1a.) For the n=4 case: 2 mistakes
n=5: 4 mistakes
n=6: 8 mistakes
For n: $2.^{n-3}$ mistakes

1b.) No, since any split would divide the data in half, assigning one half to 0 and one half to 1. This would increase the amount of errors since, even in the best case scenario, splitting by one of X_1, X_2, or X_3, there will still be $2^{n-1} - 2^{n-3}$ errors = $3 * 2^{n-3}$ or 3 times as many errors. Splitting by any other feature will make 2^{n-1} errors.

1c.) No split -> 2 zeros, 14 ones
$-(2/16\log_2(2/16) + 14/16\log_2(14/16)) = 0.543$

1d.) Each branch will have 1 zero and 7 ones
$-(⅛\log_2(⅛) + ⅞\log_2(⅞)) = 0.543$
So expected entropy = 8/16(0.543) + 8/16(0.543) = 0.543
And information gain = 0.543 - 0.543 = 0

1e.) Splitting by any of X_1, X_2, or X_3 will reduce the entropy of the output Y in table 1 by a non-zero amount.
Splitting by X_3 for example:
One branch has 2 zeros and 6 ones, the other has 8 ones.
$H(B_1) = -(2/8\log_2(2/8) + 6/8\log_2(6/8)) = 0.811$
$H(B_2) = 0$
So conditional entropy is then (8/16)(0.811) + (8/16)*0 = 0.4056

2a.) $B(q) = -q \log_2 q - (1 - q) \log_2(1 - q)$.
Take the first derivative:
$B'(q) = \log_2(1-q) - \log_2(q)$
Take the second derivative:
$B''(q) = -\frac{1}{(1-q)ln2} - \frac{1}{qln2}$

Find critical points:
$0 = \log_2(1-q) - \log_2(q)$
$q = 0.5$
Check sign on second derivative at critical point:
$B''(0.5) = -\frac{4}{ln2}$
Since the second derivative is negative, then the function is concave down at the critical point, so it is a local maximum.

2b.) If the ratio $\frac{P_k}{P_k + n_k}$ is the same for each k branch, then there is 0 information gain since the overall entropy will be the same.
$H(S) = B(\frac{P}{P+n})$, so $\frac{P}{P+n}$ is the same for each branch, then H(S) is the same for each branch, so then the conditional entropy is the same. Thus, there is no information gain since the entropy did not change.

3a.) The value of k to minimize the training set error for this dataset is k=1. Since each point can be its own neighbor, then k=1 would result in 0 training error.

3b.) Too large values of k might be bad in this dataset because a large k like k=14 would classify every point the same way. Too small values of k would be bad because it could cause overfitting to the training data.

3c.) The values of k that minimizes the leave-one-out cross-validation error for this dataset are k=5,6, and 7. Normally, we would only use KNearestNeighbors with odd k to avoid ties, so we would use k= 5 and k=7. These values have an error of 0.2857.

4.1a.) Ticket Class - Higher ticket class (1st being higher than 2nd and 2nd being higher than 3rd) had higher rates of survival. This is particularly noticeable in third class, likely because it was the largest class overall.
Sex: Assuming females were on the left, females had a noticeably higher survival rate, with the majority of them surviving.

Age: A majority of very young children (age 0-10) survived, while it seems that for most other age groups, around half as many survived as did not survive, except for age 50s, where almost as many survived as did not, and age 70s where none or almost none survived.
# of siblings/spouses aboard Titanic: The data is very skewed right, as most people had 0 siblings/spouses on the ship.
# of parents/children aboard Titanic: The data is also very skewed right, with most people having 0 parents or children on board.
Fare: The data is also very skewed right, with most of the passengers' fares being between 0 and 100.
Embarked: Most of the passengers embarked at Southampton. It seems that passengers from Cherbourg had the highest likelihood of survival, followed by Southampton, then Queenstown.

4.2b) I have implemented the RandomClassifier, and its training error is 0.485.

4.2c) The training error of the DecisionTreeClassifier is 0.014.

4.2d) The training error for the KNeighborsClassifier is as follows for varying values of k:
k=3: 0.167
k=5: 0.201
k=7: 0.240

4.2e) The average training and test error for each of the classifiers on the Titanic data set are as follows:
MajorityVoteClassifier:

- Training Error: 0.404
- Test Error: 0.407

RandomClassifier

- Training Error: 0.489
- Test Error: 0.487

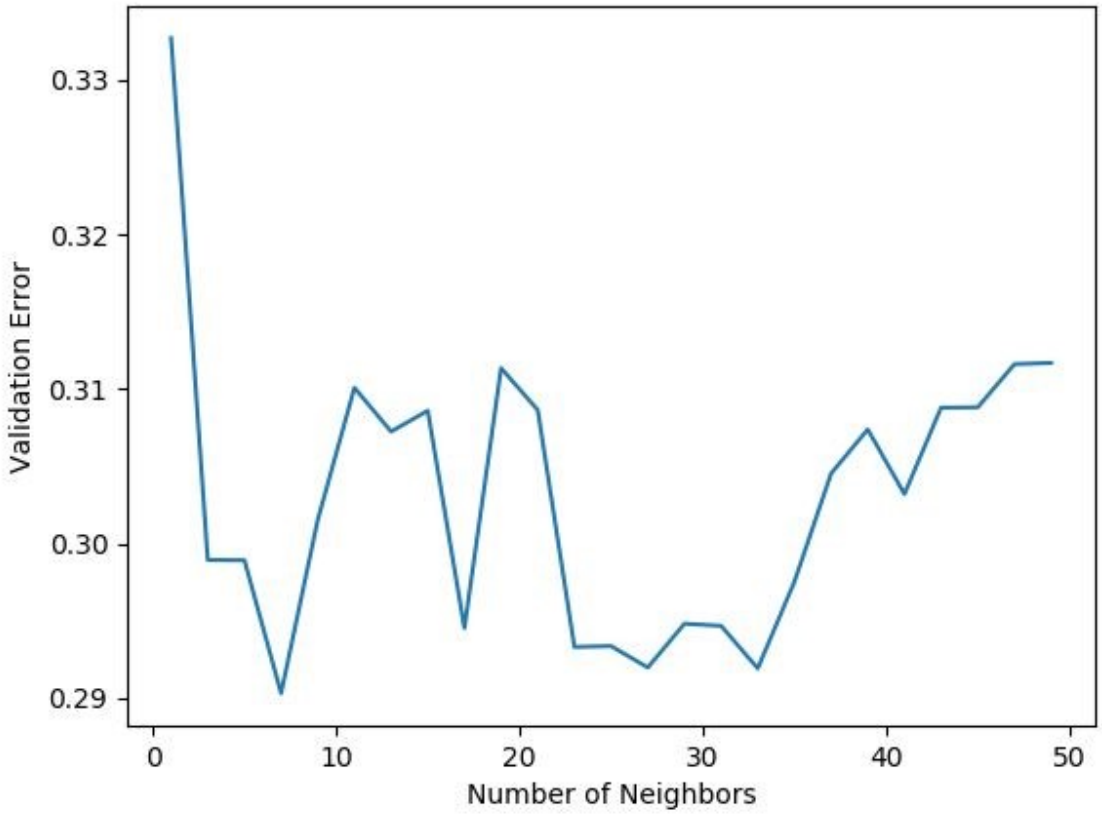
DecisionTreeClassifier

- Training Error: 0.012
- Test Error: 0.241

KNeighborsClassifier

- Training Error: 0.212
- Test Error: 0.315

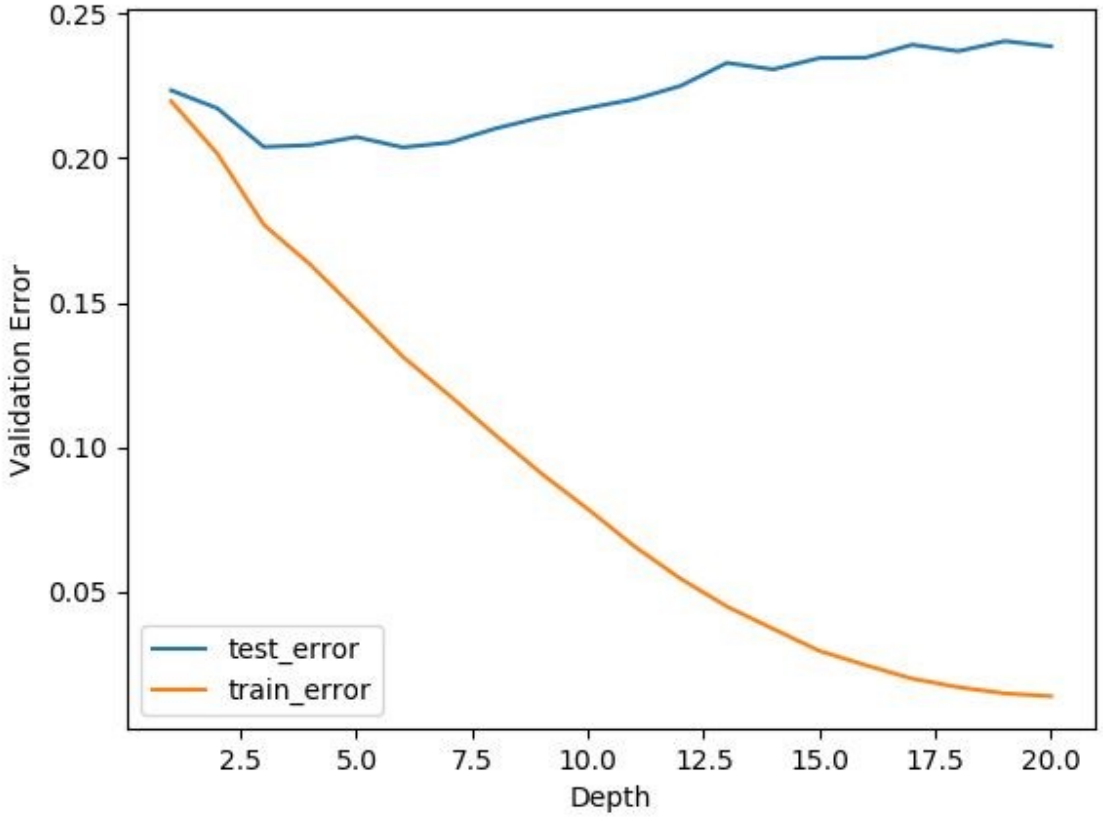
4.2f)



From the looks of it, there is a global minimum at k=7, and after a certain point (k=33) the data error continuously increases, due to using too many neighbors. Between these two points, the error fluctuates rapidly, with a sudden drop in error at k=17 and k=23.

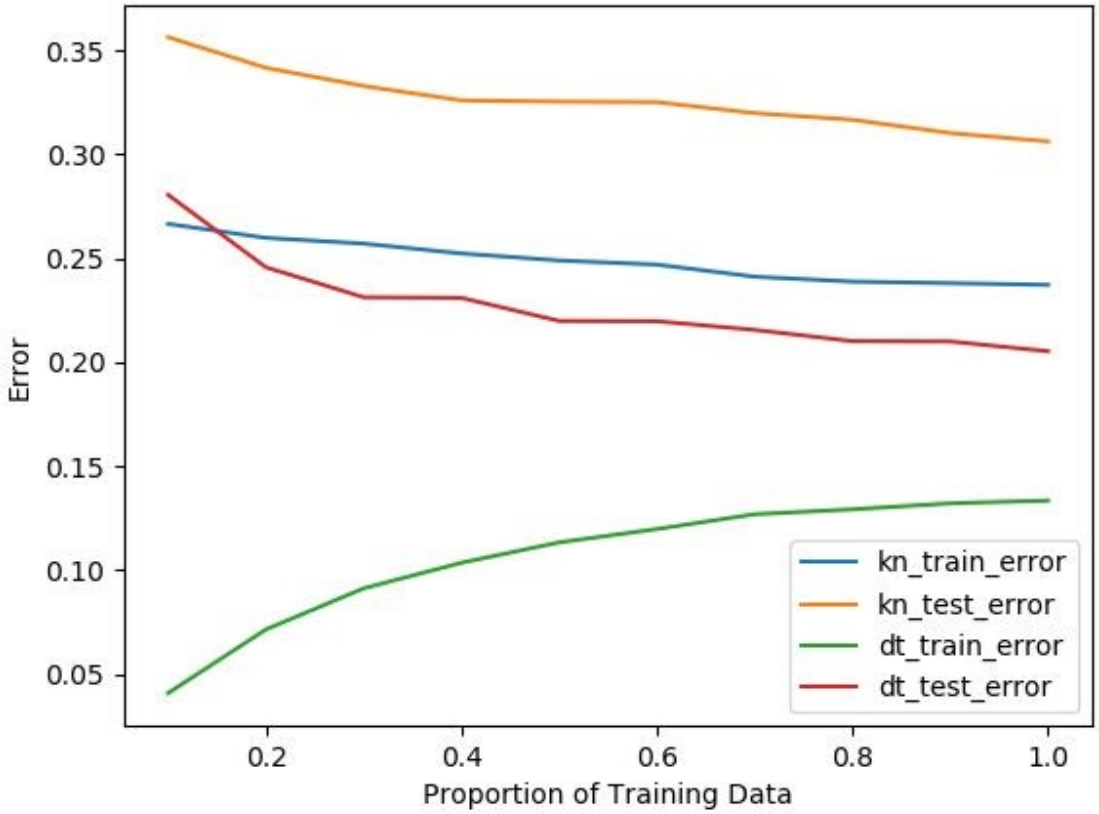
The best value of k is k=7.

4.2g)



The best depth is d=6.
There is definitely overfitting. This can be seen since any tree with a depth greater than 6 continues to reduce training error, but starts to increase testing error. This means that the model is fitting too much to the training data and stops being able to generalize well to the new testing data.

4.2h)



I found it somewhat surprising that the test error for KNeighbors didn't actually particularly much with more training. The KNeighbors training error similarly didn't change very dramatically with increased training.The DecisionTreeClassifier seemed to decrease it's test error more dramatically, most likely because the small proportion of training data was overfit on the decision tree.