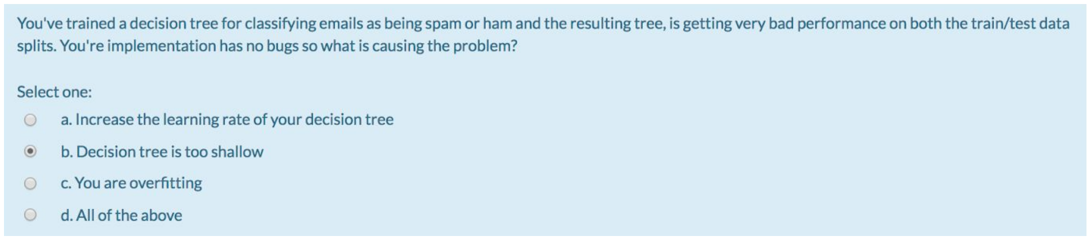
CS 146 - HW1

Jie-Yun Cheng
004460366

////////////////////////////
// Problem #1 //
////////////////////////////

(1)



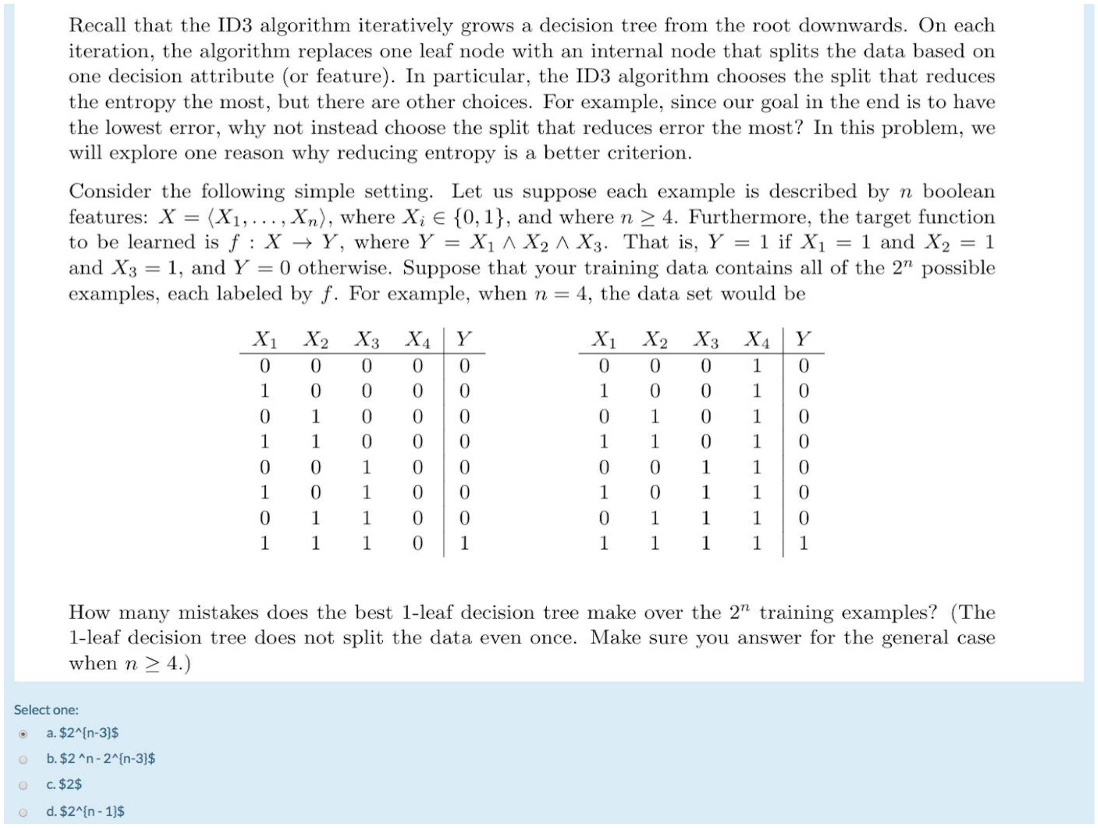
You've trained a decision tree for classifying emails as being spam or ham and the resulting tree, is getting very bad performance on both the train/test data splits. You're implementation has no bugs so what is causing the problem?

Select one:

- a. Increase the learning rate of your decision tree
- [✓ b. Decision tree is too shallow]
- c. You are overfitting
- d. All of the above

If a decision tree has very bad performance on both the train/ test data splits, then it can't be caused by overfitting, because overfitting would only cause bad performance on the test data splits. And increasing the learning rate of decision tree would have caused overfitting fast. Hence, the only possible answer is (b) the decision tree is too shallow. We can increase the number of features in the decision tree, such as having a pink color means ham.

(2)



Recall that the ID3 algorithm iteratively grows a decision tree from the root downwards. On each iteration, the algorithm replaces one leaf node with an internal node that splits the data based on one decision attribute (or feature). In particular, the ID3 algorithm chooses the split that reduces the entropy the most, but there are other choices. For example, since our goal in the end is to have the lowest error, why not instead choose the split that reduces error the most? In this problem, we will explore one reason why reducing entropy is a better criterion.

Consider the following simple setting. Let us suppose each example is described by n boolean features: $X = \langle X_1, \ldots, X_n \rangle$, where $X_i \in \{0, 1\}$, and where $n \geq 4$. Furthermore, the target function to be learned is $f : X \rightarrow Y$, where $Y = X_1 \wedge X_2 \wedge X_3$. That is, $Y = 1$ if $X_1 = 1$ and $X_2 = 1$ and $X_3 = 1$, and $Y = 0$ otherwise. Suppose that your training data contains all of the 2^n possible examples, each labeled by f. For example, when $n = 4$, the data set would be

X_1	X_2	X_3	X_4	Y
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	1

X_1	X_2	X_3	X_4	Y
0	0	0	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	0
0	0	1	1	0
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

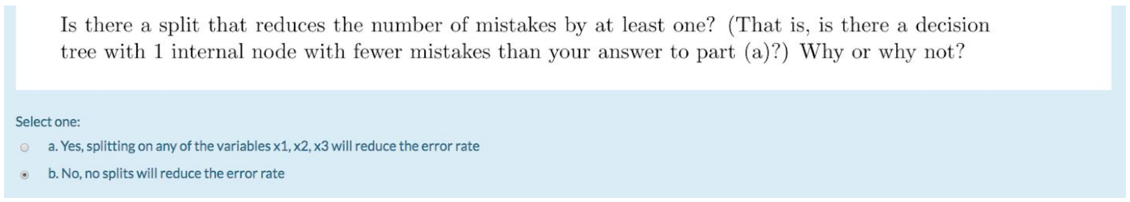
How many mistakes does the best 1-leaf decision tree make over the 2^n training examples? (The 1-leaf decision tree does not split the data even once. Make sure you answer for the general case when $n \geq 4$.)

Select one:

- [✓ a. 2^{n-3}]
- b. $2 ^n - 2^{n-3}$
- c. 2
- d. $2^{n - 1}$

The best one-leaf decision tree would have said that 100% of the time, Y = 0. For n = 4, there'd be 2 mistakes. The only mistakes that would occur would be when x1, x2, and x3 are all 1's, and the other variables can be anything. Hence, x4 can be 0 or 1, so there are 2 mistakes. For n = 5, (x4,x5) can be (0,0), (0,1), (1,0), or (1,1). And for any n above 4, the number of mistakes would have been (a) 2^(n-3)

(3)



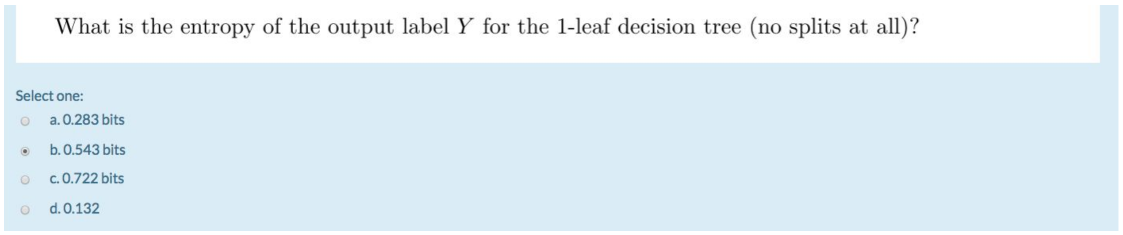
Is there a split that reduces the number of mistakes by at least one? (That is, is there a decision tree with 1 internal node with fewer mistakes than your answer to part (a)?) Why or why not?

Select one:

- a. Yes, splitting on any of the variables x1, x2, x3 will reduce the error rate
- [✓ b. No, no splits will reduce the error rate]

(b) No, splitting on any single variable would not reduce the error rate. Only when splitting on x1^x2^x3=1 would reduce the error rate.

(4)



What is the entropy of the output label Y for the 1-leaf decision tree (no splits at all)?

Select one:

- a. 0.283 bits
- [✓ b. 0.543 bits]
- c. 0.722 bits
- d. 0.132

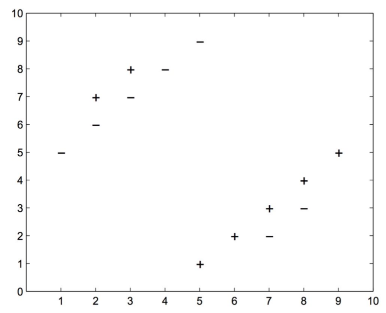
Entropy is $H =- p * log_2(p) - q * log_2(q)$. For a one-leaf decision tree here, there are p = 2/16 that outputs 1, and q = 14/16 that outputs 0. Hence, the entropy here would be $H =- \frac{2}{16} * log_2(\frac{2}{16}) - \frac{14}{16} * log_2(\frac{14}{16}) = 0.5435$. The answer is (b) 0.543 bits.

///////////////////////////
// Problem #2 //
///////////////////////////

If the ratio $\frac{p_k}{p_k+n_k}$ is the same for all k, then $\frac{p_k}{p_k+n_k} = \frac{p}{p+n}$. The information gain is

$$B(\frac{p_k}{p_k+n_k}) - \sum_{i=1}^{k} \frac{p_k+n_k}{p+n} * B(\frac{p_k}{p_k+n_k}) = B(\frac{p}{p+n}) - B(\frac{p_k}{p_k+n_k}) = 0 .$$

///////////////////////////
// Problem #3 //
///////////////////////////



(a)
To minimize the training set error for this particular data set, because the nearest neighbor for the negative data point at (2,6) for example, has the distance of 1 to the nearest positive data point, but a distance of $\sqrt{2}$ to the nearest negative data point. Considering that a point can be its own neighbor, we must set k = 1, so that the training error = 0.

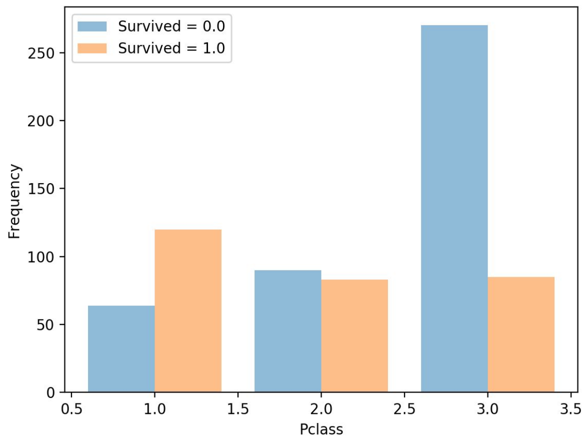
(b)
If the value of k is too large, then for the positive data point at (2,7), for example, would have been classified as a negative data point. This would increase bias, because of the lower variance, causing overfitting.

If the value of k is too small, although there would be a higher variance, the boundary would be more biased, causing underfitting.

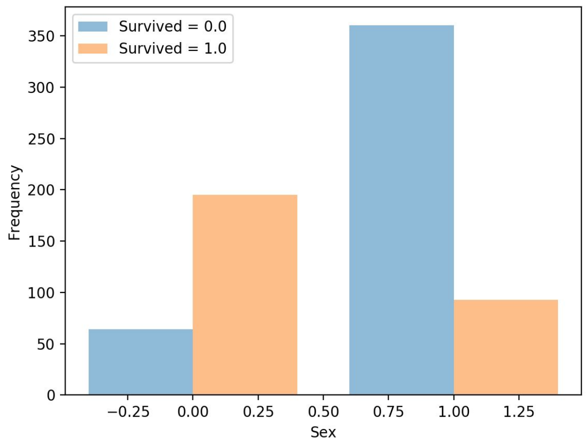
The best way is to find a value of k that is not too large or too small.

(c)
To minimize the leave-one-out cross-validation error, we need to set k = 5. This would cause errors for data points (2,7), (3,8), (7,2), and (8,3). Hence the resulting error is 4 out of 14.

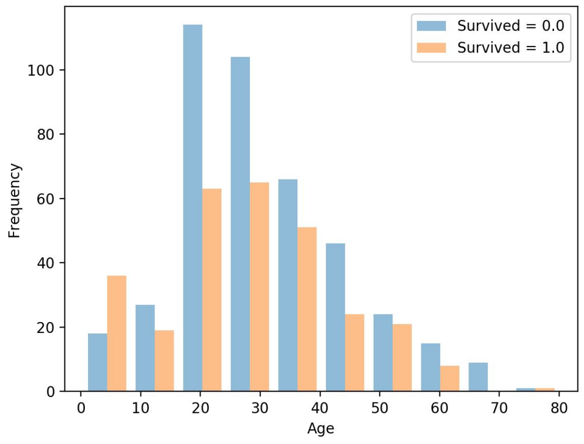
///////////////////////////////
//   Problem #4 //
///////////////////////////////
(a)



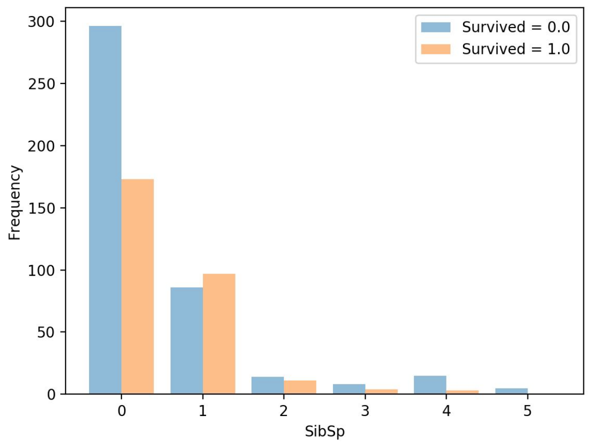
In figure 1 above, we can see that people of the higher class have a higher survival rate.



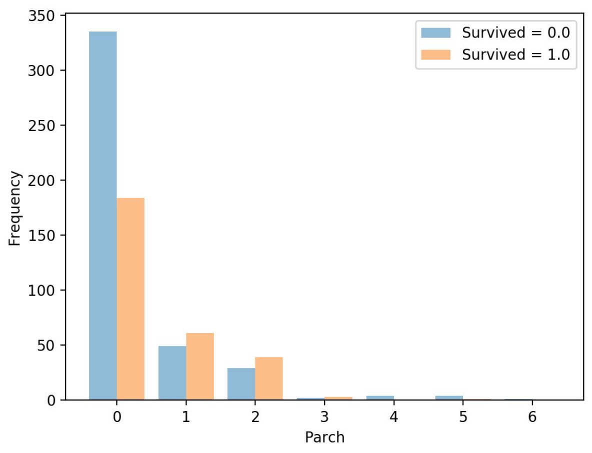
In figure 2 above, we can see that women have a much a higher survival rate than men.



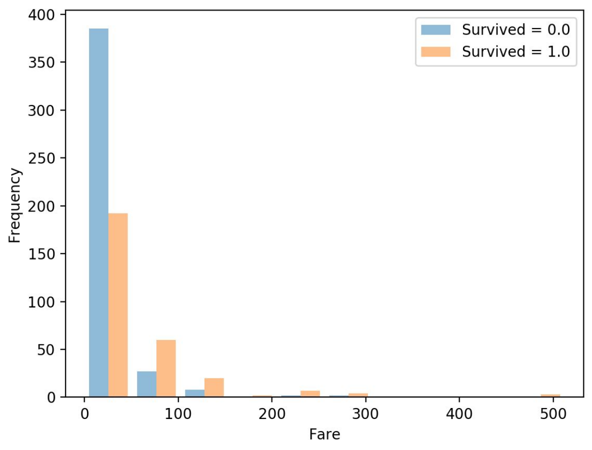
In figure 3 above, we can see that people below 10 years old have a higher survival rate.



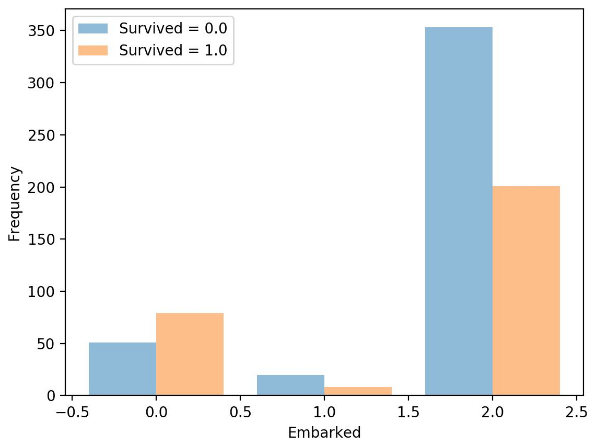
In figure 4 above, we can see that people with 1 sibling or spouse have a higher survival rate.



In figure 5 above, we can see that people with 1 or 2 parents or kids have a higher survival rate.

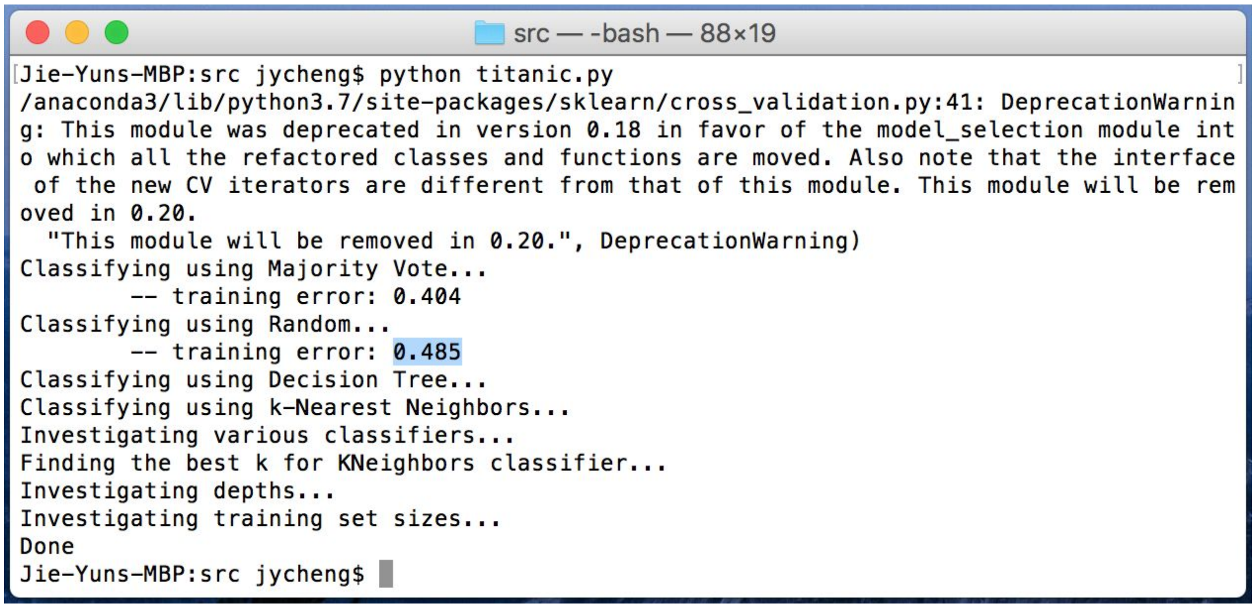


In figure 6 above, we can see that people who paid have a higher survival rate than people who didn't pay.



In figure 7 above, we can see that people who embarked from C have a higher survival rate.

(b)
I did it, and the training error for RandomClassifier was indeed 0.485. My Python code can be found on my Github:
https://github.com/chengjieyun59/UCLA_CS146_projects/blob/master/HW%201/src/titanic.py



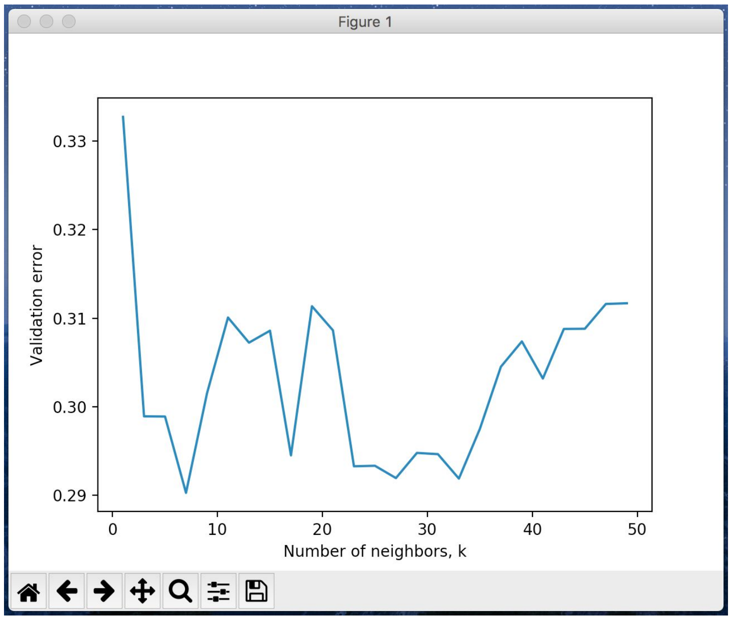
```
Jie-Yuns-MBP:src jycheng$ python titanic.py
/anaconda3/lib/python3.7/site-packages/sklearn/cross_validation.py:41: DeprecationWarnin
g: This module was deprecated in version 0.18 in favor of the model_selection module int
o which all the refactored classes and functions are moved. Also note that the interface
 of the new CV iterators are different from that of this module. This module will be rem
oved in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
Classifying using Majority Vote...
        -- training error: 0.404
Classifying using Random...
        -- training error: 0.485
Classifying using Decision Tree...
Classifying using k-Nearest Neighbors...
Investigating various classifiers...
Finding the best k for KNeighbors classifier...
Investigating depths...
Investigating training set sizes...
Done
Jie-Yuns-MBP:src jycheng$ 
```

(c)
Using entropy as the criterion, while keeping other code the same, the training error for the DecisionTree Classifier is 0.014.

(d)
Classifying using k-Nearest Neighbors...
    -- training error when k = 3: 0.167
    -- training error when k = 5: 0.201
    -- training error when k = 7: 0.240

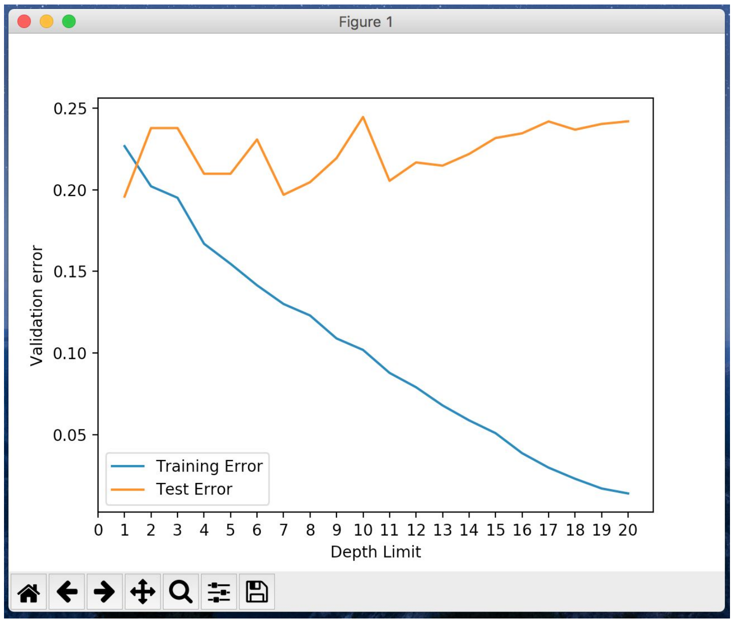
(e)
Investigating various classifiers...
    -- training error for MajorityVoteClassifier: 0.397
    -- test error for MajorityVoteClassifier: 0.434
    -- training error for RandomClassifier: 0.517
    -- test error for RandomClassifier: 0.483
    -- training error for DecisionTreeClassifier: 0.012
    -- test error for DecisionTreeClassifier: 0.241
    -- training error for KNeighborsClassifier: 0.232
    -- test error for KNeighborsClassifier: 0.336

(f)



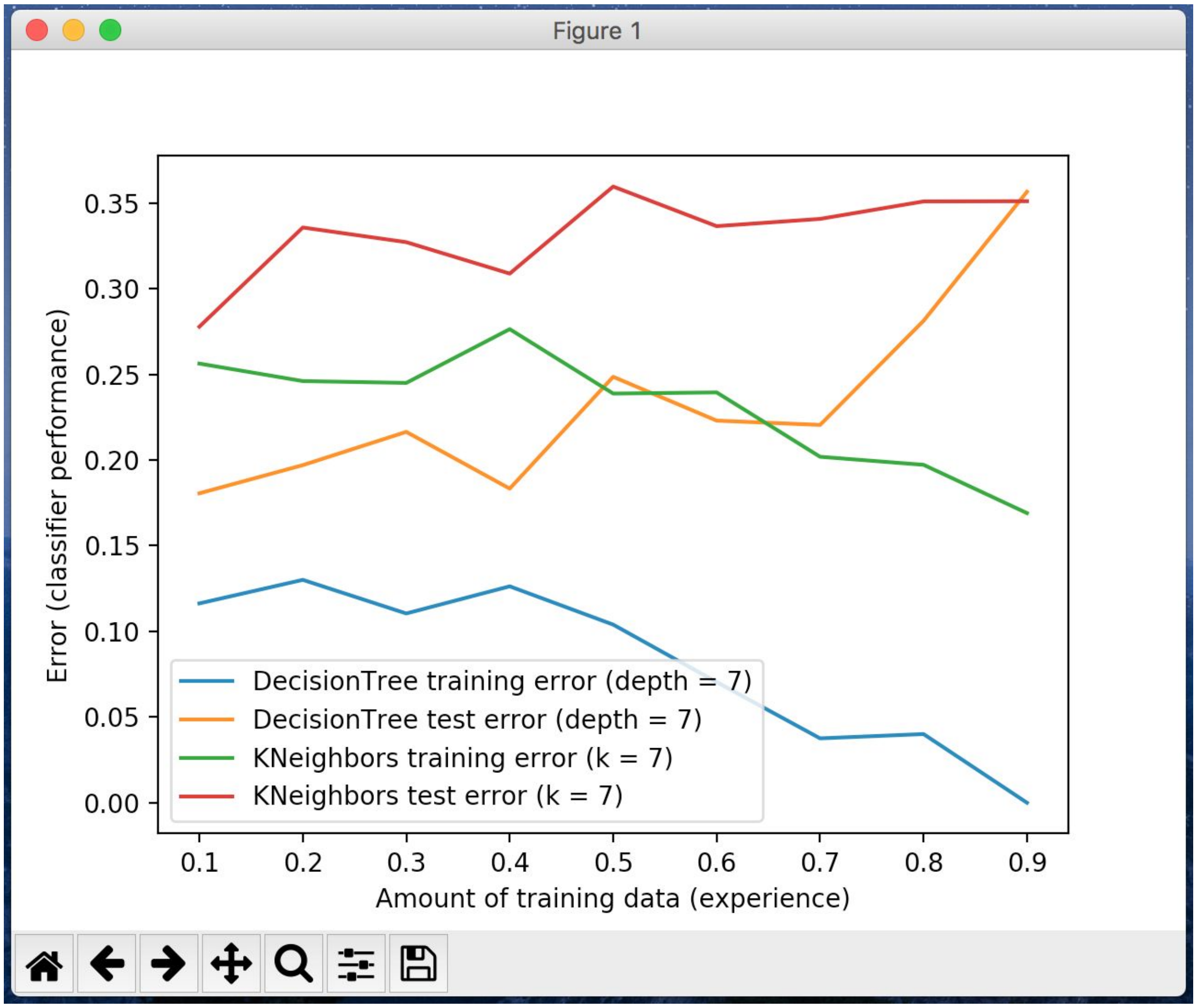
The data point that has the smallest validation error is when the number of neighbors, k = 7. Observing the graph, we can see that when k is too small or too large, there would be a lot of error. To not include too many or too few neighbors, we found the sweet spot for this example to be 7 neighbors.

(g)



The best depth limit to use for this data is 7, because it has the lowest training error. And a depth limit of 1 would have been too small and cause underfitting. Yes, I see overfitting. I observe that the test error decreases at a rather linear rate as depth increases, but the actual training error is still between 0.20 to 0.25, with a tendency to keep increasing after k = 10.

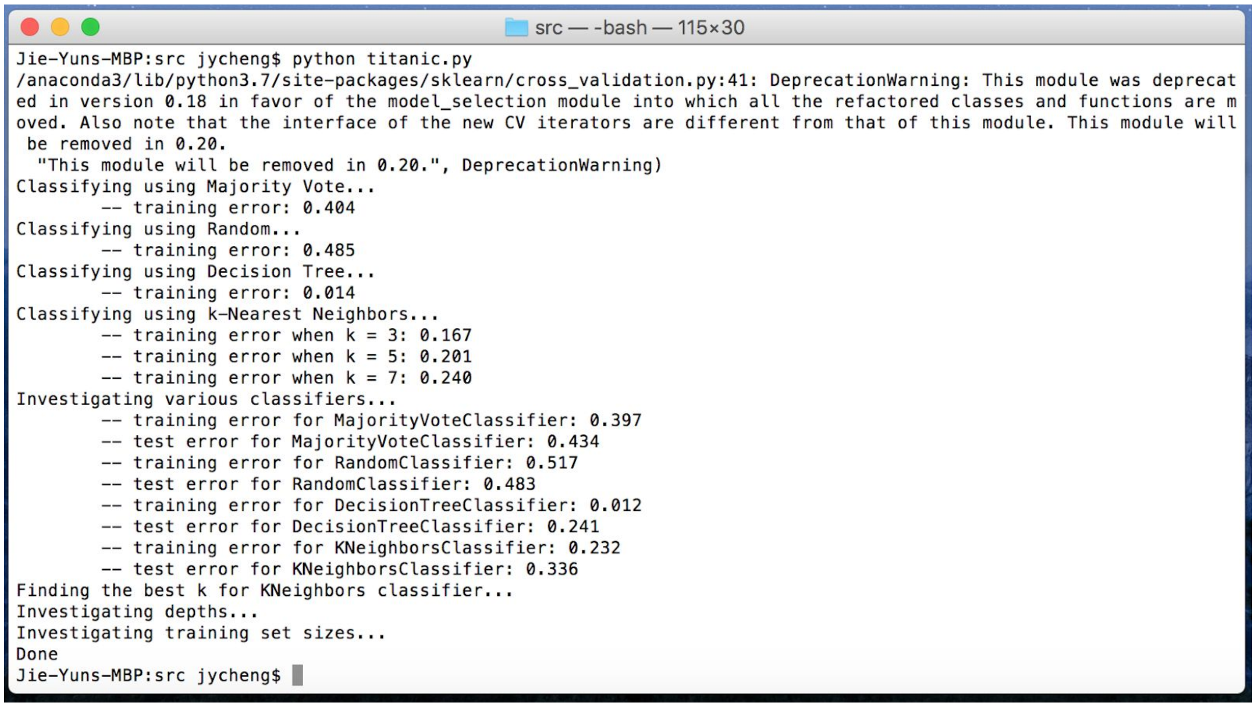
(h)



The learning curves show that for both classifiers, the test error would increase and the training error would decrease as there is more training data, because overfitting would occur. Based on the test errors, it is obvious that the decision tree classifier performs better with less error for the Titanic data, especially when we take 0.4 (e.g. 60/40 split) of the amount of data.

Proof of completion:

1. terminal commands



src — -bash — 115×30

```
Jie-Yuns-MBP:src jycheng$ python titanic.py
/anaconda3/lib/python3.7/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecat
ed in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are m
oved. Also note that the interface of the new CV iterators are different from that of this module. This module will
 be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
Classifying using Majority Vote...
        -- training error: 0.404
Classifying using Random...
        -- training error: 0.485
Classifying using Decision Tree...
        -- training error: 0.014
Classifying using k-Nearest Neighbors...
        -- training error when k = 3: 0.167
        -- training error when k = 5: 0.201
        -- training error when k = 7: 0.240
Investigating various classifiers...
        -- training error for MajorityVoteClassifier: 0.397
        -- test error for MajorityVoteClassifier: 0.434
        -- training error for RandomClassifier: 0.517
        -- test error for RandomClassifier: 0.483
        -- training error for DecisionTreeClassifier: 0.012
        -- test error for DecisionTreeClassifier: 0.241
        -- training error for KNeighborsClassifier: 0.232
        -- test error for KNeighborsClassifier: 0.336
Finding the best k for KNeighbors classifier...
Investigating depths...
Investigating training set sizes...
Done
Jie-Yuns-MBP:src jycheng$ 
```

2. Python code

https://github.com/chengjieyun59/UCLA_CS146_projects/blob/master/HW%201/src/titanic.py