**Matthew Hess**

**Red-Id: 818299658**

**4/20/18**

**Assignment 6: C Support Vector Machine Classification**

The objective is to classify the gender of a crab from its physical measurements.

The six physical features are species, front-allip, rear-width, length, width, and depth.

The data set contains 200 samples each with gender labels (1 : male, -1 : female).

I noticed that through trying to find the best ‘C’ and ‘kernel’ hyper-parameters I had to make sure that the data subsets were not changing each time I clicked run since I was manually picking up the hyper-parameters based on my results.

**Initial Splitting of Data for most optimal Hyper-Parameter Selection:**

I thought it would be appropriate to showcase my separation of the data while I worked on the optimization.

**S = [**75, 55, 70, 152, 90, 14, 198, 108, 76, 83, 153, 79, 120, 99, 166, 77, 186, 0, 36, 179, 197, 184, 133, 132, 180, 57, 164, 170, 139, 159, 188, 38, 74, 64, 103, 37, 59, 97, 13, 134, 167, 56, 181, 60, 80, 28, 31, 142, 21, 16, 154, 187, 185, 129, 141, 18, 172, 162, 88, 68, 48, 63, 150, 127, 183, 6, 161, 17, 9, 157, 192, 27, 71, 116, 43, 148, 145, 22, 81, 102, 93, 189, 67, 45, 115, 4, 113, 51, 91, 24, 69, 33, 168, 12, 87, 112, 85, 94, 25, 23, 169, 41, 123, 114, 42, 105, 30, 49, 140, 106, 193, 107, 65, 11, 147, 39, 137, 15, 163, 165, 136, 98, 158, 84, 62, 92, 44, 160, 176, 66, 174, 101, 95, 26, 86, 32, 29, 7, 54, 128, 177, 191, 131, 10, 110, 173, 119, 40, 104, 122, 53, 20, 47, 50, 46, 125, 196, 126, 117, 1, 135, 118, 109, 96, 182, 34, 78, 124, 73, 149, 175, 35, 190, 155, 3, 195, 194, 121, 100, 72, 58, 19, 130, 138, 89, 111, 8, 2, 5, 178, 199, 156, 144, 171, 82, 61, 143, 151, 52, 146**]**

**#100 training samples**

**Xtr = data[S[:100], :6]**

**Ytr = data[S[:100], 6:]**

**# 100 testing samples**

**X\_test = data[S[100:], :6]**

**Y\_test = data[S[100:], 6:].ravel()**

**S2 = [**52, 53, 16, 75, 60, 57, 85, 82, 84, 11, 43, 51, 12, 49, 90, 37, 88, 39, 63, 86, 25, 83, 22, 32, 72, 95, 46, 38, 2, 35, 76, 58, 74, 48, 59, 64, 18, 79, 45, 14, 15, 44, 9, 0, 54, 80, 31, 5, 6, 1, 69, 73, 87, 98, 4, 96, 99, 71, 92, 36, 50, 41, 19, 67, 97, 23, 61, 24, 34, 42, 33, 62, 28, 78, 13, 68, 56, 30, 70, 81, 91, 10, 94, 8, 40, 93, 65, 3, 21, 20, 26, 77, 29, 89, 17, 66, 27, 47, 55, 7**]**

**# subsets for training models**

**x\_train = Xtr[S2[:50], :6]**

**y\_train = Ytr[S2[:50]]**

**# subsets for validation**

**x\_validation = Xtr[S2[50:], :6]**

**y\_validation = Ytr[S2[50:]**

**Hyper-Parameter C Selection:**

**Trial 1:**

Manually Variable Values:

C=0.01, error: 0.30000000000000004

C=0.015, error: 0.30000000000000004

C=0.02, error: 0.24

C=0.025, error: 0.19999999999999996

C=0.4, error: 0.040000000000000036

C=0.6, error: 0.06000000000000005

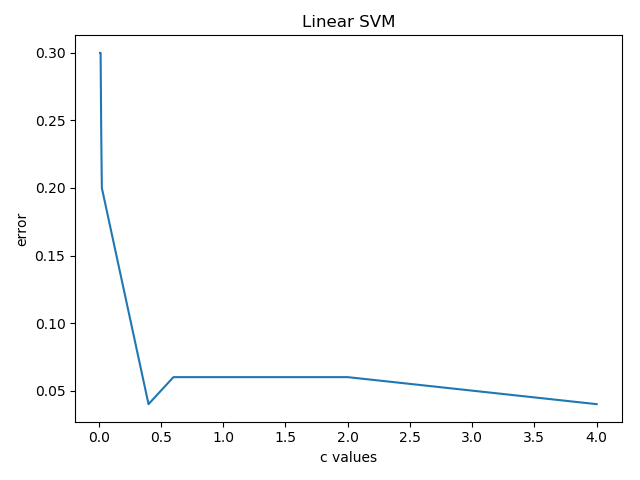
C=0.8, error: 0.06000000000000005

C=1, error: 0.06000000000000005

C=1.5, error: 0.06000000000000005

C=2, error: 0.06000000000000005

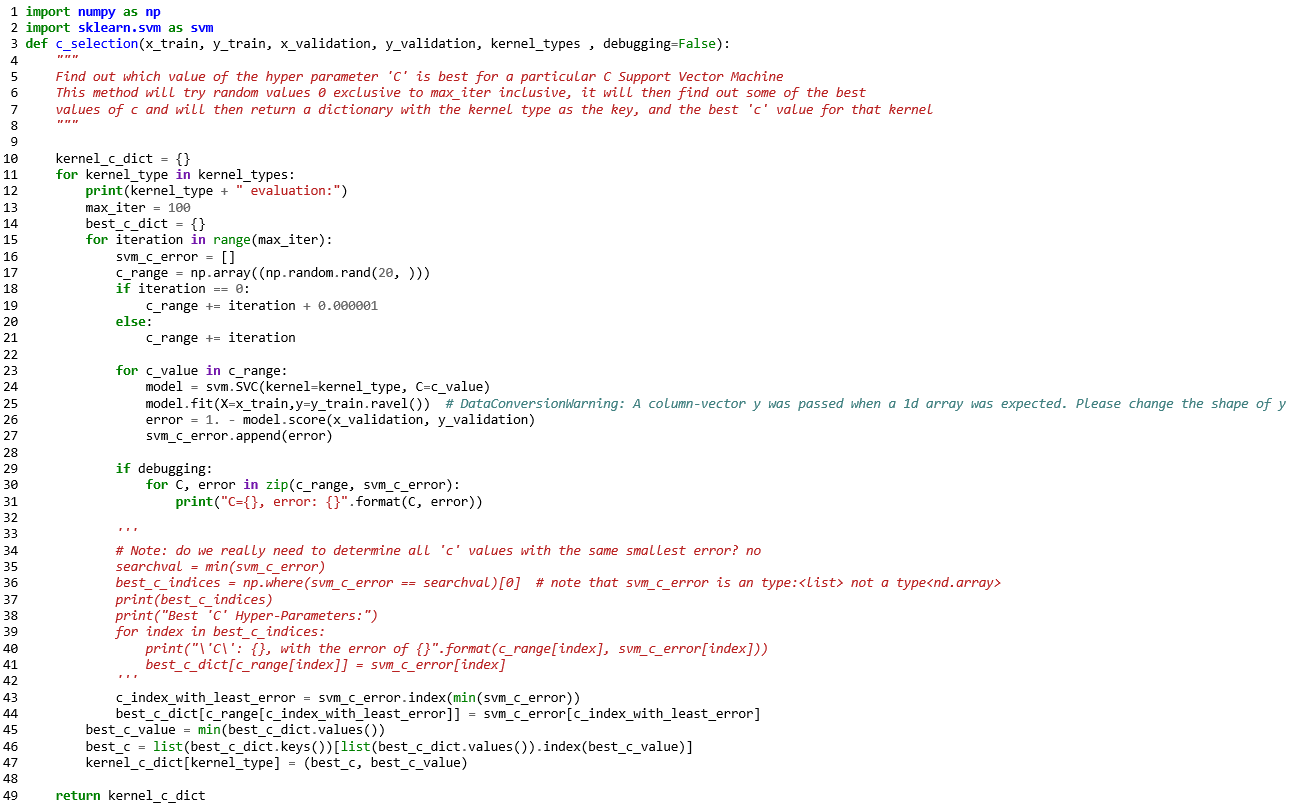
C=4, error: 0.040000000000000036



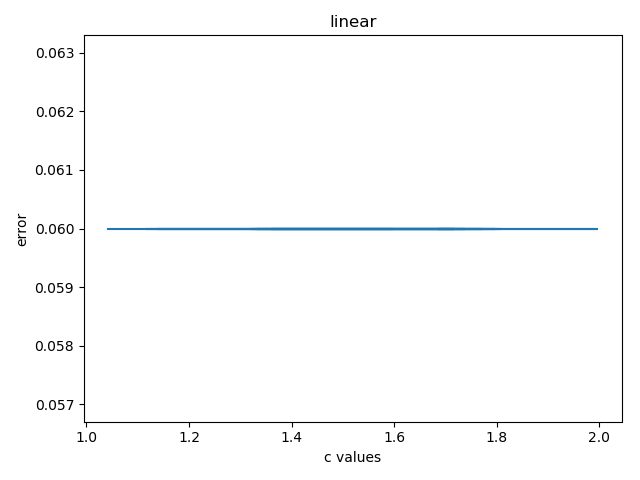
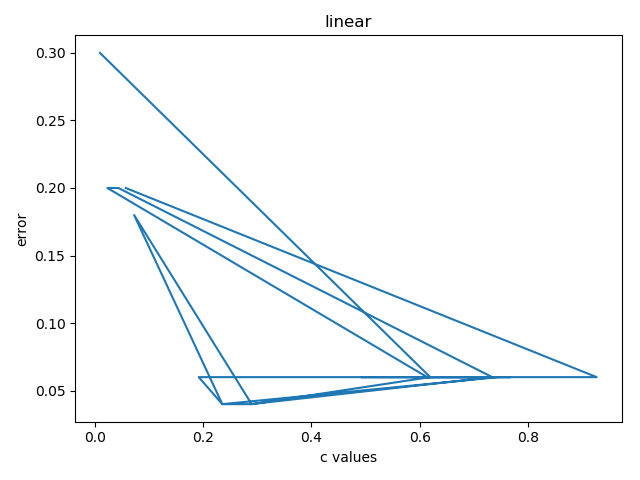
Will elaborate about this trial in trial 2 for reasons that are specified.

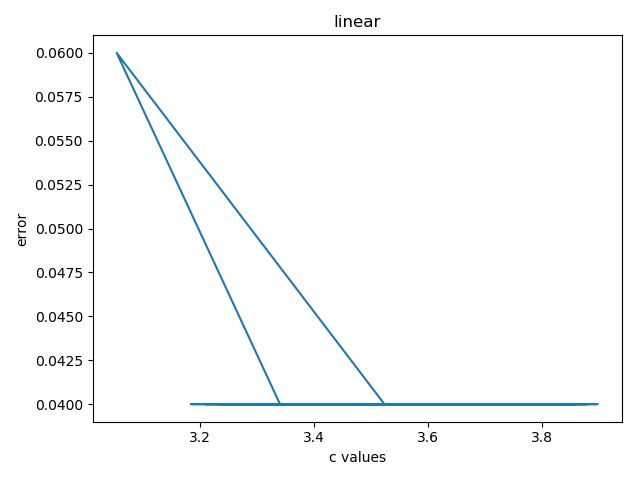
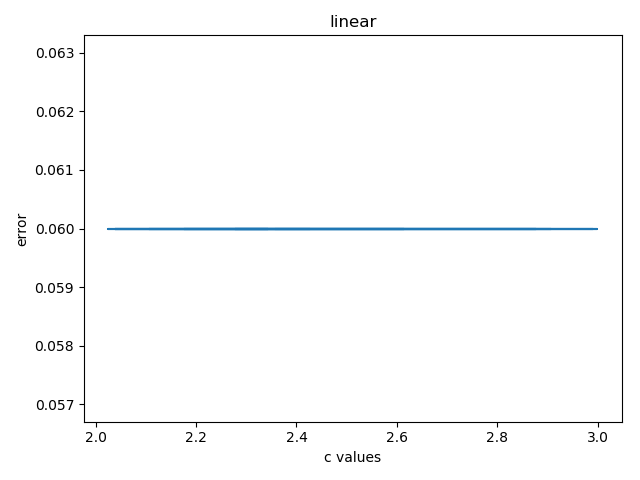
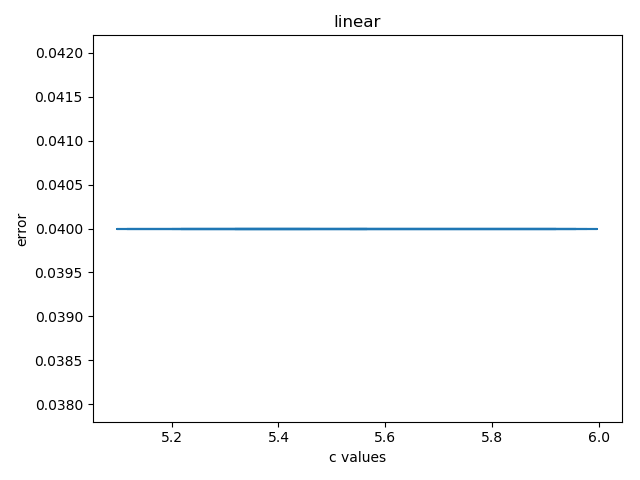
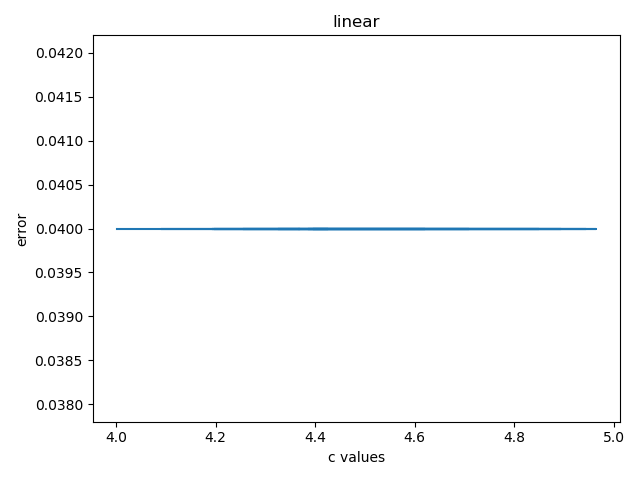
**Trial 2:**

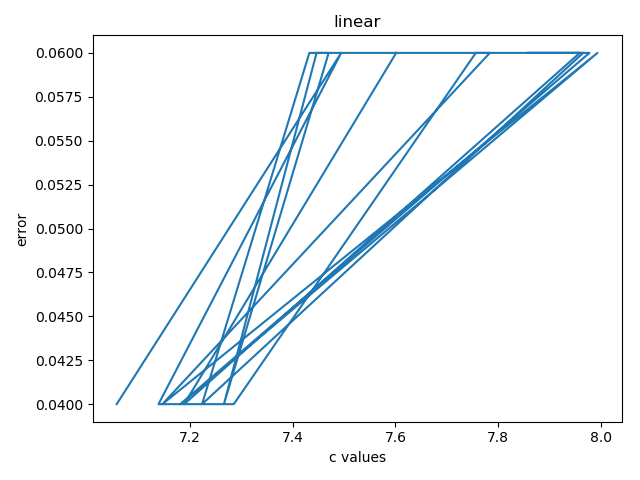
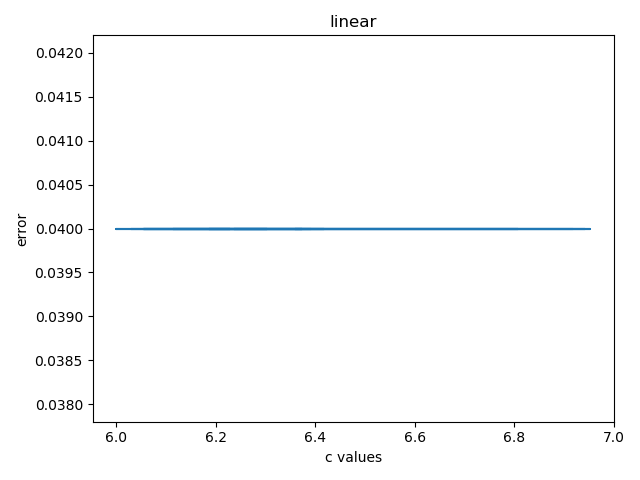
The first time I manually selected the parameters and found out that this procedure is not very effective and only detects the best c value for only one type of kernel. I wrote a script to determine the best c values for the following kernel types; ‘linear’, ‘poly’, ‘rbf’.



Invoking this method with the debugging parameter set to ‘True’ showcased something interesting. It seems that changes in ‘c’ values alter error rates only in certain intervals. For example:





With the kernel being ‘linear’ notice that ‘C’ was in the interval of 3.8 to 7 that the error rate was constant. A similar experience was carried throughout all the different kernels. This is why I thought it was a good idea to write the helper method called c\_selection().

The helper method allowed me to calculate the most optimal ‘c’ for all kernels.

The other odd observation I noticed was that when using the ‘poly’ kernel I received the same error rate along all iterations of ‘C’ values due to the fact that the degree parameter was not changed from it’s default value 3. I was not able to get helper\_function\_poly\_optimize.py to work properly to showcase that I could get non-constant error rates for the 'poly' kernel but I did figure out that changing the degree can, in fact, compute a more accurate model with less error.

**Results:**

'linear': Best ‘C’: 0.2276161066766654 with the error rate of 0.040000000000000036

'poly': Best ‘C’: 0.4143798502582356 with the error rate of 0.06000000000000005

'rbf': Best ‘C’: 0.9609096467109689 with the error rate of 0.26

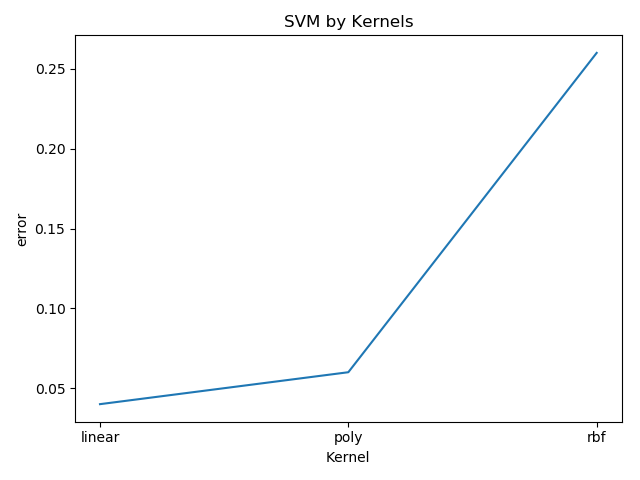


Figure Based off of best ‘C’ value for each kernel type

Chosen Model: linear

Chosen 'C' hyper-parameter: 0.2276161066766654

Confusion Matrix:

[[45 2]

[ 3 50]]

Average Accuracy: 0.95

Per-Class Precision: [0.9375 0.96153846]

Per-Class Recall: [0.95744681 0.94339623]

**Correctly labeled samples:**

Sample 1 was correctly detected. It's Gender is male. Features: [ 1. 11.7 10.6 24.9 28.5 10.4]

Sample 2 was correctly detected. It's Gender is female. Features: [ 0. 13.4 10.1 26.6 29.6 12. ]

Sample 3 was correctly detected. It's Gender is male. Features: [ 1. 13.7 12.5 28.6 33.8 11.9]

Sample 4 was correctly detected. It's Gender is female. Features: [ 1. 15.2 12.1 32.3 36.7 13.6]

Sample 5 was correctly detected. It's Gender is male. Features: [ 1. 11.9 11.4 26. 30.1 10.9]

**Incorrectly labeled samples:**

Sample 38 was invalidly detected. Features: [ 1. 11.1 9.9 23.8 27.1 9.8]

Sample 53 was invalidly detected. Features: [ 1. 9.5 8.2 19.6 22.4 7.8]

Sample 60 was invalidly detected. Features: [ 1. 13.3 11.1 27.8 32.3 11.3]

Sample 76 was invalidly detected. Features: [ 1. 12.3 11. 26.8 31.5 11.4]

Sample 100 was invalidly detected. Features: [ 1. 7.2 6.5 14.7 17.1 6.1]