In [4]:	#import Libraries, using CO3 ML example
	import pandas as pd
	import numpy as np from sklearn.model_selection import train test_split
	from skeen-model_selection_import_train_ender_
	Trom skeern.primort metrics from skeern import metrics
	From skeleri anyon import SVC
	from sklearn.metrics import precision score, recall score, accuracy score
	The manufacture and the control of t
	# 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): #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(c1.0, 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))
	trainingacturacysture.append acturacy ture (r_train, r_predict_on_training data))
	testingaccuracyscore.append(accuracy_score(Y_test, Y_predict_on_testing_data))
to 151-	# Append precision, recall, training and test accuracy in a list
711 [2]3	# Append precises, *reduct, training and test accuracy in a cist
	recallscore = []
	trainingacurayscore = []
	testingacuracyscore = []
	Notice Working February 2015
	# Repeats the above process 20 times
	for i in range(20):
	SWM_With_Linear_Kernel(X,Y,precisionscore, recallscore, trainingaccuracyscore)
	# Now I will print out average scores for precision, recall, training and test accuracy using numpy average function
	all_precision_scores = np.array(recaliscore) all_precision_scores = np.array(recaliscore)
	all_reall_scores = mp.arsy(recaliscore) all_tealing_accuracy_scores = mp.arsy(trainingaccuracyscore)
	ail_training_accuracy_scores = np.array(training_accuracyscore) all_testing_accuracy_scores = np.array(training_accuracyscore)
	and the same and t
	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 Scores:", all_precision_scores_average)
	print("Recall Score:", all_recall_scores_average)
	print("Training Accuracy Score:", all_training_accuracy_scores_average) print("Esting Accuracy Score:", all_training_accuracy_scores_average)
1	What testing were end and end of the control of the

Precision Score: 0.9479821811347918 Recall Score: 0.921875 Training Accuracy Score: 0.9673366834170851 Testing Accuracy Score: 0.9514619883040936

	der SWENTEN Net Aerhei(X,Y,perisionscore, recaliscore, trainingacuracyscore, testingacuracyscore): #Inis 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)
	<pre>#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)</pre>
	#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]:	# 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)
	# Now I will print out average scores for precision, recall, training and test accuracy using numpy average function all_precision_scores = np.array(precisionscore)

In [6]: #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

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

dof CM With DDE Vanal / V marketing rate and a restriction and that agreement and the control of marketing rate of marke

for i in range(20):

SVM_with_RBF_Kernel(X,Y,precisionscore, recallscore, trainingaccuracyscore)

Now I will print out average scores for precision, recall, training and test accuracy using numpy average function all precision scores = np.array(crecisionscore)

all_precision_scores = np.array(crecisionscore)

all_training_accuracy_scores = np.array(trainingaccuracyscore)

all_training_accuracy_scores = np.array(trainingaccuracyscore)

all_precision_scores_average = np.average(all_precision_scores)

all_precision_scores_average = np.average(all_precision_scores)

all_training_accuracy_scores_average = np.average(all_recall_scores)

print("Precision_scores_average = np.average(all_training_accuracy_scores)

print("Precision_scores_average = np.average(all_training_accuracy_scores)

print("Precision_scores_average = np.average(all_training_accuracy_scores_average)

print("Training_accuracy_scores_average)

print("Training_accuracy_scores_average)

In [8]	#Creating a SVM with Regularized kernel function here (taking reference from COT SVM RF NBDC 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
	<pre>clf = SVC(C=18080, kernel='rbf', degree=3, random_state=None, gamma='scale', probability=False)</pre>
	<pre>clf.fit(X_train, Y_train)</pre>
	#create predict of y on training and test data for accuracy computation later
	werease preases of y on training and sets away for accuracy computation later Y predict on training data of It-predict(X train)
	* predict_on_training_data = car.predict(%_train) Y predict on testing data = clf.predict(% test)
	i_breatt_oi_testing_data = cirt/preatt(i_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)
	# Now I will print out average scores for precision, recall, training and test accuracy using numpy average function
	all precision scores = no array(precisionscore)
	all reals scores = np.ar apprecisation of
	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)

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("Testing_Accuracy_Scores", all_training_accuracy_scores_average)
print("Testing_Accuracy_Scores", all_training_accuracy_scores_average)

To INI: Wheating a CIM with Pagulaginal happal function happal function happal function and fact accuracy record or pagulagina accuracy record or pagulagina accuracy record o

Precision Score: 0.9601367362398697 Recall Score: 0.9265625

Training Accuracy Score: 0.9701005025125626 Testing Accuracy Score: 0.9578947368421051