Method for leave-one-out cross validation. Accepts inputdata, outputdata and model to be fitted to training data as parameters. Return predicted output labels.

In [1]:
```
#Accept inputdata, outputdata and model to be fitted training data as parameters. Re

def leaveone (inputdata, outputdata, model):
    x_training = inputdata.copy()
    y_training = outputdata.copy()
    y_preds = []
    y_trues = []
    datasize = len(inputdata)
    for i in range (0, datasize):
        x_training = inputdata.drop(inputdata.index[i])
        x_test = inputdata.iloc[i]
        y_training = outputdata.drop(outputdata.index[i])
        y_test = outputdata.iloc[i]
        model.fit(x_training, y_training)
        y_pred = neigh.predict(np.array(x_test).reshape(1,-1))
        y_preds.append(y_pred)
        y_trues.append(y_test)
    return y_preds
```

Method for leave-group-out cross validation In this leave-group-out crossvalidation each group has 4 instances and they have been sorted so that the 4 are always together with indexs 0-3 having one group, index 4-7 other group and so on. Return predicted outputdata labels.

In [5]:
```
#In this leave-group-out crossvalidation each group has 4 instances and they have be
#return predicted outputdata labels
def leavegroup (inputdata, outputdata, model):
    y_preds = []
    y_trues = []
    for i in range (0, len(grouped), 4):

        y_training = outputdata.copy()
        x_training = inputdata.copy()

        valuex1 = inputdata.iloc[i]
        valuex2 = inputdata.iloc[i+1]
        valuex3 = inputdata.iloc[i+2]
        valuex4 = inputdata.iloc[i+3]

        valuey1 = outputdata.iloc[i]
        valuey2 = outputdata.iloc[i+1]
        valuey3 = outputdata.iloc[i+2]
        valuey4 = outputdata.iloc[i+3]

        x_test = [valuex1, valuex2, valuex3, valuex4]
        y_test = [valuey1, valuey2, valuey3, valuey4]
        y_training = outputdata.drop(outputdata.index[i:i+4])
        x_training = inputdata.drop(inputdata.index[i:i+4])
        y_training2 = np.array(y_training).reshape(-1,1)
        model.fit(x_training, y_training2)
        y_pred = neigh.predict(np.array(x_test))
        y_preds.extend(y_pred.tolist())
        y_trues.extend(y_test)

    return y_preds
```

Methods for spatial leave-one-out crossvalidation. Spatialleaveone accepts inputdata,

outputdata, coordinates, maxdistance and model as parameters. Return predicted output labels.

Calculatedistance takes two coordinates and calculates the distance between them. Returns the result.

Excludetraining accepts trainingset and list of index values that determine which data points of certain indexes will be omitted from training set. Returns a new training set.

In [7]:

```
import numpy as np

def spatialleaveone (inputdata, outputdata, coordinates, maxdistance, model):
    y_preds = list()
    y_trues = list()
    datasize = len(inputdata)
    trainingsize = (datasize -1)
    for i in range (datasize):
        excluded = list()
        x_training = inputdata.copy()
        y_training = outputdata.copy()
        x_test = inputdata.iloc[i]
        y_test = outputdata.iloc[i]
        xcoord = coordinates.iloc[i]
        x_training = inputdata.drop(inputdata.index[i])
        y_training = outputdata.drop(outputdata.index[i])
        coordinatesd = coordinates.drop(coordinates.index[i])
        for k in range (trainingsize):
            ycoord = coordinatesd.iloc[k]
            distance = calculatedistance(xcoord, ycoord)
            if (distance <= maxdistance):
                excluded.append(k)
        x_training1 = excludetraining(x_training, excluded)
        y_training1 = excludetraining(y_training, excluded)
        model.fit(x_training1, y_training1)
        y_pred = neigh.predict(np.array(x_test).reshape(1,-1))
        y_preds.extend(y_pred.tolist())
        y_trues.extend(y_test.values.tolist())
    return y_preds

def calculatedistance (coordinate1, coordinate2):
    xcoord = coordinate1.to_numpy()
    ycoord = coordinate2.to_numpy()
    distance = np.linalg.norm(xcoord-ycoord)
    return distance

def excludetraining (trainingset, excluded):
    training = trainingset.copy()
    excludelist = excluded.copy()
    training = training.drop(training.index[excludelist])
    return training
```