Ziffernpaar	3&5	3&7	3&8	5&7	5&8	7&8
Klassifikationsgüte	**0.929447852761**	**0.974440894569**	**0.951807228916**	**0.983713355049**	**0.960122699387**	**0.97124600639**

Die Testdaten wurden wie in der Vorlesung besprochen mit Hilfe eine linearen Hyperebene klassifiziert.

Codeauszug:Berechnung der Hyperebene

```
def solveLinearRegression(A,B):
    number_of_rowsA = len(A)
    number_of_rowsB = len(B)

    y_A = np.ones(number_of_rowsA)
    y_B = (-1)*np.ones(number_of_rowsB)
    y   = np.matrix(composeMatrices((y_A,y_B)))

    C = composeMatrices((A,B))
    if np.linalg.det(C.T*C)!=0:
        return (C.T*C).I*C.T*y.T
    else:
        n = len(C.T*C)
        eps = 0.0001
        i = 0
        while np.linalg.det(C.T*C +eps*2**i*np.identity(n))==0:
            i=i+1
        return (C.T*C+eps*2**i*np.identity(n)).I*C.T*y.T
```

Mit Hilfe der linearen Hyperebene wurden dann die Testdaten klassifiziert.

Codeauszug:Berechnung der Klassenzugehörigkeit und Klassifikationsgüte

```
def classifyObjects(train_Set1, train_Set2, test_Set1, test_Set2):

    train_Matrix1 = readTrainingSet(train_Set1)
    train_Matrix2 = readTrainingSet(train_Set2)

    classifier = solveLinearRegression(train_Matrix1, train_Matrix2).T

    test_Matrix = composeMatrices((test_Set1, test_Set2)).T

    test_objects1 = len(test_Set1)
    test_objects2 = len(test_Set2)
```

```
classVector1 = np.matrix(np.ones(test_objects1)).T
classVector2 = np.matrix((-1)*np.ones(test_objects2)).T

classVector = np.matrix(composeMatrices((classVector1,classVector2))).T

classified_Objects = np.sign(classifier*test_Matrix)

cc_proportion = ((classVector*classified_Objects.T).item()
                    +len(classVector.T))/  (2*len(classVector.T))


return cc_proportion
```