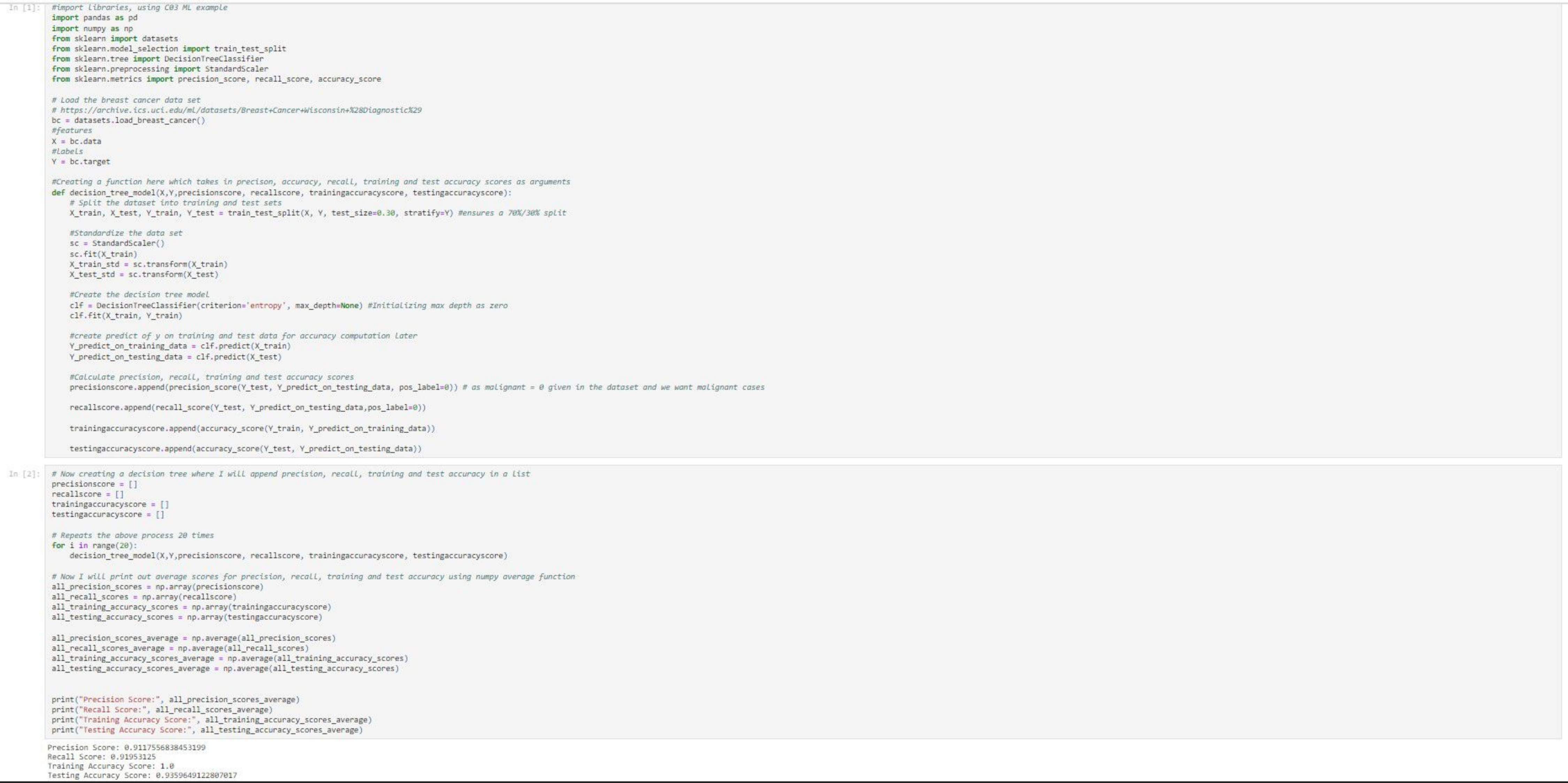


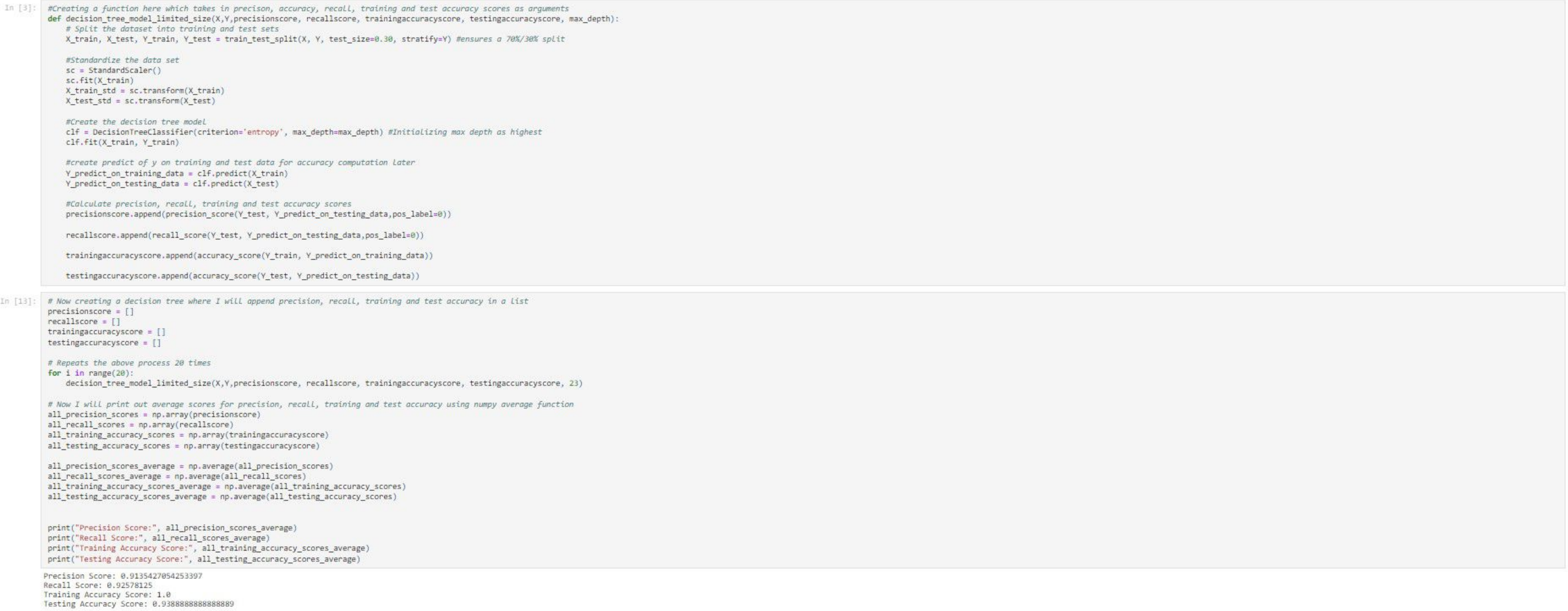
In [1]:

```python
#import libraries, using C03 ML example
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import precision_score, recall_score, accuracy_score

# Load the breast cancer data set
# https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
bc = datasets.load_breast_cancer()
#features
X = bc.data
#labels
Y = bc.target

#Creating a function here which takes in precison, accuracy, recall, training and test accuracy scores as arguments
def decision_tree_model(X,Y,precisionscore, recallscore, trainingaccuracyscore, testingaccuracyscore):
    # Split the dataset into training and test sets
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, stratify=Y) #ensures a 70%/30% split

    #Standardize the data set
    sc = StandardScaler()
    sc.fit(X_train)
    X_train_std = sc.transform(X_train)
    X_test_std = sc.transform(X_test)

    #Create the decision tree model
    clf = DecisionTreeClassifier(criterion='entropy', max_depth=None) #Initializing max depth as zero
    clf.fit(X_train, Y_train)

    #create predict of y on training and test data for accuracy computation later
    Y_predict_on_training_data = clf.predict(X_train)
    Y_predict_on_testing_data = clf.predict(X_test)

    #Calculate precision, recall, training and test accuracy scores
    precisionscore.append(precision_score(Y_test, Y_predict_on_testing_data, pos_label=0)) # as malignant = 0 given in the dataset and we want malignant cases

    recallscore.append(recall_score(Y_test, Y_predict_on_testing_data,pos_label=0))

    trainingaccuracyscore.append(accuracy_score(Y_train, Y_predict_on_training_data))

    testingaccuracyscore.append(accuracy_score(Y_test, Y_predict_on_testing_data))
```

In [2]:

```python
# Now creating a decision tree where I will append precision, recall, training and test accuracy in a list
precisionscore = []
recallscore = []
trainingaccuracyscore = []
testingaccuracyscore = []

# Repeats the above process 20 times
for i in range(20):
    decision_tree_model(X,Y,precisionscore, recallscore, trainingaccuracyscore, testingaccuracyscore)

# Now I will print out average scores for precision, recall, training and test accuracy using numpy average function
all_precision_scores = np.array(precisionscore)
all_recall_scores = np.array(recallscore)
all_training_accuracy_scores = np.array(trainingaccuracyscore)
all_testing_accuracy_scores = np.array(testingaccuracyscore)

all_precision_scores_average = np.average(all_precision_scores)
all_recall_scores_average = np.average(all_recall_scores)
all_training_accuracy_scores_average = np.average(all_training_accuracy_scores)
all_testing_accuracy_scores_average = np.average(all_testing_accuracy_scores)


print("Precision Score:", all_precision_scores_average)
print("Recall Score:", all_recall_scores_average)
print("Training Accuracy Score:", all_training_accuracy_scores_average)
print("Testing Accuracy Score:", all_testing_accuracy_scores_average)
```

```
Precision Score: 0.9117556838453199
Recall Score: 0.91953125
Training Accuracy Score: 1.0
Testing Accuracy Score: 0.9359649122807017
```



In [3]:

```python
#Creating a function here which takes in precison, accuracy, recall, training and test accuracy scores as arguments
def decision_tree_model_limited_size(X,Y,precisionscore, recallscore, trainingaccuracyscore, testingaccuracyscore, max_depth):
    # Split the dataset into training and test sets
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, stratify=Y) #ensures a 70%/30% split

    #Standardize the data set
    sc = StandardScaler()
    sc.fit(X_train)
    X_train_std = sc.transform(X_train)
    X_test_std = sc.transform(X_test)

    #Create the decision tree model
    clf = DecisionTreeClassifier(criterion='entropy', max_depth=max_depth) #Initializing max depth as highest
    clf.fit(X_train, Y_train)

    #create predict of y on training and test data for accuracy computation later
    Y_predict_on_training_data = clf.predict(X_train)
    Y_predict_on_testing_data = clf.predict(X_test)

    #Calculate precision, recall, training and test accuracy scores
    precisionscore.append(precision_score(Y_test, Y_predict_on_testing_data,pos_label=0))

    recallscore.append(recall_score(Y_test, Y_predict_on_testing_data,pos_label=0))

    trainingaccuracyscore.append(accuracy_score(Y_train, Y_predict_on_training_data))

    testingaccuracyscore.append(accuracy_score(Y_test, Y_predict_on_testing_data))
```

In [13]:

```python
# Now creating a decision tree where I will append precision, recall, training and test accuracy in a list
precisionscore = []
recallscore = []
trainingaccuracyscore = []
testingaccuracyscore = []

# Repeats the above process 20 times
for i in range(20):
    decision_tree_model_limited_size(X,Y,precisionscore, recallscore, trainingaccuracyscore, testingaccuracyscore, 23)

# Now I will print out average scores for precision, recall, training and test accuracy using numpy average function
all_precision_scores = np.array(precisionscore)
all_recall_scores = np.array(recallscore)
all_training_accuracy_scores = np.array(trainingaccuracyscore)
all_testing_accuracy_scores = np.array(testingaccuracyscore)

all_precision_scores_average = np.average(all_precision_scores)
all_recall_scores_average = np.average(all_recall_scores)
all_training_accuracy_scores_average = np.average(all_training_accuracy_scores)
all_testing_accuracy_scores_average = np.average(all_testing_accuracy_scores)


print("Precision Score:", all_precision_scores_average)
print("Recall Score:", all_recall_scores_average)
print("Training Accuracy Score:", all_training_accuracy_scores_average)
print("Testing Accuracy Score:", all_testing_accuracy_scores_average)
```

```
Precision Score: 0.9135427054253397
Recall Score: 0.92578125
Training Accuracy Score: 1.0
Testing Accuracy Score: 0.9388888888888889
```