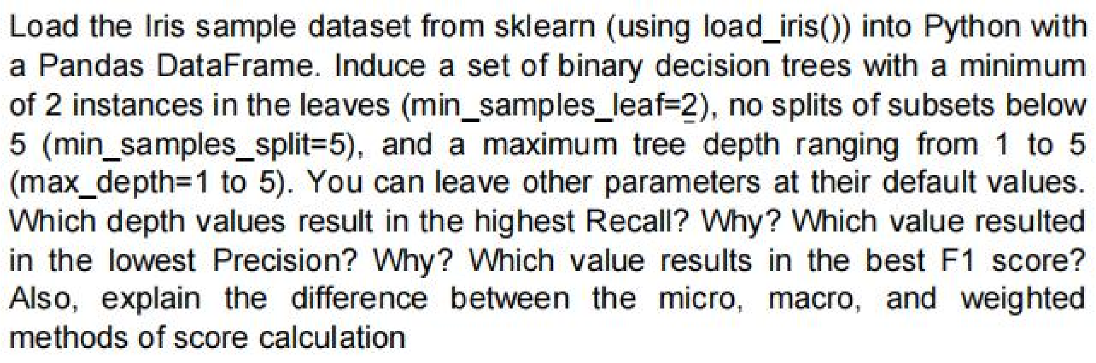
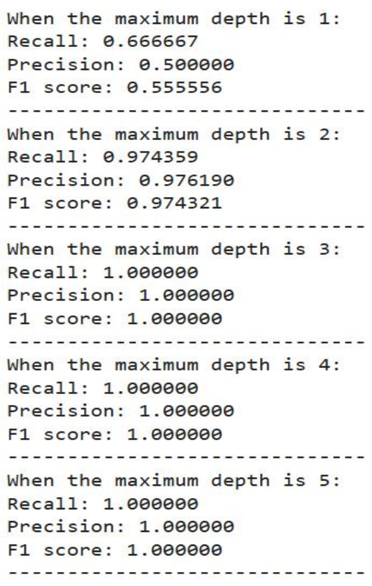
Data Mining HomeWork2.0

1 Problem 1:

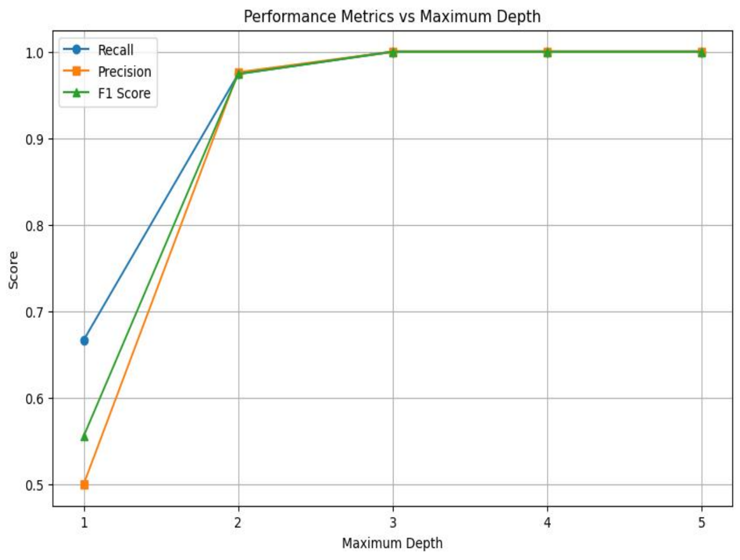


Load the Iris sample dataset from sklearn (using load_iris()) into Python with a Pandas DataFrame. Induce a set of binary decision trees with a minimum of 2 instances in the leaves (min_samples_leaf=2), no splits of subsets below 5 (min_samples_split=5), and a maximum tree depth ranging from 1 to 5 (max_depth=1 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

We can use macro method of score calculation:



```
When the maximum depth is 1:
Recall: 0.666667
Precision: 0.500000
F1 score: 0.555556
------------------------------
When the maximum depth is 2:
Recall: 0.974359
Precision: 0.976190
F1 score: 0.974321
------------------------------
When the maximum depth is 3:
Recall: 1.000000
Precision: 1.000000
F1 score: 1.000000
------------------------------
When the maximum depth is 4:
Recall: 1.000000
Precision: 1.000000
F1 score: 1.000000
------------------------------
When the maximum depth is 5:
Recall: 1.000000
Precision: 1.000000
F1 score: 1.000000
------------------------------
```



According to the data results :

- **Highest Recall(max-depth=3,4,5):** As can be seen from the data, when the maximum depth is 3, 4, or 5, the recall rate is 1.0 for all of them, which is the highest value in this set of results. The reason is that as the depth increases, the decision tree has a stronger learning ability and can capture the features of more positive - example samples, thus correctly identifying all positive examples. However, overfitting may also occur simultaneously.the model can completely differentiate between the different species with complete accuracy.
- **Lowest Precision(max-depth=1)**: When the maximum depth is 1, the precision is 0.5, which is the lowest value. Since the decision tree is very simple with a depth of 1, its ability to distinguish samples is limited,and there are not enough features for training, resulting in a tree that cannot accurately distinguish between different classes

and it will misclassify many samples, so it is in a low proportion of true positive examples among the samples predicted as positive.
- **Best F1 - score(max-depth=3,4,5):** When the maximum depth is 3, 4, or 5, the F1 - score is 1.0 for all of them, which is the best value. At this time, both the precision and recall reach the highest level, and the balance between them is excellent, so the F1 - score is also the highest.

The difference between the micro, macro and weighted methods:

(1)Micro - averaging:

- Principle: Micro - averaging calculates the overall score by considering all the individual predictions and true labels across all classes. It combines the true positives, false positives, and false negatives from all classes into a single set of counts and then computes the relevant metrics (such as precision, recall, and F1 - score) based on these combined counts.

-Use case: This method is useful when each individual prediction is considered equally important, regardless of the class. It gives an overall picture of the model's performance without emphasizing the performance on individual classes. It is often used when the dataset is balanced, and the focus is on the overall accuracy of the model.

For example:

$\text{precision}_{\text{micro}} = \frac{\sum_{i=1}^{n} \text{tp}_i}{\sum_{i=1}^{n} (\text{tp}_i + \text{fp}_i)}$, where n represents the number of classes, and tp_i and fp_i are the true positives and false positives of the i-th class respectively.

(2) Macro - averaging:

- Principle: Macro - averaging first calculates the metrics (such as precision, recall, and F1 - score) for each class separately. Then, it takes the unweighted average of these class - wise metrics to obtain the final score. This means that each class is treated equally, regardless of the number of samples in each class.

-Use case: Macro - averaging is beneficial when the classes in the dataset are imbalanced, and we want to give equal importance to the performance of the model on each class. It helps to identify if the model is performing poorly on some classes, even if it has a high overall accuracy due to performing well on dominant classes.
For example, in the case of F1-score:

$$\text{Macro-F1} = \frac{\text{F1-Class A} + \text{F1-Class B} + \text{F1-Class C}}{3}$$

(3)Weighted - averaging:

- Principle: Similar to macro - averaging, weighted - averaging first calculates the metrics for each class. However, instead of taking a simple unweighted average, it weights the class - wise metrics by the number of samples in each class. So, classes with more samples have a greater influence on the final score.

- Use case: This method is a compromise between micro - averaging and macro - averaging. It takes into account the class imbalance in the dataset while still considering the importance of each class. It is useful in situations where the classes

are imbalanced, but we want to give more weight to the performance on larger classes, as they may be more representative of the real - world scenario.
For example, in the case of F1-score:

$$\text{Weighted-F1} = \frac{\text{F1-Class A} \times N_A + \text{F1-Class B} \times N_B + \text{F1-Class C} \times N_C}{N_A + N_B + N_C}$$

2 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?

Entropy: Quantifies the uncertainty existing in a node based on the likelihoods of different classes.

$$H(D) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

```
def entropy(p):
    if 0 < p < 1:
        return -(p * np.log2(p) + (1 - p) * np.log2(1 - p))
    else:
        return 0
```

Gini Index: Assesses the probability of making a classification error.

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

```
def gini(p):
    return 1 - p * p - (1 - p) * (1 - p)
```

Misclassification Error: the proportion of instances that are incorrectly classified by a classification model.

$$Misclass = 1 - \max(p_i)$$

```
def misclassification_error(p):
    return 1 - max(p, 1 - p)
```

Information Gain: Measures the reduction in uncertainty after splitting a dataset based on a feature.

$$IG(T,A) = Entropy(T) - \sum \frac{|T_n|}{|T|} Entropy(T_n)$$

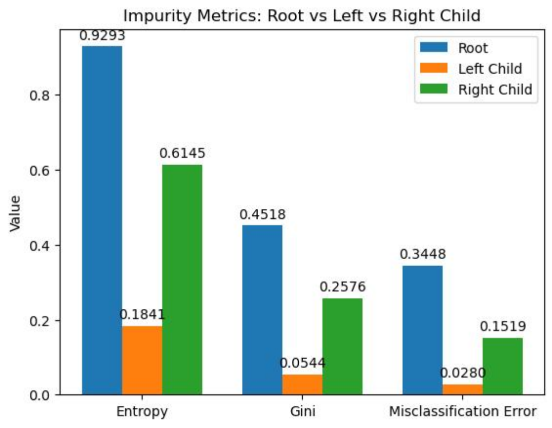
The expression $\sum \frac{|T_n|}{|T|} Entropy(T_n)$ represents the weighted - sum of the entropies of all subsets T_n obtained by splitting the original dataset T based on an attribute, where $\frac{|T_n|}{|T|}$ is the weight.

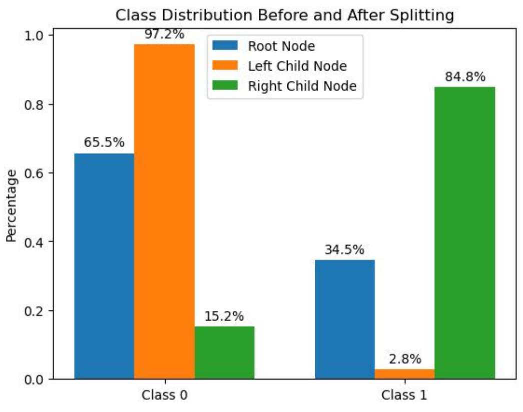
Based on this, we can calculate that:(the first question)

```
Parent node Entropy: 0.9293, Gini: 0.4518, Misclassification Error: 0.3448
After splitting Entropy: 0.3503, Gini: 0.1329, Misclassification Error: 0.0758
Information Gain: 0.5790
```

And the feature for the first split:(the second question)

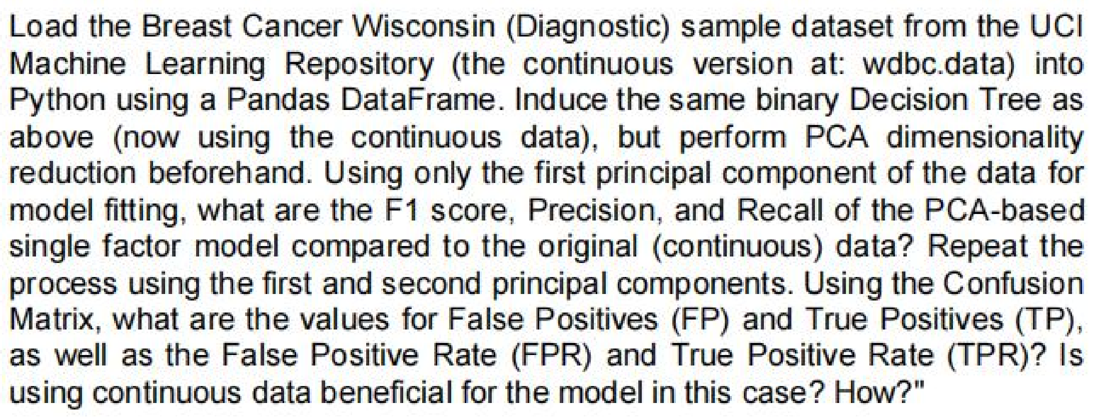
```
The first splitting feature selected: uniformity_of_cell_size
Decision boundary value: 2.5
```





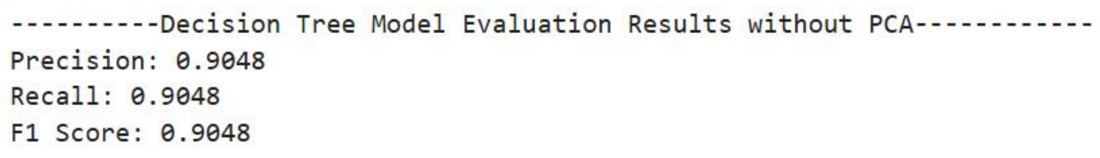
Regarding the changes in the metric values, the root node has higher values in entropy, Gini index, and misclassification error than both the left - child node and the right - child node. After splitting, there are significant decreases in all impurity metrics. After splitting, Class 0 accounted for as high as 97.2% in the left - child node, and Class 1 accounted for 84.8% in the right - child node, showing significant differences in class distribution.

3 Problem 3:

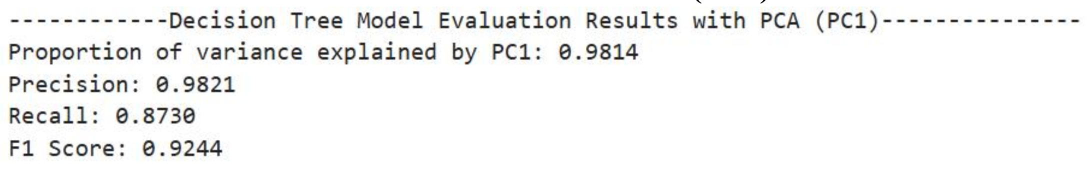

Load the Breast Cancer Wisconsin (Diagnostic) sample dataset from the UCI Machine Learning Repository (the continuous version at: wdbc.data) into Python using a Pandas DataFrame. Induce the same binary Decision Tree as above (now using the continuous data), but perform PCA dimensionality reduction beforehand. 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?"

We can evaluate the model based on three methods of constructing decision trees:

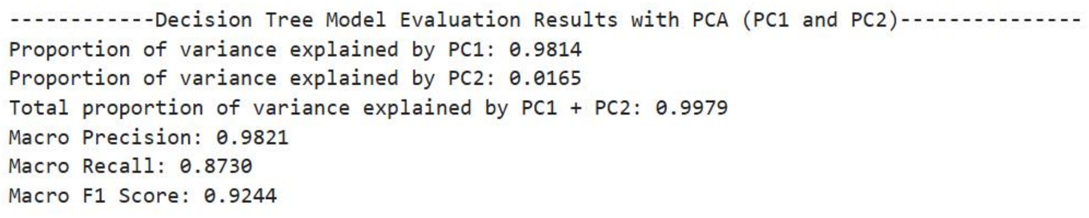
Decision Tree Model Evaluation Results without PCA:


```
----------Decision Tree Model Evaluation Results without PCA------------
Precision: 0.9048
Recall: 0.9048
F1 Score: 0.9048
```

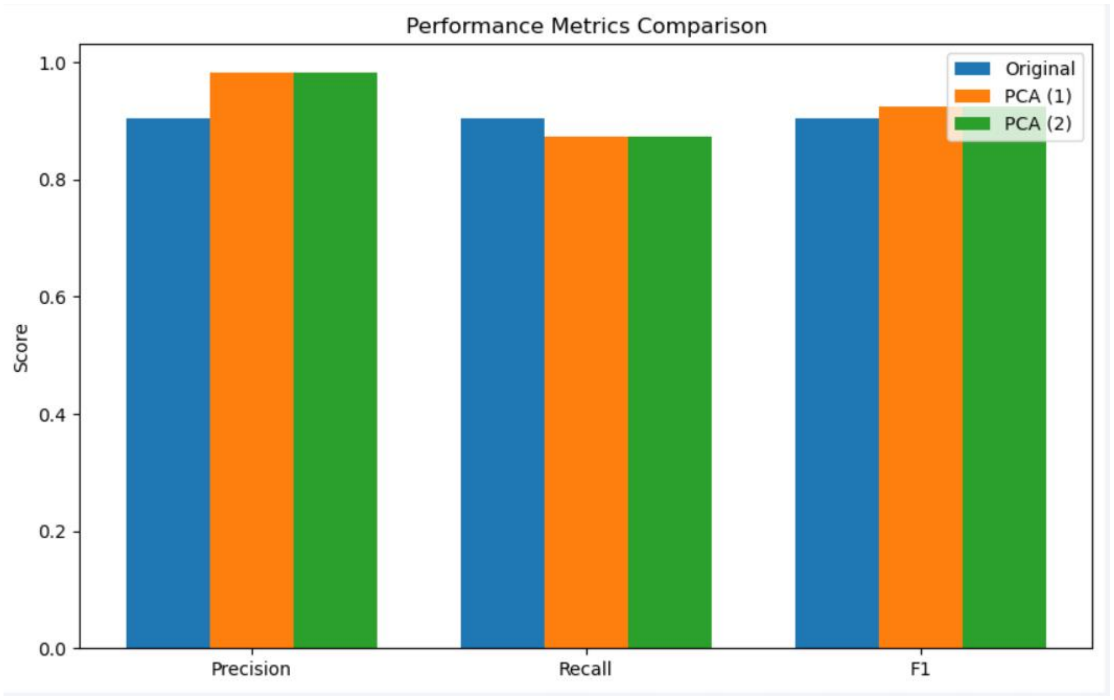
Decision Tree Model Evaluation Results with PCA (PC1):


```
-----------Decision Tree Model Evaluation Results with PCA (PC1)--------------
Proportion of variance explained by PC1: 0.9814
Precision: 0.9821
Recall: 0.8730
F1 Score: 0.9244
```

Decision Tree Model Evaluation Results with PCA (PC1 and PC2):


```
------------Decision Tree Model Evaluation Results with PCA (PC1 and PC2)--------------
Proportion of variance explained by PC1: 0.9814
Proportion of variance explained by PC2: 0.0165
Total proportion of variance explained by PC1 + PC2: 0.9979
Macro Precision: 0.9821
Macro Recall: 0.8730
Macro F1 Score: 0.9244
```

We can obtain precision, recall metrics, and F1 values in three different dimensions: please note the visualization results below:

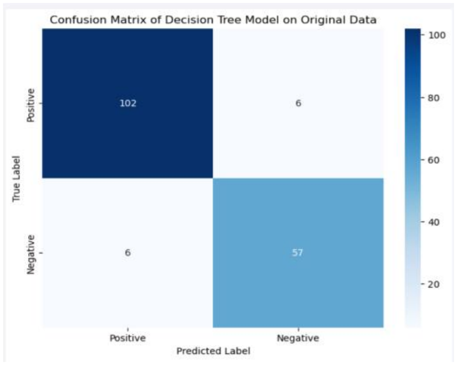


This graph compares the performance of the original data ("Original") and the data processed by two principal component analyses ("PCA (1)" and "PCA (2)") in terms of three metrics: Precision, Recall, and F1 Score. In terms of Precision, "Original" is approximately 0.9, while "PCA (1)" and "PCA (2)" are close to 1.0, with the latter two performing better. Regarding Recall, "Original" is around 0.9, and "PCA (1)" and "PCA (2)" are approximately 0.85, with the original data having a slight edge. For the F1 Score, "Original" is about 0.9, "PCA (1)" is slightly higher than 0.9, and "PCA (2)" is close to 0.9, with "PCA (1)" performing slightly better.It may It implies that the second principal component in PCA(2) exerts minimal or no influence on the model.

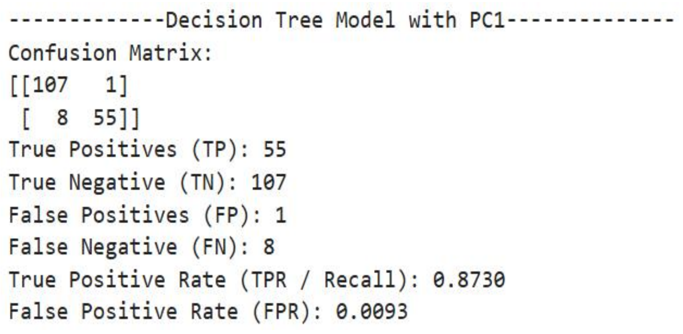
As for Confusion Matrix,we can also get values of FP,TP, FPR and TPR on three different dimensions :

Decision Tree Model on Original Data:

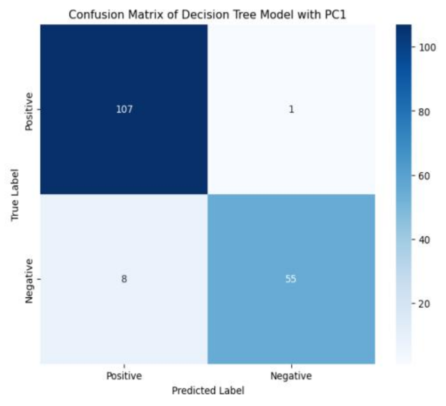
```
-------------Decision Tree Model on Original Data--------------
Confusion Matrix:
[[102   6]
 [  6  57]]
True Positives (TP): 57
True Negative (TN): 102
False Positives (FP): 6
False Negative (FN): 6
True Positive Rate (TPR / Recall): 0.9048
False Positive Rate (FPR / Recall): 0.0556
```



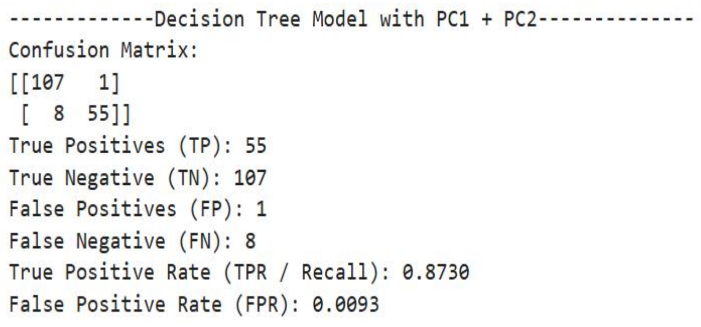
Decision Tree Model with PC1:



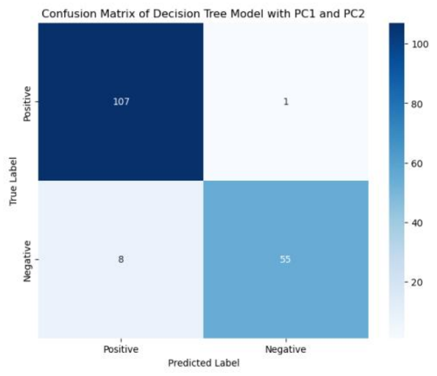
```
-------------Decision Tree Model with PC1-------------
Confusion Matrix:
[[107   1]
 [  8  55]]
True Positives (TP): 55
True Negative (TN): 107
False Positives (FP): 1
False Negative (FN): 8
True Positive Rate (TPR / Recall): 0.8730
False Positive Rate (FPR): 0.0093
```



Decision Tree Model with PC1 + PC2:



```
------------Decision Tree Model with PC1 + PC2-------------
Confusion Matrix:
[[107   1]
 [  8  55]]
True Positives (TP): 55
True Negative (TN): 107
False Positives (FP): 1
False Negative (FN): 8
True Positive Rate (TPR / Recall): 0.8730
False Positive Rate (FPR): 0.0093
```



Analysis：

In terms of the True Positive Rate (TPR), the TPR of the original data is 0.9048, which is higher than the TPR of 0.8730 after dimensionality reduction using PCA (either with one principal component or two principal components), indicating that the original continuous data has a slightly stronger ability to identify positive examples compared to the model after dimensionality reduction. In terms of the False Positive Rate (FPR), the FPR of the original data is 0.0556, which is also higher than the FPR of 0.0093 after dimensionality reduction. The model after PCA dimensionality reduction performs better in reducing the misclassification of negative examples as positive examples.

The evaluation metrics obtained by using the original continuous data are satisfactory. Meanwhile, all the metrics of the model after PCA dimensionality reduction also remain at a fairly good level. **From the perspective of evaluation, using continuous data is of some help to the model. However, it should be noted that using continuous data nay not the optimal choice for the model.** Although directly using the original continuous data can retain more information, it may introduce noise and redundant content, thus limiting the model's performance. In contrast, the continuous data processed through dimensionality reduction (such as principal component analysis, i.e., PCA) has more advantages. It can effectively eliminate noise, reduce the possibility of overfitting, and improve the model's classification performance and generalization ability.