Part 1:

a) k = 1 = 0.8888888888888888, k = 3 = 0.9166666666666666 (output in part 1 folder)
b) k = 1 = 1.0, k = 3 = 0.9788732394366197(output in part 1 folder)
c)

i: A k value of 1 in the training set increases the accuracy to 100% which is expected as the K nearest neighbor of k=1 would be itself. This doesn't translate to improved accuracy in the test set due to overfitting. A k value of 3 in the training set gave an accuracy of 98%. For the test set a k value of 3 gives the higher accuracy of 92% whilst a k value of 1 gives an accuracy of 89%. This shows that k = 3 in this example has a better accuracy for unseen data (test set).

ii: Using the "optimal" k value for the training set can lead to overfitting, especially if the value is small e.g. k=1. This causes the model to become overly sensitive to noise in the training data, which can decrease the ability of the model to make predictions on unseen data, giving lower accuracy on the test set. A slightly higher k value (e.g. k=3) gives a better balance by looking at more neighbors, which reduces the impact of noise and gives a more accurate prediction on the test set.

d) The pros of increasing k is that it increases the robustness of the model to outliers, and improves the generalization capability on unseen data (e.g. test set).
The cons are that it can lead to underfitting if the k value is too large, which makes the Model too simple.

The pros of decreasing k is that is can capture the complexity of the data better, which leads to more flexible decision boundaries.
The cons are that it increases the risk of overfitting as we saw in our training set by Making the model too sensitive to the noise in the training data. This could give poor performance on unseen data.

With extreme values of k, firstly we can look at k=1. With a small k value like this the Model makes predictions based only on the closest training example, which can lead to Overfitting.
With an extreme k of k=n(Total size of dataset), the decision would be the same for any Input, normally it would default to the most common class in the data.
This results in a biased and over-simple model.

Choosing the right k value is critical in getting the balance for the bias-variance tradeoff Which happens in the kNN algorithm. A k that is too small or too large will lead to poor Performance whether it be from overfitting or underfitting.
K-cross validation can help in finding the right k value that achieves the best Performance on unseen data.

Part 2:

a) Accuracy of 1, 100%(output in part 2 folder)
b) Accuracy of 1, 100%(output in part 2 folder)
c)

I. For rtg_A the attribute with the minimal entropy was “att0” which had an entropy of 0.422. For rtg_B, the initial split is done on att3 but minimal entropy for a single feature on its own is not shown like in rtg_A. Since “att3” is chosen as the root, it shows that it provided enough information gain initially, even though the entropy of the split is not minimal compared to other features on the tree. The minimal entropy among splits that you can see in the tree is for att2 with an entropy of 0.4898. There are others that also have lower entropy but the entropy for a single feature in the rtg_B dataset is not comparable to rtg_A as rtg_B has a much more complex structure.

ii.in rtg_A, all features except att3 are utilized. Looking at the data it looks like the reason for att3’s exclusion could be due to having no variability as all values for att3 are “0”. Because att3 is the same for all samples, splitting on the attribute would not reduce entropy at all, meaning it has 0 information gain. In rtg_B however all attributes are included.