Venice, Italy

There is always light. If only we're brave enough to see it. If only we're brave enough to be it. – Amanda Gorman

KNN and some other ideas

What we will cover

kNN

MDS

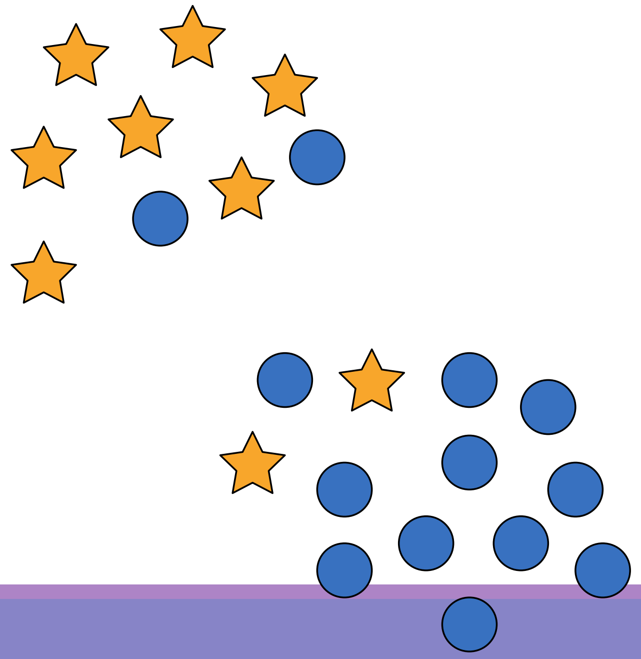
Curse of dimensionality

Model Ensemble

K-Nearest Neighbor

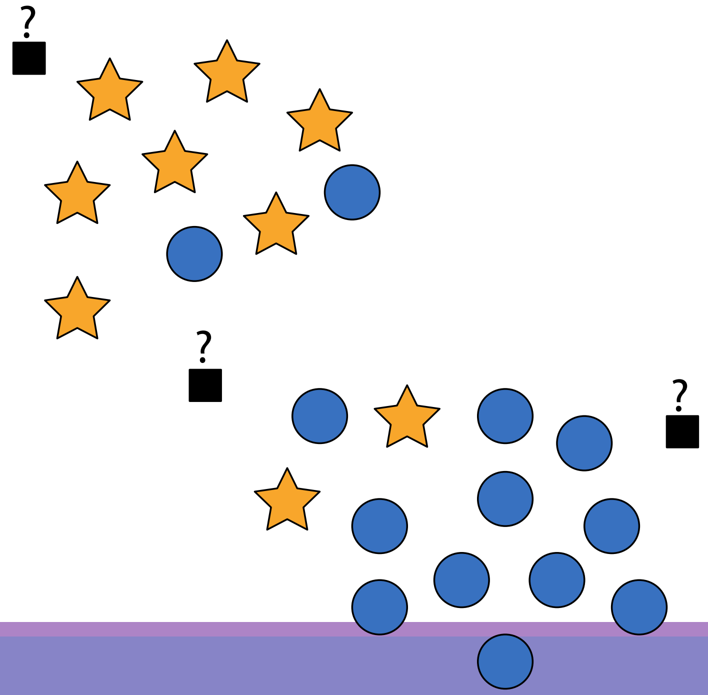
Intuition

- Identify several cases that are most similar to a given observation.
- Use the information from those ‘neighbors’ to classify/predict the new observation.



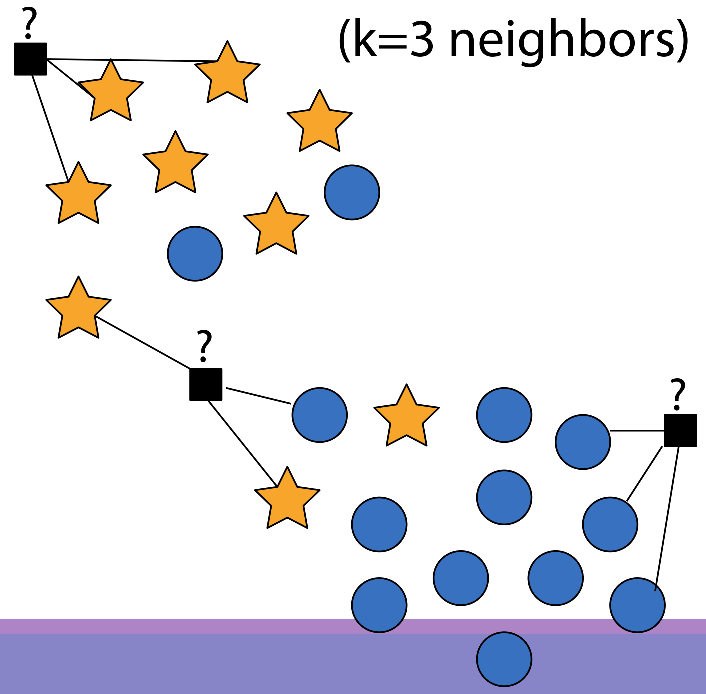
Intuition

- Identify several cases that are most like a given observation.
- Use the information from those 'neighbors' to classify/predict the new observation.



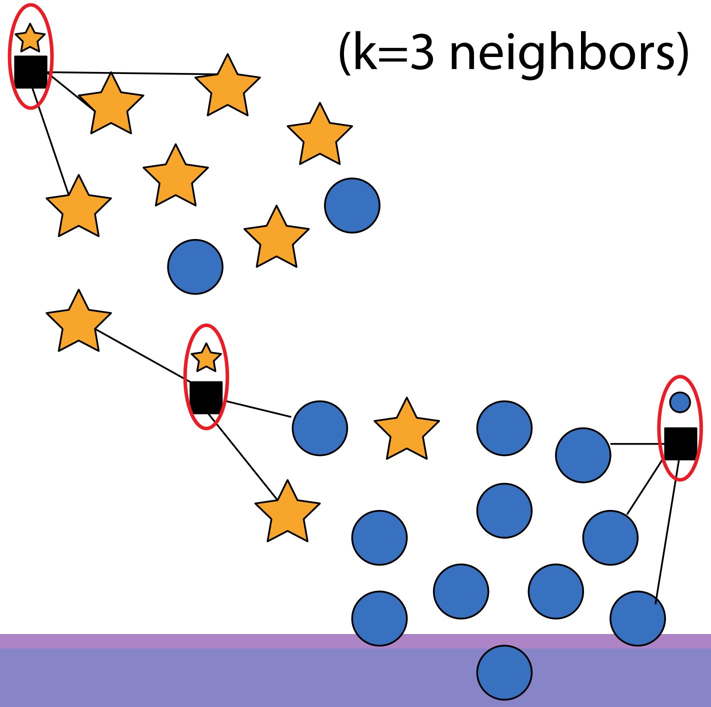
Intuition

- Identify several cases that are most like a given observation.
- Use the information from those ‘neighbors’ to classify/predict the new observation.



Intuition

- Identify several cases that are most like a given observation.
- Use the information from those ‘neighbors’ to classify/predict the new observation.



Considerations

- How should I measure nearness?
  - Numeric Attributes?
  - Ordinal Attributes?
  - Categorical Attributes?
  - How do I combine these?
- How should I combine the results of neighbors?
  - Classification:
    - Majority rules?
    - Weight votes by nearness?
  - Prediction:
    - Mean?
    - Median?
- How many neighbors should I use?

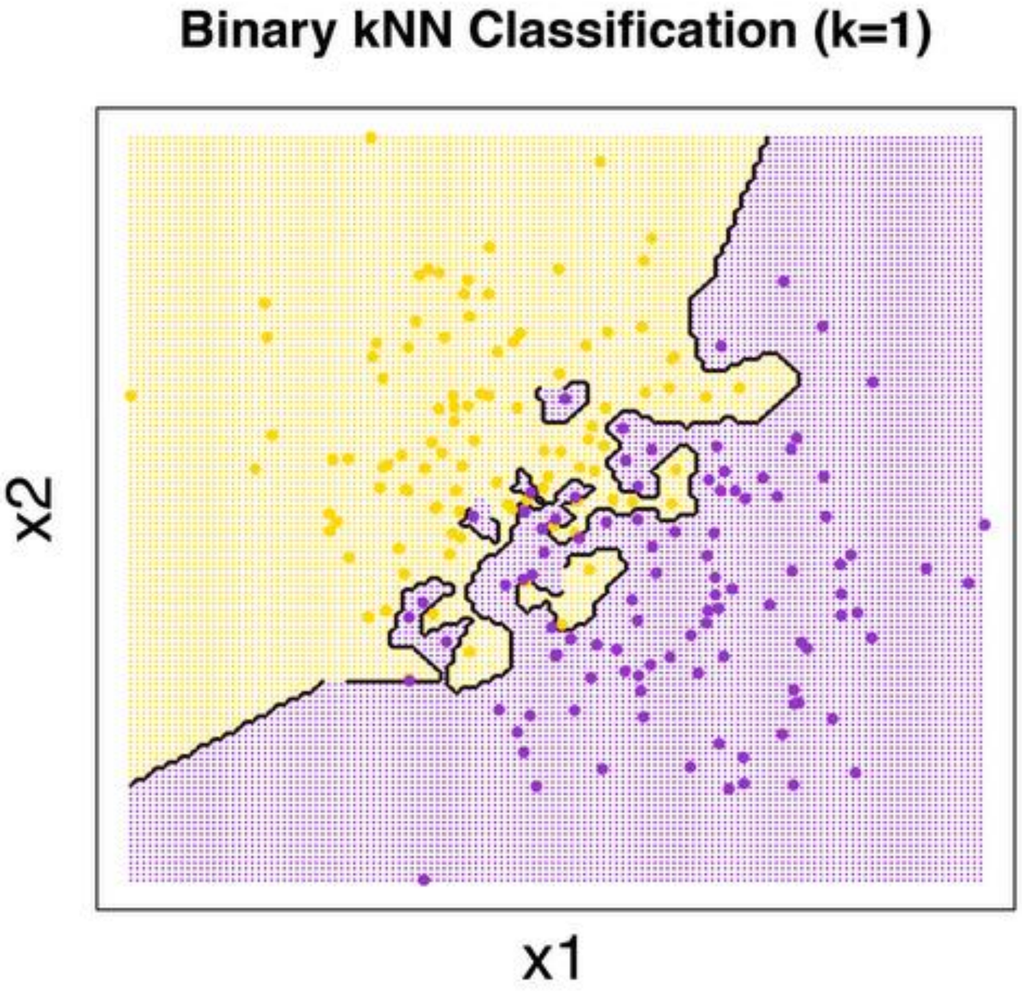
Considerations

- The methodology is simple but FLEXIBLE for creativity

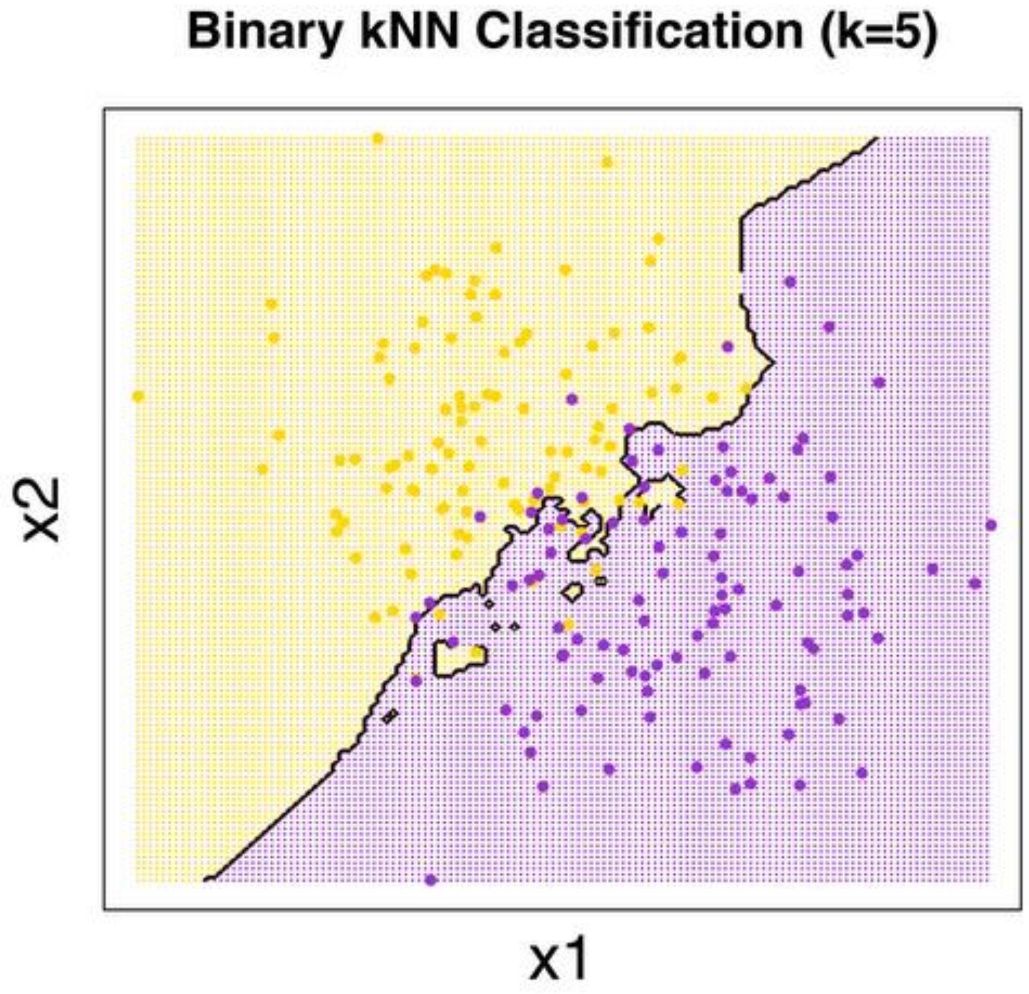
- Using a built in kNN function in R, Python or SAS will save time but lose flexibility.

- Flexible, so you can create your own distance matrices (use best “distance” for a given situation).

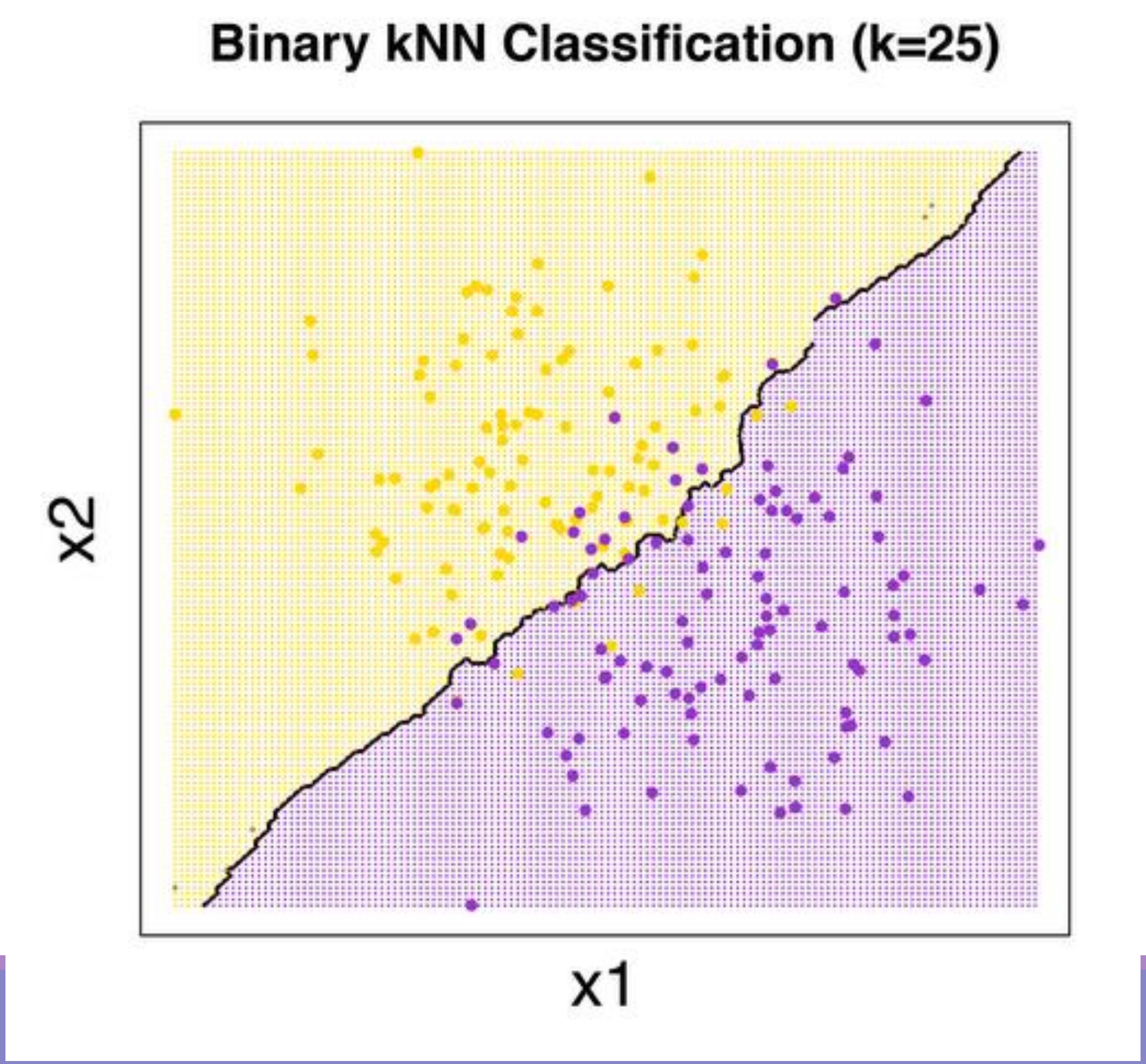
Choosing k



Choosing k



Choosing k



Binary kNN Classification (k=25)

Choosing k

- Smaller values of k => higher variance model
  - (tends toward overfitting)
- Larger values of k => higher bias model
  - (tends toward underfitting)

- Common practice to begin with $k = \sqrt{n}$ where n is the number of training examples
- Best practice to tune this parameter with a validation set or with cross-validation.

Advantages of kNN

- Easy to explain, intuitive, understandable
- Applicable to any type of data
- Makes no assumptions about the underlying distribution of the data.
- Large/representative training set is only assumption

Disadvantages of kNN

- **Computationally expensive in classification phase**
- **Requires storage for the training set**
  - (The training set IS the model!)
- Results dependent on choice of distance function, combination function, and number of neighbors, *k*.
- Susceptible to noise
- Require lots of data preprocessing and consideration for distance metrics
- Does not produce a model. Does not help us understand how the features are related to the classes.

Example

BREAST CANCER DATA (FROM AN EARLIER CLASS)

```
train.x=subset(train,select=-Target)
train.y=as.factor(train$Target)

test.x=subset(test,select=-Target)
test.y=as.factor(test$Target)
```

Need to create a matrix for the predictors and a vector for the response! Response should be a factor!!

```
predict.test=knn(train.x,test.x,train.y,k=3)
 > head(predict.test)
 [1] 1 0 0 0 0 1
```

```
sum(predict.test != test.y)/175
[1] 0.05714286
```

```
train.x=subset(train,select=-Target)
train.y=as.factor(train$Target)

test.x=subset(test,select=-Target)
test.y=as.factor(test$Target)
```

Need to create a matrix for the predictors and a vector for the response! Response should be a factor!!

```
> head(predict.test)
[1] 1 0 0 0 0 1
```

BUT WAS THIS THE CORRECT K?

```
sum(predict.test != test.y)/175
[1] 0.05714286
```

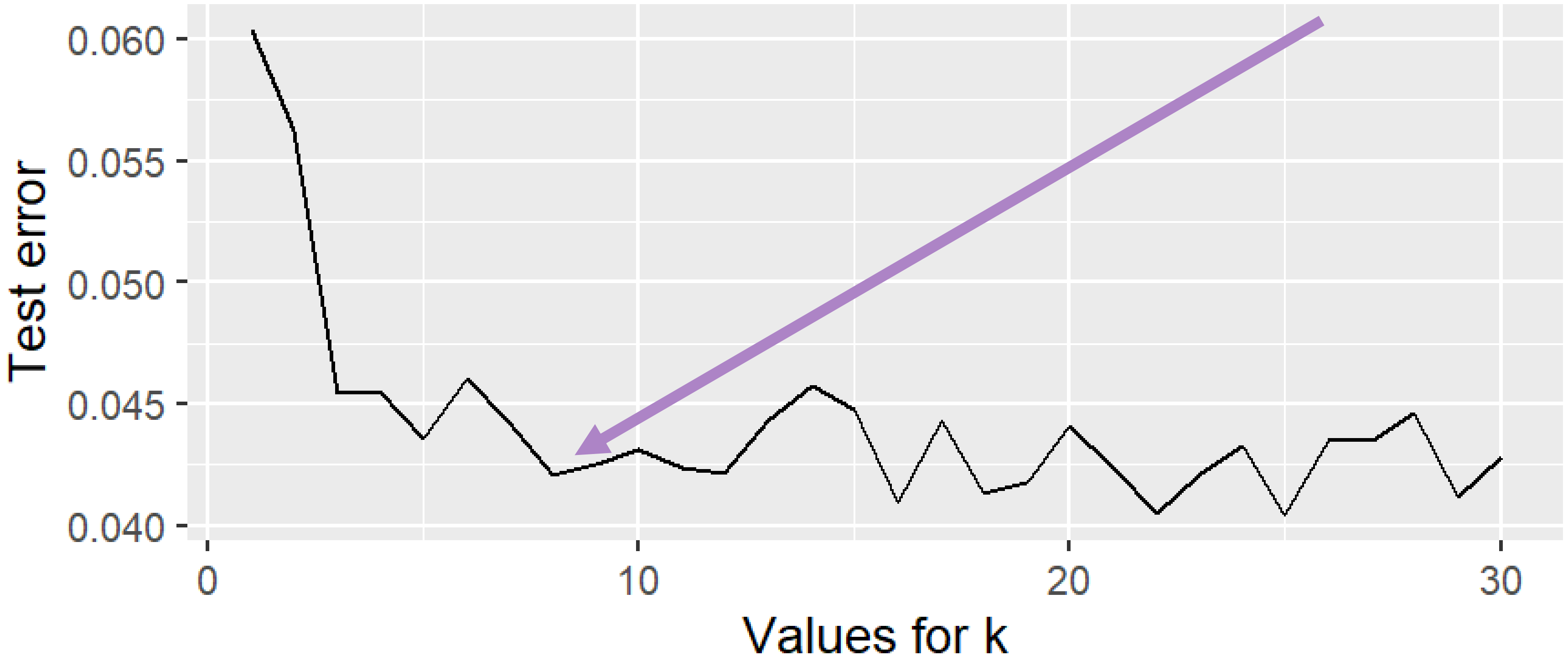
```
k.attempts=seq(1,30)
pred.error=vector(length=length(k.attempts))
temp.val=vector(length=100)

for (i in 1:length(k.attempts))
{for (j in 1:length(temp.val))
  {perm=sample(1:699)
  BC_randomOrder=BCdata[perm,]
  train = BC_randomOrder[1:floor(0.75*699),-c(1,7)]
  test = BC_randomOrder[(floor(0.75*699)+1):699,-c(1,7)]
  train.x=subset(train,select=-Target)
  train.y=as.factor(train$Target)
  test.x=subset(test,select=-Target)
  test.y=as.factor(test$Target)
  predict.test=knn(train.x,test.x,train.y,k=i)
  temp.val[j]=sum(predict.test != test.y)/175}

pred.error[i]=mean(temp.val)}

all.dat=data.frame(cbind(k.attempts,pred.error))
ggplot(all.dat,aes(x=k.attempts,y=pred.error))+geom_line()+labs(x="Values for k",y="Test error")
```

k=9 (need odd number)



Building Distance Measures (self-study)

K-NN

Building Distance Functions

- Numeric Variables (Includes some ordinal):
  - Some type of normalization or standardization is usually required
    - Standardize the variable before input to the method
  - Most common types of standardization:
    - min/max normalization (feature scaling) $\frac{x-x_{min}}{x_{max}-x_{min}}$
    - z-score standardization $\frac{x-\bar{x}}{\sigma_x}$

Building Distance Functions

- Categorical Variables (Includes some ordinal):
  - Simple Matching Distance = 0 if matching, 1 otherwise

Name	Marital Status
Sam	Single
Pam	Married
Tam	Single

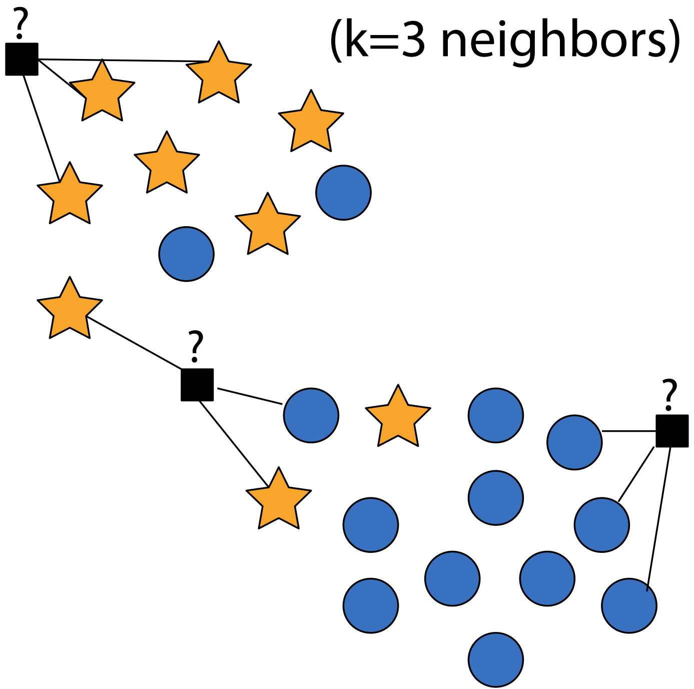
Original Variable

$$\begin{array}{cc} & \begin{array}{ccc} S & P & T \end{array} \\ \begin{array}{c} S \\ P \\ T \end{array} & \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$$

Distance Matrix

Combination Functions

- Now we have distances to each of *k* neighbors
- How do I combine that information to make a prediction for the given observation?



Combination Functions

- Numeric Target
  - Mean or Median of the neighbors’ target value
- Class Target
  - Basic approach: democracy – majority rules
  - Create probabilities for each class as the proportion of neighbors voting for each class.
  - Weighted voting: nearer neighbors have stronger votes
    - This can reduce the sensitivity to the parameter *k*

$$w_j = \frac{1}{d(i,j)^2}$$

    - Add up the weighted votes to see which category has the most

MDS versus PCA

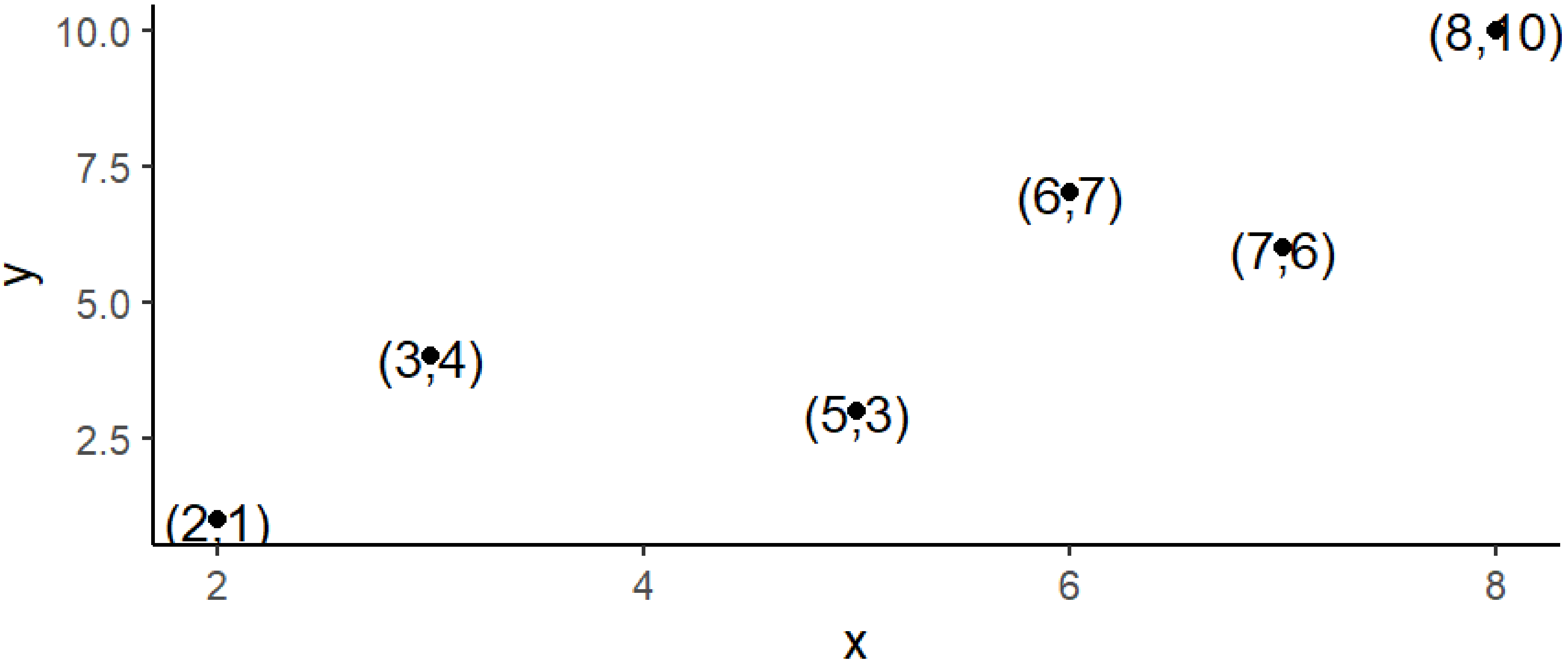
MULTIDIMENSIONAL SCALING

Small review on PCA

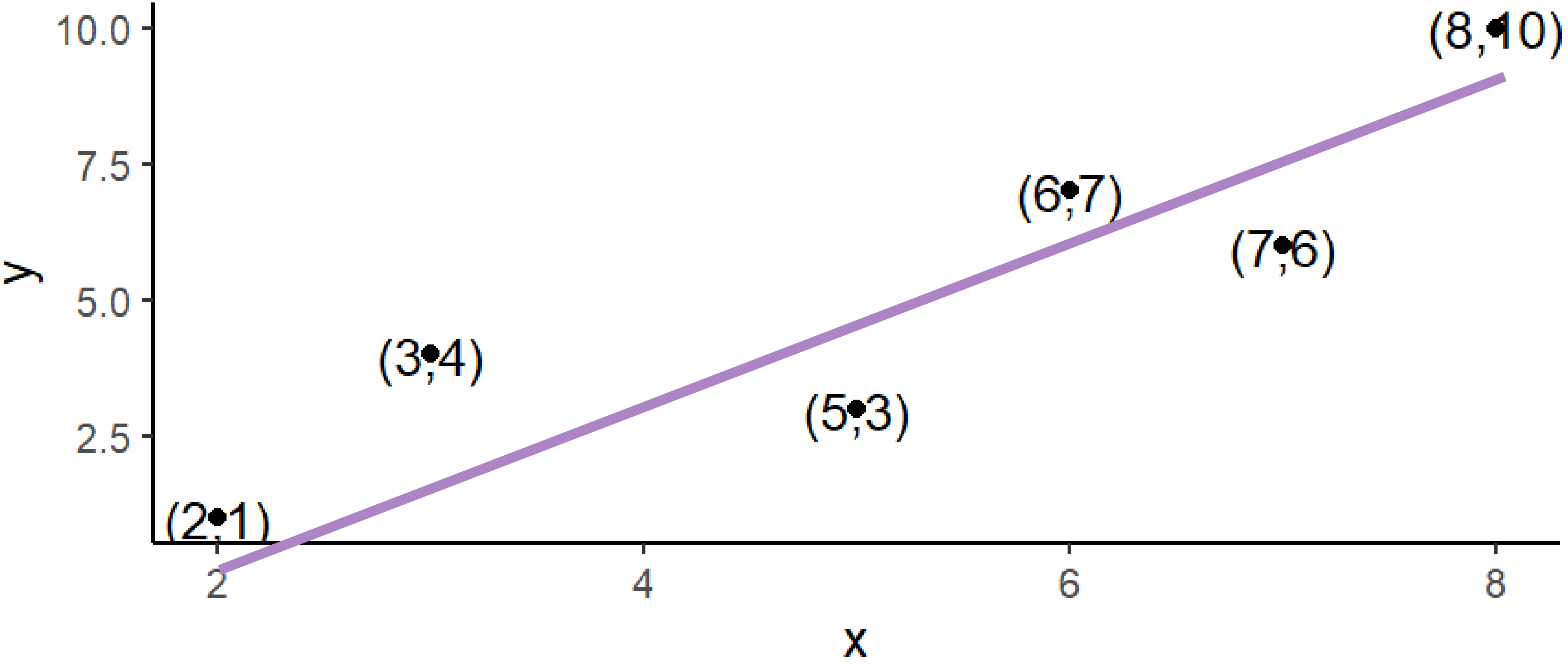
PCA is a data reduction technique

It is useful to reduce the number of dimensions

When you perform a PCA, it creates d principle component vectors (d is the number of variables you have)….then you can choose k (how many vectors to keep)



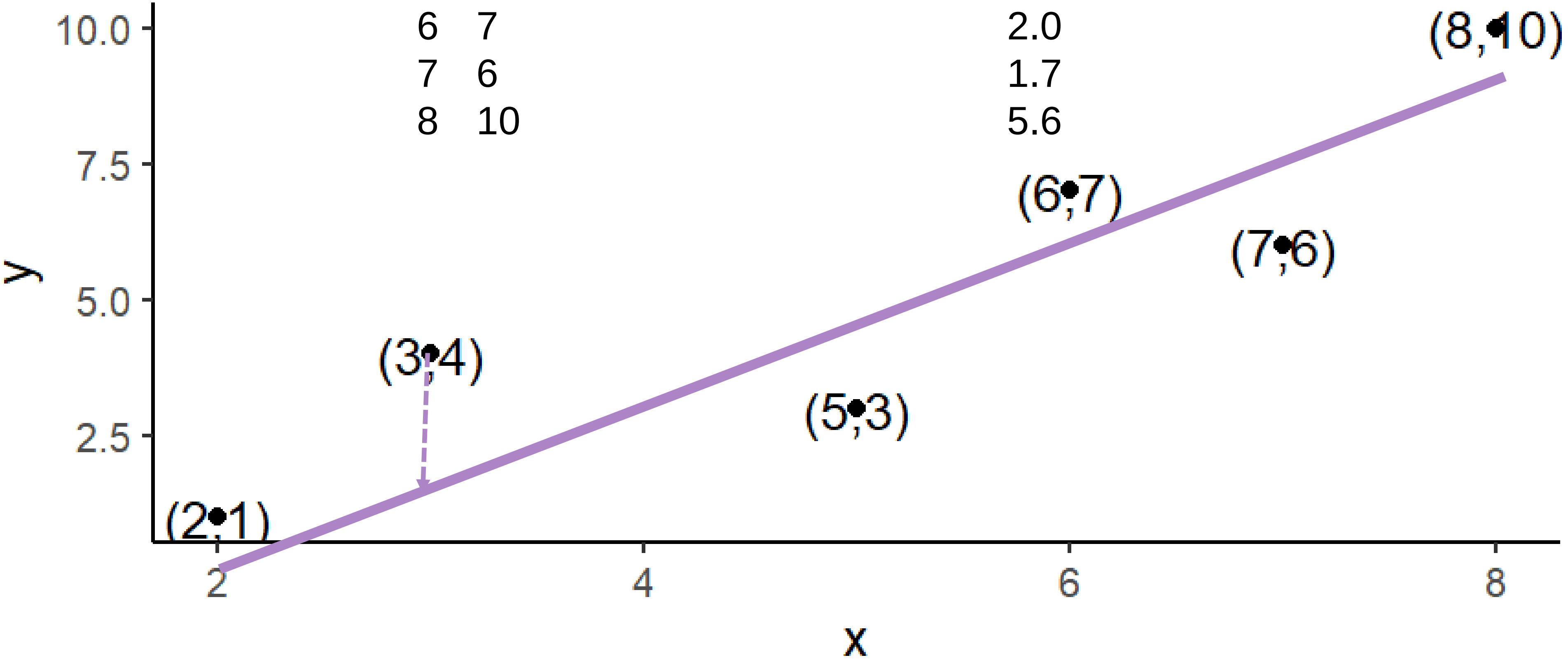
X	Y
2	1
3	4
5	3
6	7
7	6
8	10



X	Y
2	1
3	4
5	3
6	7
7	6
8	10

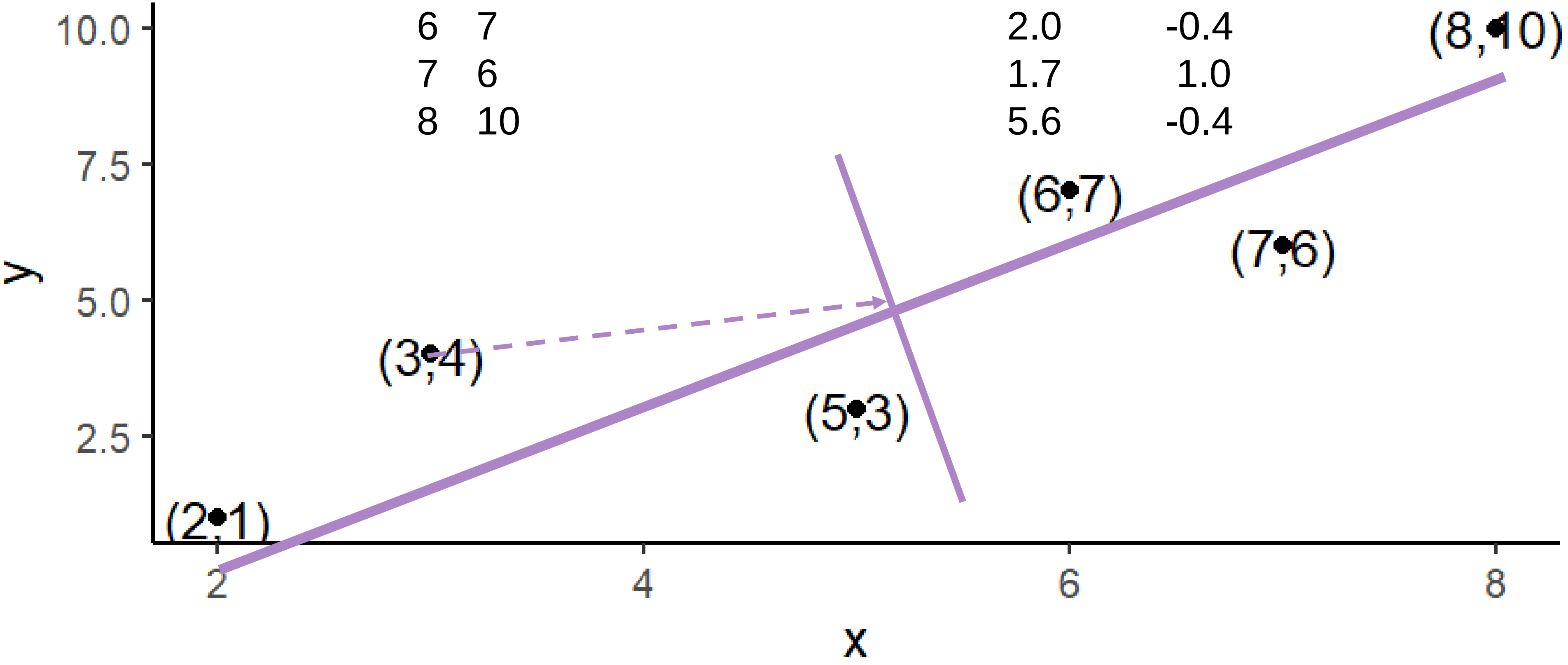
X	Y
2	1
3	4
5	3
6	7
7	6
8	10

PCA1
-5.2
-2.2
-1.9
2.0
1.7
5.6



X	Y
2	1
3	4
5	3
6	7
7	6
8	10

PCA1	PCA2
-5.2	-0.2
-2.2	-1.1
-1.9	1.1
2.0	-0.4
1.7	1.0
5.6	-0.4



New data under PCA

- Each “variable” (i.e. Principle Component) is independent of the others (we say orthogonal)
- Can see how much variance is explained by each variable (calculate the percent of variance explained by looking at the percent within each variable)
  - For example, the “sdev” for this little example is: 3.8416363 0.8804339. Which means the variance explained by each component is: 14.7581694 0.7751639. And the sum of the two variances is: 15.53333 (14.7581694 + 0.7751639). To calculate the percent of variation explained within each component, take each variance divided by the sum:

    0.95009695 0.04990327 (or cumulative variance is 0.95009695 1).

- When trying to choose how many PC to use, look at the variance explained. Would like to have a good representation of the data.

MDS

MDS is similar to PCA in that it can be used to visualize data in a lower dimension (either 2 or 3 dimensions)

To perform MDS, you need to give the algorithm a dissimilarity matrix (or distance matrix)

There are two different types of MDS

- Classical MDS - this will preserve the original distance between points (this is also referred to as metric MDS and is very similar to PCA). In R, use cmdscale(). Think about projecting onto 2-dimensional.
- Non-metric MDS – (ordinal MDS) constructs fitted distances that are in the same rank order as the original distance (can be used with qualitative data as well as quantitative data). In R, use isoMDS() in MASS package. Think about “squashing picture onto 2-dimensional).

What is the difference between PCA and MDS…PCA is more focused on the dimensions themselves (wants to maximize explained variance) where MDS is more focused on relations among the scaled objects

When to use MDS and PCA?

To visualize data, you can use either, but since MDS works to keep the same relationship among the distances, MDS is usually preferred.

If you will use the data for any analysis (for example, clustering, regression, etc), then PCA should be done.

Great reference for more info on MDS:
https://stat.ethz.ch/education/semesters/ss2013/ams/slides/v4.1.pdf

Example

NCI60 DATA SET

National Cancer Institute (NCI)

The NCI60 contains 60 cell lines of different cancer types and a couple variants to bring the number of rows up to 64 (instead of 60) in a microarray (with 6,830 gene expressions)

Used to screen potential treatments of various cancer types

Blurb from NCI:

“**The NCI-60 Human Tumor Cell Lines Screen** has served the global cancer research community for >20 years. The screen was implemented in fully operational form in 1990 and utilizes 60 different human tumor cell lines to identify and characterize novel compounds with growth inhibition or killing of tumor cell lines. It is designed to screen up to 3,000 small molecules (synthetic or purified natural products) per year for potential anticancer activity. The operation of this screen utilizes 60 different human tumor cell lines, representing leukemia, melanoma and cancers of the lung, colon, brain, ovary, breast, prostate, and kidney cancers.”

```
ex_MDS= NCI60$data
pca_ex=prcomp(ex_MDS,scale=T)
```

NOTE: Percent of variance explained from first two dimensions is only about 18%

```
d=dist(ex_MDS)
mds_ex=cmdscale(d,eig=TRUE, k=2)
```

```
$GOF
[1] 0.2319364 0.2319364
```

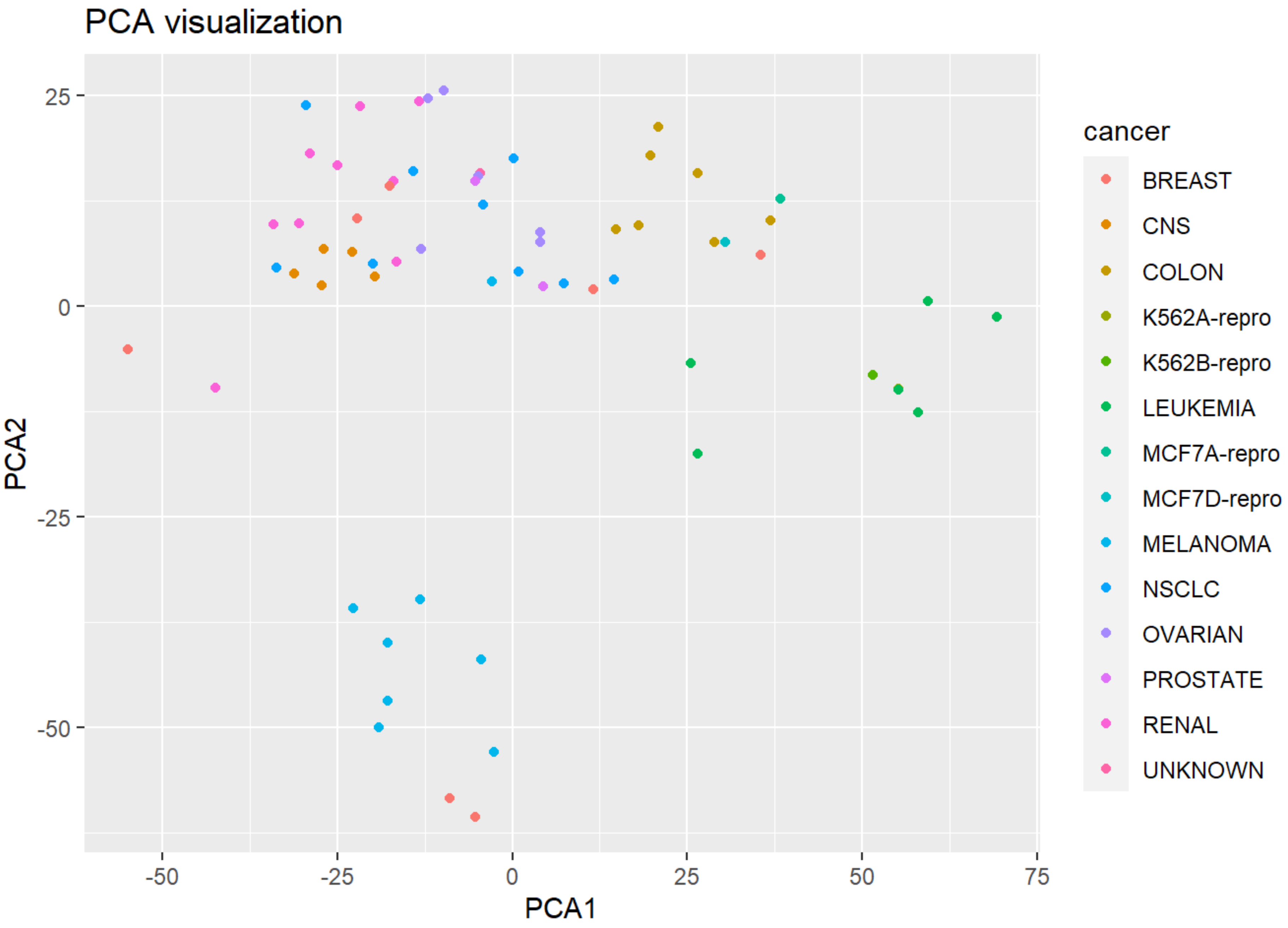
Want this number high (best to have higher than 0.8!!!). This data is NOT going to be represented well in two dimensions!

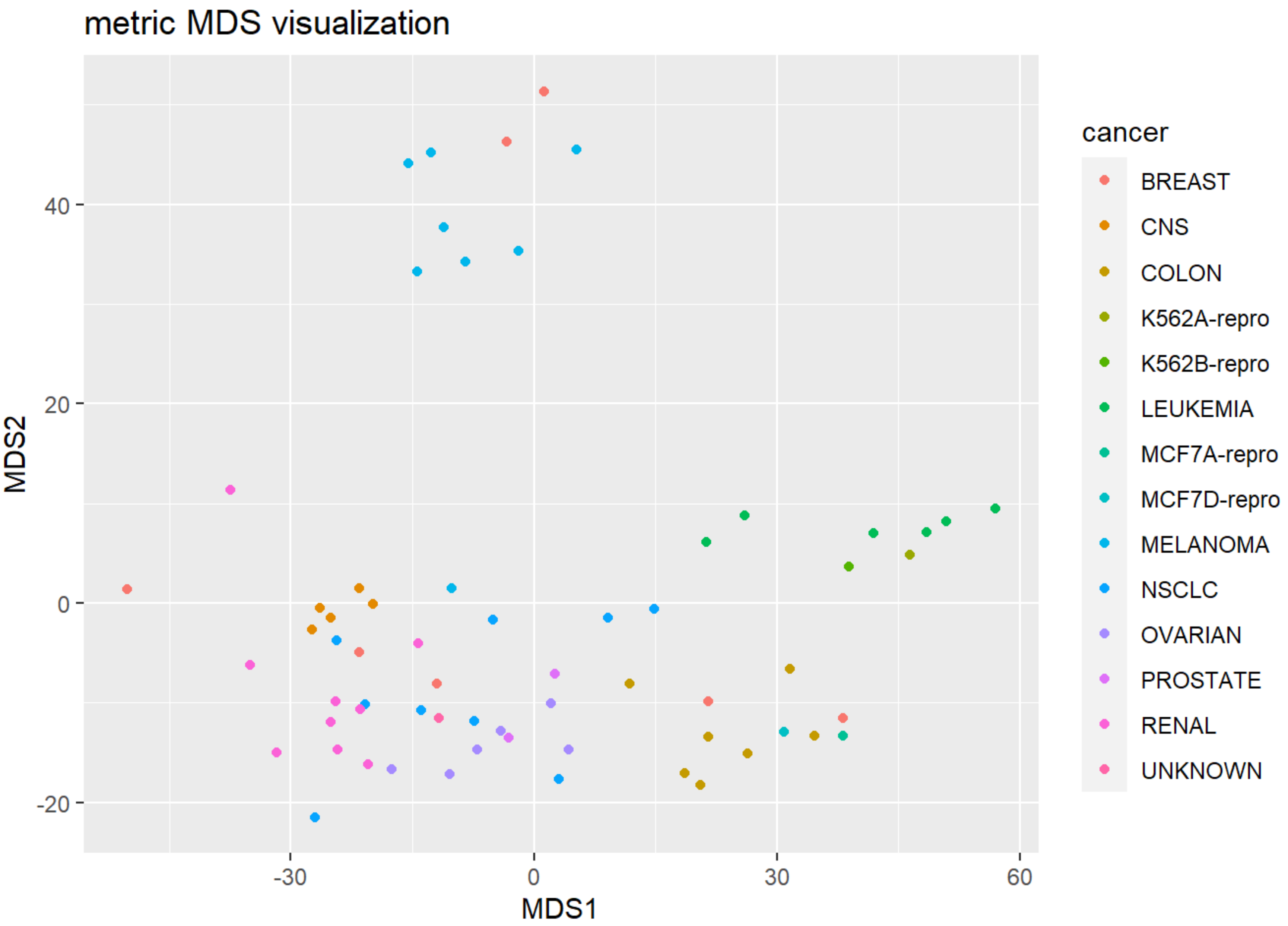
```
mds_ex=isoMDS(d, k=2)
```

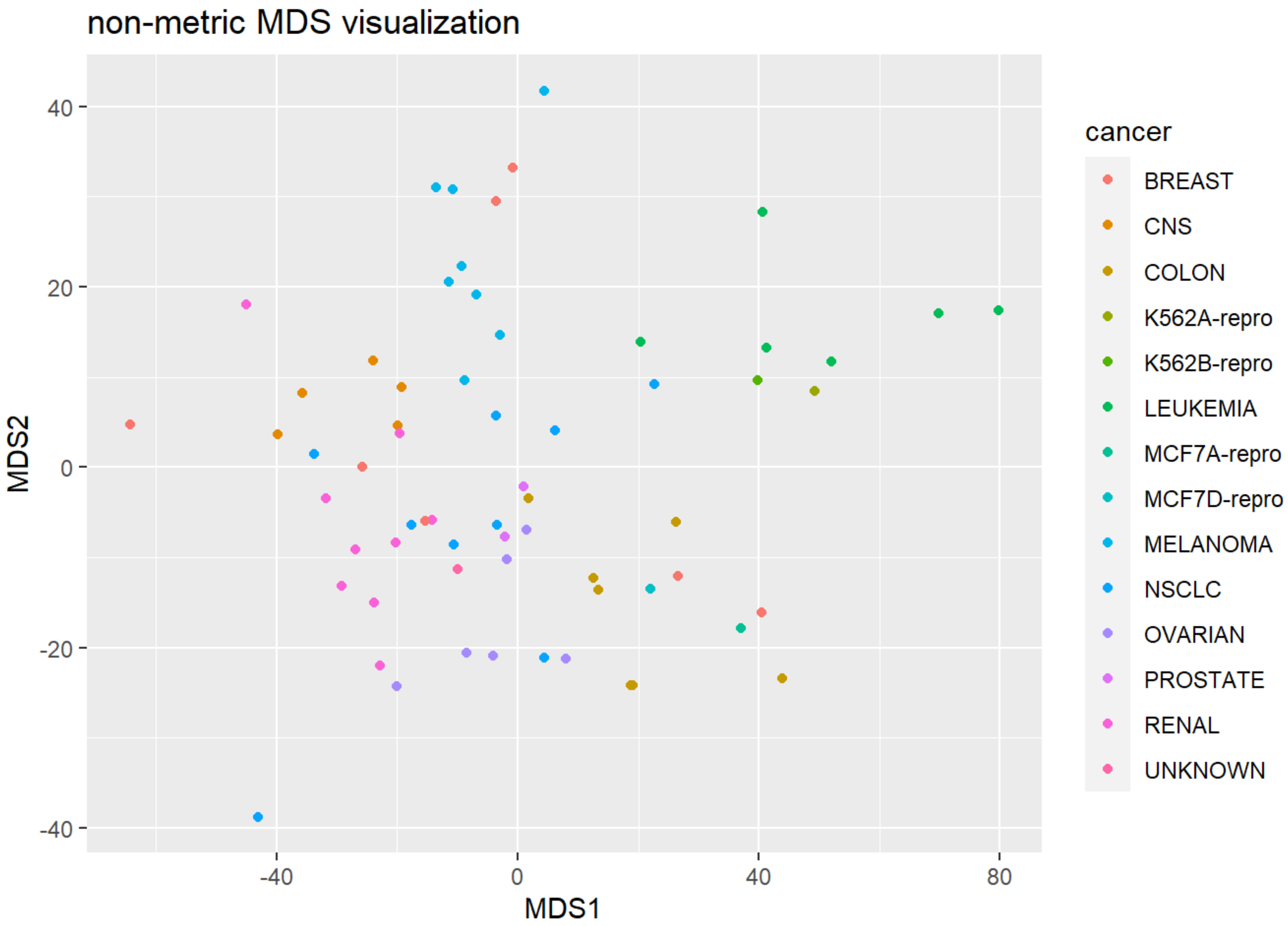
```
$stress
[1] 20.75056
```

Percent (between 0 and 100)…smaller is better..would like less than 20% (less than 10% would be ideal)!

7 of the Melanoma and 2 of the Breast Cancer act very differently in this microarray




metric MDS visualization
MDS2
40
20
0
-20
-30
0
30
60
MDS1
cancer
BREAST
CNS
COLON
K562A-repro
K562B-repro
LEUKEMIA
MCF7A-repro
MCF7D-repro
MELANOMA
NSCLC
OVARIAN
PROSTATE
RENAL
UNKNOWN


non-metric MDS visualization
MDS2
40
20
0
-20
-40
-40
0
40
80
MDS1
cancer
BREAST
CNS
COLON
K562A-repro
K562B-repro
LEUKEMIA
MCF7A-repro
MCF7D-repro
MELANOMA
NSCLC
OVARIAN
PROSTATE
RENAL
UNKNOWN

Curse of dimensionality

Curse of dimensionality

When we have a HUGE number of predictors, finding the true signal becomes difficult

Can be hidden in all of the dimensions (in training, it could look like model is getting better, but in reality, we are just adding noise)

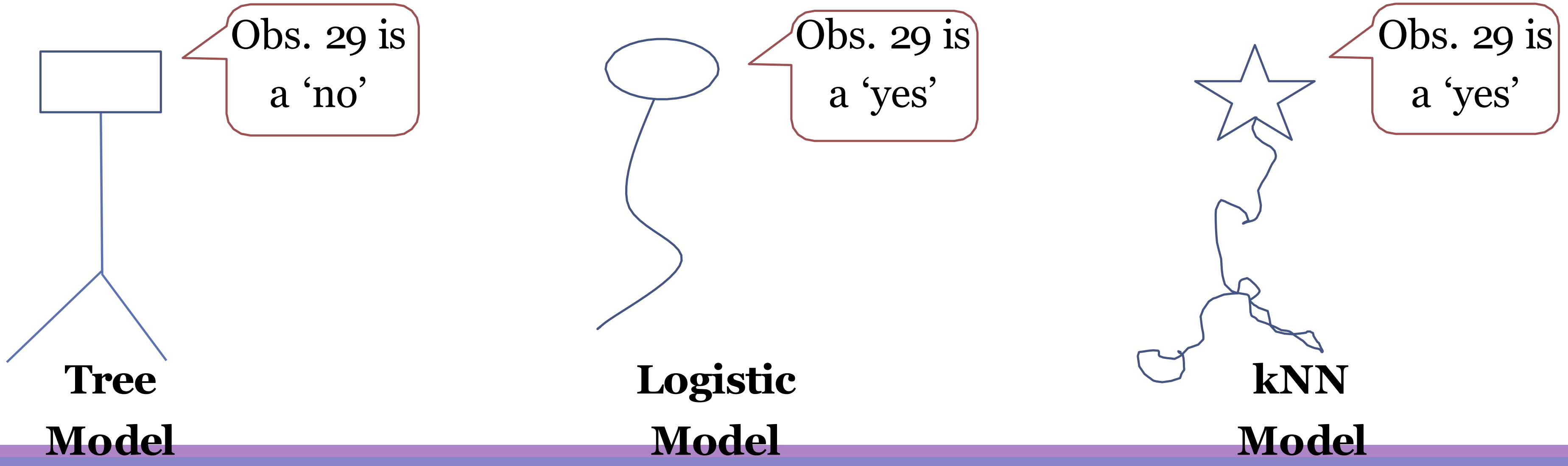
Need to be cautious when you have large p!!

Dimension reduction, removing redundancies, and/or penalized methods are important

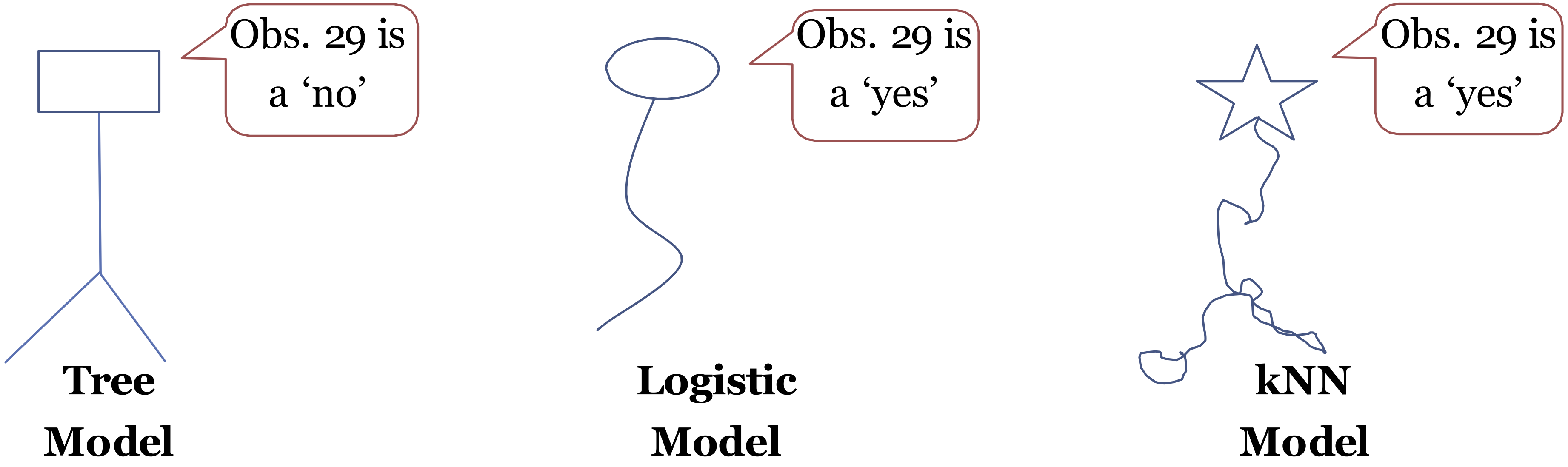
Ensemble Models

Simple Voting Ensemble

- Create a decision tree, a logistic model, and a kNN model.
- All have misclassification rate ~0.20-0.25
- If they each misclassify *different* observations…
- They may be more accurate in ensemble.



Proportion Voting Ensemble



Obs. 29 is a ‘no’

Obs. 29 is a ‘yes’

Obs. 29 is a ‘yes’

Tree Model

Logistic Model

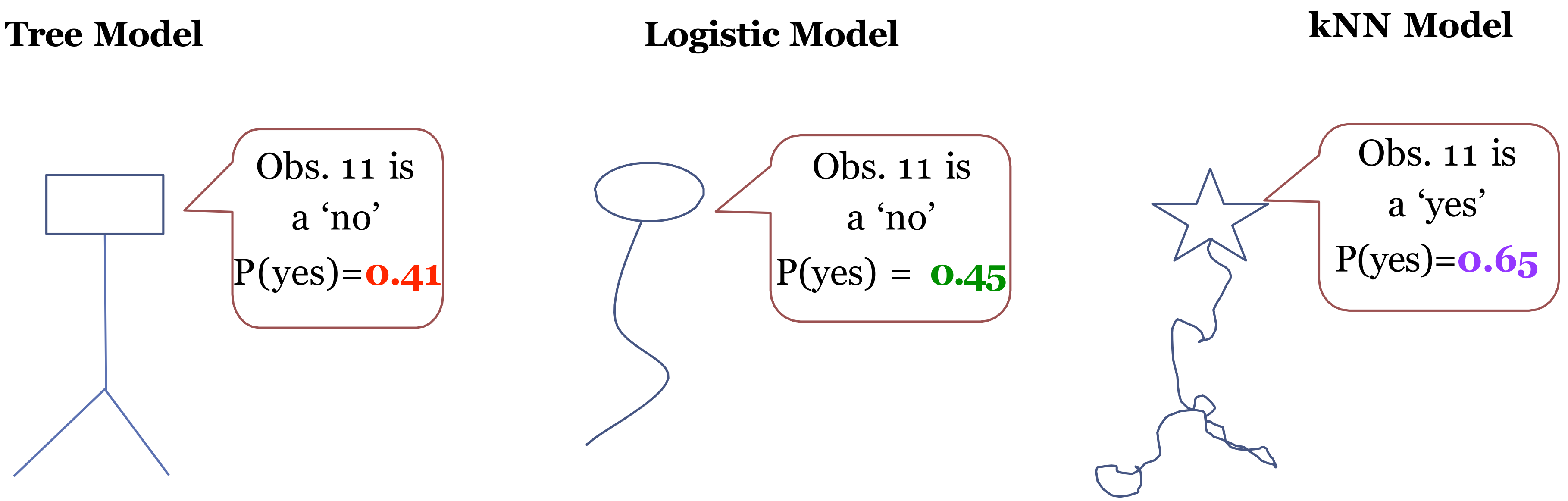
kNN Model

- An **Ensemble model** will use the input from all 3!
- Since 2 of 3 models in this ensemble say ‘Yes’, we’ll **predict observation 29 as a ‘yes’ with probability** $\frac{2}{3}$.

Average Ensembles

Can also consider averaging *every* model’s predicted probability for the event:

- P(‘no’) = (1/3)(0.59+0.55+0.35) = 0.496
- **P(‘yes’) = (1/3)(0.41+0.45+0.65) = 0.503**



Tree Model

Obs. 11 is a ‘no’ P(yes)=**0.41**

Logistic Model

Obs. 11 is a ‘no’ P(yes) = **0.45**

kNN Model

Obs. 11 is a ‘yes’ P(yes)=**0.65**

Questions