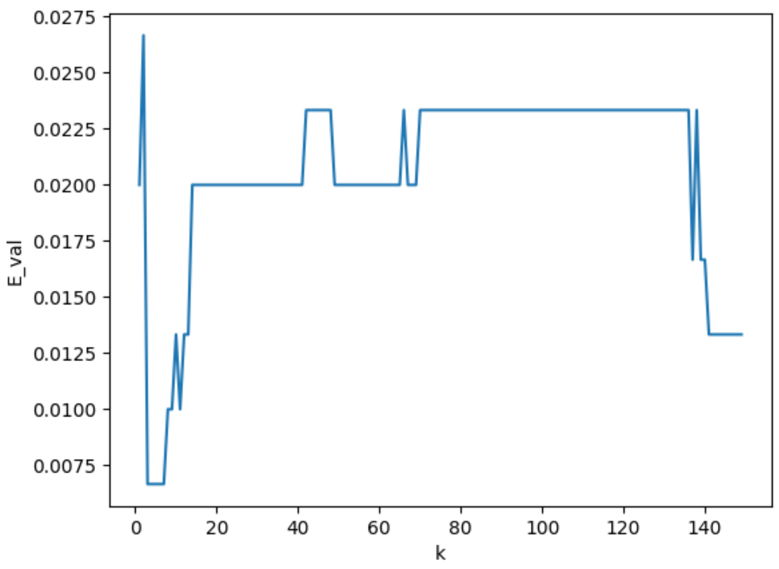
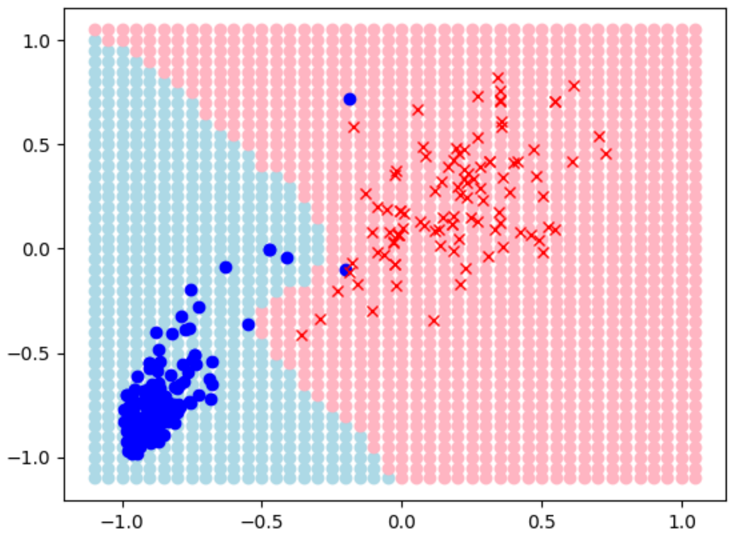
1. (a) $k = 3$ was chosen as the optimal value for k, the graph is shown:



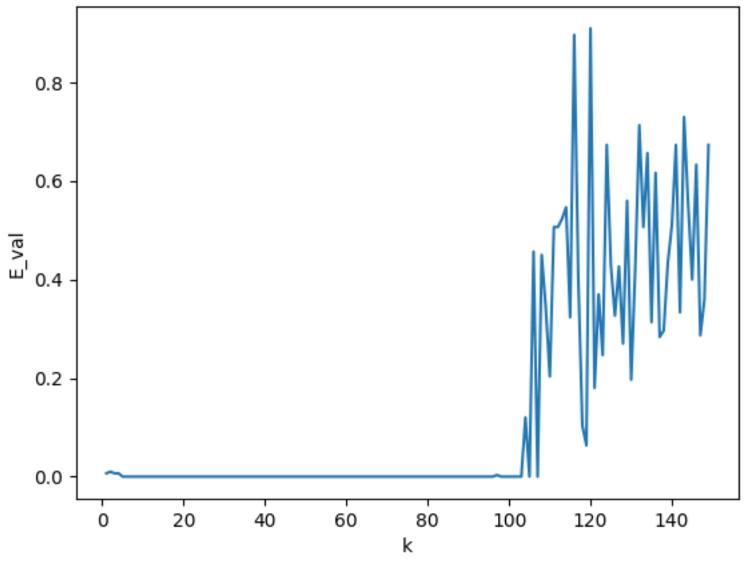
   (b) For $k = 3$, the decision boundary was:



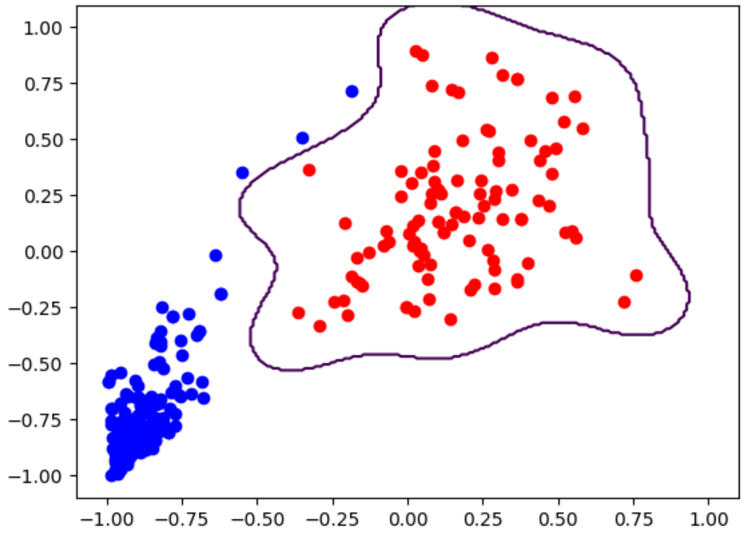
   My code uses in-sample error to choose the optimal k, so in sample and cross validation error were 0.0067

   (c) 0.0081, test error was done by checking all the points in the validation set and comparing their output to what would've been the output in the training set.

2. (a) The k chosen for my algorithm was $k = 2$:



(b) The in sample and cross validation errors were at 0.00, surprisingly low for my algorithm.



However the data does look overfitted, which may have affected the result.

(c) When tested on the validation set, the error was high at 0.33929, which is high and proves the overfitting problem.

3. (i) linear with 8th order: 0.031

(ii) k-NN: 0.0081

(iii) RBF: 0.33929

I believe that the way I did my RBF algorithm has small errors, after reading the book, all algorithms should relatively achieve the same performance, while the case is true for linear and k-NN, RBF does not. This makes sense since k-NN doesn't seem to use any sort of "regularizer", hence the decision boundaries for the k_{NN} seem more "overfitted" even though it isn't.The linear model has a smoother curve and it seems like it bases itself better than the k-NN algorithm.
We can visualize RBFs as mountains, where the nearer the neighbor is to the point, the greater weight it has on the output of the test point, RBF considers all the points, so theoretically, if RBF did work as intended, it should have the lowest E_{out} out of all the algorithms since it becomes like a glorified version of the linear model.