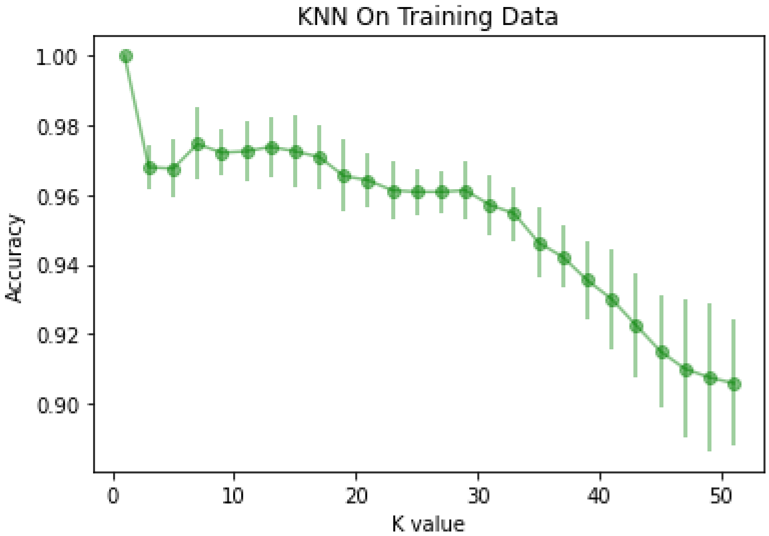
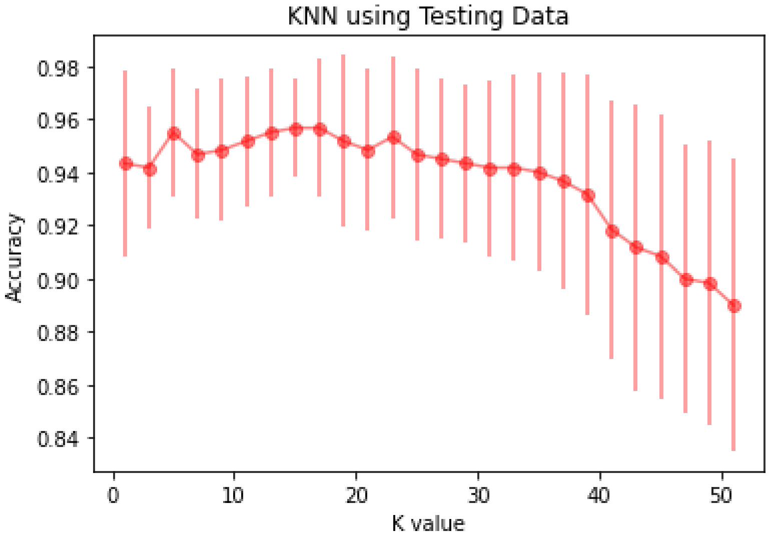
Q1.1 (10 Points) In the first graph, you should show the value of k on the horizontal axis, and on the vertical axis, the average accuracy of models trained over the training set, given that particular value of k. Also show, for each point in the graph, the corresponding standard deviation; you should 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, where k represents the number of neighbors used to make predictions. The graph shows that the accuracy initially decreases as k increases from 1 to around 5-7, and then starts to level off and even slightly increases as k continues to increase. This phenomenon can be explained by the bias-variance tradeoff in machine learning. When k is small (e.g., k=1), the model has low bias because it can fit the training data closely, 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 closely, but lower variance because it is less sensitive to noise. The point at which the accuracy levels off or starts to increase represents a balance between bias and variance.

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 well to new data. As k increases, the model may be better able to generalize to new data because it is less sensitive to noise, but if k is too large, the model may oversimplify the decision boundary and not capture the true underlying patterns in the data. Therefore, there is a sweet spot for the value of k that maximizes the accuracy on the testing set. The decreasing trend in accuracy for larger k values can be attributed to the fact that the model is becoming too simplistic and is not capturing the complexity of the data.

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.

Based on the data, we can observe the following:

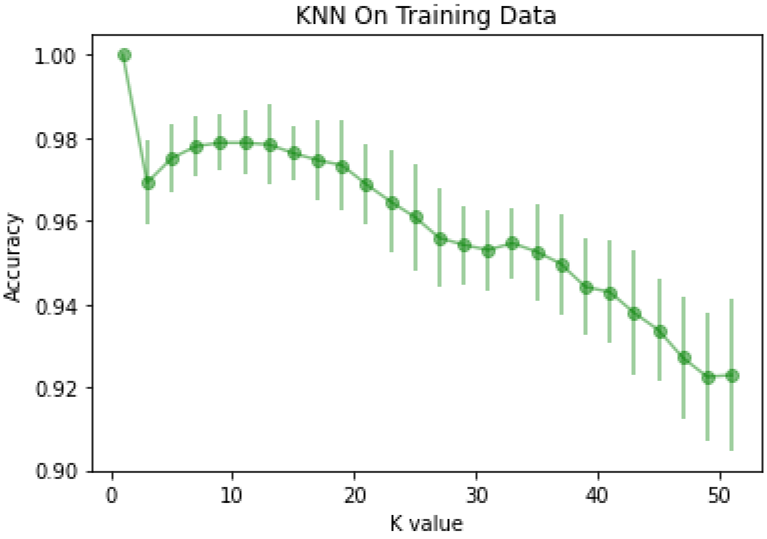
1. As the value of k increases, the performance on the training set decreases, which indicates that the model is underfitting for larger values of k.
2. As the value of k decreases, the performance on the testing set decreases after a certain value of k, which indicates that the model is overfitting for smaller values of k.

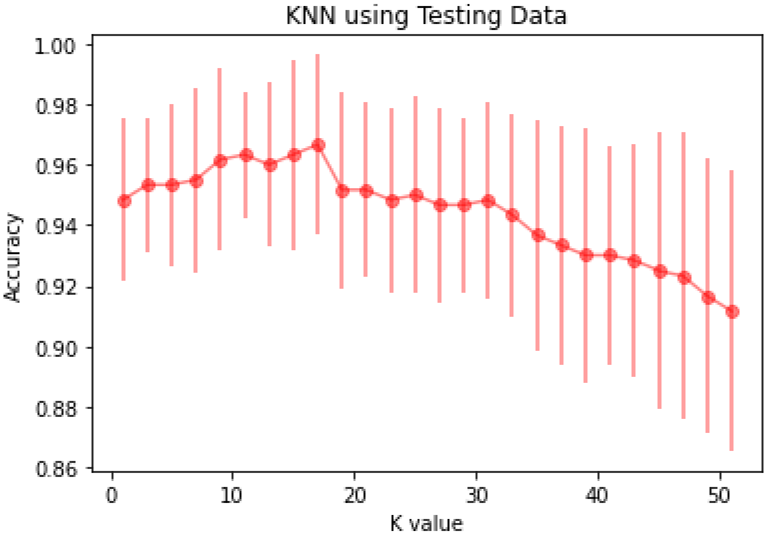
Therefore, we can conclude that the model is underfitting for larger values of k (k > 10) and overfitting for smaller values of k (k < 4). The optimal value of k seems to be in the range of 4-10 where the performance on both the training and testing sets is relatively high and consistent.

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 the algorithm on the training set decreases, while 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 strike 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 closely tailored to the training set and does not generalize well to new instances. On the other hand, a large value of k (such as k=25) tends to underfit the training data, as the algorithm is too general and cannot capture the nuances of the training data.
Therefore, to fine-tune this algorithm so that it works as well as possible in real life, we would select a value of k that falls within the range where the performance on the testing set is highest, which appears to be around k=7 to k=15. This value of k provides a good balance between bias and variance, as it is not too small to overfit the training data, and not too large to underfit the training data.

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.

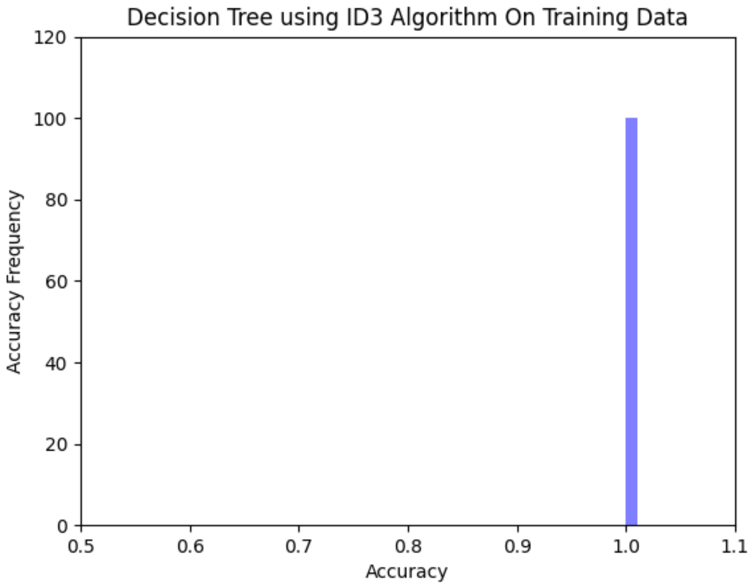




To identify the best value of k that results in k-NN performing the best on the testing set without normalization, we need to find the value of k that gives the highest accuracy on the testing set. From the given data, the highest accuracy on the testing set is 0.96666667, which is obtained with k=8 (7-15 range). Therefore, the best value of k for k-NN without feature normalization is 8.

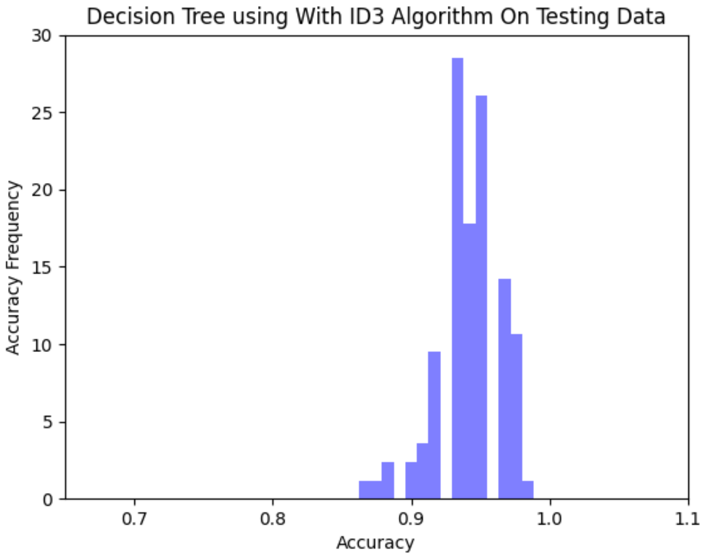
Comparing the performance of k-NN with feature normalization to k-NN without feature normalization, we can see that k-NN with feature normalization has higher accuracy on both the training and testing sets than k-NN without feature normalization. This suggests that feature normalization is beneficial for k-NN. The reason why feature normalization may improve the performance of k-NN is that k-NN is a distance-based algorithm, and features with larger scales may dominate the distance calculation. Feature normalization scales all the features to the same range, which prevents any particular feature from dominating the distance calculation. As a result, k-NN with feature normalization is better able to capture the underlying patterns in the data and make 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. Since all the accuracy values are equal to 1, the histogram shows a single bar at a height of 100%, representing that all the accuracy values are perfect.

The second histogram shows the accuracy values for the decision tree using the ID3 algorithm on the testing data. Since there is some variation in the accuracy values, the histogram shows multiple bars at different heights. The bars are clustered around the center of the histogram since most of the accuracy values are between 0.9 and 1.0. The histogram has a longer tail on the left side since there are some accuracy values that are lower than the mean.

There is more variance in the second histogram since the accuracy values in the testing data are not all the same. The training data is perfectly accurate, so there is no variance.

The second histogram does not show higher average accuracy than the first histogram. Although the accuracy values in the testing data are generally high, they are not all perfect like the accuracy values in the training data.

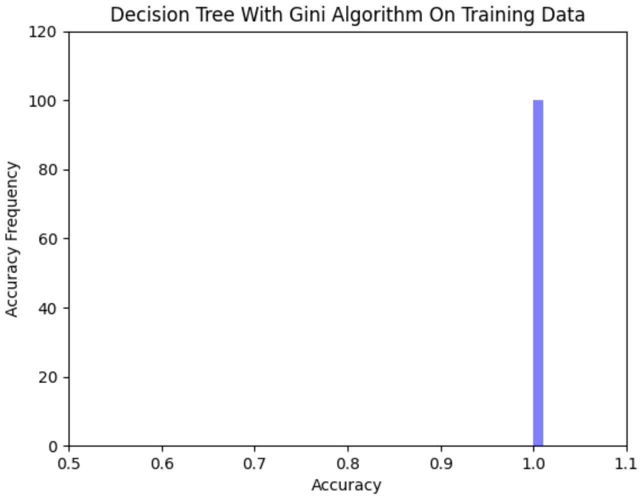
Q2.4 (8 Points) By comparing the two histograms, would you say that the Decision Trees algorithm, when used in this dataset, is underfitting, overfitting, or performing reasonably well? Explain your reasoning.

Based on the two histograms, it appears that the Decision Trees algorithm is performing reasonably 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. The histogram of the testing data, however, shows a distribution of accuracy values around 0.95, which is still quite high. This suggests that the model has generalized well to new data and is not overfitting to the training data.
While it is possible that the model could be underfitting, it is unlikely given the high accuracy achieved on both the training and testing sets. Overall, the histograms suggest that the Decision Trees algorithm is performing well on this dataset.

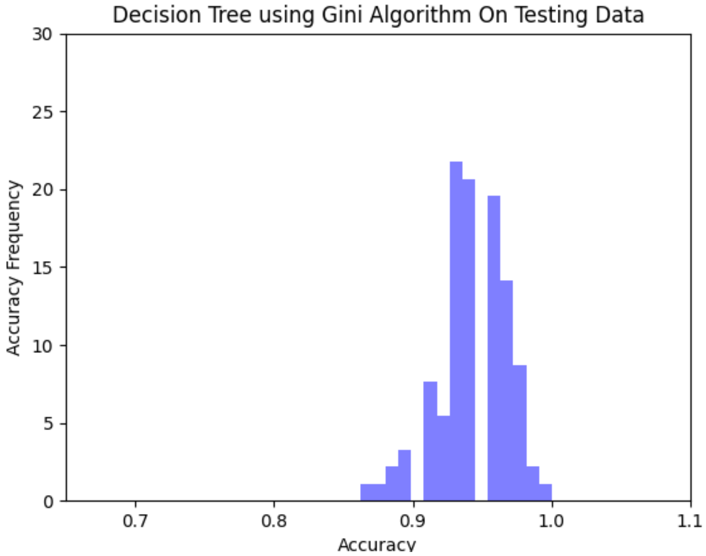
Q2.5 (6 Points) In class, we discussed how Decision Trees might be non-robust. Is it possible to experimentally confirm this property/tendency via these experiments, by analyzing the histograms you generated and their corresponding average accuracies and standard deviations? Explain your reasoning.

Yes, it is possible to experimentally confirm the non-robustness of Decision Trees by analyzing the histograms and their corresponding average accuracies and standard deviations. In general, a non-robust model is one that performs well on the training data but poorly on the test data. Looking at the histograms, we can see that the accuracy values for the training data are consistently high (all 1.0), indicating that the Decision Tree is likely overfitting the training data. However, the accuracy values for the testing data are more spread out, with a mean of 0.943 and a standard deviation of 0.025. This suggests that the Decision Tree is not performing well on the testing data, which is consistent with the non-robustness property. Therefore, we can confirm experimentally that Decision Trees are 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 Tree 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 training dataset histogram shows that all of the accuracy values are equal to 1. This means that the decision tree is able to perfectly classify all of the data points in the training set.
The testing dataset histogram, on the other hand, shows a wider 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 as perfect as it was on the training set, but still relatively good.
There is more variance in the testing dataset histogram because there is a wider range of accuracy values than in the training dataset histogram.
It is not possible to determine which histogram shows a higher average accuracy value based solely on their shape. However, since the training set is a perfect fit, it is likely that the decision tree has overfit the training set, and 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.

Based on the two histograms, it seems that the Decision Tree algorithm is likely overfitting the training data. The histogram of the training data shows perfect accuracy (all values are 1.0), which is a strong indication that the algorithm has memorized the training data and is fitting it very closely, which may result in poor generalization to unseen 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 indicates that the algorithm is not performing as well on the testing data as it did on the training data, which could be an indication of overfitting.

Additionally, the fact that the histogram of the testing data is skewed towards higher accuracy values suggests that the algorithm may be biased towards predicting the majority class, which could also be a sign of overfitting.

Therefore, based on these observations, it seems that the Decision Tree algorithm is likely overfitting the training data, and its performance on the testing data is not as good as expected, which could be an indication of poor generalization.