HW2 Report

Part 1

Q1:

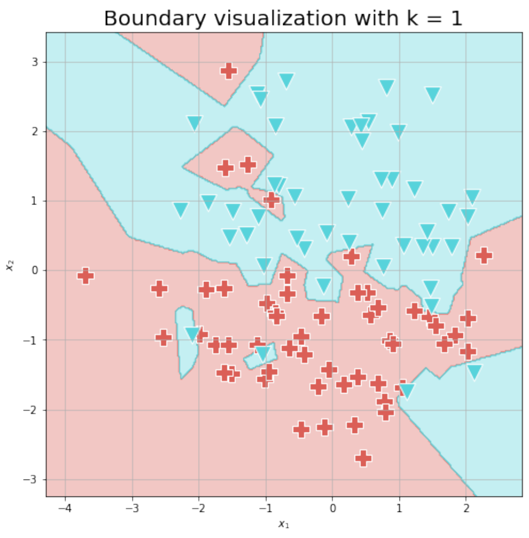


Figure 1: Boundary Visualization for KNN with K = 1

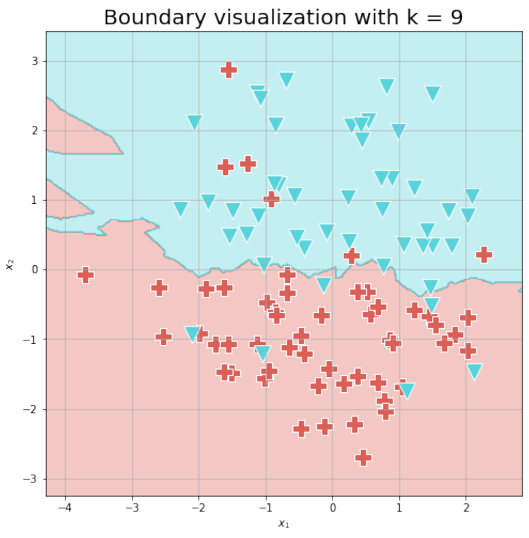
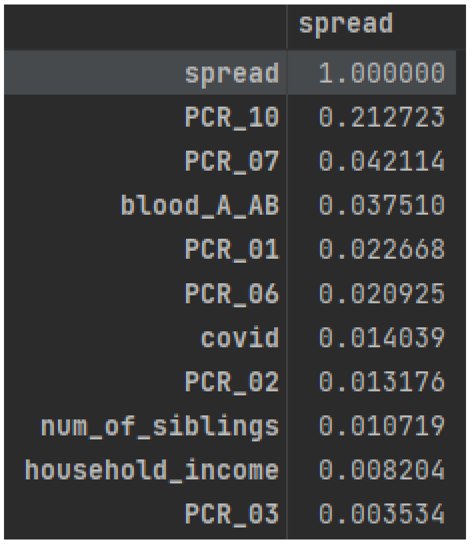


Figure 2: Boundary Visualization for KNN with K = 9

As can be seen in figure 1, the model with k equal to 1 caused overfitting to occur, as for every new sample point, its prediction took into account only the closest point to it, thereby increasing the complexity of the model since every new point is predicted locally. On the other hand, as can be seen in figure 2, the model with k equal to nine takes into account the nine closest points to the sample being predicted, and therefore takes a broader outlook on the training data in-order to tdo the prediction, thereby lowering the complexity. But at the same time, we can see that a number of points were not classified correctly, and therefore this could indicate that this model is performing underfitting.

Q2:



	spread
spread	1.000000
PCR_10	0.212723
PCR_07	0.042114
blood_A_AB	0.037510
PCR_01	0.022668
PCR_06	0.020925
covid	0.014039
PCR_02	0.013176
num_of_siblings	0.010719
household_income	0.008204
PCR_03	0.003534

Figure 3: The 10 most correlated features to `spread`

Q3:

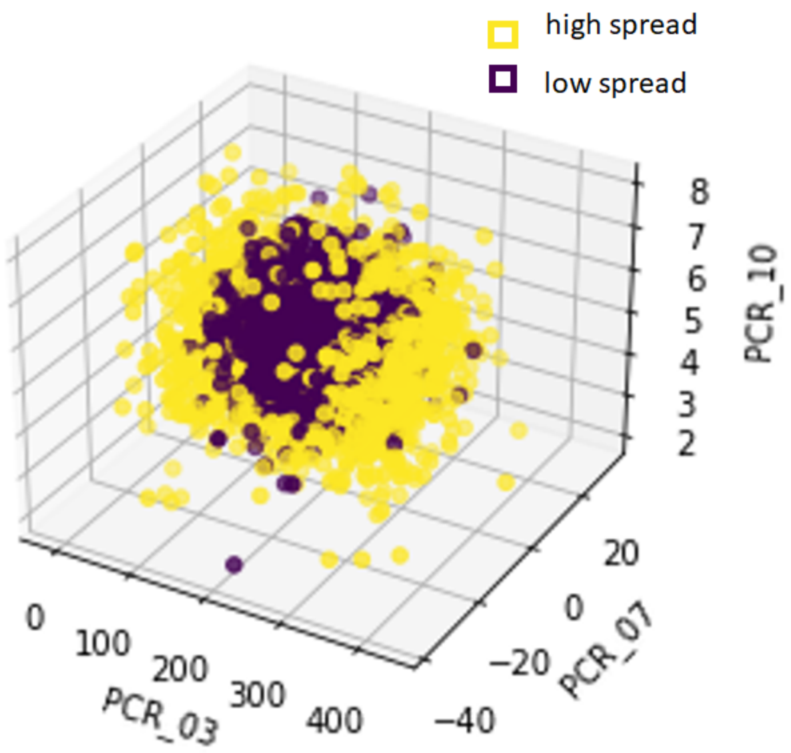


Figure 4: 3D Scatter plot of `PCR_03`, `PCR_07`, and `PCR_10` according to `spread`

Q4:

The points with label `-1` for `spread` are contained within an ellipsoid described by the following equation:

$$\frac{PCR\_03}{a^2} + \frac{PCR\_07}{b^2} + \frac{PCR\_10}{c^2} = 1$$

where a, b, c are positive real numbers.

Q5:

The training accuracy for $k = 11$ is 81.333%.

Q6:

Z-score scaling scales the data of a feature by ensuring that they have zero mean and unit standard deviation, thereby causing the data to adhere to a normal distribution. Features scaled according to this technique have their outliers handled correctly, but no guarantee on the resulting range of the data is made, and the ranges of different features scaled according to Z-score may differ from each other. This technique is preferable in cases that have outliers and when the learning model assumes that the data adheres to a normal distribution . On-the-other-hand, the min-max technique involves scaling the data of a feature to a specific range (generally between 0 and 1). Contrary to the Z-score method, this technique guarantees a uniform range across features and maintains the original distribution of the data, but does not handle outliers well. Therefore, it would be preferable to use this technique only when the feature in question has no significant outliers and_or the learning model to be used requires the feature data to fall within a certain range.

Q7:

The updated accuracy after normalizing the data is 88.5416%. This is an increase of approximately 7% over the accuracy from the non-normalized data. This can be explained by the fact that KNN is an algorithm that uses euclidean distances between data points for its predictions. Therefore, it is very sensitive to differences in scales between different features, and thus features that have undergone normalization, which equalizes or nearly equalizes the scales and ranges of different features, will do better in KNN because no single feature will dominate and have more influence over other features by way of having a larger scale.

Q8:

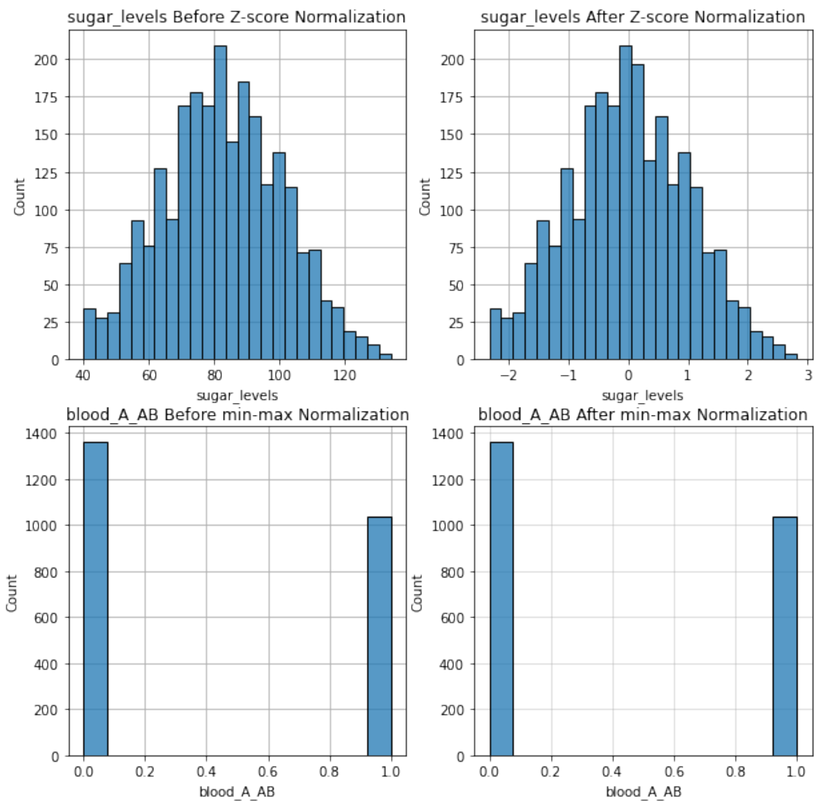


Figure 5: Histogram Plots of `sugar_levels` and `blood_A_AB` Before and After Normalization

`sugar_levels` We chose to do Z-score Normalization on `sugar_levels` since it already obeys an approximate normal distribution, as can be seen in figure 5, and therefore we would not be forcibly changing its distribution through the normalization process. Furthermore, we will not be using it in the KNN model, which works better with min-max normalized features, but rather with SGD SVM, which works better with Z-score normalized features, and a decision tree, which is mostly agnostic to Normalization.

`blood_A_AB` In contrast, we chose to do min-max normalization on `blood_A_AB`, since it does not obey a normal distribution, but rather already obeyed a binary distribution of 0 and 1, which we did not want to modify. Furthermore, it does not have outliers which could negatively affect the min-max normalization.

Q9:

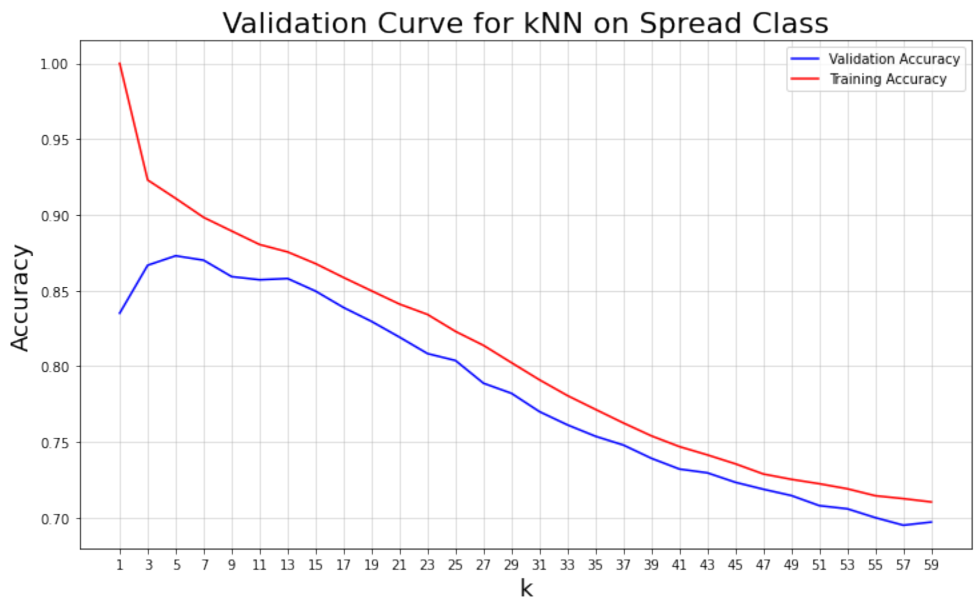
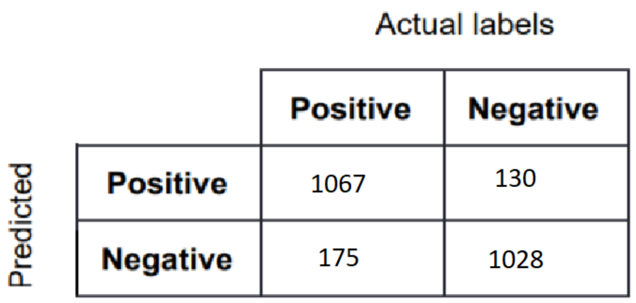


Figure 6: Validation Curve for kNN on spread Class

The optimal k value is 5, as can be seen by virtue of the fact that in figure 6, the highest point (peak) of the validation curve is at 5. The mean validation and training accuracies for $k = 5$ are 87.2916% and 91.083%, respectively.

Q10:



		Actual labels	
		Positive	**Negative**
Predicted	**Positive**	1067	130
	Negative	175	1028

Figure 7: Confusion Matrix for `spread` Class

According to the confusion matrix in figure 7, we can see that our model tends more to false negatives.

Part 2

Q11:

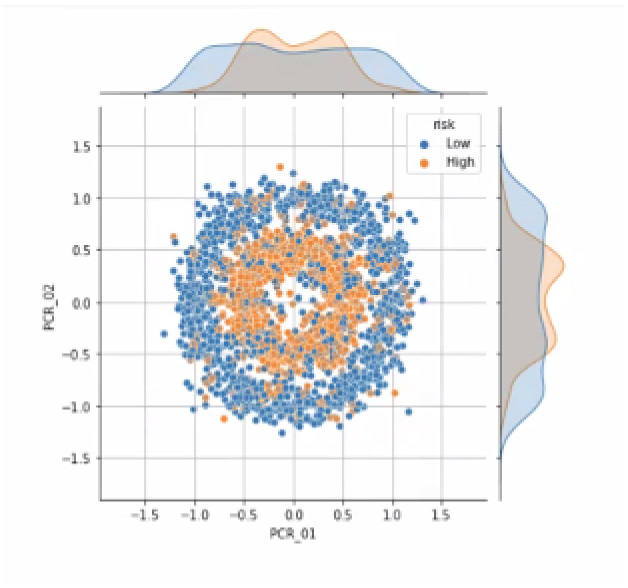


Figure 8: Joint plot of PCR_01 and PCR_02 according to risk

As can be seen from the scatter portion of the jointplot in figure 8, the plot is mostly separable into radiuses, and therefore it seems likely that `PCR_01` and `PCR_02` will be important in predicting the `risk` class.

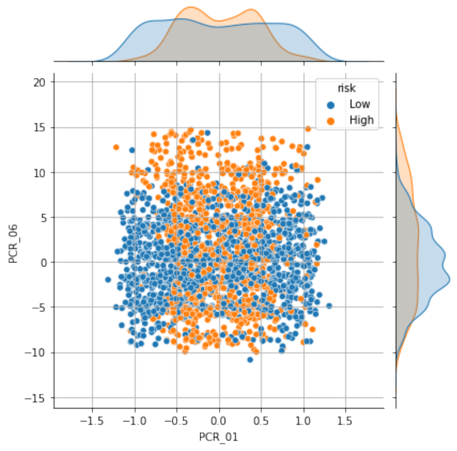
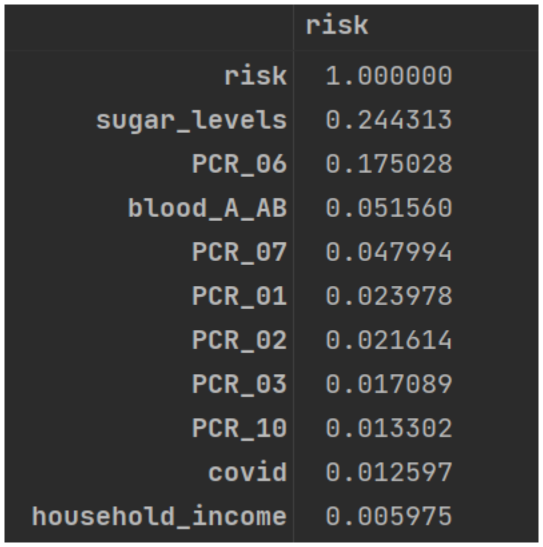


Figure 9: Joint plot of PCR_06 versus PCR_01 with respect to risk

Furthermore, from the scatter portion of the jointplot in figure 9 we noticed a mostly separable form similar to the letter "H" where the "H" itself is made up of a high proportion of points with low risk surrounded by clusters of points of high risk. Therefore, we can conclude that in addition, `PCR_06` will be important in predicting the `risk` class.

Q12:



```
                     risk
             risk    1.000000
     sugar_levels    0.244313
           PCR_06    0.175028
       blood_A_AB    0.051560
           PCR_07    0.047994
           PCR_01    0.023978
           PCR_02    0.021614
           PCR_03    0.017089
           PCR_10    0.013302
            covid    0.012597
 household_income    0.005975
```

Figure 10: The 10 most correlated features to `risk`

Q13:

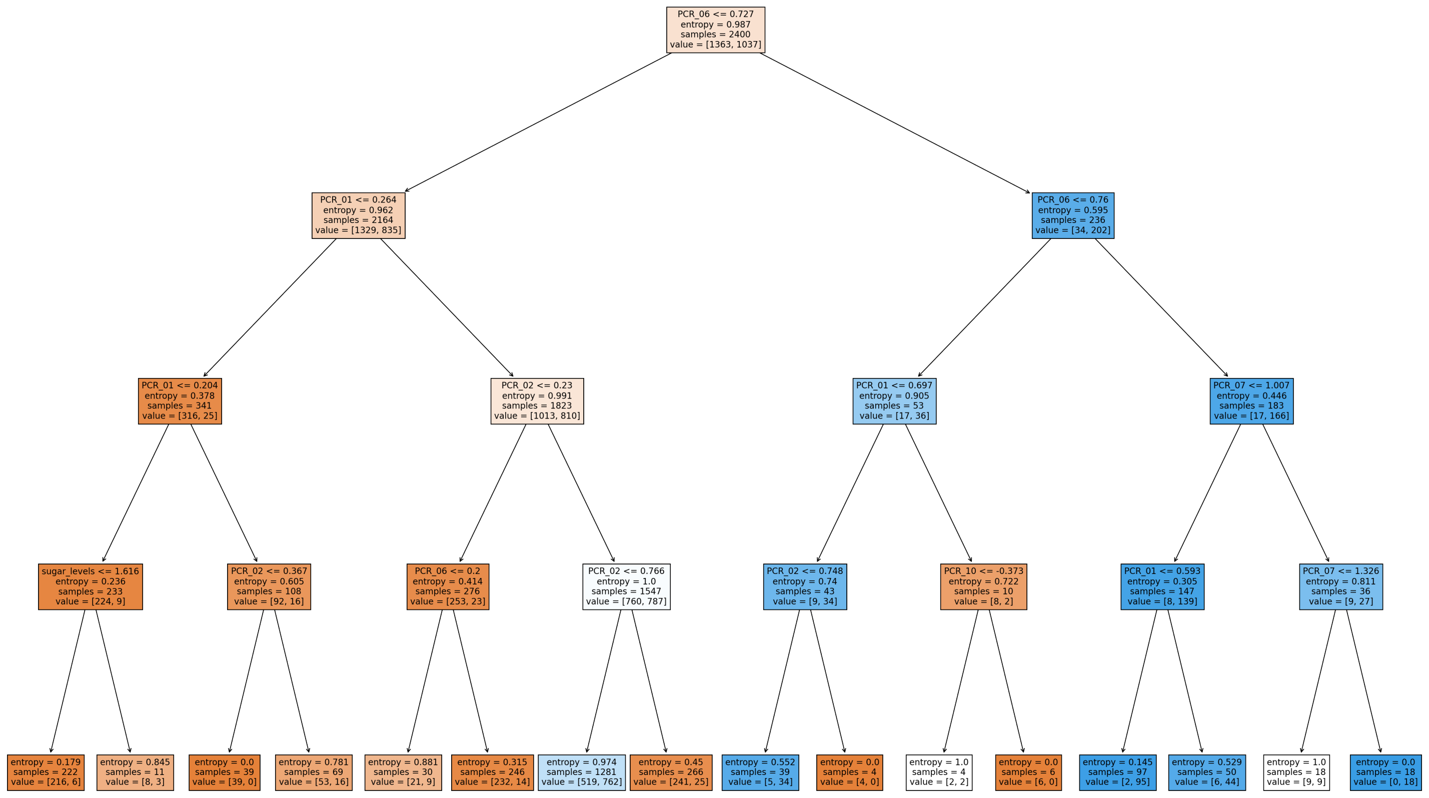

PCR_06 <= 0.727 entropy = 0.987 samples = 2400 value = [1363, 1037]
PCR_01 <= 0.264 entropy = 0.962 samples = 2164 value = [1329, 835]
PCR_06 <= 0.76 entropy = 0.595 samples = 236 value = [34, 202]
PCR_01 <= 0.204 entropy = 0.378 samples = 341 value = [316, 25]
PCR_02 <= 0.23 entropy = 0.991 samples = 1823 value = [1013, 810]
PCR_01 <= 0.697 entropy = 0.905 samples = 53 value = [17, 36]
PCR_07 <= 1.007 entropy = 0.446 samples = 183 value = [17, 166]
sugar_levels <= 1.616 entropy = 0.236 samples = 233 value = [224, 9]
PCR_02 <= 0.367 entropy = 0.605 samples = 108 value = [92, 16]
PCR_06 <= 0.2 entropy = 0.414 samples = 276 value = [253, 23]
PCR_02 <= 0.766 entropy = 1.0 samples = 1547 value = [760, 787]
PCR_02 <= 0.748 entropy = 0.74 samples = 43 value = [9, 34]
PCR_10 <= -0.373 entropy = 0.722 samples = 10 value = [8, 2]
PCR_01 <= 0.593 entropy = 0.305 samples = 147 value = [8, 139]
PCR_07 <= 1.326 entropy = 0.811 samples = 36 value = [9, 27]
entropy = 0.179 samples = 222 value = [216, 6]
entropy = 0.845 samples = 11 value = [8, 3]
entropy = 0.0 samples = 39 value = [39, 0]
entropy = 0.781 samples = 69 value = [53, 16]
entropy = 0.881 samples = 30 value = [21, 9]
entropy = 0.315 samples = 246 value = [232, 14]
entropy = 0.974 samples = 1281 value = [519, 762]
entropy = 0.45 samples = 266 value = [241, 25]
entropy = 0.552 samples = 39 value = [5, 34]
entropy = 0.0 samples = 4 value = [4, 0]
entropy = 1.0 samples = 4 value = [2, 2]
entropy = 0.0 samples = 6 value = [6, 0]
entropy = 0.145 samples = 97 value = [2, 95]
entropy = 0.529 samples = 50 value = [6, 44]
entropy = 1.0 samples = 18 value = [9, 9]
entropy = 0.0 samples = 18 value = [0, 18]

Figure 11: ID3 Decision Tree for Predicting `risk` CLass

The training accuracy of the model is 74.333%.

Q14:

PCR_01 and PCR_02, which we concluded would be important features, are the features with the fifth and sixth highest correlations with `risk`, as can be seen in figure 10, and their actual correlation values (0.023 and 0.021 respectively) are very small. It is therefore easy to see that that these features are very much not discernable as important features by merely analyzing their correlation to `risk`. Even PCR_06, which has the second highest correlation to `risk`, has a low correlation value of only 0.175, and is therefore also not easily discernable as an important feature by way of correlation alone.

All three of these features which we deemed important were used by ID3 in the decision tree. And in-fact, they were used in 11 out of 15 of the internal nodes of the tree, further proving their importance.

Q15:

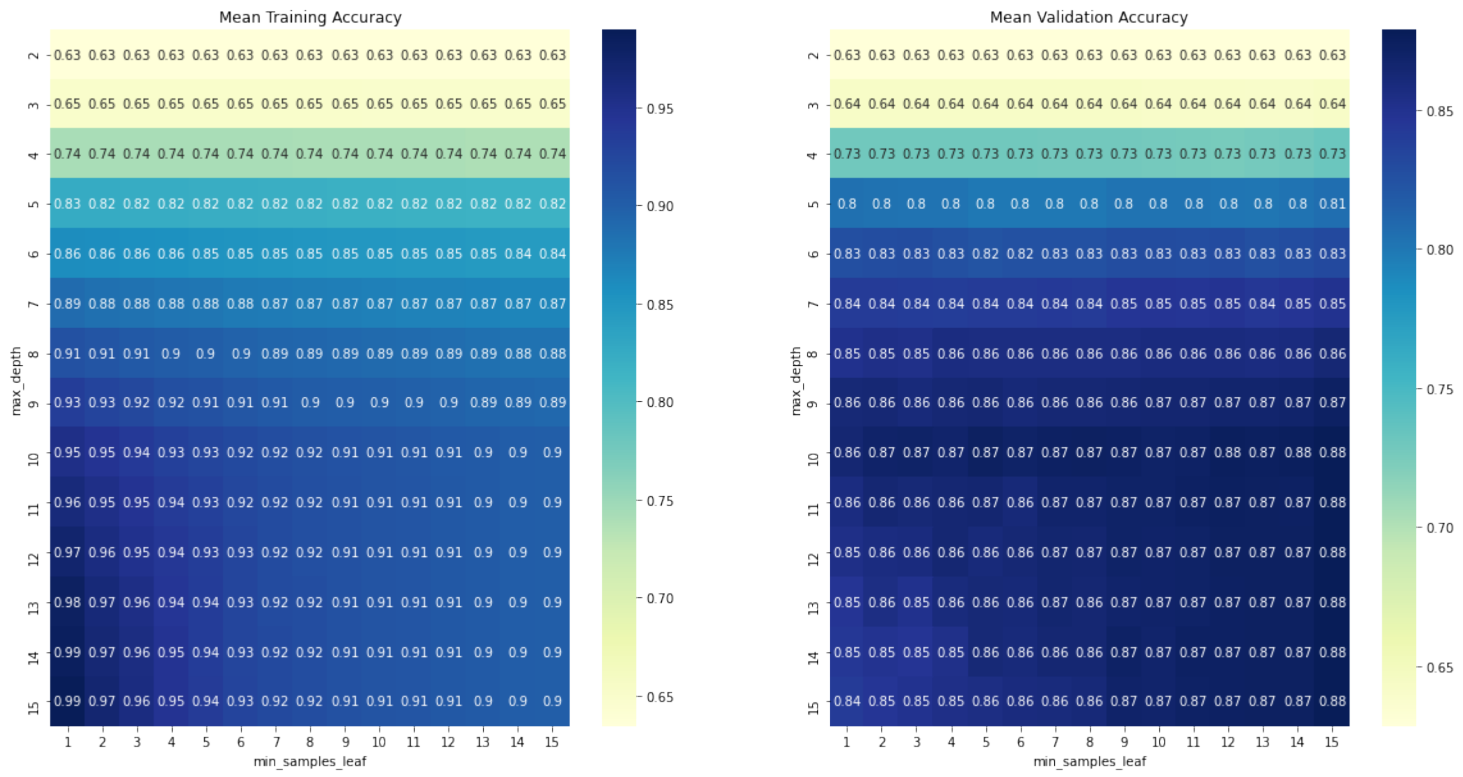


Figure 12: Heat Maps of Training and Validation Accuracies of ID3 on `risk` Class for Different Values of `max_depth` and `min_samples_leaf` Hyper-Parameters

The optimal hyperparameter combination is a `max_depth`=10 and `min_samples_leaf`=15, since they produce the highest validation accuracy of 88% with the minimal required depth and highest minimum sample leaf requirement, as can be seen in the mean validation accuracy heat map in figure 12.

The combination of `max_depth`=2 and `min_samples_leaf`=1 causes underfitting, as this leads to a low training accuracy of 63%.

The combination of `max_depth`=15 and `min_samples_leaf`=1 causes overfitting, as this leads to a high training accuracy of 99% versus a relatively low validation accuracy of 84%.

Part 3

Q16:

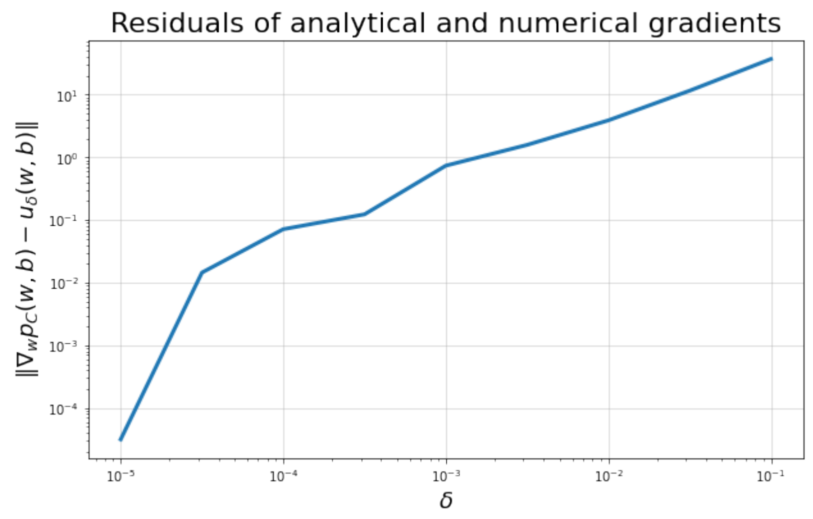


Figure 13: Plot of Residuals of analytical and numerical gradients

As we can see in figure 13, the difference between the analytic and numerical gradients increases in a monotonic fashion as the value of δ increases. This is logical, as δ is the differential size used in the definition of the numerical gradient, and therefore a smaller δ equates to a more precise estimation of the analytic gradient by the numerical gradient.

Q17:

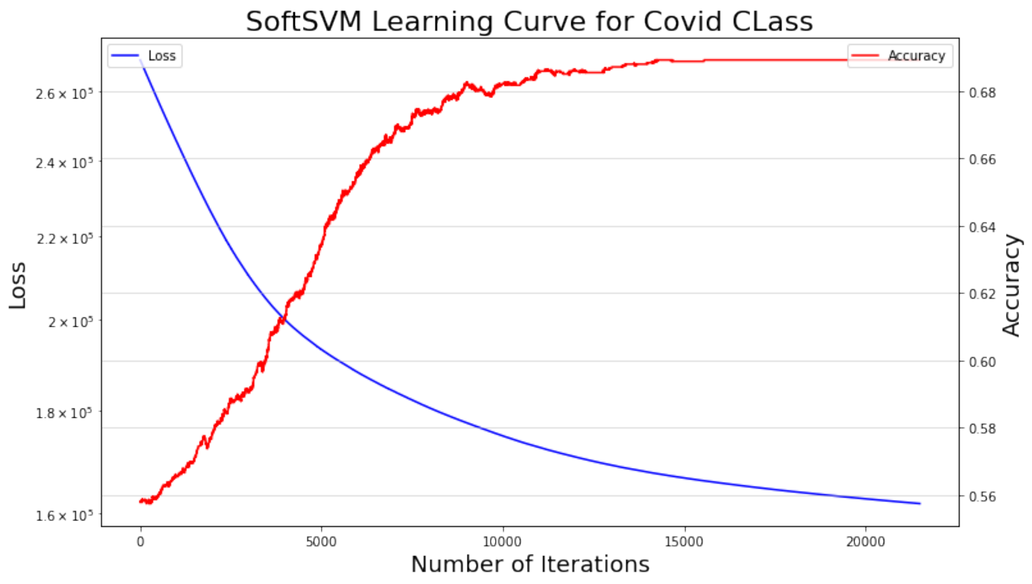


Figure 14: Learning Curve for Soft SVM Training on `virus` Class

The maximal training accuracy achieved by the model is 68.9%.

Q18:

As we saw in figure 4, the `spread` class is not linearly separable (only separable as an ellipsoid), and therefore our soft SVM model as it is without transformations would be a poor model because it relies on data that is close to linearly separable. Alternatively, the kNN model was suitable for the problem because it relies on clustering of similarly labeled data, which is the case in figure 4.

Q19:

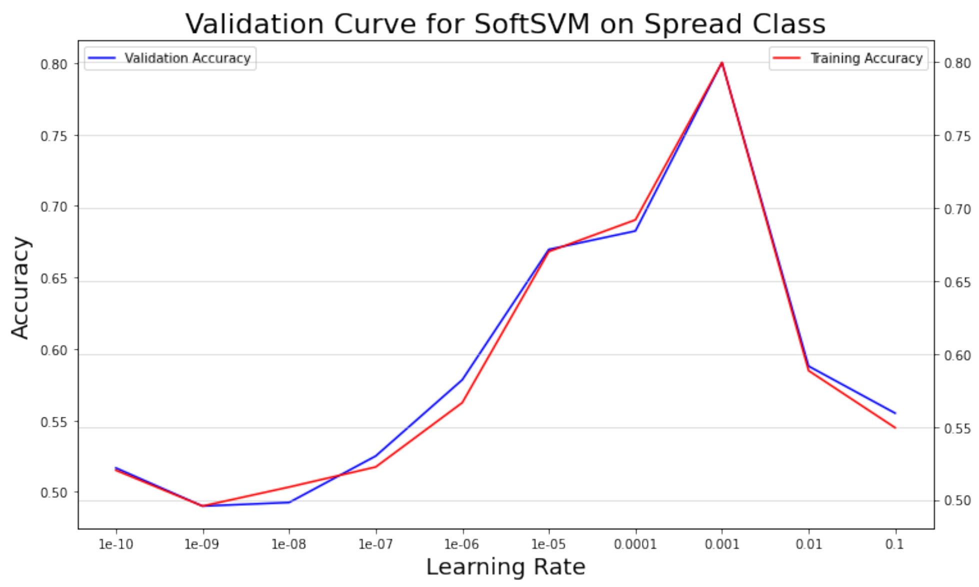


Figure 15: Validation Curve for Learning Rate on `spread Class`

First of all, from figure 15 we can conclude that the optimal learning rate for the SVM SGD is 0.001 as can be seen from the fact that a peak validation accuracy of 80% is achieved for that learning rate. With respect to the learning perspective, we can observe in the graph slight overfitting in the range of $[10^{-5}, 10^{-4}]$, since in this range, the training accuracy is above the validation accuracy. On the other hand, we can notice that there is slight underfitting in the range of $[10^{-7.5}, 10^{-5}]$ as in this range the training accuracy is below the validation accuracy.

With regards to the optimization perspective, we can see that decreasing the learning rate (step size) only increases the accuracy up to a certain point. Decreasing it past the optimal rate we found leads to a decrease in accuracy, as clearly seen in figure 15. This is probably due to the fact that the learning rate is too slow, and therefore on every iteration the SGD is taking a step that is too small towards the minima, and therefore the SGD is not able to reach the minima in the number of iterations provided (2000).

Q20:

In Q19, we used a 2 dimensional polynomial transform on the features within the primal objective of the SVM model. This approach is slower than the two dimensional kernel objective because it involves computing inner products between data points and the w vector in high dimensions, whereas in the kernel case, the dual objective is used, which only involves computing the kernel function for 2 dimensional polynomials which is of much lower dimension (and finding and storing the αs, which is not too computationally burdensome on the assumption that the support vectors are sparse).

Q21:

For the SVM model after training, we are required to keep only a single w vector of dimension 10, and therefore the space requirement for SVM after training is $O(1)$. On the other hand, for kNN, we must maintain all of the data points and all of their labels for prediction, therefore requiring us to keep m vectors of dimension equal to three and additionally a vector of size m of their associated labels, where m is the size of the input. Therefore in total, the kNN model requires $O(m)$ space after training.