The result of running P1.py:

Max Depth: 1
Precision (micro): 0.7111
Precision (macro): 0.5000
Precision (weighted): 0.5667
Recall (micro): 0.7111
Recall (macro): 0.6667
Recall (weighted): 0.7111
F1 (micro): 0.7111
F1 (macro): 0.5556
F1 (weighted): 0.6148

Max Depth: 2
Precision (micro): 0.9778
Precision (macro): 0.9762
Precision (weighted): 0.9794
Recall (micro): 0.9778
Recall (macro): 0.9744
Recall (weighted): 0.9778
F1 (micro): 0.9778
F1 (macro): 0.9743
F1 (weighted): 0.9777

Max Depth: 3
Precision (micro): 1.0000
Precision (macro): 1.0000
Precision (weighted): 1.0000
Recall (micro): 1.0000
Recall (macro): 1.0000
Recall (weighted): 1.0000
F1 (micro): 1.0000
F1 (macro): 1.0000
F1 (weighted): 1.0000

Max Depth: 4
Precision (micro): 1.0000
Precision (macro): 1.0000
Precision (weighted): 1.0000
Recall (micro): 1.0000
Recall (macro): 1.0000
Recall (weighted): 1.0000
F1 (micro): 1.0000
F1 (macro): 1.0000

F1 (weighted): 1.0000

Max Depth: 5
Precision (micro): 1.0000
Precision (macro): 1.0000
Precision (weighted): 1.0000
Recall (micro): 1.0000
Recall (macro): 1.0000
Recall (weighted): 1.0000
F1 (micro): 1.0000
F1 (macro): 1.0000
F1 (weighted): 1.0000

Best Recall at depth 3: 1.0000
Lowest Precision at depth 1: 0.5000
Best F1 Score at depth 3: 1.0000

Based on the run results, depth values of 3, 4, and 5 all achieved perfect recall (1.0000). This occurs because when the decision tree depth reaches 3, the model can create sufficiently complex decision boundaries to correctly identify all samples across all classes. The Iris dataset is relatively simple, requiring only 3 levels of depth to perfectly classify all test samples. Increasing depth beyond this point doesn't improve recall as it has already reached its theoretical maximum.

The lowest precision was observed at depth 1 with a macro precision of only 0.5000. This poor performance stems from the excessive simplicity of a depth-1 decision tree, which can only make a single split based on one feature. Such a basic model fails to capture the complex boundary relationships between the three Iris classes. Particularly for similar classes like versicolor and virginica, this simple model cannot effectively distinguish between them, resulting in numerous misclassifications and consequently low precision.

The best F1 score was achieved at depths 3, 4, and 5.

Differences between micro, macro, and weighted scoring methods:
Micro averaging: Aggregates contributions from all classes, calculating metrics based on total true positives, false positives, and false negatives. Each sample has equal weight, giving larger classes more influence on the final score.
Macro averaging: Calculates metrics independently for each class, then takes their unweighted mean. Treats all classes equally regardless of size, making it useful when minority class performance is important.
Weighted averaging: Computes the average of per-class metrics weighted by the number of true instances for each class. Represents a middle ground by accounting for class imbalance while still providing per-class metrics.

The result of running P2.py:

Feature selected for first split: uniformity_of_cell_size
Decision boundary value: 3.5

Entropy of parent node: 0.9217
Entropy after split: 0.3301
Information Gain: 0.5916

Gini of parent node: 0.4467
Gini after split: 0.1140
Gini Gain: 0.3328

Misclassification Error of parent node: 0.3368
Misclassification Error after split: 0.0607
Misclassification Error Gain: 0.2762

According to the results, the decision tree selected "uniformity_of_cell_size" as the feature for the first split, with a threshold value of 3.5. Comparing the three impurity measures, we observe that entropy decreased from 0.9217 before the split to 0.3301 after the split, yielding an information gain of 0.5916; Gini impurity reduced from 0.4467 to 0.1140, resulting in a gain of 0.3328; and misclassification error dropped from 0.3368 to 0.0607, producing a gain of 0.2762. Relative to their original values, entropy decreased by approximately 64.2%, Gini impurity by about 74.5%, and misclassification error by around 82.0%.

The result of running P3.py:

Original Data (Continuous):
F1 Score: 0.9048
Precision: 0.9048
Recall: 0.9048
Confusion Matrix:
[[102 6]
 [6 57]]
True Positives (TP): 57
False Positives (FP): 6
True Positive Rate (TPR): 0.9048

False Positive Rate (FPR): 0.0556

First Principal Component Only:
F1 Score: 0.8992
Precision: 0.8788
Recall: 0.9206
Confusion Matrix:
[[100 8]
 [5 58]]
True Positives (TP): 58
False Positives (FP): 8
True Positive Rate (TPR): 0.9206
False Positive Rate (FPR): 0.0741

First and Second Principal Components:
F1 Score: 0.8852
Precision: 0.9153
Recall: 0.8571
Confusion Matrix:
[[103 5]
 [9 54]]
True Positives (TP): 54
False Positives (FP): 5
True Positive Rate (TPR): 0.8571
False Positive Rate (FPR): 0.0463

Explained Variance Ratio:
First Principal Component: 0.4317
Second Principal Component: 0.1985
Cumulative Variance (2 components): 0.6301

Comparing the first principal component model with the original model:
F1 score: Original data (0.9048) > PCA-1 (0.8992)
Precision: Original data (0.9048) >PCA-1 (0.8788)
Recall: PCA-1 (0.9206) > Original data (0.9048)
Comparing the first two principal components model with the original model:
F1 score: Original data (0.9048) > PCA-2 (0.8852)
Precision: PCA-2 (0.9153) > Original data (0.9048)
Recall: Original data (0.9048) > PCA-2 (0.8571)
Values from the Confusion Matrix:
Original model: TP=57, FP=6, TPR=0.9048, FPR=0.0556
PCA-1 model: TP=58, FP=8, TPR=0.9206, FPR=0.0741
PCA-2 model: TP=54, FP=5, TPR=0.8571, FPR=0.0463

Continuous data is beneficial for this model. The original continuous data model achieved the highest F1 score (0.9048), indicating the best overall performance. While PCA dimensionality reduction captured the main variance in the data (two principal components explained 63.01% of variance), it did not improve model performance. The PCA-1 model had slightly higher recall but lower precision, while the PCA-2 model had slightly higher precision but notably lower recall. This suggests that the complete information preserved in the original continuous data is more important for the decision tree model's performance.