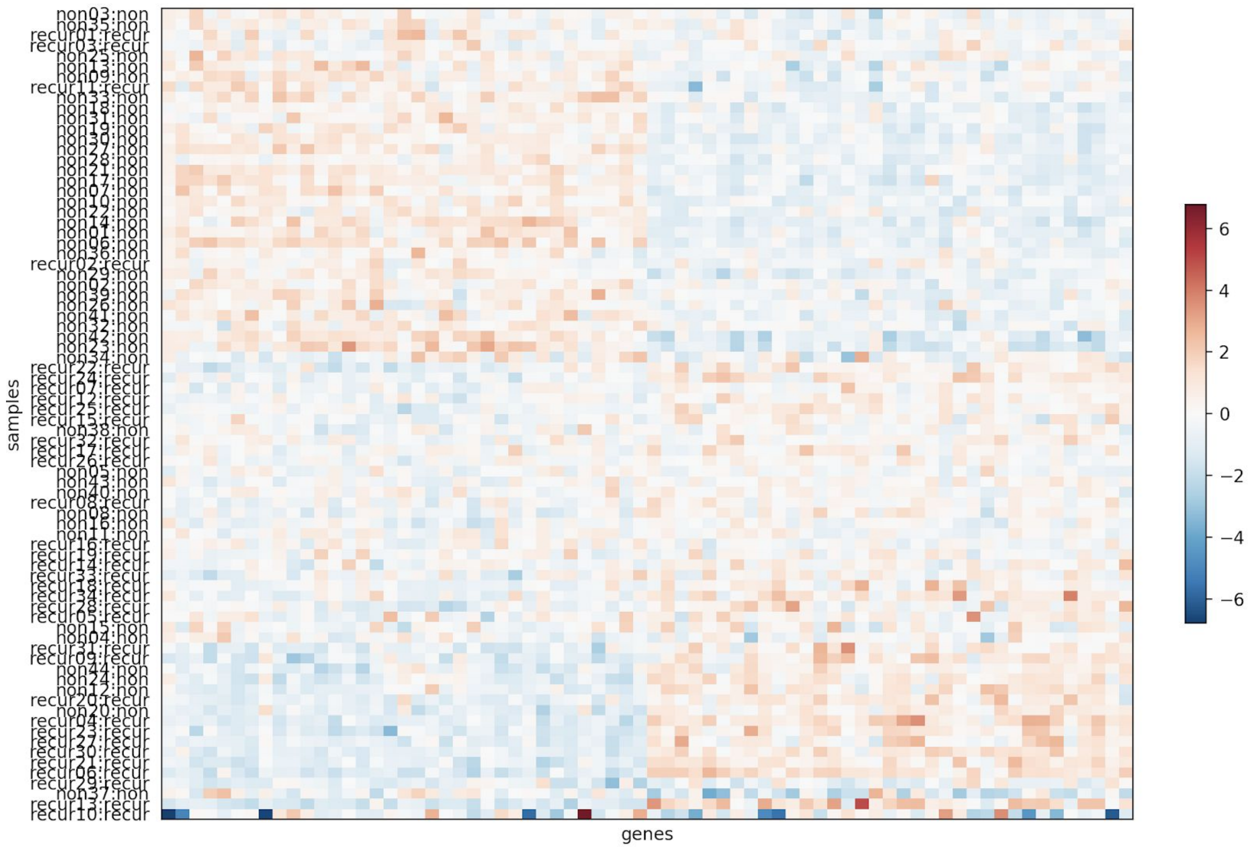
Jialing Wu
F0030kg

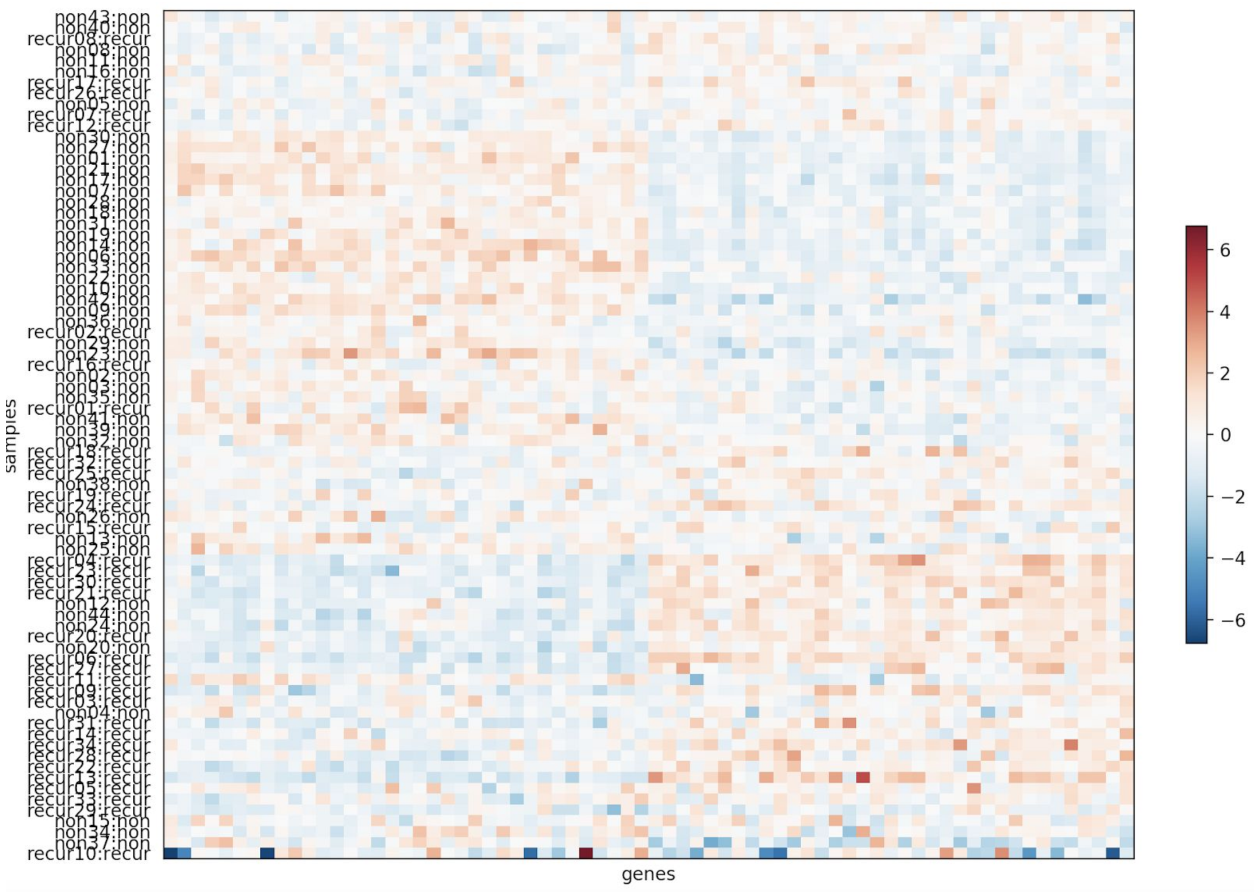
1.
Discuss the results of clustering the two real datasets, in terms of both visual patterns in the heatmap and correspondence with the known (to us, but unused by the unsupervised learning approach) class labels. How does the choice of linkage affect the results? (5 points)

Recur.csv
Average



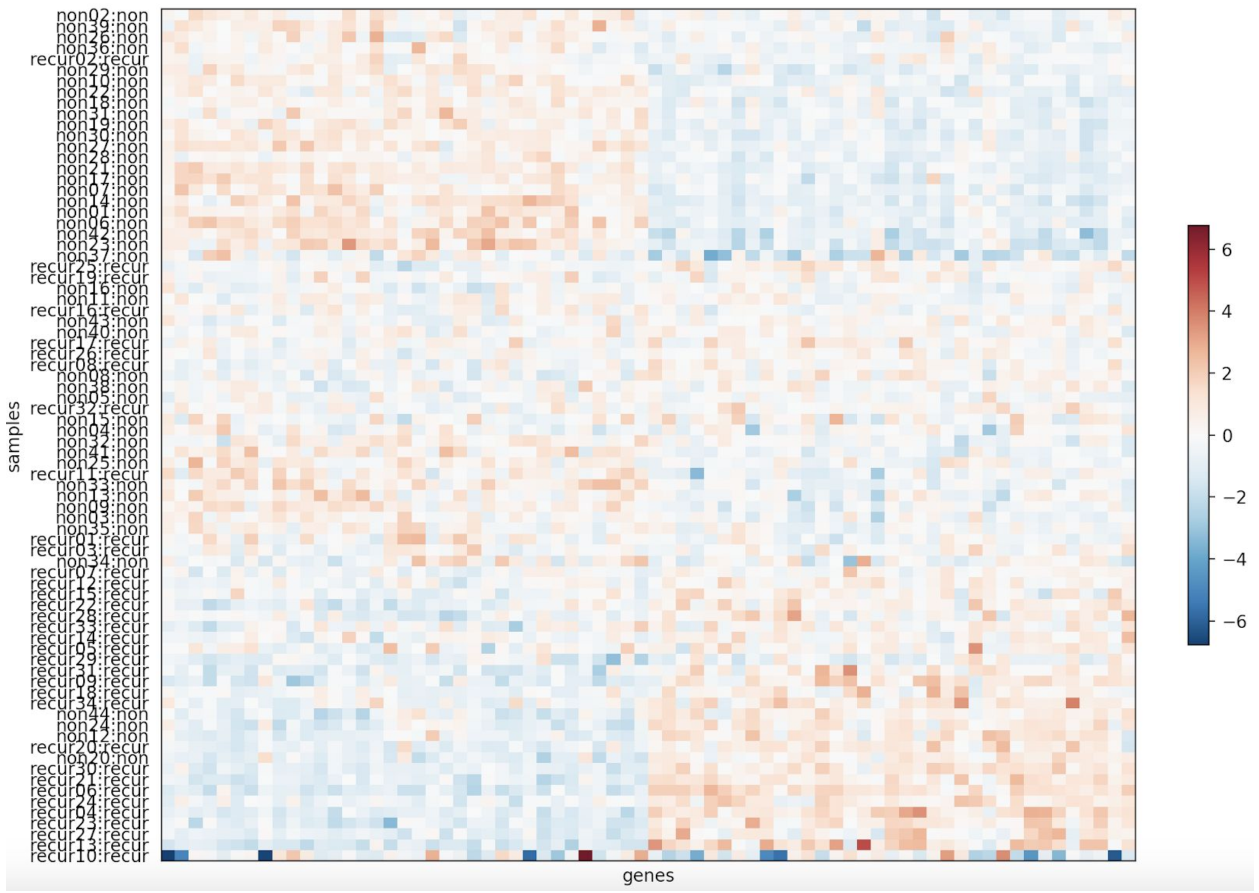
Roughly we see that upper part are (red:left, blue:right) and the labels are almost non. The bottom part are (blue:left, red:right) and the labels are mostly recur.

Min:



Roughly we see that upper part are (red:left, blue:right) and the labels are almost non. The bottom part are (blue:left, red:right) and the labels are mostly recur. Not as good as the average one.

Max:

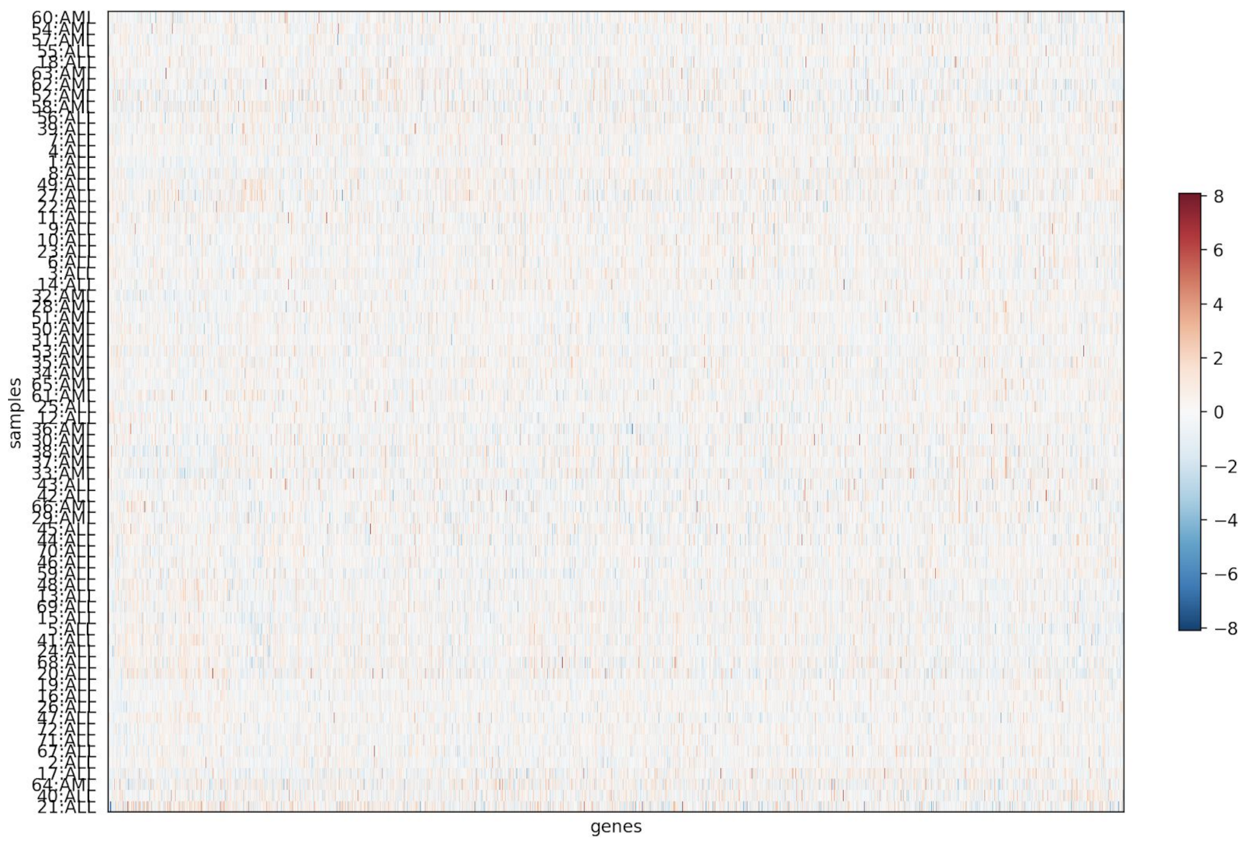


Roughly we see that upper part are (red:left, blue:right) and the labels are almost non. The bottom part are (blue:left, red:right) and the labels are mostly recur. Not as good as the average one.

For Recur.csv, the average linkage seems to be the best.

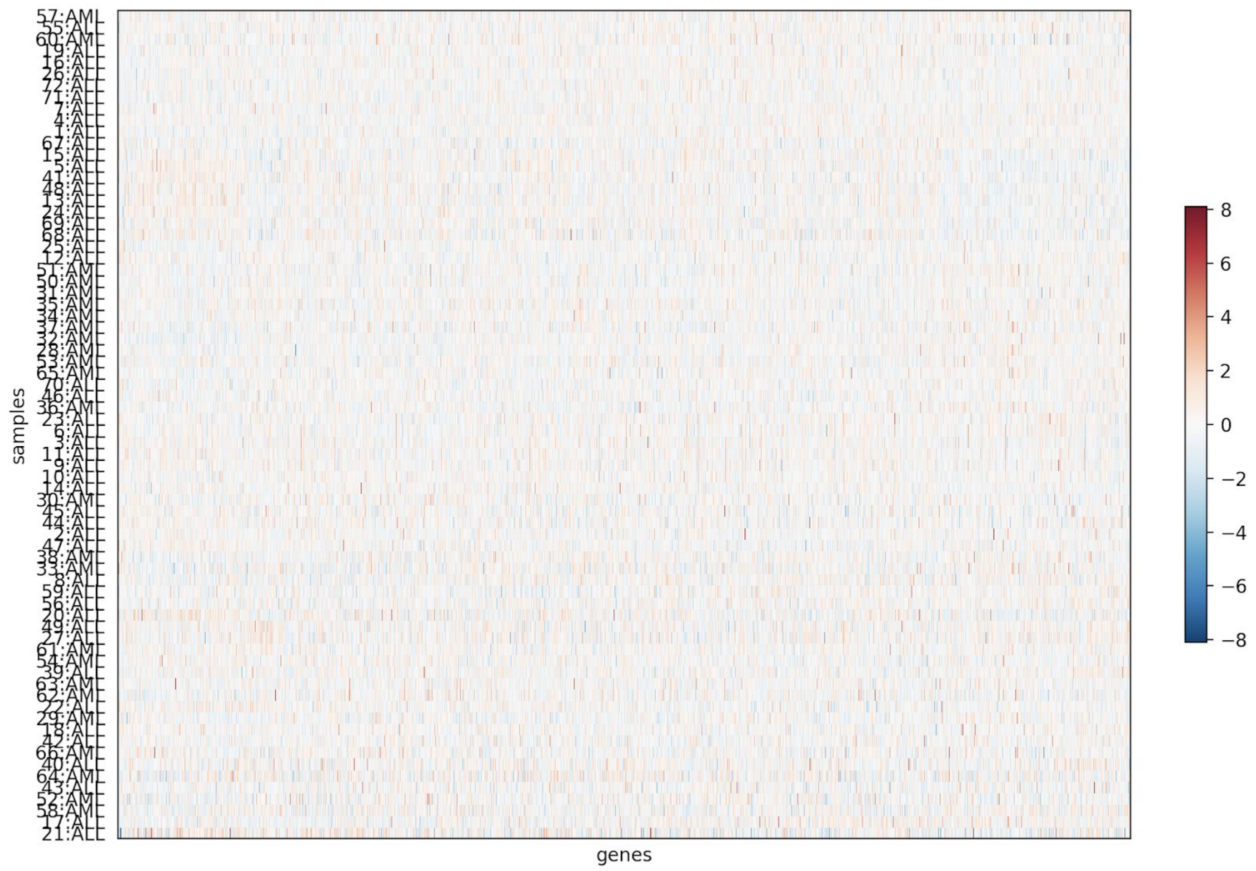
Leukemia_sm.csv

average:



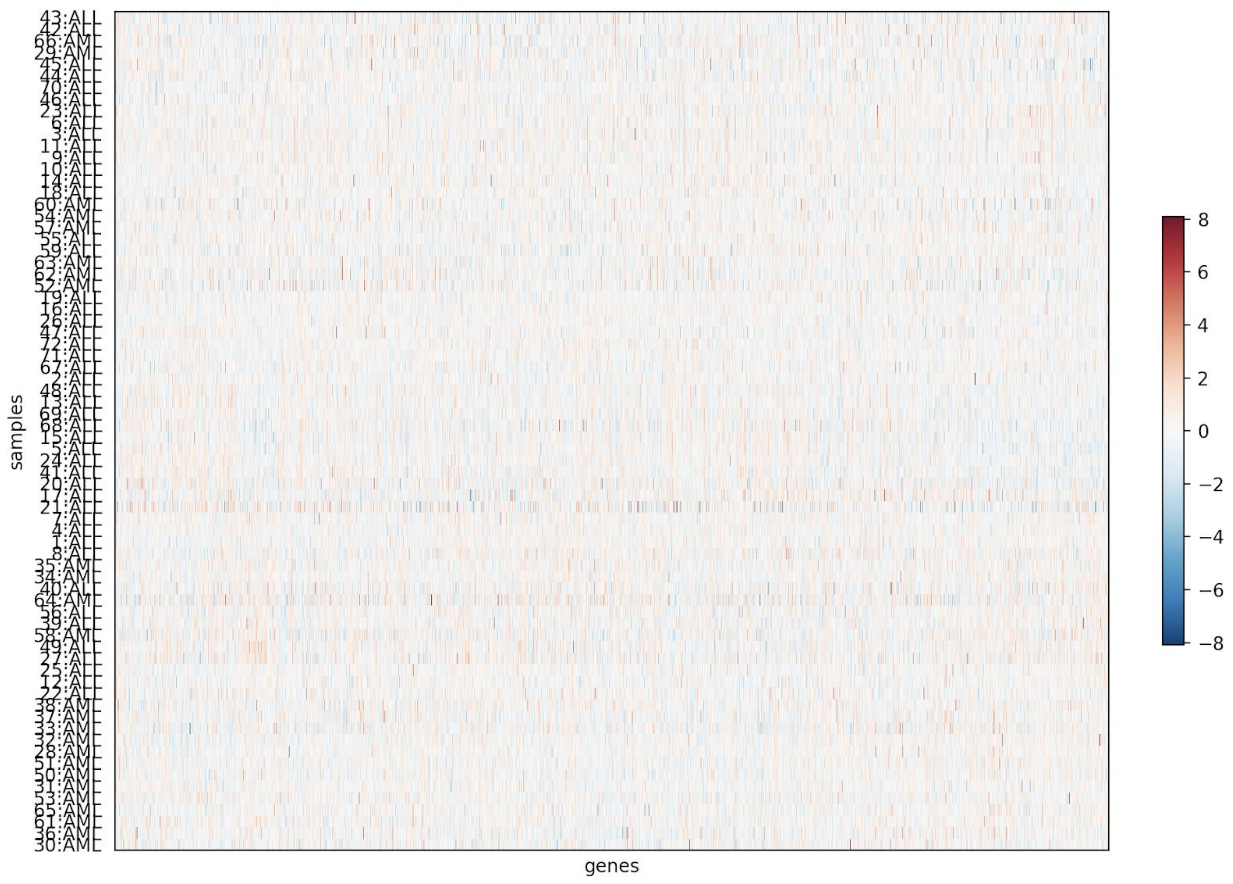
Does not work well

Min:



Does not work well

Max:



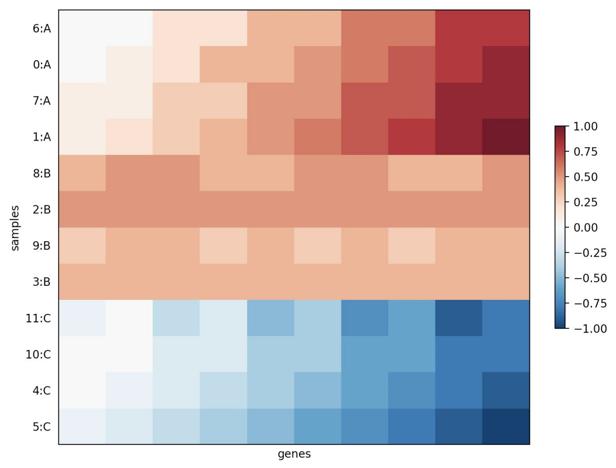
Does not work well

Because the dimension of the features are too large, the unsupervised classification does not work good.

1.

○ Devise some artificial test cases to demonstrate your code. You can use
my concoct function if you find it helpful, or come up with them some other way, or both. Test the construction of an individual tree, as well as a forest. Briefly summarize the test cases and why they give you confidence. (5 points)

I used some test case from the simple.csv. From the plot of simple.csv

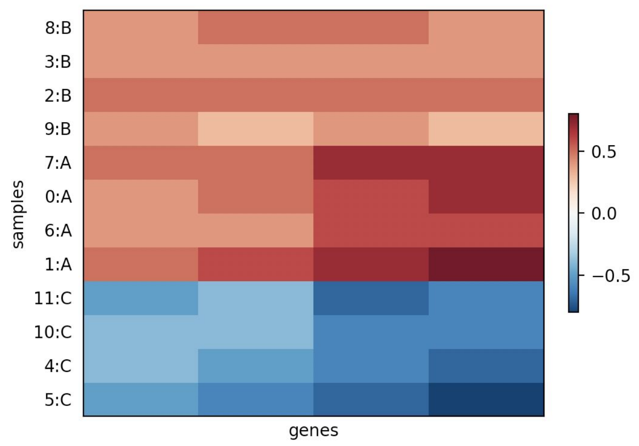


Test the decision tree
2@0.00(*C, 0@0.20(*A, *B)) looks good
6@0.55(0@0.15(*C, *B), *A) looks good
...

Test the random forest all looks good

```
2@0.00(*C, 0@0.20(*A, *B))
6@0.55(0@0.15(*C, *B), *A)
0@0.15(1@-0.05(*C, *A), *B)
0@0.20(2@0.00(*C, *A), *B)
0@0.20(1@-0.05(*C, *A), *B)
0@0.15(1@0.05(*C, *A), *B)
6@0.55(0@0.20(*C, *B), *A)
0@0.25(1@-0.05(*C, *A), *B)
0@0.25(2@0.00(*C, *A), *B)
6@0.55(0@0.15(*C, *B), *A)
2 3.67e-02
6 3.67e-02
0 2.86e-02
1 2.00e-02
```

I also only try to use only a part of the features in the simple csv to make it more difficult to classify.



Still looks good
Decision tree
0@0.00(*C, 2@0.55(*B, *A)) looks good

Random forest

```
0@0.00(*C, 2@0.55(*B, *A))
0@0.00(*C, 2@0.55(*B, *A))
0@-0.05(*C, 2@0.55(*B, *A))
2@0.55(0@0.00(*C, *B), *A)
0@0.00(*C, 2@0.55(*B, *A))
2@0.55(0@-0.05(*C, *B), *A)
0@0.00(*C, 2@0.55(*B, *A))
0@-0.05(*C, 2@0.55(*B, *A))
2@0.55(0@0.00(*C, *B), *A)
0@0.00(*C, 2@0.55(*B, *A))
0 3.67e-02
2 2.86e-02
[('0:A', 'A'), ('1:A', 'A'), ('2:B', 'B'), ('3:B', 'B'), ('4:C', 'C'), ('5:C', 'C'), ('6:A', 'A'), ('7:A', 'A'), ('8:B', 'B'), ('9:B', 'B'), ('10:C', 'C'), ('11:C', 'C')]
```

The accuracy is 100%

This gives me confidence.

○ Train forests for the two real datasets we've been using (leukemia_sm and recur). Artificially hold out some of the data, partitioning into training and testing as was done for the in-class assignment, and see how well the forest performs, over a few such splits. Examine the features and the consistency (or not) of the most important ones. Briefly summarize your findings. (5 points)

Use 80% for training and 20% for testing:

On recur.csv:
Use 10 trees, minsize 2, max_splits 4, nfeats_test 10

Trees:
7@0.24(33@0.72(21@-0.36(0@1.12(*recur, *non), 55@0.50(*recur, *non)),
21@-0.00(1@-1.22(*non, *recur), *non)), 32@-0.25(59@0.41(*non, 19@0.14(*recur, *non)),
*non))
0@-0.14(51@-0.23(23@-0.77(*recur, *non), 52@-0.93(8@0.42(*recur, *non), *recur)),
56@0.23(35@0.53(33@1.81(*non, *recur), 0@0.53(*recur, *non)), 47@0.14(27@0.85(*recur,
*non), *non)))
50@-0.14(45@1.44(4@-1.07(1@0.29(*recur, *non), 56@0.73(*non, *non)), *recur),
15@-0.46(*recur, 30@-0.15(69@-0.53(*non, *recur), 49@0.30(*non, *recur))))
12@0.30(51@0.38(30@-0.11(21@0.18(*recur, *non), 22@-1.91(*recur, *non)), 3@1.19(*recur,
*non)), 32@-0.26(26@0.10(*recur, *non), *non))
56@0.15(6@-1.05(*recur, 39@1.01(19@-1.43(*recur, *non), *recur)),
1@0.67(28@-1.02(18@-0.11(*recur, *non), 19@1.37(*recur, *non)), *non))
0@-0.14(50@-0.80(*non, 30@-0.08(*recur, 7@0.55(*recur, *non))),
29@-0.32(3@-0.16(2@-0.25(*non, *recur), *recur), 10@-1.25(*recur, 16@2.07(*non, *recur))))
0@-0.14(12@0.39(5@0.83(17@0.16(*recur, *recur), *non), 19@-0.83(*recur, *non)),
13@0.01(11@0.49(19@-1.07(*recur, *non), 1@0.62(*recur, *non)), 8@-0.71(*recur, *non)))
18@-0.03(1@0.63(3@0.11(7@1.27(*recur, *non), 52@-0.09(*recur, *non)), 53@0.41(*non,
*recur)), 59@0.36(19@-1.43(*recur, *non), 41@-0.21(*non, 11@-1.31(*non, *recur))))
29@-0.23(28@0.41(42@-0.87(*non, 25@-1.16(*recur, *recur)), 15@0.39(*non, 0@-0.65(*non,
*recur))), 67@-0.57(*non, 32@-0.34(*recur, 11@-0.55(*recur, *non))))
18@-0.03(3@0.11(64@1.08(50@-0.50(*non, *recur), 9@-0.63(*recur, *non)),
53@0.41(53@-0.96(*recur, *non), 10@-0.96(*non, *recur))), 69@0.60(0@1.28(37@0.84(*non,
*recur), *recur), 10@-0.37(*non, *recur)))

We see that the trees uses different features to split. There is some consistency in the important features. We can see some important features occur in multiple trees, like 50, 12; while some not.

On the testing set, achieves accuracy {'non': 0.7795454545454545, 'recur': 0.6323529411764706}

On recur.csv:
Use 10 trees, minsize 2, max_splits 4, nfeats_test 50

Trees:
842@0.71(1432@0.59(735@1.35(*ALL, *AML), 492@-0.26(*AML, *ALL)), 144@-2.11(*ALL,
*AML))

```
2769@0.27(375@-0.42(394@0.04(285@-1.02(*AML, *ALL), *AML), 115@1.41(*ALL, *AML)),
873@-0.01(*AML, *ALL))
656@-0.71(1004@0.58(189@-1.97(*ALL, *AML), *ALL), 955@0.55(632@1.42(*ALL,
21@0.44(*ALL, *AML)), *AML))
430@0.01(1771@1.09(397@1.65(*ALL, *AML), *AML), 2194@0.04(*AML, 1419@0.91(*ALL,
*AML)))
955@0.49(2910@-1.00(106@1.54(*AML, *ALL), *ALL), *AML)
3358@0.82(1282@-0.17(2284@0.03(1292@0.62(*AML, *ALL), *ALL), 955@0.55(*ALL,
*AML)), *AML)
1013@0.19(1052@0.60(*ALL, 238@-0.55(*ALL, *AML)), 1148@-0.43(177@0.32(*ALL, *AML),
860@1.01(*AML, *ALL)))
2506@0.07(1287@2.25(*ALL, *AML), 218@0.21(2322@1.16(*AML, *ALL), *ALL))
978@0.47(177@1.26(1151@-0.98(*AML, *ALL), *AML), 1062@1.91(*AML, *ALL))
2144@0.44(3159@1.19(1629@2.04(446@-1.53(*ALL, *ALL), *AML), *AML),
3522@1.06(768@-1.51(*ALL, *AML), *ALL))
```

We see that the trees uses different features to split. There is no consistency in the important features. I think it is because there are too many features.

On the testing set, achieves accuracy {'ALL': 0.9744680851063829, 'AML': 0.86}

2.
○ Now put the real datasets through cross-validation evaluation (10 repeats of 5-fold is fine, or vary if you like). Discuss performance over different choices for the random forest parameters. (5 points)

On recur.csv:
Use 10 trees, minsize 2, max_splits 4, nfeats_test 10
Accuracy {'non': 0.7795454545454545, 'recur': 0.6323529411764706}
Use 10 trees, minsize **3**, max_splits 4, nfeats_test 10
Accuracy {'non': 0.7613636363636364, 'recur': 0.6352941176470588}
Use 10 trees, minsize 3, max_splits **5**, nfeats_test 10
Accuracy {'non': 0.7613636363636364, 'recur': 0.65}
Use 10 trees, minsize 3, max_splits 5, nfeats_test **30**
Accuracy {'non': 0.7840909090909091, 'recur': 0.5941176470588235, }
Use **50** trees, minsize 3, max_splits 5, nfeats_test 10 **accuracy improved**
Accuracy {'non': 0.7954545454545454, 'recur': 0.7}

On leukemia
Use 10 trees, minsize 2, max_splits 4, nfeats_test 10
Accuracy {'AML': 0.796, 'ALL': 0.9680851063829787}

Use 10 trees, minsize **3**, max_splits 4, nfeats_test 10
Accuracy {'AML': 0.804, 'ALL': 0.9680851063829787}
Use 10 trees, minsize 3, max_splits **5**, nfeats_test 10
Accuracy {'AML': 0.8, ‘ALL': 0.9659574468085106}
Use 10 trees, minsize 3, max_splits 5, nfeats_test **30 accuracy improved**
Accuracy {'AML': 0.864, 'ALL': 0.9659574468085106}
Use **50** trees, minsize 3, max_splits 5, nfeats_test 50 **accuracy further improved**
Accuracy {'AML': 0.916, 'ALL': 0.9872340425531915}

Discussion:
More trees: slower but more accurate
Higher minsize: higher training accuracy but may be overfitting. Depending on the size of dataset
Lower minsize: may be underfitting.
Higher max_splits: higher training accuracy but may be overfitting. Depending on the size of dataset
Higher minsize: may be underfitting.
Higher nfeats_test: slower, maybe overfitting. **Better to set it as sqrt(num_features)**
Higher nfeats_test: quicker, may be underfitting.