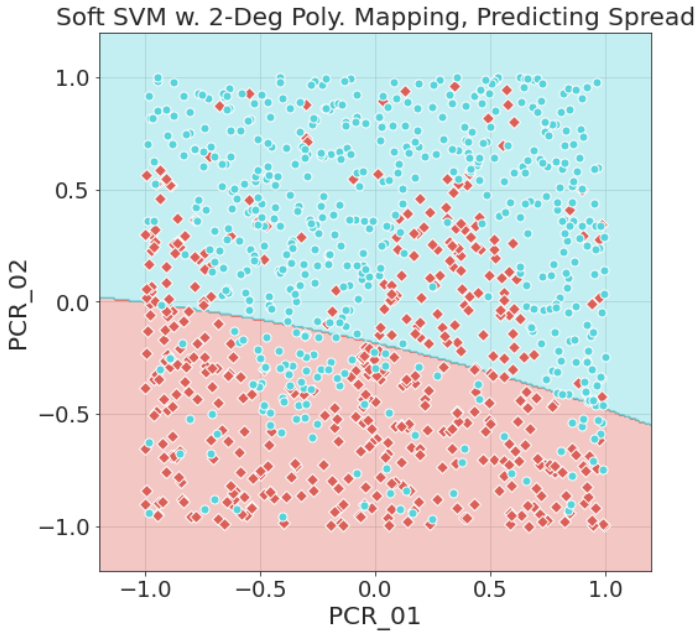


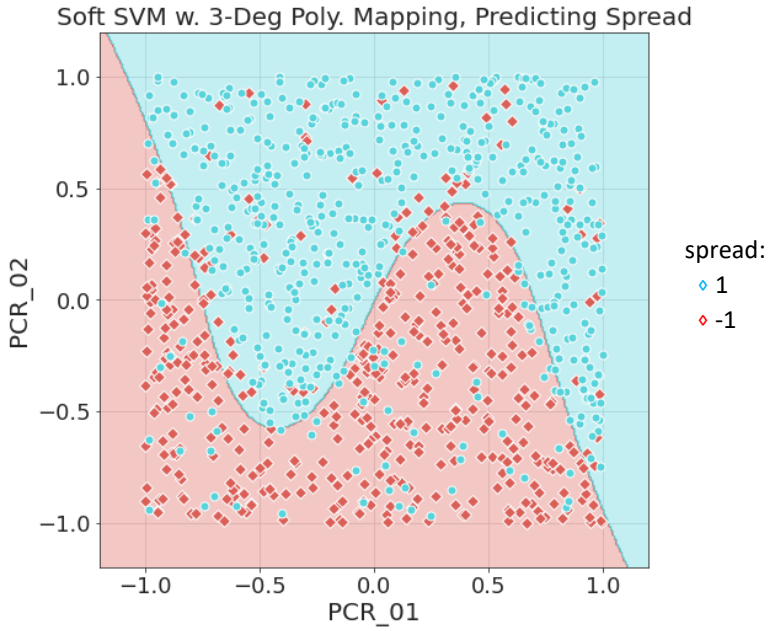
By checking the loss and score logs from the algorithm, we obtained that the minimum loss was 638.77 in iteration 754 and the maximum accuracy was 0.725 in iteration 2324. They are not attained at the same step: this is because the SVM objective loss function is the sum of two components: the hinge loss on the training samples (which is inversely related to the training accuracy) and the regularization. This creates two possible sources for the differences in minimum points:

1. The regularization component changes the objective: we are not minimizing only the training samples loss, but the sum of both components.
2. Hinge loss, which is part of the SVM objective, is not 0/1 loss, which is the loss used for calculating training accuracy. This means the classifier that minimizes the sum of hinge losses does not necessarily minimize the sum of 0/1 losses: for example, one very extreme outlier out of many samples creates a very large penalty for a hinge loss classifier but a negligible penalty for 0/1 loss, so the resulting optimal classifiers for these losses would be different (the hinge loss classifier would alter the separator to reduce the penalty for the outlier while the 0/1 classifier would not be affected by it).

(Q12)

We trained a Soft-SVM model on the data after two feature mappings: 2-deg polynomials and 3-deg polynomials.





Training accuracy: 0.73

Test accuracy: 0.756

Training accuracy: 0.845

Test accuracy: 0.84

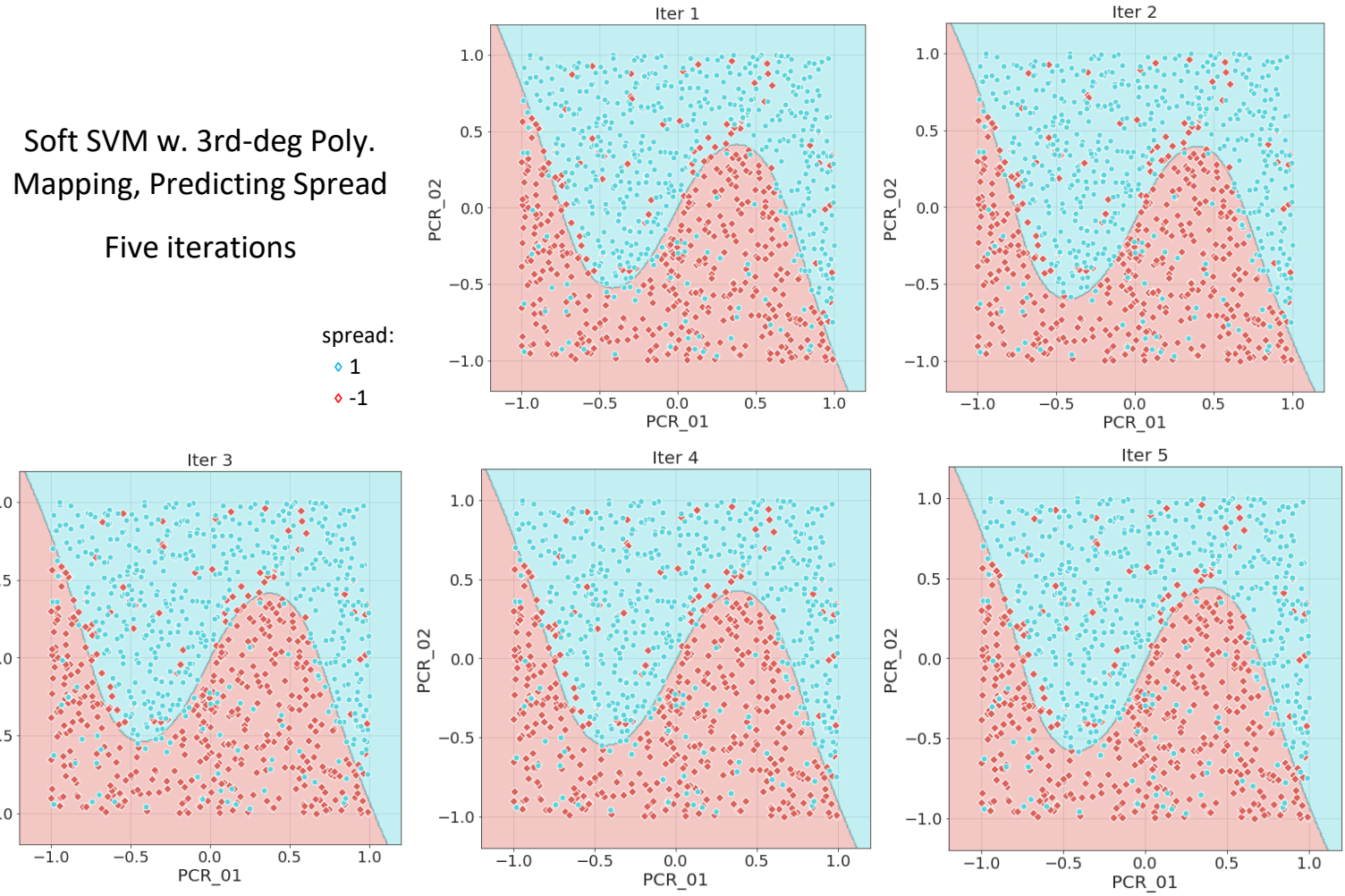
We can clearly see that the 3rd degree polynomial feature mapping has much better accuracy results and our visual impression is that it fits the distribution of the data well.

We can explain this based on the theory of feature mappings: after we execute the feature mapping, the optimization helps us find the best coefficients of a linear combination of the components of the mapping – in our case, we map (PCR_01, PCR_02) to a vector of monomials up to degree $k \in \{2,3\}$, which means in effect that we are trying to fit a k-degree polynomial in variables PCR_01, PCR_02 such that its sign predicts our two classes.

A 3rd-degree polynomial is much more expressive than a 2nd-degree polynomial, and specifically for our data, we have a "wavy" curve on the PCR_01 x PCR_02 plane with two extremums, which is suitable for a 3rd-degree polynomial, but unsuitable for a 2nd-degree polynomial (we can see that the 2nd-degree polynomial has a simpler, parabolic shape).

(Q13)

We ran five iterations of Soft-SVM with the 3rd degree polynomial mapping:



The training accuracies we obtained (by order): 0.841, 0.844, 0.838, 0.836, 0.843.

Mean: 0.8404

Standard deviation: 0.003

As can be seen from the visualization and from the standard deviation of the accuracies, the variability of the models is very low. We can explain this since we know that after the feature mapping, we are still solving a Soft-SVM problem, and we know that the Soft-SVM objective function is convex. This means it has a single global minimum, so, if we choose a correct learning rate we are assured to converge close to this minimum with Gradient Descent.

Still, there is **some** variability, which comes from:

1. Different starting points (in our implementation the starting point is chosen randomly)

2. Random batches – the SGD algorithm calculates the sub-gradient at each step using only a batch of fixed size. In our implementation we randomize a permutation on the data in the beginning of the algorithm, so in different runs we have different orders for the samples and therefore different batches which leads to different steps.

(Q14)

Let us observe the behavior of RBF when $\gamma \to \infty$. Let $\{(x_i, y_i), \alpha_i\}_{i\in[m]}$ be our sample set, labels, and dual coefficients, respectively. Let x be some point we wish to classify, and denote $i^* = argmin_{i\in[m],\alpha_i>0}\|x - x_i\|^2$. So:

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

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

Since $e^{-\gamma\|x-x_{i^*}\|^2} > 0$,

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

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

As $\gamma \to \infty$, since $\|x - x_i\|^2 - \|x - x_{i^*}\|^2 > 0$ for every $i \neq i^*$ (according to the simplification assumption that there are no two points of equal distance, and since x_{i^*} has the minimal distance), $e^{-\gamma\left(\|x-x_i\|^2-\|x-x_{i^*}\|^2\right)} \to 0$, and therefore $\sum_{\boldsymbol{i^*\neq i}\in[m],\alpha_i>0} \alpha_i y_i e^{-\gamma\left(\|x-x_i\|^2-\|x-x_{i^*}\|^2\right)} \to 0$.

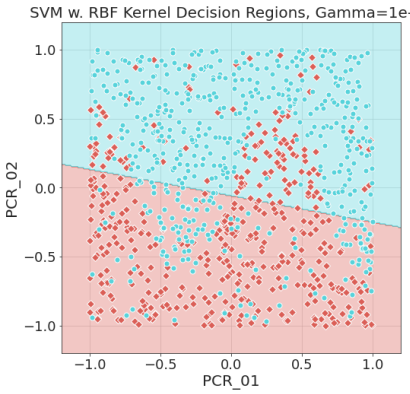
So

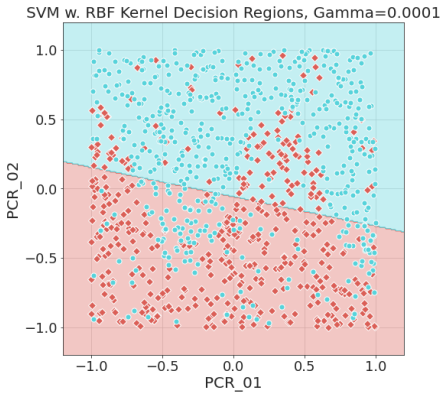
$$\lim_{\gamma\to\infty} sign\left(\alpha_{i^*}y_{i^*} + \sum_{\boldsymbol{i^*\neq i}\in[m],\alpha_i>0} \alpha_i y_i e^{-\gamma\left(\|x-x_i\|^2-\|x-x_{i^*}\|^2\right)}\right) = sign(\alpha_{i^*}y_{i^*}) \overset{\alpha_{i^*}>0}{\cong} sign(y_{i^*}) = y_{i^*}$$

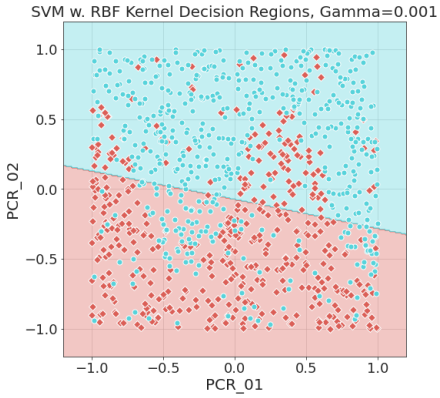
As required.

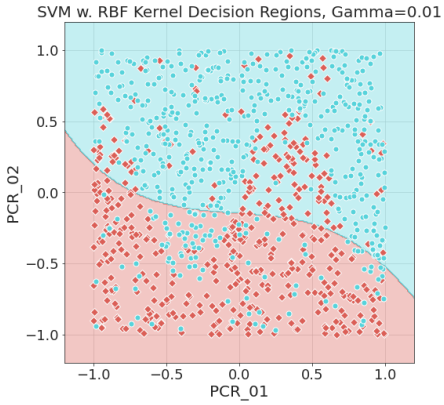
(Q15)

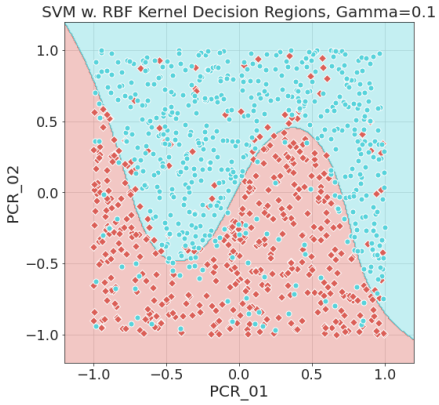
The following plots show decision regions for SVM with a RBF Kernel for different gamma values:

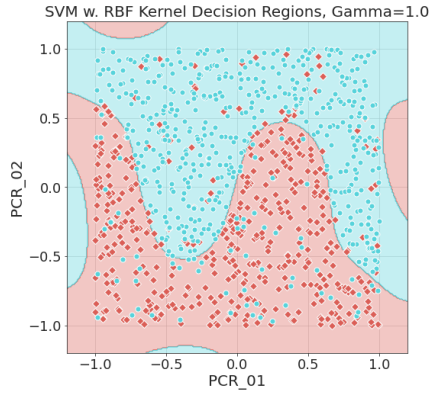


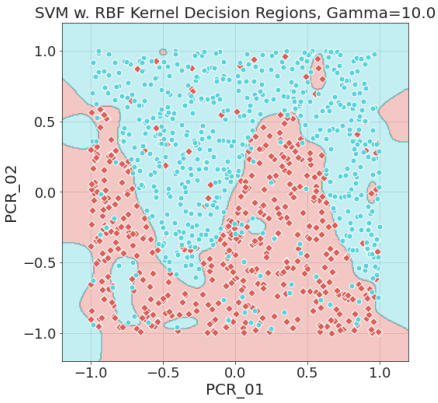


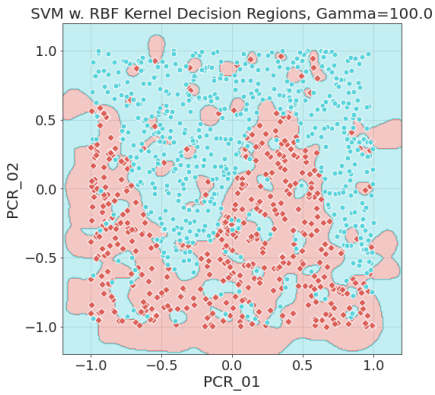


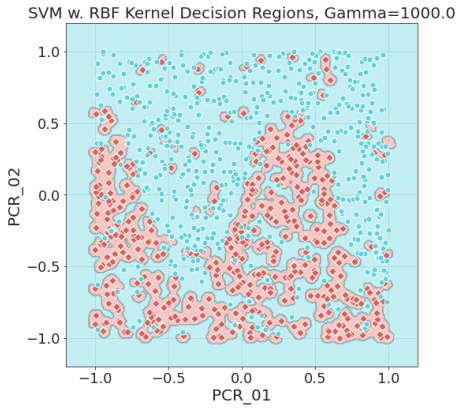






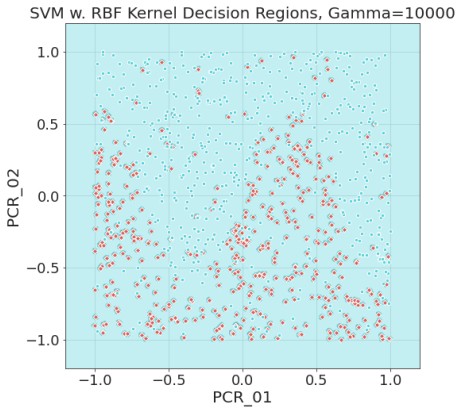






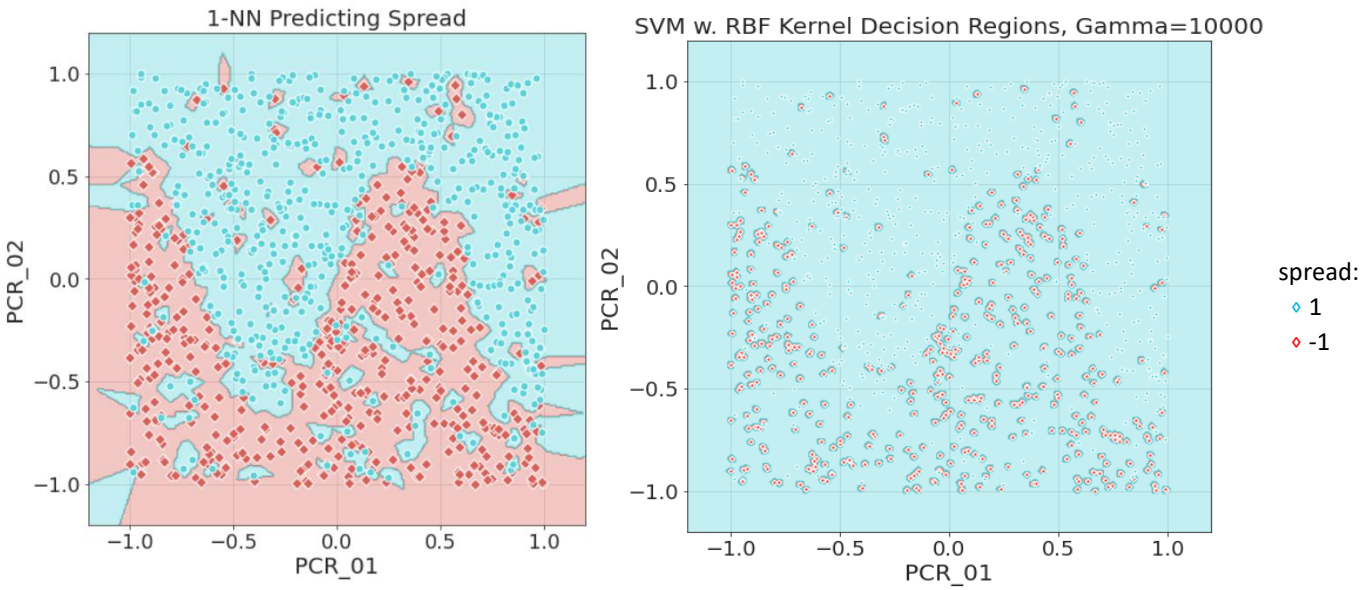
spread:

- 1
- -1



(Q16)

Let us see them side by side:



We expected from the theoretical derivation to obtain a model similar to 1-NN for very large gamma values in the SVM RBF kernel. However, we got a model in which the "red" (-1) decision boundaries are very small and concentrated around the red points, while the "blue" (1) decision regions dominate most of the plane.

We think the reason for this is **the intercept b** in the formulation of the SVM problem in the library we are using. According to the documentation,

Once the optimization problem is solved, the output of decision_function for a given sample x becomes:

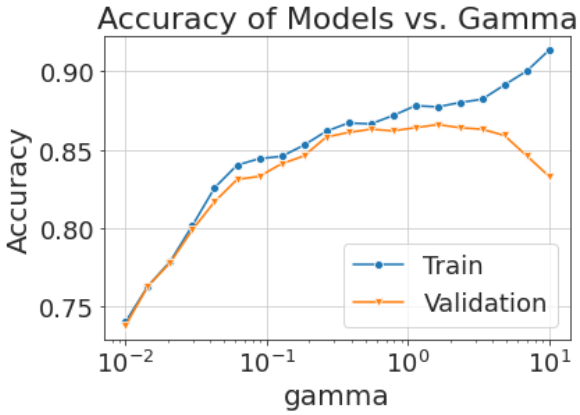
$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b,$$

The optimization of the dual SVM problem chooses the dual coefficients α_i and the intercept b to bring the dual objective function to a minimum. Notice that *b* was not present in **our** formulation of the problem, in which we proved theoretically that for $\gamma \to \infty$ the RBF SVM should behave like 1NN.

Now, what occurs is that *b* obtains some value – in the case of our plot above, *b=0.08*. When the decision function is calculated, $K(x_i, x) = \exp\left(-10{,}000 \cdot \|x_i - x\|^2\right)$ is extremely small almost on every point x of the plane and for any training sample x_i. Therefore, and since α_i also obtains some fixed value which does not come close to compensating for $K(x_i, x)$, the dominating factor in the decision factor becomes b, unless one is very close to a training sample (and as γ grows, one needs to get exponentially closer). Since b in our case is positive, most of the plane is colored blue, and the model does not exhibit a 1-NN behavior that we would expect from a model without an intercept.

(Q17)

The following plot shows the mean training and validation accuracies for 8-fold validation for different values of gamma (20 values sampled at logarithmic intervals in the range 0.01, 10):



We can see that our plots behave as expected according to the theory of underfitting/overfitting, where gamma is the relevant parameter.

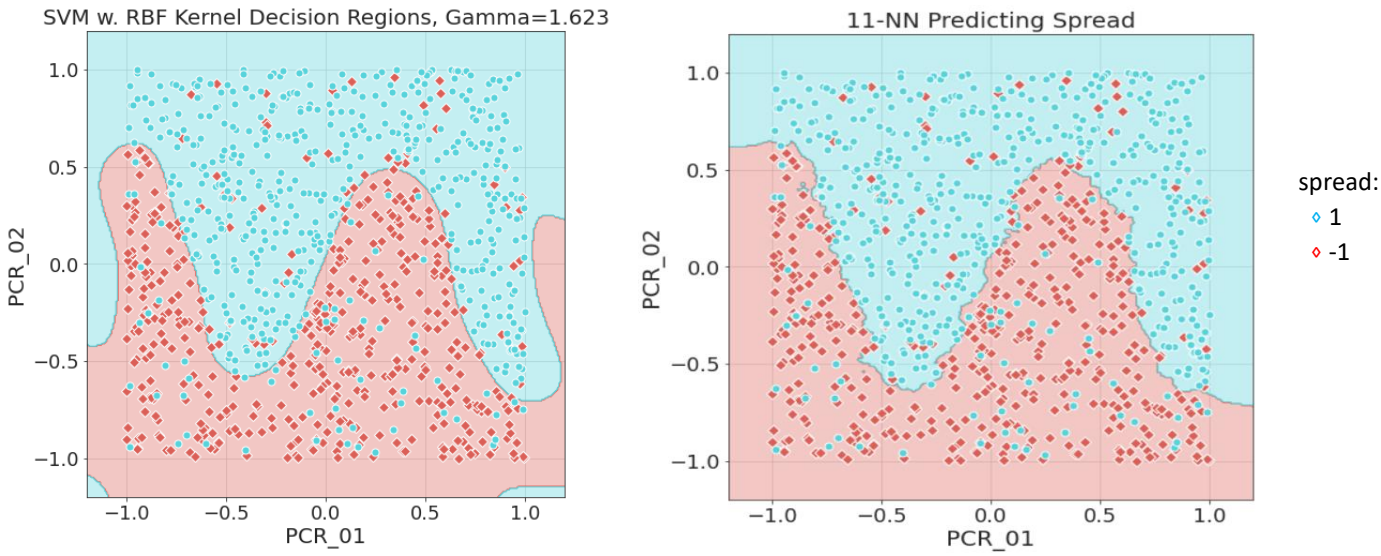
For low values of gamma, we have underfitting, which means both training and validation accuracies are low since the model barely fits the data. This occurs since the values of $K(x_i, x) = e^{-\gamma\|x-x_i\|^2}$ are significant for many points x_i, so many of x's neighbors have a significant impact on the decision function (similarly to high k values in kNN).

Around gamma=1 we reach a peak ("sweet spot"), and as gamma continues to grow, training accuracy grows and validation accuracy decreases since the model is overfit to the training set and therefore does not generalize well to the validation set. This occurs since $K(x_i, x)$ is significant only for x_i's that are very close to x, so very few of x's neighbors have a significant impact on the decision function (similarly to low k values in kNN).

The optimal value of gamma (best validation accuracy) is gamma=1.623 with a mean training accuracy of 0.877 and a mean validation accuracy of 0.866.

(Q18)

The following plot shows the decision regions for the optimal gamma we found, side by side with the optimal kNN model (11-NN):



The test accuracy for the RBF kernel model is 0.872, while the test accuracy for the 11-NN model is 0.892. This means that based on the test accuracy alone, the 11-NN model is better for the task of predicting *spread* than the RBF model.

However, we can notice some qualitative differences between the plots:

1. In general, the RBF model is “smoother”, meaning that near the edges of the boundary decisions, it is less overfit to the training samples compared to the “jagged” formation of the 11-NN model. This is an advantage of the RBF model since we assume (reasonably) that the underlying distribution is smooth.

2. The RBF model has strange behaviors of the decision regions near the edges of the feature plane, which is a disadvantage since outliers in the test set with edge feature values can be classified wrongly. This may be remedied by choosing a slightly smaller value of gamma, closer to 0.1: we can see from the plot in (Q17) that for $\gamma \in [0.1,2]$ the validation accuracies are approximately the same, but the decision boundaries for $\gamma = 0.1$ (in Q15) seem more “reasonable”, intuitively.