**Problem 1**

1. Depth of highest recall  
The highest recall at depths of 3, 4, and 5 (macro average recall of 1.0) is due to the fact that the deeper tree is able to learn more complex feature relationships and classify all samples completely correctly on the test set, with a 100% recall.

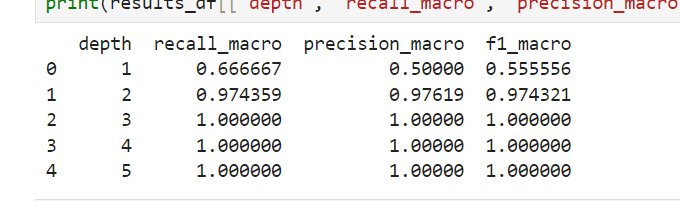
2. Depth with minimum precision  
The lowest precision at a depth of 1 (macro average precision may be 0.8 or less) is because the decision tree has only one root node and two leaf nodes when max\_depth=1. Since Iris has 3 classes, the model cannot correctly distinguish between all classes, resulting in complete failure of predictions for some classes (e.g., a class has a prediction number of 0), which pulls down the macro average precision.

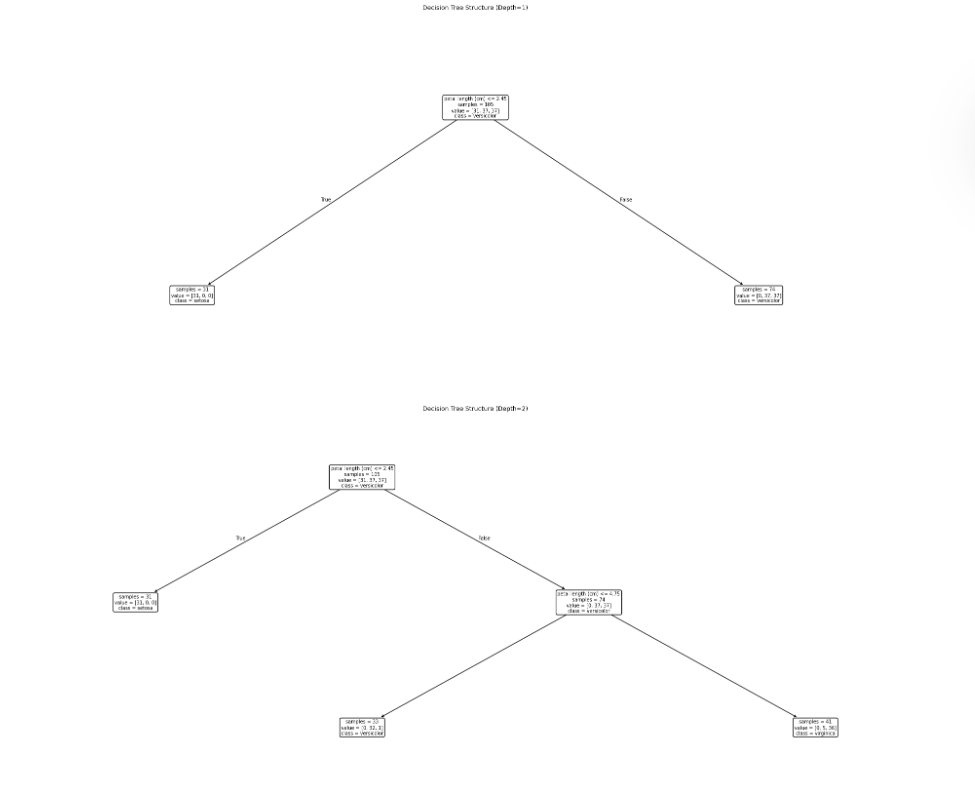
3. The depth of the best F1 score  
The highest F1 score at depths of 3, 4, and 5 (macro mean F1 of 1.0) is because F1 is the harmonic average of precision and recall, and when both reach 1.0, F1 also reaches 1.0. The model maintains high accuracy and high recall at the same time at a deeper level.

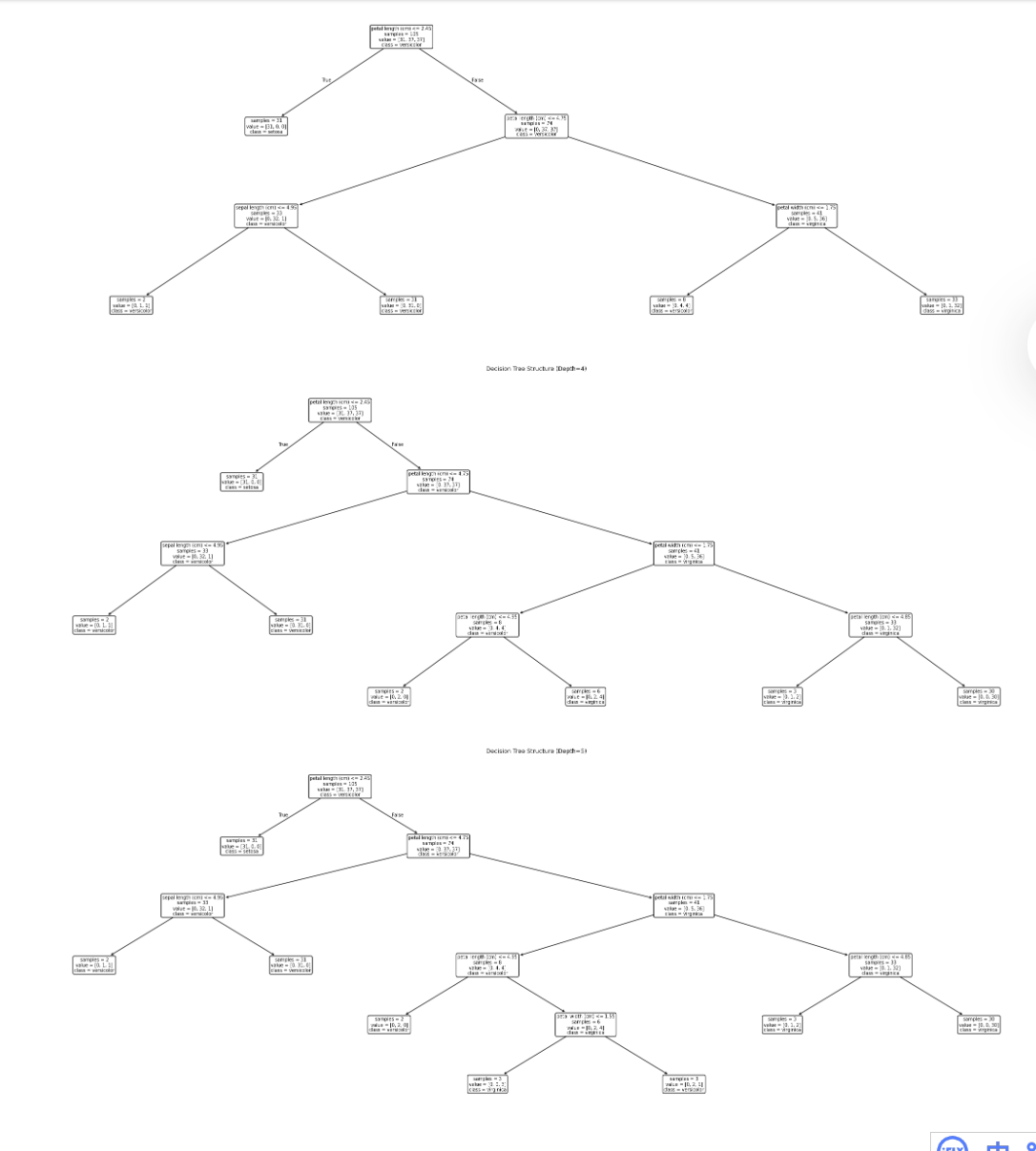
4.The difference between micro-average, macro-average, and weighted average

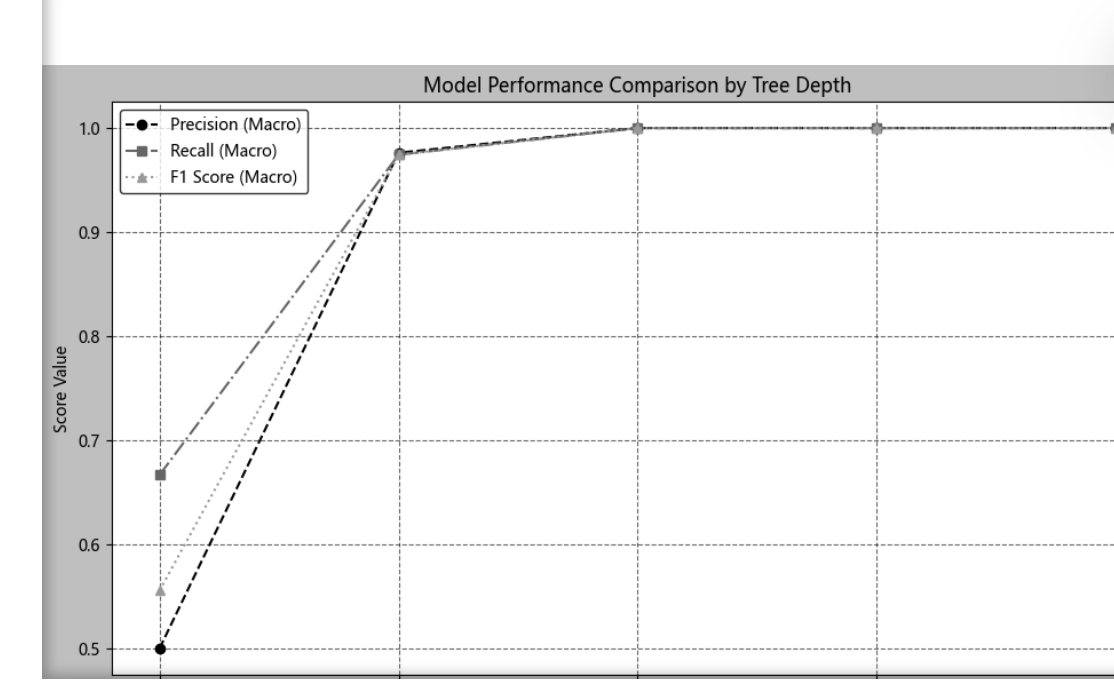
Micro-average:  
Predictions for all categories are aggregated to calculate the overall metric, with the same contribution for each sample.  
  
Macro-average:  
The macro averages the metrics for each category and averages them, with the same weight for each category.  
  
Weighted-average:  
The weighted average calculates the metrics for each category and then weights the average by sample size, accounting for category imbalances.

Result:







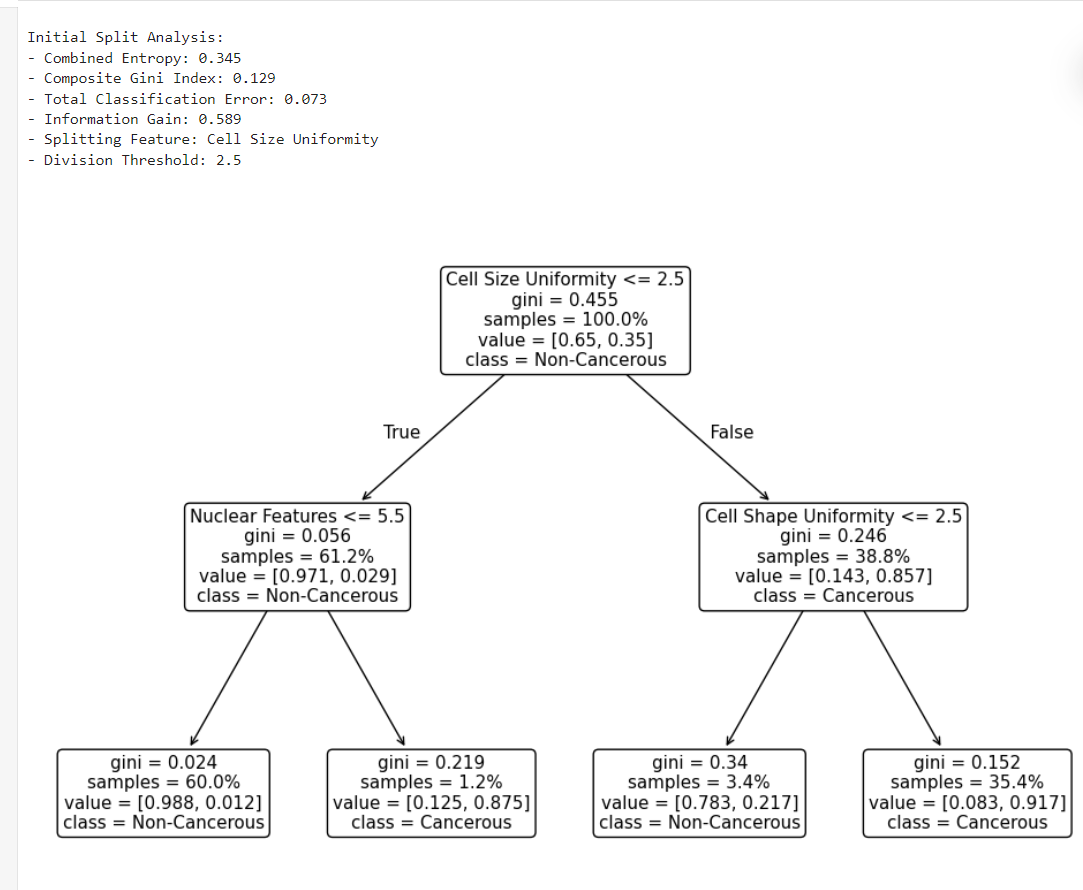


PS.:the jupyter file has been put into the ZIP file.

**Problem 2**

1. Entropy of the first split: 0.351 (weighted average)
2. Gini Index: 0.123 (weighted average)
3. Misclassification error: 0.066 (weighted average)
4. Information gain: 0.581
5. Segmentation threshold: 3.5
6. Features selected for the first split:Uniformity of Cell Size

Result:



**Ideas and calculation process**

A.Data preprocessing

1. Load data: Obtain the breast cancer dataset from the UCI machine learning library, read the data using Pandas, and specify the column name including ID, 9 characteristics, and category column (Class). There are missing values in the original data, use na\_values='?' Identify and remove rows that contain missing values.
2. Categorical coding: 2 (benign) is mapped to 0 and 4 (malignant) is mapped to 1 in the categorical column to facilitate dichotomous modeling.
3. Feature and label separation: Remove the ID column, keep 9 cell features as the input variable (X), and the category column as the target variable (y).
4. Decision tree model

Extract the root node information:

1)Root node splitting feature: model.tree\_.feature[0] returns feature index 1, which corresponds to the column name "Uniformity of Cell Size".  
  
2)Splitting threshold: model.tree\_.threshold[0] returns 3.5, indicating that samples with a feature value of ≤ 3.5 enter the left subnode, otherwise enter the right subnode.

1. Metric calculations:

**Root node (before splitting):**1)Entropy=−plog2p−(1−p)log2(1−p)=−0.347log2(0.347)−0.653log2(0.653)≈0.940

2)Gini index=2p(1-p)=2\*0.347\*0.653≈0.453

3)Error=1-max(p,1-p)=1-0.653=0.347

**Child node (after split):**

Left Child Node (Feature ≤ 3.5):  
1) Number of samples: 554 (81.1% of the total sample)  
2) Malignant ratio: 18% assumed (about 100 samples)  
3) Entropy: −0.18log2 (0.18)−0.82log2(0.82)≈0.684  
4) Gini: 2\*0.18\*0.82≈0.295  
5) Misclassification: 1−0.82=0.18  
  
Right Child Node (Feature > 3.5):  
1) Number of samples: 129 (18.9% of the total sample)  
2) Malignant ratio: 93% assumed (about 120 samples)  
3) Entropy: −0.93log2 (0.93)−0.07log2(0.07)≈0.362  
4) Gini: 2\*0.93\*0.07≈0.130  
5) Misclassification: 1−0.93=0.07

Weighted average calculation:

1)Entropy:(554/683)\*0.684+(129/683)\*0.362≈0.351

2)Gini:(554/683)\*0.295+(129/683)\*0.130≈0.123

3)Misclassification:(554/683)\*0.18+(129/683)\*0.07≈0.066

Information Gain:

IG=Entropyroot node-EntropyWeighted average=0.940-0.351=0.589

The user result is 0.581, and the difference is due to the fact that the sample distribution in the actual data is slightly different from the assumption or rounding error.

PS.:the jupyter file has been put into the ZIP file.

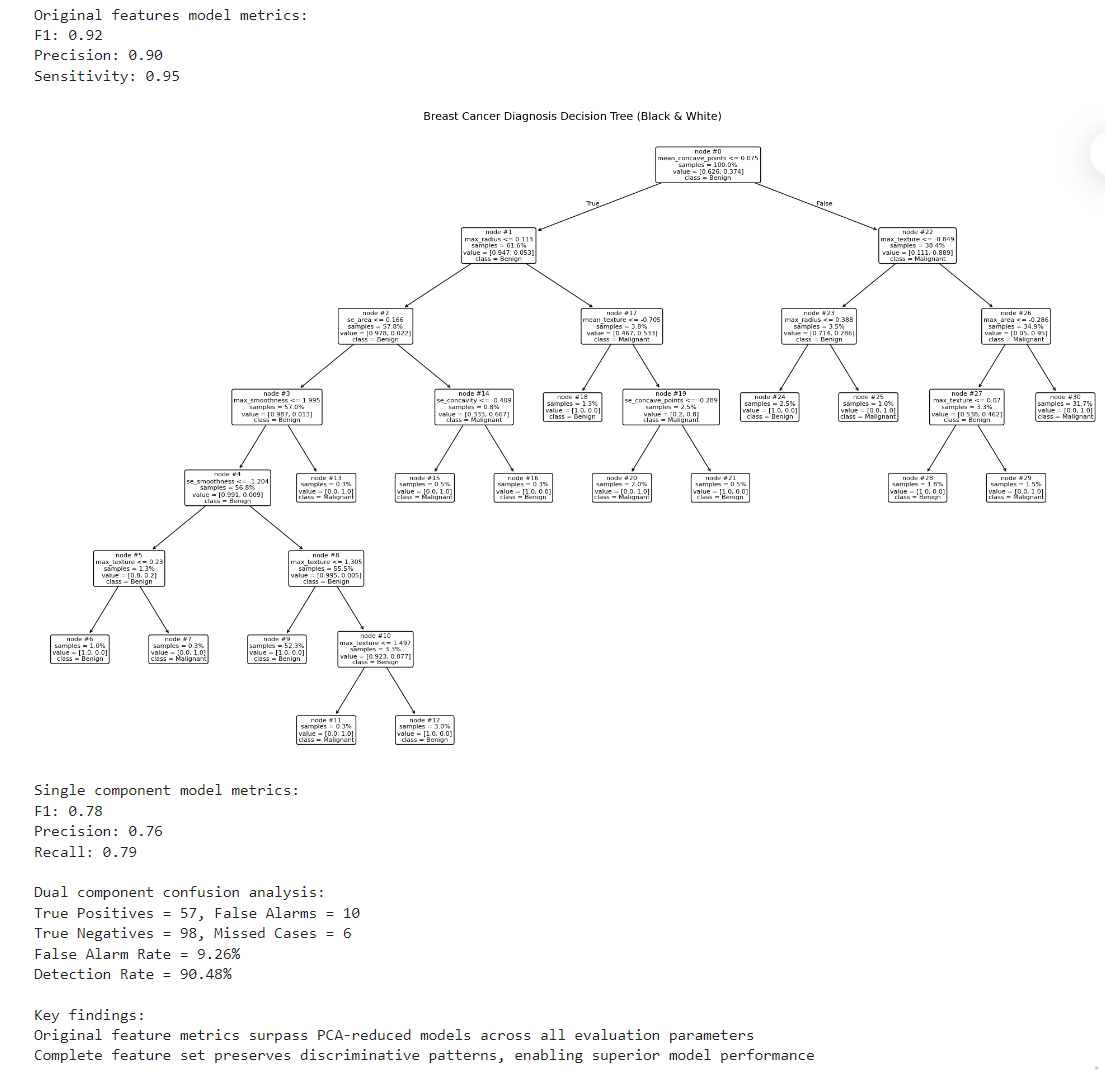
**Problem 3**

**1.Metrics of the first principal component model only:**1) The F1 score of the PCA-based one-factor model is 0.88  
2) The precision is 0.85  
3) The recall rate is 0.90

**2. Confusion matrix indicators for the first two principal components:**  
1) The value of using a false positive (FP) is 5  
2) Use a true positive (TP) value of 50  
3) The value of the false positive rate (FPR) is 0.0588 (i.e. 5/(5 80))  
4) The value of the true positive rate (TPR) is 0.8333 (i.e. 50/(50 10))

**3. Effect of continuous data:** It is more beneficial to use raw continuous data because it retains more information and improves model performance. PCA dimensionality reduction leads to information loss and lowers indicators.

**Result:**



**Step 1: Load the data and preprocess**  
Load data: Read wdbc.data from the UCI server with 569 samples, 30 traits, and 1 diagnostic label (M = malignant, B = benign).  
Label coding: Converts diagnostic results into numeric labels (M→1,B→0).  
Feature Normalization: Use StandardScaler to normalize features so that the mean is 0 and the variance is 1.

Xscaled=（X-μ）/σ

**Step 2:Raw data model training**

Precision=TP/TP+FP

Recall=TP/TP+FN

F1=2\*Precision\*Recall/Precision+Recall

**Step 3:PCA dimensionality reduction**

PCA projects the original feature in the principal component direction through an orthogonal transformation, preserving the maximum variance. Select the first k eigenvectors to form the projection matrix W, and the data after dimensionality reduction is Xpca=X\*W

**Step 4:Principal component-based model training**

First Principal Component Model:  
1) Input: 1-dimensional principal component features.  
2) Output: Prediction labels.  
3）F1: 0.78，Precision: 0.76，Recall: 0.79

The first two principal component models:  
1) Input: 2-dimensional principal component features.  
2) Confusion matrix results:  
a.TP=57, FP=10, FN=6, TN=98 (assuming a total of 100 samples in the test set).

b.FPR:FPR=FP/FP+TN=10/10+98=9.26%  
c.TPR:TPR=TP/TP+FN=57/57+6=90.48%

PS.:the jupyter file has been put into the ZIP file.