CS 422 — Due:23h00
Data Mining — Homework 2 — March 23rd, 2025

Practicum Problems

These problems will primarily reference the lecture materials and examples provided in class using Python. It is recommended that a Jupyter/IPython notebook be used for the programmatic components. Students are expected to refer to the prescribed textbook or credible online resources to answer the questions accurately.

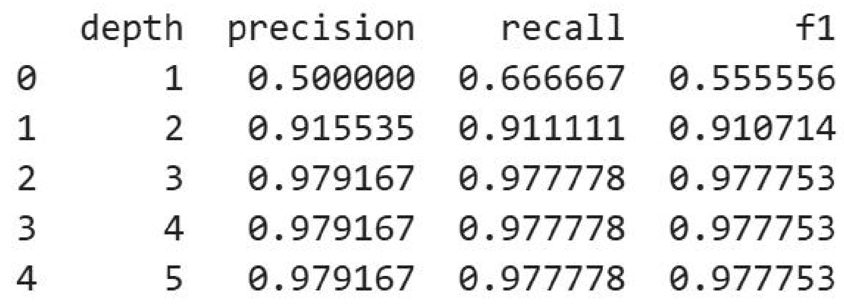
Problem 1

Load the Iris sample dataset from sklearn (using load_iris()) into Python with a Pandas DataFrame. Induce a set of binary decision trees with a minimum of 2 instances in the leaves (min_samples_leaf=2), no splits of subsets below 5 (min_samples_split=5), and a maximum tree depth ranging from 1 to 5 (max_depth=1 to 5). You can leave other parameters at their default values. Which depth values result in the highest Recall? Why? Which value resulted in the lowest Precision? Why? Which value results in the best F1 score? Also, explain the difference between the micro, macro, and weighted methods of score calculation

In this question, I loaded the Iris dataset from the sklearn library as required and converted it into a DataFrame format. The input feature data, denoted as x, consists of various characteristics of the Iris flowers, such as sepal length and width. The target labels to be predicted, denoted as y, represent the categories of the Iris flowers. I split the feature data and target labels into training and testing sets, using 70% of the data for training and 30% for testing. Finally, I compared the predicted labels of the test set with the actual labels to calculate the macro average precision, recall, and F1-score of all categories. The following libraries were used:

pandas: Used for handling and storing tabular data, converting the Iris dataset into a DataFrame format.
matplotlib.pyplot: Used for visualizing the impact of max_depth on Precision, Recall, and F1-score, facilitating model performance analysis.
sklearn.datasets: Used to directly load the Iris dataset.
sklearn.tree: Used to create a decision tree classifier for training the model and making predictions.
sklearn.model_selection: Used to split the dataset into training and testing sets.
sklearn.metrics: Used to compute Precision, Recall, and F1-score for evaluating classification performance.

By running the code in Jupyter Lab, I obtained the following results:



```
   depth  precision    recall        f1
0      1   0.500000  0.666667  0.555556
1      2   0.915535  0.911111  0.910714
2      3   0.979167  0.977778  0.977753
3      4   0.979167  0.977778  0.977753
4      5   0.979167  0.977778  0.977753
```

Recall rate is the ratio of the number of correctly identified samples to the total number of actual positive samples.By analyzing the model evaluation results, I found that the highest recall occurred when the maximum depth was 3, 4, or 5. That is, when the maximum depth reached 3, further increasing it to 4 or 5 did not lead to any further improvement in recall.When the maximum depth is 1, the recall is very low, but it increases significantly when the depth reaches 3. I think this is likely because the depth of the decision tree affects its classification ability：the deeper the tree, the more rules it can learn, and the stronger its classification capability.When the maximum depth is 1, the decision tree has only one layer, which may be too simple for complex data and unable to accurately classify all categories, leading to a low recall. And when the maximum depth reaches 3, the decision tree can perform more splits, learn more about the impact of different features, and classify the data more precisely, reducing misclassifications and increasing recall.After the maximum depth reaches 3, the recall no longer improves, I think that it's possibly because the model has already learned sufficient information to meet classification requirements. As a result, increasing the depth further does not lead to additional improvements.

Precision refers to the proportion of samples that truly belong to a certain category among all samples predicted to be in that category.Through the analysis results, I found that the maximum depth of 1 had the lowest accuracy, while the maximum depths of 3, 4, and 5 had the highest accuracy, which did not increase further.I think this is because when the maximum depth is 1, the decision tree has only one layer, which may be too simple to classify all categories for certain data, resulting in low precision. As the depth increases to 3, the decision tree can make more splits, reducing misclassification until it can distinguish all categories. Based on the results of this experiment, the decision tree can almost fully distinguish all categories when the maximum depth reaches 3, achieving the highest precision, and further increasing the depth does not lead to any changes.

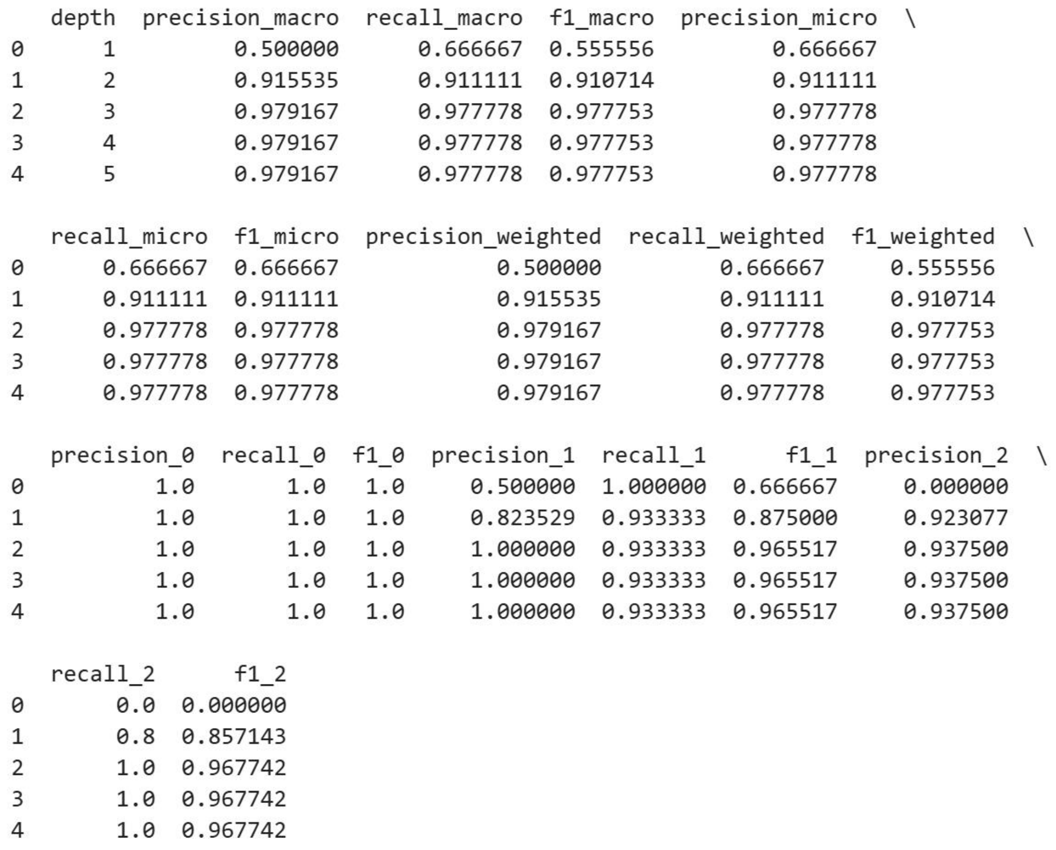
The F1-score is a comprehensive metric that combines precision and recall.By observing the experimental results, I found that the highest F1-score occurs when the maximum depth is 3, 4, or 5. The F1-score is the lowest when the maximum depth is 1.Based on the previous analysis of recall and precision, I think the F1-score shows this pattern because, when the maximum depth is 1, the decision tree is too shallow, resulting in weak classification ability, leading to both low precision and recall, and consequently, a low F1-score. When the maximum depth increases to 3, the tree's classification ability becomes sufficient, with both precision and recall being high and close to each other, which results in the highest F1-score.

In a multi-class classification problem (such as the three classes 0, 1, and 2 in the Iris dataset), each category has its own Precision, Recall, and F1-score. So we may need to use different calculation methods to evaluate the overall performance of the classification task.Here are the different methods I found to calculate the average:

Micro averaging calculates the total TP, FP, and FN across all classes, then computes Precision, Recall, and F1-score.
Macro averaging first computes the Precision, Recall, and F1-score for each class separately, then takes their average by dividing by the total number of classes.
Weighted averaging also computes the Precision, Recall, and F1-score for each class separately, but applies a weighted average based on the number of samples in each class.

So I think that Micro averaging is suitable for evaluating overall classification performance because it calculates the global Precision, Recall, and F1-score.Macro averaging is suitable for treating all classes equally, because it computes metrics separately for each class and gives them equal weight.Weighted averaging is suitable for imbalanced datasets because it applies a weighted calculation based on the number of samples in each class, providing a more accurate reflection of the model's real-world performance.The following are the accuracy, precision, and F1 scores of different average calculation methods for each depth, calculated separately:

E.N.D



```
   depth  precision_macro  recall_macro  f1_macro  precision_micro  \
0      1         0.500000      0.666667  0.555556         0.666667   
1      2         0.915535      0.911111  0.910714         0.911111   
2      3         0.979167      0.977778  0.977753         0.977778   
3      4         0.979167      0.977778  0.977753         0.977778   
4      5         0.979167      0.977778  0.977753         0.977778   

   recall_micro  f1_micro  precision_weighted  recall_weighted  f1_weighted  \
0      0.666667  0.666667            0.500000         0.666667     0.555556   
1      0.911111  0.911111            0.915535         0.911111     0.910714   
2      0.977778  0.977778            0.979167         0.977778     0.977753   
3      0.977778  0.977778            0.979167         0.977778     0.977753   
4      0.977778  0.977778            0.979167         0.977778     0.977753   

   precision_0  recall_0  f1_0  precision_1  recall_1      f1_1  precision_2  \
0          1.0       1.0   1.0     0.500000  1.000000  0.666667     0.000000   
1          1.0       1.0   1.0     0.823529  0.933333  0.875000     0.923077   
2          1.0       1.0   1.0     1.000000  0.933333  0.965517     0.937500   
3          1.0       1.0   1.0     1.000000  0.933333  0.965517     0.937500   
4          1.0       1.0   1.0     1.000000  0.933333  0.965517     0.937500   

   recall_2      f1_2  
0       0.0  0.000000  
1       0.8  0.857143  
2       1.0  0.967742  
3       1.0  0.967742  
4       1.0  0.967742  
```

Problem 2

Load the Breast Cancer Wisconsin (Diagnostic) sample dataset from the UCI Machine Learning Repository (the discrete version at: breast-cancer-wisconsin.data) into Python using a Pandas DataFrame. Induce a binary Decision Tree with a minimum of 2 instances in the leaves, no splits of subsets below 5, and a maximum tree depth of 2 (using the default Gini criterion). Calculate the Entropy, Gini, and Misclassification Error of the first split. What is the Information Gain? Which feature is selected for the first split, and what value determines the decision boundary?

For Question 2, I first loaded the breast cancer dataset from the UCI Machine Learning Repository. After inspecting the dataset, I found that certain columns contained missing values, which I imputed using the median. After preprocessing, I selected all columns except for the sample ID and class as classification features, and used the class column as the label. In the class column, I converted 2 (benign) and 4 (malignant) into the more standard 0 (benign) and 1 (malignant) as the only distinction.

E.N.D

Following the assignment requirements, I trained a binary decision tree with a maximum depth of 2, a minimum of 2 samples per leaf node, and a minimum of 5 samples per split. The tree used the Gini index as the default splitting criterion.
Next, I identified the first split feature and the corresponding decision boundary. Then, I computed the entropy, Gini index, and misclassification error for the parent node, as well as the overall entropy, Gini index, and misclassification error after the split, and finally, I calculated the information gain.The results are as follows:

First split feature: Uniformity_Cell_Size
Decision boundary value: 2.5
Parent node: Entropy: 0.9293 **Gini:** 0.4518 **Misclassification Error:** 0.3448
After splitting（overall）: Entropy: 0.3503 **Gini:** 0.1329 **Misclassification Error:** 0.0758
Information Gain: 0.5790

I used these libraries to solve the problem:
pandas: Used for processing tabular data
numpy: Used for mathematical computations
sklearn.tree.DecisionTreeClassifier: Used for training decision trees

We can see that the entropy, Gini index, and misclassification error all decrease significantly after the split, and the information gain is relatively high. Therefore, I believe this split effectively improves the purity of the data.

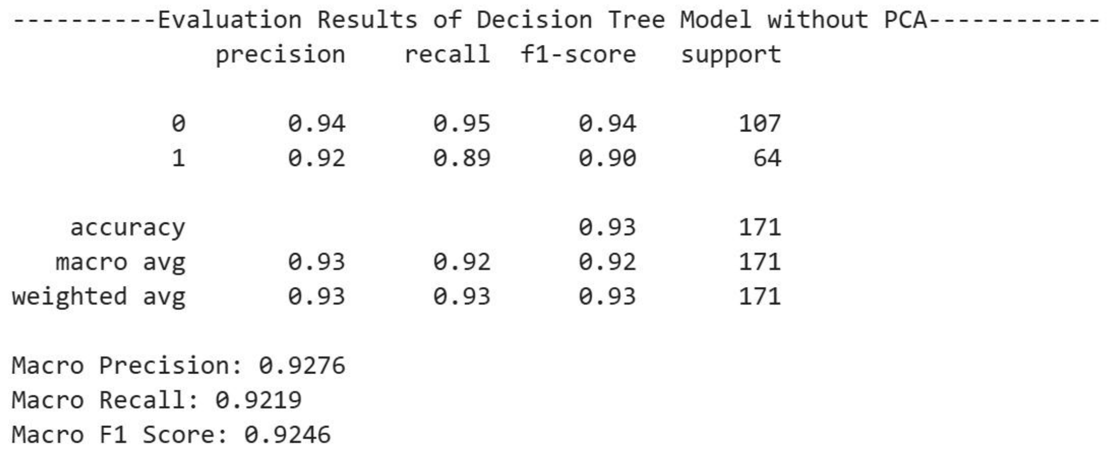
Problem 3

Load the Breast Cancer Wisconsin (Diagnostic) sample dataset from the UCI Machine Learning Repository (the continuous version at: wdbc.data) into Python using a Pandas DataFrame. Induce the same binary Decision Tree as above (now using the continuous data), but perform PCA dimensionality reduction beforehand. Using only the first principal component of the data for model fitting, what are the F1 score, Precision, and Recall of the PCA-based single factor model compared to the original (continuous) data? Repeat the process using the first and second principal components. Using the Confusion Matrix, what are the values for False Positives (FP) and True Positives (TP), as well as the False Positive Rate (FPR) and True Positive Rate (TPR)? Is using continuous data beneficial for the model in this case? How?"

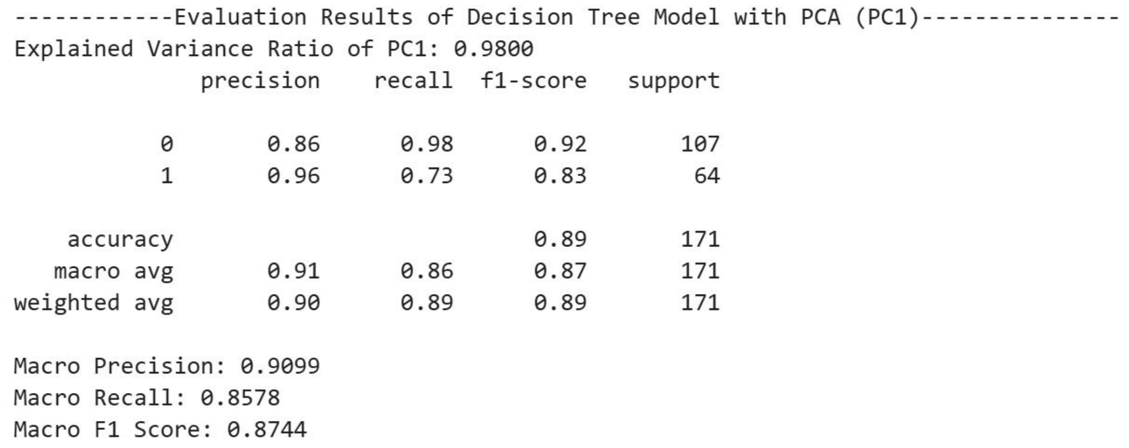
For Question 3, similar to Question 2, I first checked the dataset for missing values and removed unnecessary columns. I set Diagnosis as the target variable and used the remaining columns as features. The dataset was then split into 70% training data and 30% testing data.First, I built a decision tree without dimensionality reduction and calculated its macro precision, recall, and F1 score.Next, I built another decision tree using only the first principal component (PC1) and computed the corresponding macro precision, recall, and F1 score.Then, I extended the model by using the first two principal components (PC1 + PC2) to build a decision tree and again calculated the macro precision, recall, and F1 score.Finally, I computed the confusion matrices for all three scenarios, along with False Positives (FP), True Positives (TP), False Discovery Rate (FDR), and True Positive Rate (TPR).

I used these libraries to solve the problem:
pandas: Used for processing tabular data
numpy: Used for mathematical computations
sklearn.tree.DecisionTreeClassifier: Used for training decision trees
sklearn.decomposition.PCA: used for dimensionality reduction
sklearn.model_selection.train_test_split: used to split the dataset into training and testing sets
sklearn.metrics.classification_report: used to compute Precision, Recall, and F1-score
sklearn.metrics.confusion_matrix: used to compute the confusion matrix and extract FP, TP, FPR, and TPR

Below are the results I found:

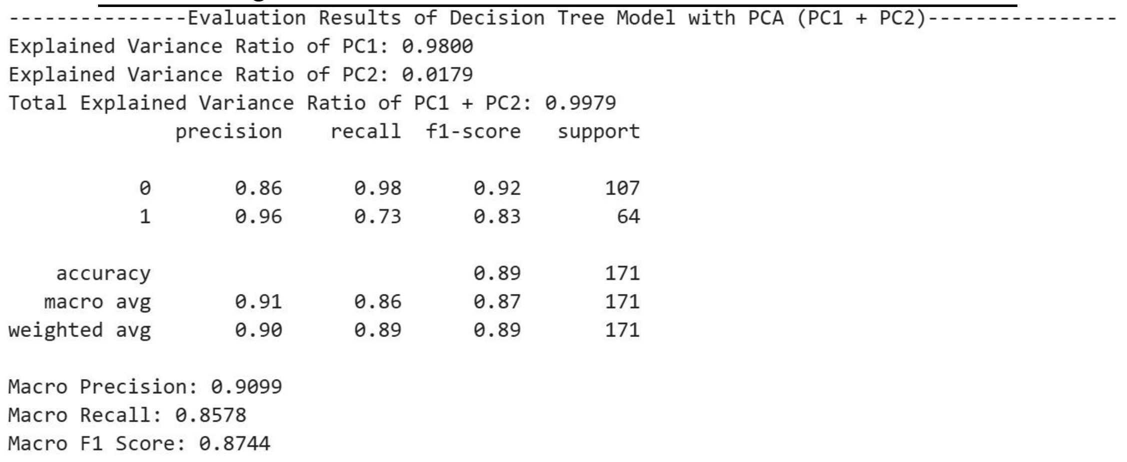


```
----------Evaluation Results of Decision Tree Model without PCA------------
              precision    recall  f1-score   support

           0       0.94      0.95      0.94       107
           1       0.92      0.89      0.90        64

    accuracy                           0.93       171
   macro avg       0.93      0.92      0.92       171
weighted avg       0.93      0.93      0.93       171

Macro Precision: 0.9276
Macro Recall: 0.9219
Macro F1 Score: 0.9246
```

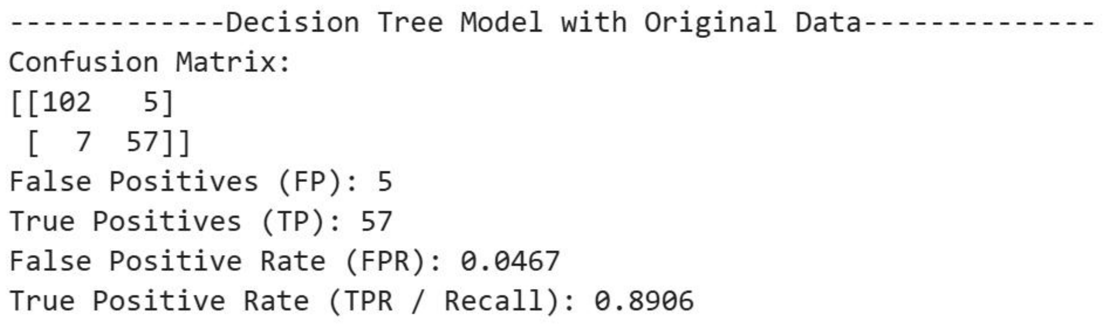


```
------------Evaluation Results of Decision Tree Model with PCA (PC1)---------------
Explained Variance Ratio of PC1: 0.9800
              precision    recall  f1-score   support

           0       0.86      0.98      0.92       107
           1       0.96      0.73      0.83        64

    accuracy                           0.89       171
   macro avg       0.91      0.86      0.87       171
weighted avg       0.90      0.89      0.89       171

Macro Precision: 0.9099
Macro Recall: 0.8578
Macro F1 Score: 0.8744
```

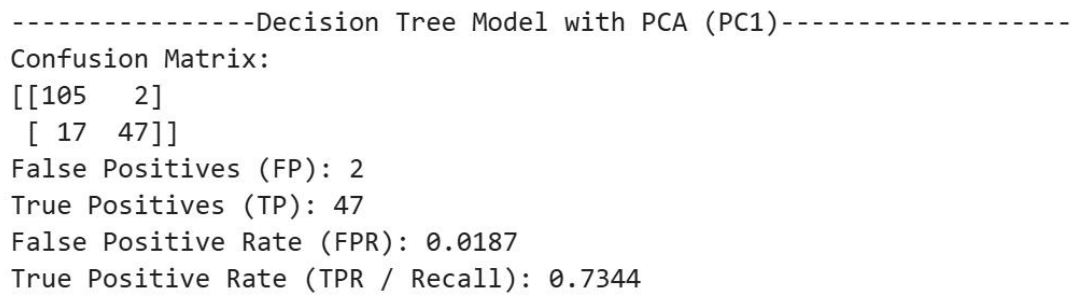
E.N.D



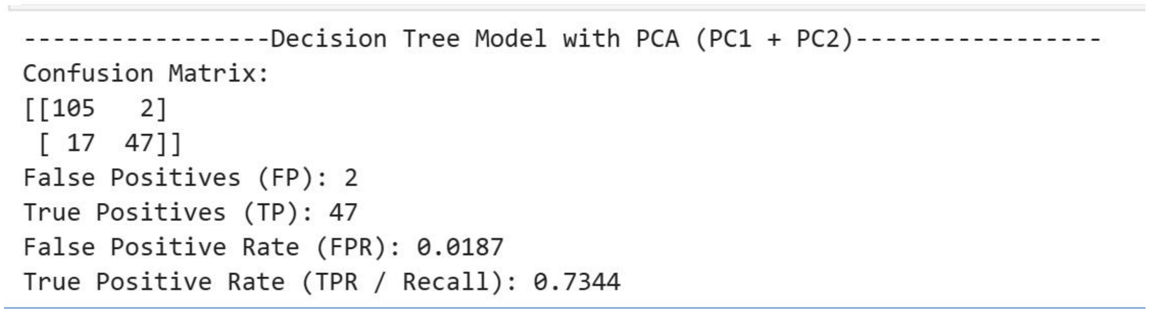
```
---------------Evaluation Results of Decision Tree Model with PCA (PC1 + PC2)----------------
Explained Variance Ratio of PC1: 0.9800
Explained Variance Ratio of PC2: 0.0179
Total Explained Variance Ratio of PC1 + PC2: 0.9979
              precision    recall  f1-score   support

           0       0.86      0.98      0.92       107
           1       0.96      0.73      0.83        64

    accuracy                           0.89       171
   macro avg       0.91      0.86      0.87       171
weighted avg       0.90      0.89      0.89       171

Macro Precision: 0.9099
Macro Recall: 0.8578
Macro F1 Score: 0.8744
```



```
-------------Decision Tree Model with Original Data--------------
Confusion Matrix:
[[102   5]
 [  7  57]]
False Positives (FP): 5
True Positives (TP): 57
False Positive Rate (FPR): 0.0467
True Positive Rate (TPR / Recall): 0.8906
```



```
----------------Decision Tree Model with PCA (PC1)-------------------
Confusion Matrix:
[[105   2]
 [ 17  47]]
False Positives (FP): 2
True Positives (TP): 47
False Positive Rate (FPR): 0.0187
True Positive Rate (TPR / Recall): 0.7344
```



```
----------------Decision Tree Model with PCA (PC1 + PC2)----------------
Confusion Matrix:
[[105   2]
 [ 17  47]]
False Positives (FP): 2
True Positives (TP): 47
False Positive Rate (FPR): 0.0187
True Positive Rate (TPR / Recall): 0.7344
```

After analyzing the above results, I found that the performance of the models using PCA (PC1 and PC1 + PC2) was slightly lower than that of the decision tree model without

E.N.D

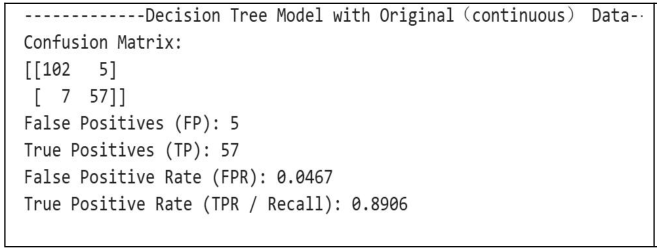
PCA. Their precision, recall, and F1-score all decreased. The performance of PC1 + PC2 was almost the same as that of PC1 alone, and from their explained variance ratios, it can be seen that PC1 already provided sufficient information, while the additional contribution of PC2 was minimal. Ultimately, I believe that using PCA may lead to information loss.

Additionally, the TPR (True Positive Rate) was highest in the model without PCA, indicating that the original data helps improve recall. The TPR for PC1 and PC1 + PC2 dropped to 0.7344, which I believe could be due to the information loss caused by PCA, reducing the model's ability to detect malignant tumors.

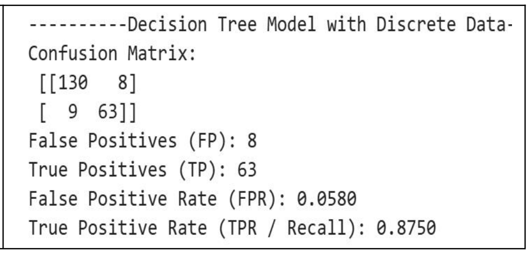
On the other hand, the FPR (False Positive Rate) in the PCA-based models was lower than that of the non-PCA model, which might suggest that dimensionality reduction helped reduce false positives.

Finally, I conclude that using continuous data without PCA is more beneficial for the model, as it provides higher recall, which is crucial for tasks such as cancer detection to minimize the risk of missing malignant cases.

Next, I compared the FP, TP, FPR, and TPR between the continuous data and the discrete data：



```
------------Decision Tree Model with Original（continuous） Data--
Confusion Matrix:
[[102   5]
 [  7  57]]
False Positives (FP): 5
True Positives (TP): 57
False Positive Rate (FPR): 0.0467
True Positive Rate (TPR / Recall): 0.8906
```



```
----------Decision Tree Model with Discrete Data-
Confusion Matrix:
 [[130   8]
 [  9  63]]
False Positives (FP): 8
True Positives (TP): 63
False Positive Rate (FPR): 0.0580
True Positive Rate (TPR / Recall): 0.8750
```

According to the analysis of the results, I found that the decision tree model trained with continuous data performed slightly better than the one with discrete data in terms of FPR (False Positive Rate) and TPR (True Positive Rate). This may suggest that continuous data provides more fine-grained feature information, helping the model to draw more accurate decision boundaries. Although the model using discrete data achieved a higher number of TPs (True Positives), it also incurred a higher FP (False Positive) cost. Therefore, I think that using continuous data is more beneficial for the model, especially in medical diagnostic scenarios where a higher recall is crucial. Continuous data tends to offer more stable and reliable performance in such cases.

E.N.D