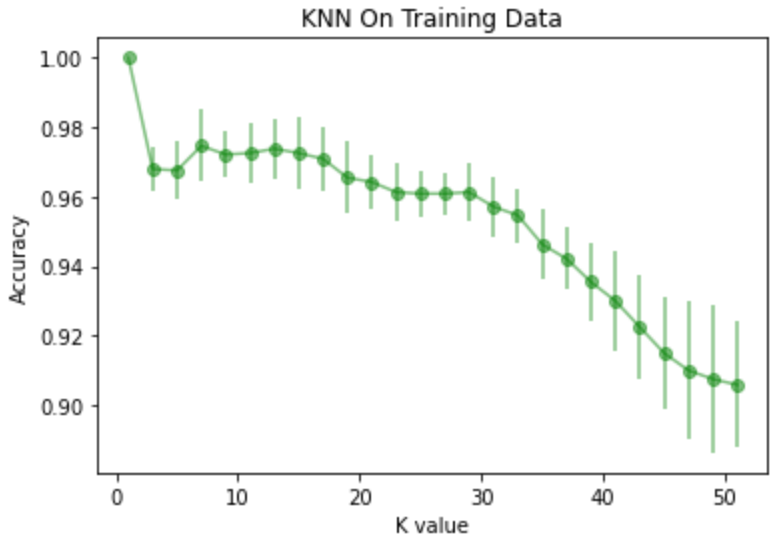
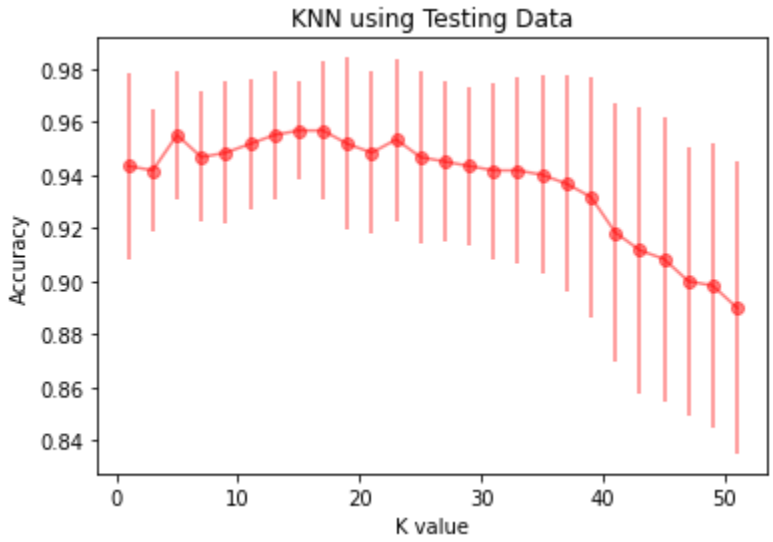
Q1.1 (10 Points) In the first graph, you should show the value of k on the horizontal axis, and on the vertical axis, the average accuracy of models trained over the training set, given that particular value of k. Also show, for each point in the graph, the corresponding standard deviation; you should do this by adding error bars to each point. The graph should look like the one in Figure 2 (though the "shape" of the curve you obtain may be different, of course).



Q1.2 (10 Points) In the second graph, you should show the value of k on the horizontal axis, and on the vertical axis, the average accuracy of models trained over the testing set, given that particular value of k. Also show, for each point in the graph, the corresponding standard deviation by adding error bars to the point.



Q1.3 (8 Points) Explain intuitively why each of these curves look the way they do. First, analyze the graph showing performance on the training set as a function of k. Why do you think the graph looks like that? Next, analyze the graph showing performance on the testing set as a function of k. Why do you think the graph looks like that?

The first graph shows the performance of the KNN algorithm on the training set as a function of k. The graph shows that the accuracy initially decreases as k increases from 1 to around 5-7, and then starts to level and even slightly increases as k continues to increase. This nature can be explained by the bias-variance tradeoff. When k is small (k=1), the model has low bias because it can fit the training data tightly, but high variance because it is sensitive to noise in the data. As k increases, the model has higher bias because it averages over more neighbors and may not fit the training data as tightly, but lower variance because it is less sensitive to noise.

The second graph shows the performance of the KNN algorithm on the testing set as a function of k. Unlike the first graph, the accuracy initially increases as k increases from 1 to around 13-16, and then starts to decrease as k continues to increase. This can be explained by the fact that when k is small, the model may overfit the training data and not generalize the new data quite well. As k increases, the model is able to generalize to new data better because it becomes less sensitive to noise. But if k is too large, the model may oversimplify the decision boundary and not be able to return accurate results. Therefore, there is a value of k that maximizes the accuracy on the testing set. The decreasing trend in accuracy for larger k values are the result of the model becoming too simplistic.

Q1.4 (6 Points) We say that a model is underfitting when it performs poorly on the training data (and most likely on the testing data as well). We say that a model is overfitting when it performs well on training data but it does not generalize to new instances. Identify and report the ranges of values of k for which k-NN is underfitting, and ranges of values of k for which k-NN is overfitting.

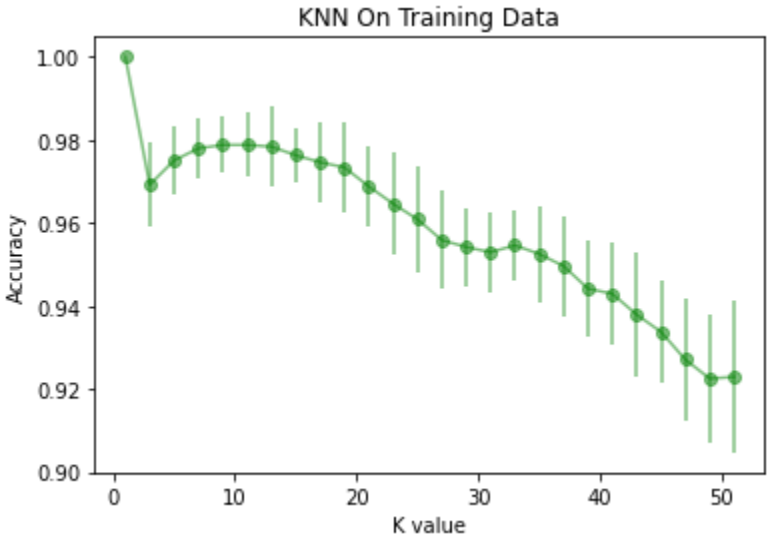
We can observe from the results that as the value of k increases, the performance on the training data decreases. This allows us to conclude that the model is underfitting for larger values of k on training data. Also, as the value of k decreases, the performance on the testing data decreases after reaching a certain value of k. This allows us to conclude that the model is overfitting for smaller values of k on testing data.

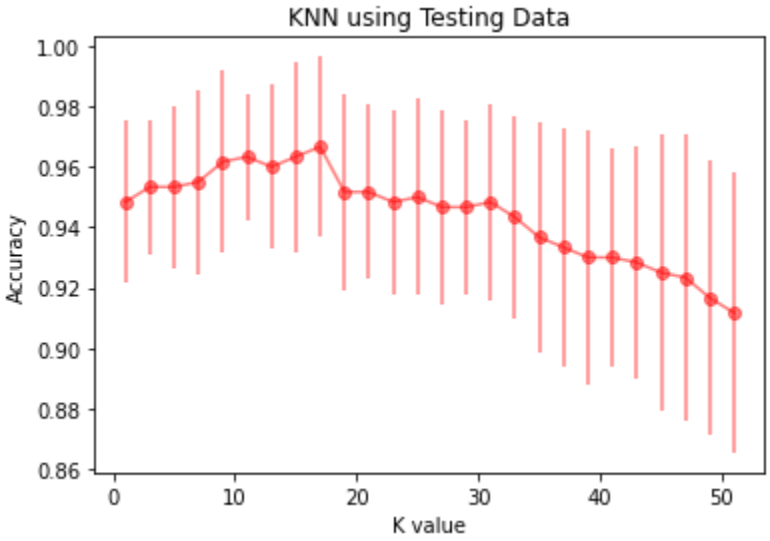
Therefore, we can conclude that the model is underfitting for larger values of k (k>10) and overfitting for smaller values of k (k<4). I would state that the optimal value of k seems to be in the range of 4-10 where the performance on both the training and testing sets is high.

Q1.5 (6 Points) Based on the analyses made in the previous question, which value of k you would select if you were trying to fine-tune this algorithm so that it worked as well as possible in real life? Justify your answer.

Based on the analyses made in the previous question, we can see that as the value of k increases, the performance of KNN on the training set decreases. On the other hand, the performance on the testing set initially increases, and then starts to decrease after a certain value of k. This suggests that we need to consider a balance between the bias and variance of the algorithm.
A small value of k (such as k=1) tends to overfit the training data, as the algorithm is too biased on the training set and does not generalize well to new instances. On the other hand, a large value of k (k=25) tends to underfit the training data, as the algorithm is too general.
Therefore, to fine-tune this algorithm in order to get the most optimal results, we would select a value of k that is within the range k=7 to k=15 where the performance on the testing set is highest. This value of k provides a good balance between bias and variance.

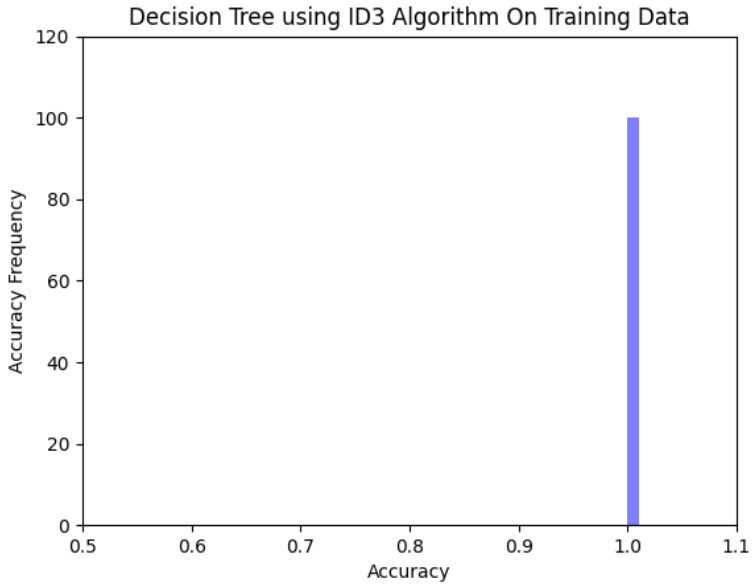
Q1.6 (10 Points) In the experiments conducted earlier, you normalized the features before running k-NN. This is the appropriate procedure to ensure that all features are considered equally important when computing distances. Now, you will study the impact of omitting feature normalization on the performance of the algorithm. To accomplish this, you will repeat Q1.2 and create a graph depicting the average accuracy (and corresponding standard deviation) of k-NN as a function of k, when evaluated on the testing set. However, this time you will run the algorithm without first normalizing the features. This means that you will run k-NN directly on the instances present in the original dataset without performing any pre-processing normalization steps to ensure that all features lie in the same range/interval. Now (a) present the graph you created; (b) based on this graph, identify the best value of k; that is, the value of k that results in k-NN performing the best on the testing set; and (c) describe how the performance of this version of k-NN (without feature normalization) compares with the performance of k-NN with feature normalization. Discuss intuitively the reasons why one may have performed better than the other.





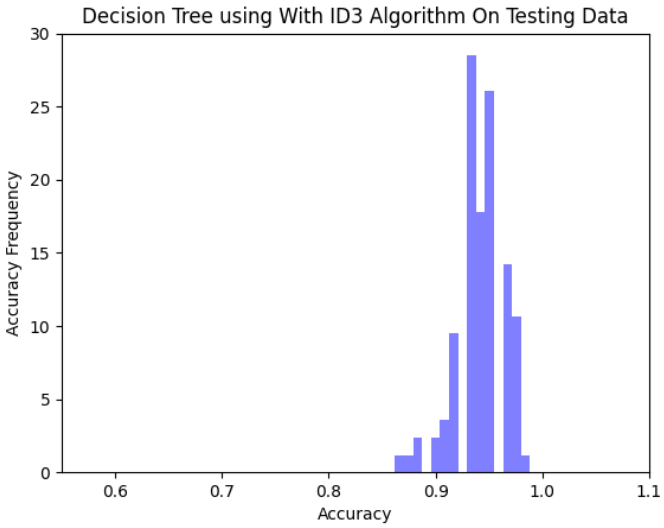
We can observe that the highest accuracy on the testing set is 0.96666667 obtained when k=10 (k in the range 7-15).
We observe that k-NN with feature normalization has higher accuracy on both the training and testing sets. This allows us to conclude that normalization is beneficial for k-NN. I believe that the reason why feature normalization may improve the performance of k-NN is because of k-NN being a distance based algorithm. Hence, features with larger scales may dominate the distance calculation and alter our results. Normalizing all the features scales them to the same range, preventing any dominant feature from dominating the resulting calculation. As a result, k-NN with feature normalization is able to make more accurate predictions.

Q2.1 (12 Points) In the first histogram, you should show the accuracy distribution when the algorithm is evaluated over training data. The horizontal axis should show different accuracy values, and the vertical axis should show the frequency with which that accuracy was observed while conducting these 100 experiments/training processes. The histogram should look like the one in Figure 3 (though the “shape” of the histogram you obtain may be different, of course). You should also report the mean accuracy and its standard deviation.



The mean accuracy for decision Tree using ID3 on training data is 1.0, and the std is 0.0

Q2.2 (12 Points) In the second histogram, you should show the accuracy distribution when the algorithm is evaluated over testing data. The horizontal axis should show different accuracy values, and the vertical axis should show the frequency with which that accuracy was observed while conducting these 100 experiments/training processes. You should also report the mean accuracy and its standard deviation.



The mean accuracy for decision Tree using ID3 on training data is 0.9383908045977009, and the std is 0.023863796362962722

Q2.3 (12 Points) Explain intuitively why each of these histograms looks the way they do. Is there more variance in one of the histograms? If so, why do you think that is the case? Does one histogram show higher average accuracy than the other? If so, why do you think that is the case?

The first histogram shows the accuracy values for the decision tree using the ID3 algorithm on the training data. The histogram shows a single bar at a height of 100 concluding that all the accuracy values are equal to 1.
The second histogram shows the accuracy values for the decision tree using the ID3 algorithm on the testing data. The histogram shows multiple bars at different heights portraying variation in the accuracy values. The bars are clustered around the center of the histogram since most of the accuracy values are between 0.9 and 1.0.
Hence, we observe variance in the second histogram since the accuracy values in the testing data vary. The training data seems to be perfectly accurate, so there is no variance.
As a result, the second histogram has lower average accuracy than the first histogram. Although the accuracy values in the testing data are high even though they are not perfect like the accuracy values observed for the training data.

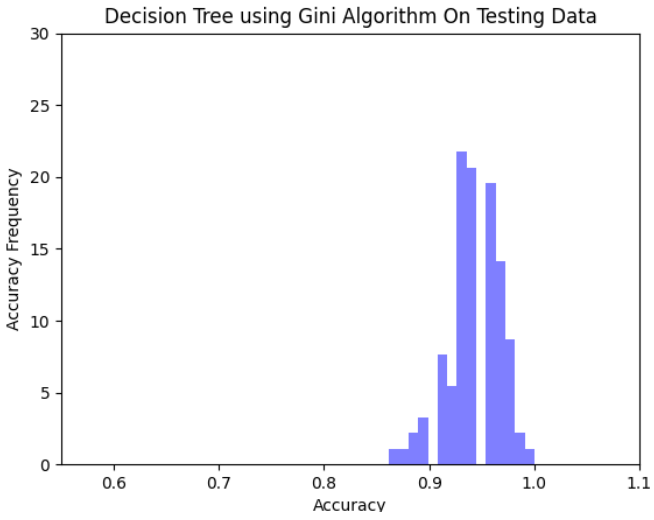
Q2.4 (8 Points) By comparing the two histograms, would you say that the Decision Trees algorithm, when used in this dataset, is underfitting, overfitting, or performing reasonably well? Explain your reasoning.

Based on the two histograms, it appears that the decision trees algorithm is performing quite well on this dataset. The histogram of the training data shows all accuracy values to be 1, indicating that the model has achieved perfect accuracy on the training set which is astounding for a real world perspective. The histogram of the testing data gives us a mean accuracy value of around 0.94 which is still quite high. This suggests that the model has generalized well to new data and is not overfitting to the training data.
Even though the model being underfitting is a possibility, it is unlikely given the high accuracy achieved on both the training and testing sets. Hence from our histograms, we can conclude that the decision trees algorithm is performing well on this dataset.

Q2.5 (6 Points) In class, we discussed how Decision Trees might be non-robust. Is it possible to experimentally confirm this property/tendency via these experiments, by analyzing the histograms you generated and their corresponding average accuracies and standard deviations? Explain your reasoning.

Yes, as discussed in class, iit is possible to experimentally confirm the non-robustness of decision trees by analyzing the histograms and their corresponding average accuracies and standard deviations. As discussed, a non-robust model performs well on the training data but poorly on the test data.
The accuracy values for the training data are perfect (1.0 for all 100 instances). This indicates that the decision tree is overfitting the training data. – (1)
The accuracy values for the testing data are more diverse: a mean of 0.94 and a standard deviation of 0.024. This indicates that the decision tree is not performing well on the testing data. – (2)
From (1) and (2), we can say that our results are consistent with the non-robustness property, hence we can confirm that the decision tree is non-robust on this dataset.

[QE.1] Extra points (15 Points) Repeat the experiments Q2.1 to Q2.4, but now use the Gini criterion for node splitting, instead of the Information Gain criterion.



The mean accuracy for decision trees using Gini on training data is 1.0, and the std is 0.0

The mean accuracy for decision Tree using Gini on testing data is 0.9419540229885055, and the std is 0.02530694121414253

Explain intuitively why each of these histograms looks the way they do. Is there more variance in one of the histograms? If so, why do you think that is the case? Does one histogram show higher average accuracy than the other? If so, why do you think that is the case?

The histogram of the training data shows all accuracy values to be 1, indicating that the model has achieved perfect accuracy on the training set which is astounding for a real world perspective.
The testing dataset histogram shows a wider and more diverse range of accuracy values. The majority of the values fall between 0.92 and 0.97, but there are also some values as low as 0.86 and as high as 0.99. This indicates that the decision tree's performance on the testing set is not quite as good as it was on the training set even though it returns high accuracy results.
Since the training set is a perfect fit, it is likely that the decision tree overfits the training set. Hence, the testing set histogram may show a slightly lower average accuracy.

By comparing the two histograms, would you say that the Decision Trees algorithm, when used in this dataset, is underfitting, overfitting, or performing reasonably well? Explain your reasoning.

Comparing the two histograms, we can conclude the DT algorithm is likely overfitting the training data. The histogram of the training data shows perfect accuracy (1.0 for all 100 instances). This indicates that the algorithm is fitting the data very closely resulting in poor generalization to new data.
On the other hand, the histogram of the testing data shows a wider range of accuracy values, with the majority of the values ranging between 0.9 and 0.98. This again indicates overfitting as the algorithm is not performing as well on the testing data as it did on the training data. Hence, based on these observations, it seems that the decision tree algorithm is likely overfitting the training data.