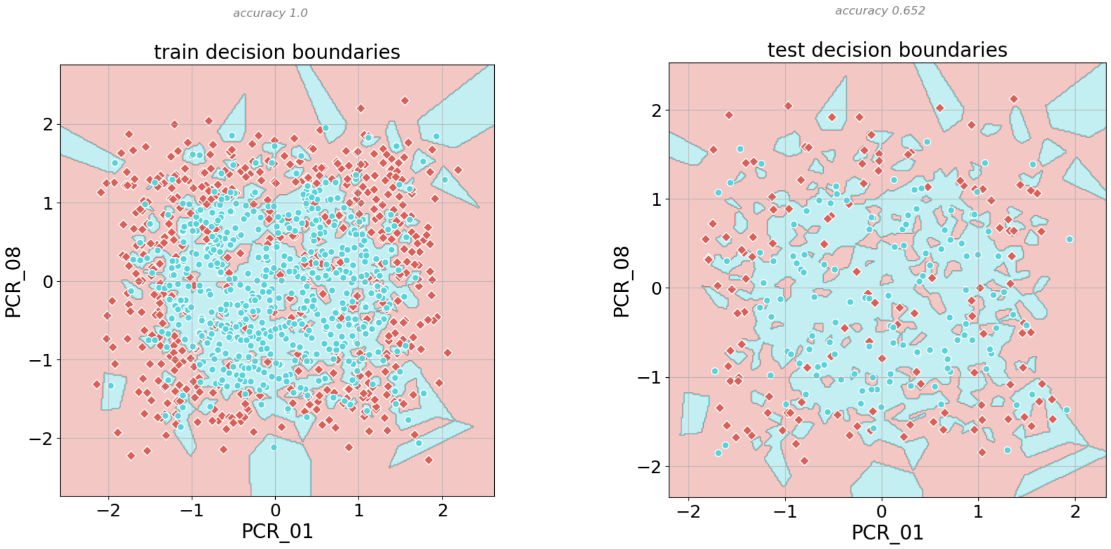
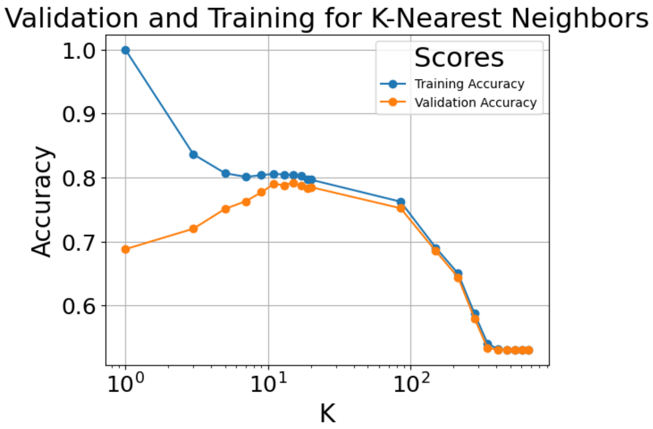
Part 1: Basic model selection with k-Nearest Neighbors

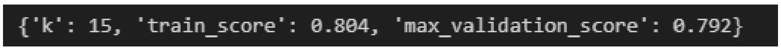
1.



2.



The best score is:

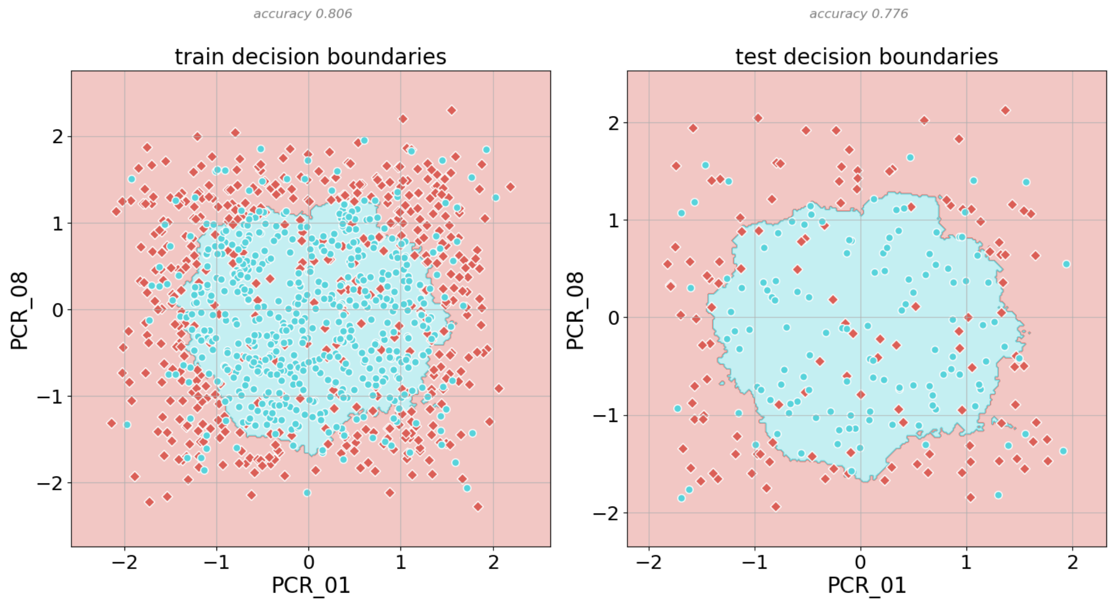


```
{'k': 15, 'train_score': 0.804, 'max_validation_score': 0.792}
```

The model is overfitting when the model scores significantly better on the training data compared to validation (unable to generalize). It happens when k ~ 1,3,5 probably because the small neighbor size mandates more sensitivity to noise and thus it's harder for the model to capture the true distribution.

The model is underfitting when it seems it's not able to learn from the data leading to low train and validation scores. It happens when we have large K's such as 150 + probably because when considering these many points the predictions show low variance and are not very useful to predict a single point.

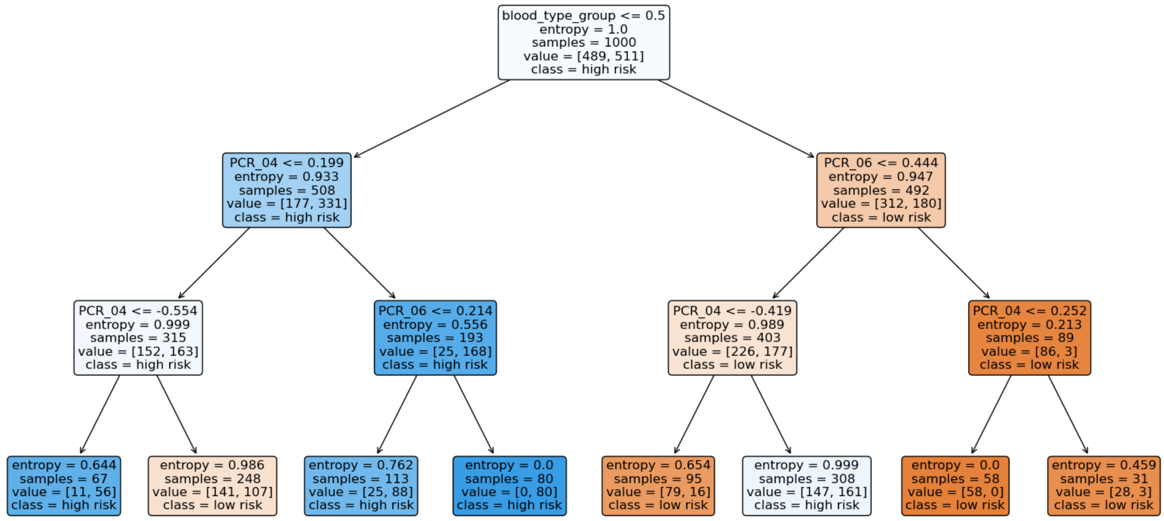
3.



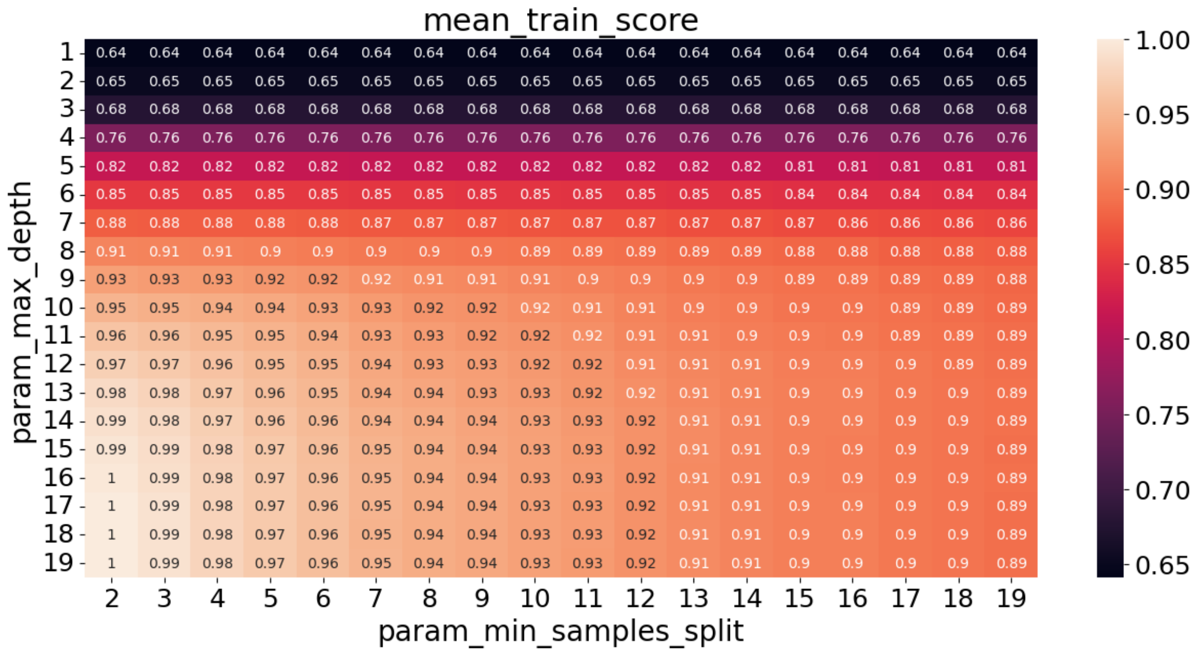
4. Compared to K=1 we can see that the decision boundary is less dotted. This means that when K=15 the model is less sensitive to noise and seem to better predict the true distribution better.

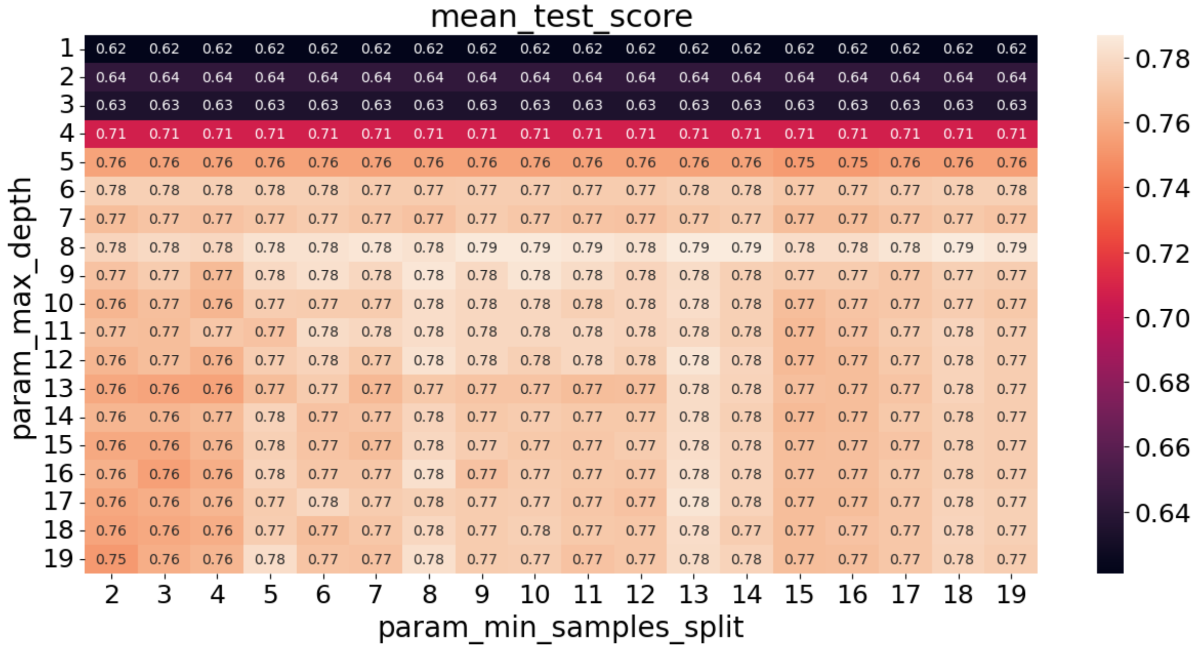
Part 2: Decision trees

5.

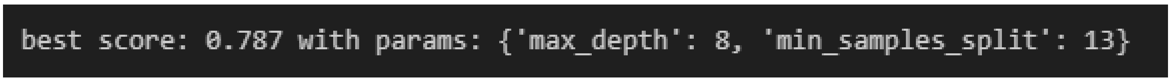


6.





c. (one among) the best combination is:



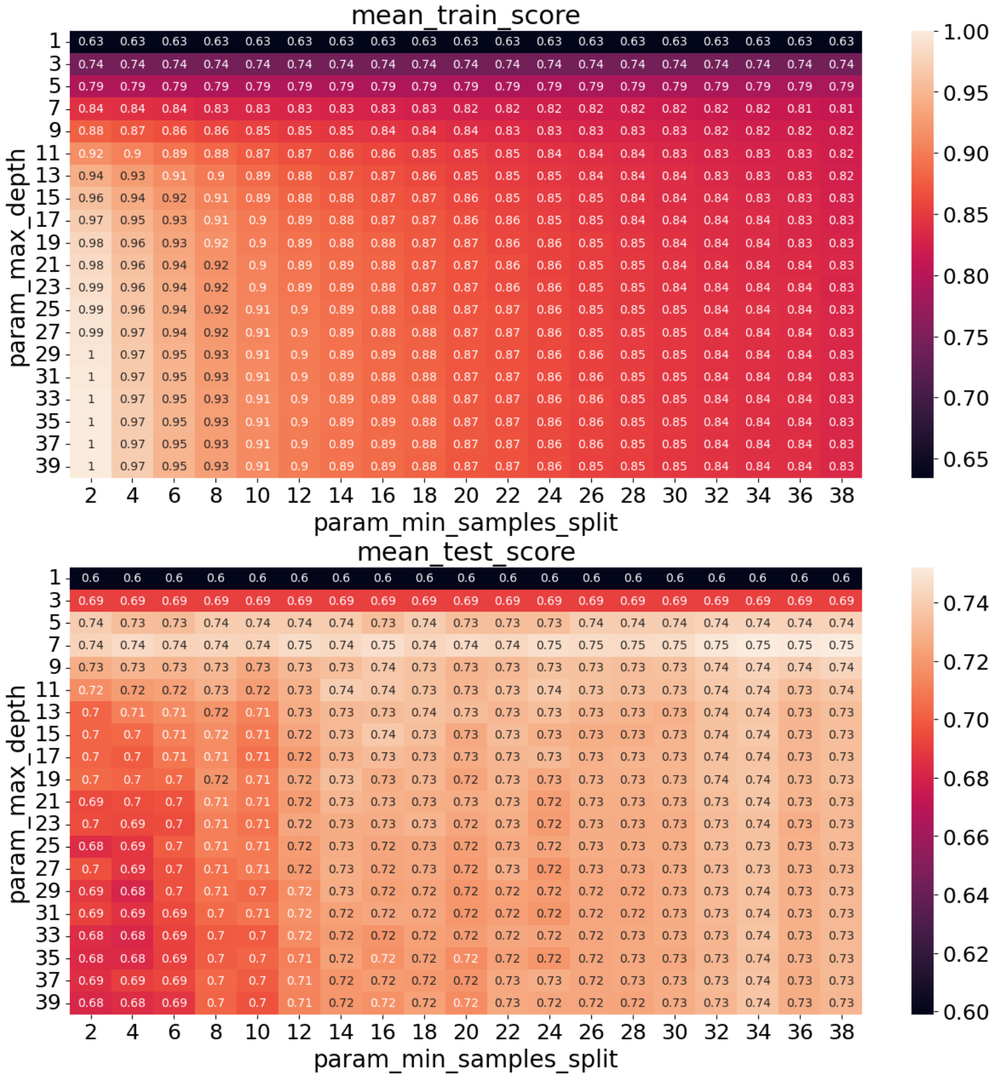
```
best score: 0.787 with params: {'max_depth': 8, 'min_samples_split': 13}
```

d. a combination that causes under-fitting is max_depth= 1 and min_sample_split = 2

e. a combination that causes over-fitting is max_depth= 19 and min_sample_split = 2

f. the first combination causes underfitting because a tree of depth 1 doesn't have enough expressive power to actually learn something and make case specific predictions. The second combination has too much power and can create exact cases to fit the training data perfectly but in the process losses generalization.

We don’t know if the overfitting is considered noticeable enough so we’re adding another graph to better showcase this (same idea just more noticeable)–

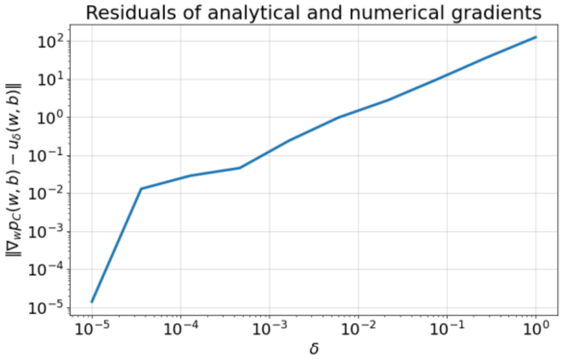


7. we tried every combination 5 times (for each fold as validation set) so $5 * (19 \times 18) = 1710$.

In CV we need to try all the possible combinations so number of times will be the cross product of the sets time the folds. If we were to add a third parameter we’ll get a lot more combinations to check.

8. the test score with the best parameters is 0.76

9.



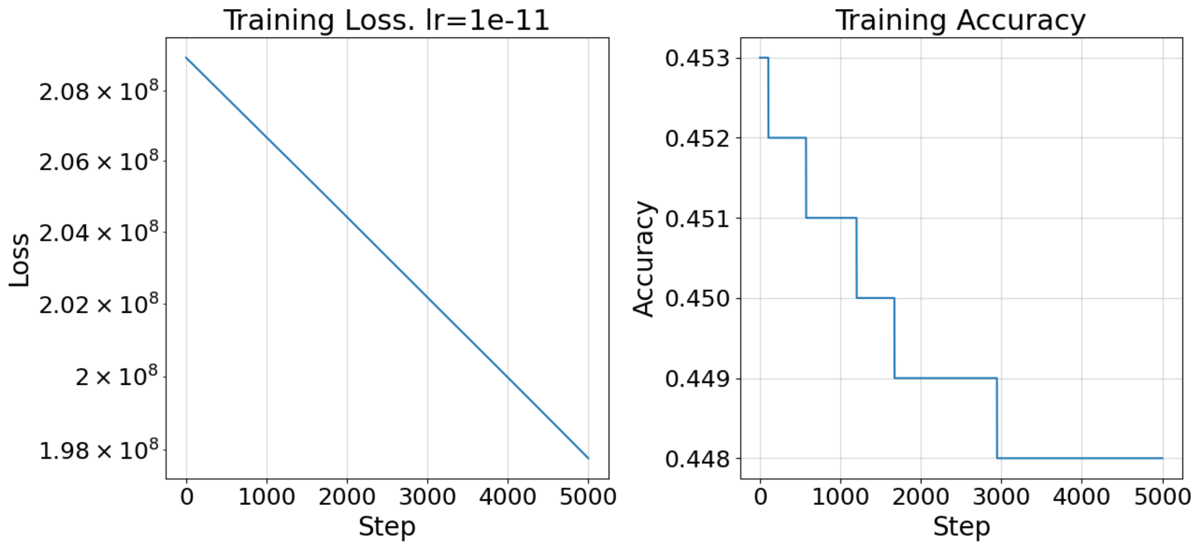
The graph makes sense because as $\delta \to 0$ the numerical gradient becomes the analytical gradient (as known from calculus) so we see the values getting closer to each other.

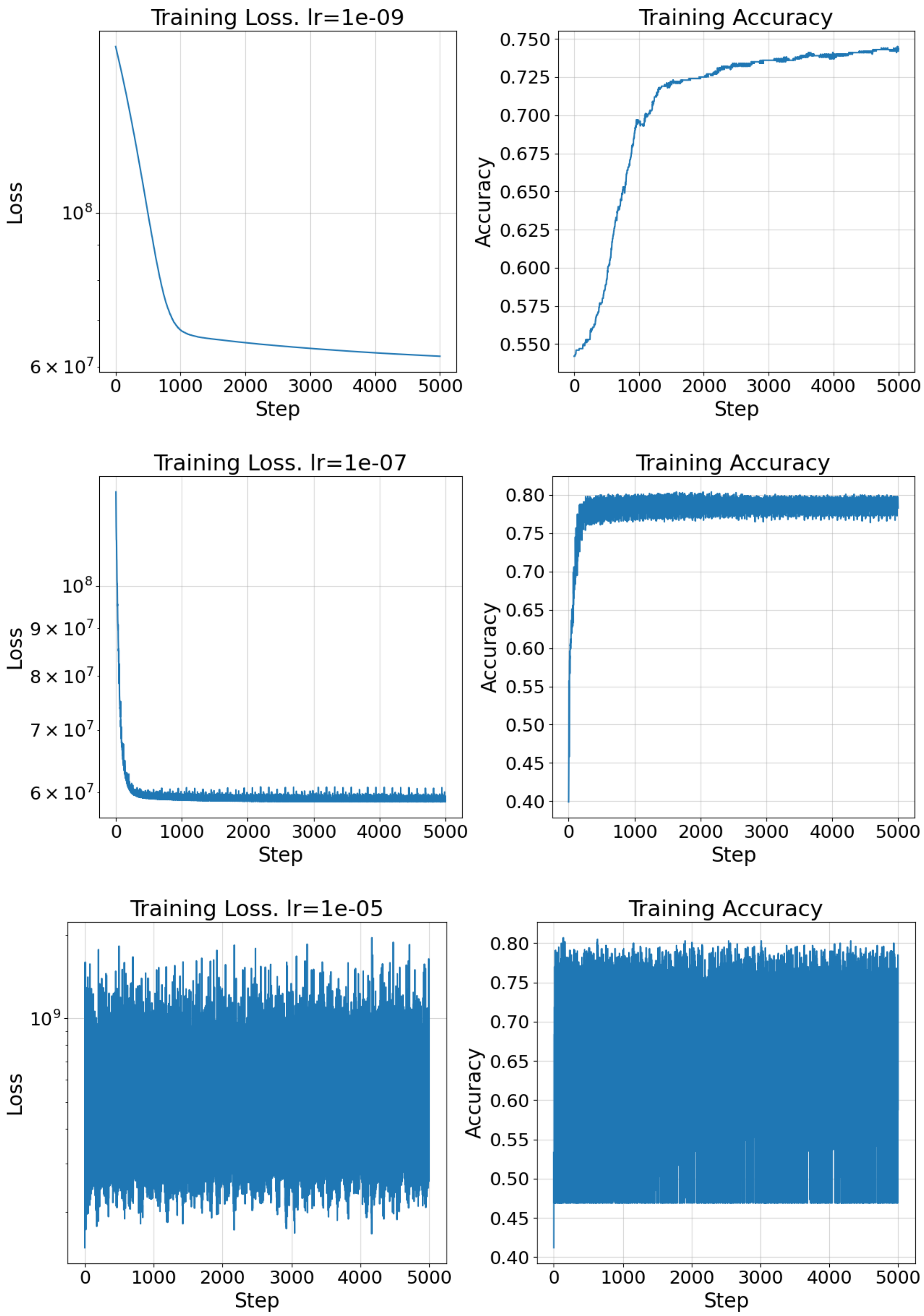
10. the train loss goes down as expected, meaning our optimization method works.

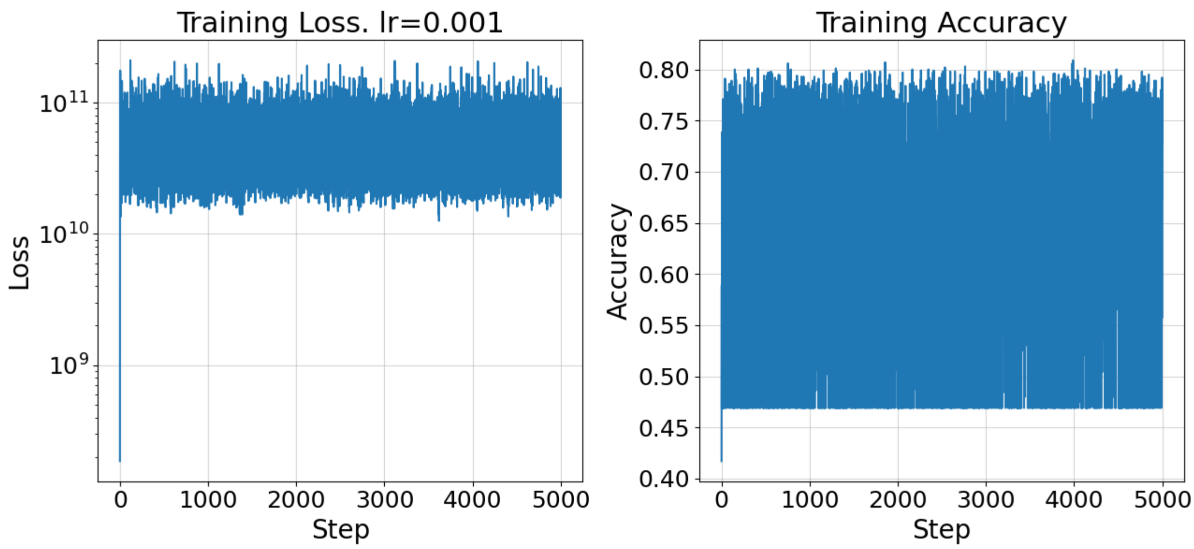
For the accuracy we will notice that the learning rate is very low and C is very high, meaning the model put emphasis on minimizing hinge loss which we can see doesn't mean accuracy (overfitting). All these fluctuations are caused by the emphasis on hinge-loss while the learning rate ensures we converge (and don't miss) to the minima in the loss plain.

11.

Plots:

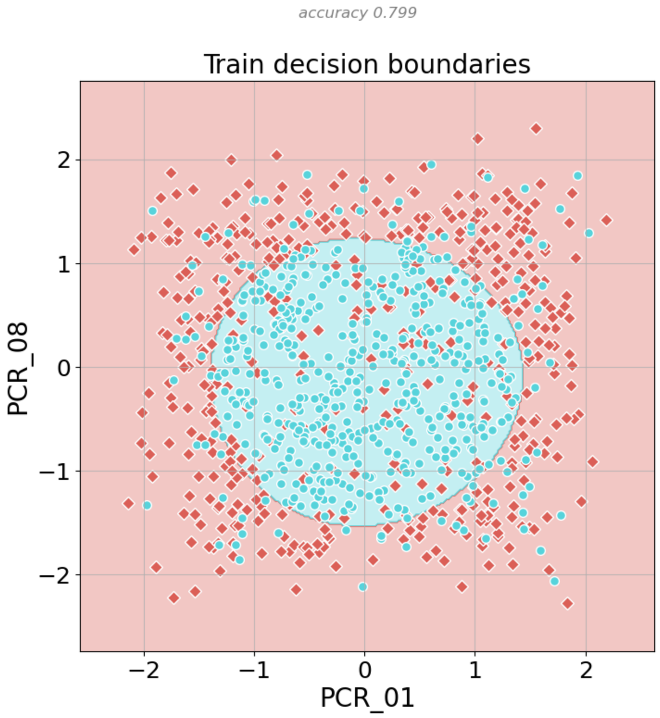


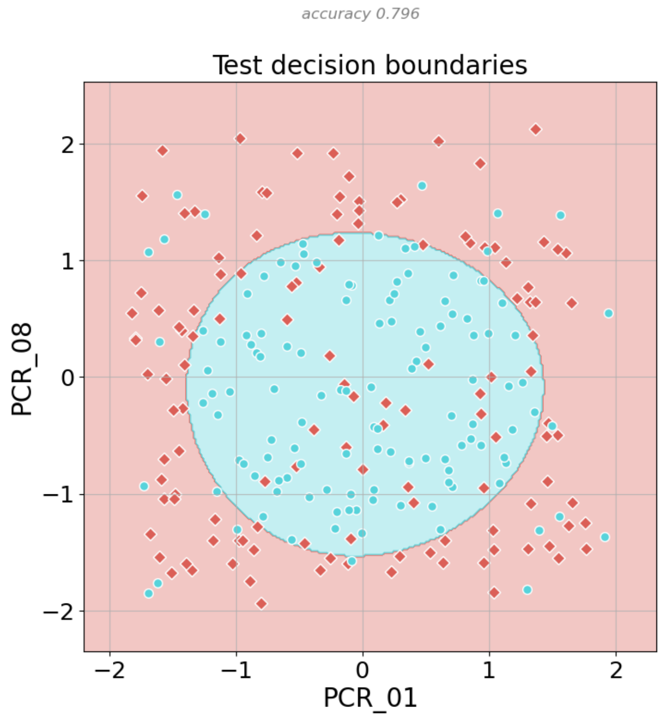

Training Loss. lr=1e-09
Loss
10^8
6×10^7
0
1000
2000
3000
4000
5000
Step
Training Accuracy
Accuracy
0.750
0.725
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0
1000
2000
3000
4000
5000
Step
Training Loss. lr=1e-07
Loss
10^8
9×10^7
8×10^7
7×10^7
6×10^7
0
1000
2000
3000
4000
5000
Step
Training Accuracy
Accuracy
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
1000
2000
3000
4000
5000
Step
Training Loss. lr=1e-05
Loss
10^9
0
1000
2000
3000
4000
5000
Step
Training Accuracy
Accuracy
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
1000
2000
3000
4000
5000
Step



We would choose 1e-9 as the learning rate because as it converges to the minima in a fairly stable way (in contrast to 1e-7) and maybe with more iterations would even surpass 1e-7.

12.





13.

a. $\varphi(S_i)$ is a vector of length d where $\varphi(S_i)_k$ is 1 iff S_i has the word k. When performing the dot product of two vectors we are multiplying each location k in the two vectors, which result of that is 1 iff both vectors had 1 in the location k, and then summing over the ones that we got. The result of that will be the total number of word both sets have in common.

this kernel is defined because upholding- $K_t = \varphi(S_i)^T\varphi(S_j)$ $(= |S_i \cap S_j|)$

b. $K_{spam}(S_i, S_j) = e^{|S_i \cap S_j|} = e^{K_t}$

so it's defined from kernel rules 1 and the previous result.

c. let denote $f(S_i) = e^{-||\varphi(S_i)||_1}$ then

$$k_b = e^{-|S_1 \cup S_2|} = e^{-|S_1|-|S_2|+|S_1 \cap S_2|} = e^{-|S_1|} * e^{|S_1 \cap S_2|} * e^{-|S_2|} = f(S_1) * K_{spam} * f(S_2)$$

Valid kernel from rule 2 and previous result.

d. same as in the dry hw.

if $k_1 = \varphi_1(x)^T\varphi_1(x')$ is a valid kernel and $k_2 = \varphi_2(x)^T\varphi_2(x')$ then

$$\begin{aligned} k_1(x,x') * k_2(x,x') &= \varphi_1(x)^T\varphi_1(x') * \varphi_2(x)^T\varphi_2(x') \\ &= \sum\nolimits_i \varphi_1^i(x) * \varphi_1^i(x') \sum\nolimits_j \varphi_2^j(x) * \varphi_2^j(x') \\ &= \sum\nolimits_{i,j} \varphi_1^i(x)\,\varphi_2^j(x)\varphi_1^i(x')\varphi_2^j(x') \end{aligned}$$

Now we will define $\varphi^k(y) = \varphi_1^i(y) * \varphi_2^j(y)$ so

$$k(x,x') = k_1(x,x') * k_2(x,x') = \sum\nolimits_k \varphi^k(x) * \varphi^k(x') = \varphi(x)^T * \varphi(x')$$

Showing that k is a valid kernel.

e.

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

Is a valid kernel from d.

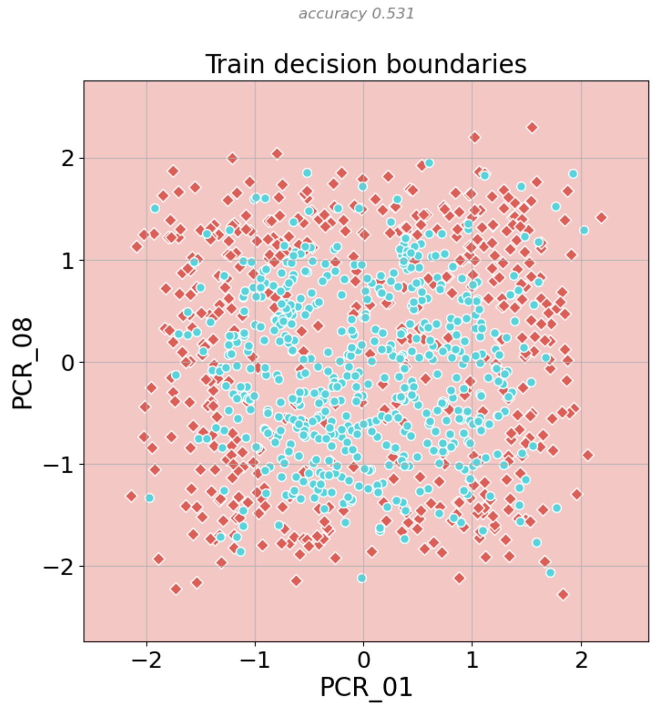
f. we saw in the blogpost that in RBF the numerator can be thought of as an equivalent to similarity. If we are trying to find a good measure for similarity of sets a good start would look at the number of words that can be found in both sets, but a better way would be also considering the portions those words make from all the words in the sets.

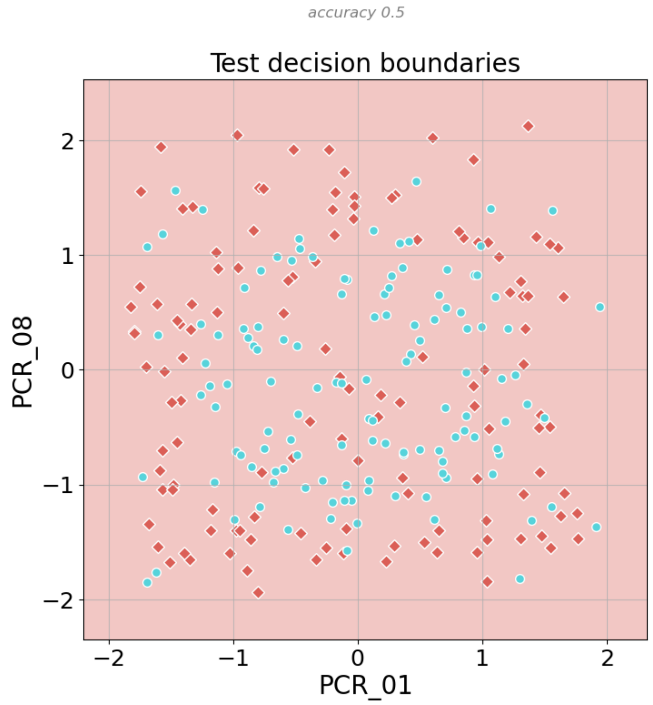
g. we can throw in more measures for similarity like length checking.

Say $I_l = \begin{cases} 0, & |S_i| \neq |S_j| \\ -1, & |S_i| = |S_j| \end{cases}$

And $K = e^{|S_i \cap S_j| - |S_i \cup S_j| + I_l}$

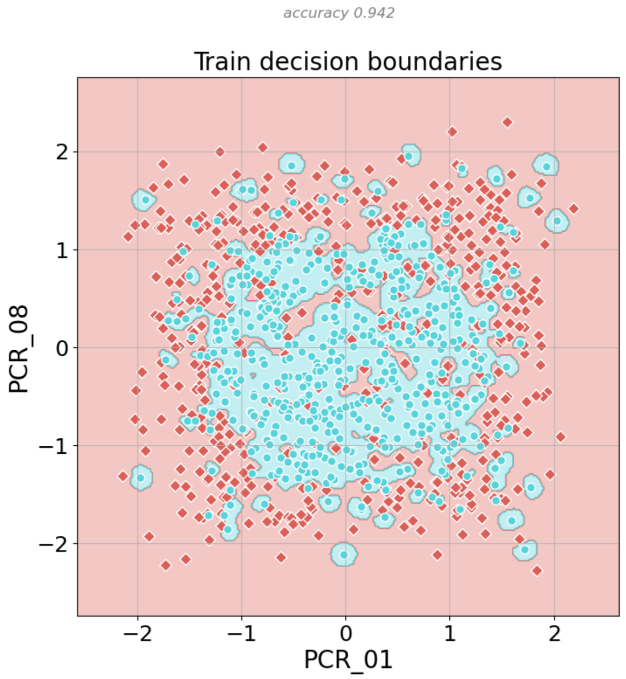
14.

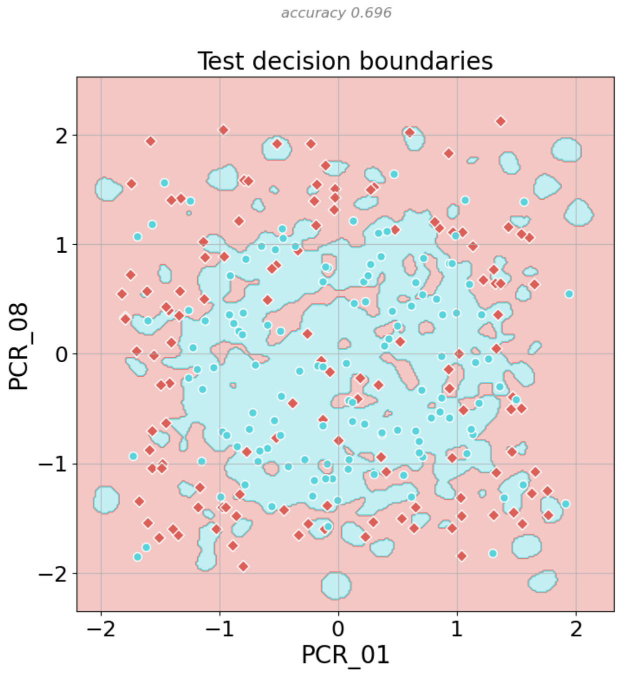




The model is under-fitting as it seems it doesn't learn much hence the low train and test scores. If we're thinking of the Knn analogy than the classification of a point takes into consideration alot of train point thus we always get the same classification (the most prominent classification in the training set).

15.

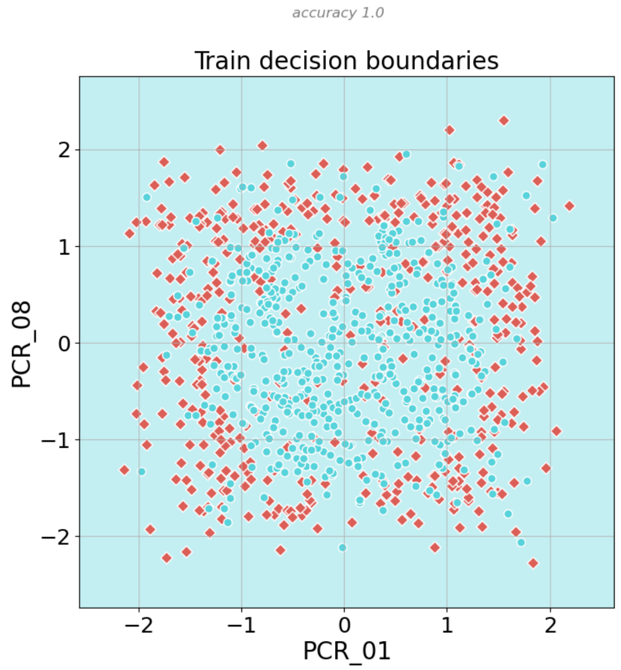


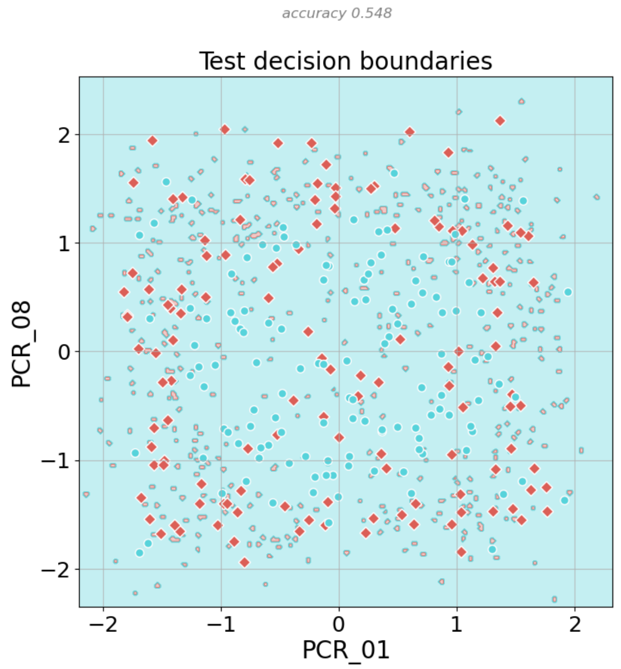


$\gamma = 200$ seems a close simulation to Knn with k=1. At Q3 we found that the optimal k is 15 which created smoother decision lines.

Considering the hint maybe the difference can be caused by the numerical instability when the exponent of e is a minus big number. The result will be very close to 0 and thus the farther points with $\alpha > 0$ won't be considered properly when calculating the weighted sum of the dual problem, resulting in only the closer points have an effect.

16.





The model is overfitting. The decision boundaries depend solely on the training data like in Knn when k=1.