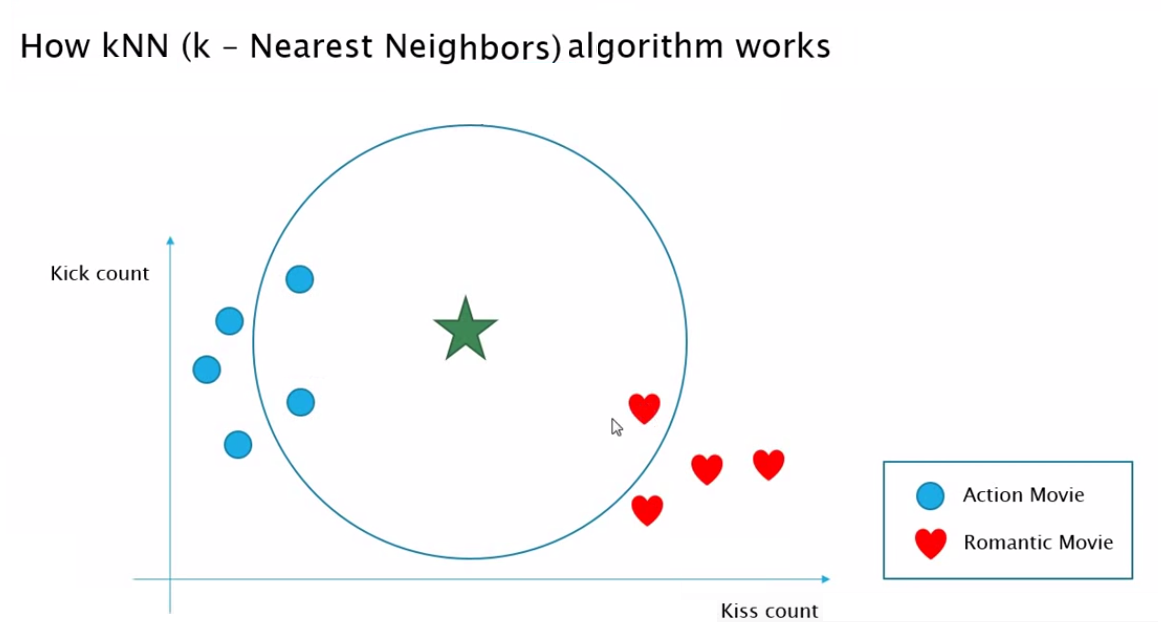
**K- Near Neighbors (KNN)**

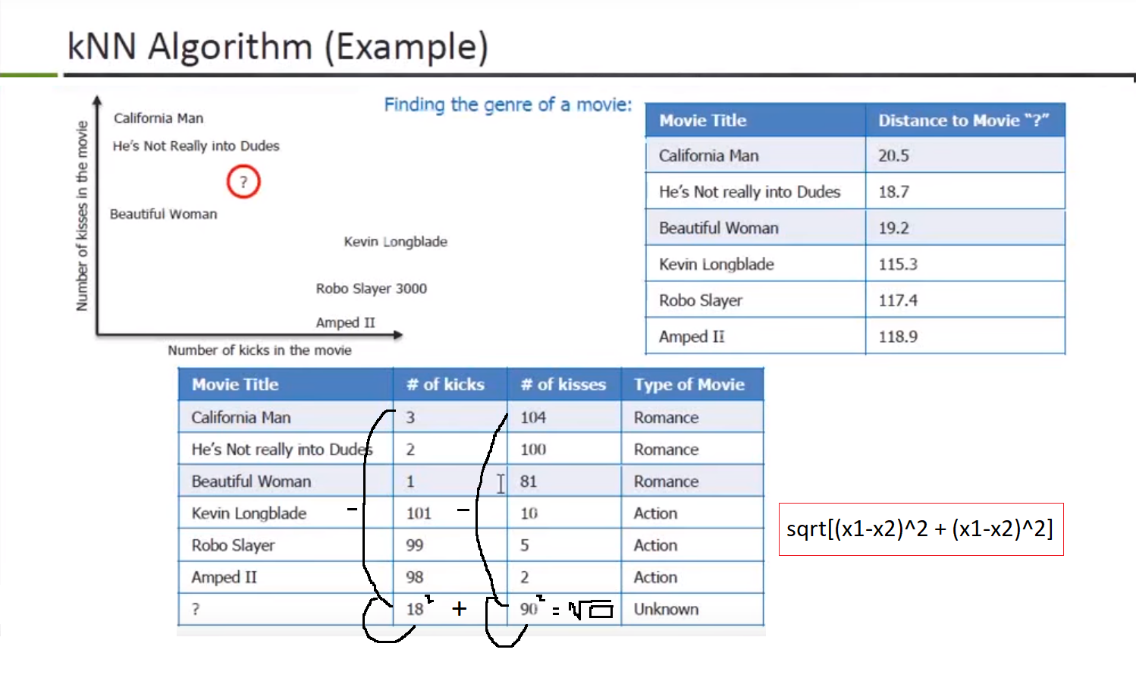
KNN is a simple algorithm in pattern recognition. It is a non-paramentric method, which **measures distance between the scenario** of a single query and a set of scenarios in a data set. It is mainly used for classification and regression

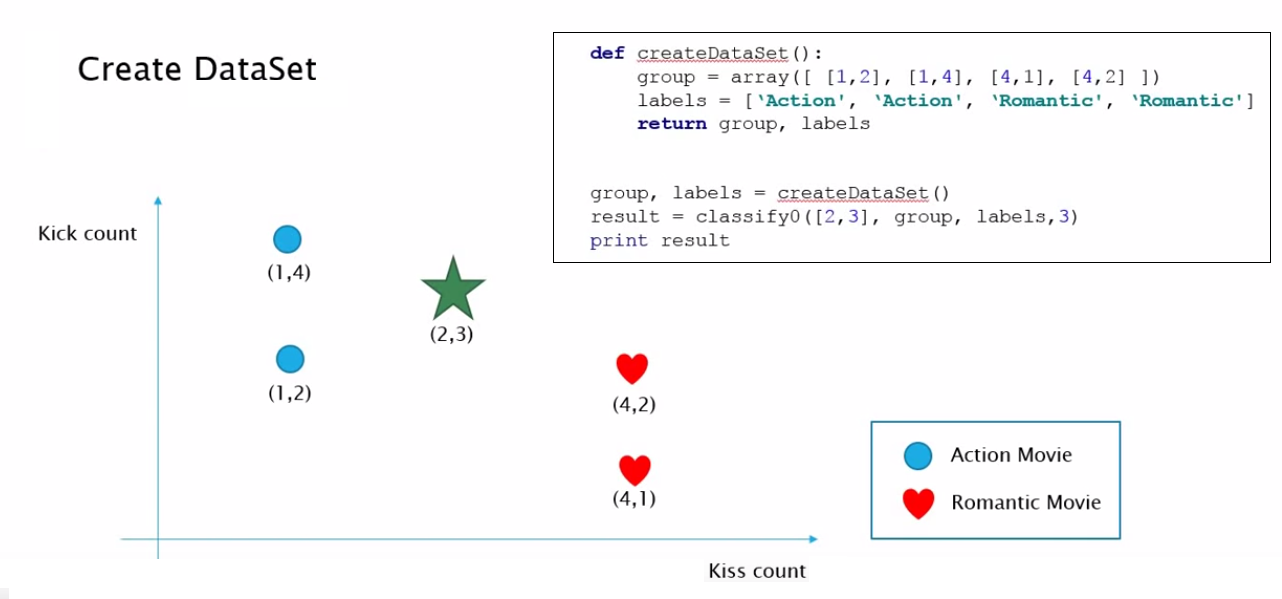
**KNN** is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification

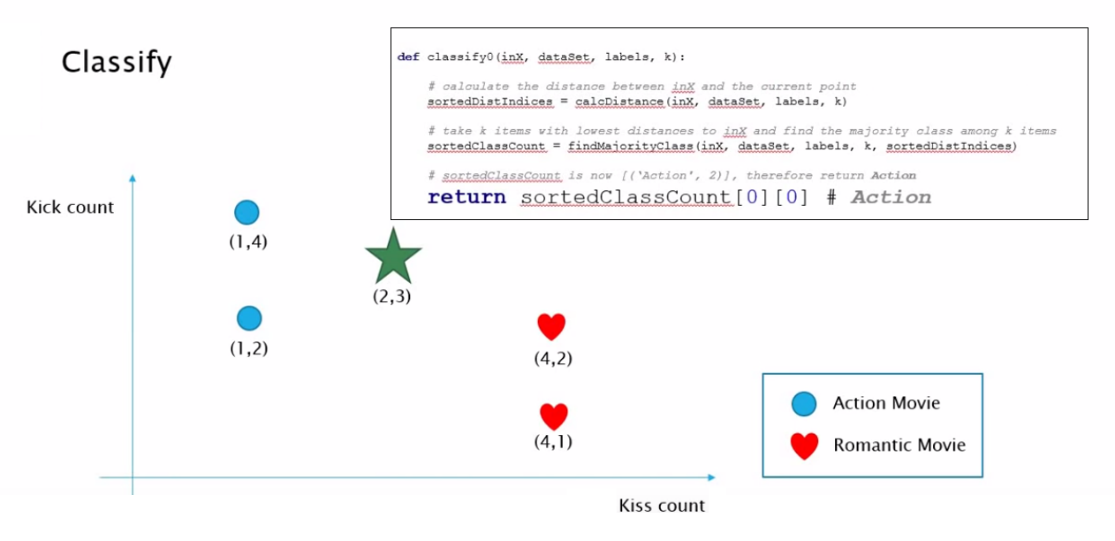
The **KNN algorithm** is among the simplest of all Machine learning Algorithms

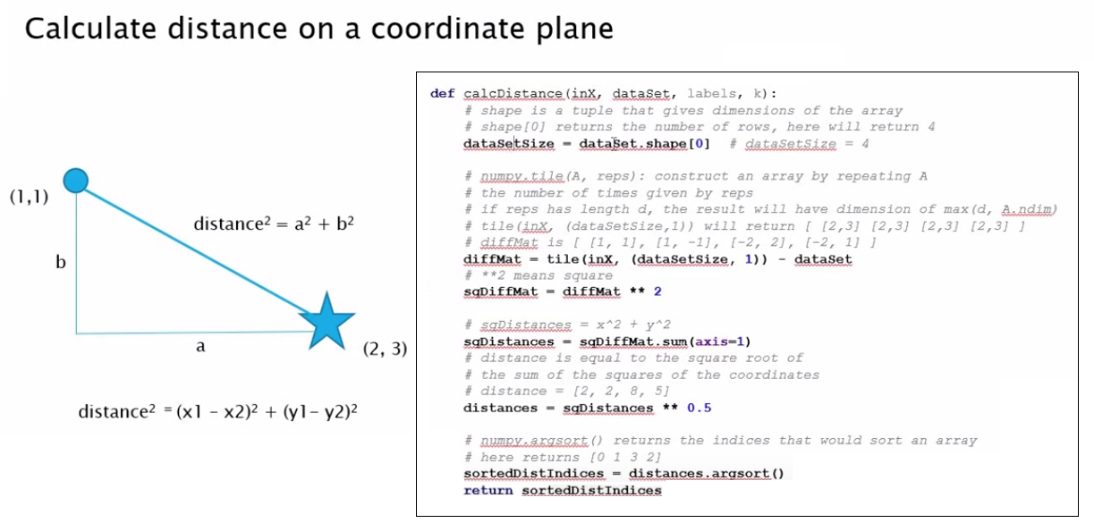
In **KNN classification**, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its **K nearest neighbors** (k is a positive integer typically small)

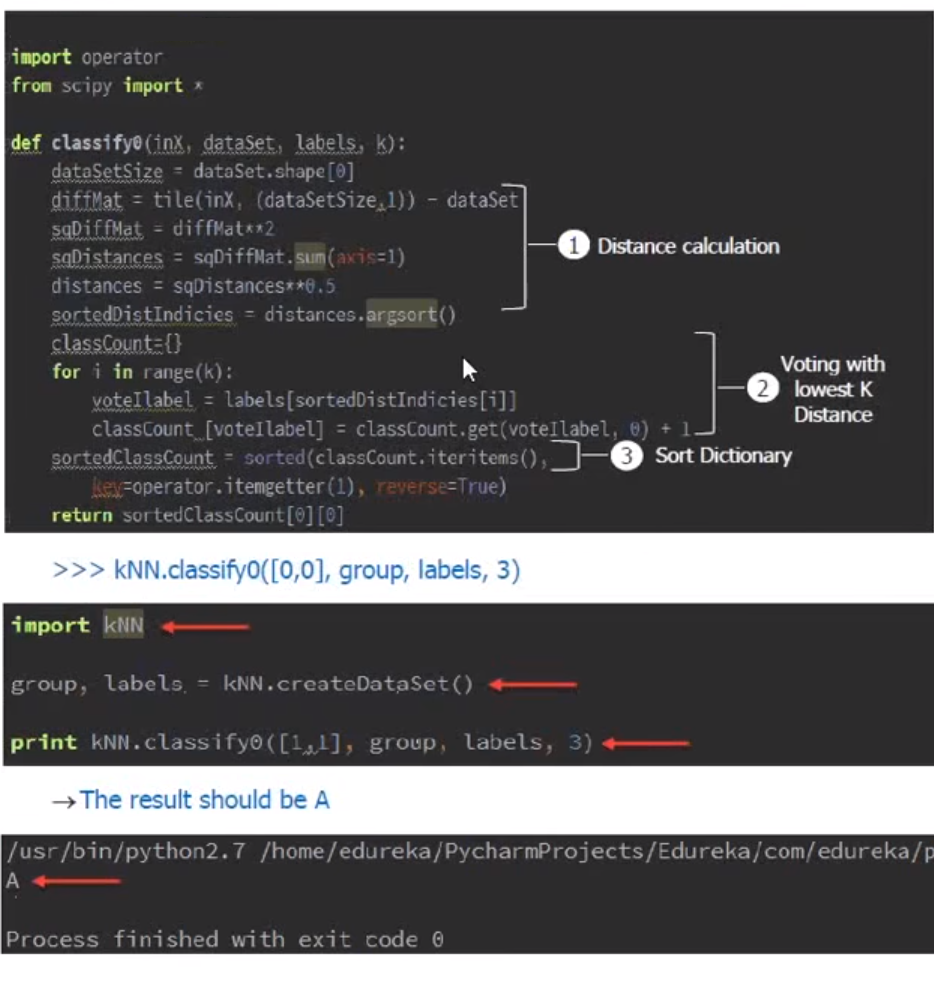










****

**from numpy import \***

**import operator**  
  
  
**class** kNN():  
  
def createDataSet():

group = array([ [1.0, 2.0], [1.0,4.0], [4.0, 1.0], [4.0, 2.0] ])

labels = ['Action', 'Action', 'Romantic', 'Romantic']

return group, labels   
  
def classify0(inX, dataSet, labels, k):

# Distance Calculation

dataSetSize = dataSet.shape[0]

diffMat = tile(inX, (dataSetSize, 1)) - dataSet

sqDiffMat = diffMat \*\* 2

sqDistances = sqDiffMat.sum(axis=1)

distances = sqDistances \*\* 0.5

sortedDistIndicies = distances.argsort()

classCount = {}

#Vointing with lowest K Distance

for i in range(k):

voteIlabel = labels[sortedDistIndicies[i]]

classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1

#Sort Dictionary - #iteritems()= items()

sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1), reverse=True)

return sortedClassCount[0][0]

group, labels = createDataSet()

result = classify0([2.0, 3.0], group, labels,3)

print (result)

>> Action