CS 596 Homework Assignment 4 & 5
Machine Learning

*add source code

Part 1:

(Question 1)

The following operations have been performed on the MNIST dataset for the implementation of the feedforward neural network (FNN).

The code snippets are presented below:

```
# Model parameters as external flags
flags = tf.flags
FLAGS = flags.FLAGS
flags.DEFINE_float('learning_rate', 0.01, 'Learning rate for the training.')
flags.DEFINE_integer('max_steps', 2000, 'Number of steps to run trainer.')
flags.DEFINE_integer('hidden1', 150, 'Number of units in hidden layer 1.')
flags.DEFINE_integer('batch_size', 400,
  'Batch size. Must divide dataset sizes without remainder.')
flags.DEFINE_string('train_dir', 'tf_logs',
  'Directory to put the training data.')
flags.DEFINE_float('reg_constant', 0.1, 'Regularization constant.')

FLAGS._parse_flags()
print('\nParameters:')
for attr, value in sorted(FLAGS.__flags.items()):
  print('{} = {}'.format(attr, value))
print()

IMAGE_PIXELS = 784
CLASSES = 10
# Put logs for each run in separate directory
logdir = FLAGS.train_dir + '/' + datetime.now().strftime('%Y%m%d-%H%M%S') + '/'

beginTime = time.time()
```

The 3 sets of hyper-parameters tested by me are the following:

1) Set 1
   Parameters:
   batch_size = 400
   hidden1 = 150
   learning_rate = 0.01
   max_steps = 2000
   Activation function: relu
   Output function: Tanh

The Validation Test accuracy of this model was achieved to be 0.875

Confusion matrix:
[[630 0 5 2 0 5 16 1 4 0]
[0 1105 1 3 0 1 4 0 21 0]
[15 47 822 20 24 0 27 17 38 2]
[6 10 27 877 1 23 6 19 32 9]
[2 10 3 0 773 0 19 1 5 169]
[31 31 4 119 21 578 39 8 42 19]
[19 7 19 1 13 16 880 0 3 0]
[5 49 21 0 10 0 0 856 6 81]
[10 32 13 46 12 15 17 9 784 36]
[14 16 9 11 42 6 1 23 6 851]]

The average accuracy for this model is 0.863

PrecisionArray:
[0.90276454 0.84544759 0.89194915 0.81278962 0.85320088 0.89751553
0.87215064 0.91648822 0.83315622 0.73378265]

RecallArray:
[0.96632653 0.97356828 0.81589147 0.86831683 0.78716904 0.64798206
0.91858038 0.83268482 0.80492813 0.86323092]

This model appears to be the best among the 3 models used for this assignment.

2) Set 2
Parameters:
batch_size = 400
hidden1 = 100
learning_rate = 0.001
max_steps = 2000
Activation function: tanh
Output function: sigmoid

Validation Test accuracy of this model was achieved to be 0.5751

Confusion matrix:
[[633 1 0 4 0 1 36 0 4 1]
[3 132 5 6 1 1 9 0 78 0]
[130 151 111 37 13 3 161 4 217 10]
[220 93 6 537 20 7 53 6 27 41]
[39 40 29 3 391 6 135 3 84 252]
[432 39 1 78 22 86 109 8 31 86]
[139 15 3 1 9 8 753 2 20 8]
[70 81 10 14 261 0 2 385 73 132]
[65 116 1 74 57 8 120 4 432 97]
[50 46 15 15 225 6 16 10 42 512]]

The average accuracy for this model is 0.5301

PrecisionArray:
[0.44834214 0.6394052 0.81382979 0.69830949 0.39139139 0.68253968
0.54017217 0.91232227 0.42857143 0.48224608]

RecallArray:
[0.95204082 0.9092511 0.29651163 0.53168317 0.39816701 0.09641256
0.78601253 0.37451362 0.44353183 0.57879088]

3) Set 3
Parameters:
batch_size = 400
hidden1 = 120

learning_rate = 0.001
max_steps = 2000
Activation function: relu
Output function: Softmax

Validation Test accuracy of this model was achieved to be 0.386

Confusion matrix:
[[7 4 25 8 2 0 11 5 750 41]
[0 501 0 2 2 0 0 0 150 0]
[0 356 47 12 23 1 2 0 615 6]
[0 20 2 228 3 1 1 2 811 2]
[0 27 8 65 98 0 0 2 628 14]
[2 72 18 57 10 0 6 0 416 11]
[0 819 9 7 30 3 6 19 256 9]
[0 11 12 50 104 0 0 22 654 455]
[0 8 5 9 2 0 0 1 947 2]
[0 39 6 91 15 0 0 5 639 14]]

The average accuracy for this model is 0.361

PrecisionArray:
[0.6200453 0.6208046 0.20543624 0.25857143 0.62510035 0.
0.19905923 0.42355714 0.20054864 0.52224590]

RecallArray:
[0.00725345 0.70119515 0.03055264 0.22574257 0.09979633 0.
0.00626305 0.02140078 0.97227926 0.03369673]

The above set of parameters resulted in the least favorable results from all 3 of the specified results.

(Question 2)

The implementation of the best model on the selected testing samples

Parameters:
batch_size = 400
hidden1 = 150
learning_rate = 0.01
max_steps = 2000
Activation function: Relu
Output: Tanh
Validation Test accuracy 0.8572

Confusion matrix:
[[905 0 5 2 0 5 9 1 4 0]
[0 992 2 3 0 0 4 0 27 1]
[20 35 546 24 23 0 29 18 39 2]
[6 8 40 356 1 19 7 19 33 10]
[1 12 5 0 750 0 20 1 6 162]
[30 30 3 102 21 435 40 8 47 21]
[18 7 19 1 11 16 654 0 2 0]
[5 46 22 0 9 0 0 751 10 89]
[8 24 10 48 12 14 19 7 799 31]
[15 16 8 11 51 5 1 20 8 874]]

The average accuracy for this model is 0.8507

PrecisionArray:
[0.70037951 0.7614286 0.9029915 0.85325719 0.845392952 0.86666667
905383360.92 0.81948718 0.73445378]

RecallArray:
[0.75836735 0.8840065 0.8220154 0.7568713 0.62527902 0.52219731
0.83545574 0.65452101 0.76552854 0.8656416]

The 10 testing images for which the model made wrong predictions are:

The reason for the failures can be considered as not proper edges drawn or captured in the images and thus the edge detection isn't working and the model is not trained well.
This can be improved using increasing the number of train images and then filter out during the preprocessing of the images.

PART 2:

(Question 1)

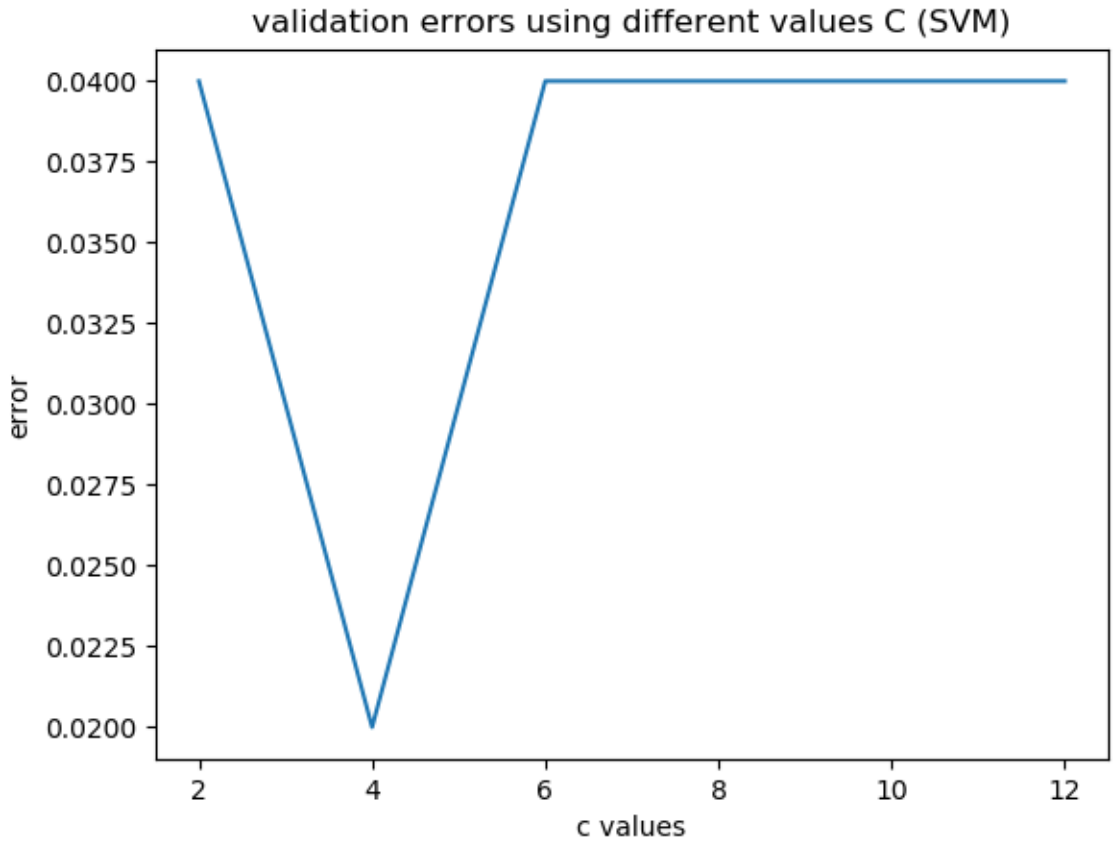
In this assignment, we try to develop a SVM classifier that can identify the gender of a crab from its physical measurements.

Linear SVM:

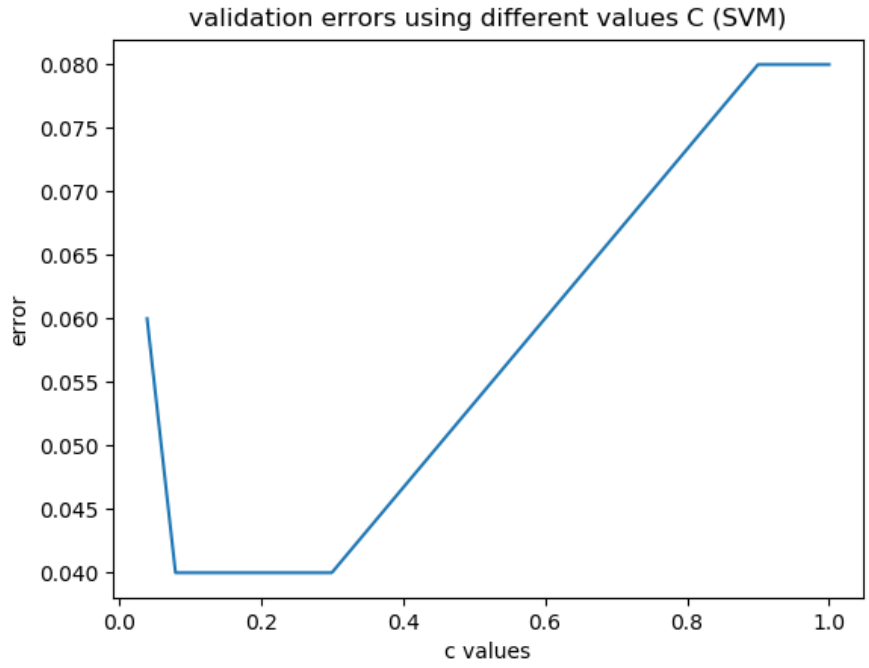
Linear Kernel is used when the data is linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set. One of the examples where there are a lot of features, is Text Classification, as each alphabet is a new feature.

Plotting of the linear models using different values of C:

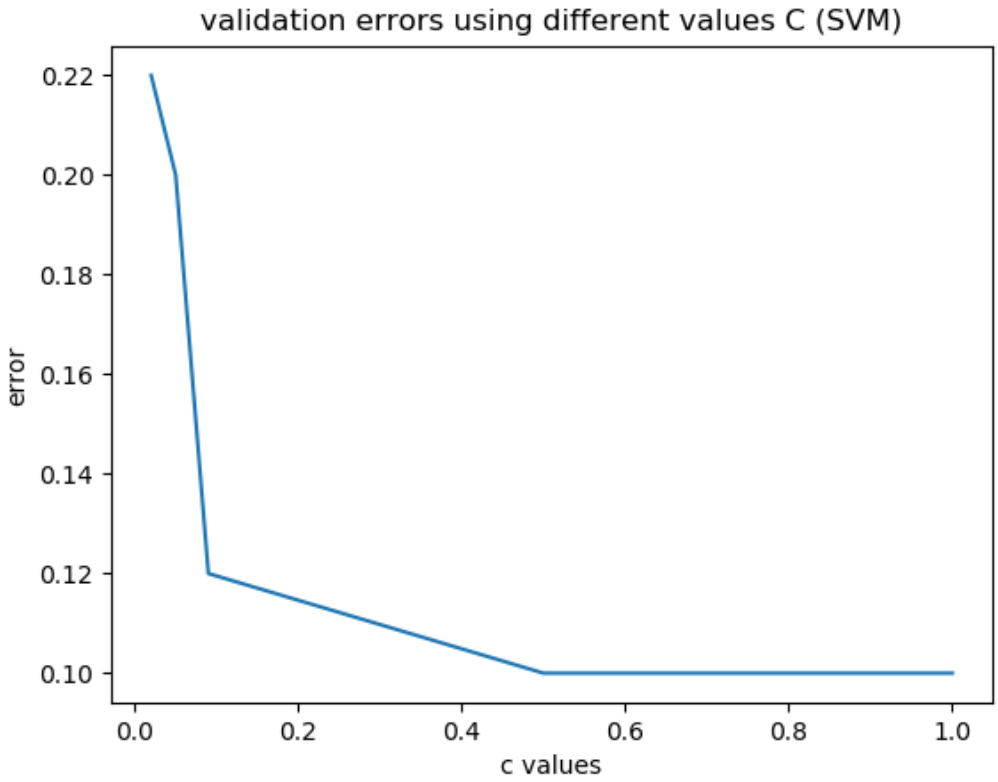
C = [2 , 4 , 6, 8, 10, 12]



C = [0.02 , 0.05 , 0.09, 0.50, 0.9, 1]



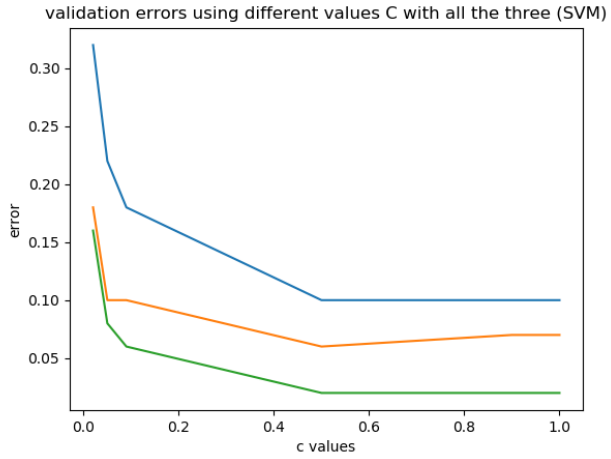
C = [0.04 , 0.08 , 0.3, 0.6, 0.9, 1]



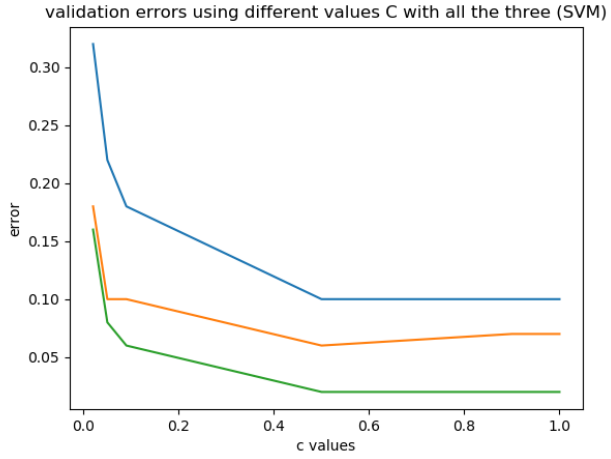
Here, in the above examples, the horizontal direction represents different values of C, and the vertical direction represents validation errors.

Plotting the validation errors using different values for C in all 3 kernels:

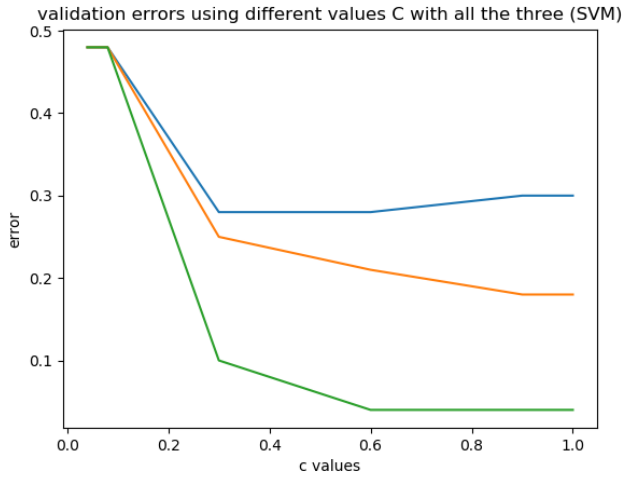
Linear Kernel:



Polynomial Kernel:

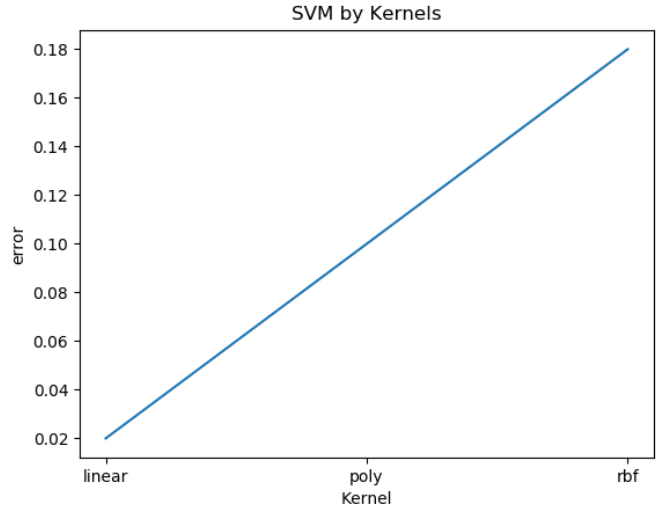


RBF Kernel:

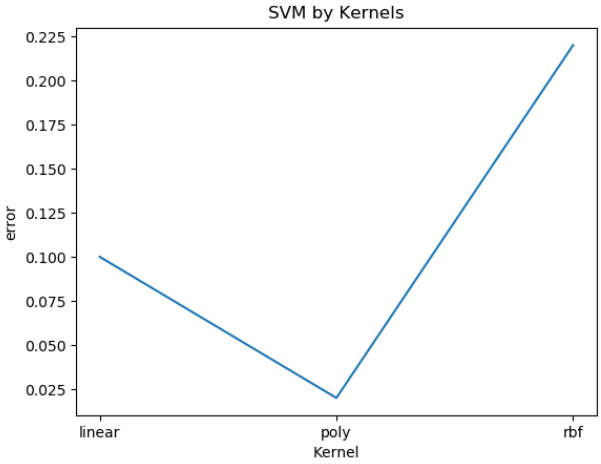


Plotting the validation errors while using linear, RBF kernel, or Polynomial kernel (with other hyper-parameters fixed)

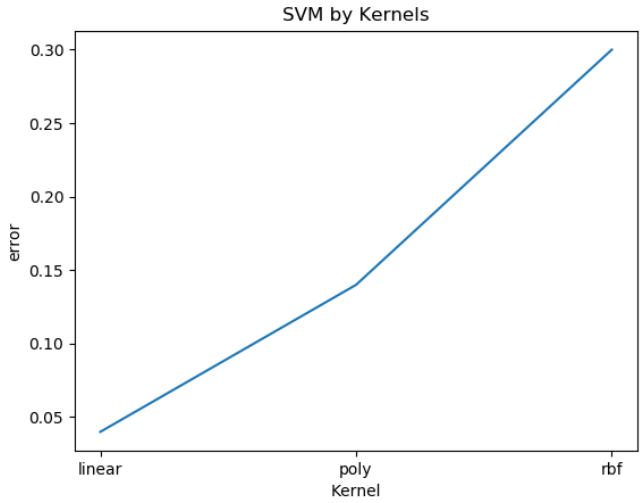
C = [2 , 4 , 6, 8, 10, 12]



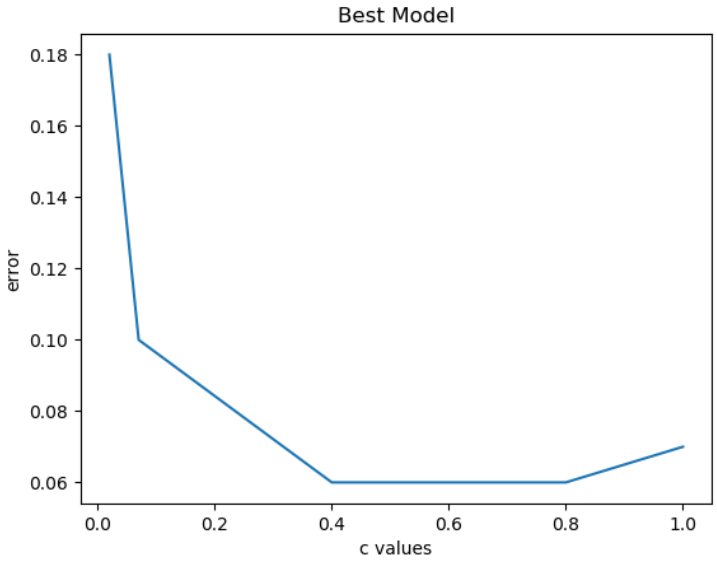
C = [0.02 , 0.05 , 0.09, 0.50, 0.9, 1]

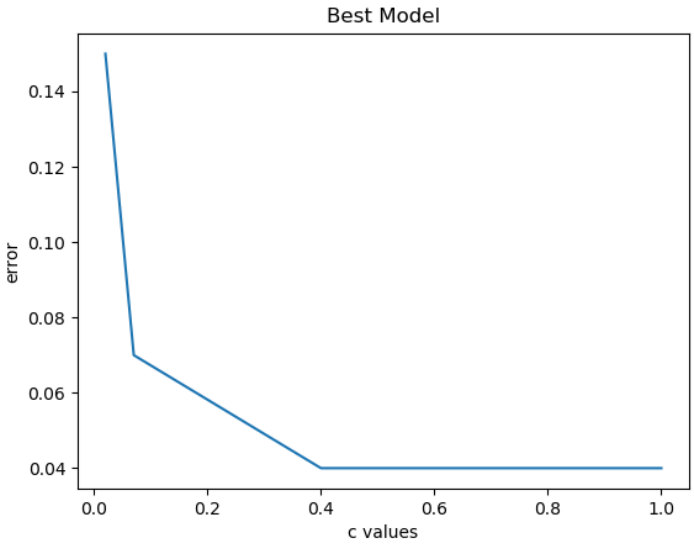


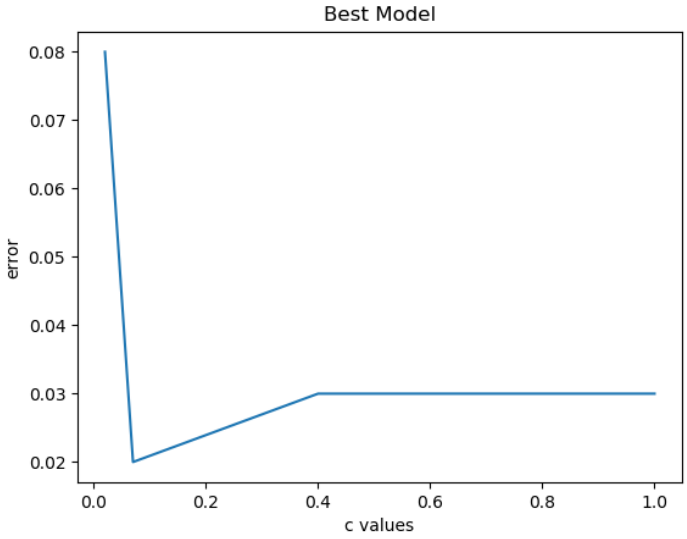
C = [0.04 , 0.08 , 0.3, 0.6, 0.9, 1]



The best models for the respective same 3 C values:







For the 1st image:

The Confusion Matrix:

[[49 4]

 [0 47]]

Average Accuracy: 0.96

Per-Class Precision: [1.   0.92156863]

Per-Class Recall: [0.9245283 1.   ]

Failure Cases where female was categorized as male or vice versa:

-1.0 1.0

-1.0 1.0

-1.0 1.0

-1.0 1.0

True Cases:where female was categorized as female and male as mail

-1.0 -1.0

1.0 1.0

1.0 1.0

-1.0 -1.0

1.0 1.0

-1.0 -1.0

For the 2nd image:

Confusion Matrix:

[[45 1]

 [6 48]]

Average Accuracy: 0.93

Per-Class Precision: [0.88235294 0.97959184]

Per-Class Recall: [0.97826087 0.88888889]

Failure Cases where female was categorized as male or vice versa:

1.0 -1.0

1.0 -1.0

1.0 -1.0

-1.0 1.0

1.0 -1.0

True Cases:where female was categorized as female and male as mail

-1.0 -1.0

1.0 1.0

1.0 1.0

1.0 1.0

1.0 1.0

-1.0 -1.0

For the 3rd image:

Confusion Matrix:

[[46 2]

 [1 51]]

Average Accuracy: 0.97

Per-Class Precision: [0.9787234 0.96226415]

Per-Class Recall: [0.95833333 0.98076923]

Failure Cases where female was categorized as male or vice versa:

-1.0 1.0

1.0 -1.0

-1.0 1.0

True Cases:where female was categorized as female and male as mail

1.0 1.0

1.0 1.0

1.0 1.0

1.0 1.0

-1.0 -1.0

1.0 1.0

We can say that the failure occurs when
Case 1: Male is identified as female that is 1 is predicted as -1.
Case 2: Female is identified as male that is -1 is predicted as 1.
In all other cases true value are predicted.
Also, among the 3 kernels, the RBF tends to perform the best with the accuracy of 97% and the best_c is about 0.4