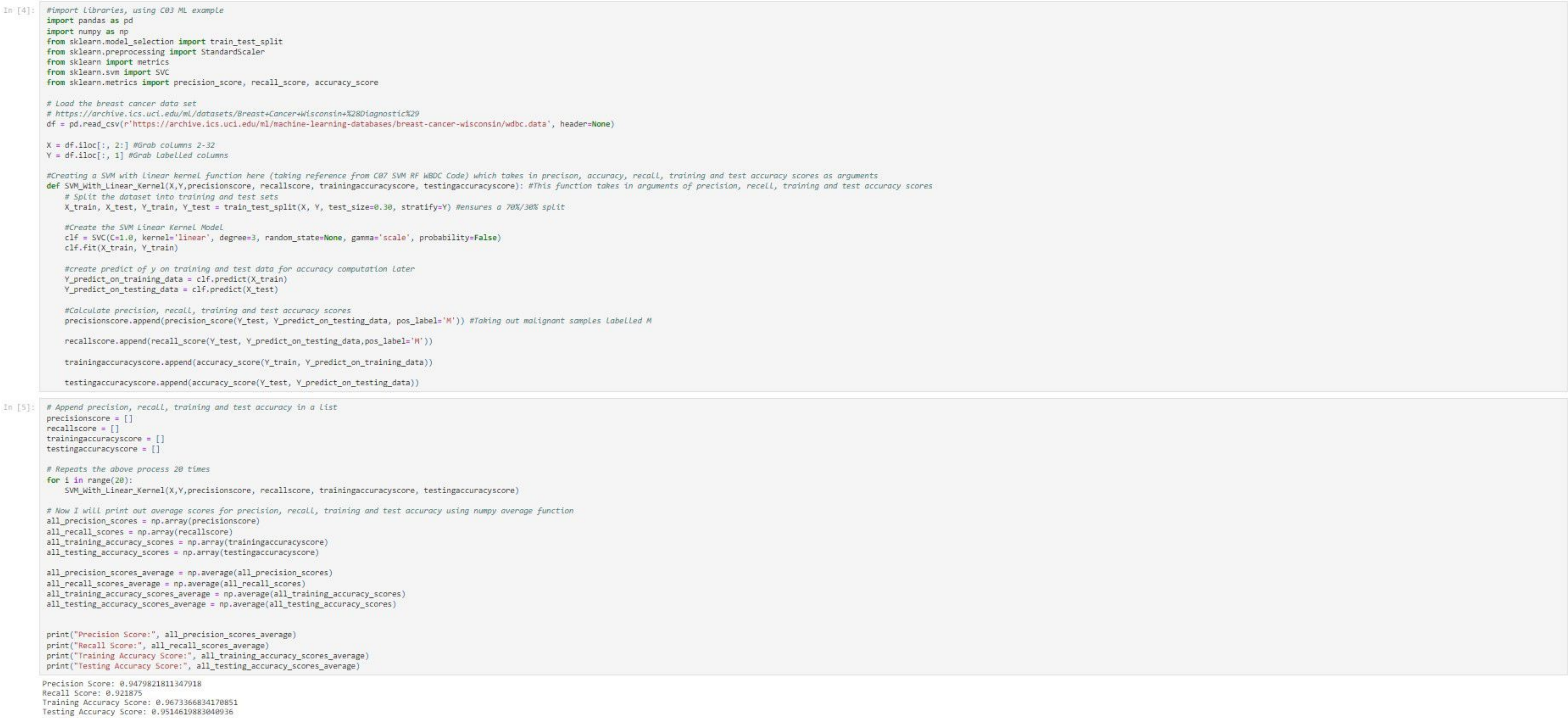


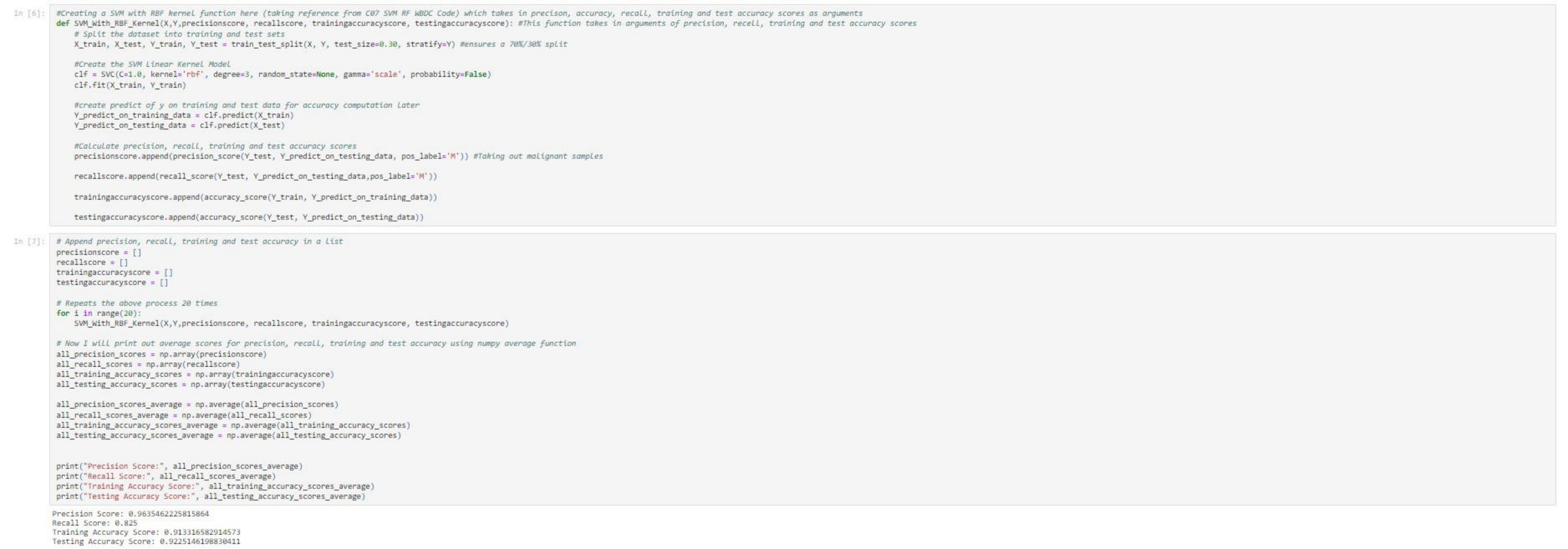
In [4]:

```
#import libraries, using C03 ML example
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
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
df = pd.read_csv(r'https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data', header=None)

X = df.iloc[:, 2:] #Grab columns 2-32
Y = df.iloc[:, 1] #Grab labelled columns

#Creating a SVM with linear kernel function here (taking reference from C07 SVM RF WBDC Code) which takes in precison, accuracy, recall, training and test accuracy scores as arguments
def SVM_With_Linear_Kernel(X,Y,precisionscore, recallscore, trainingaccuracyscore, testingaccuracyscore): #This function takes in arguments of precision, recell, training and test accuracy scores
    # 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

    #Create the SVM Linear Kernel Model
    clf = SVC(C=1.0, kernel='linear', degree=3, random_state=None, gamma='scale', probability=False)
    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='M')) #Taking out malignant samples labelled M

    recallscore.append(recall_score(Y_test, Y_predict_on_testing_data,pos_label='M'))

    trainingaccuracyscore.append(accuracy_score(Y_train, Y_predict_on_training_data))

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

In [5]:

```
# 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):
    SVM_With_Linear_Kernel(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: 0.9479821811347918
Recall Score: 0.921875
Training Accuracy Score: 0.9673366834170851
Testing Accuracy Score: 0.9514619883040936
```



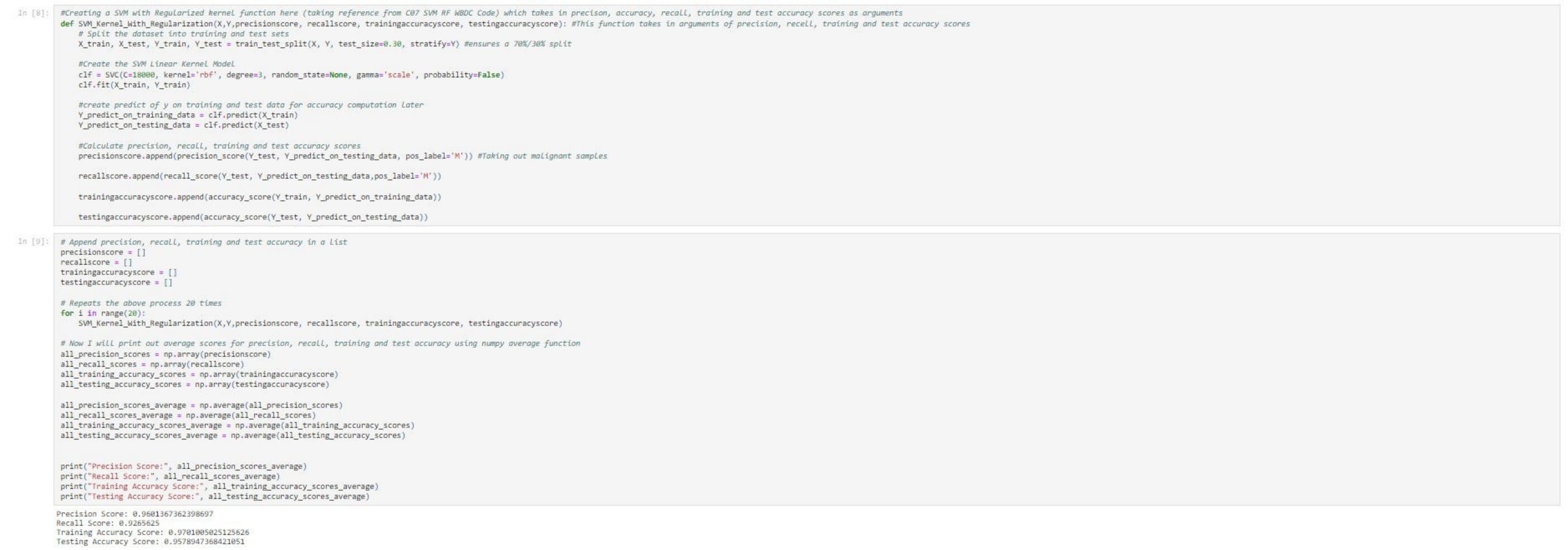
In [6]:

```python
#Creating a SVM with RBF kernel function here (taking reference from C07 SVM RF WBDC Code) which takes in precison, accuracy, recall, training and test accuracy scores as arguments
def SVM_With_RBF_Kernel(X,Y,precisionscore, recallscore, trainingaccuracyscore, testingaccuracyscore): #This function takes in arguments of precision, recell, training and test accuracy scores
    # 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

    #Create the SVM Linear Kernel Model
    clf = SVC(C=1.0, kernel='rbf', degree=3, random_state=None, gamma='scale', probability=False)
    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='M')) #Taking out malignant samples

    recallscore.append(recall_score(Y_test, Y_predict_on_testing_data,pos_label='M'))

    trainingaccuracyscore.append(accuracy_score(Y_train, Y_predict_on_training_data))

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

In [7]:

```python
# 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):
    SVM_With_RBF_Kernel(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: 0.9635462225815864
Recall Score: 0.825
Training Accuracy Score: 0.913316582914573
Testing Accuracy Score: 0.9225146198830411
```



```
In [8]: #Creating a SVM with Regularized kernel function here (taking reference from C07 SVM RF WBDC Code) which takes in precison, accuracy, recall, training and test accuracy scores as arguments
        def SVM_Kernel_With_Regularization(X,Y,precisionscore, recallscore, trainingaccuracyscore, testingaccuracyscore): #This function takes in arguments of precision, recell, training and test accuracy scores
            # 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

            #Create the SVM Linear Kernel Model
            clf = SVC(C=18000, kernel='rbf', degree=3, random_state=None, gamma='scale', probability=False)
            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='M')) #Taking out malignant samples

            recallscore.append(recall_score(Y_test, Y_predict_on_testing_data,pos_label='M'))

            trainingaccuracyscore.append(accuracy_score(Y_train, Y_predict_on_training_data))

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

```
In [9]: # 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):
            SVM_Kernel_With_Regularization(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: 0.9601367362398697
Recall Score: 0.9265625
Training Accuracy Score: 0.9701005025125626
Testing Accuracy Score: 0.9578947368421051
```