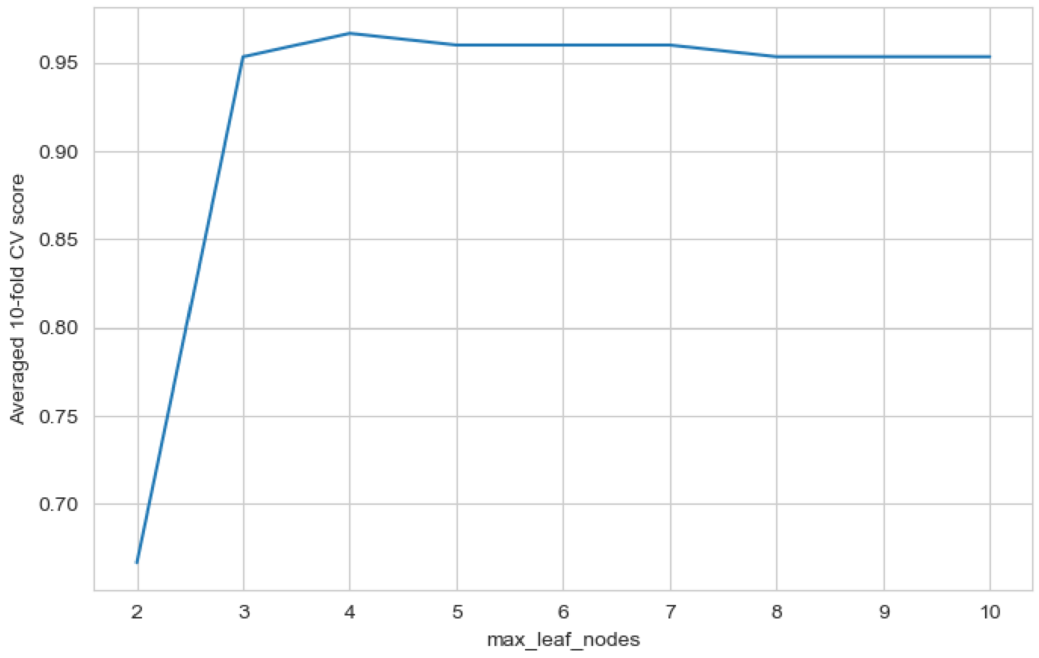
Q1: Using the optimal max depth obtained in Task 4, create a model for a range of ‘max_leaf_nodes’ (2:10). Provide the plot of averaged CV scores for all’ max_leaf_nodes‘ , tunings. What is the best max score and the best leaf node value?

```
maxleaf_cv = []
leaf_counts = []
for n in range(2, 11):
    dt = DecisionTreeClassifier(max_leaf_nodes=n, random_state=47)
    dt.fit(X_train, y_train)
    predict = dt.predict(X_test)
    cv = cross_val_score(dt, predictors, target, cv=10)
    nodecount = dt.tree_.node_count
    print("max_leaf_nodes={}".format(n), "Average 10-Fold CV Score:{}".format(np.mean(cv)),
          "Node count:{}".format(nodecount))
    maxleaf_cv.append(np.mean(cv))
    leaf_counts.append(nodecount)
# Plot averaged CV scores for all max_leaf_nodes tunings
fig, axes = plt.subplots(1, 1, figsize=(8, 5))
axes.set_xticks(range(2, 11))
k = range(2, 11)
plt.plot(k, maxleaf_cv)
plt.xlabel("max_leaf_nodes")
plt.ylabel("Averaged 10-fold CV score")
plt.show()
```

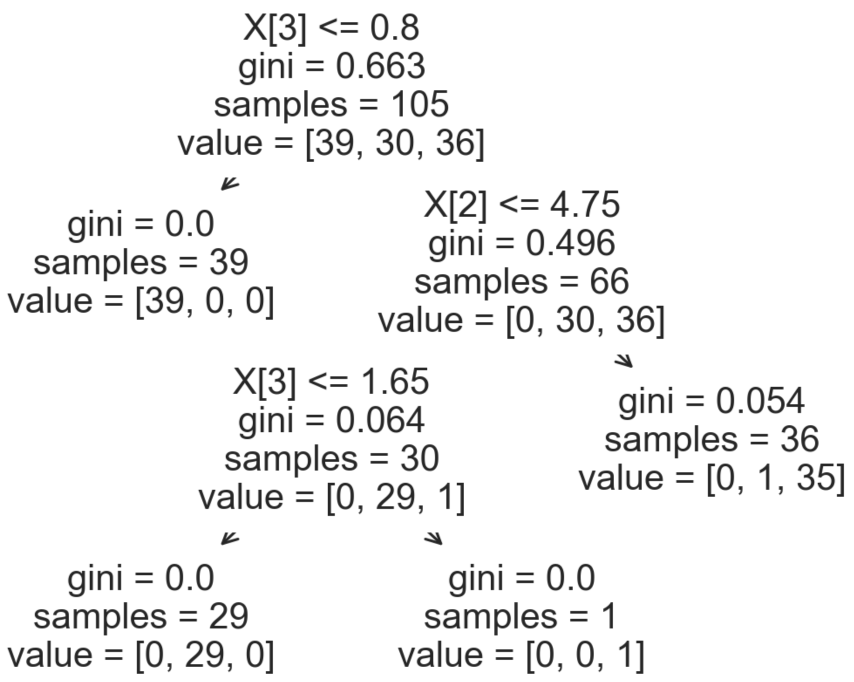
```
max_leaf_nodes=2 Average 10-Fold CV Score:0.6666666666666667 Node count:3
max_leaf_nodes=3 Average 10-Fold CV Score:0.9533333333333334 Node count:5
max_leaf_nodes=4 Average 10-Fold CV Score:0.9666666666666666 Node count:7
max_leaf_nodes=5 Average 10-Fold CV Score:0.96 Node count:9
max_leaf_nodes=6 Average 10-Fold CV Score:0.96 Node count:11
max_leaf_nodes=7 Average 10-Fold CV Score:0.96 Node count:11
max_leaf_nodes=8 Average 10-Fold CV Score:0.9533333333333334 Node count:11
max_leaf_nodes=9 Average 10-Fold CV Score:0.9533333333333334 Node count:11
max_leaf_nodes=10 Average 10-Fold CV Score:0.9533333333333334 Node count:11
```



The best max score is 4.

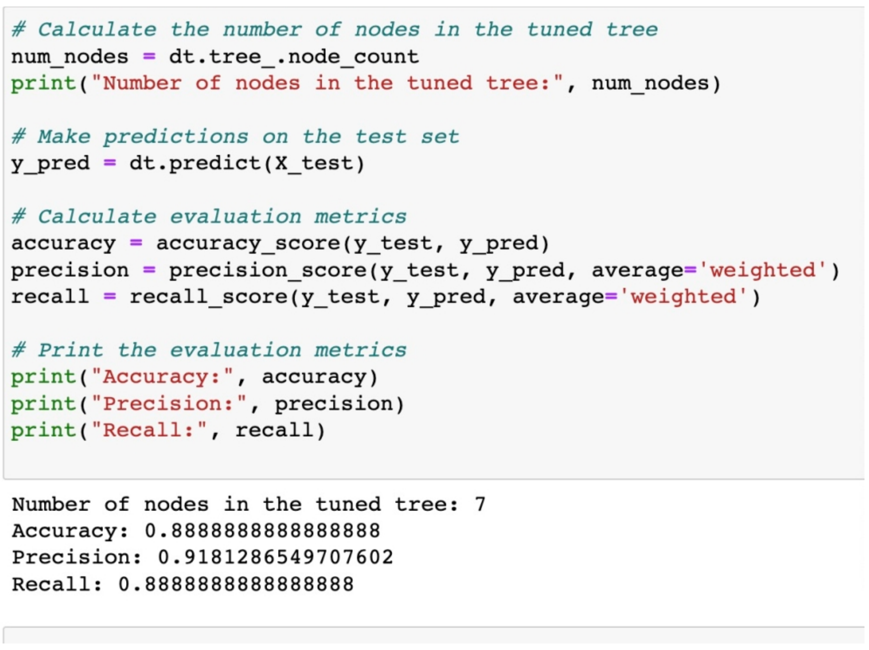
The best leaf node value is 0.966.

Q2: Tune your tree by providing the optimal 'max_leaf_nodes' obtained in Q1. Provide your tuned tree's plot (Adjust the resolution of the plot so it is readable). Explain your tree.



I created a decision tree with a maximum depth of 3 and a maximum of 4 leaf nodes. For the root node, the x[3] represents the petal_wid, if it is <= 0.8, it goes left child. Others go right child, and right child has two children, and if x[2] <= 4.75, goes left child x[3], right child is gini = 0.054. For x[3] <= 1.65, it goes left child which is gini = 0, samples = 29, otherwise, it goes gini = 0.0, samples = 1.

Q3: How many nodes are used in your tuned tree? Calculate and provide the evaluation metrics including 'accuracy score, Precision and Recall). Explain your findings.



```
# Calculate the number of nodes in the tuned tree
num_nodes = dt.tree_.node_count
print("Number of nodes in the tuned tree:", num_nodes)

# Make predictions on the test set
y_pred = dt.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
```

```
Number of nodes in the tuned tree: 7
Accuracy: 0.8888888888888888
Precision: 0.9181286549707602
Recall: 0.8888888888888888
```

7 nodes are used in my tuned tree. There are 3 level of the tree, and 4 leaf nodes.
Accuracy is 0.889 which indicate the model correctly classifies approximately 88.89% of the samples in the test set.
Precision is 0.918 means 91.8% of the samples predicted a certain class are actually belonging to that class.
Recall is 0.889 means the model can correctly identify approximately 88.9% of the samples that belong to a specific class.