UCLA CS 145 Homework #2

DUE DATE: Tuesday 02/05/2019 11:59 pm

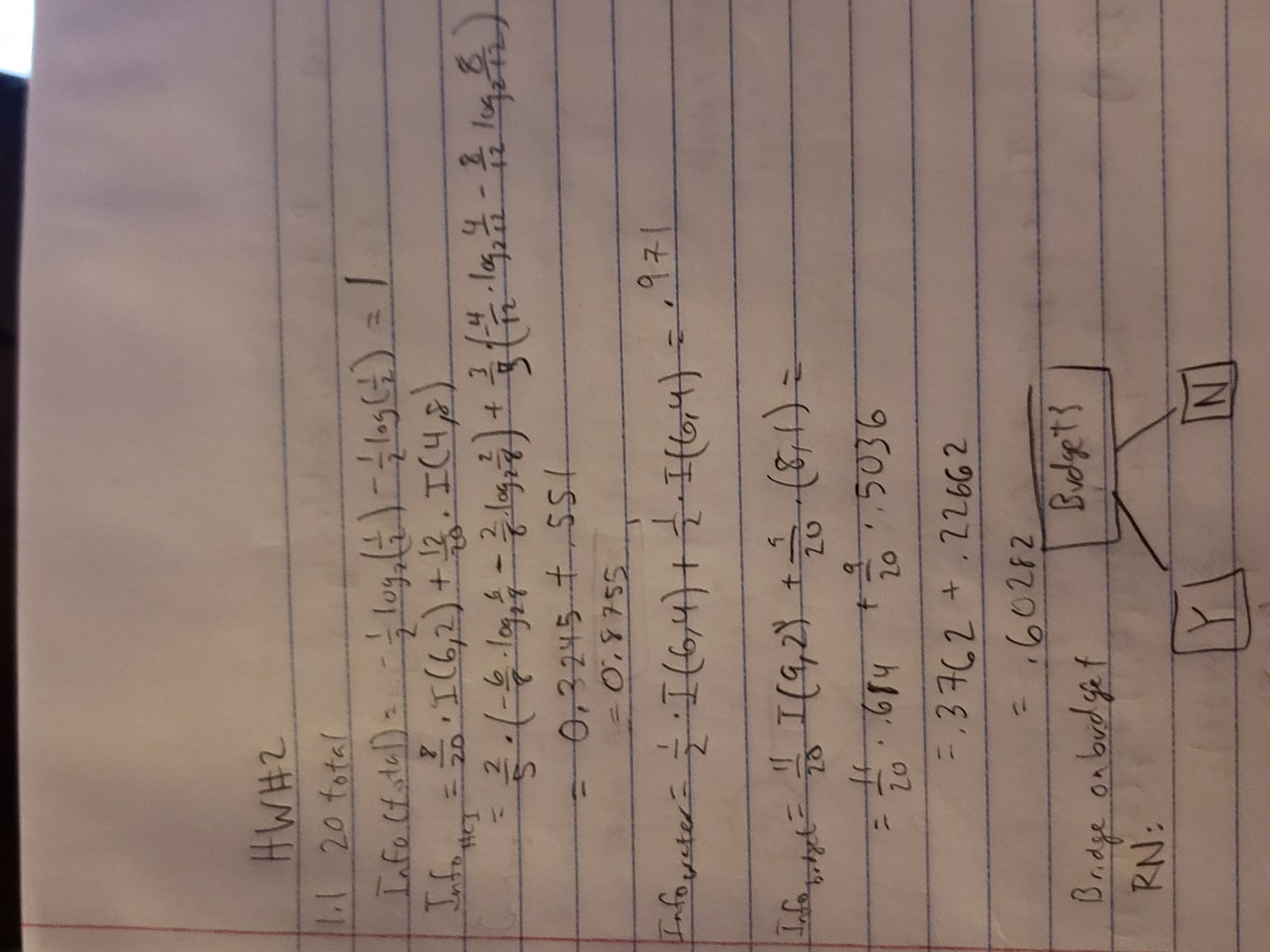
Note:

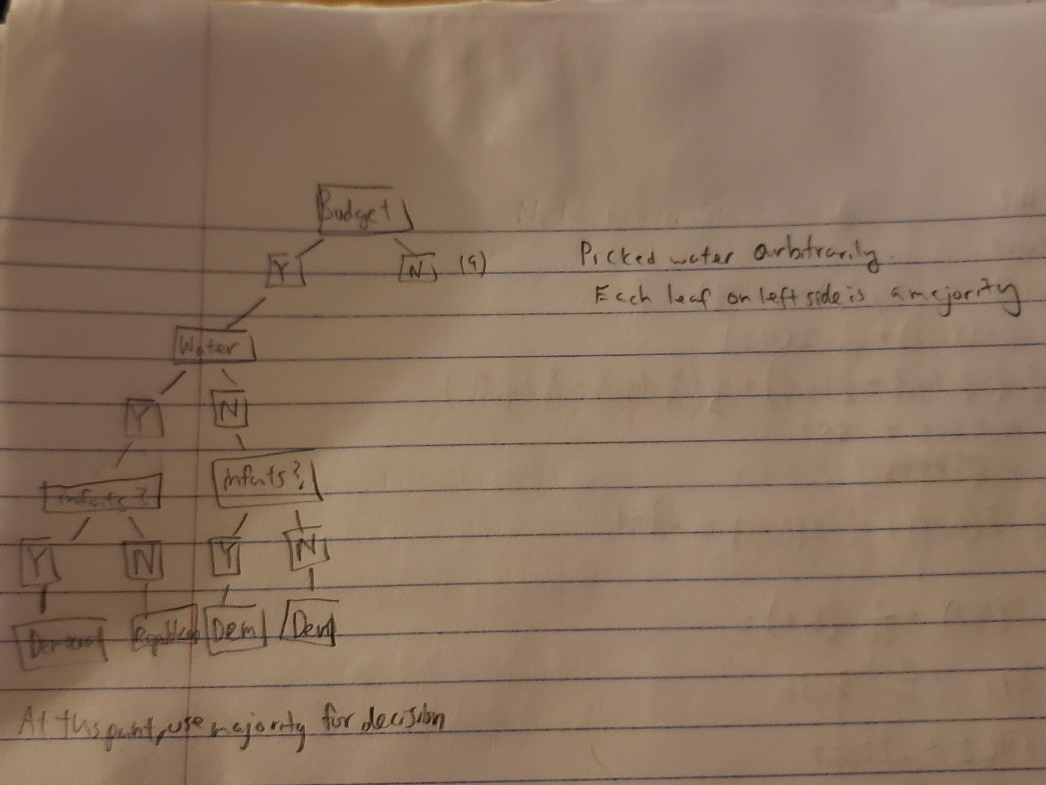
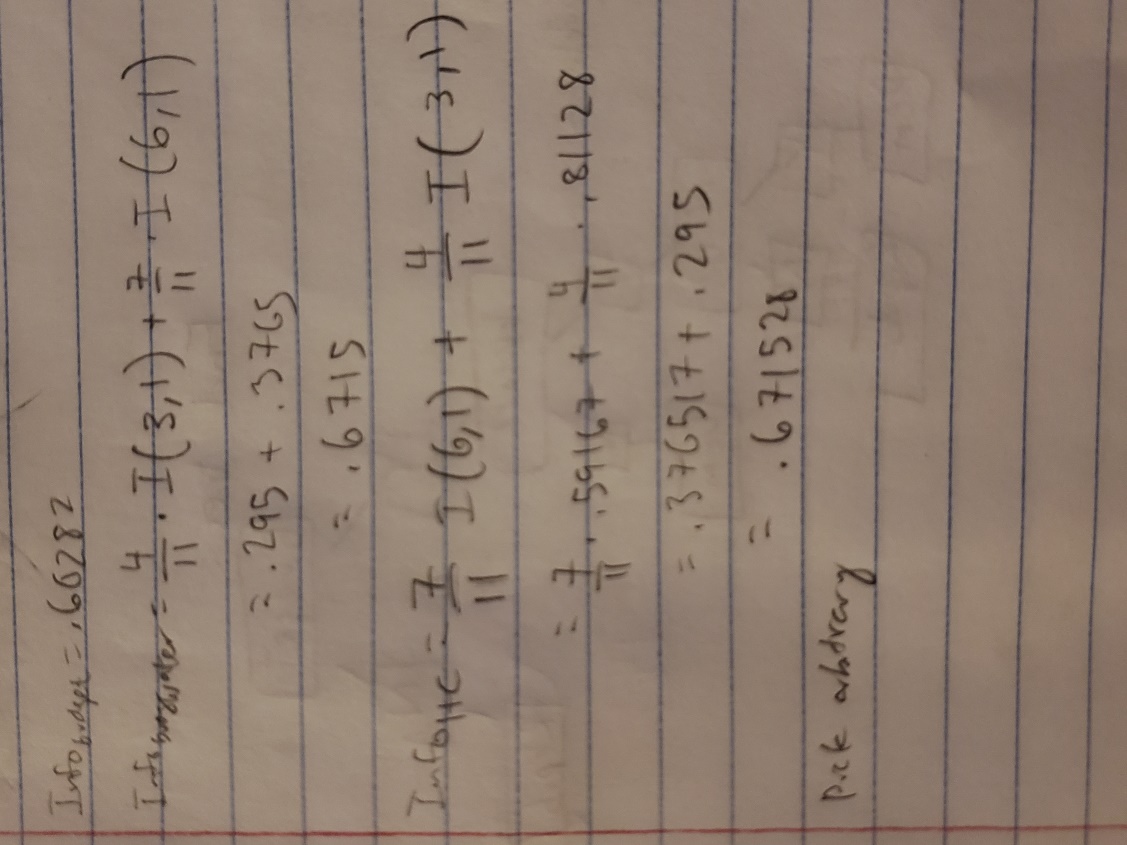
* You are expected to submit both a report and code. The submission format is specified on CCLE under HW2 description.
* “########## Please Fill Missing Lines Here ##########” is used where input from you is needed in the code file.

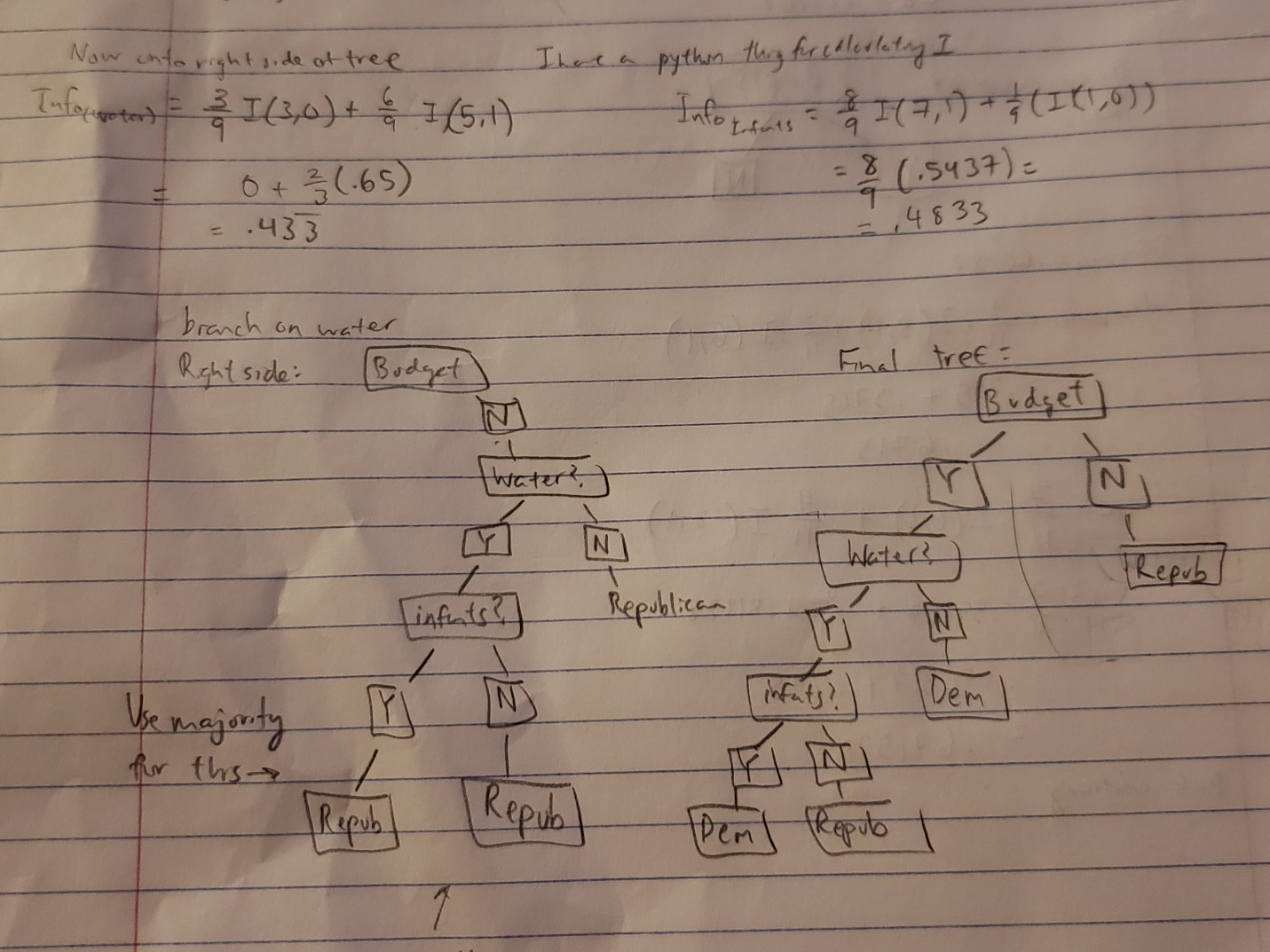
## Decision Tree

1.1. Construct a decision tree for samples from the congressional voting records dataset with the first three attributes in the UCI machine learning repository. Information gain is used to select the attributes. Please write down the major steps in the construction process, i.e., you need to show the information gain for each candidate attribute when a new node is created in the tree.

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Vote for handicapped-infants? | Vote for water-project-cost-sharing | Vote for budget-resolution-adoption |
| Democrat | Y | N | Y |
| Republican | N | Y | N |
| Democrat | Y | Y | Y |
| Republican | N | Y | N |
| Democrat | N | Y | N |
| Democrat | N | Y | Y |
| Democrat | Y | N | Y |
| Democrat | Y | Y | Y |
| Republican | N | Y | Y |
| Republican | Y | Y | N |
| Democrat | N | N | Y |
| Republican | N | Y | N |
| Republican | N | N | N |
| Democrat | N | N | Y |
| Republican | N | N | N |
| Republican | N | Y | N |
| Democrat | Y | N | Y |
| Democrat | Y | N | Y |
| Republican | N | N | N |
| Republican | Y | N | Y |







1.2 In DecisionTree\DecisionTree.py, fill in the missing lines for building a decision tree model, using (a) information gain and (b) gain ratio. Output the accuracy on the test data and compare the two versions of decision tree. Which attribute selection measure do you want to choose for the dataset and why.

Info Gain:

best\_feature is: legs

best\_feature is: fins

best\_feature is: toothed

best\_feature is: eggs

best\_feature is: hair

best\_feature is: hair

best\_feature is: toothed

best\_feature is: aquatic

{'legs': {0: {'fins': {0.0: {'toothed': {0.0: 7.0, 1.0: 3.0}},

1.0: {'eggs': {0.0: 1.0, 1.0: 4.0}}}},

2: {'hair': {0.0: 2.0, 1.0: 1.0}},

4: {'hair': {0.0: {'toothed': {0.0: 7.0, 1.0: 5.0}}, 1.0: 1.0}},

6: {'aquatic': {0.0: 6.0, 1.0: 7.0}},

8: 7.0}}

Test accuracy: 0.8571428571428571

Gain Ratio:

best\_feature is: feathers

best\_feature is: backbone

best\_feature is: airborne

best\_feature is: predator

best\_feature is: milk

best\_feature is: fins

best\_feature is: legs

{'feathers': {0: {'backbone': {0.0: {'airborne': {0.0: {'predator': {0.0: 6.0,

1.0: 7.0}},

1.0: 6.0}},

1.0: {'milk': {0.0: {'fins': {0.0: {'legs': {0.0: 3.0,

4.0: 5.0}},

1.0: 4.0}},

1.0: 1.0}}}},

1: 2.0}}

Test accuracy: 0.8095238095238095

We want to use Info gain because it has the best accuracy. I would surmise that even though gain ratio typically is better and reduces bias, for our data in particular, info gain is better, probably because the data has uneven splits which is bad for gain ratio.

## Support Vector Machine

2.1 The table shown below contains 20 data points and their class labels.

|  |  |  |  |
| --- | --- | --- | --- |
| Point # | x1 | x2 | Class (y) |
| 1 | 0.52 | -1 | 1 |
| 2 | 0.91 | 0.32 | 1 |
| 3 | -1.48 | 1.23 | 1 |
| 4 | 0.01 | 1.44 | 1 |
| 5 | -0.46 | -0.37 | 1 |
| 6 | 0.41 | 2.04 | 1 |
| 7 | 0.53 | 0.77 | 1 |
| 8 | -1.21 | -1.1 | 1 |
| 9 | -0.39 | 0.96 | 1 |
| 10 | -0.96 | 0.08 | 1 |
| 11 | 2.46 | 2.59 | -1 |
| 12 | 3.05 | 2.87 | -1 |
| 13 | 2.2 | 3.04 | -1 |
| 14 | 1.89 | 2.64 | -1 |
| 15 | 4.51 | -0.52 | -1 |
| 16 | 3.06 | 1.3 | -1 |
| 17 | 3.16 | -0.56 | -1 |
| 18 | 2.05 | 1.54 | -1 |
| 19 | 2.34 | 0.72 | -1 |
| 20 | 2.94 | 0.13 | -1 |

Suppose by solving the dual form of the quadratic programming of svm, we can derive the ’s for each data point as follows:

= 0.5084

= 0.4625

= 0.9709

Others = 0

(a) Please point out the support vectors in the training points.

(b) Calculate the normal vector of the hyperplane:

(c) Calculate the bias b, according to , where indicates the support vectors and is the total number of support vectors.

(d) Write down the learned decision boundary function (the hyperplane)

by substituting and with learned values in the formula.

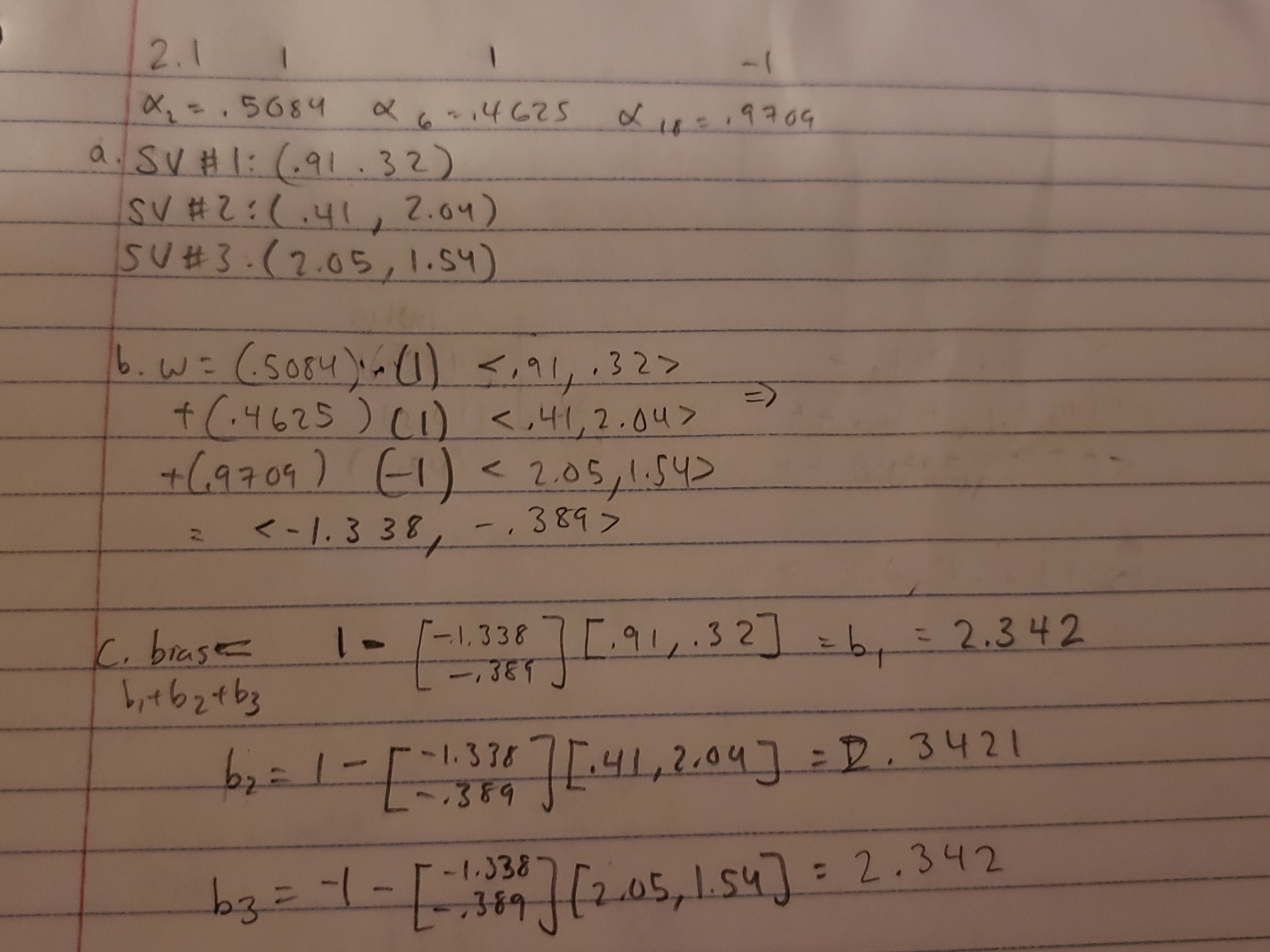
(e) Suppose there is a new data point , please use the decision boundary to

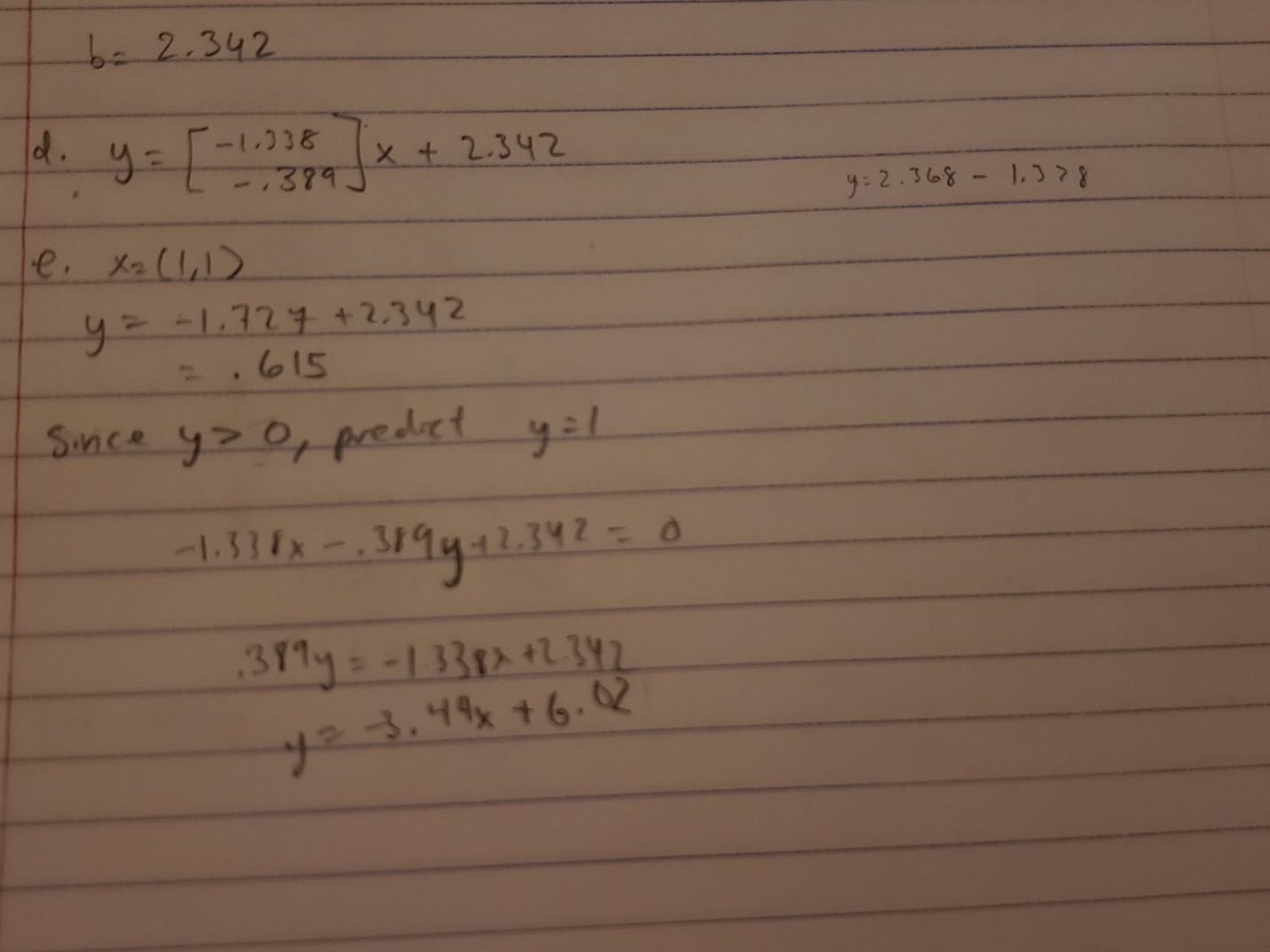
predict its class label.

(f) Show a plot of the data points and your decision boundary line (x1 feature on x-axis,

x2 feature on y-axis) in your report. Plot both data points and decision boundary in the

same graph, and use different colors to represent points in different classes (y).





2.2 In SVM\svm.py, fill in the missing lines for support vector machines. Output the accuracy on the test data and compare (a) hard margin and soft margin SVM for linear classifier; and (b) different kernels for soft margin SVM. Which model do you want to choose?

Margin: Hard

kernel Type: Linear

pcost dcost gap pres dres

0: -1.4438e+02 -3.3911e+02 5e+03 6e+01 2e+00

1: -3.8214e+02 -5.6098e+02 3e+03 4e+01 1e+00

2: -6.6231e+02 -9.2563e+02 3e+03 4e+01 1e+00

3: -1.3110e+03 -1.7230e+03 3e+03 4e+01 1e+00

4: -2.7029e+03 -3.4114e+03 4e+03 3e+01 1e+00

5: -9.7014e+03 -1.1663e+04 4e+03 3e+01 1e+00

6: -3.6797e+04 -4.2248e+04 6e+03 3e+01 1e+00

7: -1.1615e+05 -1.2907e+05 1e+04 3e+01 1e+00

8: -5.5869e+05 -5.9758e+05 4e+04 3e+01 1e+00

9: -1.7063e+06 -1.8091e+06 1e+05 3e+01 1e+00

10: -2.8334e+07 -2.8527e+07 2e+05 3e+01 1e+00

11: -3.5391e+09 -3.5429e+09 4e+06 3e+01 1e+00

12: -7.1603e+09 -7.1677e+09 7e+06 3e+01 1e+00

13: -9.3453e+09 -9.3547e+09 9e+06 3e+01 1e+00

14: -1.6524e+10 -1.6540e+10 2e+07 3e+01 1e+00

15: -2.7493e+10 -2.7518e+10 3e+07 3e+01 1e+00

Terminated (singular KKT matrix).

1098 support vectors out of 1098 points

Test accuracy: 0.5547445255474452

Margin: Soft

kernel Type: Linear

pcost dcost gap pres dres

0: -1.7838e+04 -1.4216e+08 4e+08 6e-01 2e-10

1: 2.0347e+04 -3.4085e+07 6e+07 8e-02 2e-10

2: 3.5993e+04 -9.5151e+06 2e+07 2e-02 1e-10

3: 4.9596e+04 -6.0614e+06 1e+07 1e-02 7e-11

4: 5.3650e+04 -3.2836e+06 5e+06 6e-03 5e-11

5: 5.2427e+04 -1.2117e+06 2e+06 1e-03 4e-11

6: 2.1260e+04 -2.6704e+05 3e+05 3e-06 4e-11

7: -1.0283e+03 -9.9499e+04 1e+05 9e-07 4e-11

8: -4.8524e+03 -4.9722e+04 4e+04 4e-07 4e-11

9: -5.2399e+03 -4.9623e+04 4e+04 4e-07 4e-11

10: -6.3385e+03 -2.2126e+04 2e+04 9e-08 4e-11

11: -6.8781e+03 -2.2626e+04 2e+04 9e-08 4e-11

12: -7.6981e+03 -2.1745e+04 1e+04 7e-08 4e-11

13: -6.7996e+03 -2.0773e+04 1e+04 7e-08 4e-11

14: -8.1136e+03 -1.6840e+04 9e+03 2e-08 5e-11

15: -8.7965e+03 -1.6725e+04 8e+03 1e-08 5e-11

16: -8.6221e+03 -1.6538e+04 8e+03 1e-08 4e-11

17: -8.8831e+03 -1.6609e+04 8e+03 9e-09 5e-11

18: -9.2486e+03 -1.5360e+04 6e+03 6e-09 4e-11

19: -9.2427e+03 -1.4163e+04 5e+03 3e-09 5e-11

20: -9.4498e+03 -1.2471e+04 3e+03 2e-09 5e-11

21: -9.5107e+03 -1.2150e+04 3e+03 1e-09 5e-11

22: -9.5965e+03 -1.1545e+04 2e+03 4e-10 5e-11

23: -9.5977e+03 -1.1531e+04 2e+03 4e-10 5e-11

24: -9.7482e+03 -1.1573e+04 2e+03 2e-10 5e-11

25: -1.0001e+04 -1.0944e+04 9e+02 7e-11 5e-11

26: -1.0111e+04 -1.0701e+04 6e+02 3e-11 5e-11

27: -1.0304e+04 -1.0347e+04 4e+01 9e-13 5e-11

28: -1.0323e+04 -1.0324e+04 5e-01 8e-12 5e-11

29: -1.0324e+04 -1.0324e+04 5e-03 4e-12 5e-11

Optimal solution found.

34 support vectors out of 1098 points

Test accuracy: 0.9890510948905109

Margin: Hard

kernel Type: Polynomial

pcost dcost gap pres dres

0: -4.7482e+01 -9.8592e+01 3e+03 5e+01 2e+00

1: -7.5146e+01 -4.3014e+01 1e+03 1e+01 6e-01

2: -2.8437e+01 -4.5099e+00 2e+02 3e+00 1e-01

3: -2.0095e+00 -1.0459e+00 1e+01 1e-01 5e-03

4: -9.6162e-01 -6.9578e-01 4e+00 5e-02 2e-03

5: -5.1195e-01 -4.4619e-01 1e+00 2e-02 6e-04

6: -4.0510e-01 -3.6983e-01 8e-01 8e-03 3e-04

7: -3.5782e-01 -3.2222e-01 5e-01 4e-03 2e-04

8: -3.2195e-01 -2.7646e-01 3e-01 2e-03 8e-05

9: -2.9806e-01 -2.5521e-01 2e-01 1e-03 6e-05

10: -2.7864e-01 -2.3925e-01 2e-01 1e-03 4e-05

11: -2.5877e-01 -2.1672e-01 1e-01 6e-04 3e-05

12: -2.4433e-01 -1.9951e-01 1e-01 5e-04 2e-05

13: -2.2587e-01 -1.8386e-01 8e-02 3e-04 1e-05

14: -2.0033e-01 -1.7072e-01 7e-02 2e-04 8e-06

15: -1.8442e-01 -1.6328e-01 6e-02 1e-04 6e-06

16: -1.8011e-01 -1.6239e-01 6e-02 1e-04 5e-06

17: -1.6899e-01 -1.5830e-01 4e-02 7e-05 3e-06

18: -1.6418e-01 -1.5664e-01 3e-02 5e-05 2e-06

19: -1.5921e-01 -1.5479e-01 1e-02 2e-05 1e-06

20: -1.5442e-01 -1.5397e-01 1e-02 1e-05 5e-07

21: -1.5260e-01 -1.5374e-01 8e-03 7e-06 3e-07

22: -1.5251e-01 -1.5349e-01 2e-03 1e-06 5e-08

23: -1.5319e-01 -1.5341e-01 3e-04 1e-07 5e-09

24: -1.5336e-01 -1.5340e-01 4e-05 3e-17 1e-13

25: -1.5340e-01 -1.5340e-01 5e-07 1e-16 2e-13

26: -1.5340e-01 -1.5340e-01 5e-09 3e-16 1e-13

Optimal solution found.

19 support vectors out of 1098 points

Test accuracy: 0.927007299270073

Margin: Soft

pcost dcost gap pres dres

0: -8.9993e+02 -7.6966e+07 2e+08 4e-01 2e-06

1: 2.0423e+04 -1.4910e+07 3e+07 5e-02 2e-06

2: 2.6926e+04 -1.6663e+06 3e+06 5e-03 7e-07

3: 1.3337e+04 -2.0412e+05 4e+05 5e-04 1e-07

4: 4.4128e+03 -5.4525e+04 9e+04 1e-04 2e-08

5: 1.2674e+03 -1.3679e+04 2e+04 2e-05 4e-09

6: 4.1532e+02 -3.8486e+03 6e+03 5e-06 9e-10

7: 1.4911e+02 -1.3070e+03 2e+03 1e-06 3e-10

8: 5.6834e+01 -4.4006e+02 6e+02 4e-07 1e-10

9: 1.8764e+01 -1.0151e+02 1e+02 7e-08 3e-11

10: 5.8413e+00 -2.3618e+01 3e+01 1e-08 7e-12

11: 1.1303e+00 -2.7148e+00 4e+00 2e-16 1e-12

12: 1.4499e-01 -5.6055e-01 7e-01 2e-16 6e-13

13: -2.4286e-02 -4.2777e-01 4e-01 1e-16 3e-13

14: -1.1167e-01 -2.2497e-01 1e-01 2e-16 2e-13

15: -1.3095e-01 -1.9550e-01 6e-02 2e-16 1e-13

16: -1.4060e-01 -1.7556e-01 3e-02 2e-16 8e-14

17: -1.4784e-01 -1.6171e-01 1e-02 2e-16 1e-13

18: -1.4954e-01 -1.5895e-01 9e-03 2e-16 1e-13

19: -1.5208e-01 -1.5520e-01 3e-03 2e-16 1e-13

20: -1.5314e-01 -1.5374e-01 6e-04 2e-16 1e-13

21: -1.5336e-01 -1.5345e-01 9e-05 2e-16 1e-13

22: -1.5340e-01 -1.5340e-01 1e-06 2e-16 1e-13

23: -1.5340e-01 -1.5340e-01 1e-08 2e-16 1e-13

Optimal solution found.

19 support vectors out of 1098 points

Test accuracy: 0.927007299270073

Margin: Hard

kernel Type: Gaussian

Gaussian Kernel computing: I am not efficiently implemented. Please consider smarter implementation

pcost dcost gap pres dres

0: -5.2888e+01 -1.4857e+02 3e+03 4e+01 2e+00

1: -5.4674e+01 -1.5650e+02 9e+02 1e+01 6e-01

2: -4.0143e+01 -1.4099e+02 5e+02 6e+00 3e-01

3: -1.6203e+01 -1.3322e+02 2e+02 1e+00 5e-02

4: -4.3242e+01 -9.8574e+01 8e+01 4e-01 2e-02

5: -4.4512e+01 -9.8854e+01 7e+01 3e-01 2e-02

6: -6.2935e+01 -8.9393e+01 3e+01 8e-02 4e-03

7: -7.1084e+01 -8.5844e+01 2e+01 2e-02 9e-04

8: -7.7948e+01 -8.3122e+01 5e+00 3e-03 2e-04

9: -7.8491e+01 -8.3026e+01 5e+00 2e-03 1e-04

10: -8.1488e+01 -8.2533e+01 1e+00 4e-04 2e-05

11: -8.2330e+01 -8.2421e+01 9e-02 4e-06 2e-07

12: -8.2389e+01 -8.2417e+01 3e-02 3e-08 1e-09

13: -8.2415e+01 -8.2415e+01 7e-04 6e-10 3e-11

14: -8.2415e+01 -8.2415e+01 2e-05 3e-13 2e-14

Optimal solution found.

35 support vectors out of 1098 points

Test accuracy: 1.0

Margin: Soft

kernel Type: Gaussian

Gaussian Kernel computing: I am not efficiently implemented. Please consider smarter implementation

pcost dcost gap pres dres

0: 2.4426e+06 -5.9935e+07 1e+08 2e-01 1e-12

1: 2.2323e+06 -8.4591e+06 1e+07 2e-02 1e-12

2: 7.8986e+05 -1.8404e+06 3e+06 3e-03 6e-13

3: 1.4457e+05 -2.1098e+05 4e+05 2e-12 5e-13

4: 2.0551e+04 -2.5059e+04 5e+04 2e-12 2e-13

5: 2.7894e+03 -3.6645e+03 6e+03 5e-13 9e-14

6: 3.2380e+02 -5.8973e+02 9e+02 5e-14 3e-14

7: 8.6171e+01 -2.8114e+02 4e+02 1e-13 1e-14

8: 2.4243e+01 -2.5707e+02 3e+02 2e-14 1e-14

9: -3.1586e+01 -1.4121e+02 1e+02 6e-14 1e-14

10: -3.6462e+01 -1.3991e+02 1e+02 9e-14 1e-14

11: -6.1108e+01 -1.1246e+02 5e+01 3e-14 1e-14

12: -7.0696e+01 -1.0087e+02 3e+01 2e-14 1e-14

13: -7.7258e+01 -8.9722e+01 1e+01 6e-14 1e-14

14: -8.0183e+01 -8.5449e+01 5e+00 1e-13 2e-14

15: -8.2041e+01 -8.2884e+01 8e-01 1e-14 2e-14

16: -8.2355e+01 -8.2487e+01 1e-01 1e-13 2e-14

17: -8.2413e+01 -8.2418e+01 6e-03 1e-13 2e-14

18: -8.2415e+01 -8.2416e+01 5e-04 4e-15 2e-14

19: -8.2415e+01 -8.2415e+01 1e-05 4e-14 2e-14

Optimal solution found.

35 support vectors out of 1098 points

Test accuracy: 1.0

I would use the soft or hard margin with a Gaussian kernel because its test accuracy is perfect.