Lecture 12: Supervised Learning and Classification

(download slides and .py files from Stellar to follow along)

Fredo Durand, John Guttag, Ana Bell, Eric Grimson

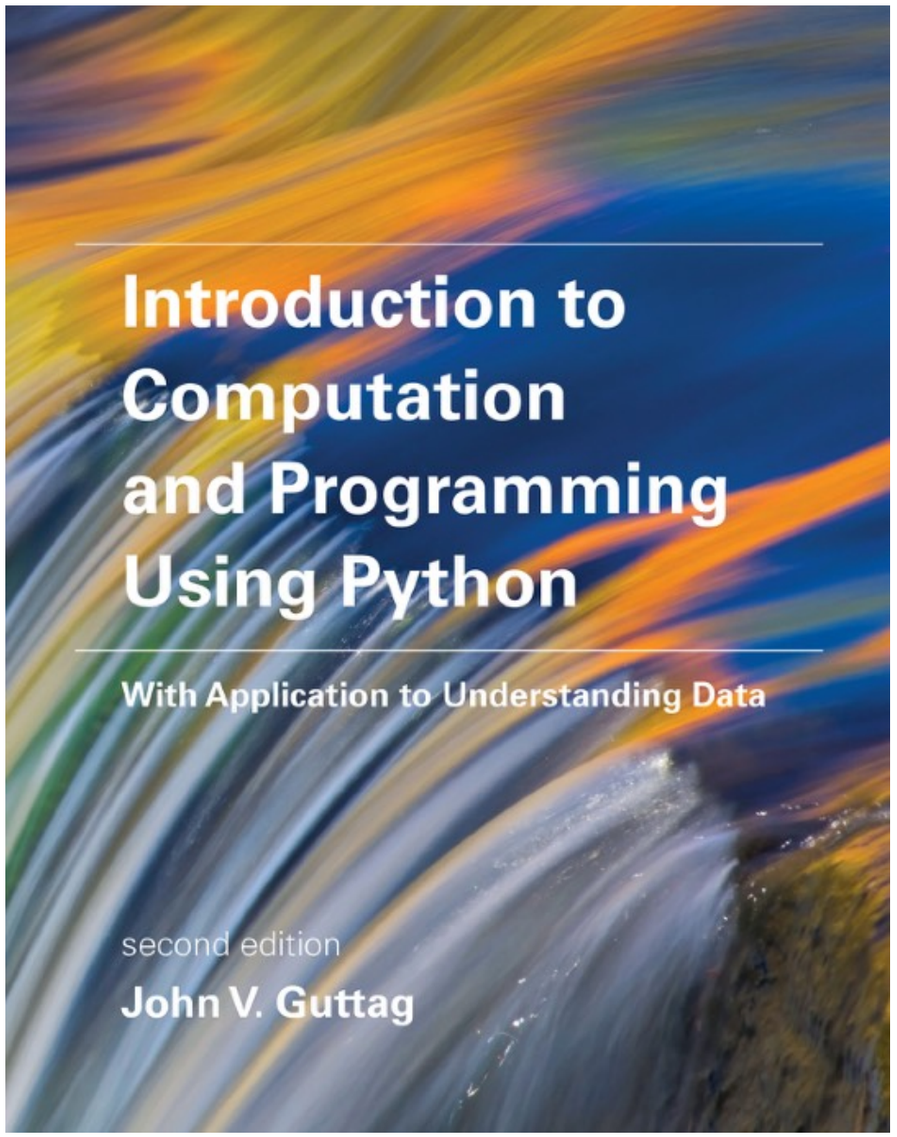
MIT Department of Electrical Engineering and Computer Science

Microquiz 4

- After this lecture

Relevant Reading

- Chapter 24

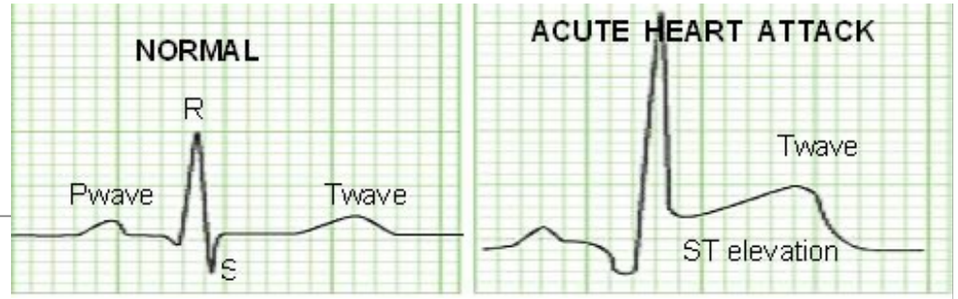


https://mitpress.mit.edu/sites/default/files/Guttag_errata_revised_083117.pdf

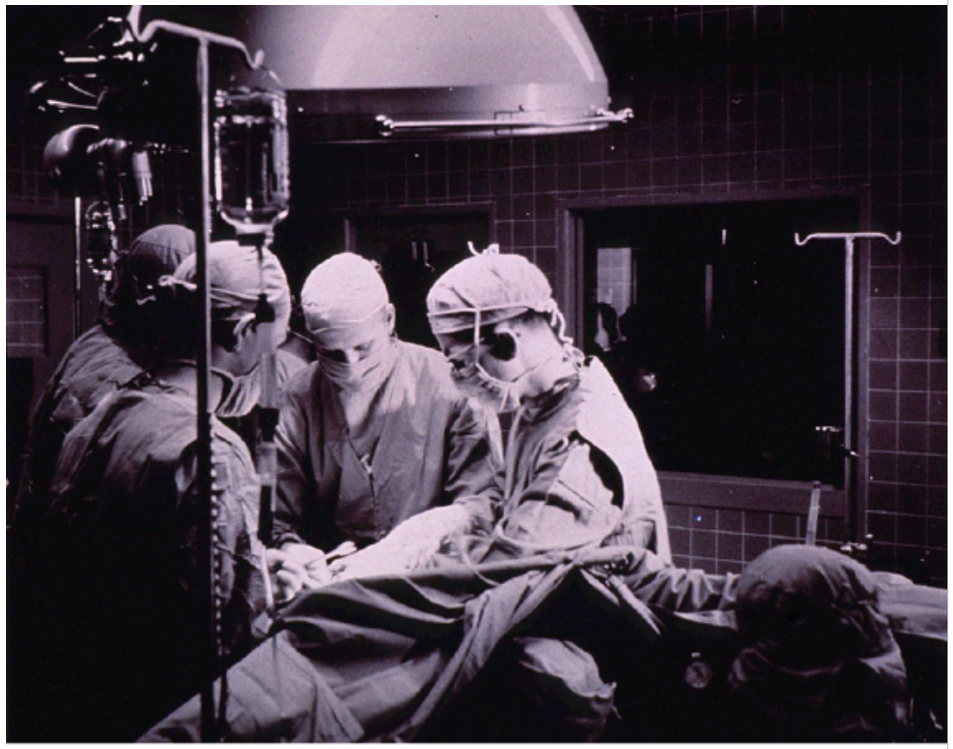
Last time

- Unsupervised learning
  - Clustering
  - Usually an exploratory tool
  - Sometimes used for classification when supervision (labels) are available
- Metrics

Recall Example from Monday



- Risk of death from heart attack
- Hypothesis is that certain measurable factors may be predictive of mortality
- Approach: Cluster based on attribute values; examine purity of clusters relative to outcomes

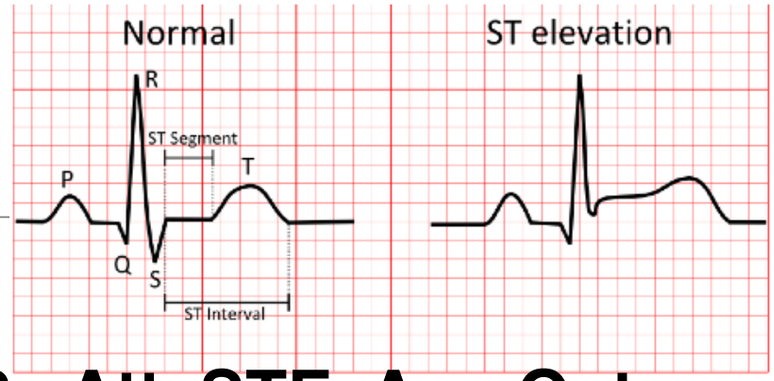


Why cluster?

- Are there sub-populations that might emerge from the data?
- Could these reflect different variants of disease?

An Example

- Data
  - Large number of patients
    - **Heart rate in beats per minute**
    - **Number of past heart attacks**
    - **ST elevation (binary)**
    - **Age**
- Outcome (death) based on features
  - Probabilistic, not deterministic
  - E.g., older people with multiple heart attacks at higher risk
- Approach
  - Cluster
  - Examine purity of clusters relative to outcomes



	HR	Att	STE	Age	Outcome
P000:	89.	1.	0.	66.	1
P001:	59.	0.	0.	72.	0
P002:	73.	0.	0.	73.	0
P003:	56.	1.	0.	65.	0
P004:	75.	1.	1.	68.	1
P005:	68.	1.	0.	56.	0
P006:	73.	1.	0.	75.	1
P007:	72.	0.	0.	65.	0
P008:	73.	1.	0.	64.	1
P009:	73.	0.	0.	58.	0
P010:	100.	0.	0.	75.	0
P011:	79.	0.	0.	31.	0
P012:	81.	0.	0.	58.	0
P013:	89.	1.	0.	50.	1
P014:	81.	0.	0.	70.	0

1 means mortality

Run It

```
      Test k-means (k = 2)
Cluster of size 118, frac. pos. = 0.331,
num. pos. = 39
Cluster of size 132, frac. pos. = 0.333,
num. pos. = 44
```

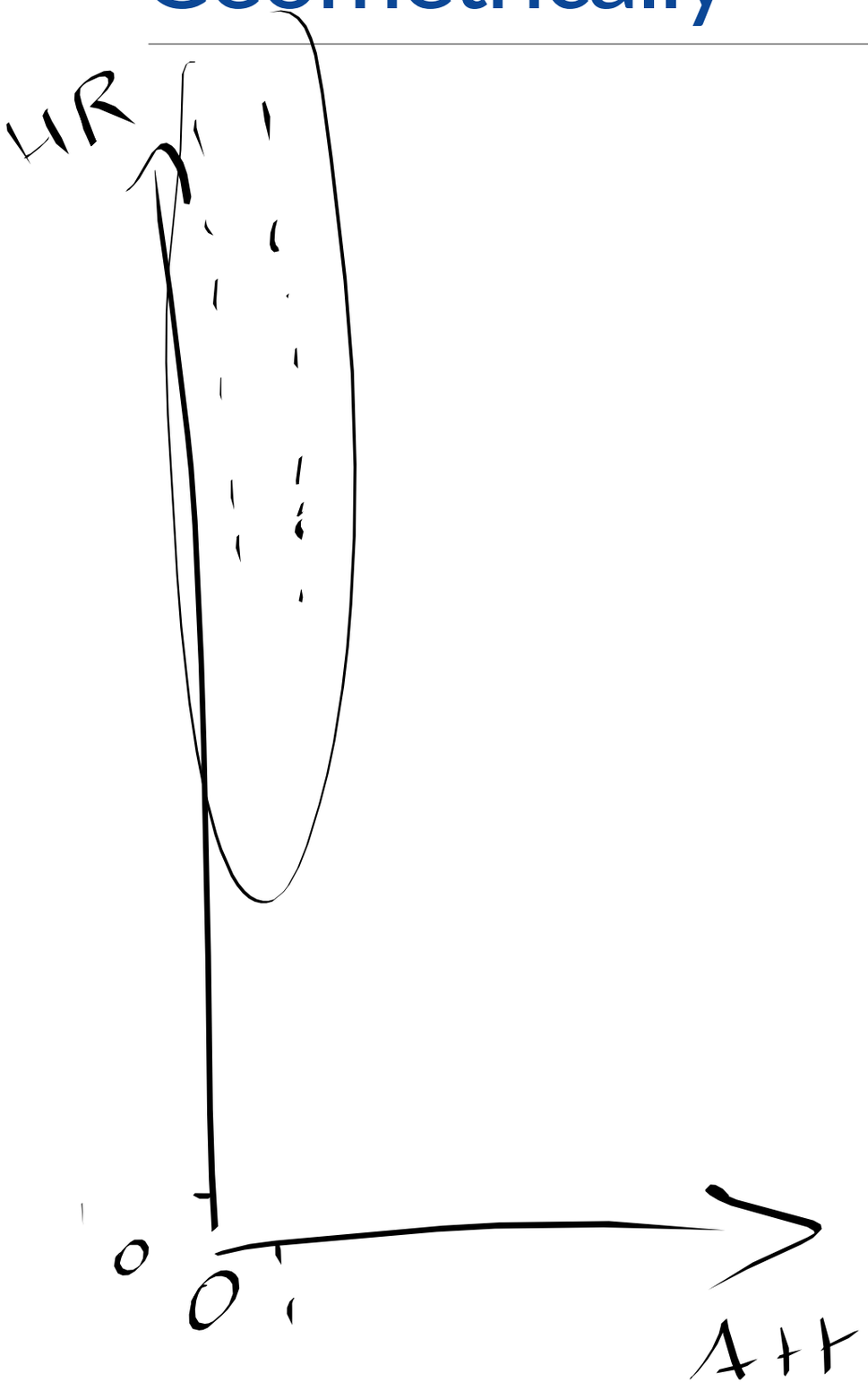
What do you think? Good result?

What's the Problem?

- Features at vastly different scales
- ST elevation binary
- Number of heart attacks low single digits
- Heart rate double or triple digits

	HR	Att	STE	Age	Outcome
P000	89.	1.	0.	66.	1
P001	59.	0.	0.	72.	0
P002	73.	0.	0.	73.	0
P003	56.	1.	0.	65.	0
P004	75.	1.	1.	68.	1
P005	68.	1.	0.	56.	0
P006	73.	1.	0.	75.	1
P007	72.	0.	0.	65.	0
P008	73.	1.	0.	64.	1
P009	73.	0.	0.	58.	0
P010	100.	0.	0.	75.	0
P011	79.	0.	0.	31.	0
P012	81.	0.	0.	58.	0
P013	89.	1.	0.	50.	1
P014	81.	0.	0.	70.	0

Geometrically



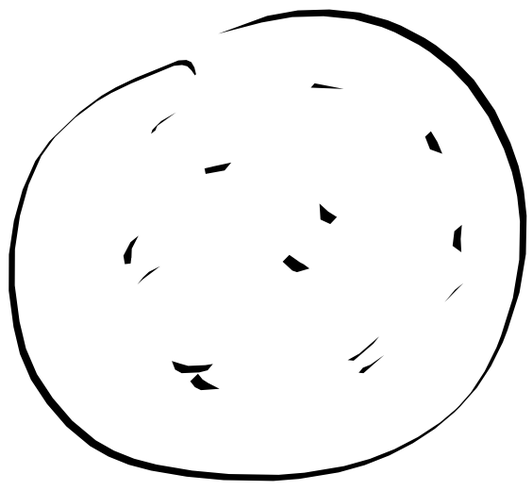
	HR	Att	STE	Age	Outcome
P000:	89.	1.	0.	66.	:1
P001:	59.	0.	0.	72.	:0
P002:	73.	0.	0.	73.	:0
P003:	56.	1.	0.	65.	:0
P004:	75.	1.	1.	68.	:1
P005:	68.	1.	0.	56.	:0
P006:	73.	1.	0.	75.	:1
P007:	72.	0.	0.	65.	:0
P008:	73.	1.	0.	64.	:1
P009:	73.	0.	0.	58.	:0
P010:	100.	0.	0.	75.	:0
P011:	79.	0.	0.	31.	:0
P012:	81.	0.	0.	58.	:0
P013:	89.	1.	0.	50.	:1
P014:	81.	0.	0.	70.	:0

Idea: scaling

scale HR

→ 0 - 1

scale age down



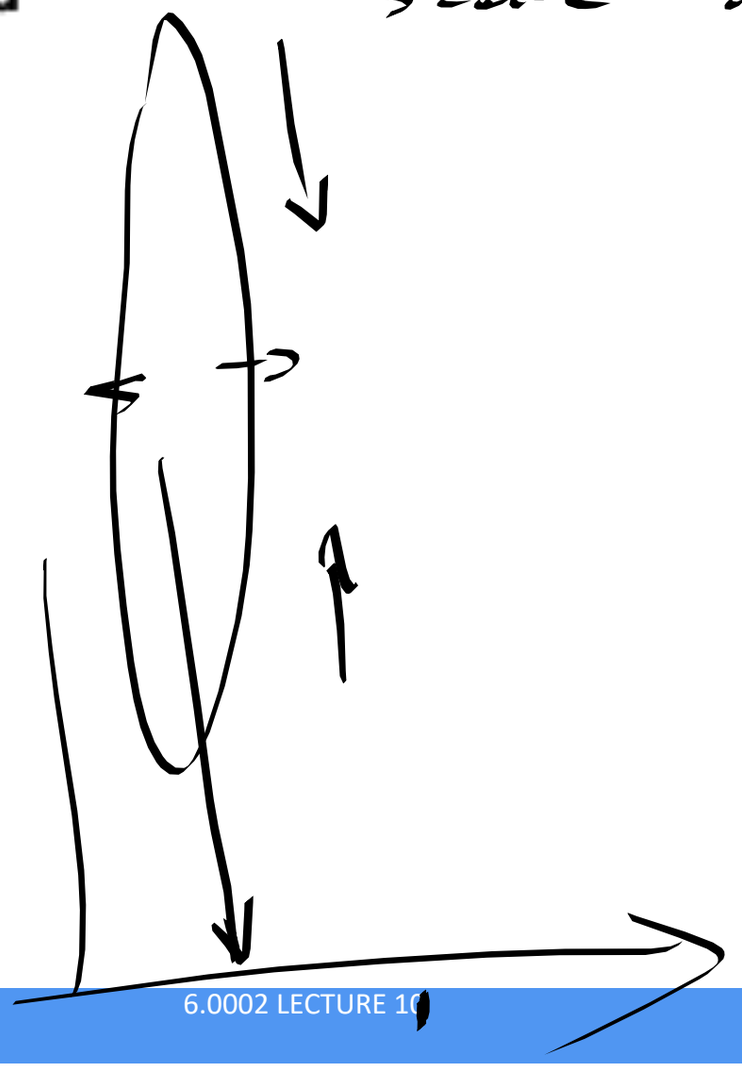
	HR	Att	STE	Age	Outcome
P000:	89.	1.	0.	66.	:1
P001:	59.	0.	0.	72.	:0
P002:	73.	0.	0.	73.	:0
P003:	56.	1.	0.	65.	:0
P004:	75.	1.	1.	68.	:1
P005:	68.	1.	0.	56.	:0
P006:	73.	1.	0.	75.	:1
P007:	72.	0.	0.	65.	:0
P008:	73.	1.	0.	64.	:1
P009:	73.	0.	0.	58.	:0
P010:	100.	0.	0.	75.	:0
P011:	79.	0.	0.	31.	:0
P012:	81.	0.	0.	58.	:0
P013:	89.	1.	0.	50.	:1
P014:	81.	0.	0.	70.	:0

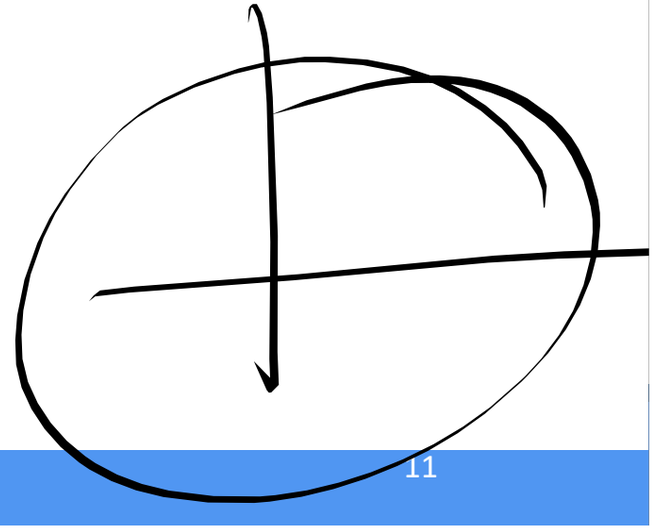
Scaling by mean and standard deviation (z-scaling)

```
def scaleData(vals):
    #assumes vals an array of numbers
    mean = sum(vals)/len(vals)
    sd = numpy.std(vals)
    vals = vals - mean
    return vals/sd
```

subtract mean
scale by sd

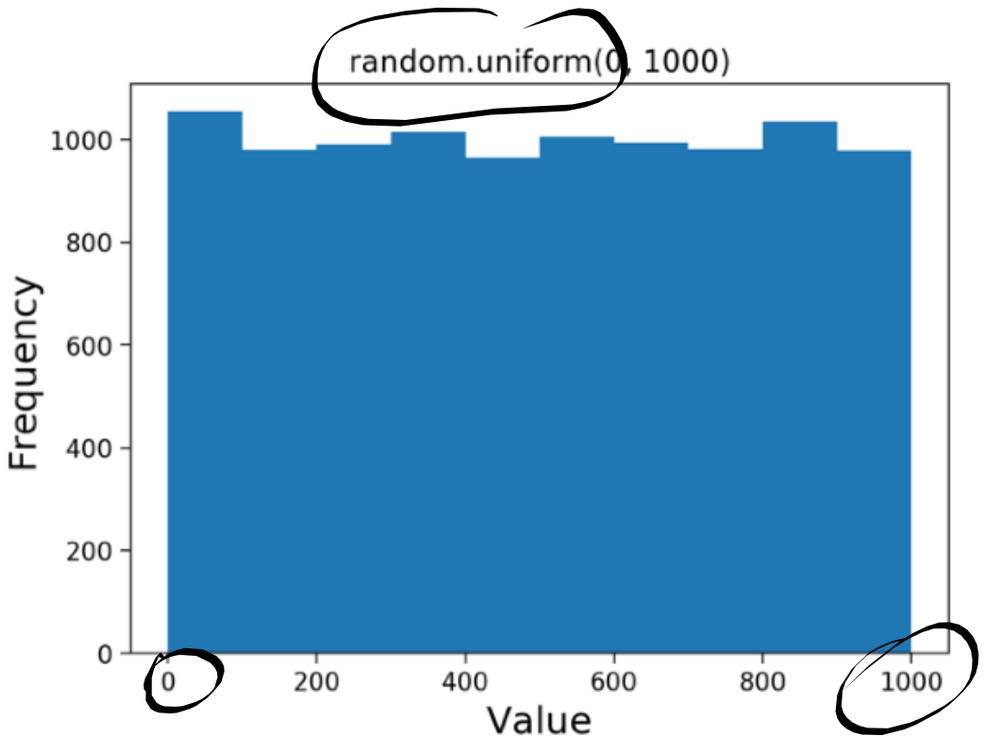
Poll

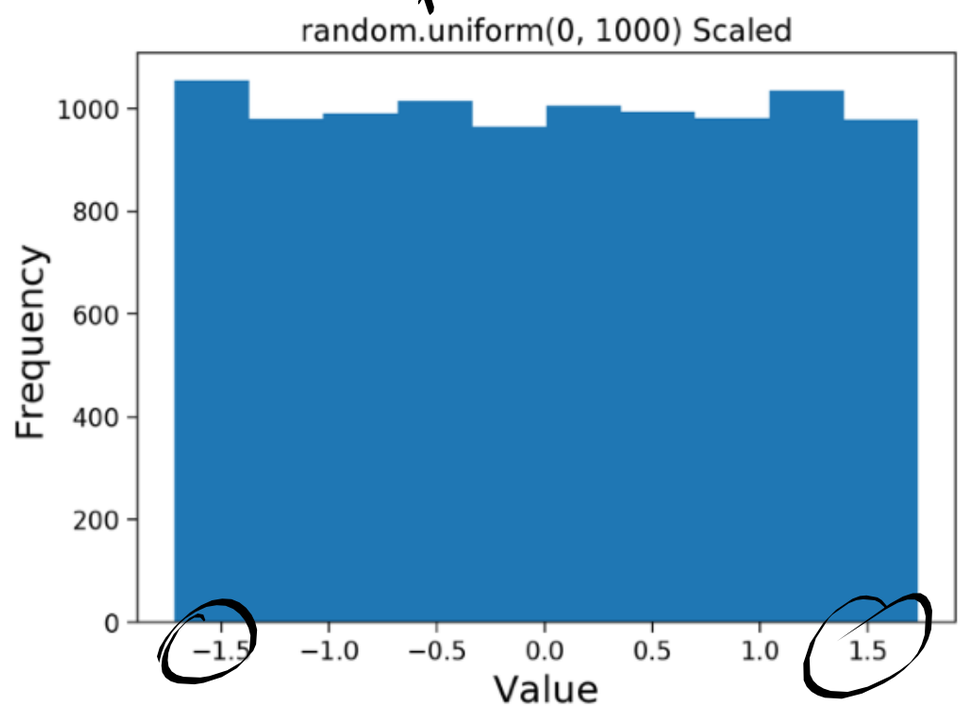


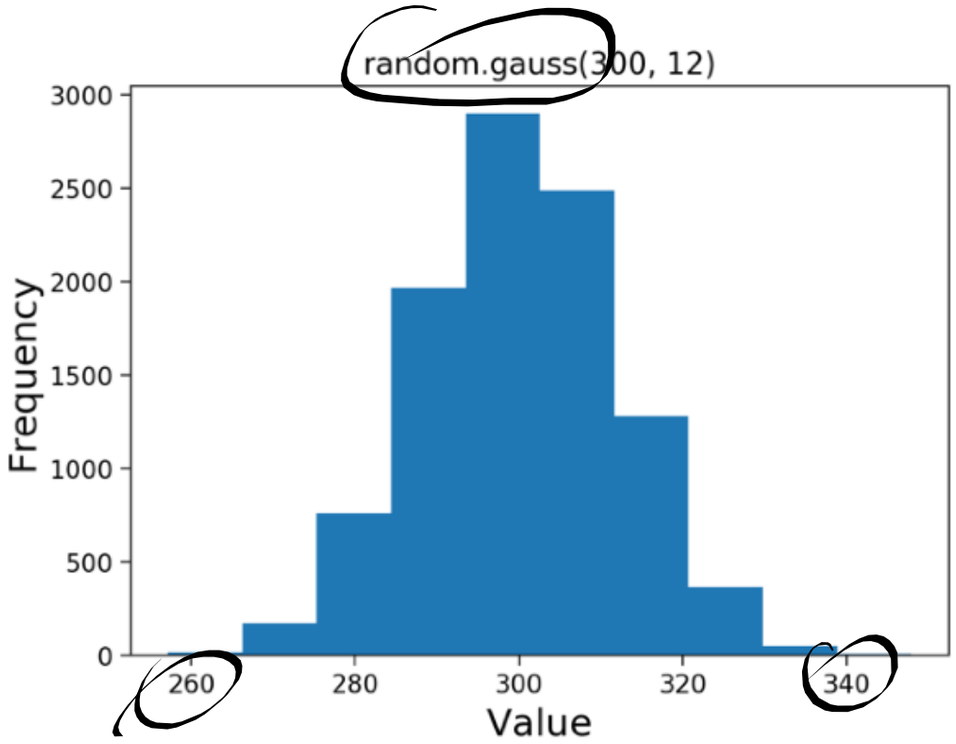


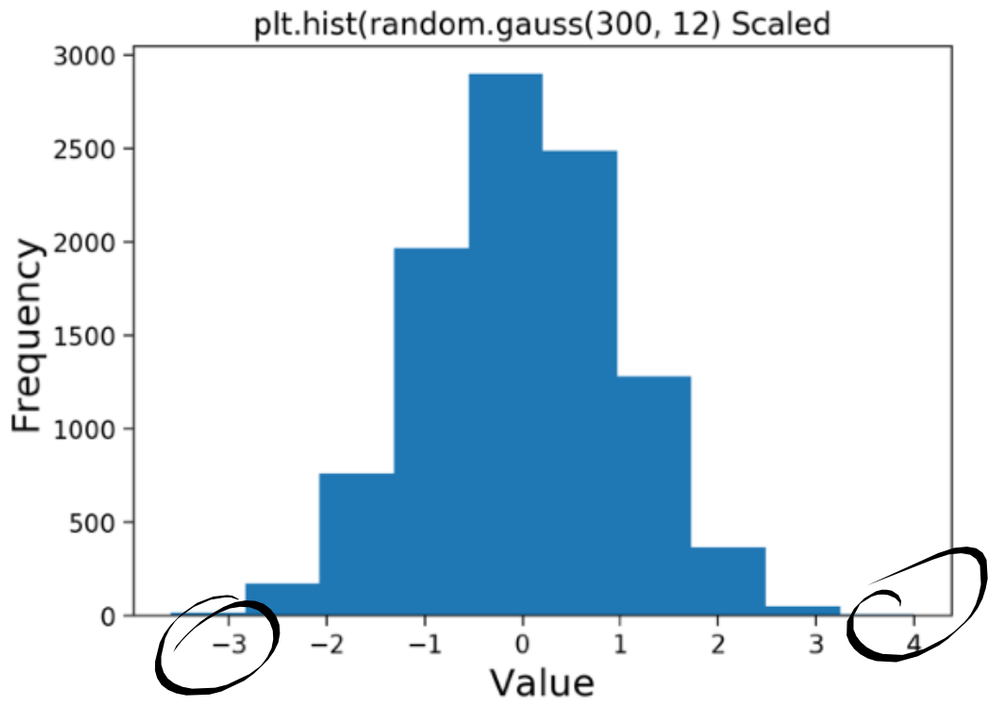
Scaling Changes Values But Not Shapes

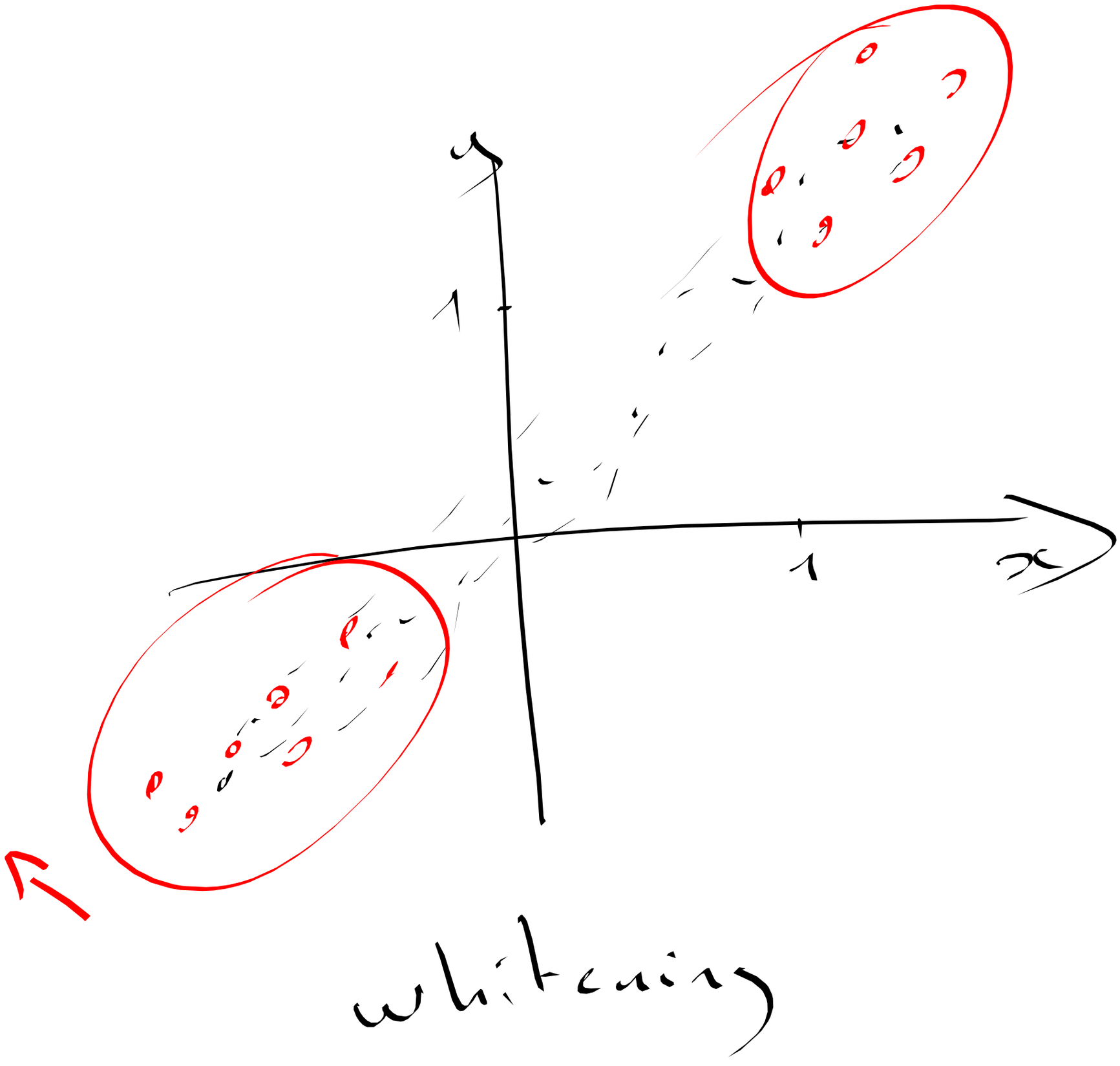
after scaling










1
1
whitening

Result of Clustering with Scaling

Without scaling:

```
Test k-means (k = 2)
Cluster of size 118, frac. pos. = 0.331,
num. pos. = 39
Cluster of size 132, frac. pos. = 0.333,
num. pos. = 44
```

With scaling:

Try patients = getData(True) #scale features

Test k-means (k = 2)

Cluster of size 224, frac. pos. = 0.290, num. pos. = 65

Cluster of size 26, frac. pos. = 0.692, num. pos. = 18

Result of Clustering with Scaling

Without scaling:

```
     Test k-means (k = 2)
Cluster of size 118, frac. pos. = 0.331,
num. pos. = 39
Cluster of size 132, frac. pos. = 0.333,
num. pos. = 44
```

With scaling:

Try patients = getData(True) #scale features

Test k-means (k = 2)

Cluster of size 224, frac. pos. = 0.290, num. pos. = 65

Cluster of size 26, frac. pos. = 0.692, num. pos. = 18

Happy with sensitivity? Or at least happier?

Where are most of the deaths?

A Hypothesis

- Different subgroups of positive patients have different characteristics
- How might we test this?
- Try some other values of k

```
patients = getData(True)
for k in (2,4,6):
    print('\n      Test k-means (k = ' + str(k) + ')')
    posFracs = testClustering(patients, k, 2)
```

Testing Multiple Values of k

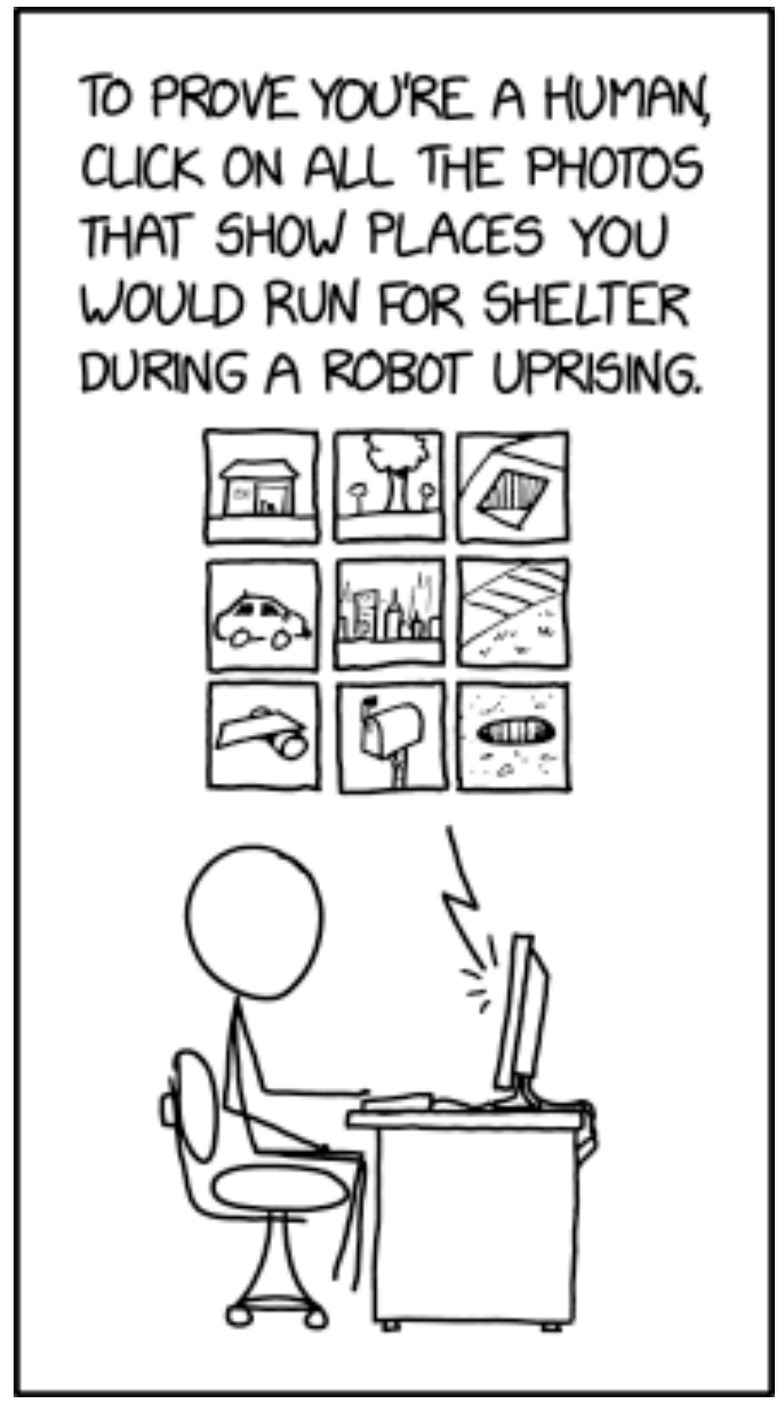
Test k-means (k = 2)
Cluster of size 224, frac. pos. = 0.290, num. pos. = 65
Cluster of size 26, frac. pos. = 0.692, num. pos. = 18

Test k-means (k = 4)
Cluster of size 26, frac. pos. = 0.692, num. pos. = 18
Cluster of size 86, frac. pos. = 0.081, num. pos. = 7
Cluster of size 76, frac. pos. = 0.711, num. pos. = 54
Cluster of size 62, frac. pos. = 0.065, num. pos. = 4

Test k-means (k = 6)
Cluster of size 49, frac. pos. = 0.020, num. pos. = 1
Cluster of size 26, frac. pos. = 0.692, num. pos. = 18
Cluster of size 45, frac. pos. = 0.089, num. pos. = 4
Cluster of size 54, frac. pos. = 0.093, num. pos. = 5
Cluster of size 36, frac. pos. = 0.778, num. pos. = 28
Cluster of size 40, frac. pos. = 0.675, num. pos. = 27

Questions?

- Scaling is always a good idea
- So-called Hyper-parameters (k) matter



Today: Supervised Learning

- Given training data
  - Observation
  - Corresponding outcome (label)
- Predict outcome given new unseen observations

- e.g.
  - Medical data: Heart rate, age, etc. => mortality
  - Photo => class of object (car vs. road vs. tree)
  - Sound recording => text
- The availability of training labels is what makes it supervised (you are given example input/outputs)

Today: Supervised Learning

- Regression
  - Predict a real number associated with a feature vector

Already spent time on this, right?

- *Classification*
  - Predict a discrete value (label) associated with a feature vector

Today's Lecture

- Metrics for evaluating a classification model
- Two common classification methods
  - K-nearest neighbors
  - Logistic regression
  - Representing two extremes of Machine learning continuum

Evaluation: Accuracy

- Accuracy is the fraction predicted correctly
- Certainly important, but is it enough?
- Consider a disease that occurs in 1% of population
  - Predicting disease-free has an accuracy of ? POLL

Evaluation: Accuracy

- Accuracy is the fraction predicted correctly
- Certainly important, but is it enough?
- Consider a disease that occurs in 1% of population
  - Predicting disease-free has an accuracy of 0.99
  - Still probably not a very useful test

A Fuller Picture of Performance

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	True positive	false negative
	condition negative	false positive	true negative

A Fuller Picture of Performance: confusion matrix

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Confusion Matrix

Sensitivity and Specificity

$$sensitivity = \frac{true\ positive}{true\ positive + false\ negative}$$

actually positive

$$specificity = \frac{true\ negative}{true\ negative + false\ positive}$$

all actually negative

Sensitivity and Specificity

$$sensitivity = \frac{true\ positive}{true\ positive + false\ negative}$$

=> total number of positives

Ability to know what
is present
Aka recall

$$specificity = \frac{true\ negative}{true\ negative + false\ positive}$$

=> total number of negatives

Ability to know what is
not present
Aka precision

Sensitivity and Specificity

$$sensitivity = \frac{true\ positive}{true\ positive + false\ negative} = \frac{14}{22}$$

Ability to know what is present

$$specificity = \frac{true\ negative}{true\ negative + false\ positive} = \frac{76}{78}$$

Ability to know what is not present

Rapid Strep Test

100 kids, 22 with strep throat
Diagnosed with strep: 14 with, 2 without
Diagnosed strep-free: 8 with, 76 without

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive 22	True Positive (TP) 14	False Negative (FN) (type II error) 8
	condition negative 78	False Positive (FP) (Type I error) 2	True Negative (TN) 76

More metrics: Predictive Value given prediction

If the test is *positive*, how likely is it that he really ***has*** the disease? How *worried* should he be?

If the test is *negative*, how likely is it that he really ***does NOT*** have it? How *reassured* should he be?

http://sphweb.bumc.bu.edu/otlt/MPH-Modules/QuantCore/PH717_Screening/PH717_Screening5.html

$$positive\ predictive\ value = \frac{true\ positive}{true\ positive + false\ positive}$$

Note the different denominator

all predicted positive

$$negative\ predictive\ value = \frac{true\ negative}{true\ negative + false\ negative}$$

Note the different denominator

all predicted negative

More metrics: Predictive Value given prediction

If the test is *positive*, how likely is it that he really ***has*** the disease? How *worried* should he be?

If the test is *negative*, how likely is it that he really ***does NOT*** have it? How *reassured* should he be?

http://sphweb.bumc.bu.edu/otlt/MPH-Modules/QuantCore/PH717_Screening/PH717_Screening5.html

$$positive\ predictive\ value = \frac{true\ positive}{true\ positive + false\ positive}$$

$$negative\ predictive\ value = \frac{true\ negative}{true\ negative + false\ negative}$$

Diagnosed with strep: 14 with, 2 without

Diagnosed strep-free: 8 with, 76 without

Predictive Value

If the test is *positive*, how likely is it that he really ***has*** the disease? How *worried* should he be?

If the test is *negative*, how likely is it that he really ***does NOT*** have it? How *reassured* should he be?

http://sphweb.bumc.bu.edu/otlt/MPH-Modules/QuantCore/PH717_Screening/PH717_Screening5.html

$$positive\ predictive\ value = \frac{true\ positive}{true\ positive + false\ positive}$$

$$negative\ predictive\ value = \frac{true\ negative}{true\ negative + false\ negative}$$

Diagnosed with strep: 14 with, 2 without

Diagnosed strep-free: 8 with, 76 without

Positive predictive value: 14 / (14+2) = 0.87

Positive predictive value: 76 / (76+8) = 0.90

Recap

$$sensitivity = \frac{true\ positive}{true\ positive + false\ negative} = \frac{14}{14+8} \approx .64$$

$$specificity = \frac{true\ negative}{true\ negative + false\ positive} = \frac{76}{76+2} \approx .97$$

$$positive\ predictive\ value = \frac{true\ positive}{true\ positive + false\ positive} = 0.87$$

$$negative\ predictive\ value = \frac{true\ negative}{true\ negative + false\ negative} = 0.90$$

Diagnosed with strep: 14 with, 2 without

Diagnosed strep-free: 8 with, 76 without

Complete picture

- https://en.wikipedia.org/wiki/Sensitivity_and_specificity

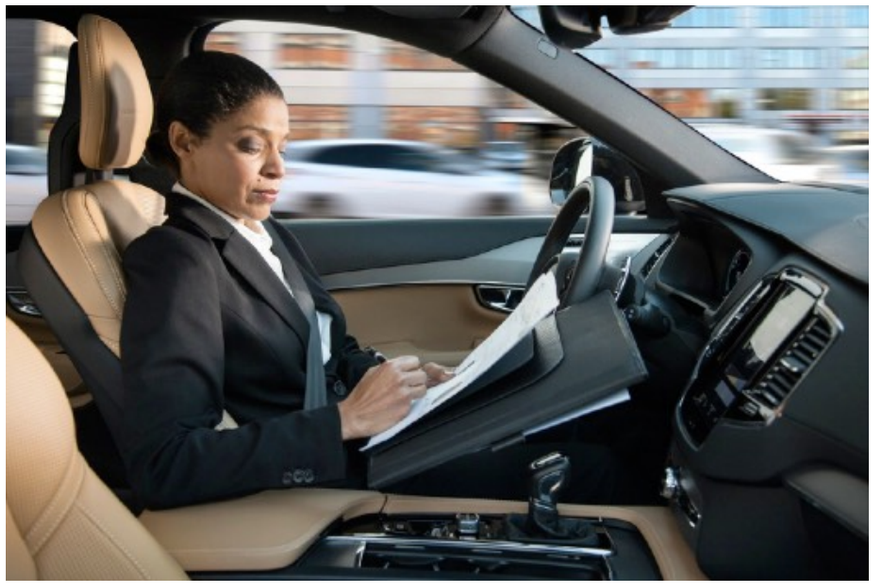
		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

Which Metric is Most Important?

Poll

Which Metric is Most Important?

- It depends



		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Base rate

- Pay particular attention when the two classes are unbalanced
  - e.g. disease that is present in 1% of population
  - More on Monday

Questions?


WE ADDED A NEW PERFORMANCE TEST, BUT LEARNED THAT THE TEST ITSELF IS FLAWED.
NOW OUR PRODUCT FAILS OUR OWN TESTS AND OUR CUSTOMERS ARE ASKING TO SEE THE TEST RESULTS.
DO I HAVE PERMISSION TO FAKE THE TEST DATA?
I DIDN'T EVEN KNOW DATA CAN BE REAL.
DilbertCartoonist@gmail.com
Dilbert.com
© 2010 Scott Adams, Inc./Dist. by UFS, Inc.
8-11-10

Today's Lecture

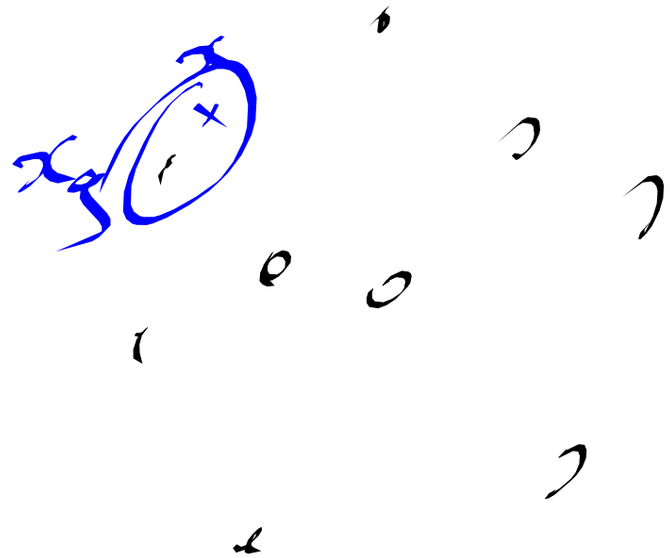
- Metrics for evaluating a classification model
- Two common classification methods
  - K-nearest neighbors
  - Logistic regression
  - Representing two extremes of Machine learning continuum

Back to supervised learning

- Given training data:
  - Observations xi
  - Labels or outcome yi
- Predict y given unseen x

Back to supervised learning

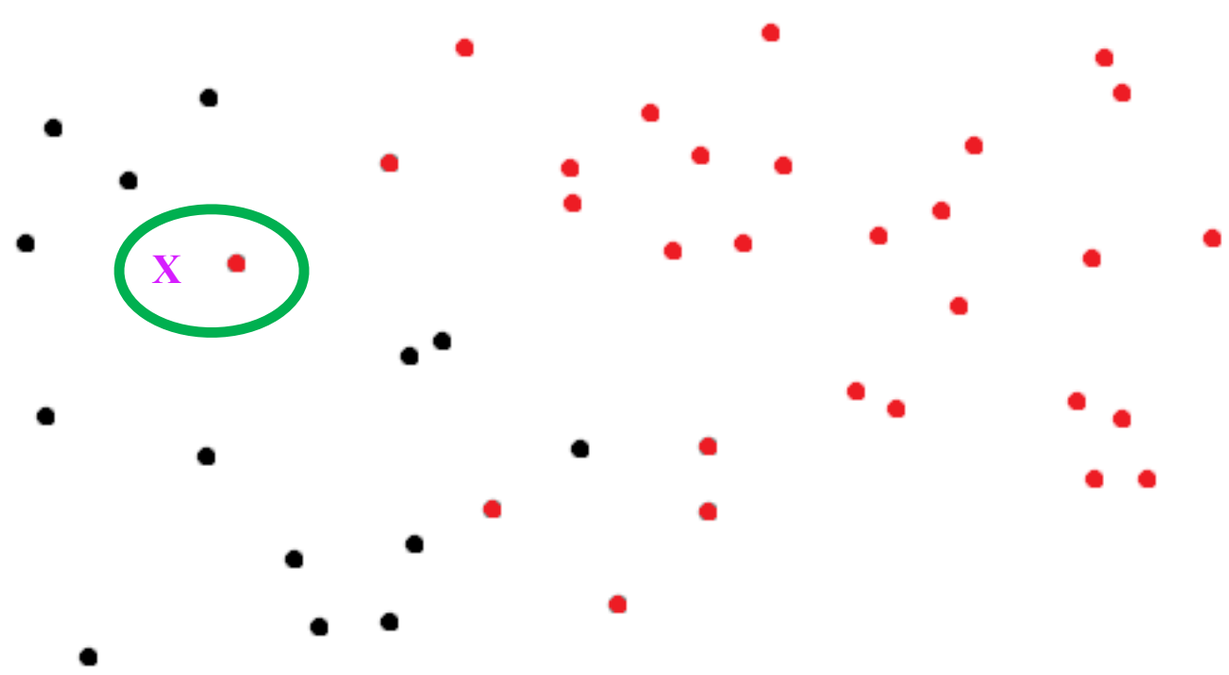
- Given training data:
  - Observations xi
  - Labels or outcome yi
- Predict y given unseen x



- Idea: use closest xi and assume outcome will be the same
  - Related to what we did earlier, but no clustering

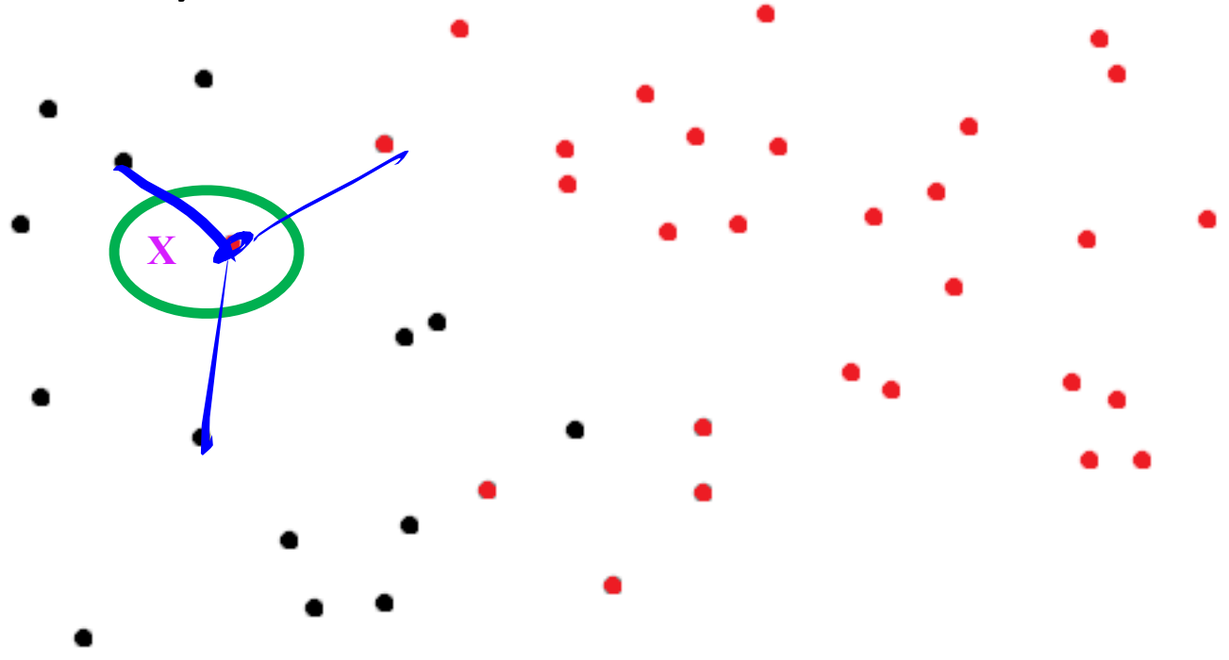
Using Distance Matrix for Classification

- Simplest approach is probably nearest neighbor
- When predicting the label of a new example
  - Find the nearest example in the training data
  - Predict the label associated with that example



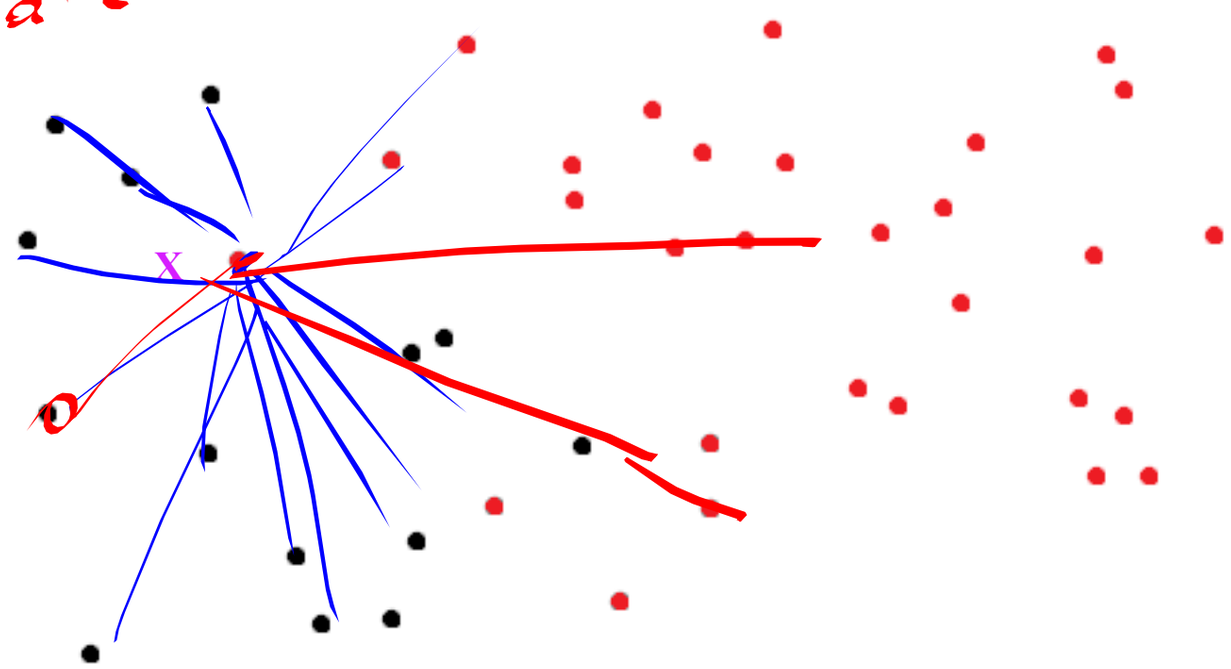
K Nearest Neighbor

- Simplest approach is probably nearest neighbor
- When predicting the label of a new example
  - Find the nearest example in the training data
  - Predict the label associated with that example
- Extension: k nearest neighbors KNN
  (majority vote of k nearest)



Finding k nearest neighbors

- initialize with first k
- try all remaining xi
  - compute distance
  - test < max current k
  - update



Code

- Idea:
  - Maintain list of k nearest neighbors so far
    - And list of their distance
  - Initialize with first k data points
  - For each of remaining N-k datapoint:
    - Test if distance smaller than farthest inn so far
    - If yes, replace the neighbor with new point

Code

```
def findKNearest(example, exampleSet, k):
    kNearest, distances = [], []
    #Build lists containing first k examples and their distances
    for i in range(k):
        kNearest.append(exampleSet[i])
        distances.append(example.distance(exampleSet[i]))
    maxDist = max(distances) #Get maximum distance
    #Look at examples not yet considered
    for e in exampleSet[k:]:
        dist = example.distance(e)
        if dist < maxDist:
            #replace farther neighbor by this one
            maxIndex = distances.index(maxDist)
            kNearest[maxIndex] = e
            distances[maxIndex] = dist
            maxDist = max(distances)
    return kNearest, distances
```

```
def KNearestClassify(training, testSet, label, k):
    """Assumes training & testSet lists of examples, k an int
       Predicts whether each example in testSet has label
       Returns number of true positives, false positives,
          true negatives, and false negatives"""
    truePos, falsePos, trueNeg, falseNeg = 0, 0, 0, 0
    for testCase in testSet:
        nearest, distances = findKNearest(testCase, training, k)
        #conduct vote
        numMatch = 0
        for i in range(len(nearest)):
            if nearest[i].getLabel() == label:
                numMatch += 1
        if numMatch > k//2: #guess label
            if testCase.getLabel() == label:
                truePos += 1
            else:
                falsePos += 1
        else: #guess not label
            if testCase.getLabel() != label:
                trueNeg += 1
            else:
                falseNeg += 1
    return truePos, falsePos, trueNeg, falseNeg
```

Questions?


WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

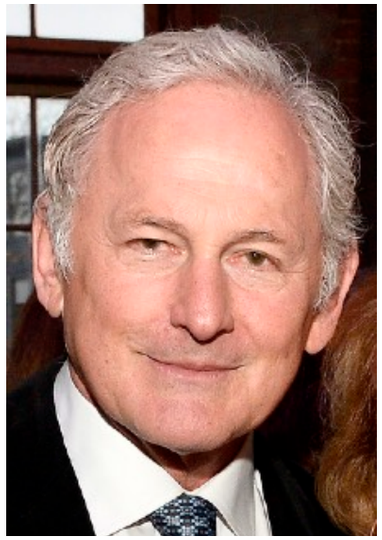
The Titanic Disaster

POLL

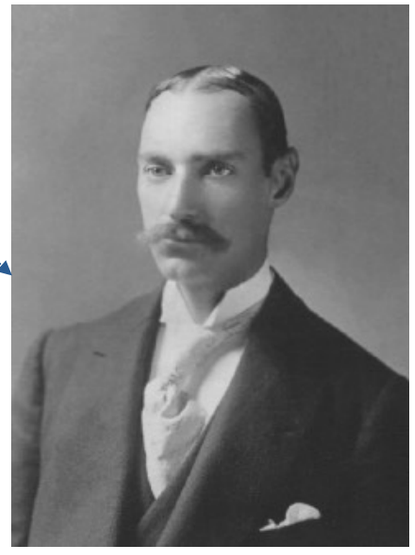
- RMS Titanic sank in the North Atlantic the morning of 15 April 1912, after colliding with an iceberg. Of the 1,300 passengers aboard, 812 died. (703 of 918 crew members died.)
- Database of 1046 passengers
  - Cabin class
    - 1st, 2nd, 3rd
  - Age
  - Gender
  - Outcome
    - (59% of passengers died)
  - Name

Sample of Data

1,0.92,M,1,Allison, Master. Hudson Trevor
1,2.0,F,0,Allison, Miss. Helen Loraine
1,30.0,M,0,Allison, Mr. Hudson Joshua Creighton
1,25.0,F,0,Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
1,39.0,M,0,Andrews, Mr. Thomas Jr
1,47.0,M,0,Astor, Col. John Jacob
1,18.0,F,1,Astor, Mrs. John Jacob (Madeleine Talmadge Force)
1,24.0,F,1,Aubart, Mme. Leontine Pauline
1,26.0,F,1,Barber, Miss. Ellen ""Nellie""
1,80.0,M,1,Barkworth, Mr. Algernon Henry Wilson
1,24.0,M,0,Baxter, Mr. Quigg Edmond
2,33.0,F,1,West, Mrs. Edwy Arthur (Ada Mary Worth)
2,66.0,M,0,Wheadon, Mr. Edward H
2,31.0,M,1,Wilhelms, Mr. Charles
2,26.0,F,1,Wright, Miss. Marion
2,24.0,F,0,Yrois, Miss. Henriette (""Mrs Harbeck"")
3,42.0,M,0,Abbing, Mr. Anthony
3,13.0,M,0,Abbott, Master. Eugene Joseph
3,16.0,M,0,Abbott, Mr. Rossmore



Victor Garber as Thomas Andrews



John Jacob Astor IV

Class Passenger

```
class Passenger(object):
    featureNames = ('C1', 'C2', 'C3', 'age', 'male gender')
    def __init__(self, pClass, age, gender, survived, name):
        self.name = name
        self.featureVec = [0, 0, 0, age, gender]
        self.featureVec[pClass - 1] = 1
        self.label = survived
        self.cabinClass = pClass
    def distance(self, other):
        return minkowskiDist(self.featureVec, other.featureVec, 2)
```

Why treat cabin class as three binary variables?

Should we scale age?

Let's Try KNN

```
def KNearestClassify(training, testSet, label, k):
    """Assumes training & testSet lists of examples, k an int
       Predicts whether each example in testSet has label
       Returns number of true positives, false positives,
          true negatives, and false negatives"""

for k in range(1, 5):
    knn = lambda training, testSet:\
                KNearestClassify(training, testSet,
                                 'Survived', k)
    numSplits = 10
    print('Average of', numSplits,
          '80/20 splits using KNN (k=' + str(k) + ')')
    truePos, falsePos, trueNeg, falseNeg =\
          randomSplits(examples, knn, numSplits)

    print('Average of LOO testing using KNN (k=' + str(k) + ')')
    truePos, falsePos, trueNeg, falseNeg =\
          leaveOneOut(examples, knn)
```

Observations About Results

Average of 10 80/20 splits using KNN (k=3)

Accuracy = 0.785

Sensitivity = 0.712

Specificity = 0.838

Pos. Pred. Val. = 0.761

> 59 %

Observations About Results

Average of 10 80/20 splits using KNN (k=3)
Accuracy = 0.785
Sensitivity = 0.712
Specificity = 0.838
Pos. Pred. Val. = 0.761

Accuracy significantly better than 59%
(which is the fraction of passengers that died,
and therefore your accuracy if you guess 1 all the times)

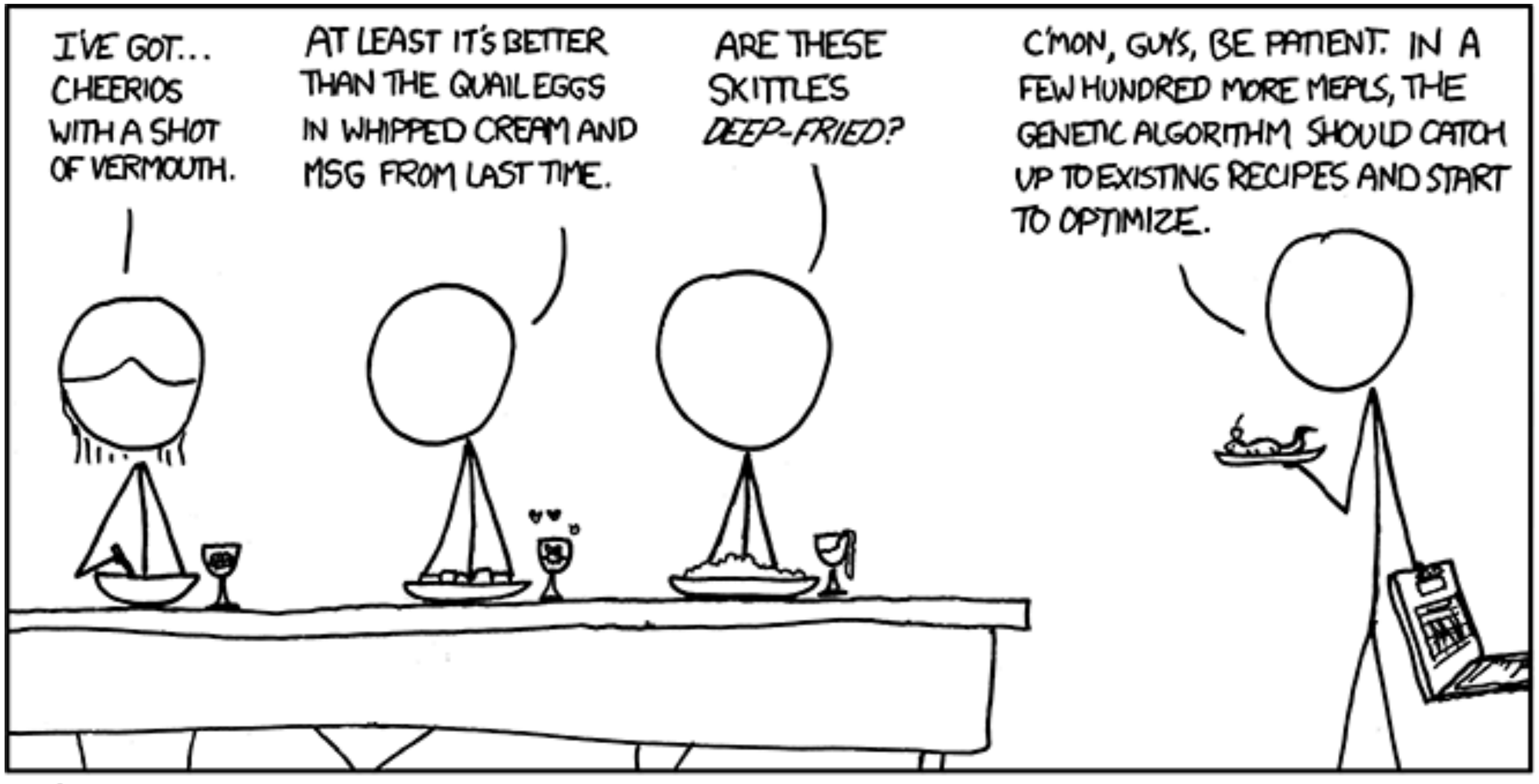
What's in a name and a k?

- Knn has little to do with k-means
  - Except there is a magical parameter k
  - And some distance
- K means is an iterative clustering method
  - Which can sometimes also be used for classification

Advantages and Disadvantages of KNN

- Advantages
  - Learning fast, no explicit training
  - No theory required
  - Easy to explain method and results
  - Scales well with more data
- Disadvantages
  - Memory intensive and predictions can take a long time
    - What is complexity of brute force algorithm?
  - No model to shed light on process that generated data
  - Needs a fair amount of training data

Questions?


I'VE GOT... CHEERIOS WITH A SHOT OF VERMOUTH.
AT LEAST IT'S BETTER THAN THE QUAIL EGGS IN WHIPPED CREAM AND MSG FROM LAST TIME.
ARE THESE SKITTLES DEEP-FRIED?
C'MON, GUYS, BE PATIENT. IN A FEW HUNDRED MORE MEALS, THE GENETIC ALGORITHM SHOULD CATCH UP TO EXISTING RECIPES AND START TO OPTIMIZE.
WE'VE DECIDED TO DROP THE CS DEPARTMENT FROM OUR WEEKLY DINNER PARTY HOSTING ROTATION.

Today's Lecture

- Metrics for evaluating a classification model
- Two common classification methods
  - K-nearest neighbors
  - Logistic regression
  - Representing two extremes of Machine learning continuum

Can we use something like linear regression for classification?

- Given a set of xi = (xi1, xi2, xi3, xi4..) and yi
- Find a formula like

$$y = a x_1 + b x_2 + c x_3 + d x_4 + e$$

Can we use something like linear regression?

- Given a set of xi = (xi1, xi2, xi3, xi4..) and yi
- Find a formula like
  - Yi = a xi1 +b xi2 +c xi3+d xi4
  - Each weight (a, b, c, d) tells usia the feature suggests a positive outcome (positive weight) resp. negative
  - Called a linear classifier

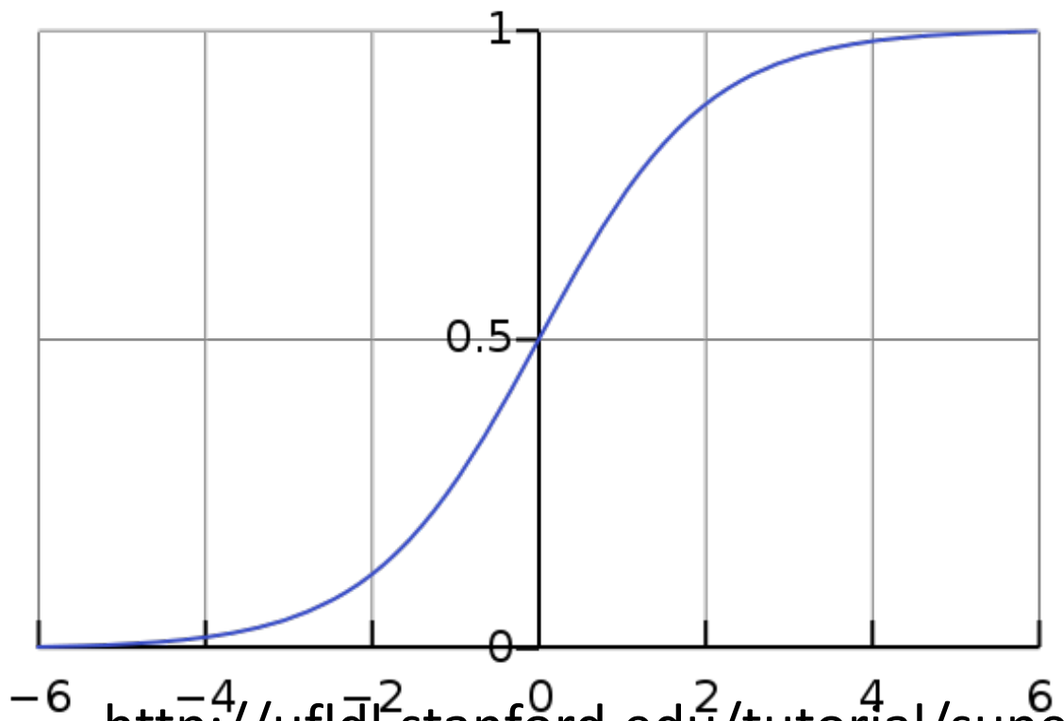
Can we use something like linear regression?

- Given a set of xi = (xi1, xi2, xi3, xi4..) and yi
- Find a formula like
  - Yi = a xi1 +b xi2 +c xi3+d xi4
  - Each weight (a, b, c, d) tells usia the feature suggests a positive outcome (positive weight) resp. negative
  - Called a linear classifier

- Problem:
  linear functions take values in R, we want binary values

Logistic function

- Add a rectifying function that maps real values to [-1, 1]

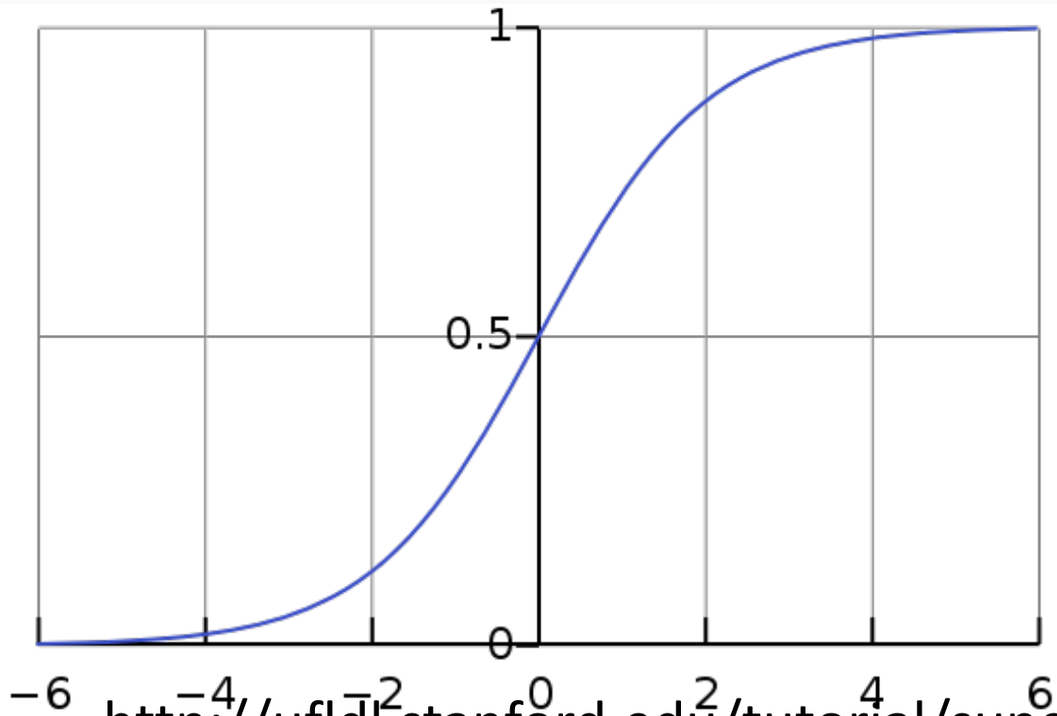


http://ufldl.stanford.edu/tutorial/supervised/LogisticRegression/

Logistic regression - math teaser

$$P(y = 1|x) = h_\theta(x) = \frac{1}{1 + \exp(-\theta^\top x)} \equiv \sigma(\theta^\top x),$$
$$P(y = 0|x) = 1 - P(y = 1|x) = 1 - h_\theta(x).$$

$$J(\theta) = -\sum_i \left(y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right).$$



http://ufldl.stanford.edu/tutorial/supervised/LogisticRegression/

Logistic Regression

- Analogous to linear regression
- Designed explicitly for predicting probability of an event
  - Dependent variable (label) can only take on a finite set of values. Usually 0 or 1
- Finds weights for each feature
  - Positive implies variable positively correlated with outcome
  - Negative implies variable negatively correlated with outcome
  - Absolute magnitude related to strength of the correlation
- Optimization problem a bit complex, key is use of a log function—won't make you look at it

Class LogisticRegression

```
import sklearn.linear_model
```

fit(sequence of feature vectors, sequence of labels)

Returns object of type LogisticRegression

coef_

Returns weights of features

predict_proba(feature vector)

Returns probabilities of labels

Building a Model

```
def buildModel(examples, toPrint = True):
    featureVecs, labels = [],[]
    for e in examples:
        featureVecs.append(e.getFeatures())
        labels.append(e.getLabel())
    LogisticRegression = sklearn.linear_model.LogisticRegression
    model = LogisticRegression().fit(featureVecs, labels)
```

Applying Model

```
def applyModel(model, testSet, label, prob = 0.5):
    testFeatureVecs = [e.getFeatures() for e in testSet]
    probs = model.predict_proba(testFeatureVecs)
    truePos, falsePos, trueNeg, falseNeg = 0, 0, 0, 0
    for i in range(len(probs)):
        if probs[i][1] > prob:
            if testSet[i].getLabel() == label:
                truePos += 1
            else:
                falsePos += 1
        else:
            if testSet[i].getLabel() != label:
                trueNeg += 1
            else:
                falseNeg += 1
    return truePos, falsePos, trueNeg, falseNeg
```

Putting It Together

```
def lr(trainingData, testData, prob = 0.5):
    model = buildModel(trainingData, False)
    results = applyModel(model, testData, 'Survived', prob)
    return results

numSplits = 10
print('Average of', numSplits, '80/20 splits LR')
truePos, falsePos, trueNeg, falseNeg =\
      divide80_20(examples, lr, numSplits)

print('Average of LOO testing using LR')
truePos, falsePos, trueNeg, falseNeg =\
      leaveOneOut(examples, lr)
```

Results

KNN

Average of 10 80/20 splits using KNN

- Accuracy = 0.785
- Sensitivity = 0.712
- Specificity = 0.838
- Pos. Pred. Val. = 0.761

Logistic Regression

Average of 10 80/20 splits LR

- Accuracy = 0.771
- Sensitivity = 0.697
- Specificity = 0.824
- Pos. Pred. Val. = 0.742

Results

KNN

Average of 10 80/20 splits using KNN
- Accuracy = 0.785
- Sensitivity = 0.712
- Specificity = 0.838
- Pos. Pred. Val. = 0.761

Logistic Regression

Average of 10 80/20 splits LR
- Accuracy = 0.771
- Sensitivity = 0.697
- Specificity = 0.824
- Pos. Pred. Val. = 0.742

Logistic regression also provides insight about variables

Also facilitates tradeoff between precision and recall (see next slides)

Recap: Today's Lecture

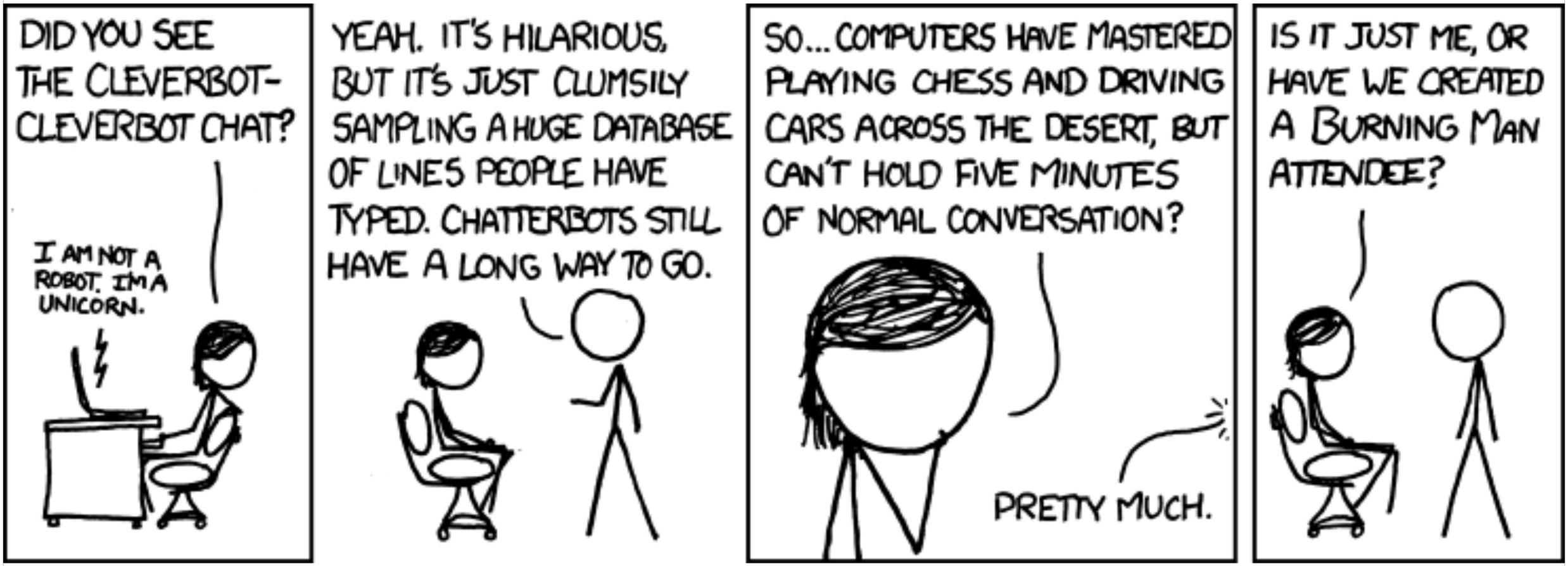
- Metrics for evaluating a classification model
- Two common classification methods
  - K-nearest neighbors
  - Logistic regression
  - Representing two extremes of Machine learning continuum

Recap: Today's Lecture

- **K-nearest neighbors**
- "memory-based" learning (non-parametric)
- No training
- Expensive at test time (memory, compute)
  - Can be helped by clustering pre-processing to reduce samples
- Very flexible, more powerful with more data
- Needs a lot of data to work
- Easy to understand how training data impacts prediction
- Scale/metrics are critical
  - Careful: Geometry of high dimensional spaces is complicated

- **Logistic regression**
- Model-based learning (aka parametric)
- Cheap at test time
- Interpretable model
  - Although be careful with this because features can be correlated
- Gives probability
- Can work with limited data
- Constrained model
  - Not all decision boundaries are straight lines
- Mostly independent of scale

Questions?


DID YOU SEE THE CLEVERBOT-CLEVERBOT CHAT?
I AM NOT A ROBOT. I'M A UNICORN.
YEAH. IT'S HILARIOUS, BUT IT'S JUST CLUMSILY SAMPLING A HUGE DATABASE OF LINES PEOPLE HAVE TYPED. CHATTERBOTS STILL HAVE A LONG WAY TO GO.
SO... COMPUTERS HAVE MASTERED PLAYING CHESS AND DRIVING CARS ACROSS THE DESERT, BUT CAN'T HOLD FIVE MINUTES OF NORMAL CONVERSATION?
PRETTY MUCH.
IS IT JUST ME, OR HAVE WE CREATED A BURNING MAN ATTENDEE?