ECE5984 – Applications of Machine Learning
Lecture 12 – Classification

Creed Jones, PhD

Course update

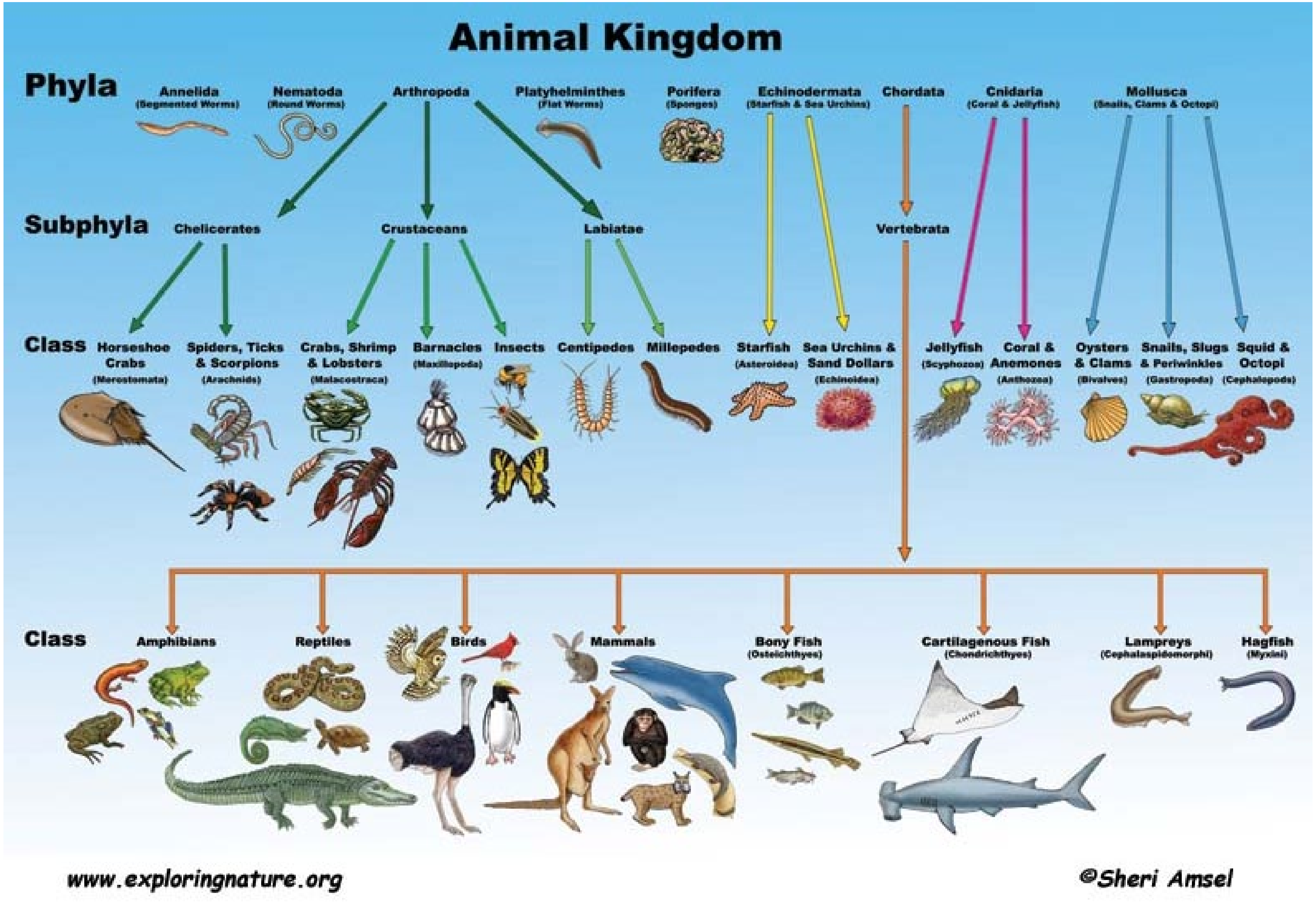
- HW3 is posted
  - Due Tuesday, March 1
- Project I has been posted
  - Due Tuesday, March 22
- Quiz 3 TODAY
  - Quiz 4 next Thursday, March 3
- Spring break in two weeks
  - I won't have office hours during Spring break

Today’s Objectives

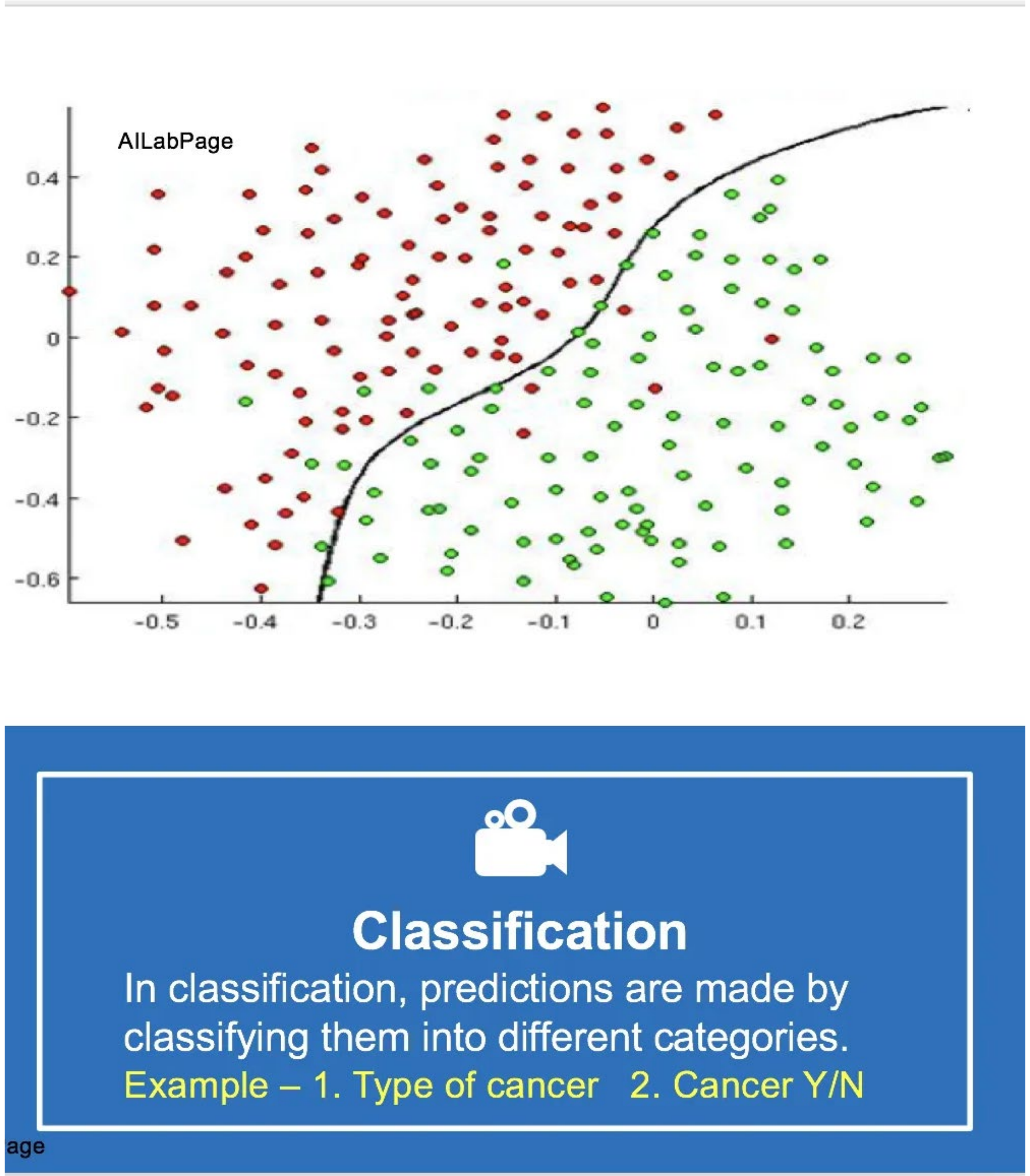
- Classification
  - *K*-nearest neighbor
- Handling Noisy Data
  - Weighted nearest neighbor
- Data Normalization
- Continuous Targets using kNN
- Other Similarity Measures
  - Binary similarity measures
  - Cosine similarity

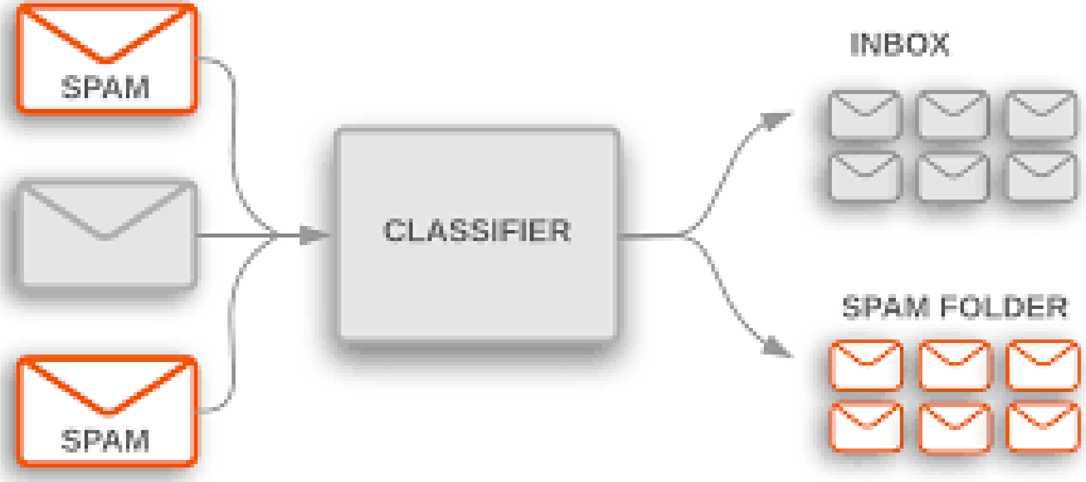
Distance or similarity measures can be used to assign vectors to classes

- This is called classification
  - Assigning new vectors to existing classes defined by a set of labeled samples
- Classification can be binary (two-class)
  - Spam / Not spam
  - Good / Bad
- Or Multiclass
  - Orange / Lime / Grapefruit / Lemon
  - Red / Green / Blue



Animal Kingdom
Phyla
Annelida
(Segmented Worms)
Nematoda
(Round Worms)
Arthropoda
Platyhelminthes
(Flat Worms)
Porifera
(Sponges)
Echinodermata
(Starfish & Sea Urchins)
Chordata
Cnidaria
(Coral & Jellyfish)
Mollusca
(Snails, Clams & Octopi)
Subphyla
Chelicerates
Crustaceans
Labiatae
Vertebrata
Class
Horseshoe Crabs
(Merostomata)
Spiders, Ticks & Scorpions
(Arachnids)
Crabs, Shrimp & Lobsters
(Malacostraca)
Barnacles
(Maxillopoda)
Insects
Centipedes
Millepedes
Starfish
(Asteroidea)
Sea Urchins & Sand Dollars
(Echinoidea)
Jellyfish
(Scyphozoa)
Coral & Anemones
(Anthozoa)
Oysters & Clams
(Bivalves)
Snails, Slugs & Periwinkles
(Gastropoda)
Squid & Octopi
(Cephalopods)
Class
Amphibians
Reptiles
Birds
Mammals
Bony Fish
(Osteichthyes)
Cartilagenous Fish
(Chondrichthyes)
Lampreys
(Cephalaspidomorphi)
Hagfish
(Myxini)
www.exploringnature.org
©Sheri Amsel



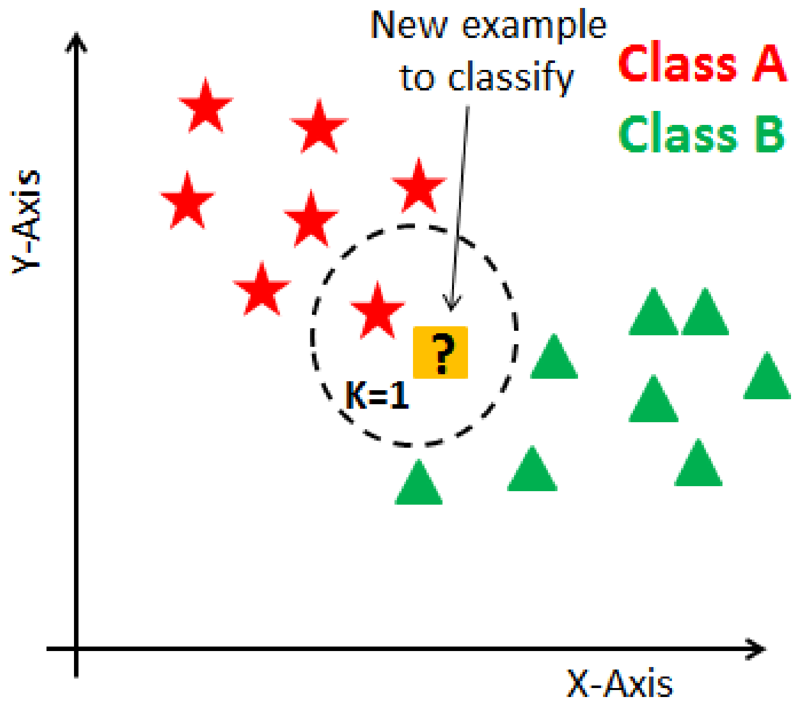


Vinod Sharma's Blog

One common classification technique is called the *nearest-neighbor* method

- Given a set of training instances and a new vector to be classified:
  - Iterate across the instances in memory and find the instance that is the shortest distance from the new vector in the feature space.
  - The vector is assigned the target value of the nearest neighbor.

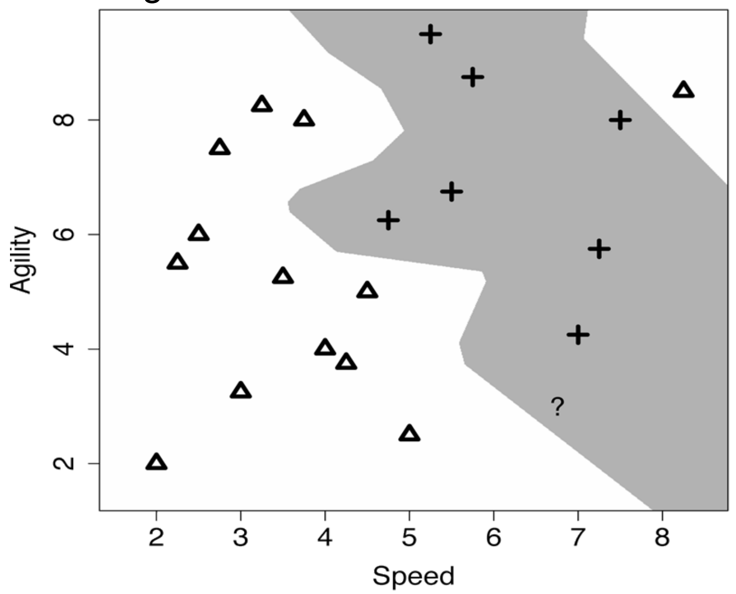
- If new vectors are added to the training set, results will depend on the order of presentation



The *nearest neighbor* algorithm is intuitive – a new sample is labeled to be the same as the one closest to it in feature space

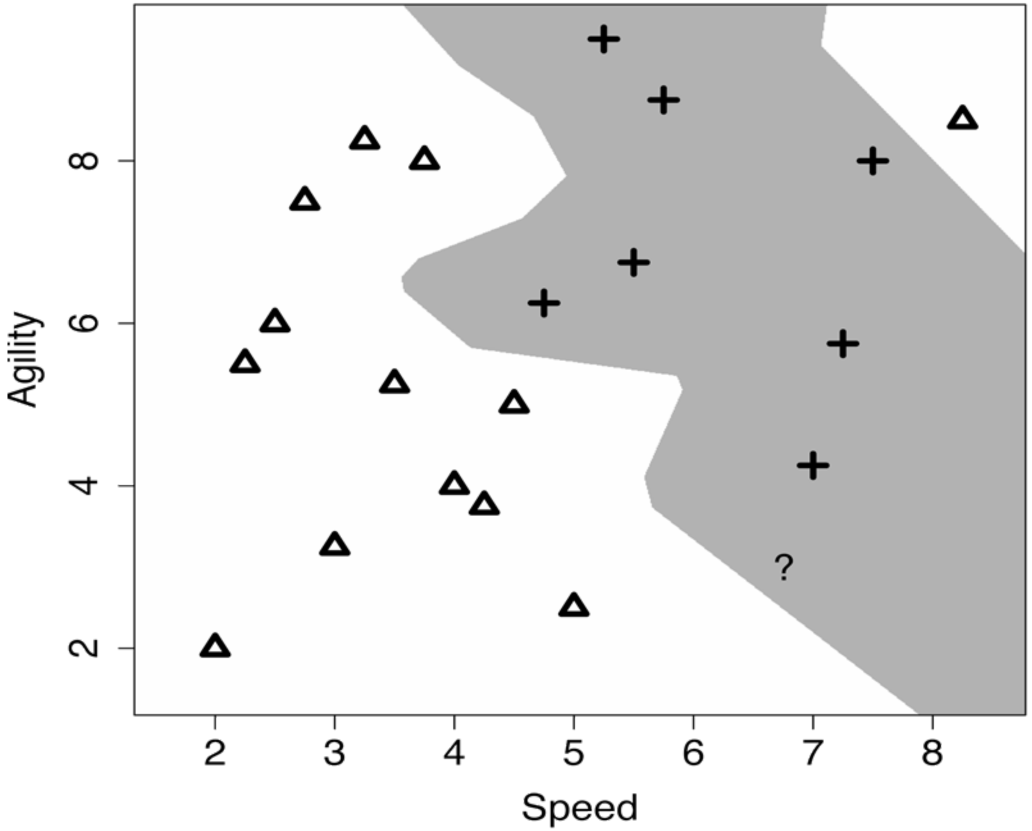
- The *nearest neighbor* model predicts the target level to be the same as the nearest neighbor to the query q
- The decision boundary marks the parts of the feature space that would be classed as each of the possible values

The decision boundary using nearest neighbor classification.



HANDLING NOISY DATA

Is the instance at the top right of the diagram really noise?



The *k nearest neighbors* algorithm can reduce sensitivity to a single outlier

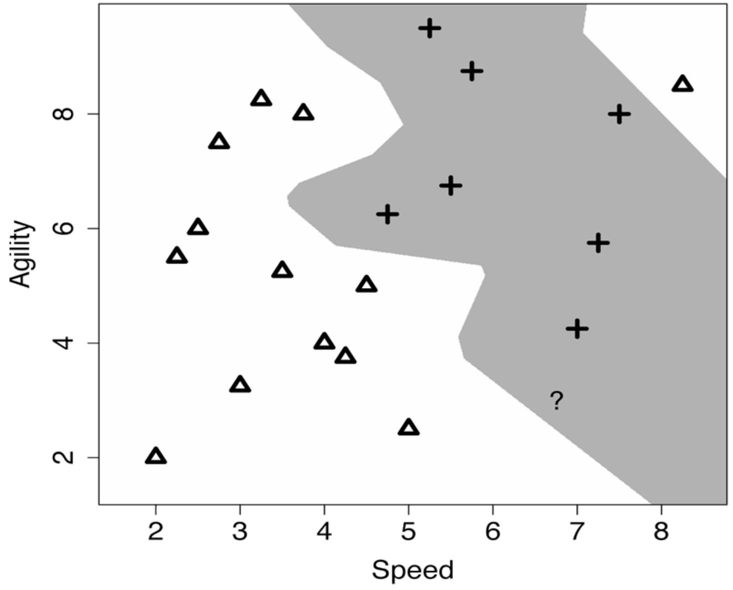
- The *k nearest neighbors* model predicts the target level with the majority vote from the set of k nearest neighbors to the query q:

$$\mathbb{M}_k(\boldsymbol{q}) = \underset{l \in levels(t)}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(t_i, l)$$

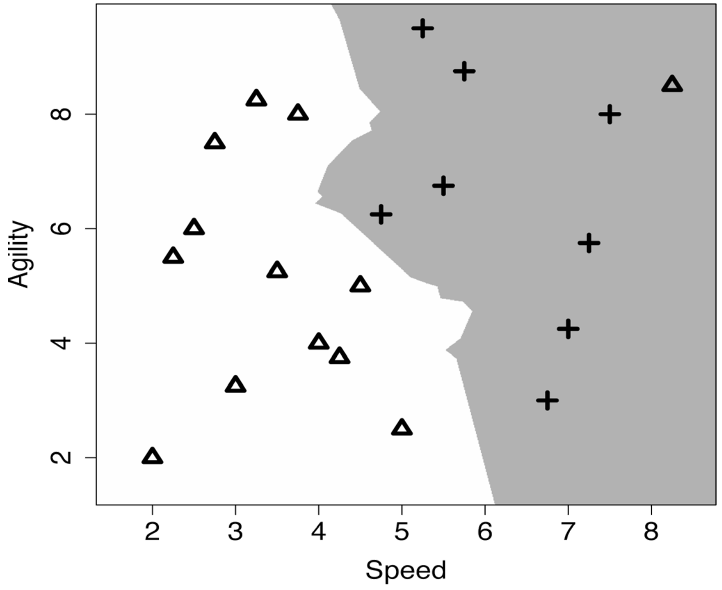
$\delta(a, b)$ is the *Kronecker delta*

$$\delta(a,b) = \begin{cases} 1, a = b \\ 0, a \neq b \end{cases}$$

The decision boundary using nearest neighbor classification.

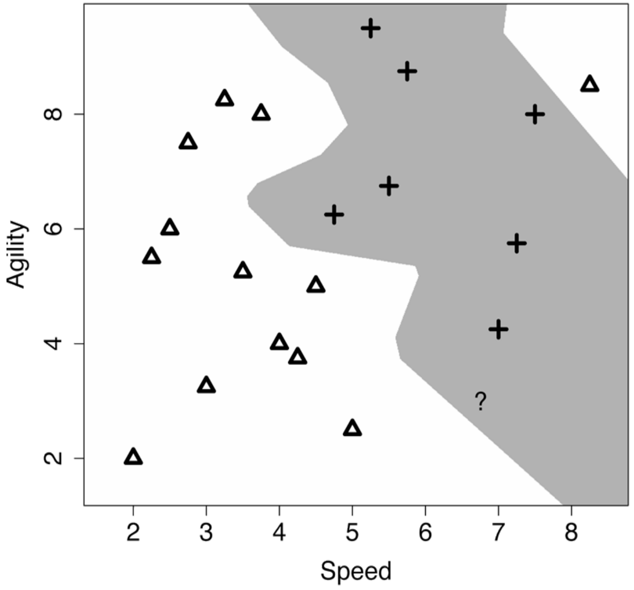


The decision boundary using majority classification of the nearest 3 neighbors.

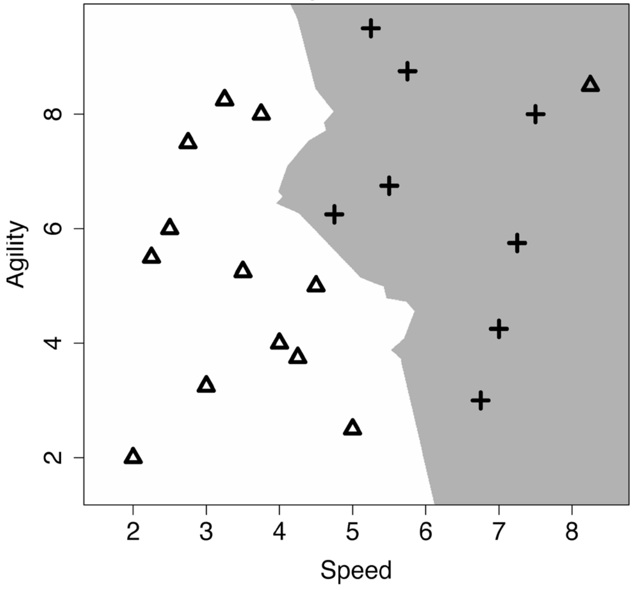


Larger values of *k* can suppress more outliers but can cause inconsistencies

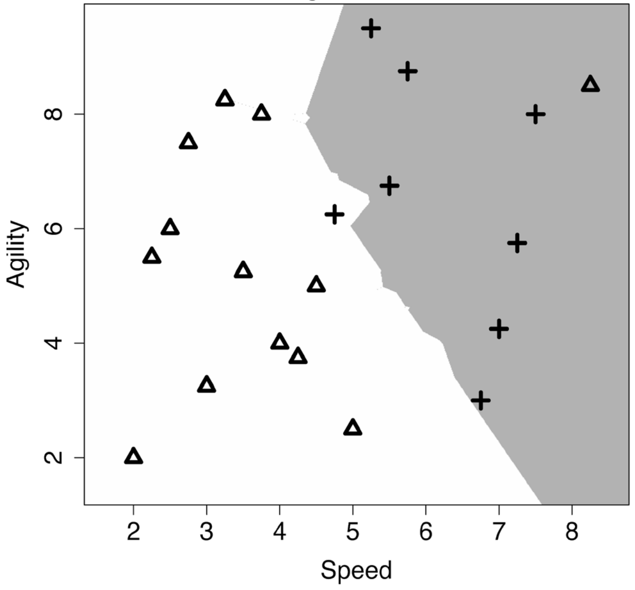
The decision boundary using nearest neighbor classification.



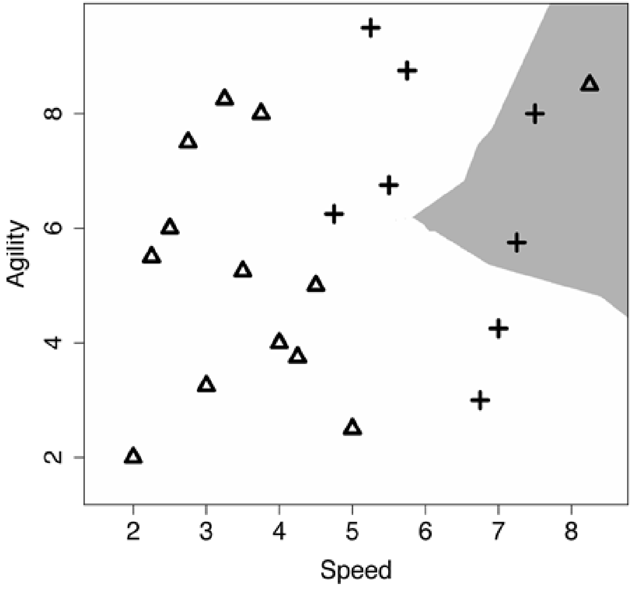
The decision boundary using majority classification of the nearest 3 neighbors.



The decision boundary using majority classification of the nearest 5 neighbors.



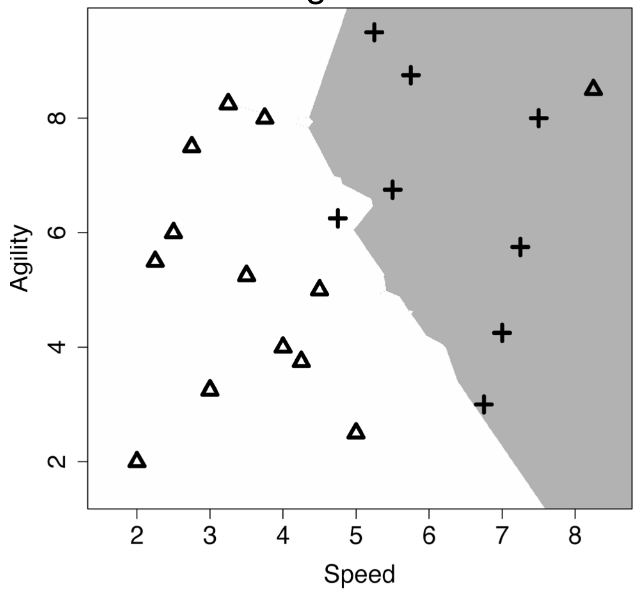
The decision boundary using majority classification of the nearest 15 neighbors.



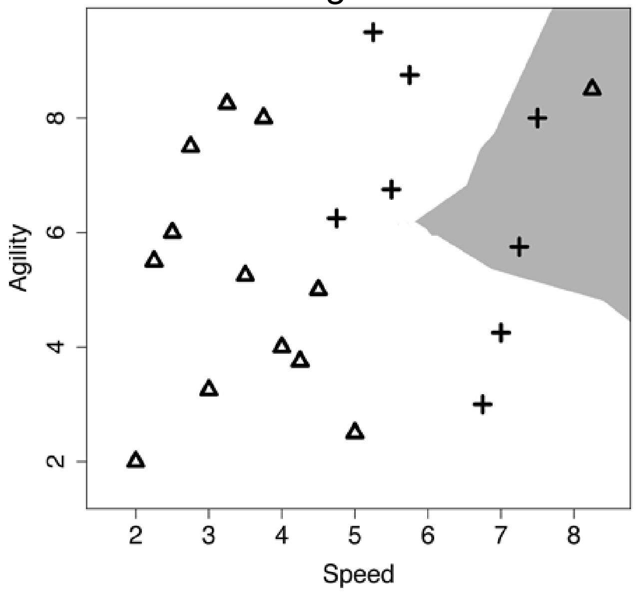
An imbalanced dataset (nonuniform distribution of the target) increases the risks of a high *k*

- If we have a higher proportion of one value (as here – there are more “no” than “yes”), then it’s more likely for any new vector to be close to *k* samples in the majority class
- The majority class(es) tend to expand excessively

The decision boundary using majority classification of the nearest 5 neighbors.



The decision boundary using majority classification of the nearest 15 neighbors.



The *weighted k nearest neighbor* clustering algorithm reduces the effect of class vectors further from the new vector

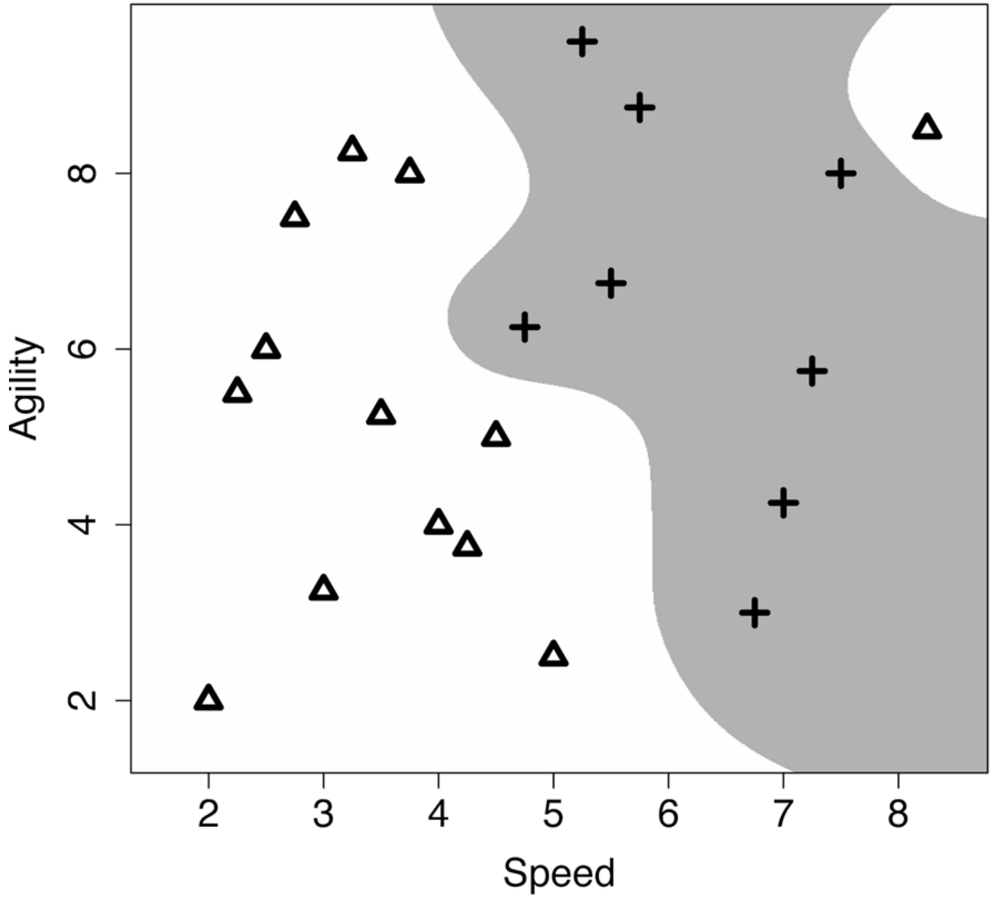
- In a distance weighted k nearest neighbor algorithm the contribution of each neighbor to the classification decision is weighted by the reciprocal of the squared distance between the neighbor $\boldsymbol{d}$ and the query $\boldsymbol{q}$:

$$w = \frac{1}{dist(\boldsymbol{q}, \boldsymbol{d})^2}$$

- So, the query is not simple voting, but includes the distance weighting:

$$\mathbb{M}_k(\boldsymbol{q}) = \underset{l \in levels(t)}{\operatorname{argmax}} \sum_{i=1}^{k} \frac{\delta(t_i, l)}{dist(\boldsymbol{q}, \boldsymbol{d})^2}$$

The weighted *k* nearest neighbor model decision boundary, with *k*=21 (!)



Weighted *k* nearest neighbor is robust, but there are two situations that can cause it to be a poor choice

1. If the dataset is extremely unbalanced, then the weighting is still not enough to prevent the algorithm from preferring the majority class
   - But if the training set is far from balanced, we may need to think about whether simple distance is the only criterion to use in clustering…

2. If the dataset is large, it may be impractical to calculate the inverse of the squared distance for all training instances

   - But, we have to compute the distances anyway, so the added time may not be too long (the algorithm still has the same order of growth)

```
import …

dirname = "C:/Data/Solar Flare/"
filename = "FlareData.xlsx"
df = pandas.read_excel(dirname + filename)        # read an Excel spreadsheet
print('File ', filename, ' is of size ', df.shape)
df['Zurich Class'].replace(('A','B','C','D','E','F','H'), range(7), inplace=True)
df['Spot Size'].replace(('X','R','S','A','H','K'), range(6), inplace=True)
df['Spot Dist'].replace(('X','O','I','C'), range(4), inplace=True)
labels = df.columns
targetlabels = ['C class']
unusedlabels = ['M class', 'X class']
featurelabels = labels.drop(targetlabels+unusedlabels).values    # get just the predictors
print(df.dtypes)

features = df[featurelabels]
target = df[targetlabels]

X = features.to_numpy(np.float64)
Y = target.to_numpy(np.float64)
(trainX, testX, trainY, testY) = modelsel.train_test_split(X, Y, test_size=0.3, random_state=98043)

for nnbrs in [1, 2, 5, 8, 12]:
    for weighting in ['uniform', 'distance']:
        clf = nbrs.KNeighborsClassifier(n_neighbors=nnbrs, weights=weighting)      # create object to classify
        clf = clf.fit(trainX, trainY.ravel())  # train it on this data
        accuracy = clf.score(testX, testY)
        print("NN classifier, N = {0}, weighting = {1}; classification accuracy = {2:6.4}“
                            .format(nnbrs, weighting, accuracy))
```

NN classifier, N = 1, weighting = uniform; classification accuracy = 0.7835

DATA NORMALIZATION

Problems occur when we the dimensions of the dataset vary widely in range…

A dataset listing the salary and age information for customers and whether or not the purchased a pension plan.

ID	Salary	Age	Purchased
1	53700	41	No
2	65300	37	No
3	48900	45	Yes
4	64800	49	Yes
5	44200	30	No
6	55900	57	Yes
7	48600	26	No
8	72800	60	Yes
9	45300	34	No
10	73200	52	Yes

The marketing department wants to decide whether or not they should contact a customer with the following profile:

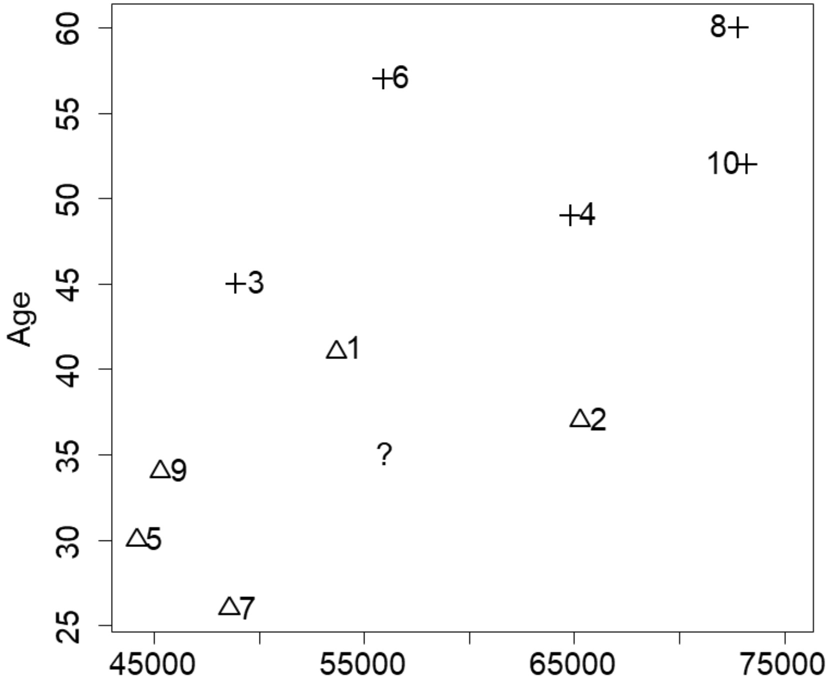
$$(\text{SALARY} = 56,000, \text{AGE} = 35)$$

Consider the nearest neighbor classification result on this data, with the query [56000, 35]

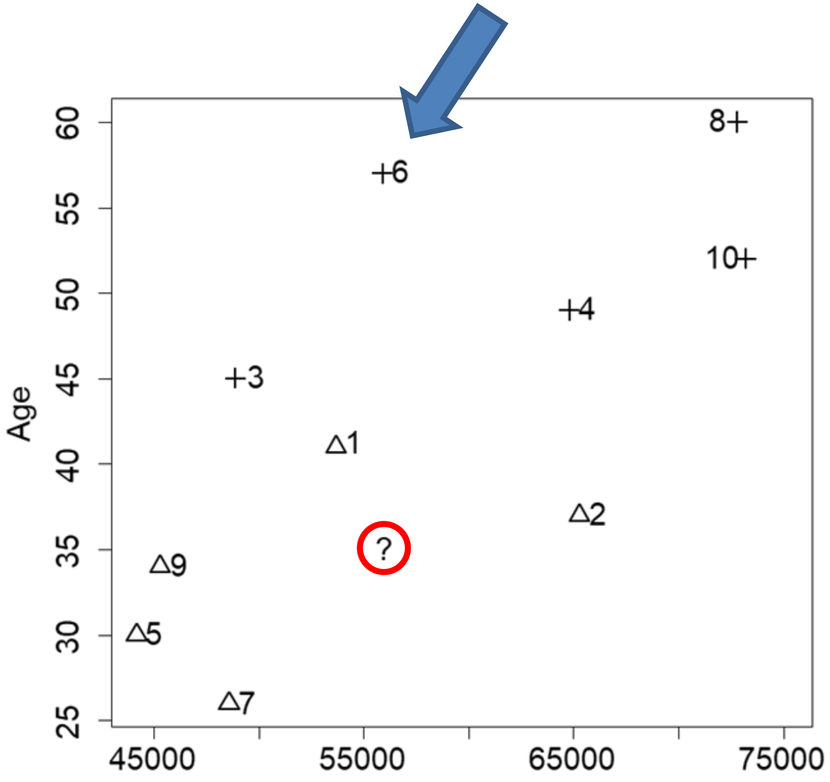
A dataset listing the salary and age information for customers and whether or not the purchased a pension plan.

ID	Salary	Age	Purchased
1	53700	41	No
2	65300	37	No
3	48900	45	Yes
4	64800	49	Yes
5	44200	30	No
6	55900	57	Yes
7	48600	26	No
8	72800	60	Yes
9	45300	34	No
10	73200	52	Yes

The salary and age feature space. The instances are labelled with their IDs, triangles represent negative instances and crosses represent positive instances. The location of the query (SALARY = 56, 000, AGE = 35) is indicated by the ?.

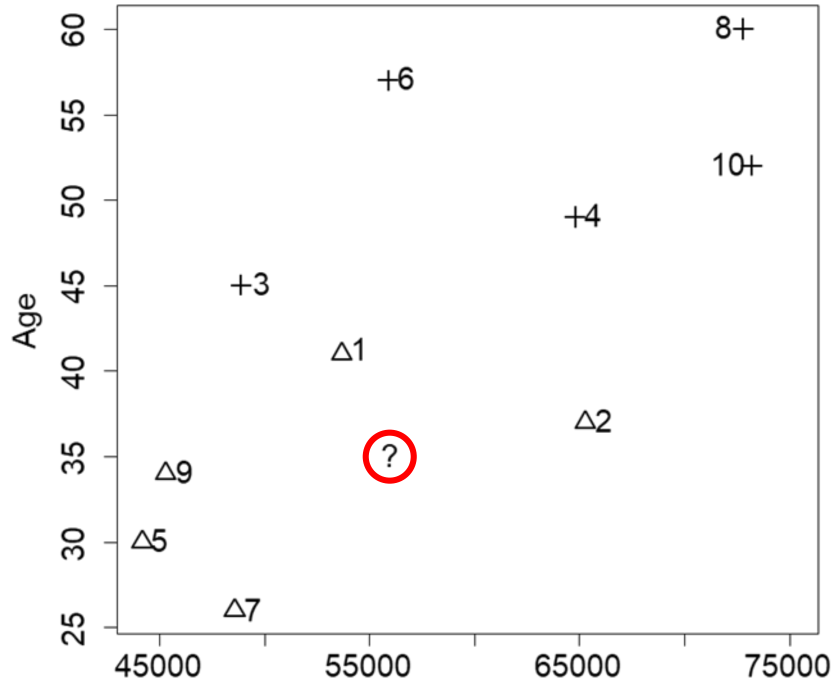


The training sample closest to [56000, 35] is example 6 (P=Yes), which does not look right!



ID	Salary	Age	Purch.	Salary and Age Dist.	Neigh.	Salary Only Dist.	Neigh.	Age Only Dist.	Neigh.
1	53700	41	No	2300.0078	2	2300	2	6	4
2	65300	37	No	9300.0002	6	9300	6	2	2
3	48900	45	Yes	7100.0070	3	7100	3	10	6
4	64800	49	Yes	8800.0111	5	8800	5	14	7
5	44200	30	No	11800.0011	8	11800	8	5	5
6	55900	57	Yes	102.3914	1	100	1	22	9
7	48600	26	No	7400.0055	4	7400	4	9	3
8	72800	60	Yes	16800.0186	9	16800	9	25	10
9	45300	34	No	10700.0000	7	10700	7	1	1
10	73200	52	Yes	17200.0084	10	17200	10	17	8

Why does NN find example 6 as the closest match?

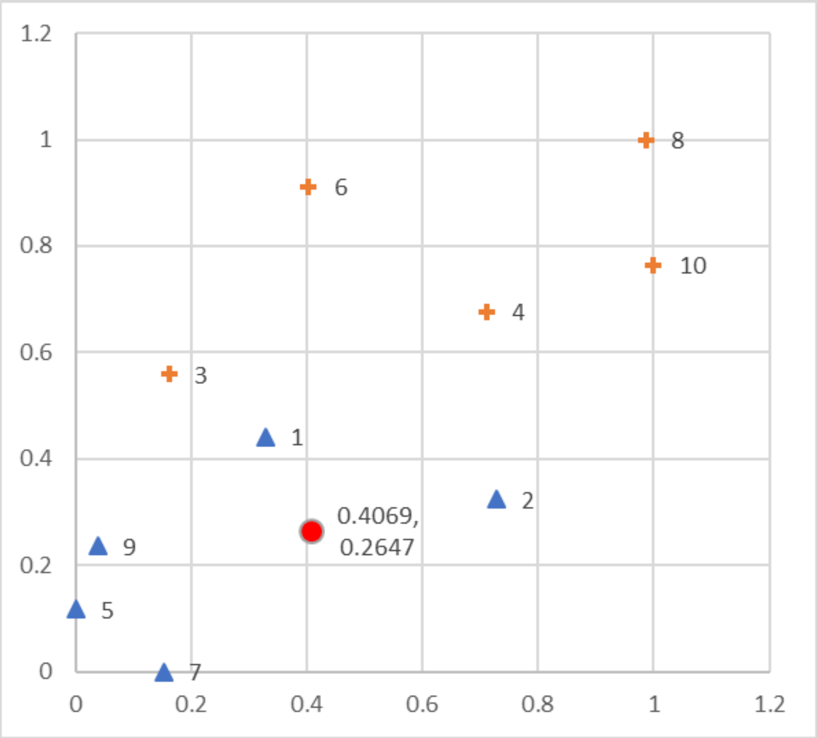


- This odd prediction is caused by features taking different ranges of values, this is equivalent to features having different variances.
- We can adjust for this using range (min-max) normalization; the equation for range normalization is

$$a'_i = \frac{\big(a_i - min(a)\big)(high - low)}{max(a) - min(a)} + low$$

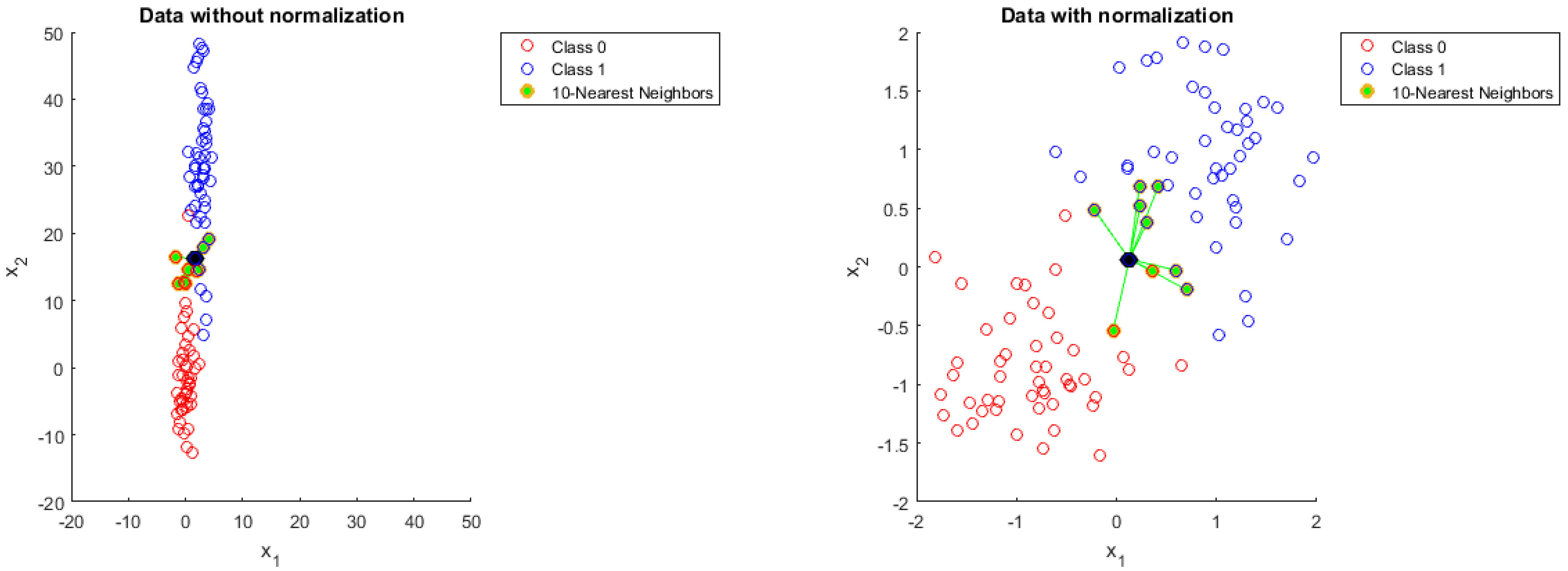
By using min-max normalization to transform to the range [0,1] in all dimensions, we can make the distance comparisons more consistent

VIRGINIA TECH
ece
BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING



ID	Normalized Dataset Salary	Age	Purch.	Salary and Age Dist.	Neigh.	Salary Only Dist.	Neigh.	Age Only Dist.	Neigh.
1	0.3276	0.4412	No	0.1935	1	0.0793	2	0.17647	4
2	0.7276	0.3235	No	0.3260	2	0.3207	6	0.05882	2
3	0.1621	0.5588	Yes	0.3827	5	0.2448	3	0.29412	6
4	0.7103	0.6765	Yes	0.5115	7	0.3034	5	0.41176	7
5	0.0000	0.1176	No	0.4327	6	0.4069	8	0.14706	3
6	0.4034	0.9118	Yes	0.6471	8	0.0034	1	0.64706	9
7	0.1517	0.0000	No	0.3677	3	0.2552	4	0.26471	5
8	0.9862	1.0000	Yes	0.9361	10	0.5793	9	0.73529	10
9	0.0379	0.2353	No	0.3701	4	0.3690	7	0.02941	1
10	1.0000	0.7647	Yes	0.7757	9	0.5931	10	0.50000	8

Normalizing the data is an important thing to do for almost all machine learning algorithms, not just nearest neighbor!



https://stats.stackexchange.com/questions/287425/why-do-you-need-to-scale-data-in-knn

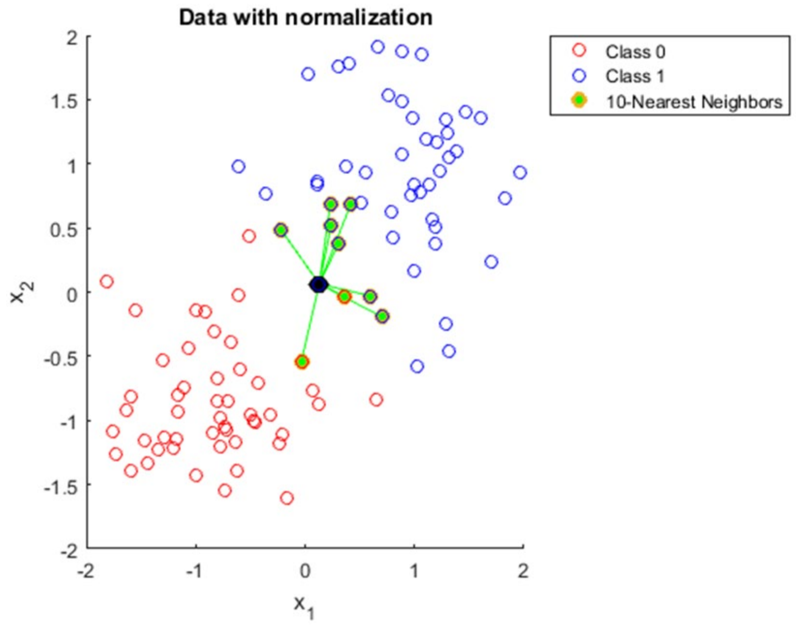
PREDICTING CONTINUOUS TARGETS USING KNN

We can use *k* nearest neighbor to predict continuous values of new query vectors by assigning the average of the values of the neighboring samples

- We return the average value of the target variable (not a dimension of the feature space) for the *k* neighbors

$$\mathbb{M}_k(q) = \frac{1}{k}\sum_{i=1}^{k} t_i$$

where t_i is the target variable for the i_{th} sample

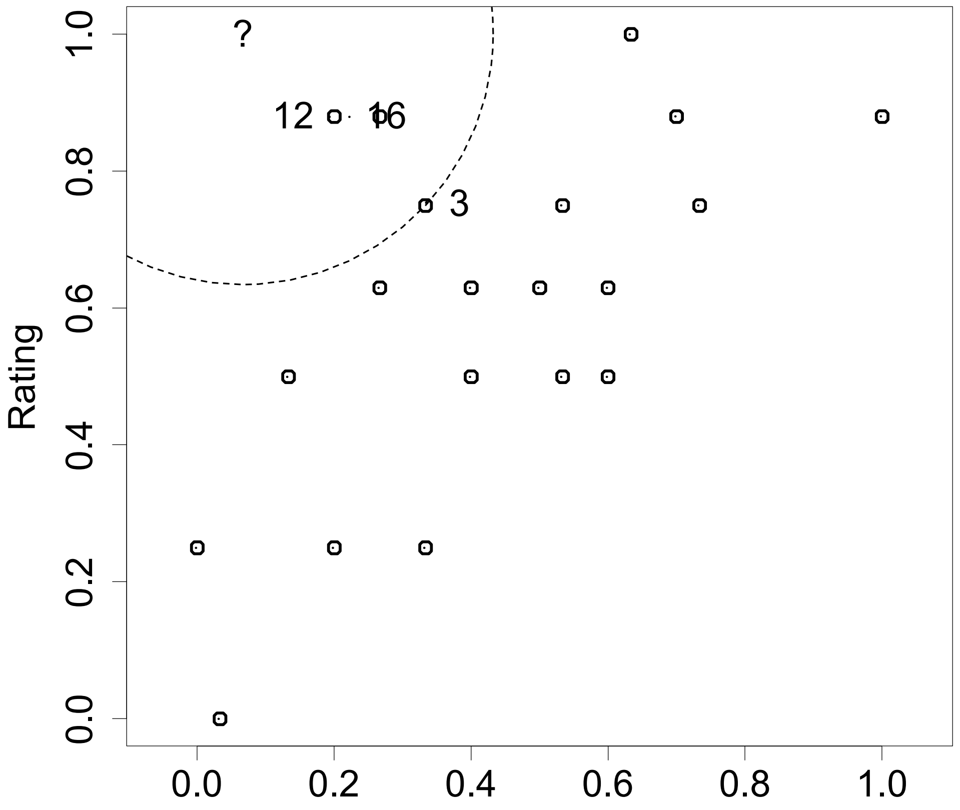


A dataset of whiskeys listing age (in years), rating (1 to 5, with 5 being the best) and the bottle price of each whiskey – in raw and range normalized form

ID	Age	Rating	Price
1	0	2	30.00
2	12	3.5	40.00
3	10	4	55.00
4	21	4.5	550.00
5	12	3	35.00
6	15	3.5	45.00
7	16	4	70.00
8	18	3	85.00
9	18	3.5	78.00
10	16	3	75.00
11	19	5	500.00
12	6	4.5	200.00
13	8	3.5	65.00
14	22	4	120.00
15	6	2	12.00
16	8	4.5	250.00
17	10	2	18.00
18	30	4.5	450.00
19	1	1	10.00
20	4	3	30.00

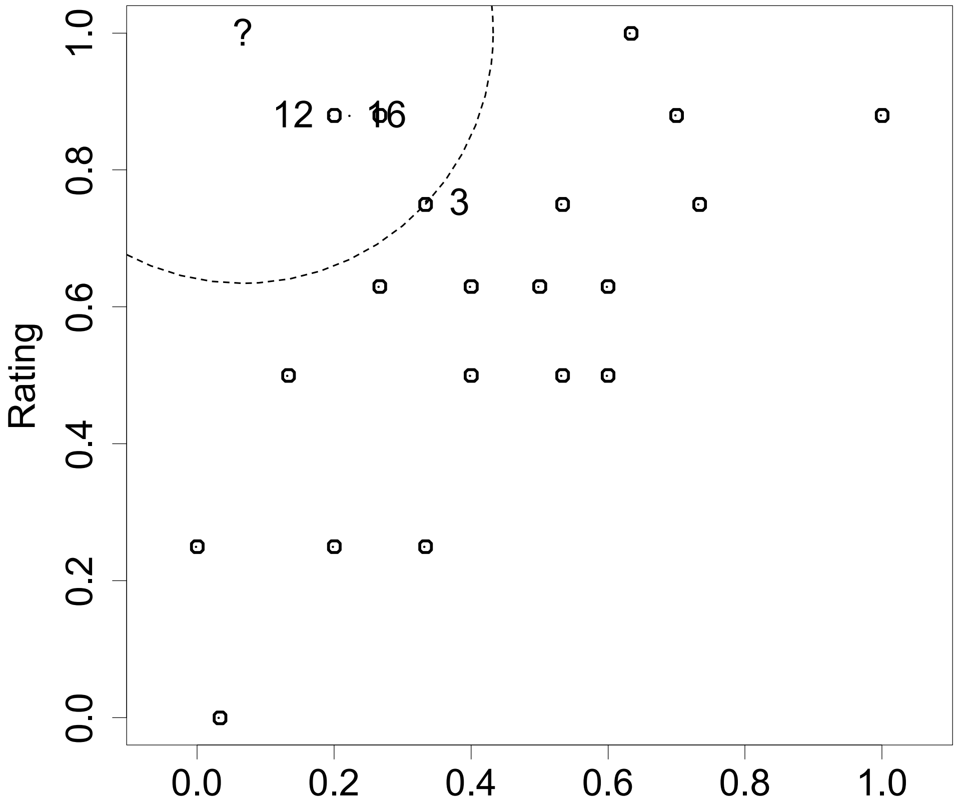
ID	Age	Rating	Price
1	0.0000	0.25	30.00
2	0.4000	0.63	40.00
3	0.3333	0.75	55.00
4	0.7000	0.88	550.00
5	0.4000	0.50	35.00
6	0.5000	0.63	45.00
7	0.5333	0.75	70.00
8	0.6000	0.50	85.00
9	0.6000	0.63	78.00
10	0.5333	0.50	75.00
11	0.6333	1.00	500.00
12	0.2000	0.88	200.00
13	0.2667	0.63	65.00
14	0.7333	0.75	120.00
15	0.2000	0.25	12.00
16	0.2667	0.88	250.00
17	0.3333	0.25	18.00
18	1.0000	0.88	450.00
19	0.0333	0.00	10.00
20	0.1333	0.50	30.00

ID	Age	Rating	Price
1	0.0000	0.25	30.00
2	0.4000	0.63	40.00
3	0.3333	0.75	55.00
4	0.7000	0.88	550.00
5	0.4000	0.50	35.00
6	0.5000	0.63	45.00
7	0.5333	0.75	70.00
8	0.6000	0.50	85.00
9	0.6000	0.63	78.00
10	0.5333	0.50	75.00
11	0.6333	1.00	500.00
12	0.2000	0.88	200.00
13	0.2667	0.63	65.00
14	0.7333	0.75	120.00
15	0.2000	0.25	12.00
16	0.2667	0.88	250.00
17	0.3333	0.25	18.00
18	1.0000	0.88	450.00
19	0.0333	0.00	10.00
20	0.1333	0.50	30.00



The [AGE, RATING] feature space for the normalized whiskey dataset. The location of the query instance is indicated by “?”. The circle plotted with a dashed line demarcates the border of the neighborhood around the query when $k = 3$. The three nearest neighbors to the query are labelled with their ID values.

ID	Age	Rating	Price
1	0.0000	0.25	30.00
2	0.4000	0.63	40.00
3	0.3333	0.75	55.00
4	0.7000	0.88	550.00
5	0.4000	0.50	35.00
6	0.5000	0.63	45.00
7	0.5333	0.75	70.00
8	0.6000	0.50	85.00
9	0.6000	0.63	78.00
10	0.5333	0.50	75.00
11	0.6333	1.00	500.00
12	0.2000	0.88	200.00
13	0.2667	0.63	65.00
14	0.7333	0.75	120.00
15	0.2000	0.25	12.00
16	0.2667	0.88	250.00
17	0.3333	0.25	18.00
18	1.0000	0.88	450.00
19	0.0333	0.00	10.00
20	0.1333	0.50	30.00



The model will return a price prediction that is the average price of the three neighbors:

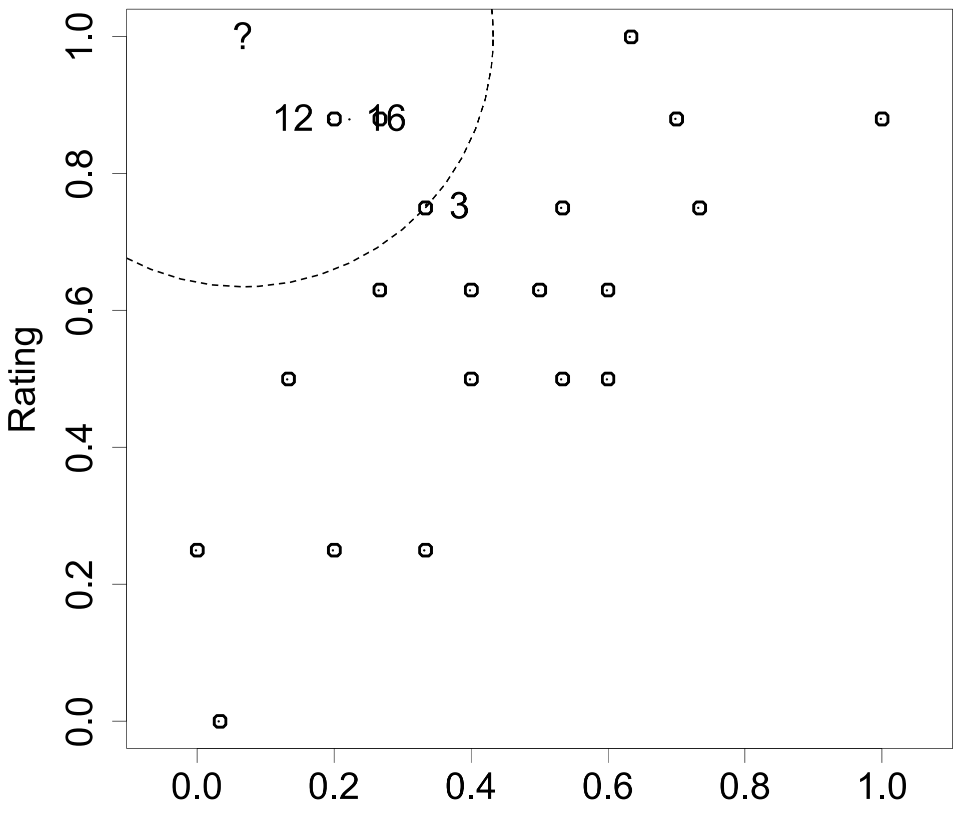
$$Price = \frac{200 + 250 + 55}{3} = 168.33$$

In weighted *k* nearest neighbor classification, we use the weights when calculating the average (since more similar training samples should count for more)

- In a weighted k nearest neighbor the model prediction equation is changed to:

$$\mathbb{M}_k(q) = \frac{\sum_{i=1}^{k} \frac{t_i}{dist(\boldsymbol{q}, \boldsymbol{d_i})^2}}{\sum_{i=1}^{k} \frac{1}{dist(\boldsymbol{q}, \boldsymbol{d_i})^2}}$$

ID	Price	Distance	Weight	Price*Weight
1	30.00	0.7530	1.7638	52.92
2	40.00	0.5017	3.9724	158.90
3	55.00	0.3655	7.4844	411.64
4	550.00	0.6456	2.3996	1319.78
5	35.00	0.6009	2.7692	96.92
6	45.00	0.5731	3.0450	137.03
7	70.00	0.5294	3.5679	249.75
8	85.00	0.7311	1.8711	159.04
9	78.00	0.6520	2.3526	183.50
10	75.00	0.6839	2.1378	160.33
11	500.00	0.5667	3.1142	1557.09
12	200.00	0.1828	29.9376	5987.53
13	65.00	0.4250	5.5363	359.86
14	120.00	0.7120	1.9726	236.71
15	12.00	0.7618	1.7233	20.68
16	250.00	0.2358	17.9775	4494.38
17	18.00	0.7960	1.5783	28.41
18	450.00	0.9417	1.1277	507.48
19	10.00	1.0006	0.9989	9.99
20	30.00	0.5044	3.9301	117.9
		Totals:	*99.2604*	*16249.85*



The model will return a price prediction that is the *weighted* average price of the three neighbors:

$$Price_{wt} = \frac{200 \cdot 29.9376 + 250 \cdot 17.9775 + 55 \cdot 7.4844}{29.9376 + 17.9775 + 7.4844} = 196.60$$

Compare: $Price_{unwt} = \frac{200+250+55}{3} = 168.33$

OTHER SIMILARITY MEASURES

Several similarity measures have been specially defined for binary vectors; one is *Russel-Rao*

- The Russel-Rao similarity of two binary vectors is the ratio between the number of co-presences and the total number of binary features considered.

$$sim_{RR}(\boldsymbol{q}, \boldsymbol{d}) = \frac{CP(\boldsymbol{q}, \boldsymbol{d})}{|\boldsymbol{q}|}$$

- Consider a pair of customer interaction samples, and a new query q:

ID	Profile	FAQ	Help Forum	Newsletter	Liked	Signup
1	1	1	1	0	1	Yes
2	1	0	0	0	0	No
q	*1*	*0*	*1*	*0*	*0*	*???*

Russel-Rao similarity is the ratio of the number of attributes that are positive in both vectors to the total number of attributes

$$sim_{RR}(\boldsymbol{q}, \boldsymbol{d}) = \frac{CP(\boldsymbol{q}, \boldsymbol{d})}{|\boldsymbol{q}|}$$

ID	Profile	FAQ	Help Forum	Newsletter	Liked	Signup
1	1	1	1	0	1	Yes
2	1	0	0	0	0	No
q	*1*	*0*	*1*	*0*	*0*	*???*
	1: co-p	1: ab-p	1: co-p	1: co-a	1: ab-p	

- Consider q and d_1: co-presence = 2, co-absence = 1, presence-absence = 0, absence-presence = 2

$$sim_{RR}(\boldsymbol{q}, \boldsymbol{d_1}) = \frac{CP(\boldsymbol{q}, \boldsymbol{d_1})}{|\boldsymbol{q}|} = \frac{2}{5} = \mathbf{0.4}$$

- Consider q and d_2: co-p = 1, co-a = 3, pr-a = 1, ab-p = 0

$$sim_{RR}(\boldsymbol{q}, \boldsymbol{d_2}) = \frac{CP(\boldsymbol{q}, \boldsymbol{d_2})}{|\boldsymbol{q}|} = \frac{1}{5} = 0.2$$

Sokal-Michener similarity counts the number of both positive and negative "agreements", divided by the total number of attributes

$$sim_{SM}(\boldsymbol{q},\boldsymbol{d}) = \frac{CP(\boldsymbol{q},\boldsymbol{d}) + CA(\boldsymbol{q},\boldsymbol{d})}{|\boldsymbol{q}|}$$

ID	Profile	FAQ	Help Forum	Newsletter	Liked	Signup
1	1	1	1	0	1	Yes
2	1	0	0	0	0	No
q	*1*	*0*	*1*	*0*	*0*	*???*

- Consider q and d_1: co-p = 2, co-a = 1, pr-a = 0, ab-p = 2

$$sim_{SM}(\boldsymbol{q},\boldsymbol{d_1}) = \frac{CP(\boldsymbol{q},\boldsymbol{d_1}) + CA(\boldsymbol{q},\boldsymbol{d_1})}{|\boldsymbol{q}|} = \frac{2+1}{5} = 0.6$$

- Consider q and d_2: co-p = 1, co-a = 3, pr-a = 1, ab-p = 0

$$sim_{RR}(\boldsymbol{q},\boldsymbol{d_2}) = \frac{CP(\boldsymbol{q},\boldsymbol{d_2}) + CA(\boldsymbol{q},\boldsymbol{d_2})}{|\boldsymbol{q}|} = \frac{4}{5} = \mathbf{0.8}$$

Jaccard similarity does not include co-absences in the numerator or the denominator

$$sim_J(\boldsymbol{q},\boldsymbol{d}) = \frac{CP(\boldsymbol{q},\boldsymbol{d})}{CP(\boldsymbol{q},\boldsymbol{d}) + PA(\boldsymbol{q},\boldsymbol{d}) + AP(\boldsymbol{q},\boldsymbol{d})}$$

ID	Profile	FAQ	Help Forum	Newsletter	Liked	Signup
1	1	1	1	0	1	Yes
2	1	0	0	0	0	No
q	*1*	*0*	*1*	*0*	*0*	*???*

- Consider q and d_1: co-p = 2, co-a = 1, pr-a = 0, ab-p = 2

$$sim_J(\boldsymbol{q},\boldsymbol{d_1}) = \frac{CP(\boldsymbol{q},\boldsymbol{d_1})}{CP(\boldsymbol{q},\boldsymbol{d_1}) + PA(\boldsymbol{q},\boldsymbol{d_1}) + AP(\boldsymbol{q},\boldsymbol{d_1})} = \frac{2}{2+0+2} = 0.5$$

- Consider q and d_2: co-p = 1, co-a = 3, pr-a = 1, ab-p = 0

$$sim_J(\boldsymbol{q},\boldsymbol{d_2}) = \frac{CP(\boldsymbol{q},\boldsymbol{d_2})}{CP(\boldsymbol{q},\boldsymbol{d_2}) + PA(\boldsymbol{q},\boldsymbol{d_2}) + AP(\boldsymbol{q},\boldsymbol{d_2})} = \frac{1}{1+1+0} = 0.5$$

Cosine similarity is the cosine of the angle between the two vectors that extend from the origin to each instance; it's insensitive to vector magnitude

$$sim_{cos}(\boldsymbol{q}, \boldsymbol{d}) = \frac{\sum_{i=1}^{m} q_i d_i}{\sqrt{\sum_{i=1}^{m} {q_i}^2}\sqrt{\sum_{i=1}^{m} {d_i}^2}}$$

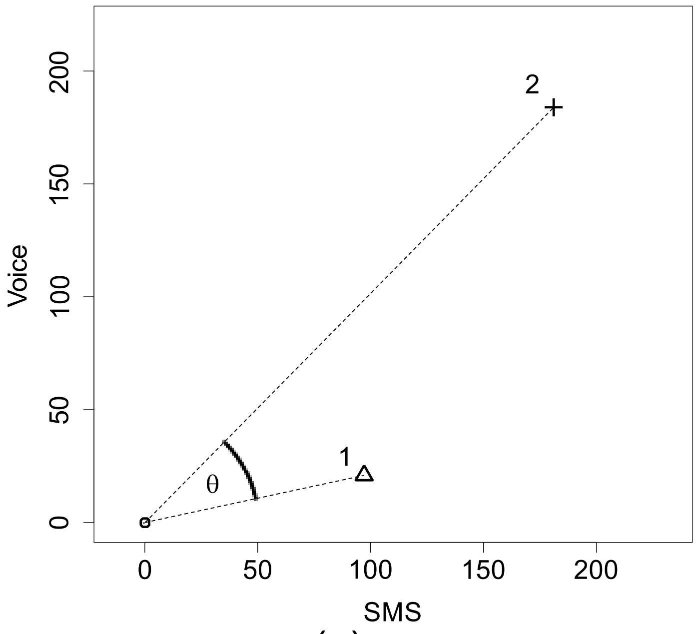
- As an example, $d_1 = \langle 97, 21 \rangle$, $d_2 = \langle 181, 184 \rangle$

$$sim_{cos}(\boldsymbol{q}, \boldsymbol{d}) = \frac{\sum_{i=1}^{m} q_i d_i}{\sqrt{\sum_{i=1}^{m} {q_i}^2}\sqrt{\sum_{i=1}^{m} {d_i}^2}} = \frac{97 \cdot 181 + 21 \cdot 184}{\sqrt{97^2 + 21^2}\sqrt{181^2 + 184^2}} = 0.8362$$

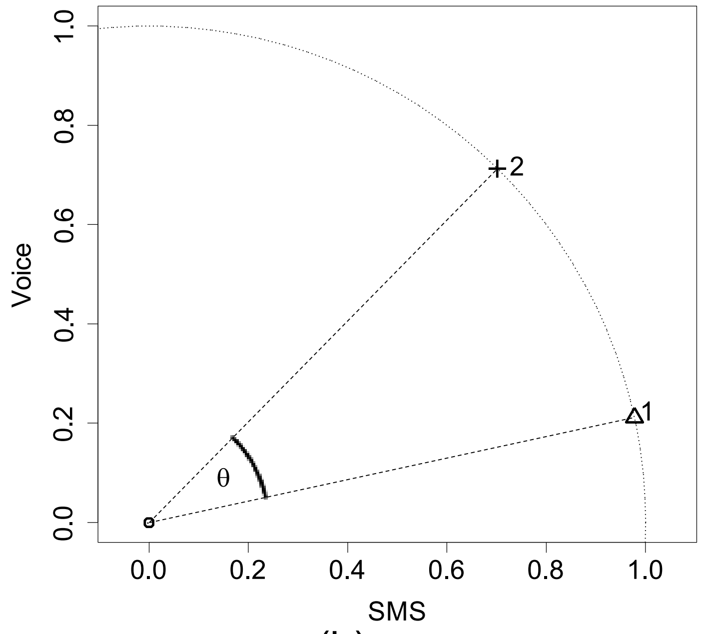
- Note the effect of scaling: $d_1 = \langle 97, 21 \rangle$, $d_3 = \langle 362, 368 \rangle$

$$sim_{cos}(\boldsymbol{q}, \boldsymbol{d}) = \frac{\sum_{i=1}^{m} q_i d_i}{\sqrt{\sum_{i=1}^{m} {q_i}^2}\sqrt{\sum_{i=1}^{m} {d_i}^2}} = \frac{97 \cdot 362 + 21 \cdot 368}{\sqrt{97^2 + 21^2}\sqrt{362^2 + 368^2}} = 0.8362$$

Cosine similarity, graphically



(a)



(b)

(a) The θ represents the inner angle between the vector emanating from the origin to instance $\mathbf{d}_1$ (SMS $= 97$, VOICE $= 21$) and the vector emanating from the origin to instance $\mathbf{d}_2$ (SMS $= 181$, VOICE $= 184$);

(b) shows $\mathbf{d}_1$ and $\mathbf{d}_2$ normalized to the unit circle.

These similarity measures all have their applications; consider the meaning of the dimensions and the problem at hand

- Russel-Rao (binary vectors): $sim_{RR}(\boldsymbol{q}, \boldsymbol{d}) = \frac{CP(\boldsymbol{q},\boldsymbol{d})}{|\boldsymbol{q}|}$
- Sokal-Michener (binary vectors): $sim_{SM}(\boldsymbol{q}, \boldsymbol{d}) = \frac{CP(\boldsymbol{q},\boldsymbol{d})+CA(\boldsymbol{q},\boldsymbol{d})}{|\boldsymbol{q}|}$
- Jaccard (binary vectors): $sim_J(\boldsymbol{q}, \boldsymbol{d}) = \frac{CP(\boldsymbol{q},\boldsymbol{d})}{CP(\boldsymbol{q},\boldsymbol{d})+PA(\boldsymbol{q},\boldsymbol{d})+AP(\boldsymbol{q},\boldsymbol{d})}$
  - *CP*=co-presence, *CA*=co-absence, *PA*=presence-absence, *AP*=absence-presence
- Cosine (any vectors): $sim_{cos}(\boldsymbol{q}, \boldsymbol{d}) = \frac{\sum_{i=1}^{m} q_i d_i}{\sqrt{\sum_{i=1}^{m} {q_i}^2}\sqrt{\sum_{i=1}^{m} {d_i}^2}}$

Let's consider the lunch training set and query from last lecture; we map "Kinda" and Maybe to Y,N

- How does Russel-Rao do?
  - $sim_{RR}(\boldsymbol{q}, \boldsymbol{sand}) = \frac{CP(q,sand)}{|q|} = \frac{3}{9} = \mathbf{0.333}$
  - $sim_{RR}(\boldsymbol{q}, \boldsymbol{burg}) = \frac{CP(q,burg)}{|q|} = \frac{3}{9} = \mathbf{0.333}$
  - $sim_{RR}(\boldsymbol{q}, \boldsymbol{pizz}) = \frac{CP(q,pizz)}{|q|} = \frac{2}{9} = 0.222$
  - $sim_{RR}(\boldsymbol{q}, \boldsymbol{sala}) = \frac{CP(q,sala)}{|q|} = \frac{1}{9} = 0.111$
- Try Sokal-Michener and ignore ambiguous values (Y,N)
  - $sim_{SM*}(\boldsymbol{q}, \boldsymbol{sand}) = \frac{CP(q*,sand)+CA(q*,sand)}{|q*|} = \frac{1+2}{5} = \mathbf{0.6}$
  - $sim_{RR*}(\boldsymbol{q}, \boldsymbol{burg}) = \frac{CP(q*,burg)+CA(q*,burg)}{|q*|} = \frac{0+2}{4} = 0.5$
  - $sim_{RR*}(\boldsymbol{q}, \boldsymbol{pizz}) = \frac{CP(q*,pizz)+CA(q*,pizz)}{|q*|} = \frac{0+3}{6} = 0.5$
  - $sim_{RR*}(\boldsymbol{q}, \boldsymbol{sala}) = \frac{CP(q*,sala)}{|q*|} = \frac{1+1}{5} = 0.4$

	Sandwich	Burger	Pizza	Salad	*Query*
Bread	Y	Y,N	Y,N	N	*Y*
Meat	Y	N	Y,N	Y,N	*N*
Cheese	Y,N	Y,N	Y	Y,N	*N*
Lettuce	Y,N	Y,N	N	Y	*N*
Tomato	Y,N	Y,N	Y,N	Y,N	*Y*
Cucumber	N	N	N	Y,N	*N*
Mayo	Y,N	Y,N	N	N	*Y*
Meat Patty	N	Y	N	N	*N*
Temperature	Cold	Hot	Hot	Cold	*Cold*

```python
"""
nearestNeighbor.py, Created on Thu Feb 20 17:41:33 2020
@author: crjones4
"""

from sklearn import neighbors
import pandas as pd
import numpy.linalg as linalg
import numpy as np

def distance(a, b):
    # Euclidean
    return linalg.norm(a-b)

pathName = "C:\\Data\\"
dataFrame = pd.read_excel(pathName + 'DraftData.xlsx', sheet_name='rawData')
X = dataFrame.drop(["ID", "Draft"], axis=1)
y = dataFrame.Draft
newX = [5, 4]           # define new data as a list, can be an array as well

minDist = 100000
count = 0
for row in X.iterrows():        # NOTE! this is slow and only for use on small ADS
    samp = row[1]
    dist = distance(newX, samp)
    if (dist < minDist):
        minDist = dist
        minRow = row
        minTarget = y[count]
    count = count + 1
print("Min at", minRow)
print(minDist, minTarget)

# Now solve the problem using the enarest neignbor class
clf = neighbors.KNeighborsClassifier(n_neighbors = 4, weights='uniform')
clf.fit(X,y)
result = clf.predict(np.asarray(newX).reshape(1,-1))    # to suit input format
print("Using scikit class: ", result)
```

```
('Min at', (9L, Speed      4.25
Agility    3.75
Name: 9, dtype: float64))
(0.7905694150420949, u'No')
('Using scikit class: ', array([u'No'], dtype=object))
```

Today's Objectives

- Classification
- Handling Noisy Data
  - *K*-nearest neighbor
  - Weighted nearest neighbor
- Data Normalization
- Continuous Targets using kNN
- Other Similarity Measures
  - Binary similarity measures
  - Cosine similarity