CS 412

JAN 20^{TH} – NEAREST NEIGHBORS

$\delta = 0.05$

Error Bounds

PAC

To be within ϵ with 95% probability requires *N* flips:

$P(|\hat{\theta} - \theta^*| < \epsilon) < \delta$

ϵ = .2	→	N = 10
ϵ = .1	→	N = 38
ϵ = .05	→	N = 149
ϵ = .03	→	N = 414
ϵ = .01	→	N = 931

Summary

Probability important for:

- Estimating prediction uncertainty
- Analyzing random samples

Many important concepts:

- Random variables
- Marginalization
- Approximation bounds

Data Separation

Before your model construction begins,
you need to separate your data into at least two sets

- Training
- Testing
- *Validation (which we'll discuss later)*

You should not conduct any analysis of the data before separating it

- This is called *data snooping* and can result in an inaccurate estimate of our final reported error

The only way we can estimate the final reported error of our model
is by judging on the test set

Data Separation

The training set is used to train the data and is usually a smaller section of the data as a whole (much less than half)

By not analyzing or training on a large set of the data, what is the advantage?

Data Separation

The training set is used to train the data and is usually a smaller section of the data as a whole (much less than half)

By not analyzing or training on a large set of the data, what is the advantage?

When we’ve created our “choice” model, we can run it on the test data *only once* and get an estimate of the error without having to induce a penalty

Approach

A bank creates an investment ML algorithm that forcasts whether a certain bond value will go up or down. As input, it receives a group of financial features that have all been standardized for mean and variance. The bank separated the data into two groups, training and testing. After building the model on the training data, the model trained on the testing data can correctly determine whether the bond will go up or down with 52% accuracy.

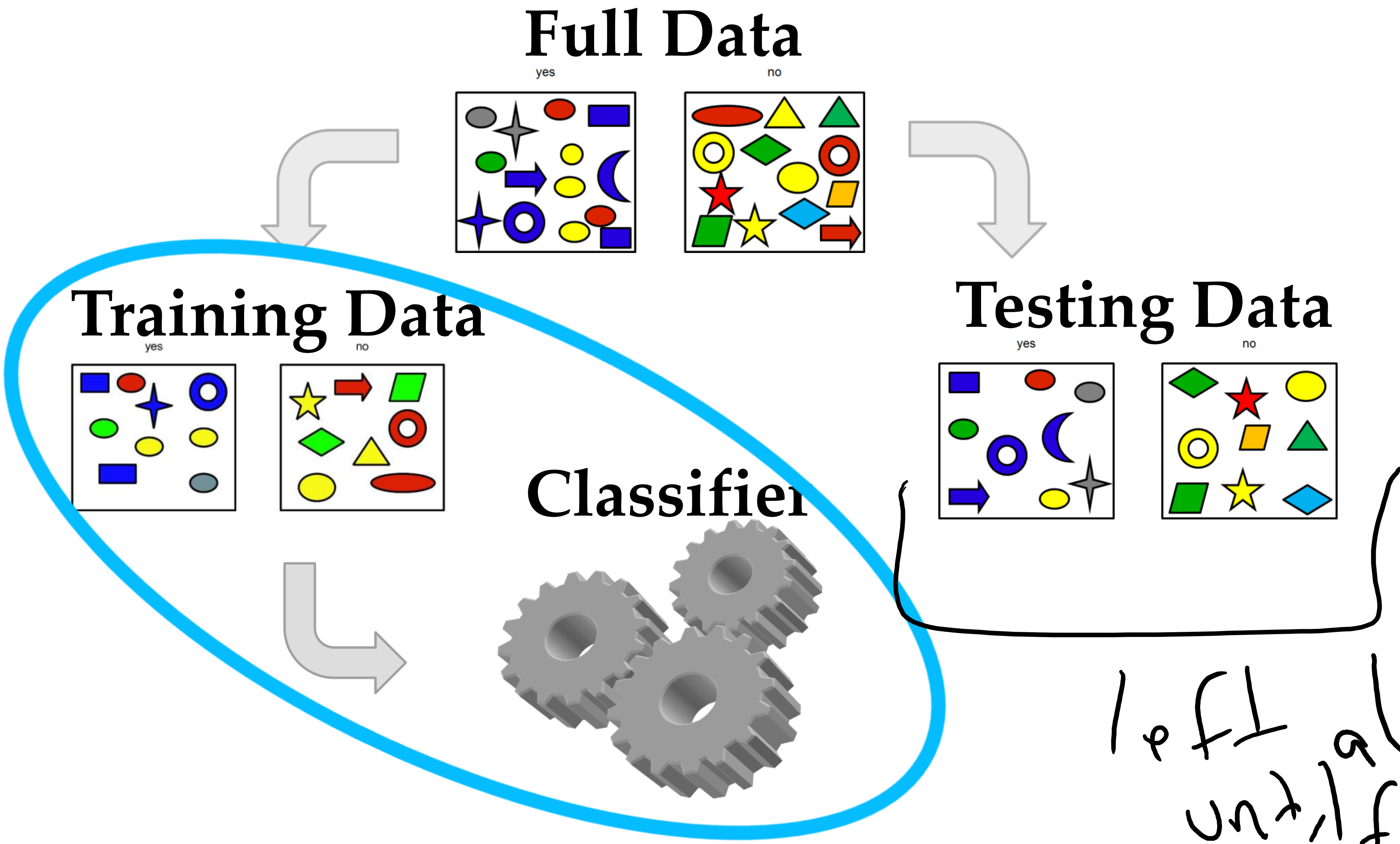
When this model was applied to the actual market, it consistently was wrong less than 50% of the time. What went wrong?

Approach

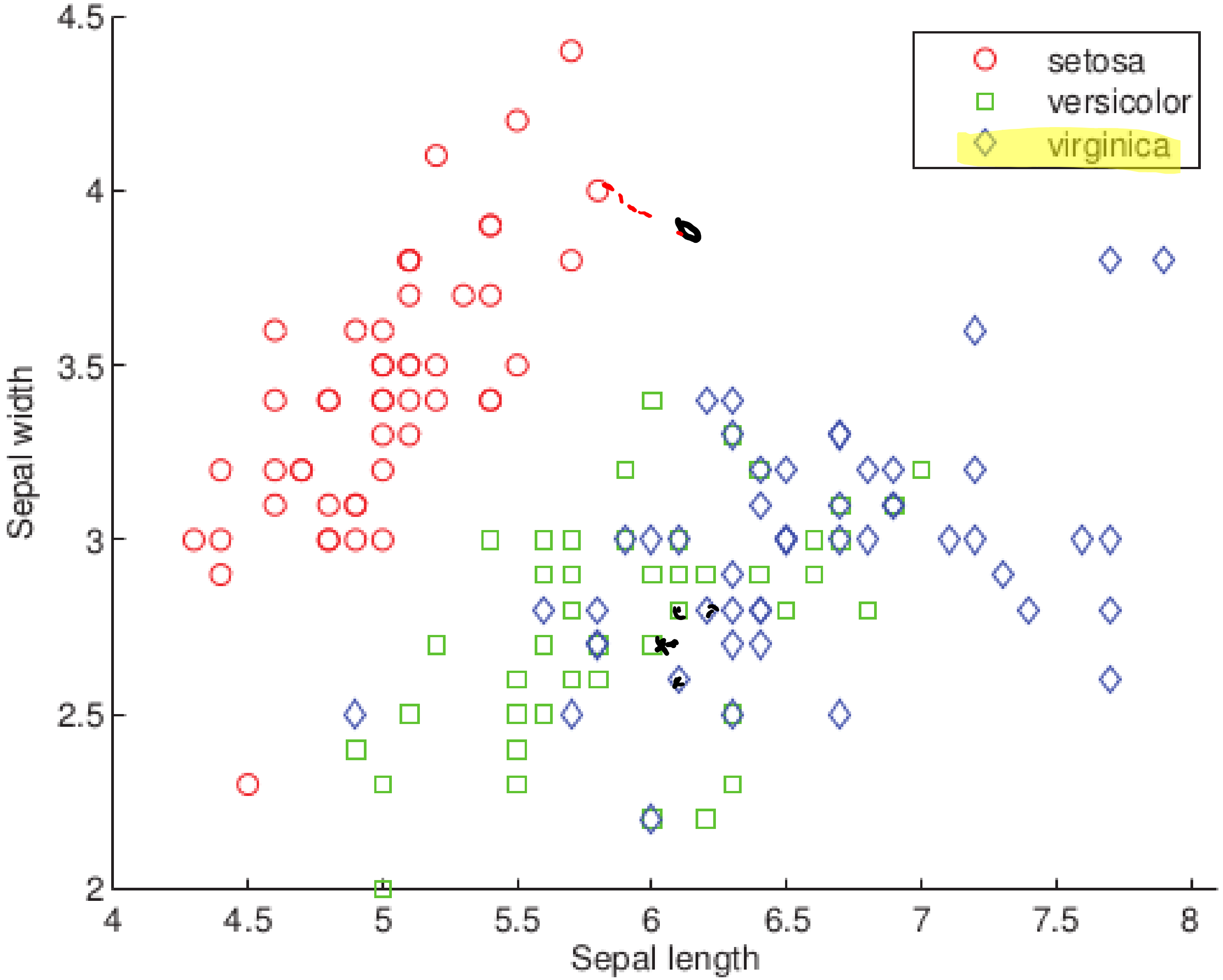
A bank creates an investment ML algorithm that forcasts whether a certain bond value will go up or down. As input, it receives a group of financial features that have all been standardized for mean and variance. The bank separated the data into two groups, training and testing. After building the model on the training data, the model trained on the testing data can correctly determine whether the bond will go up or down with 52% accuracy.

When this model was applied to the actual market, it consistently was wrong less than 50% of the time. What went wrong?

The bank normalized both the training and testing sets, meaning the training set was actually impacted by the values in the test set.



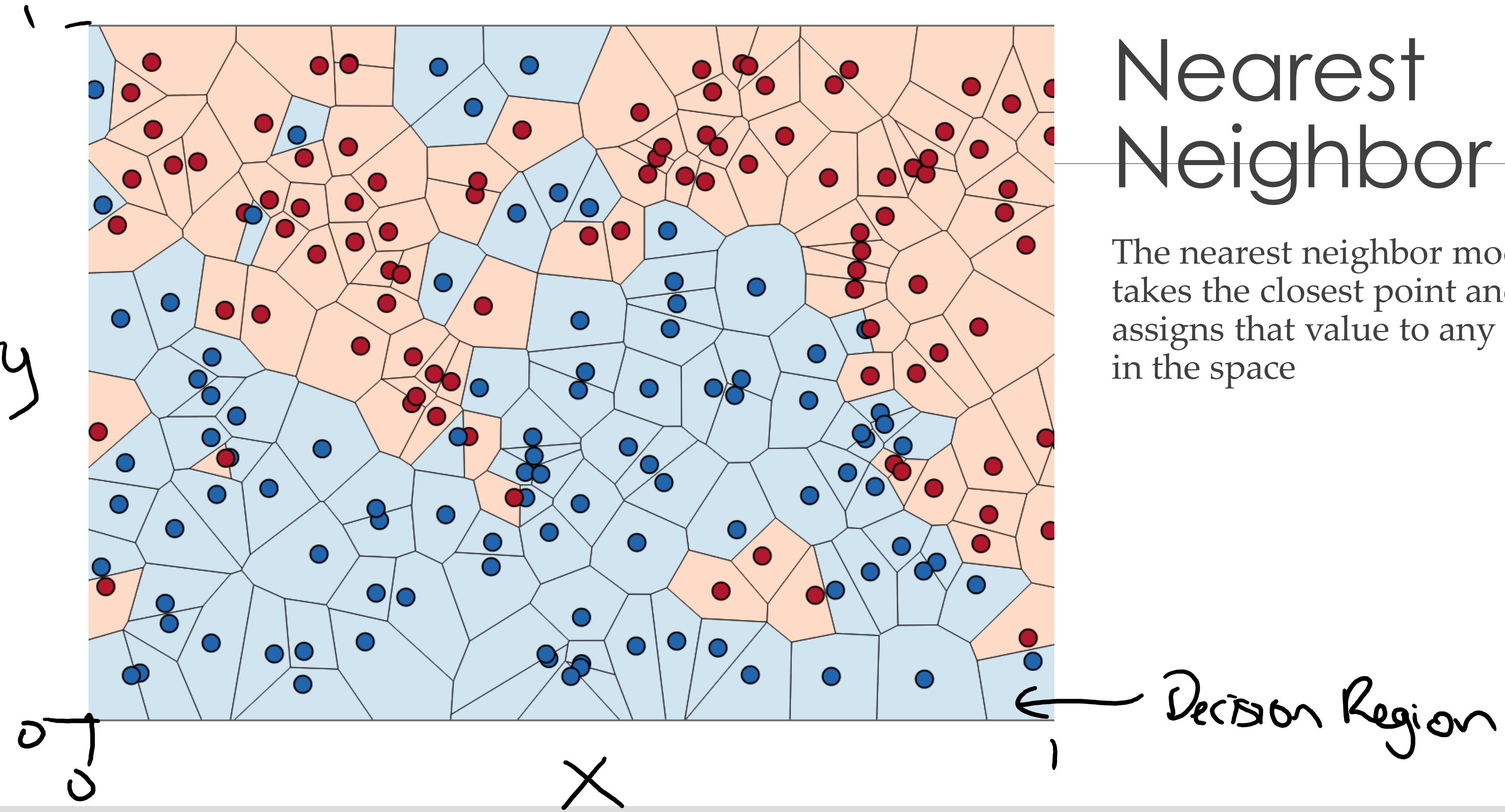
left alone
until finished



Approach

Given the following data, what are some approaches that we could use to make a simple model?

if k = n



Nearest Neighbor

The nearest neighbor model takes the closest point and assigns that value to any region in the space

Nearest Neighbor

What are some observations we could make about this modeling technique?

Number of neighbors – K (usually odd)

for training we have 100% accuracy – Overfitting

Nearest Neighbor

What are some observations we could make about this modeling technique?

- Needs to store and run through all of the data
- Not clearly defined by what we mean by distance
- Does not deal well with regions of high overlap
- Categorical features?

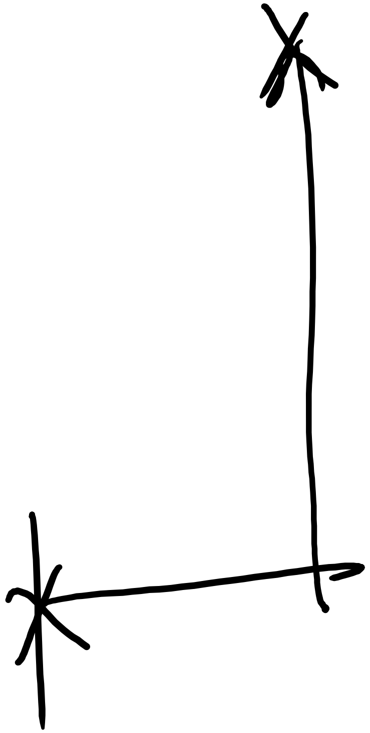
Nearest Neighbor

What are some observations we could make about this modeling technique?

- Needs to store and run through all of the data
- Not clearly defined by what we mean by distance
  - Euclidian – straight line distance
  - Manhattan – sum of feature differences (going by street blocks)
  - Chebyshev – closest by maximum dimension
- Does not deal well with regions of high overlap
- Categorical features?
  - Need to create (n-1) dummy features

Euclid: $\sqrt[n]{\sum_i^2}$

Man $= \sum |i|$



k-Nearest Neighbor

For binary classification problems

- Take the majority classification of the k-nearest neighbors (where k is odd)

n

For numerical output problems

- Take the average of the k-nearest neighbors OR
- Take the weighted average of the k-nearest neighbors

k-Nearest Neighbor

For binary classification problems

- Take the majority classification of the k-nearest neighbors (where k is odd)

For numerical output problems

- Take the average of the k-nearest neighbors OR
- Take the weighted average of the k-nearest neighbors (Gaussian distribution)

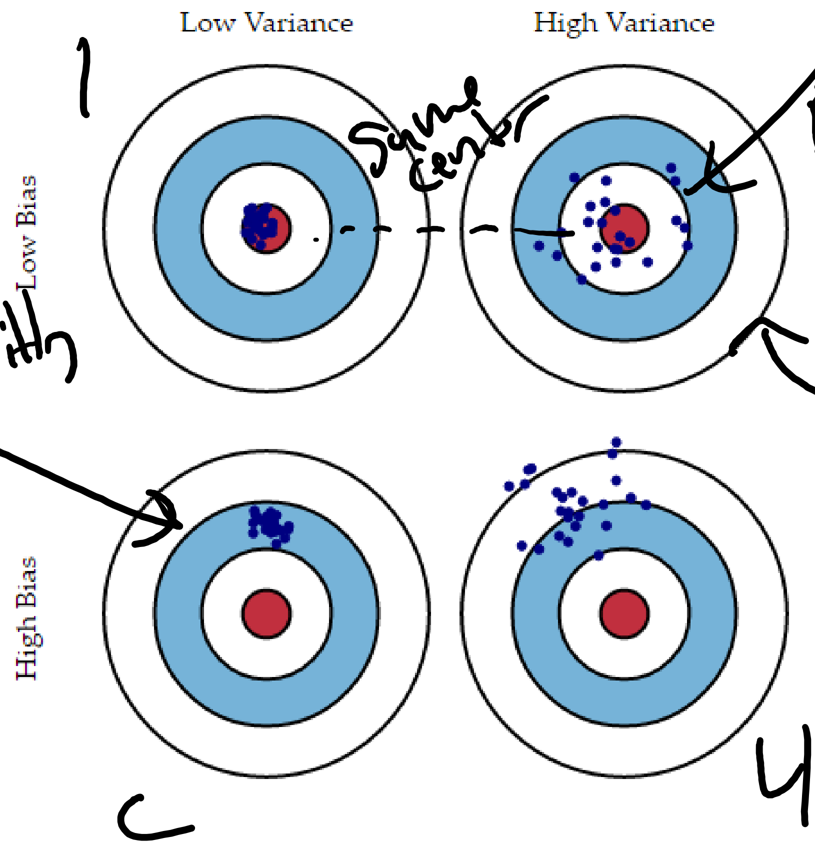
Normal

K=n acc = most common point

K=1 acc = 100% on the training set

Nearest Neighbor Regression Analysis

model f_D



underfitting

higher expected distance to center

overfitting

Bias: $E[\hat{f}_D(x)] - f(x)$

Variance: $E[(\hat{f}_D(x) - E[\hat{f}_D(x)])^2]$

notice f(x) is not here

$E(|\hat{X} - X|) = 0$

How do we evaluate?

train validation

After test data is split off

K=5

K-Fold Cross Validation:

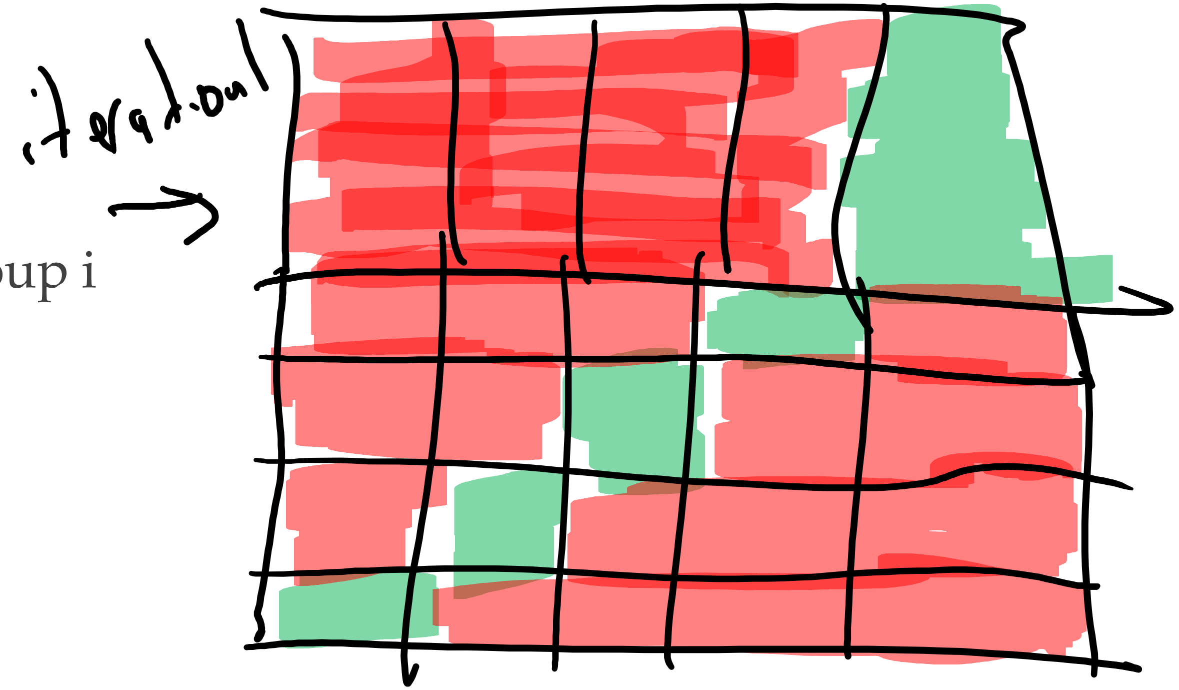
Split data into K groups.

For i=1:K

Train classifier on all except group i

Evaluate on group i

Return average evaluation



How do we evaluate?

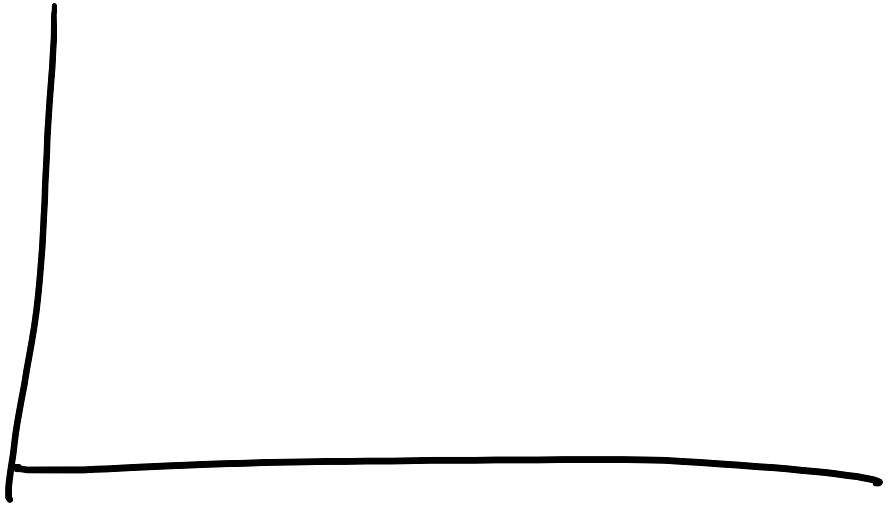
Leave One Out Cross Validation (LOOCV): = n-fold cross validation

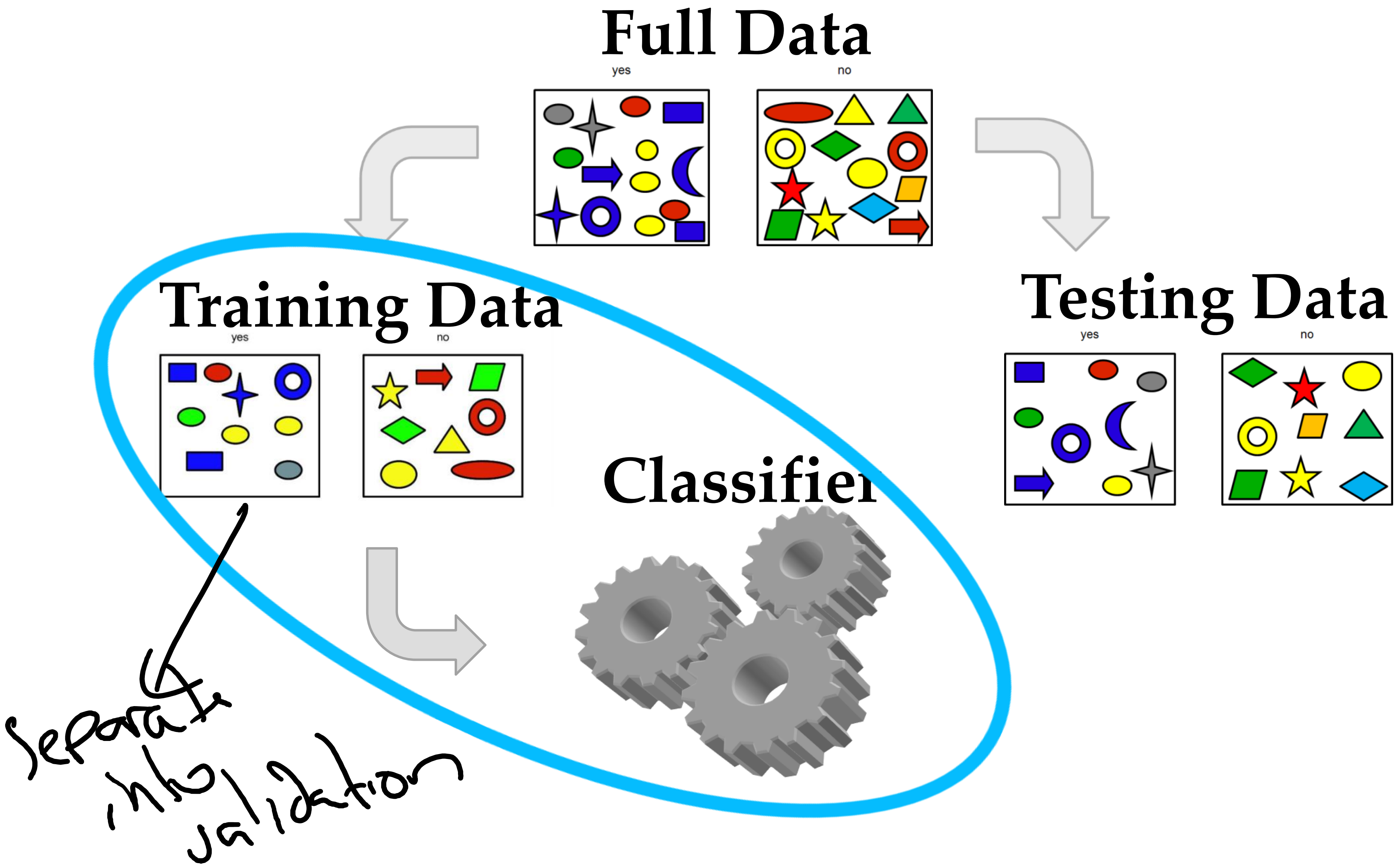
For all data points (k) in the validation set:

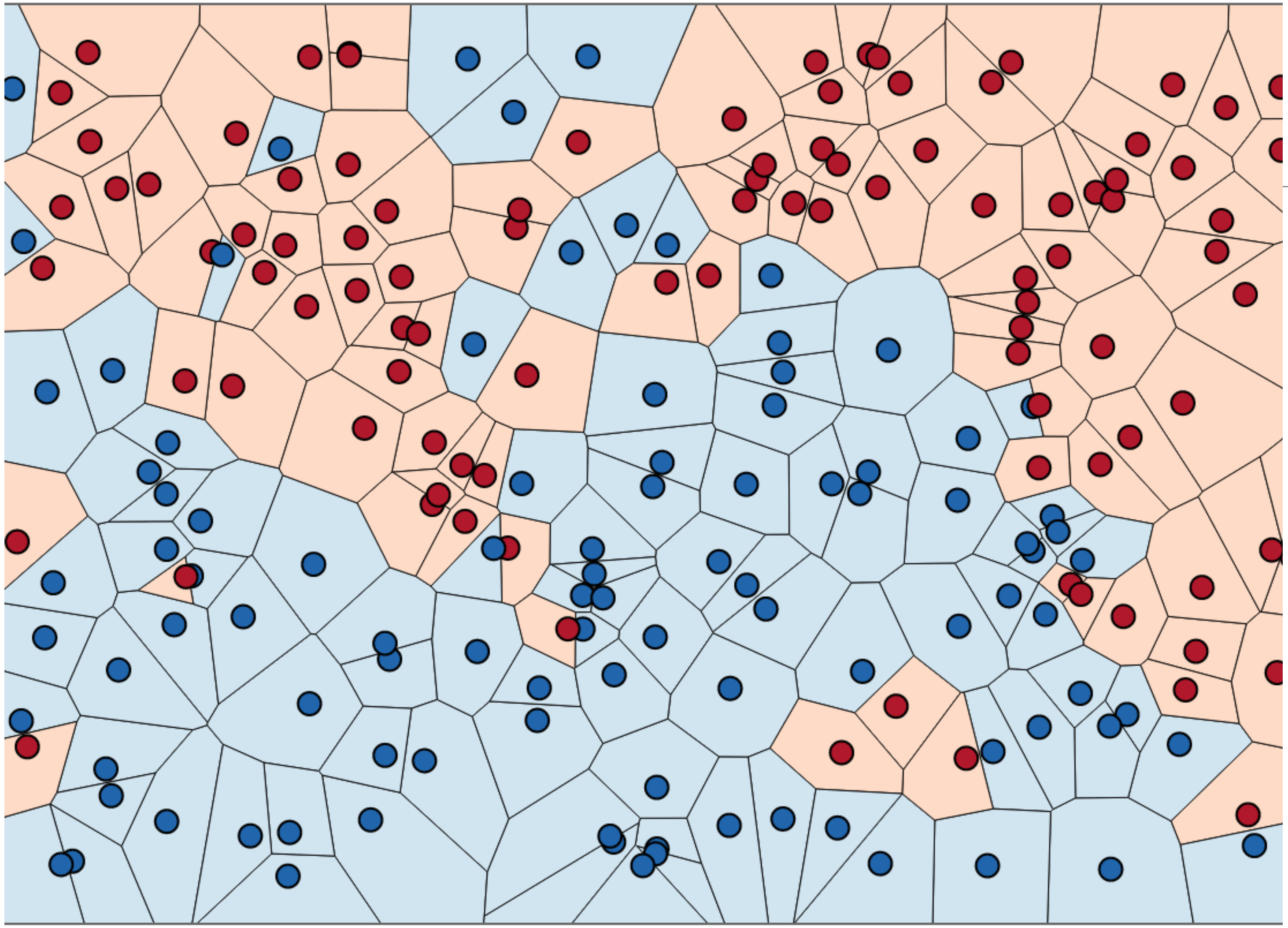
Train on all other data points

Test on the single point k

Return the average evaluation




Full Data
yes
no
Training Data
yes
no
Testing Data
yes
no
Classifier
separate into validation



Nearest Neighbor

The nearest neighbor model takes the closest point and assigns that value to any region in the space

Nearest Neighbor

What are some observations we could make about this modeling technique?

- Needs to store and run through all of the data
- Not clearly defined by what we mean by distance
  - Euclidian – straight line distance
  - Manhattan – sum of feature differences (going by street blocks)
  - Chebyshev – closest by maximum dimension
- Does not deal well with regions of high overlap
- Categorical features?
  - Need to create (n-1) dummy features

Eye color (B, Green, Gray, Brown

→ Blue (1,0)

→ Gray (1,0)

→ Brown (1,0

if (0,0,0) → Green

k-Nearest Neighbor

For binary classification problems

- Take the majority classification of the k-nearest neighbors (where k is odd)

For numerical output problems

- Take the average of the k-nearest neighbors OR
- Take the weighted average of the k-nearest neighbors

k-Nearest Neighbor

For binary classification problems
- Take the majority classification of the k-nearest neighbors (where k is odd)

For numerical output problems
- Take the average of the k-nearest neighbors OR
- Take the weighted average of the k-nearest neighbors (Gaussian distribution)

k-Nearest Neighbor

For binary classification problems

- Take the majority classification of the k-nearest neighbors (where k is odd)

For numerical output problems

- Take the average of the k-nearest neighbors OR
- Take the weighted average of the k-nearest neighbors (Gaussian distribution)

We may also want to take representative samples

This is k-clustering, and we'll discuss this later

How do we evaluate a model?

EPE *(estimated prediction error)* for a given model *f*

$EPE(f) = E(Y\text{-}f(X))^2$ ← Squared error term

Where X is the vector of input attributes,
f(X) is the predicted output and
Y is the actual output

$y = \{0, 1\}$

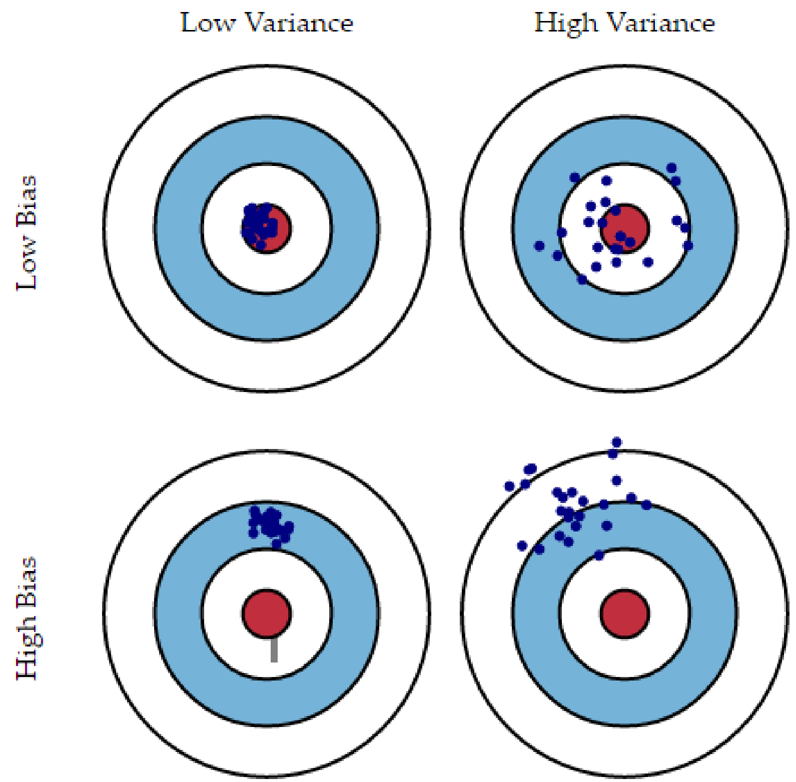
$x = \{0, 1\}$

We will usually try to find the model which minimizes the *estimated error*, but in the classification model, this is just accuracy

$(y-x)^2 = \{0, 1\}$

This may not show the whole picture!

Nearest Neighbor Regression Analysis



Bias: $\hat{E}[f_D(x)] - f(x)$

Variance: $\hat{E}[(f_D(x) - \hat{E}[f_D(x)])^2]$

PAC:

$$P(|\hat{\theta} - \theta^*| > \epsilon) < \delta$$

How do we evaluate?

K-Fold Cross Validation:

Split data into K groups.

For i=1:K

Train classifier on all except group i

Evaluate on group i

Return average evaluation

How do we evaluate?

Leave One Out Cross Validation (LOOCV):

For all data points (k) in the validation set:

Train on all other data points

Test on the single point k

Return the average evaluation

Homework 1

For HW 1, you'll be solving a binary classification problem with k-nearest neighbors for different levels of *k*

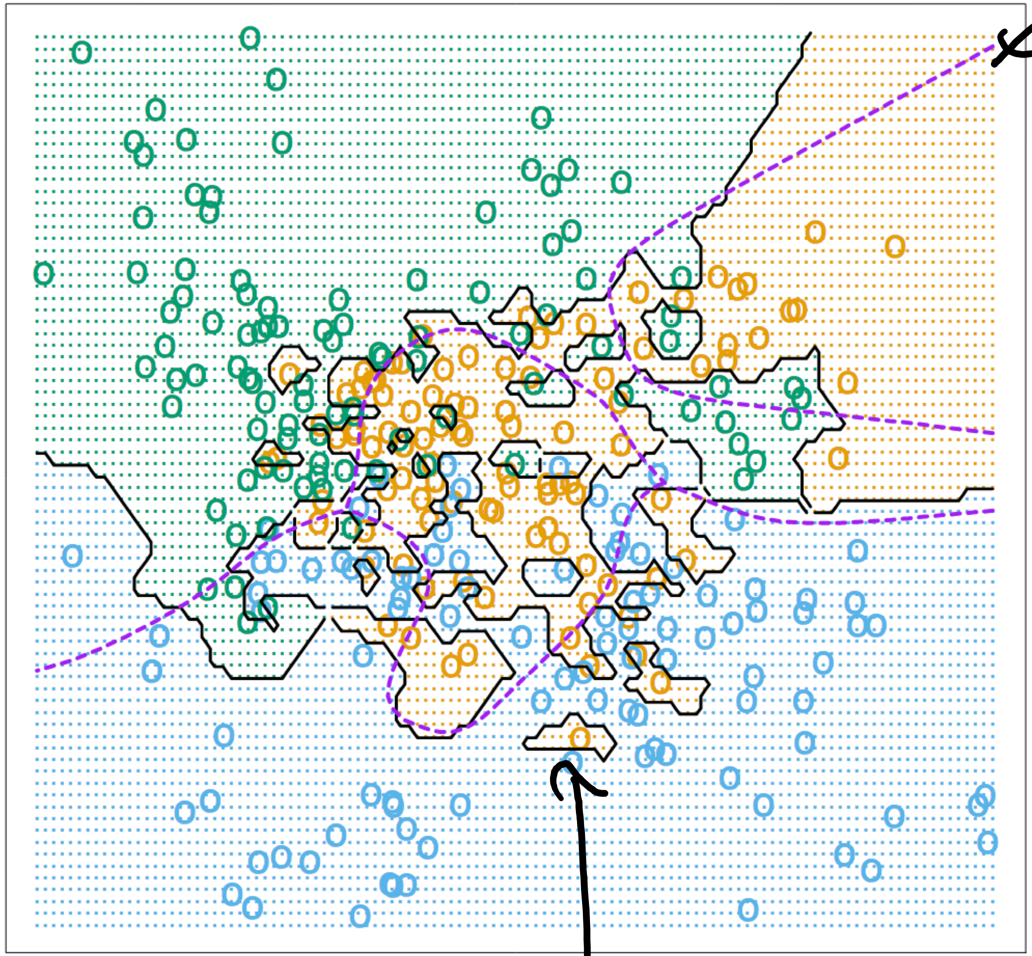
(1, 5)

You will also be running your models with differing sizes of cross-validation and comparing runtimes

Make sure that you separate your test data before begin!

1-Nearest Neighbor

Bayes Boundary



Must be l

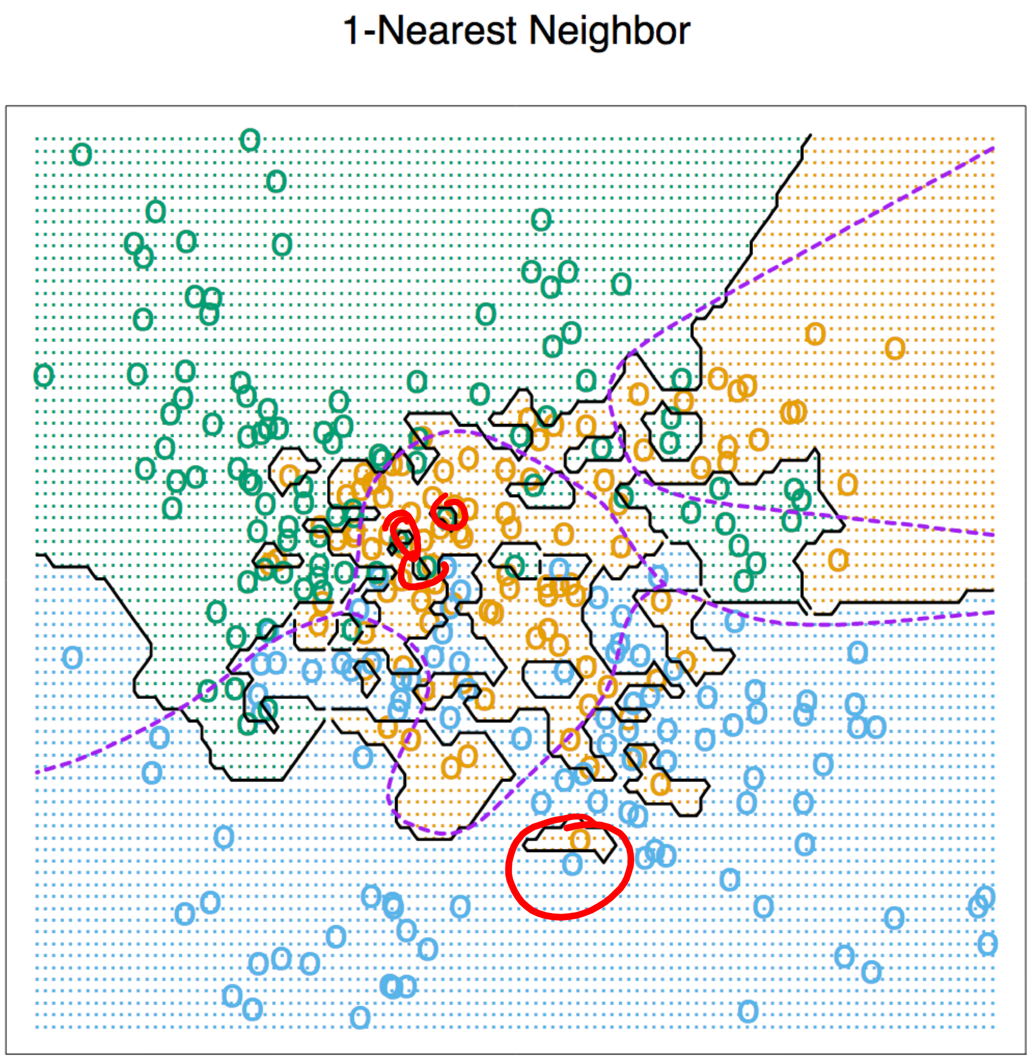
Results

What are some observations we can make about this model?

k =1

3-classification

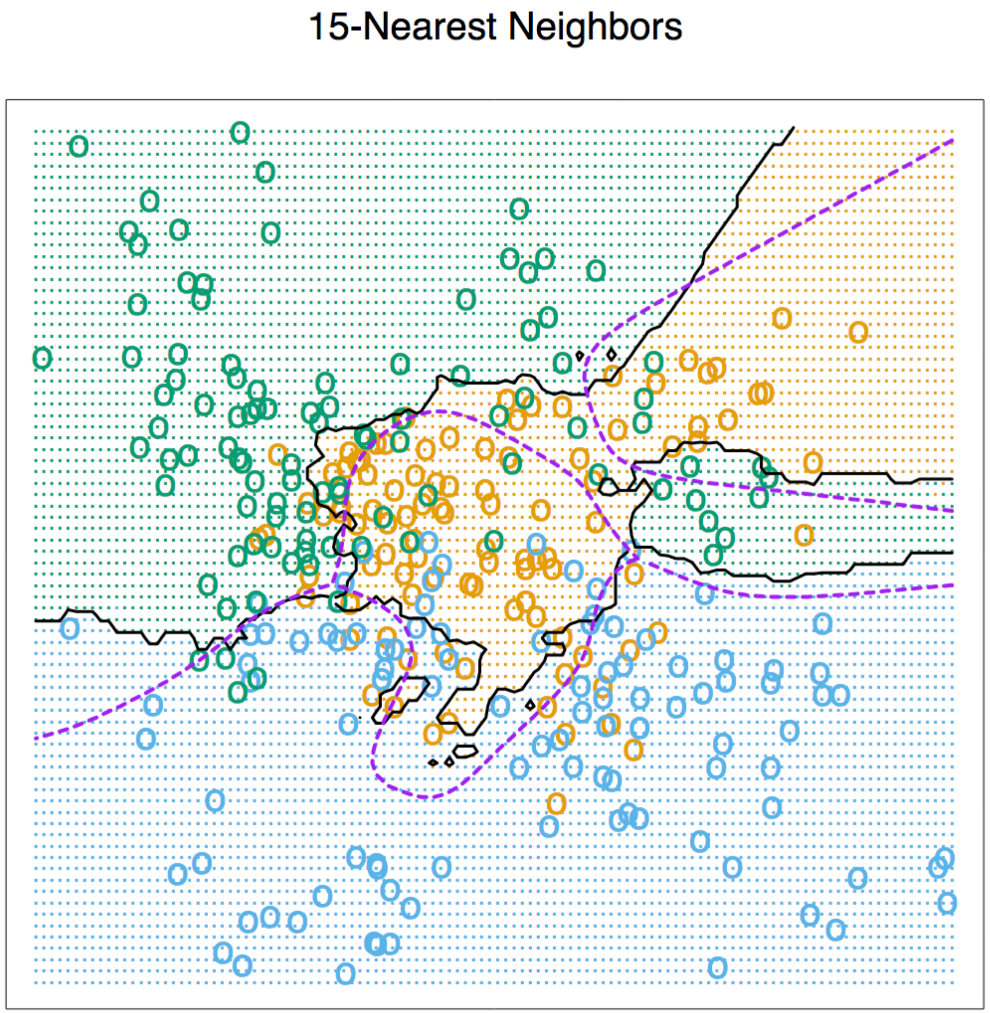
Over or underfit



Results

What are some observations we can make about this model?

- 3 classifications
  - blue
  - orange
  - purple
- Purple line is called the Bayes decision boundary
- Does this suffer from overfitting?



Results

What are some observations we can make about this model?

- 3 classifications
  - blue
  - orange
  - purple
- Purple line is called the Bayes decision boundary
- Does this suffer from overfitting?

bias has increased

variance