X-Informatics Case Study: e-Commerce and Life Style Informatics: Recommender Systems Technologies IV: k'th Nearest Neighbor Algorithms (Python Track)

June 19 2013

Geoffrey Fox

gcf@indiana.edu

http://www.infomall.org/X-InformaticsSpring2013/index.html

Associate Dean for Research, School of Informatics and Computing

Indiana University Bloomington

2013

Big Data Ecosystem in One Sentence

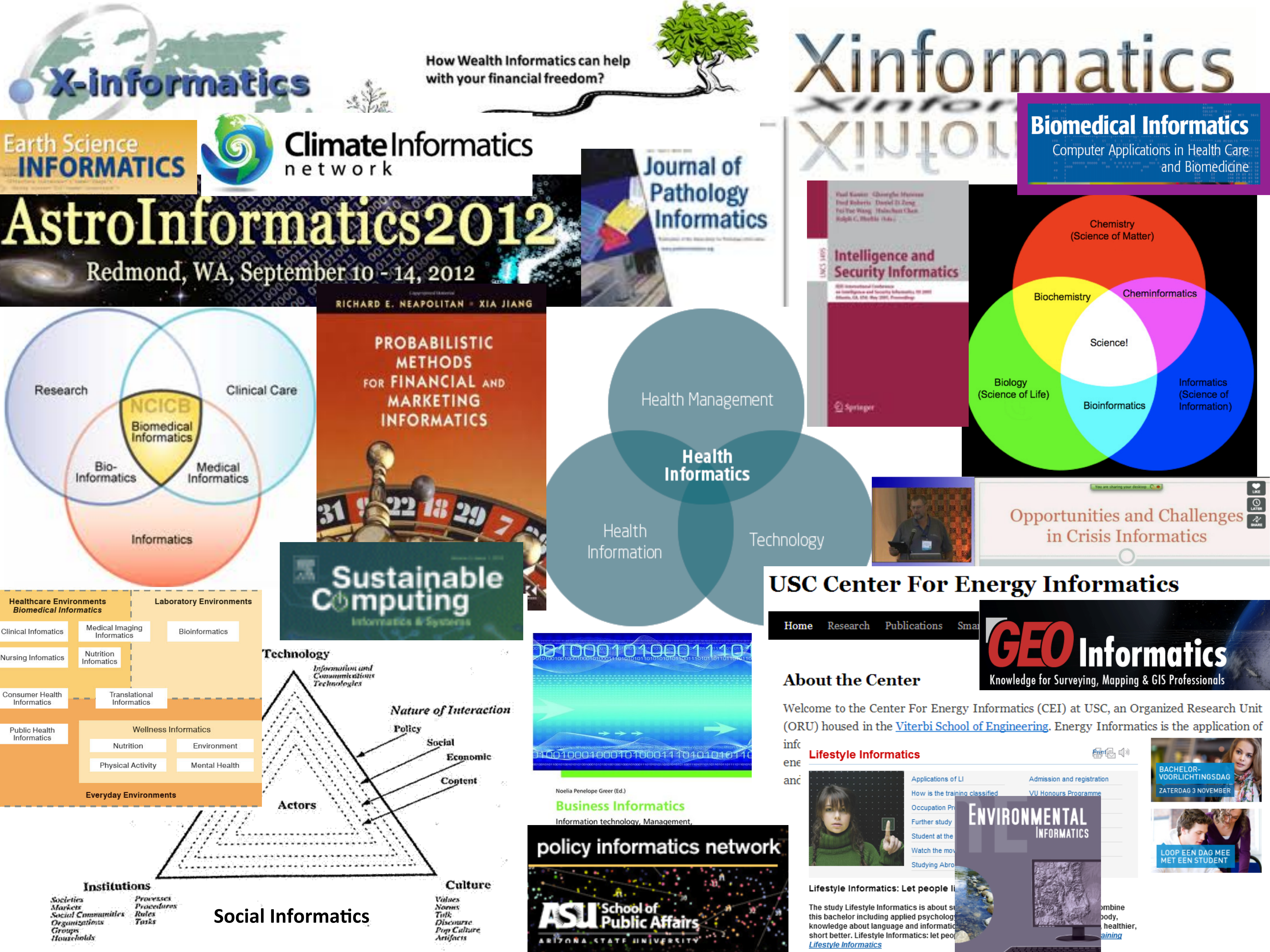
Use Clouds running Data Analytics Collaboratively processing Big Data to solve problems in X-Informatics (or e-X)

X = Astronomy, Biology, Biomedicine, Business, Chemistry, Climate, Crisis, Earth Science, Energy, Environment, Finance, Health, Intelligence, Lifestyle, Marketing, Medicine, Pathology, Policy, Radar, Security, Sensor, Social, Sustainability, Wealth and Wellness with more fields (physics) defined implicitly

Spans Industry and Science (research)

Education: Data Science see recent New York Times articles

http://datascience101.wordpress.com/2013/04/13/new-york-times-data-science-articles/


X-informatics
How Wealth Informatics can help with your financial freedom?
Xinformatics
Earth Science INFORMATICS
Climate Informatics network
Biomedical Informatics
Computer Applications in Health Care and Biomedicine
AstroInformatics2012
Redmond, WA, September 10 - 14, 2012
Journal of Pathology Informatics
Intelligence and Security Informatics
Springer
Chemistry (Science of Matter)
Biochemistry
Cheminformatics
Science!
Biology (Science of Life)
Bioinformatics
Informatics (Science of Information)
Research
Clinical Care
NCICB
Biomedical Informatics
Bio-Informatics
Medical Informatics
Informatics
RICHARD E. NEAPOLITAN · XIA JIANG
PROBABILISTIC METHODS FOR FINANCIAL AND MARKETING INFORMATICS
Health Management
Health Informatics
Health Information
Technology
Opportunities and Challenges in Crisis Informatics
Sustainable Computing
Informatics & Systems
Healthcare Environments
Biomedical Informatics
Laboratory Environments
Clinical Infomatics
Medical Imaging Informatics
Bioinformatics
Nursing Infomatics
Nutrition Infomatics
Consumer Health Informatics
Translational Informatics
Public Health Informatics
Wellness Informatics
Nutrition
Environment
Physical Activity
Mental Health
Everyday Environments
Technology
Information and Communications Technologies
Nature of Interaction
Policy
Social
Economic
Content
Actors
Institutions
Societies
Markets
Social Communities
Organizations
Groups
Households
Processes
Procedures
Rules
Tasks
Culture
Values
Norms
Talk
Discourse
Pop Culture
Artifacts
Social Informatics
Noelia Penelope Greer (Ed.)
Business Informatics
Information technology, Management,
policy informatics network
ASU School of Public Affairs
ARIZONA STATE UNIVERSITY
GEO Informatics
Knowledge for Surveying, Mapping & GIS Professionals
ENVIRONMENTAL INFORMATICS
BACHELOR-VOORLICHTINGSDAG
ZATERDAG 3 NOVEMBER
LOOP EEN DAG MEE MET EEN STUDENT

USC Center For Energy Informatics

Home Research Publications Smar

About the Center

Welcome to the Center For Energy Informatics (CEI) at USC, an Organized Research Unit (ORU) housed in the Viterbi School of Engineering. Energy Informatics is the application of info

ene

and

Lifestyle Informatics

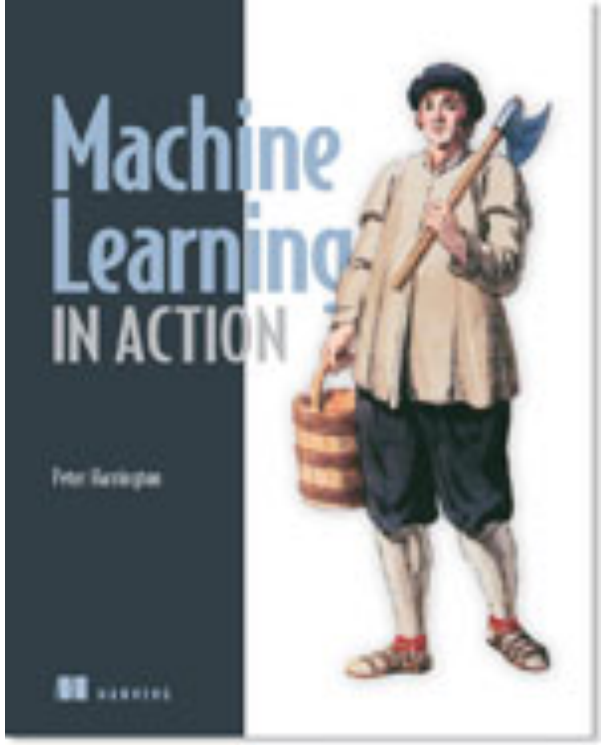
Applications of LI
How is the training classified
Occupation Pr
Further study
Student at the
Watch the mo
Studying Abro
Admission and registration
VU Honours Programme

Lifestyle Informatics: Let people li

The study Lifestyle Informatics is about s ombine this bachelor including applied psycholog body, knowledge about language and informatic healthier, short better. Lifestyle Informatics: let peo aining
Lifestyle Informatics

Python K-Nearest Neighbor Algorithms

Python Example



- **Machine Learning in Action**

  Peter Harrington

  April, 2012 | 384 pages
  ISBN: 9781617290183
- http://www.manning.com/pharrington/MLiA_SourceCode.zip
- Chapter 2
- We will use software from this book with minor enhancements to write out file in format for 3D plotting program

Files Used in Recommender Engine Python

- datingTestSet2.txt # A dataset of 1000 points
- ConvertToPviz.py # Python driver to convert dataset to a format readable by PlotViz program (3D point plotter)
- DatingRatingforPviz.txt # Result of converting 1000 point file
- DatingRating-OriginalLabels.pviz The previous file saved by Plotviz with same data but some formatting additions
- kNN.py # Code from book to implement k'th nearest neighbor and various utilities
- kNNDriver.py # Driver to invoke kNN.py and produce results discussed

Create Dataset of 2D and 3D Values and Rating (Labels)

- Two “artificial” examples
  - Start with a simple example: 4 2D vectors with Labels
  - Then move onto 1000 3D Labels

- #   Set up a dataset of 4 (2D vector) items -- each of which has one of two labels
- def createDataSet():
-     group = array([[1.0,1.1],[1.0,1.0],[0,0],[0,0.1]])
-     labels = ['A','A','B','B']
-     return group, labels

Python Example I: Four 2D Vectors

- # Set Directory
- %cd d:\\Python\RecommenderCh02
- import kNN

- FigDating = figure() # This figure will have 3 parts (subplots)

- # Set up 4 points and associate color to two groups
- group,labels = kNN.createDataSet()
- colormap1 = { 'A':'red', 'B':'blue'}
- ColoredGroupLabels = []
- for things in labels:
-     ColoredGroupLabels.append(colormap1[things])

- # Plot four points
- ax1 = FigDating.add_subplot(311, xlim=(-0.1,1.1), ylim=(-.05,1.15))
- ax1.scatter(group[:,0], group[:,1], s= 20, c= ColoredGroupLabels, marker = 'o')

Python Example I: Four 2D vectors Contd.

- # Here we classify and plot 3 points
- # Original points circles and test points crosses
- # Original points re two clusters – each of two points – red or blue
- # Color of test points indicates classification

- testvector = [.2, .2]
- answer = kNN.classify0(testvector,group, labels, 3)
- # type answer to see result or we will plot
- ax1.scatter(testvector[0], testvector[1], s= 20, c= colormap1[answer], marker = 'x')
- testvector = [.5, .5]
- answer = kNN.classify0(testvector,group, labels, 3)
- ax1.scatter(testvector[0], testvector[1], s= 20, c= colormap1[answer], marker = 'x')
- testvector = [.75, .75]
- answer = kNN.classify0(testvector, group, labels, 3)
- ax1.scatter(testvector[0], testvector[1], s= 20, c= colormap1[answer], marker = 'x')

Finding Nearest Neighbors

- testvector = [.75, .75]
- answer = kNN.classify0(testvector, group, labels, 3)
- ax1.scatter(testvector[0], testvector[1], s= 20, c= colormap1[answer], marker = 'x')

- # kNN.classify0 implements k nearest neighbor algorithm
- # It takes testvector and finds its Euclidean distance to all the points in group
- # Then its sorts distances and finds the k (=3 in this case) smallest distances
- # It returns label that appears most often in set of k points whose label is specified in labels
- # We will look at code in detail later on

Python Example II: 1000 3D vectors

- # Read in 1000 points in file and assign colors
- datingDataMat,datingLabels = kNN.file2matrix('datingTestSet2.txt')
- datingLabelArray = array(datingLabels)

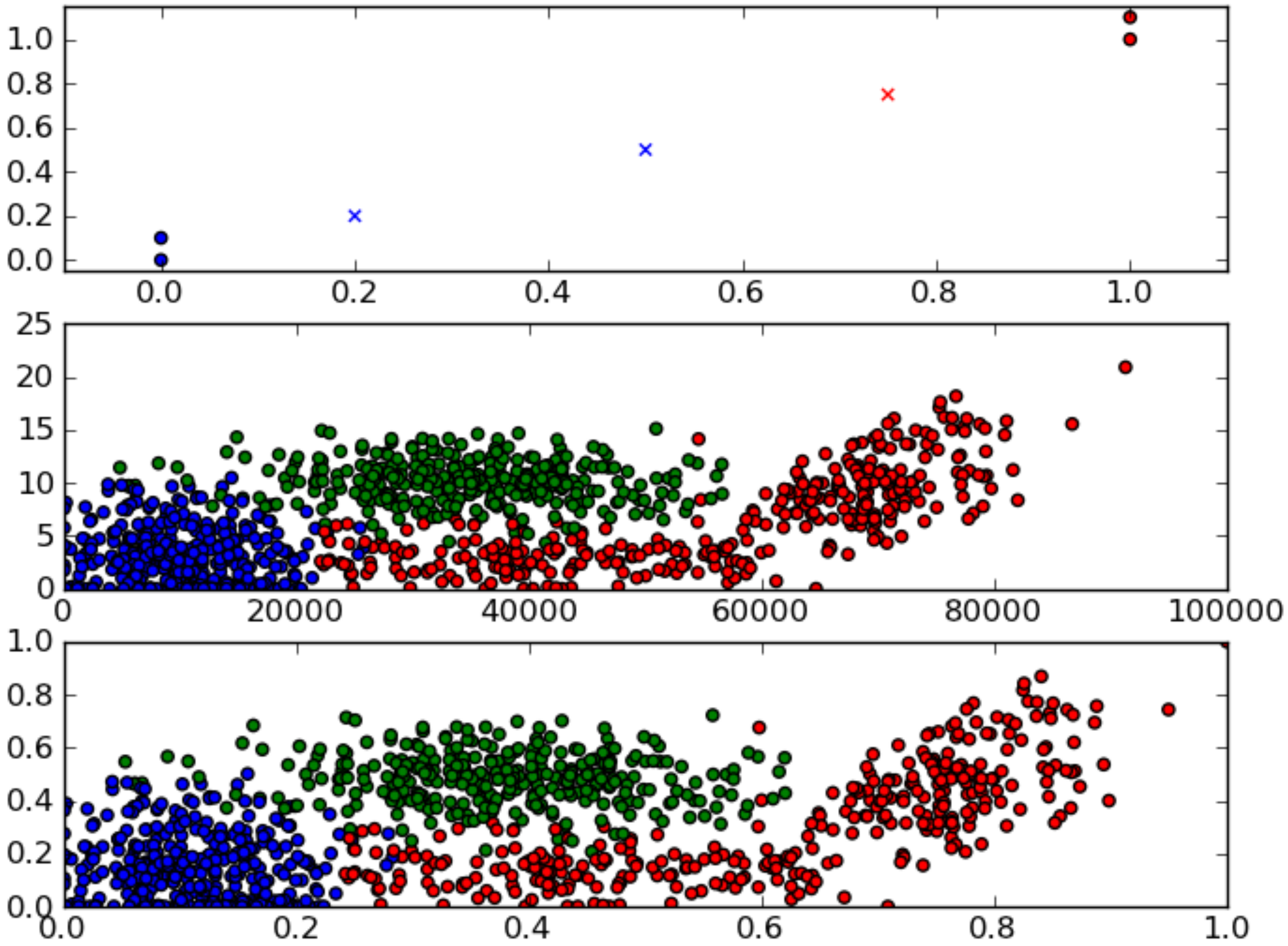
- colormap2 = { 1:'red', 2:'blue', 3:'green' }

- ColoredDatingLabel = []
- for things in datingLabelArray:
-     ColoredDatingLabel.append(colormap2[things])

Python Example II: 1000 3D vectors Contd.

- # Plot unnormalized results
- ax2 = FigDating.add_subplot(312, xlim=(0,100000), ylim=(0,25))
- ax2.scatter(datingDataMat[:,0], datingDataMat[:,1], s= 20, c= ColoredDatingLabel, marker = 'o')

- # Normalize and Plot normalized results
- normMat, ranges, minVals = kNN.autoNorm(datingDataMat)
- ax3 = FigDating.add_subplot(313, xlim=(0,1), ylim=(0,1))
- ax3.scatter(normMat[:,0], normMat[:,1], s = 20, c= ColoredDatingLabel, marker = 'o')
- # One typically normalizes in some way – one approach is to make mean 0 and standard deviation 1 in each coordinate direction
- # autoNorm makes minimum value 0 and maximum 1 in each coordinate
- show()   # Plot complete



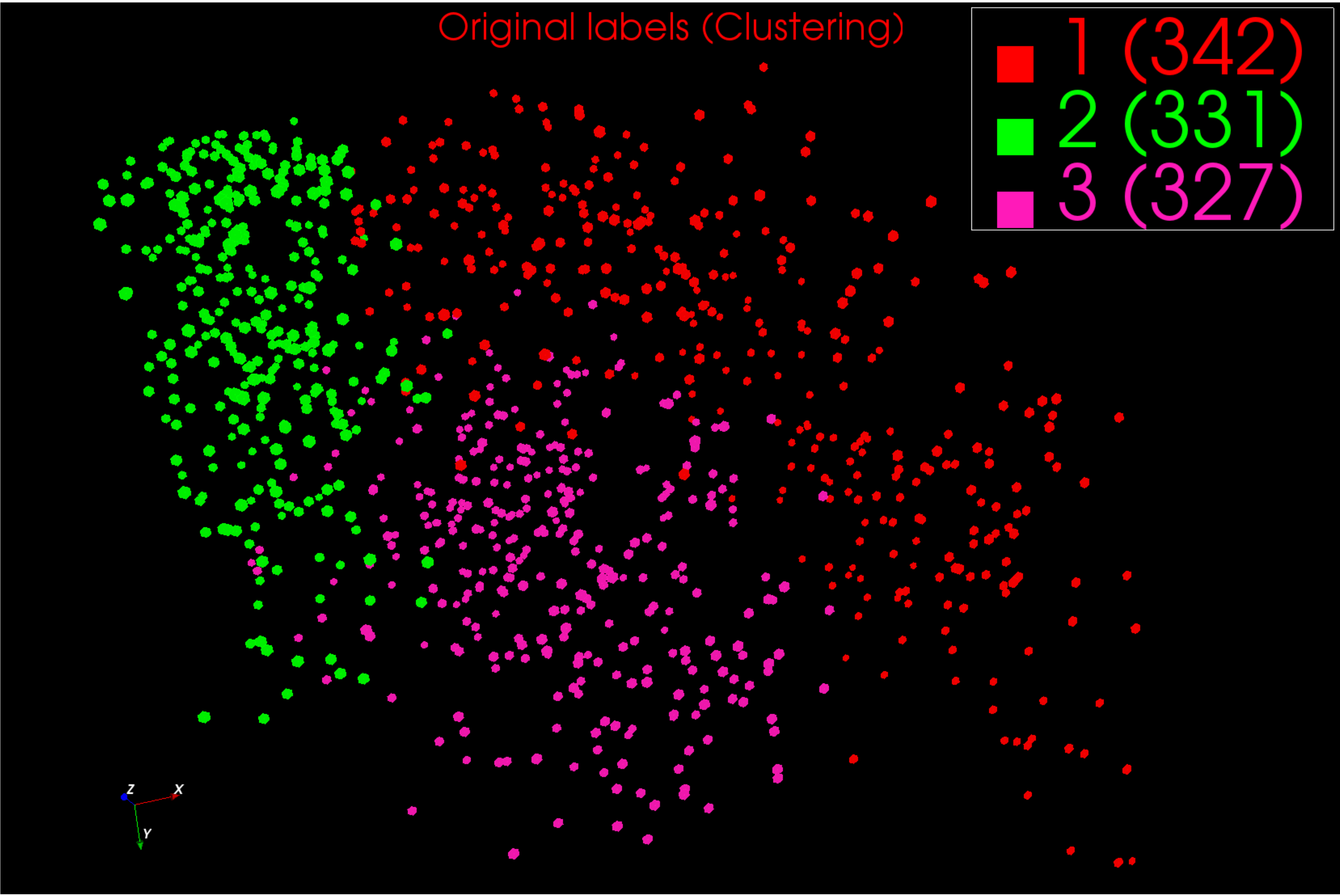
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
20000
40000
60000
80000
100000

Visualization

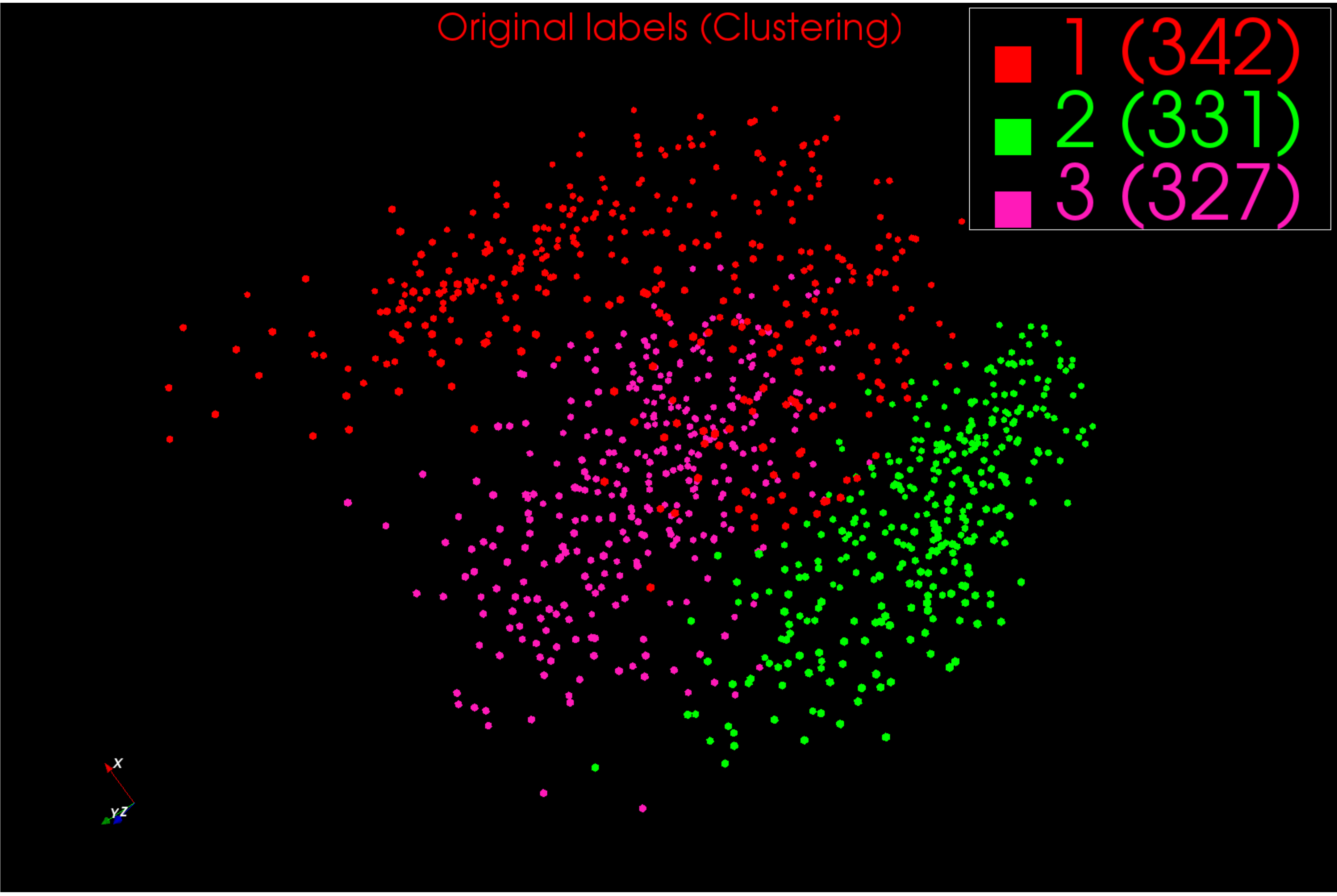
PlotViz

- Now we look at PlotViz as a 3D viewer available at http://salsahpc.indiana.edu/pviz3/
- We will demonstrate now with first some snapshots and then interactive rotation
- The cluster separation varies according to view taken

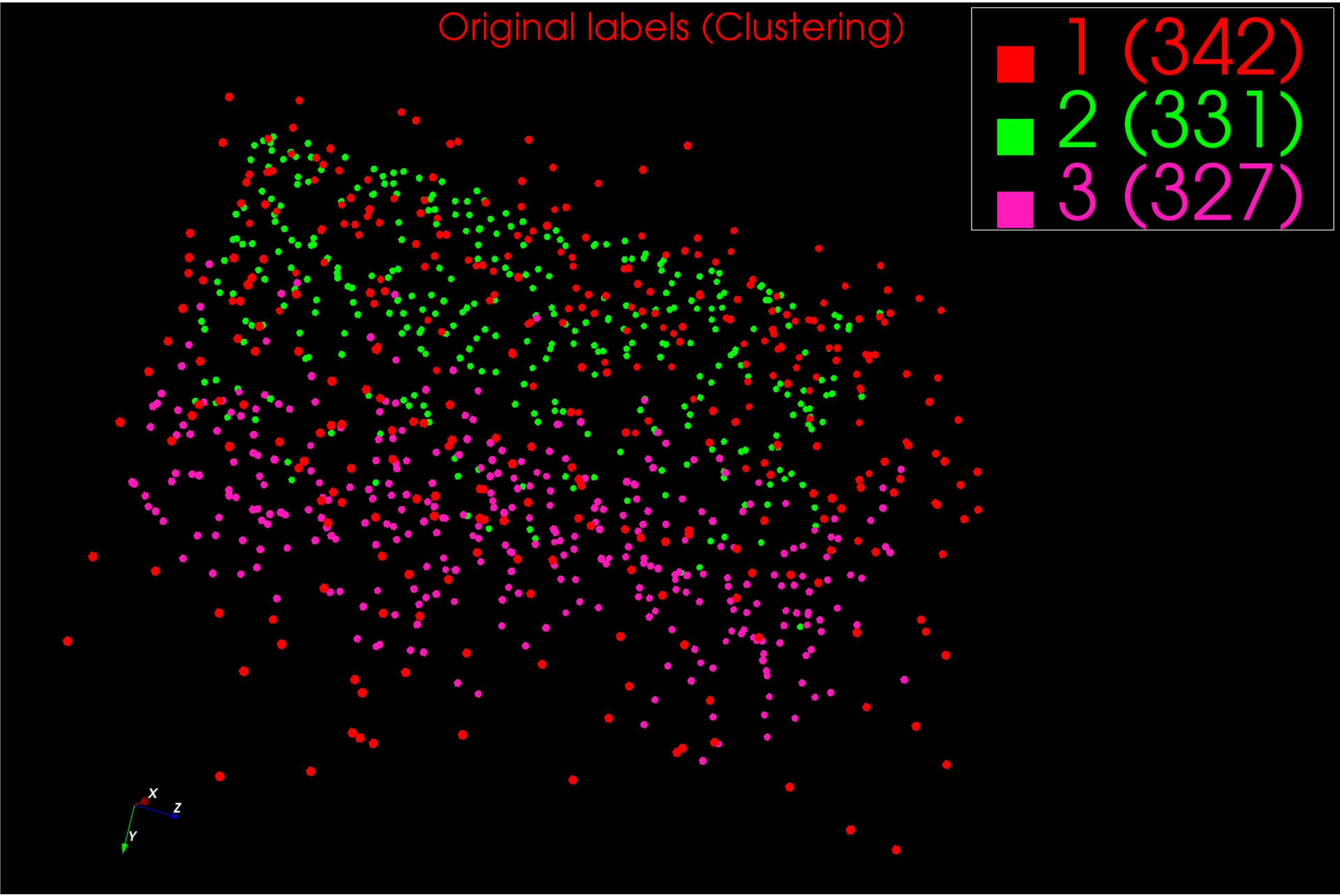
One of 3D Rotated 2D Views of Clusters



One of 3D Rotated 2D Views of Clusters



One of 3D Rotated 2D Views of Clusters



Testing K-Nearest Neighbor Algorithms

Test k NN Classification

- # final line in driver is
- NumberBad = kNN.datingClassTest(0.1)
  gives
- The total error rate is: 0.050000
- Number Bad 5.000000
- # The argument(0.1 here) of datingClassTest represents a fraction f indicates that first f.Total Number of Vectors is a test set to be classified based on final (1-f). Total Number of Vectors
- # Here the first 100 points are classified based on final 900
- # Function prints out number (here 5 or 5% of 100) of vectors whose k NN classification (using k=3 fixed in code) is different from input label
- # Note from picture the regions are not clearly separated and so points at edges of regions can be classified differently from this technique

Annotated kNN Classifier Function

- #	Use k Nearest Neighbors to classify inX accortding to existing dataSet with known ratings in labels
- def classify0(inX, dataSet, labels, k):
- dataSetSize = dataSet.shape[0]   # Number of entries in dataSet
- diffMat = tile(inX, (dataSetSize,1)) – dataSet # Array of same shape as dataSet holding inX-dataSet entry in each position
- sqDiffMat = diffMat**2 # Square entries in diffMat
- sqDistances = sqDiffMat.sum(axis=1)   # Sum over vector components
- distances = sqDistances**0.5   # Traditional Euclidean distance for each dataSet entry
- sortedDistIndicies = distances.argsort()   # argsort returns indices that sort distances
- classCount={} # An empty dictionary key:value pairs key = labels value = number times this label in set of k
- for i in range(k):   # Run over k nearest neighbors
- voteIlabel = labels[sortedDistIndicies[i]] # Label of this neighbor
- classCount[voteIlabel] = classCount.get(voteIlabel,0) + 1   # .get gets current count for voteIlabel and returns zero if first time voteIlabel seen (default = 0)
- sortedClassCount = sorted(classCount.iteritems(), key=operator.itemgetter(1), reverse=True)   # Sort classCount (highest to lowest) by voteIlabel count
- # Note classCount entries are Label:# votes and we sort on # votes but return Label
- return sortedClassCount[0][0]   # The label that occurs most often in k nearest neighbors
- # As in many such languages, compact but rather opaque and you need to know .get .iteritems and .itemgetter