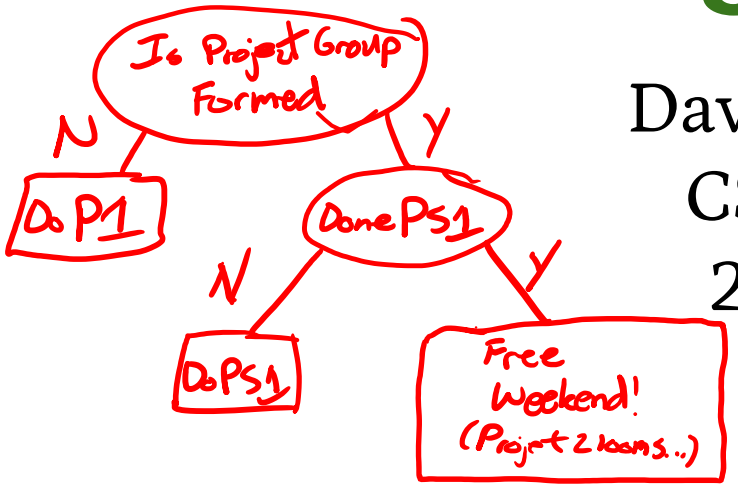
Decision Trees, Forests, and Linear & Logistic Regression



David Quigley
CSCI 5622
2021 Fall

Course Logistics

- Project: Groups due 9/9!
    - Everyone submits an identical document
    - Project 2: Pitch and Project 2.1: Pitch Feedback instructions are now viewable on Canvas
- Problem Sets: Problem Set 1 Due 9/16
    - Piazza is pretty active
        - If you have a question, it may have already been answered there!

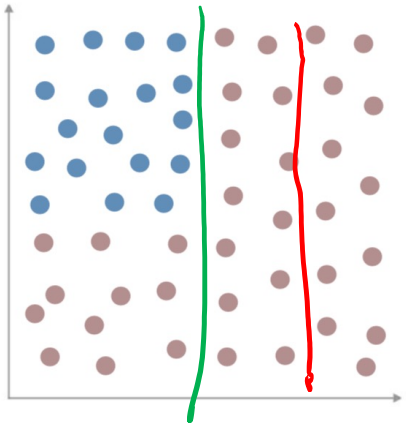
Participation Activity: Naïve Bayes

Decision Trees – Minimizing Entropy END

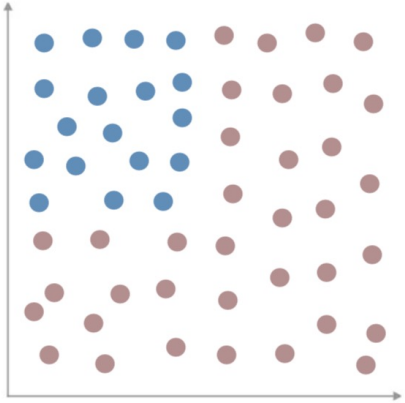
We are creating a split to minimize the entropy of our set...

But really, we're creating *two* sets, each with their own entropy, but a smaller number of samples in each.

D_{par} = Data found in Parent Node
D_{left} = Data found in Left Node
D_{right} = Data found in Right Node
$I()$ = Impurity function (entropy)
x_i = feature for split



Decision Trees – Information Gain



D_{par} = Data found in Parent Node
D_{left} = Data found in Left Node
D_{right} = Data found in Right Node
I() = Impurity function (entropy)
x_i = feature for split

$$IG(D_{par}, xi) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(D_{left}) - \frac{|(D_{right})|}{|(D_{par})|} * I(D_{right})$$

Decision Trees – Choosing a Split

$$IG(D_{par}, xi) = I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

Split based on the feature x_i with the highest Information Gain!

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

Compute th

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

D_{par} = {(Pos, No), (Pos, No), (Neg, Yes), (Pos, Yes)} | D_{par} | = 4 I(D_{par}) = 1

D_{left} = {(Pos, No) (Pos, No) (Pos, Yes)} | D_{left} | = 3 I(D_{left}) =

D_{right} = {(Neg, Yes)} | D_{right} | = 1 I(D_{right}) = 0

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

D_{par} = {(pos, no); (pos, no); (neg, yes); (pos, yes)}

D_{left} = {(pos, no); (pos, no); (pos, yes)}

D_{right} = {(neg, yes)}

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

D_{par} = {(pos, no); (pos, no); (neg, yes); (pos, yes)} $|D_{par}|$ = $I(D_{par})$ =

D_{left} = {(pos, no); (pos, no); (pos, yes)} $|D_{left}|$ = $I(D_{left})$ =

D_{right} = {(neg, yes)} $|D_{right}|$ = $I(D_{right})$ =

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

$| D_{par} | =$

$| D_{left} | =$

$| D_{right} | =$

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

D_{par} = {(pos, no); (pos, no); (neg, yes); (pos, yes)} $|D_{par}| = 4$ $I(D_{par})$ =

D_{left} = {(pos, no); (pos, no); (pos, yes)} $|D_{left}| = 3$ $I(D_{left})$ =

D_{right} = {(neg, yes)} $|D_{right}| = 1$ $I(D_{right})$ =

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

$I(D_{par})$ =

$I(D_{left})$ =

$I(D_{right})$ =

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

$I(D_{left})$ =

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

$I(D_{right})$ =

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

D_{par} = {(pos, no); (pos, no); (neg, yes); (pos, yes)} $|D_{par}| = 4$ $I(D_{par}) = 1$

D_{left} = {(pos, no); (pos, no); (pos, yes)} $|D_{left}| = 3$ $I(D_{left}) = 0.918$

D_{right} = {(neg, yes)} $|D_{right}| = 1$ $I(D_{right}) = 0$

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = (1 - \frac{3}{4} * 0.918 - \frac{1}{4} * 0)$$

D_{par} = {(pos, no); (pos, no); (neg, yes); (pos, yes)} $|D_{par}| = 4$ $I(D_{par}) = 1$

D_{left} = {(pos, no); (pos, no); (pos, yes)} $|D_{left}| = 3$ $I(D_{left}) = 0.918$

D_{right} = {(neg, yes)} $|D_{right}| = 1$ $I(D_{right}) = 0$

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_A) = 0.3112$$

D_{par} = {(pos, no); (pos, no); (neg, yes); (pos, yes)} $|D_{par}| = 4$ $I(D_{par}) = 1$

D_{left} = {(pos, no); (pos, no); (pos, yes)} $|D_{left}| = 3$ $I(D_{left}) = 0.918$

D_{right} = {(neg, yes)} $|D_{right}| = 1$ $I(D_{right}) = 0$

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_B) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

D_{par} = $|D_{par}|$ = $I(D_{par})$ =

D_{left} = $|D_{left}|$ = $I(D_{left})$ =

D_{right} = $|D_{right}|$ = $I(D_{right})$ =

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_B) = 0$$

D_{par} = {(neg, no); (pos, no); (pos, yes); (neg, yes)} $|D_{par}| = 4$ $I(D_{par}) = 1$

D_{left} = {(pos, no); (pos, yes)} $|D_{left}| = 2$ $I(D_{left}) = 1$

D_{right} = {(neg, no); (neg, yes)} $|D_{right}| = 2$ $I(D_{right}) = 1$

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_C) = (I(D_{par}) - \frac{|(D_{left})|}{|(D_{par})|} * I(Dleft) - \frac{|(D_{right})|}{|(D_{par})|} * I(Dright)$$

D_{par} = $|D_{par}|$ = $I(D_{par})$ =

D_{left} = $|D_{left}|$ = $I(D_{left})$ =

D_{right} = $|D_{right}|$ = $I(D_{right})$ =

Decision Trees – Splitting Cancer

Test A	Test B	Test C	Cancer?
Pos	Neg	Neg	No
Pos	Pos	Neg	No
Neg	Pos	Pos	Yes
Pos	Neg	Pos	Yes

$$IG(D_{par}, T_C) = 1$$

D_{par} = {(neg, no); (neg, no); (pos, yes); (pos, yes)} $|D_{par}| = 4$ $I(D_{par}) = 1$

D_{left} = {(pos, yes); (pos, yes)} $|D_{left}| = 2$ $I(D_{left}) = 0$

D_{right} = {(neg, no); (neg, no)} $|D_{right}| = 2$ $I(D_{right}) = 0$

Decision Trees – Measures of Impurity

We used Entropy as the measure of impurity / messiness / chaos

Is this the only option? The correct option?

Decision Trees – Measures of Impurity

We used Entropy as the measure of impurity / messiness / chaos

Is this the only option? The correct option?

Is it sunny in Yuma? 87% yes

Decision Trees – Measures of Impurity

We used Entropy as the measure of impurity / messiness / chaos

Is this the only option? The correct option?

Is it sunny in Yuma? 87% yes

Misclassification Error = min(p, 1-p) = 13%

Decision Trees – Measures of Impurity

We used Entropy as the measure of impurity / messiness / chaos

Is this the only option? The correct option?

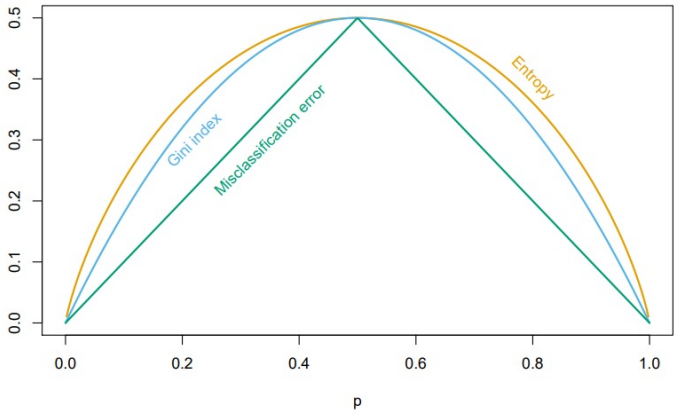
Gini Index: $1 - \Sigma_{c=1...n} p_c^2$

Binary case: $1 - p^2 - (1 - p)^2 = 2p(p-1)$

Decision Trees – Measures of Impurity

We used Entropy as the measure of impurity / messiness / chaos

Is this the only option? The correct option?



Considerations: Differentiability, Computational Speed

Different implementations (CART, C4.5) use different functions

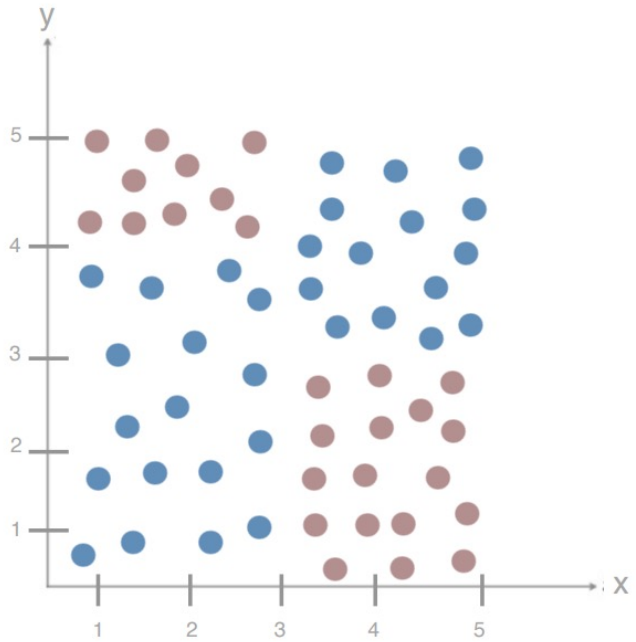
Decision Trees – Adding Complexity

How do we decide the order of our splits?

How do we decide the location of splits?

With continuous variables?

How do we know we're done?



Decision Trees – Continuous Variables

Test A	Test B	Test C	Cancer?
Pos	Neg	3	No
Pos	Pos	2	No
Neg	Pos	6	Yes
Pos	Neg	5	Yes

Eyeball – What's our best splitting point here for Test C?

3 ~ 5

4

2, 3 No

5, 6 Yes

Decision Trees – Continuous Variables

Test A	Test B	Test C	Cancer?
Pos	Neg	3	No
Pos	Pos	2	No
Neg	Pos	6	Yes
Pos	Neg	5	Yes

Eyeball – What's our best splitting point here for Test C?

How did you do make this decision?

Decision Trees – Continuous Variables

Test A	Test B	Test C	Cancer?
7	4	3	No
9	8	2	No
2	6	6	Yes
8	3	5	Yes

Eyeball – What's our best splitting point here?

Decision Trees – Continuous Variables

Test A	Test B	Test C	Cancer?
7	4	3	No
9	8	2	No
2	6	6	Yes
8	3	5	Yes

Eyeball – What's our best splitting point here?

Try all options for all variables

$$IG(D_{par}, x_{i,j})$$

Decision Trees – Continuous Variables

Test A	Test B	Test C	Cancer?
7	4	3	No
9	8	2	No
2	6	6	Yes
8	3	5	Yes

Eyeball – What's our best splitting point here?

Try all options for all variables

$$IG(D_{par}, x_{i,j})$$

Sort first, *then* compute in one pass

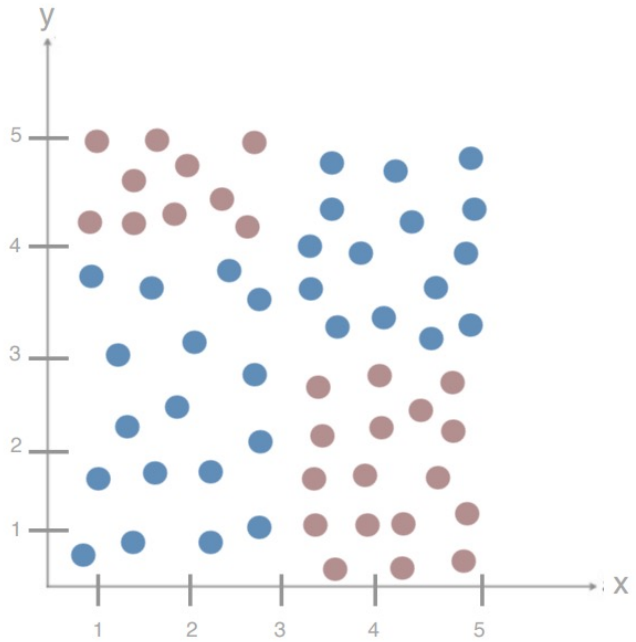
Decision Trees – Adding Complexity

How do we decide the order of our splits?

How do we decide the location of splits?

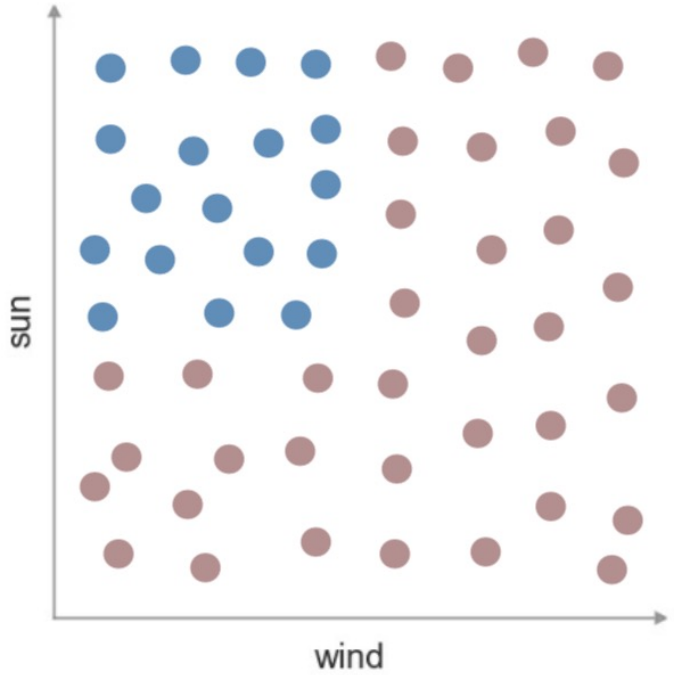
With continuous variables?

How do we know we're done?



Decision Trees – Stopping Point

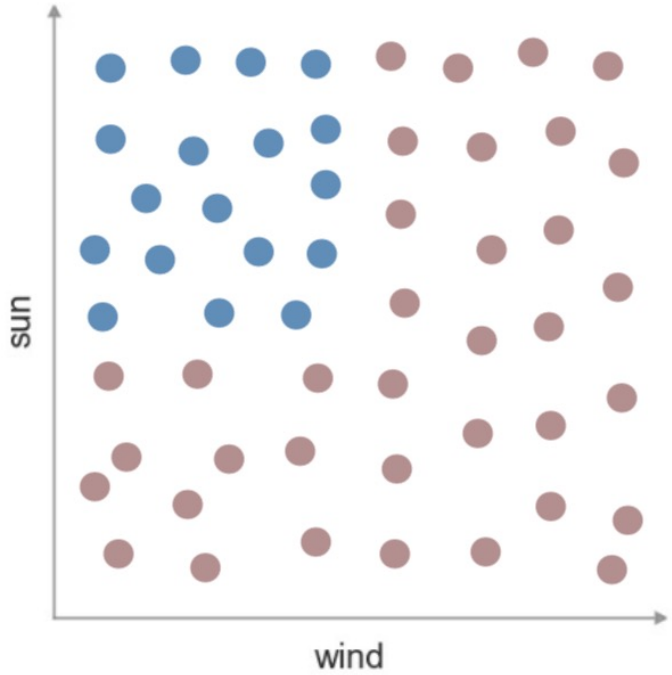
Creating Leaves



Decision Trees – Stopping Point

Creating Leaves

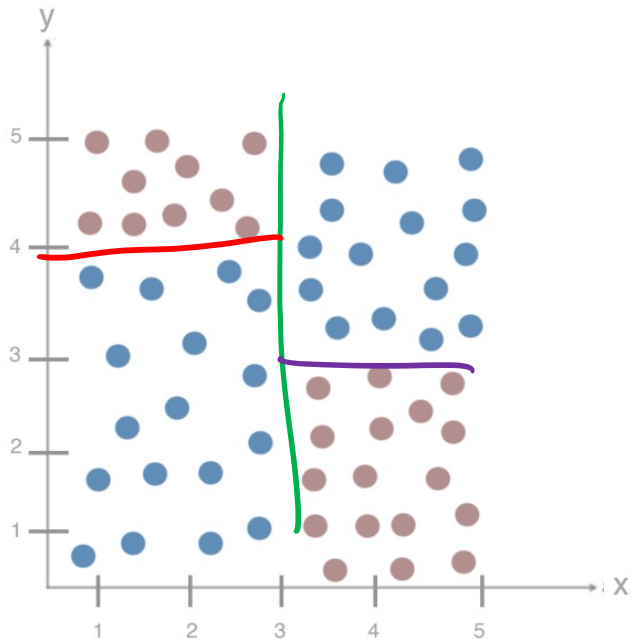
When we reach 0 entropy



Decision Trees – Stopping Point

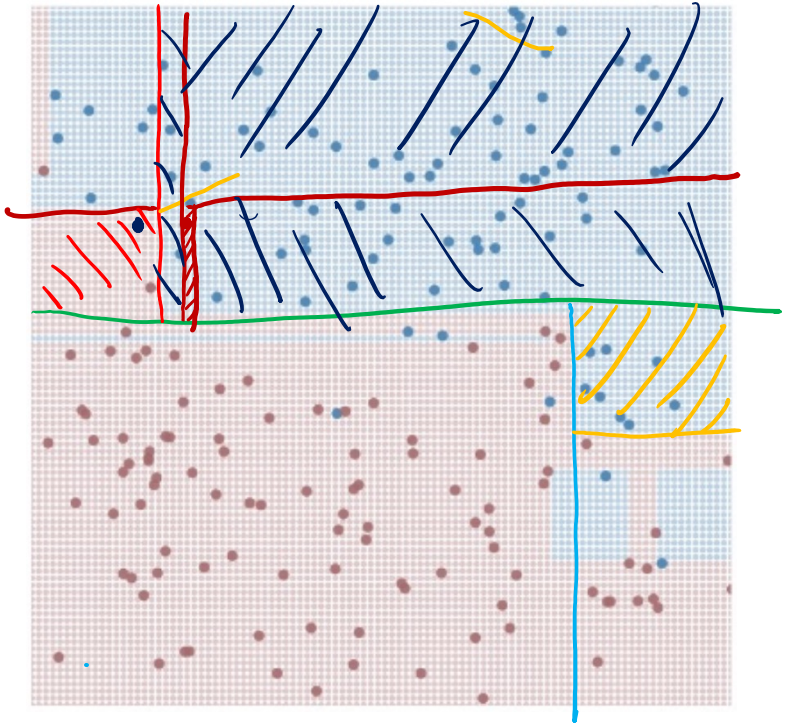
Creating Leaves

When we reach 0 entropy



Decision Trees – Stopping Point

Large, Complex Trees



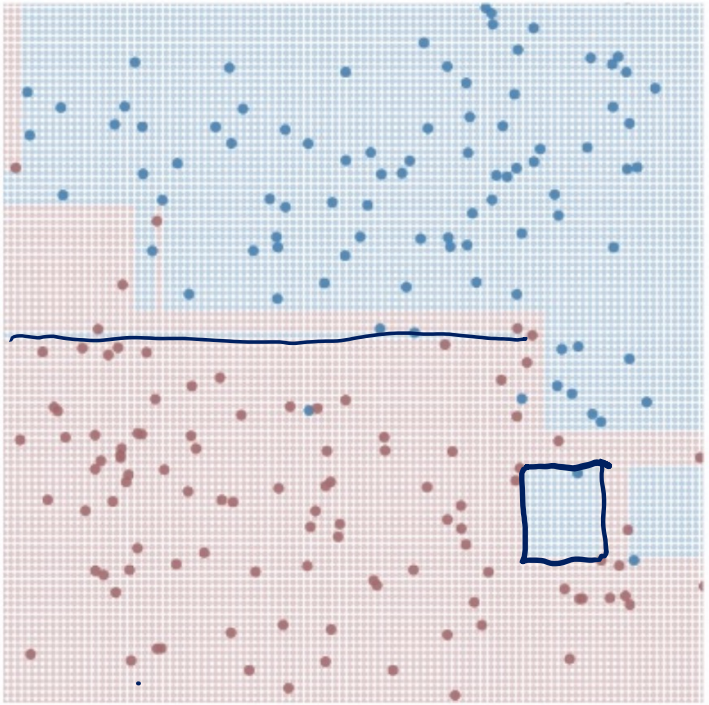
Decision Trees – Stopping Point

Large, Complex Trees

dimensions spirals out of control

Prone to *high variance*

Overfitting



Decision Trees – Stopping Point

Large, Complex Trees

dimensions spirals out of control

Prone to *high variance*

Overfitting

What do you want to do about it?

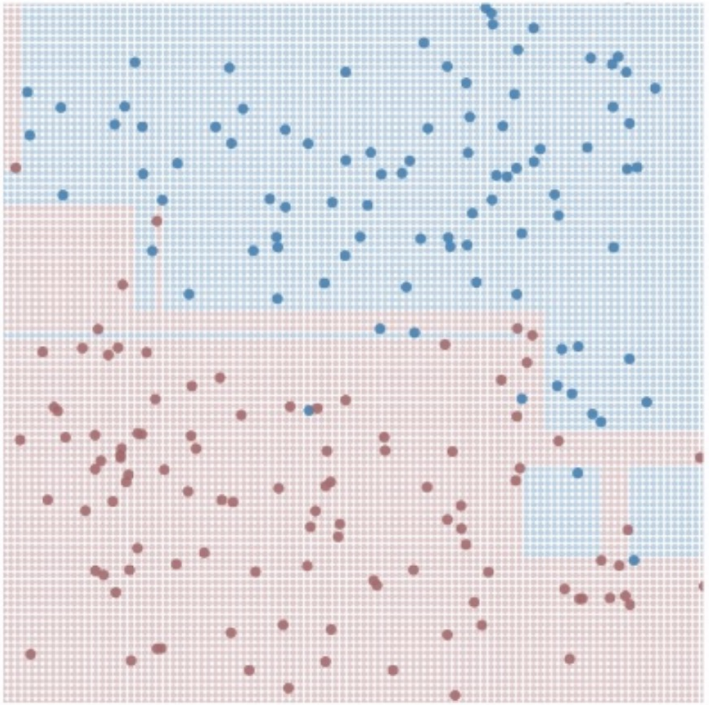
limit number splits

set low nonzero IG

limit threshold for split

test for overfit

prune for sparsity



Decision Trees – Limiting Complexity

Set a maximum depth for your tree

n depends on data & setting

Decision Trees – Limiting Complexity

Set a maximum depth for your tree

Set a maximum number of leaves for your tree (proxy on depth, but allows for deep and shallow branches)

Decision Trees – Limiting Complexity

Set a maximum depth for your tree

Set a maximum number of leaves for your tree (proxy on depth, but allows for deep and shallow branches)

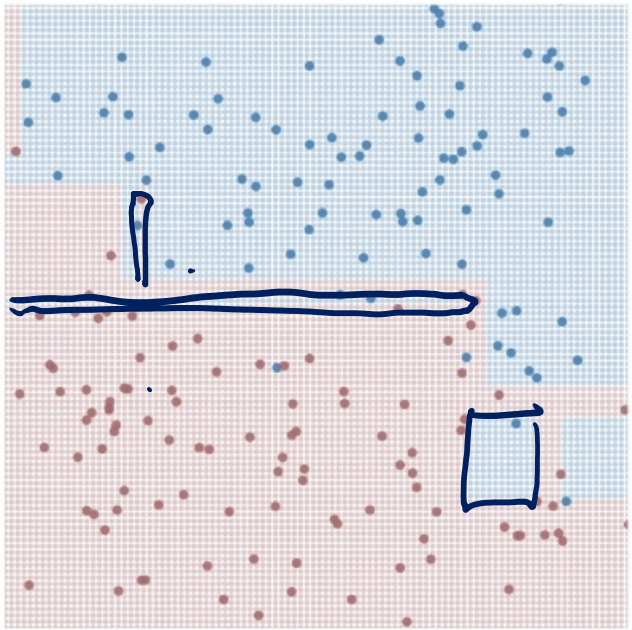
Penalize larger trees (more leaves)

Decision Trees – Pruning

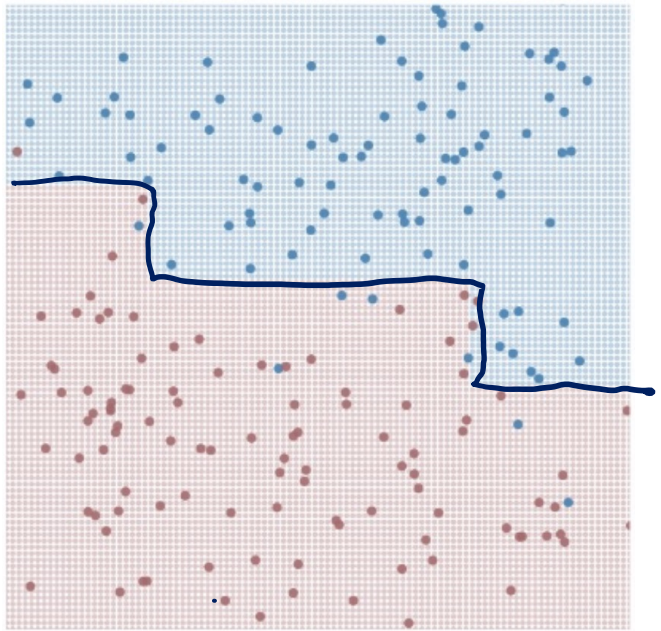
Remove segments (subtrees from your tree)

Decision Trees – Pruning

Remove segments (subtrees from your tree)



→



Decision Trees – Limiting Complexity

Penalize larger trees (more leaves)

$$\frac{\text{err}(\text{prune}(T,t),S) - \text{err}(T,S)}{|\text{leaves}(T)| - |\text{leaves}(\text{prune}(T,t))|}$$

https://en.wikipedia.org/wiki/Decision_tree_pruning

Ensemble Methods, Boosting, Bagging, & Random Forests

Thinking back...

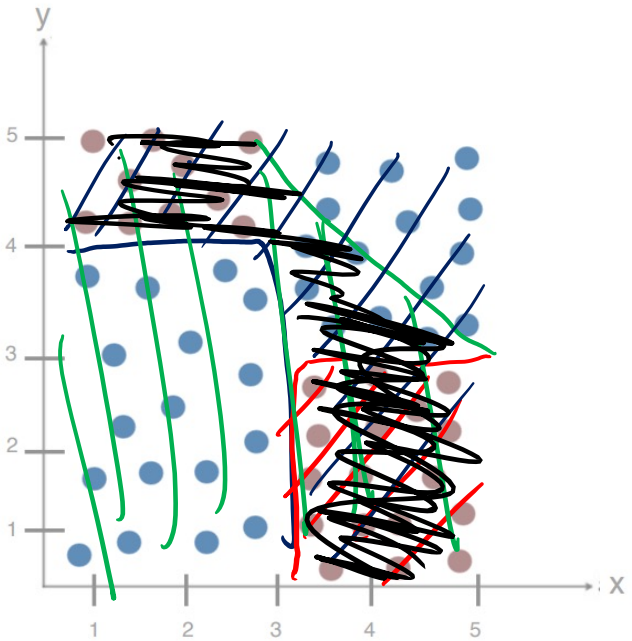
Multi-class decisions (~~Week 3~~)

We built a bunch of different models and relied on *consensus*

	 vs 	 vs 	 vs 
	?	B	A
	A	A	?
	B	?	B

Intuition

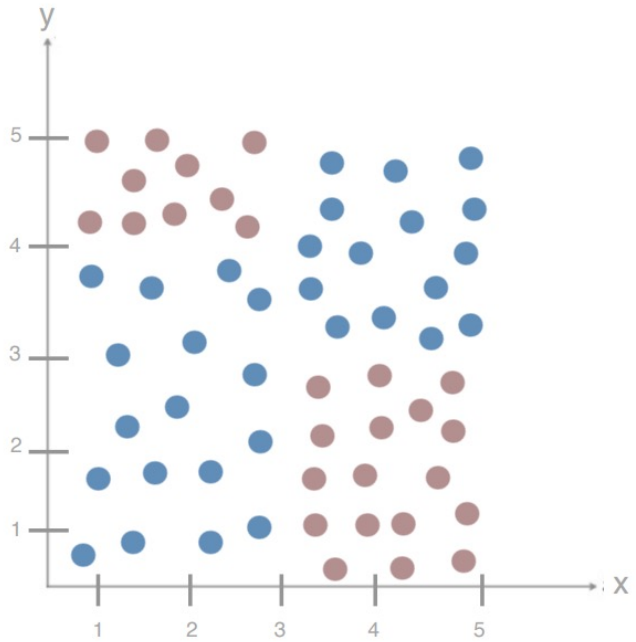
Let’s build a few “bad” models



Intuition

Let’s build a few “bad” models

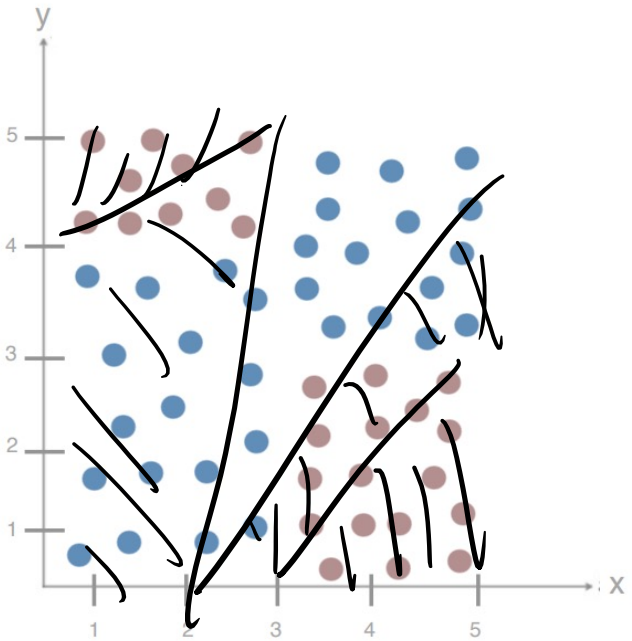
Let’s then combine the predictions!



Intuition

Let’s build a few “bad” models
Let’s then combine the predictions!

Let’s add even more “bad” models!



Ensemble Methods

Build a bunch of *dumb* models on the training data

There are a bunch of ways to do this, stay tuned!

Feed our test data to all the models, get predictions

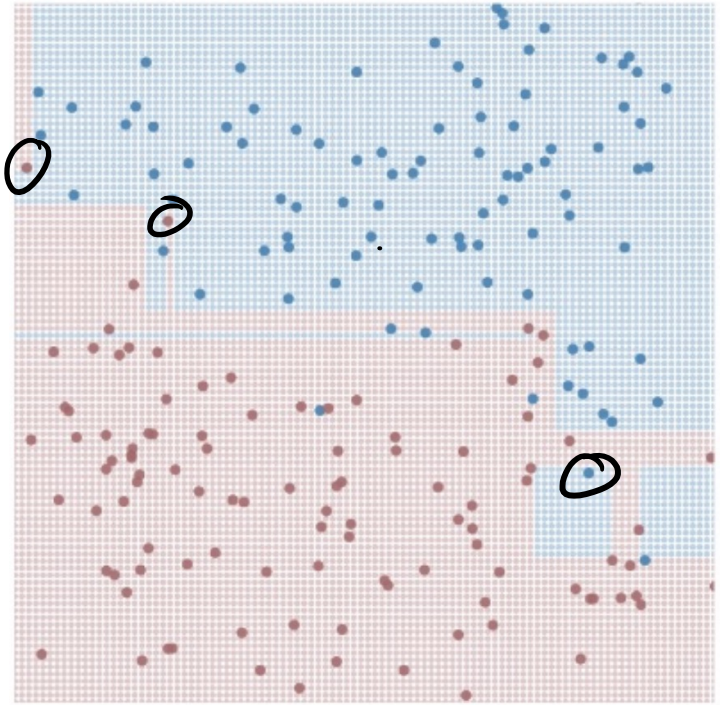
Take the *best* class from all of these models as our answer

Ensemble Learning – Dataset

X_1	X_2	X_3	X_4	X_5	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**	**Single Family?**
1200	1	1.5	1998	200,000	No
1800	2	2	1985	450,000	Yes
800	1	1	2017	250,000	No
2500	3	2	1975	500,000	No
2800	4	2.5	1983	400,000	Yes
…	…	…	…	…	

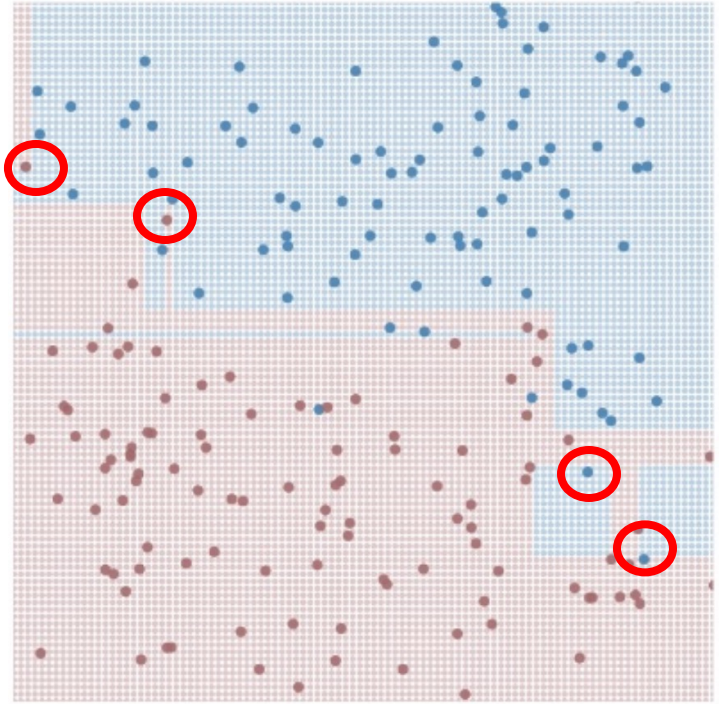
Building “Dumb” Models – Step 1

Maybe 10% of the points in training are causing errors with testing...



Building “Dumb” Models – Step 1

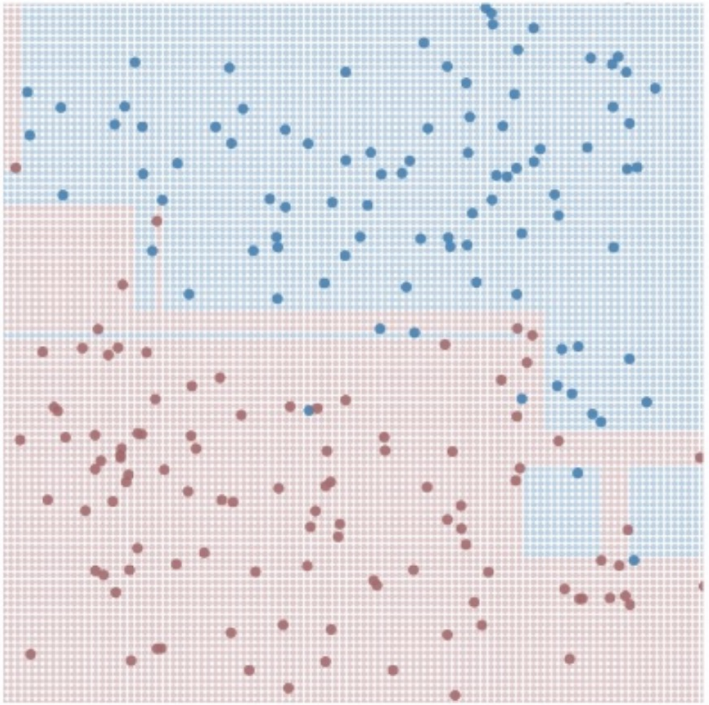
Maybe 10% of the points in training are causing errors with testing...



Building “Dumb” Models – Step 1

Maybe 10% of the points in training are causing errors with testing...

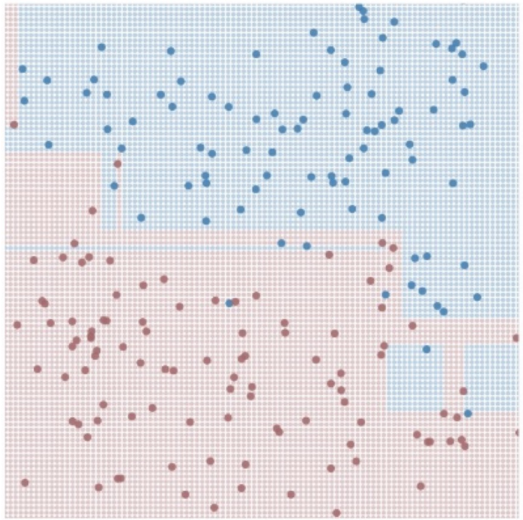
What happens if I build a tree based on a resampling of my points?



Building “Dumb” Models – Resampling

Pick N’ (number in subsample) points

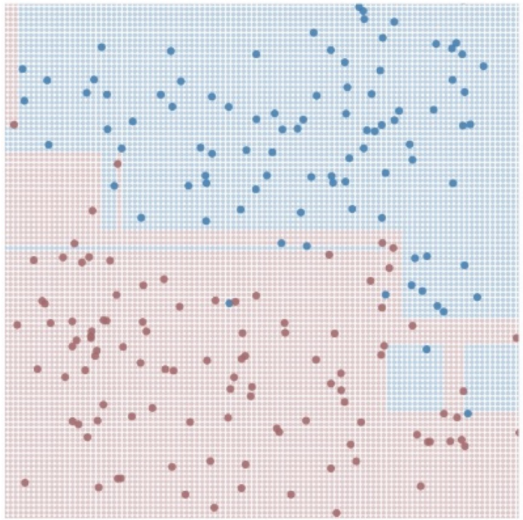
Train model on subsample



Building “Dumb” Models – Resampling

Pick N’ points

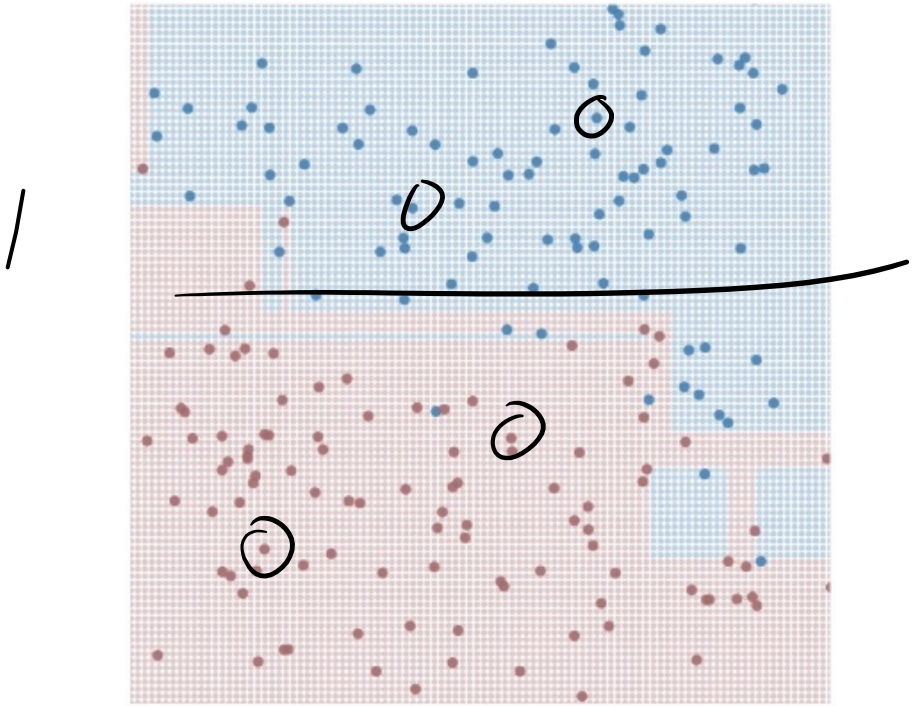
Train model on subsample



Building “Dumb” Models – Step 1

Maybe 10% of the points in training are causing errors with testing...

What happens if I build a tree based on 10% of the points?

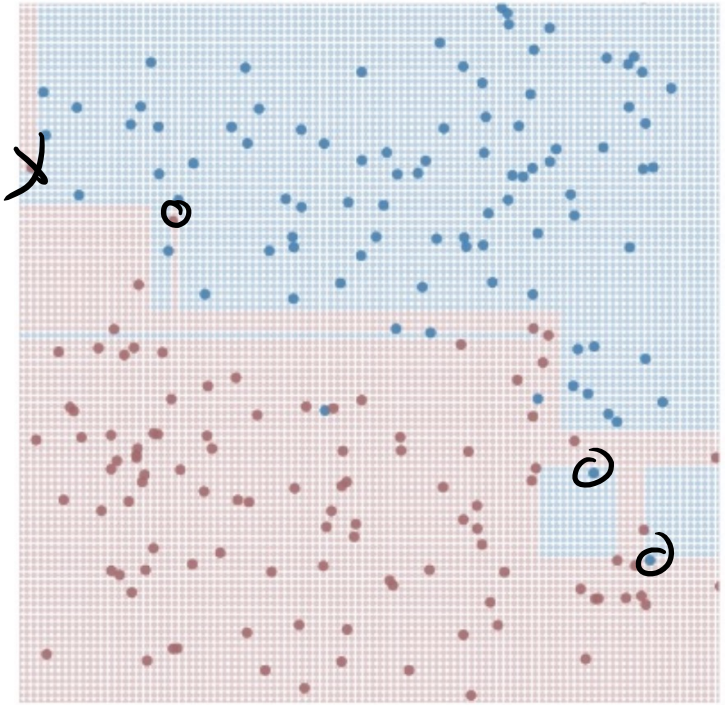


Building “Dumb” Models – Step 1

Maybe 10% of the points in training are causing errors with testing...

What happens if I build a tree based on 10% of the points?

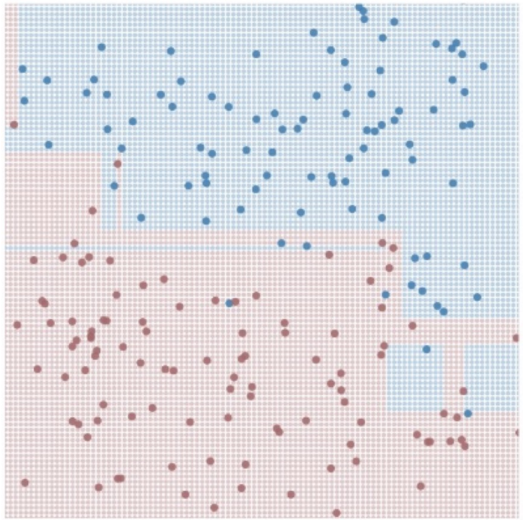
What happens if I build a tree based on 90% of the points?



Building “Dumb” Models – Resampling

Small Subsets

- Prone to high bias (underfitting)



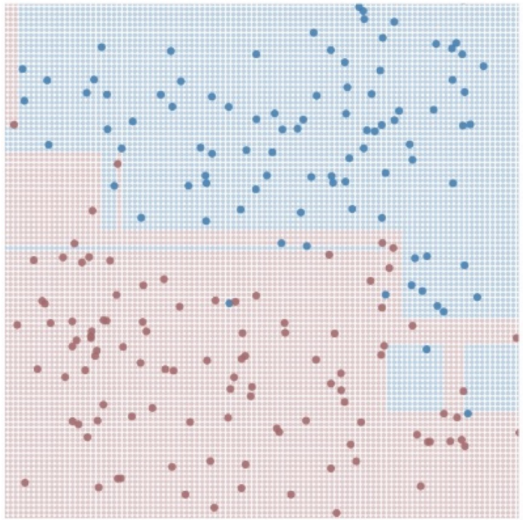
Building “Dumb” Models – Resampling

Small Subsets

- Prone to high bias (underfitting)

Large Subsets

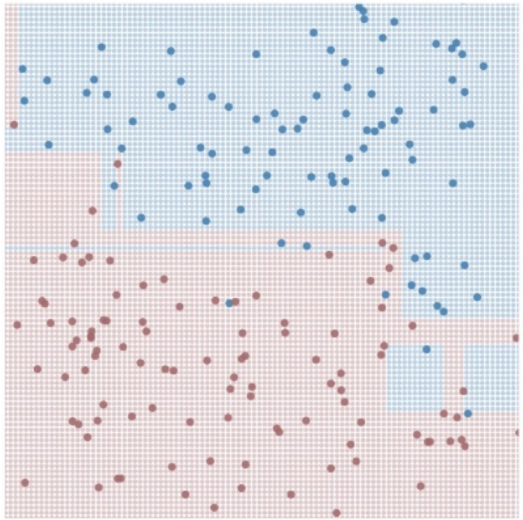
- Prone to behave exactly like our dataset



Building “Dumb” Models – Bootstrapping

Large Subsets *with replacement*

- We will oversample some points, neglect others



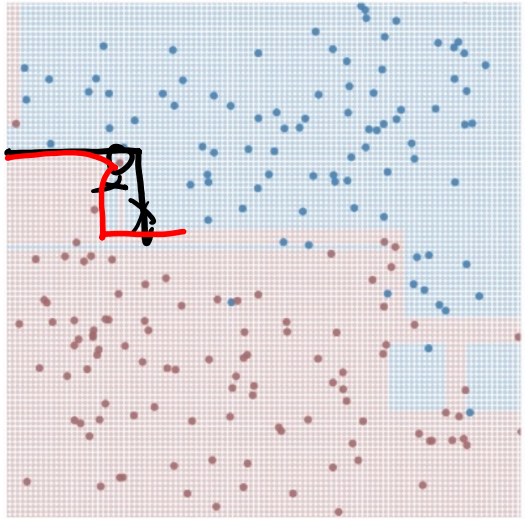
Building "Dumb" Models – Bootstrapping

Large Subsets *with replacement*

- We will oversample some points, neglect others

Sampling N' = N

- Expected 1 – 1/e (approx. 63.2%) coverage



https://en.wikipedia.org/wiki/Bootstrap_aggregating

Building "Dumb" Models – Bootstrapping

Large Subsets *with replacement*

- We will oversample some points, neglect others

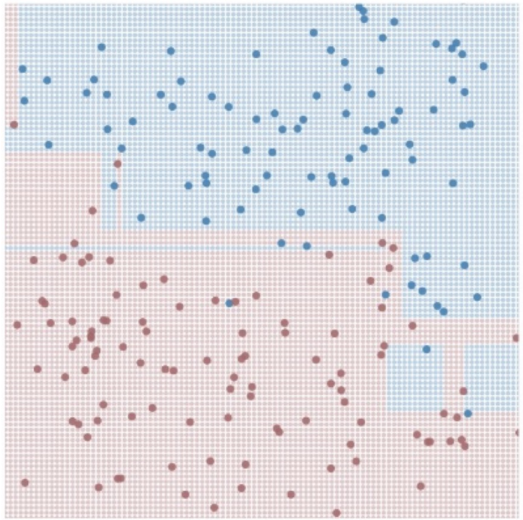
Sampling N' = N

- Expected 1 – 1/e (approx. 63.2%) coverage

N is too large for my memory

- Reduce N' – but realize the bias-variance tradeoff

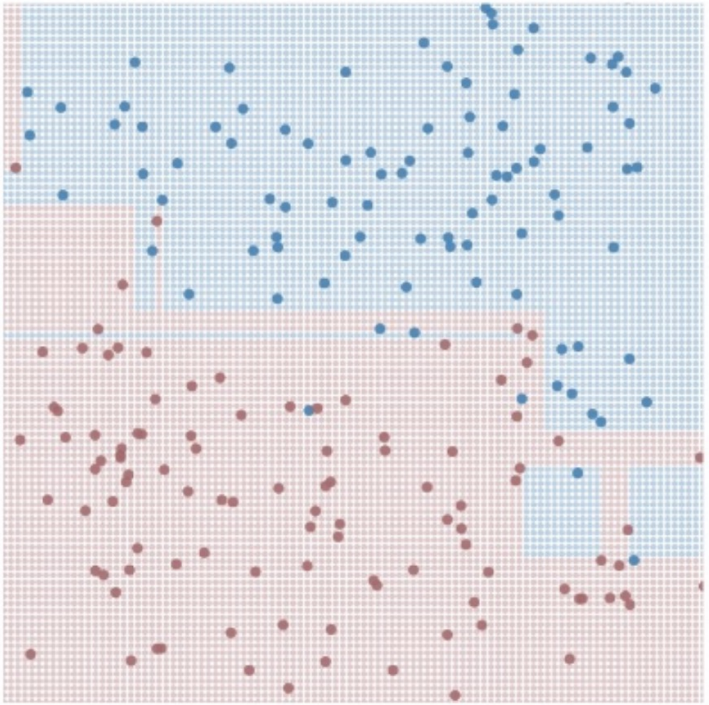
https://en.wikipedia.org/wiki/Bootstrap_aggregating



Building “Dumb” Models – Step 2

Maybe 10% of the points in training are causing errors with testing...

What happens if I build a tree based on a resampling of the points?

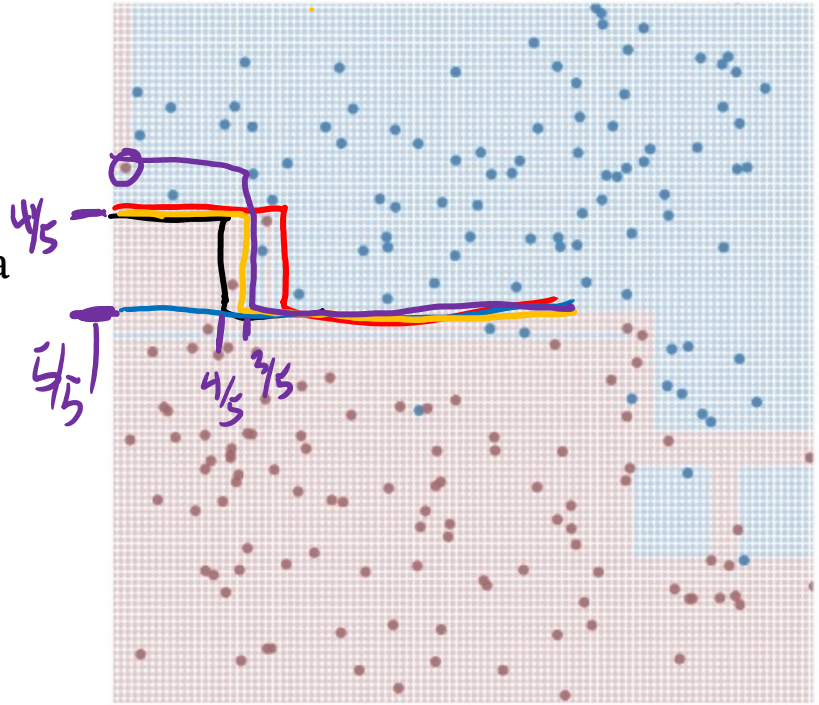


Building "Dumb" Models – Step 2

Maybe 10% of the points in training are causing errors with testing...

What happens if I build a tree based on a resampling of the points?

What happens if I build a bunch of trees based on a resampling of the points?



Building "Dumb" Models – Bagging (Bootstrap Aggregating

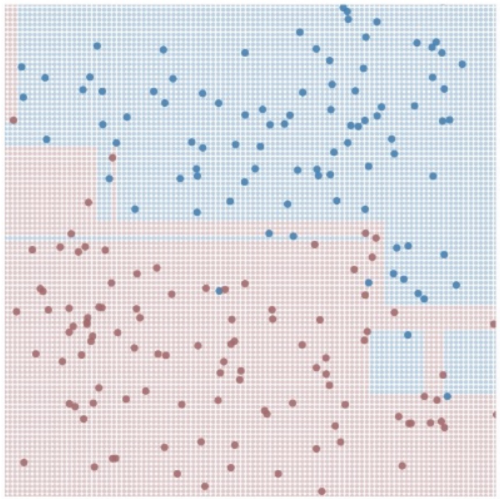
TRAINING

For M (# Models) iterations:

    Select N points with replacement

    Train a decision tree

    Hang on to that decision tree



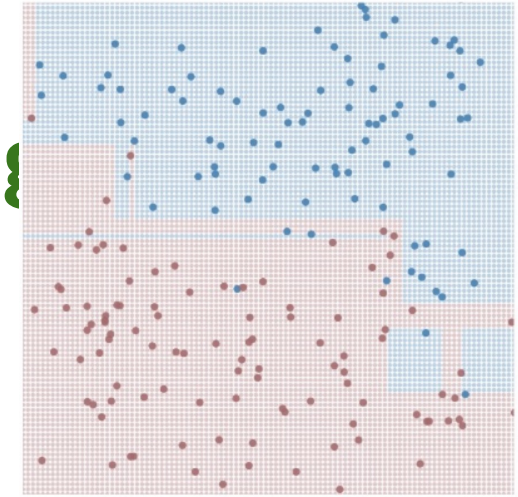
TESTING

For each test point:

    Ask all the decision trees what decision to make

    Accept Y (the best answer)

Building "Dumb" Models – Bagging



TRAINING

For M iterations:

    Select N points with replacement

    Train a decision tree

    Hang on to that decision tree

TESTING

For each test point:

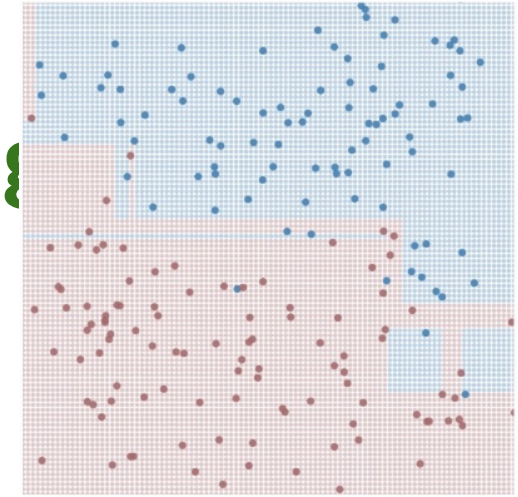
    Ask all the decision trees what decision to make

    Accept Y

Ensemble Learning

Build a group of models $H(x) = h_1(x), h_2(x), \ldots h_k(x)$

Building "Dumb" Models – Bagging



TRAINING

For K iterations:

    Select N points with replacement

    Train a decision tree

    Hang on to that decision tree

TESTING

For each test point:

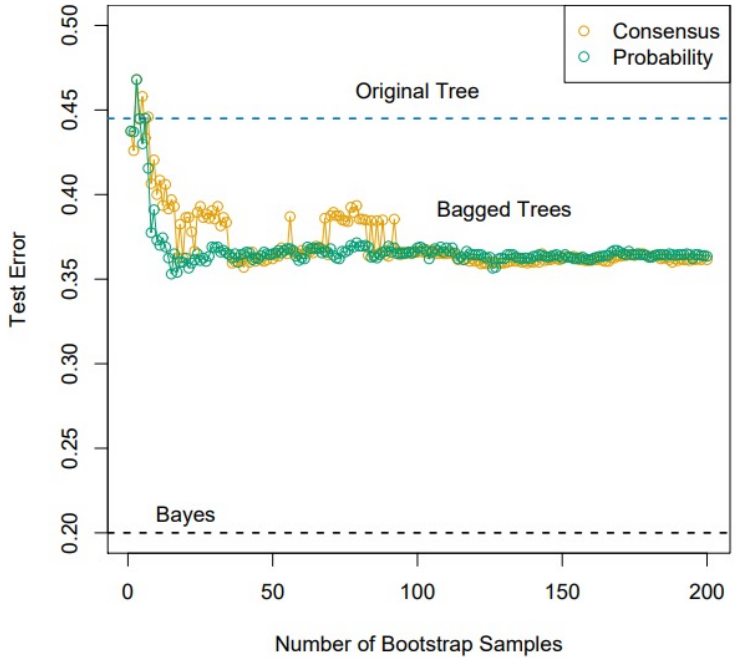
    Ask all the decision trees what decision to make

    Accept Y

Bagging – Appropriate number of models

In short – it depends

- Think of this as a stopping parameter
- You'll see numbers like 30, 50, 100 used a lot...



Bagging – Choosing Consensus

END

Binary decision from each tree

Take the *majority* decision

Thursday

Course Logistics

- Project: Groups due *now*!
  - As in about a couple minutes ago
    - If you didn't submit, *go do it right this instant. Watch lecture later.*
  - Project 2: Pitch and Project 2.1: Pitch Feedback instructions are now viewable on Canvas
- Problem Sets: Problem Set 1 Due 9/15

Guest Speaker Tuesday

- Nikhil Krishnaswamy – CSU
- Works at the intersection of Human Computer Communication (NLP, etc.) and HCI problems
- Part of the Institute for ~~Human~~ student-AI Teaming
    - An national NSF Funded Institute housed here at CU

Building "Dumb" Models – Bagging

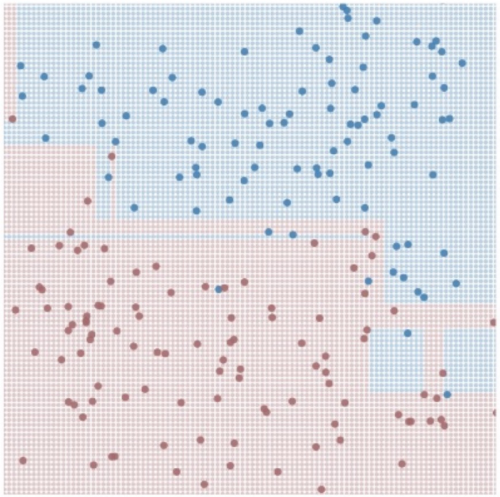
TRAINING

For K iterations:

    Select N points with replacement

    Train a decision tree

    Hang on to that decision tree



TESTING

For each test point:

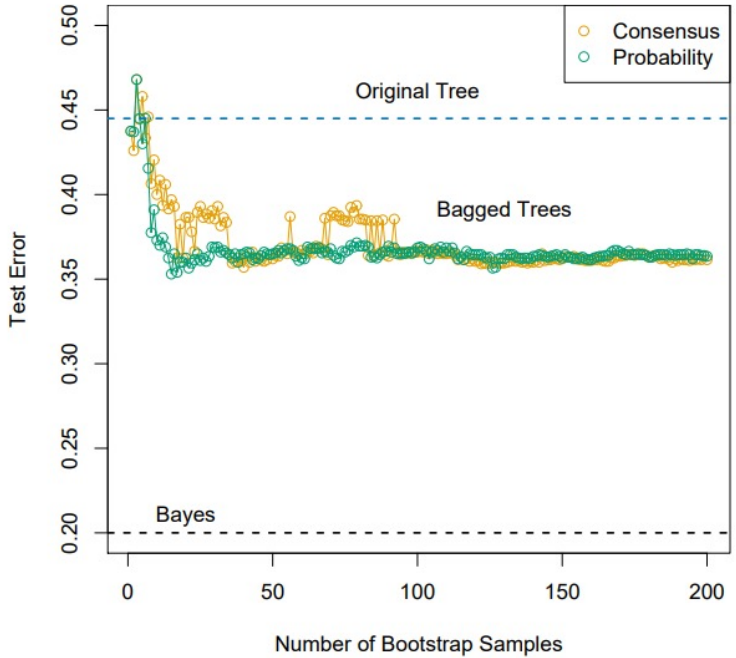
    Ask all the decision trees what decision to make

    Accept Y

Bagging – Appropriate number of models

In short – it depends

- Think of this as a stopping parameter
- You'll see numbers like 30, 50, 100 used a lot...

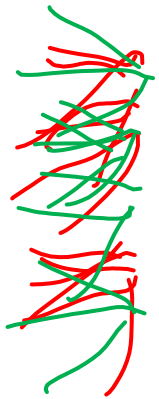


Bagging – Choosing Consensus

Binary decision from each tree

Take the *majority* decision

Ensemble Learning – Example (walkthrough)



X_1	X_2	X_3	X_4	X_5	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**	**Single Family?**
1200	1	1.5	1998	200,000	No
1800	2	2	1985	450,000	Yes
800	1	1	2017	250,000	No
2500	3	2	1975	500,000	No
2800	4	2.5	1983	400,000	Yes
…	…	…	…	…	

Ensemble Learning

Build a group of models $H(x) = h_1(x), h_2(x), \ldots h_k(x)$

$$H(x) = sign(\sum_{k=1}^{K} h_k(x))$$

Bagging – Benefits

Bagging – Benefits

Reducing the variance (overfitting)?

Bagging – Benefits

Reducing the variance (overfitting)

Variance – Sensitivity to *the training set*

Yes

Bagging – Benefits

Reducing the variance (overfitting)

Variance – Sensitivity to *the training set*

Yes

Reducing the bias (underfitting)?

Bagging – Benefits

Reducing the variance (overfitting)

Variance – Sensitivity to *the training set*

Yes

Reducing the bias (underfitting)

Bias – Sensitivity *to the model*

No

Ensemble Learning - Improvements

We have aggregated models helping reduce the variance, but how can we reduce the bias?

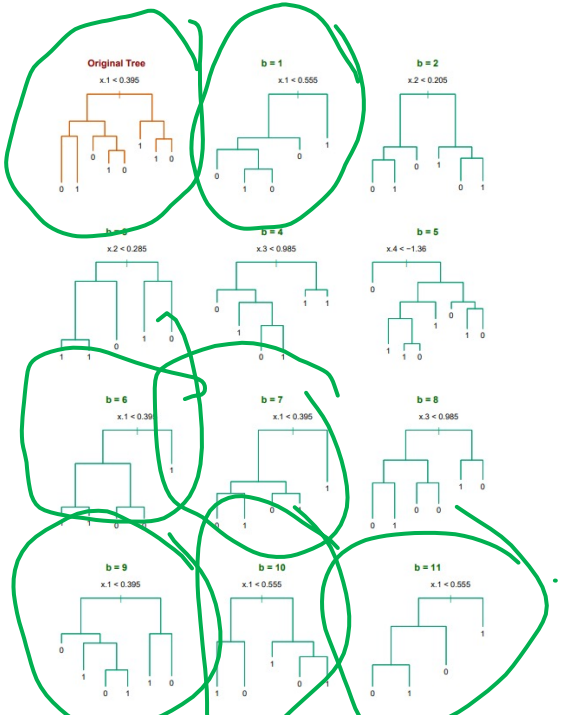
Bias – Sensitivity *to the model*

Ensemble Learning - Improvements

We have aggregated models helping reduce the variance, but how can we reduce the bias?

Bias – Sensitivity *to the model*

Looking at some example trees...

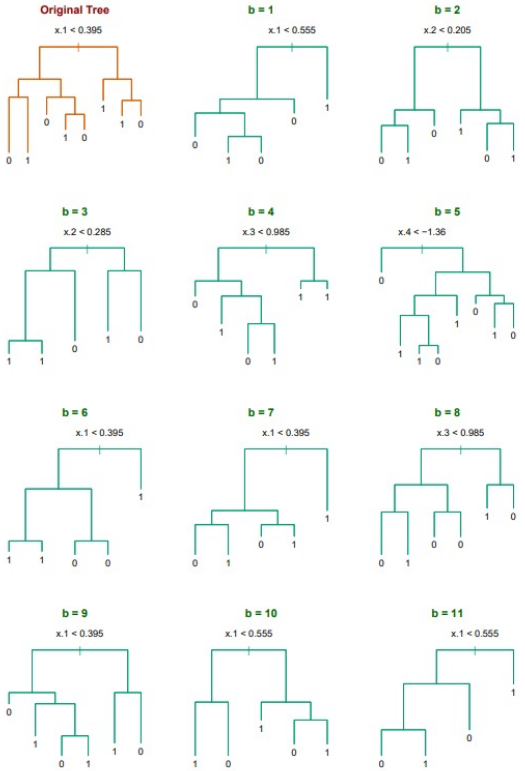


Ensemble Learning – Reducing Bias

The trees are demonstrating high correlation

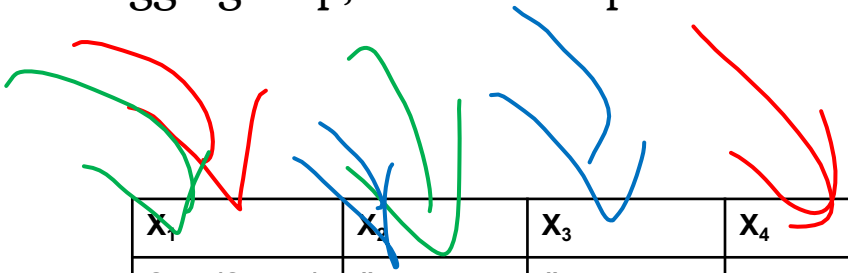
- Many of our trees are using an identical splitting point for the first decision!
- This correlation is a source of bias

We need to force the trees to care about different features



Ensemble Learning – Feature Subsets

Within the bagging loop, force each split to be made based on a *subset* of features



X_1	X_2	X_3	X_4	X_5	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**	**Single Family?**
1200	1	1.5	1998	200,000	No
1800	2	2	1985	450,000	Yes
800	1	1	2017	250,000	No
2500	3	2	1975	500,000	No
2800	4	2.5	1983	400,000	Yes
…	…	…	…	…	

Ensemble Methods– Random Forests

```
TRAINING
For M iterations:
        Select N’ points with replacement
        Train a decision tree:
                For each split:
                        Choose F features to split on:
                                Choose your best split from them
        Hang on to that decision tree
TESTING
For each test point:
        Ask all the decision trees what decision to make
        Accept Y
```

Random Forests – Choosing # Features (F)

In short – it depends!

- Treat this as a tuning parameter
  - Similar to K in KNN, λ in Ridge / Lasso Regression

Sqrt(|X|) is a good default

X_1	X_2	X_3	X_4	X_5	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**	**Single Family?**
1200	1	1.5	1998	200,000	No
1800	2	2	1985	450,000	Yes
800	1	1	2017	250,000	No
2500	3	2	1975	500,000	No
2800	4	2.5	1983	400,000	Yes
…	…	…	…	…	

Ensemble Methods

We’re building all of these ideas on “Dumb” models, but surely we can do better?

The goodness comes out through aggregation, but some models may be “better” than others

- Can we try to trust these models more?

Boosting

Interlude – A Weak Learner

Assumptions – binary classification problem
both classes are 50% likely

Expectations

Ensemble Methods – A Weak Learner

Assumptions - binary classification problem
both classes are 50% likely

Expectations - A random guess model gets 50% accuracy

$$err = \frac{1}{n} * \sum_{i1 \ \ldots \ N} I(yi \neq G(x_i))$$

Ensemble Methods – A Weak Learner

Assumptions - binary classification problem
both classes are 50% likely

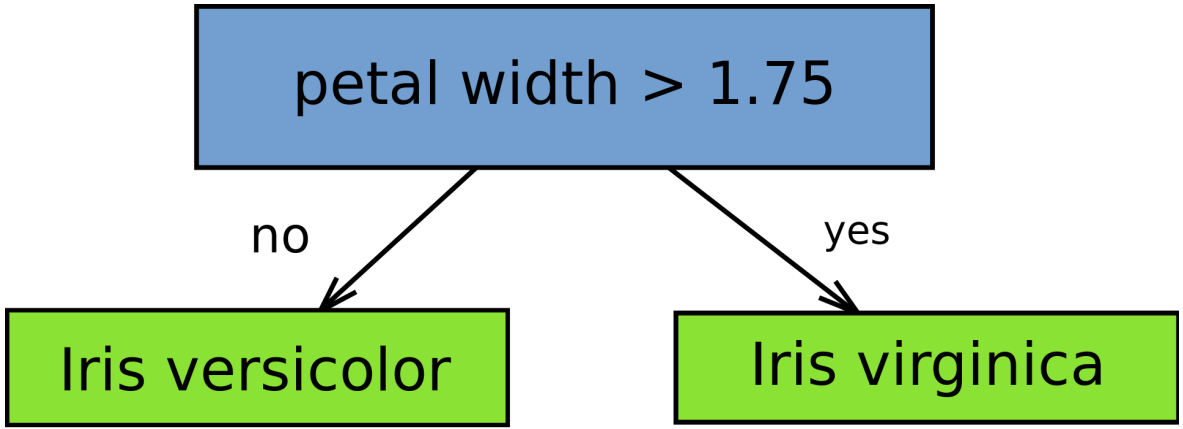
Expectations - A random guess model gets 50% accuracy

A “weak” learning model would perform slightly better than chance
We can only determine this through training accuracy / error

Weak Learner – Decision Stump

A 1-layer decision tree (a.k.a. a decision tree with a single split)

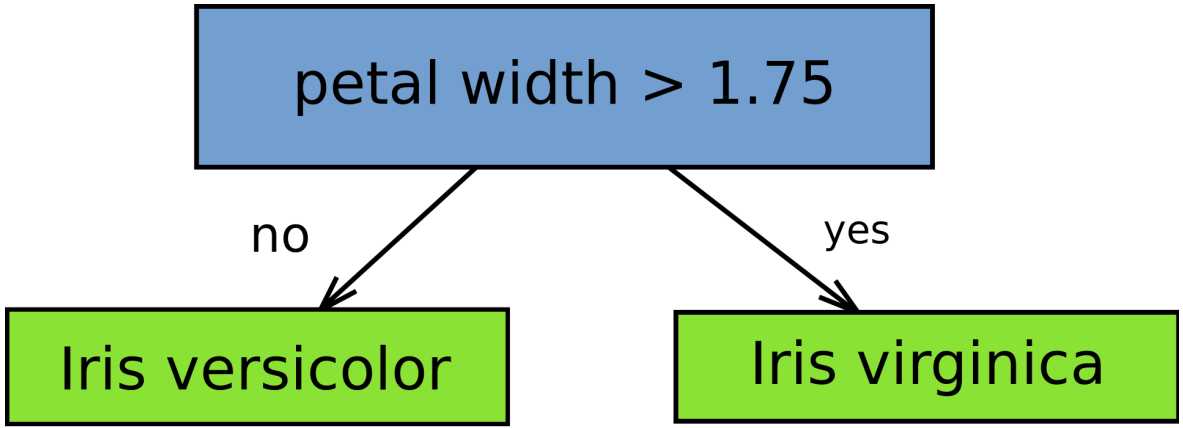
Entropy = $-p * \log_2(p) - (1-p) * \log_2(1-p)$



Weak Learner – Decision Stump

A 1-layer decision tree (a.k.a. a decision tree with a single split)

Entropy = $-p * \log_2(p) - (1-p) * \log_2(1-p)$



This could serve as a restriction on # dimensions, tree size for random forest!

Ensemble Methods - Boosting

All animals are equal, but some animals are more equal than others.

- George Orwell (Animal Farm, Chapter 10)

Ensemble Methods - Boosting

All animals are equal, but some animals are more equal than others.

- George Orwell (Animal Farm, Chapter 10)

All classifiers are equal, but some classifiers are more equal than others.

In an ensemble classifier, can we favor some of the individual classifiers more than others?

Boosting – Favoring Certain Points

Create a bunch of classifiers

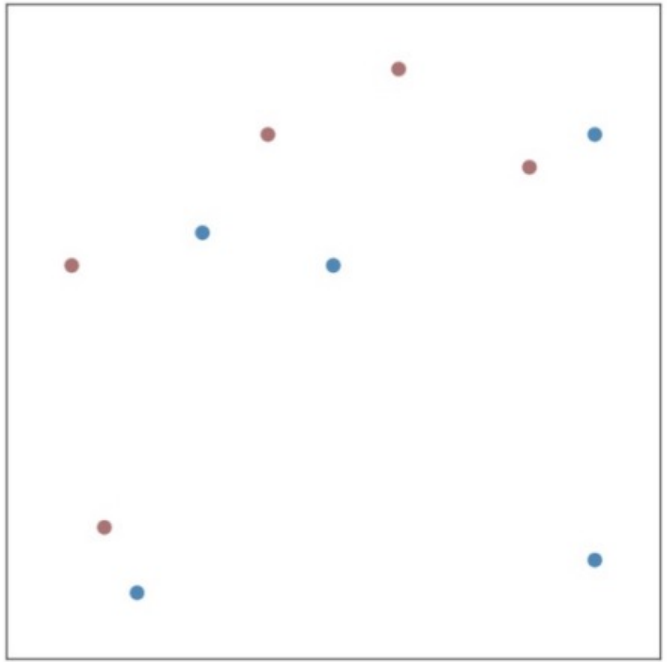
Weight each model k based on how accurate it is

$$H(x) = sign(\sum_{k=1}^{K} h_k(x))$$

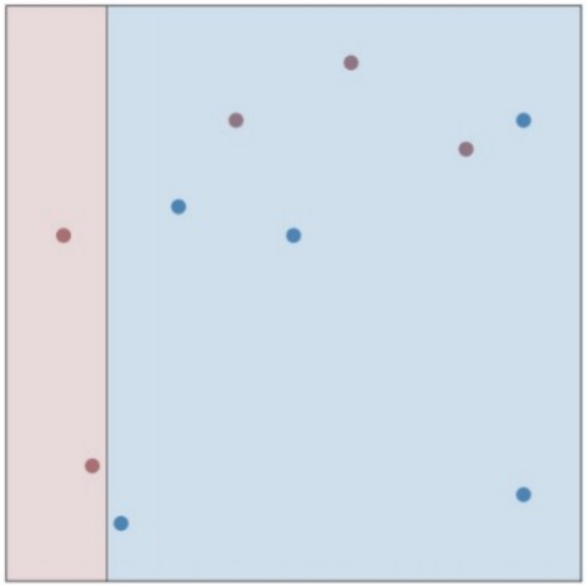
becomes

$$H(x) = sign(\sum_{k=1}^{K} \alpha_k h_k(x))$$

Weights for Individual Classifiers

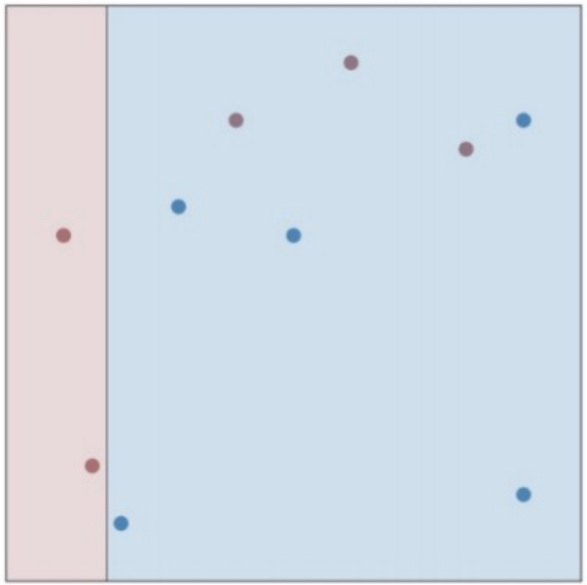


Weights for Individual Classifiers



Weights for Individual Classifiers

$$err = \sum_{i=1}^{m} I(yi \neq h(x_i))$$

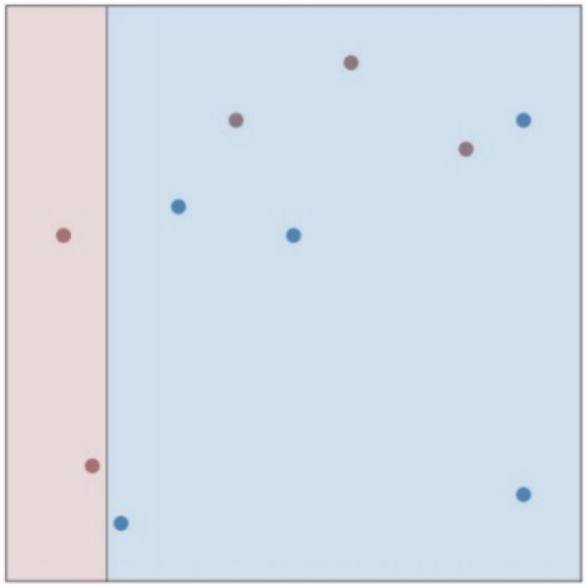


Weights for Individual Classifiers

$$err = \sum_{i=1}^{m} I(yi \neq h(x_i))$$

Err = 3

Error range = [0, ∞]

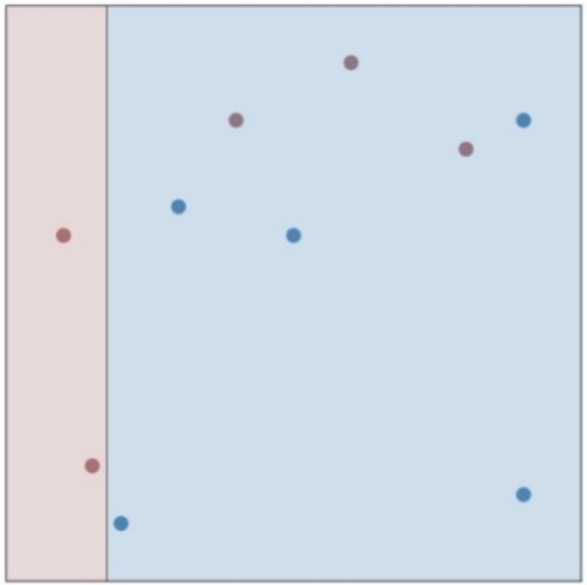


Weights for Individual Classifiers

$$err = \frac{\sum_{i=1}^{m} I(yi \neq h(x_i))}{m}$$

Err = 0.3

Error range = [0, 1]



Weights for Individual Classifiers

Option 1: Linear weight based on error

$$\alpha_k = 1 - err$$

Weights for Individual Classifiers

Option 1: Linear weight based on error

$$\alpha_k = 1 - err$$

A 10% accuracy model (i.e. incorrect class 9 out of 10 times)

$\alpha_k = .1$

Weights for Individual Classifiers

Option 1: Linear weight based on error

$$\alpha_k = 1 - err$$

A 10% accuracy model (i.e. incorrect class 9 out of 10 times)

$\alpha_k = .1$

Need a weight function that does better…

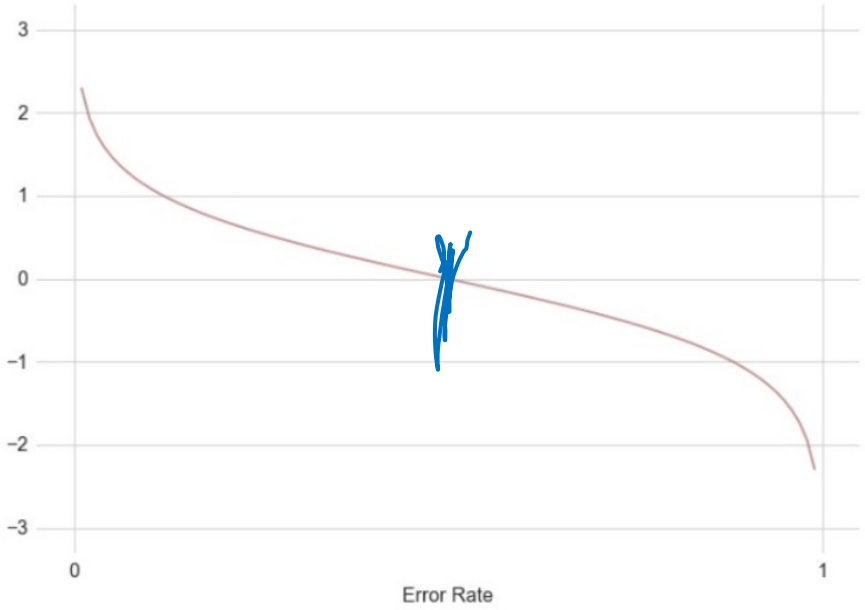
Exponential Weights for Individual Classifiers

$$\alpha_k = \frac{1}{2}\log\left(\frac{1 - errk}{err_k}\right)$$

Grows for low error

Approaches 0 for error .5 (chance)

Negative penalty for high error

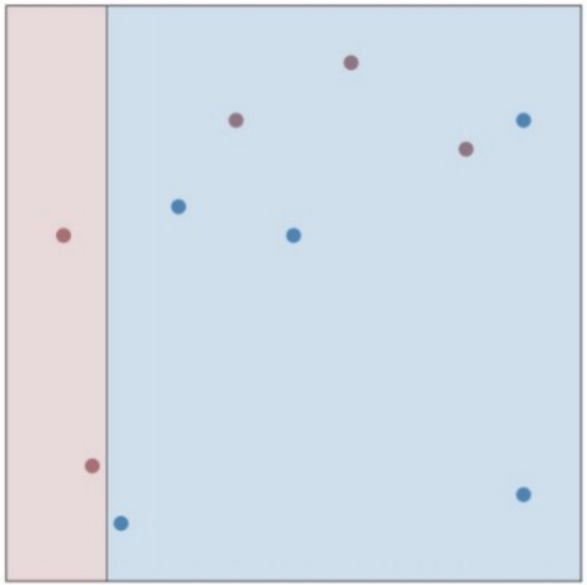


Weights for Individual Classifiers

$$err = \frac{\sum_{i=1}^{m} I(yi \neq h(x_i))}{m}$$

Err = 0.3

$$\alpha_k = \frac{1}{2}\log(\frac{1 - err_k}{err_k})$$



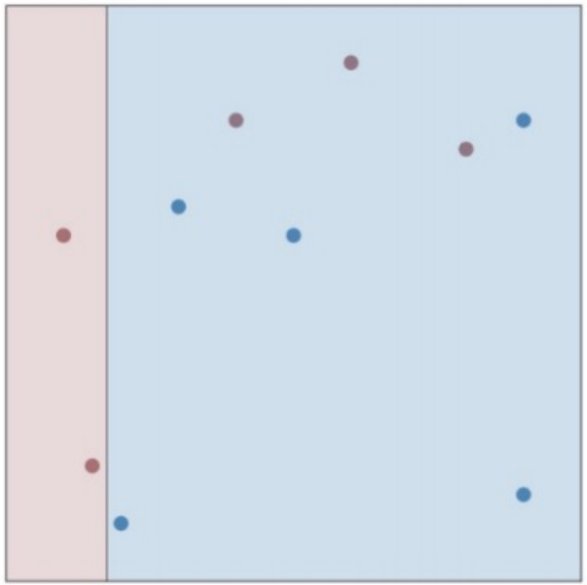
Weights for Individual Classifiers

$$err = \frac{\sum_{i=1}^{m} I(yi \neq h(x_i))}{m}$$

Err = 0.3

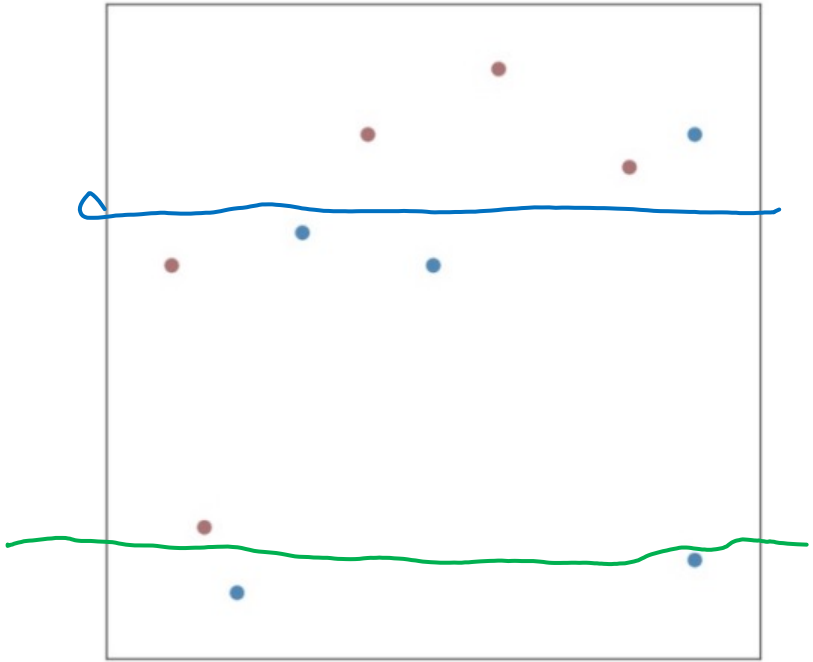
$$\alpha_k = \frac{1}{2}\log(\frac{1 - err_k}{err_k})$$

α_k = 0.42



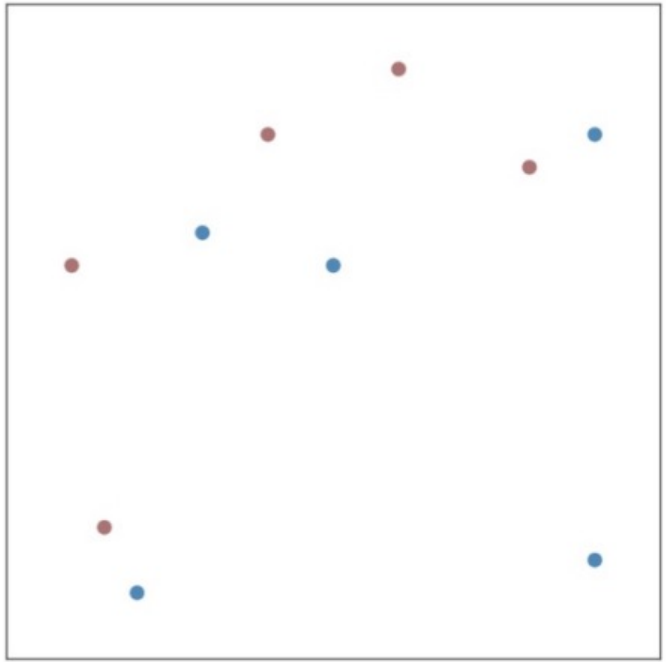
Weights for Individual Classifiers

Could we have done a better split on this model than 70% accuracy (30% error)?



Weights for Individual Classifiers

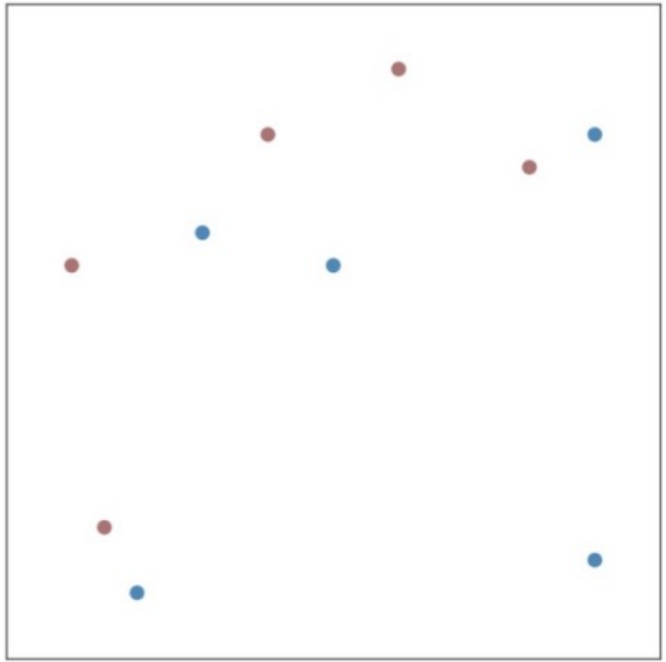
What's going to happen when we create 30, 50, 100 classifiers here?



Weights for Individual Classifiers

What's going to happen when we create 30, 50, 100 classifiers here?

Entropy will always look the same for new models!
Even if we use a random forest, we have 2 dimensions!



Ensemble Methods - Boosting

All animals are equal, but some animals are more equal than others.

- George Orwell (Animal Farm, Chapter 10)

Ensemble Methods - Boosting

All animals are equal, but some animals are more equal than others.

- George Orwell (Animal Farm, Chapter 10)

All points are equal, but some points are more equal than others.

Can we favor points that are proving difficult to accurately classify?

Boosting – Favoring Certain Points

Iteratively create a bunch of classifiers

Weight each model *k* based on how accurate it is

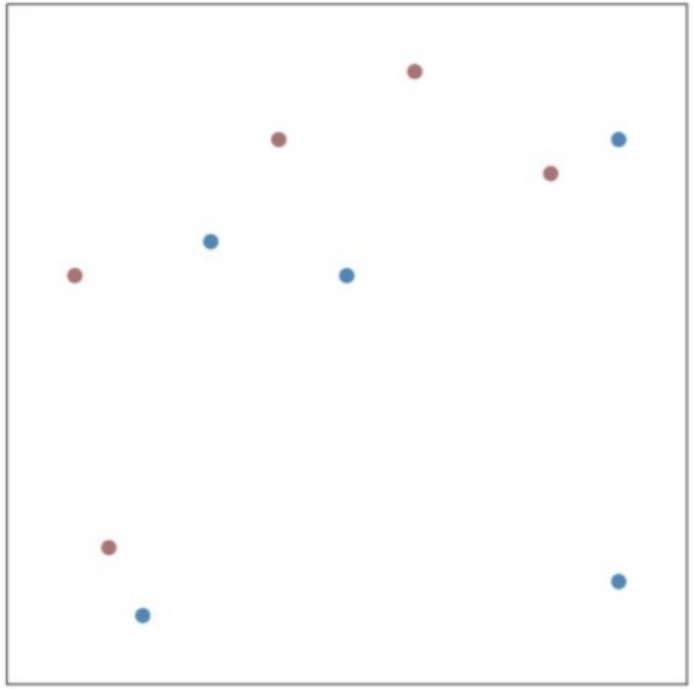
$$H(x) = sign(\sum_{k=1}^{K} \alpha_k h_k(x))$$

Weight each point based on how well it was classified for $h_k(x)$

If we classified it correctly, it was easy, lower the weight

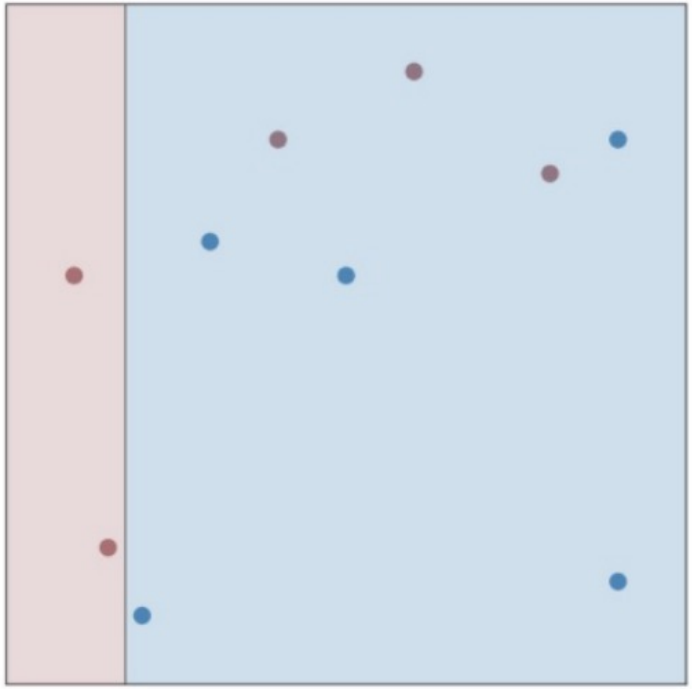
If we classified incorrectly, it was hard, raise the weight

Boosting – Favoring certain Points

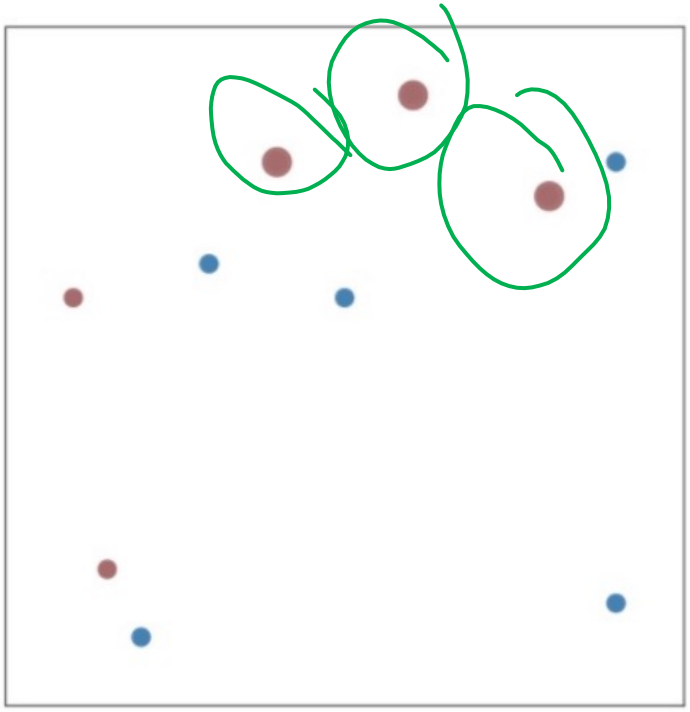


$$\text{err}_1 = 0.30$$

$$\alpha_1 = 0.42$$

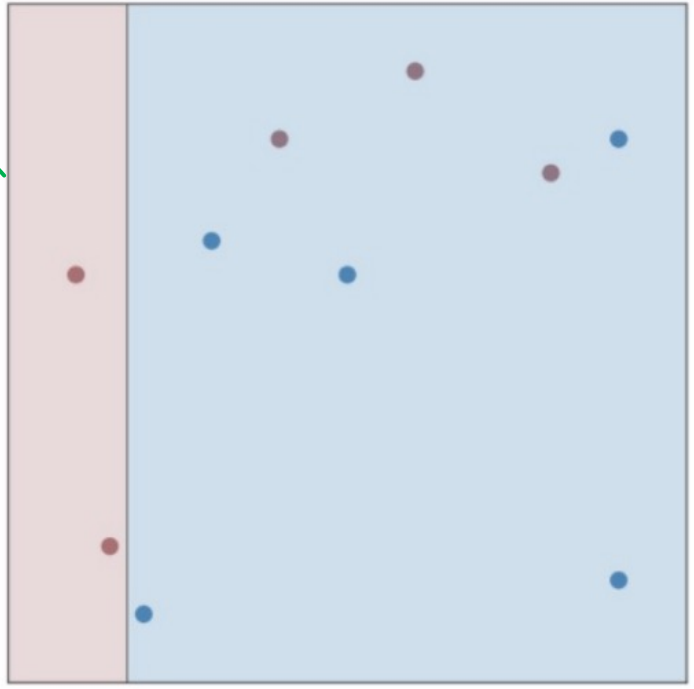


Boosting – Favoring certain Points



$$\text{err}_1 = 0.30$$

$$\alpha_1 = 0.42$$



Initializing our Weights

Assume all points are equal

$$w_i = 1/n$$

Updating our Weights

Up weight for incorrect
Down weight for correct
Based on accuracy of the model

Updating our Weights – Adaboost

Up weight for incorrect

Down weight for correct

Based on accuracy of the model

$$w_i = \frac{w_i}{Z_k} * \exp(-\alpha_k y_i h_k(xi))$$

*Z_k *is a normalization factor – we can just normalize afterwards instead*

$$\sum_{i=1}^{n} w_i = 1$$

Adaboost in practice

TRAINING
Initialize weights
For k = 1 → K

1) Fit Classifier
2) Compute error
3) Compute weight for classifier k
4) Update weights

TESTING
For test point

1) Output $H(x) = sign(\sum_{k=1}^{K} \alpha_k h_k(x))$

Fitting Classifier

Entropy = $-p * \log_2(p) - (1-p) * \log_2(1-p)$

p = *fraction of positive points*

$$p = \frac{\sum_{i=1}^{n} I(yi=1)}{n}$$

Fitting Classifier

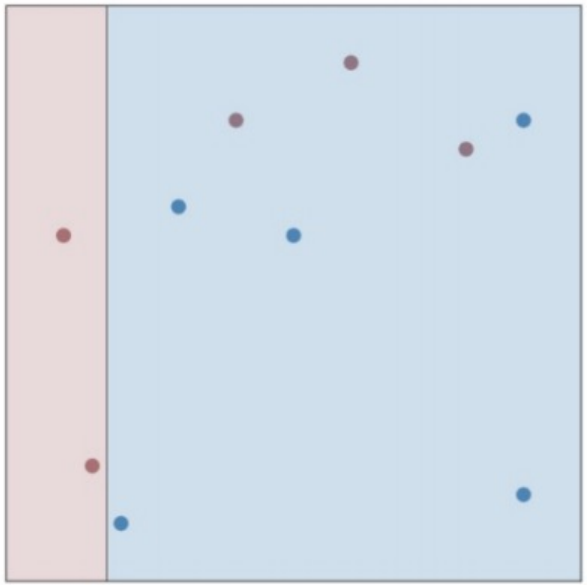
Entropy = $-p * \log_2(p) - (1-p) * \log_2(1-p)$

~~p = *fraction of positive points*~~

~~$p = \frac{\sum_{i=1}^{n} I(yi=1)}{n}$~~

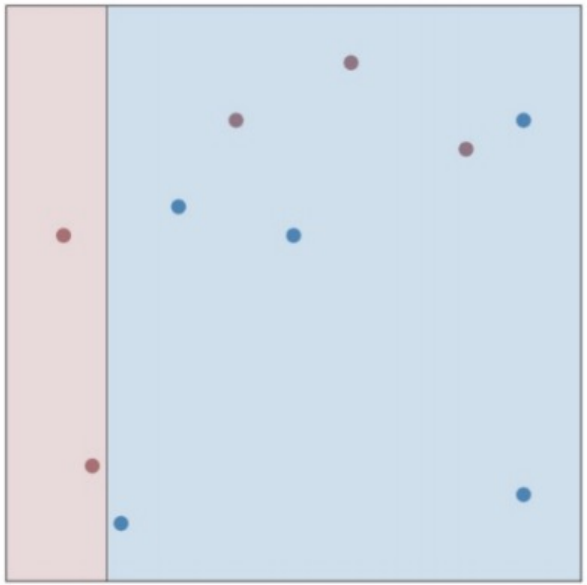
p = *weighted fraction of positive points*

$$p = \frac{\sum_{i=1}^{n} w_i I(yi=1)}{n \sum_{i=1}^{n} w_i}$$



Weights for Individual Classifiers

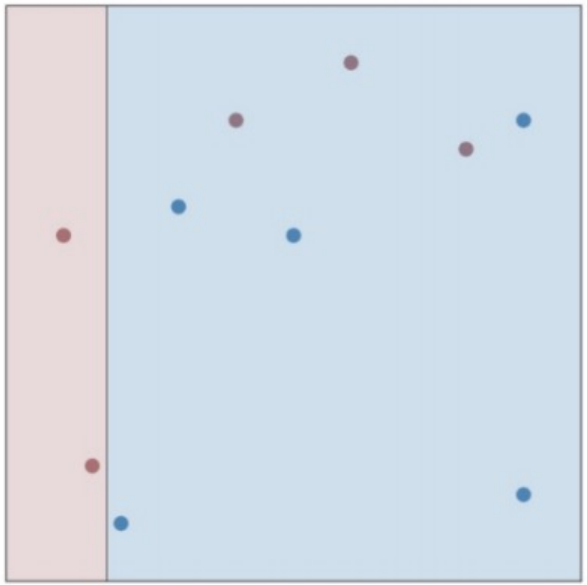
$$err = \frac{\sum_{i=1}^{m} I(yi \neq h(x_i))}{m}$$



Weights for Individual Classifiers

$$\sout{err = \frac{\sum_{i=1}^{m} I(yi \neq h(x_i))}{m}}$$

$$err = \frac{\sum_{i=1}^{m} w_i I(yi \neq h(x_i))}{\sum_{i=1}^{m} w_i}$$



Adaboost in practice

TRAINING
Initialize weights
For k = 1 → K

1) Fit Classifier
2) Compute error
3) Compute weight for classifier k
4) Update weights

TESTING
For test point

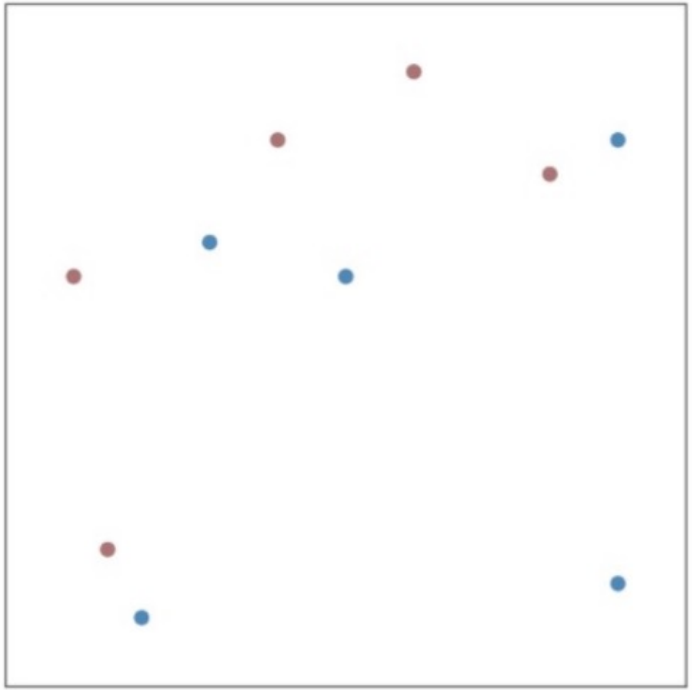
1) Output $H(x) = sign(\sum_{k=1}^{K} \alpha_k h_k(x))$

Adaboost – Working Example

First Decision Stump

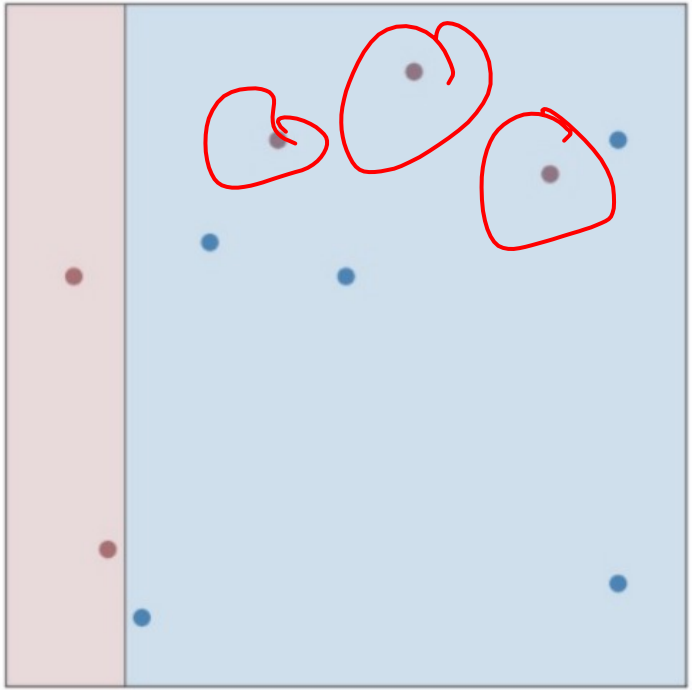
Adaboost – Working Example

First Decision Stump



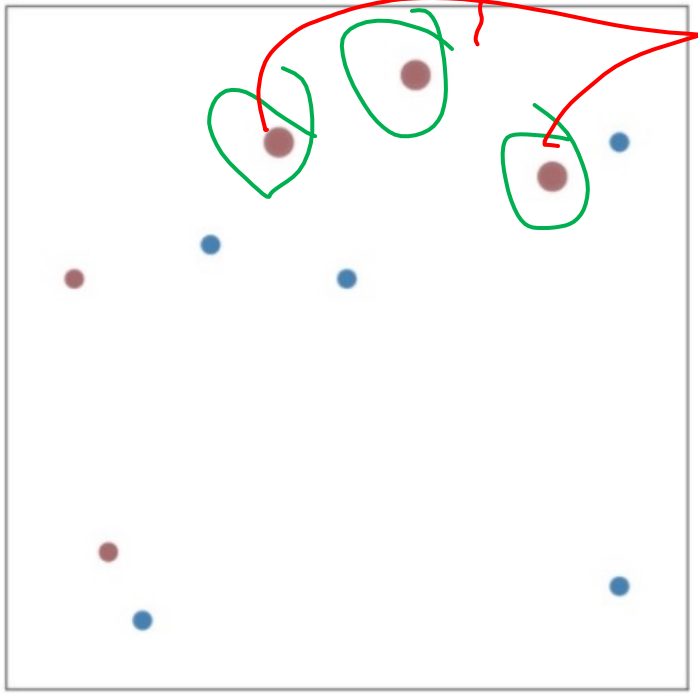
$$\text{err}_1 = 0.30$$

$$\alpha_1 = 0.42$$



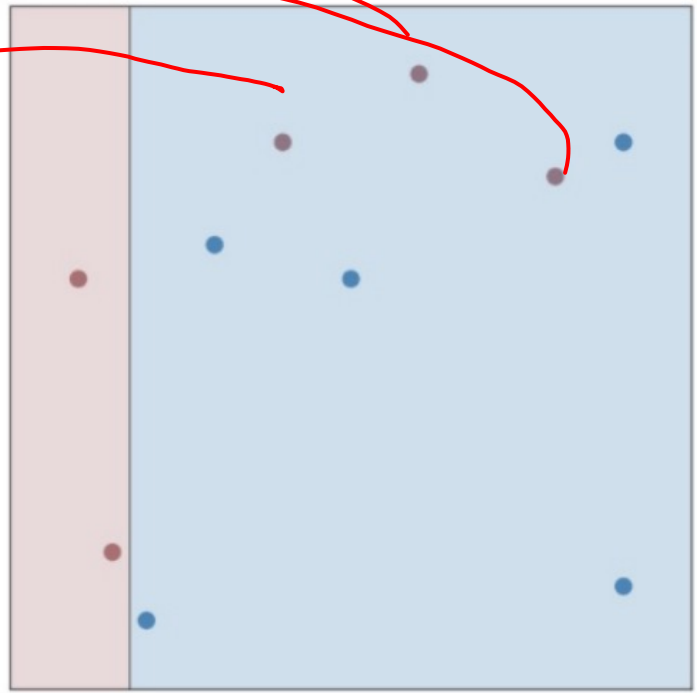
Adaboost – Working Example

First Decision Stump



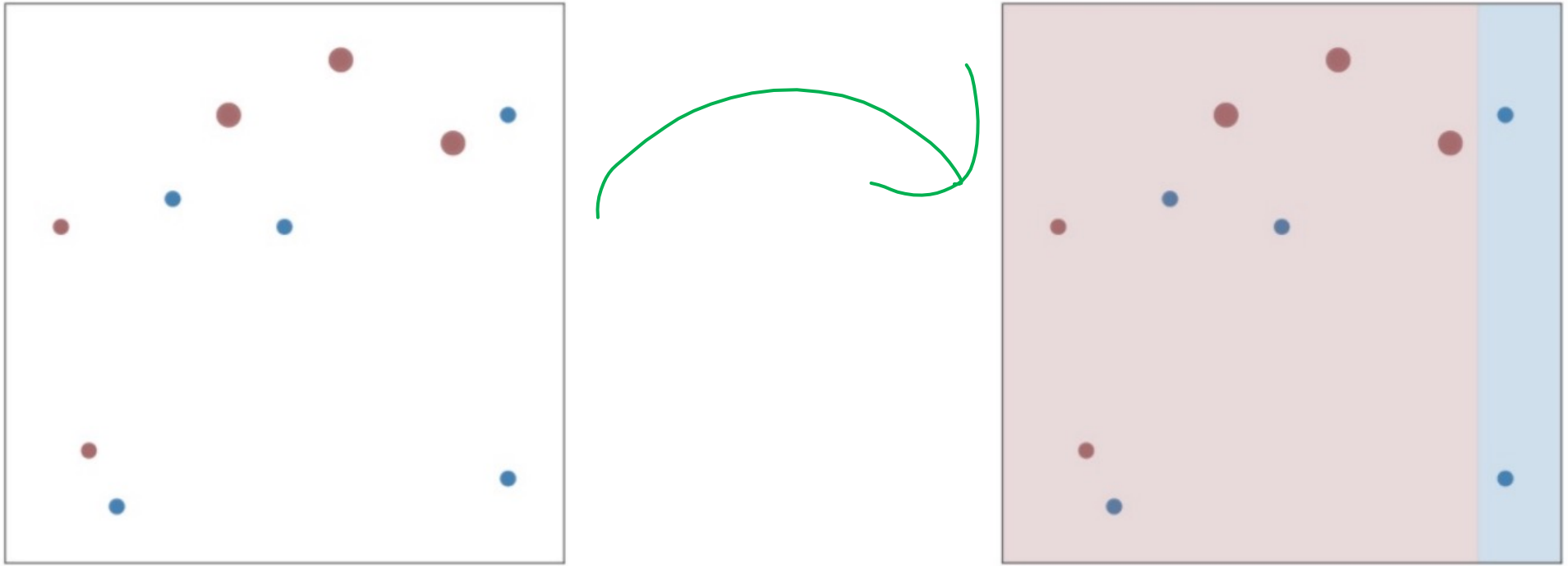
$$\text{err}_1 = 0.30$$

$$\alpha_1 = 0.42$$



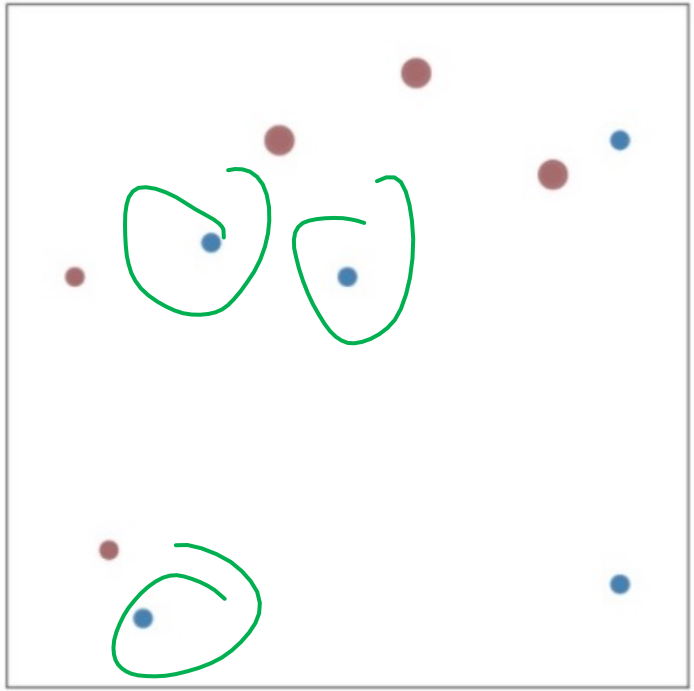
Adaboost – Working Example

Second Decision Stump



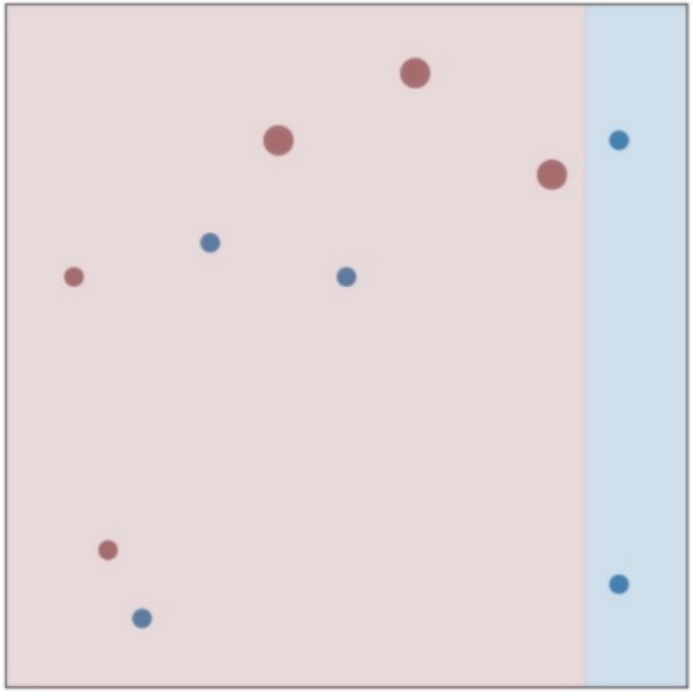
Adaboost – Working Example

Second Decision Stump



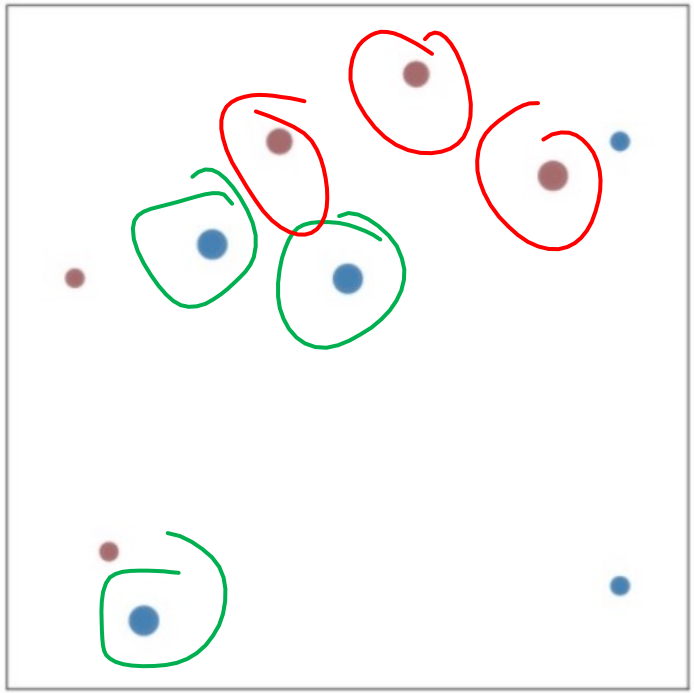
$$\text{err}_2 = 0.21$$

$$\alpha_2 = 0.65$$



Adaboost – Working Example

Second Decision Stump



$$\text{err}_2 = 0.21$$

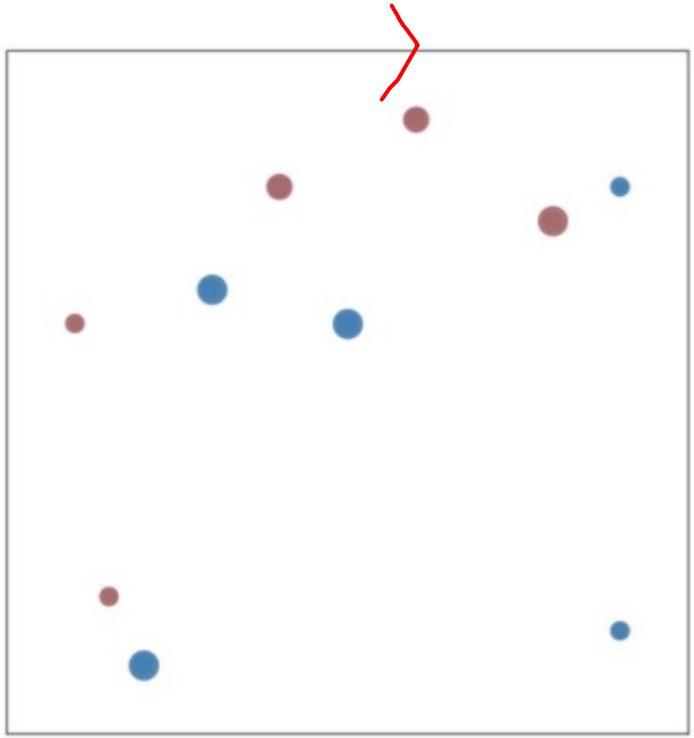
$$\alpha_2 = 0.65$$

Adaboost – Working Example

Third Decision Stump

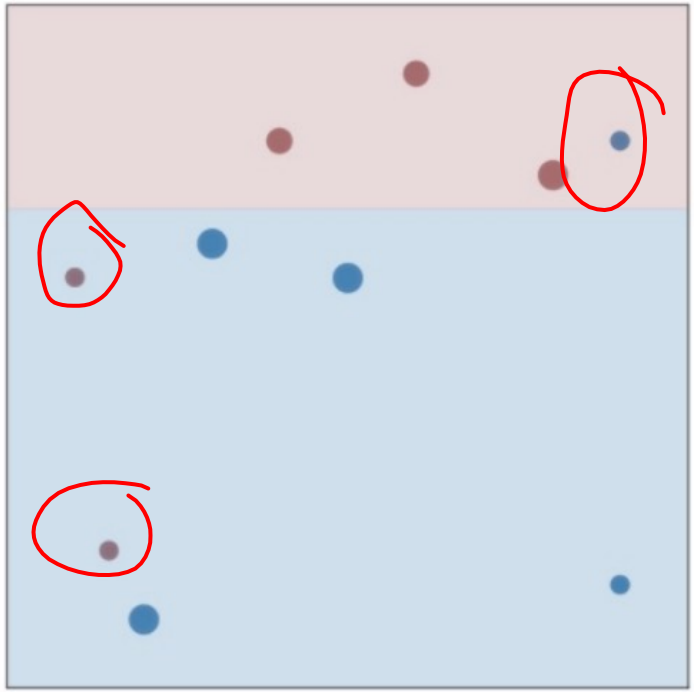
Adaboost – Working Example

Third Decision Stump



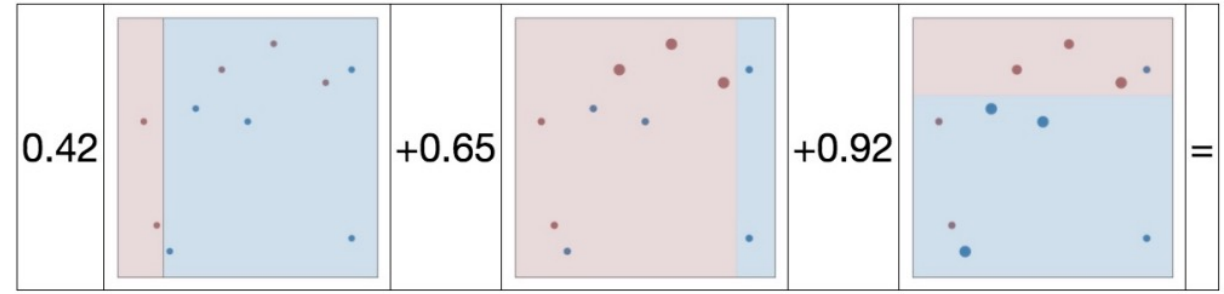
$$\text{err}_3 = 0.14$$

$$\alpha_3 = 0.92$$



Adaboost – Working Example

Final Ensemble Model



Boosting - Benefits

Reducing the variance?

Variance – Sensitivity to *the training set*

Reducing the bias?

Bias – Sensitivity *to the model*

Boosting – Benefits

Reducing the variance

Variance – Sensitivity to *the training set*

Yes

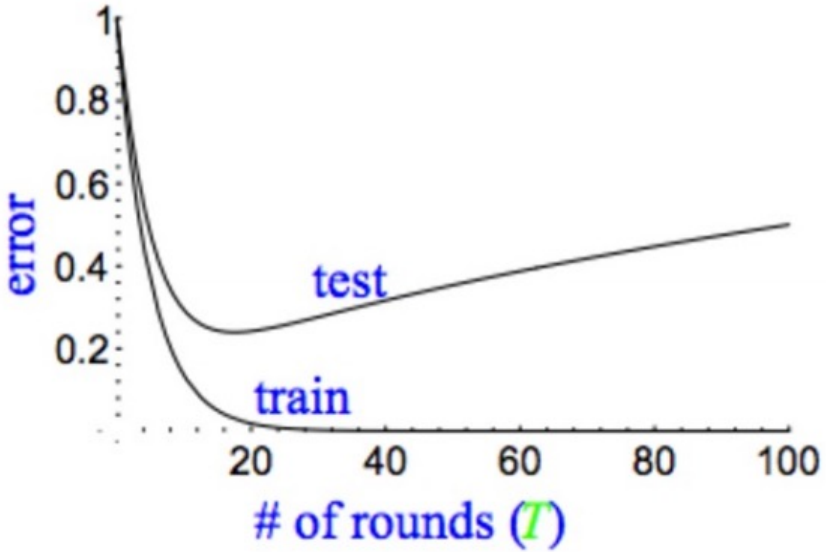
Reducing the bias

Bias – Sensitivity *to the model*

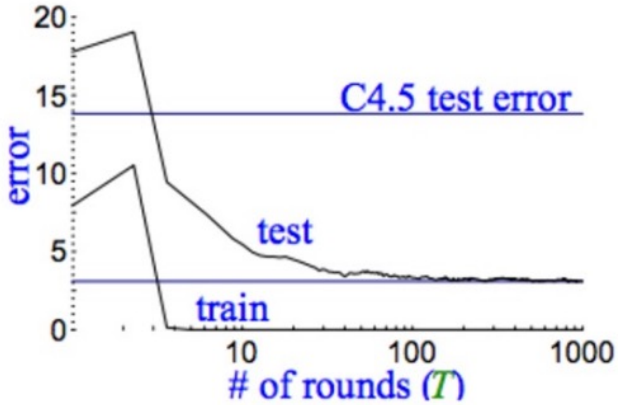
Yes

Model Performance

Traditional Models



Boosted Models



(boosting C4.5 on "letter" dataset)

Benefits of Boosting

Fast

Easy to Train

Little tuning (K)

Weak-classifier Independent

Boosting “gotchas”

Data-dependent

Weak-learner-dependent

- Can overfit with too strong of a learner
  - α can grow too large
- Can underfit with too weak of a learner

Comparison Activity

Deciding More Complex Things

Binary Classification

Classified As *Ground Truth*	**Cancer**	**Not Cancer**
Cancer	True Positive (Hit)	False Negative (Miss)
Not Cancer	False Positive (False Alarm)	True Negative (Correct Rejection)

Multi-Class Example

Multi-Class Example - Dogs

Height	**Weight**	**Color**	***Dog Type***
22 in	62 lbs	Black	*Black Lab*
15 in	45 lbs	White, brown	*English Bulldog*
9 in	19 lbs	Black	*Dachshund*

Classification → Binary Classification

Is this tweet positive or negative?

$Y = \{pos, neg\}$

$y = \text{argmax}_c\ p(c|x, D)$

Is this tweet positive or not positive?

$Y = \{1, 0\}$

$p(y=0 \mid x, D) = 1 - p(y=1 \mid x, D)$

Classification vs. Binary Classification

$Y = \{1, 2, 3, ..., n\}$

$y = \text{argmax}_c \, p(c|x, D)$

Determining the decade of birth

Species of tree

Type of road sign

$Y = \{1, 0\}$

$p(y=0 \mid x, D) = 1 - p(y=1 \mid x, D)$

Positive or not positive tweets

Is the road clear of obstructions?

90's kid?

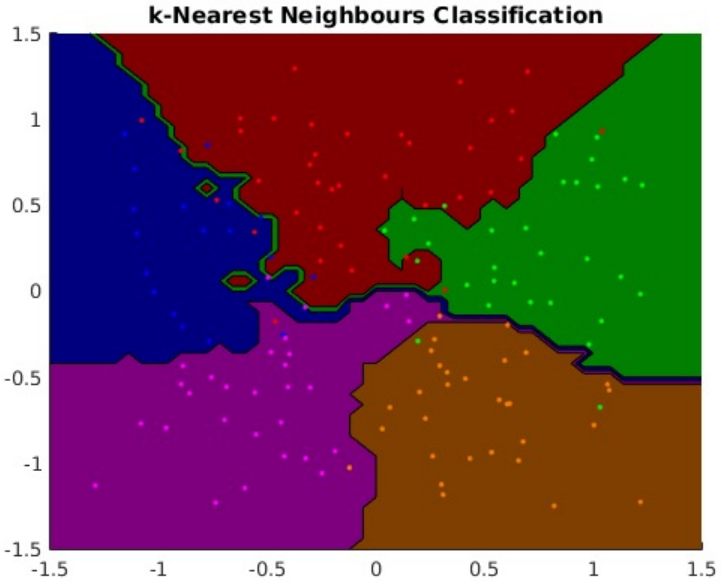
Classification vs. Binary Classification

Some methods are inherently capable of multi-class classification

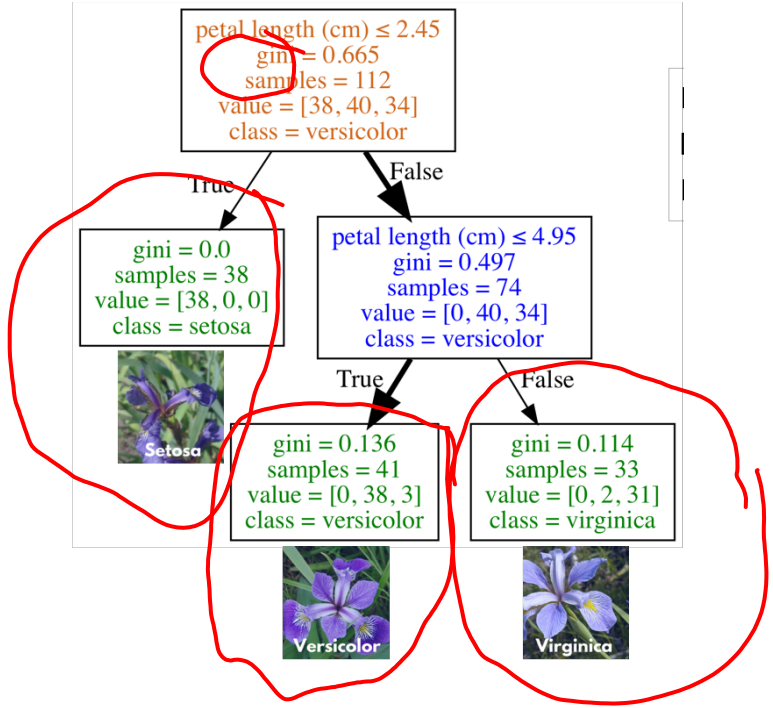
Naïve Bayes - $y = \text{argmax}_c\ p(c|x, D)$

KNN – find $N_k(x, D)$, assign the majority (plurality) label to x

Multi-Class Classification - KNN



Multi-Class Classification – Decision Trees



Interlude – Types of Data

Continuous Numeric

- Maps to a “real” number
- Distances between numbers have meaning

Discrete Numeric

- Maps to a discrete, infinite set (e.g. integers)
- Distances between numbers have meaning

Ordinal

- Entries have an *order*, but distances lose meaning

Categorical

- Entries are each unique, but don’t have further restrictions

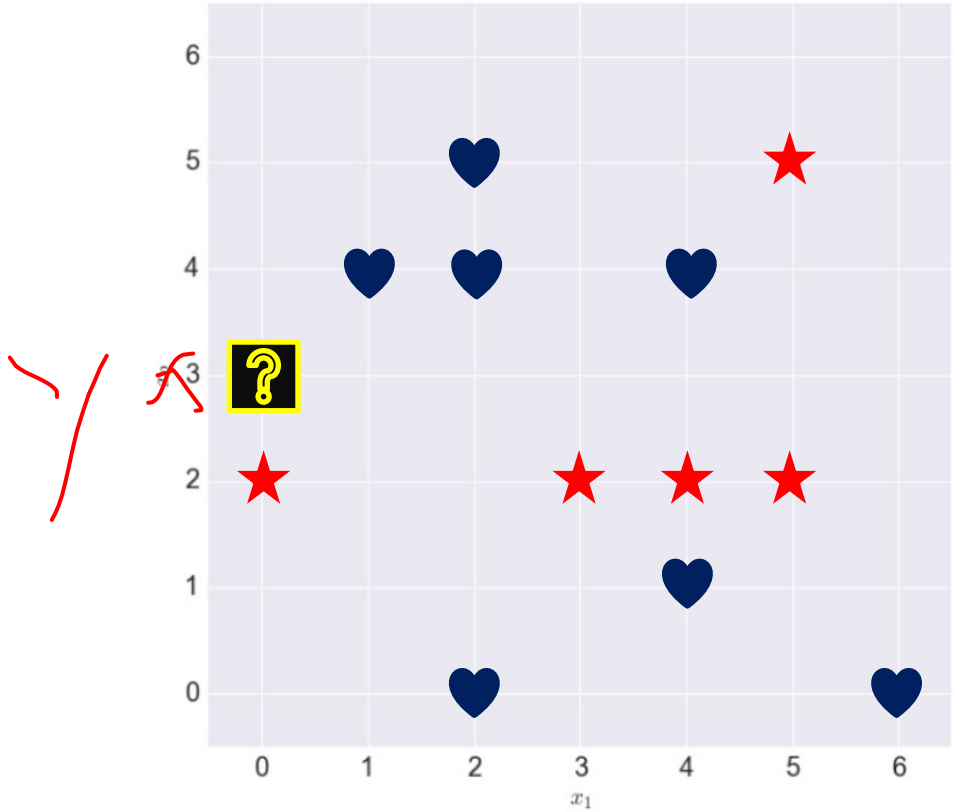
“Classify” a Numeric Problem?

- …

It’s not Classification, it’s Regression!

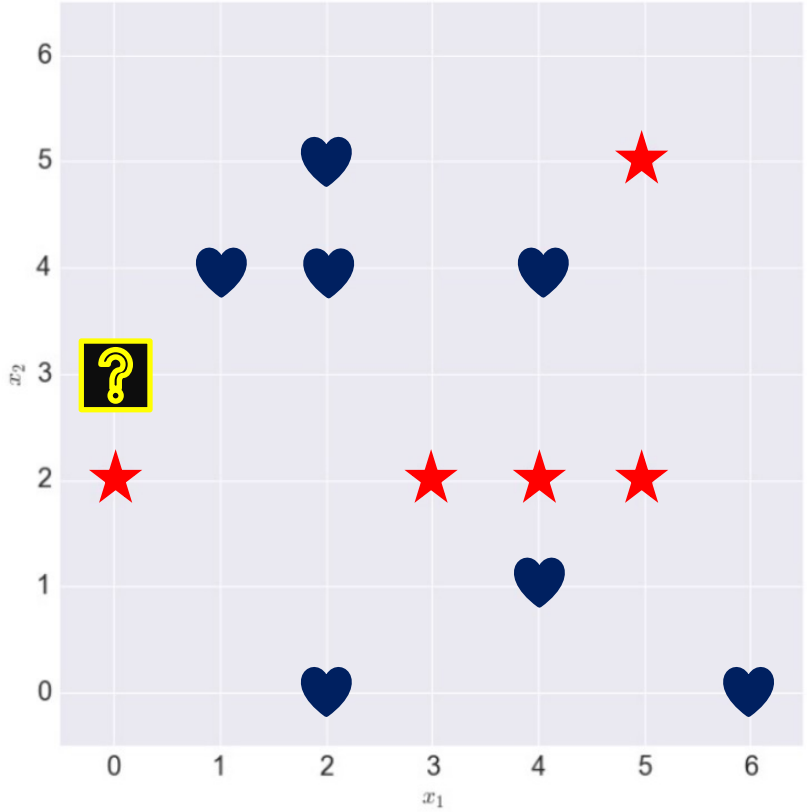
Regression – K-Nearest Neighbors

Our Y values are now Numeric



Regression – K-Nearest Neighbors

Our Y values are now Numeric

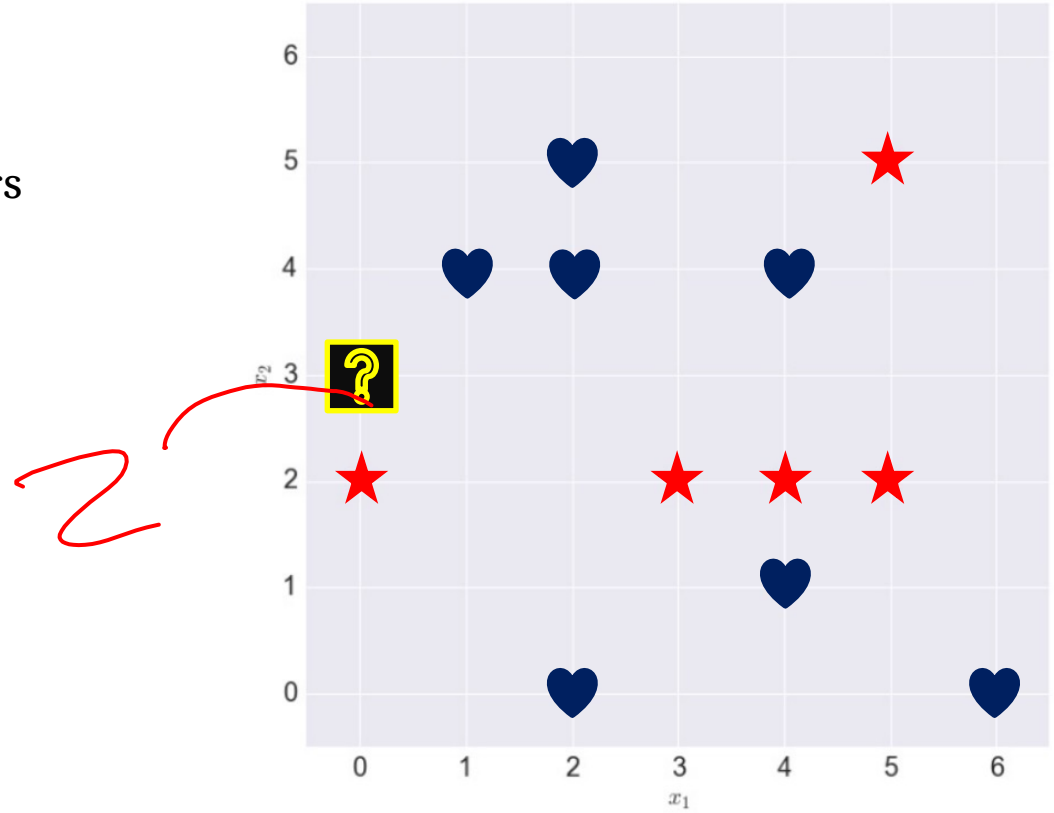


Regression – K-Nearest Neighbors

Our Y values are now Numeric

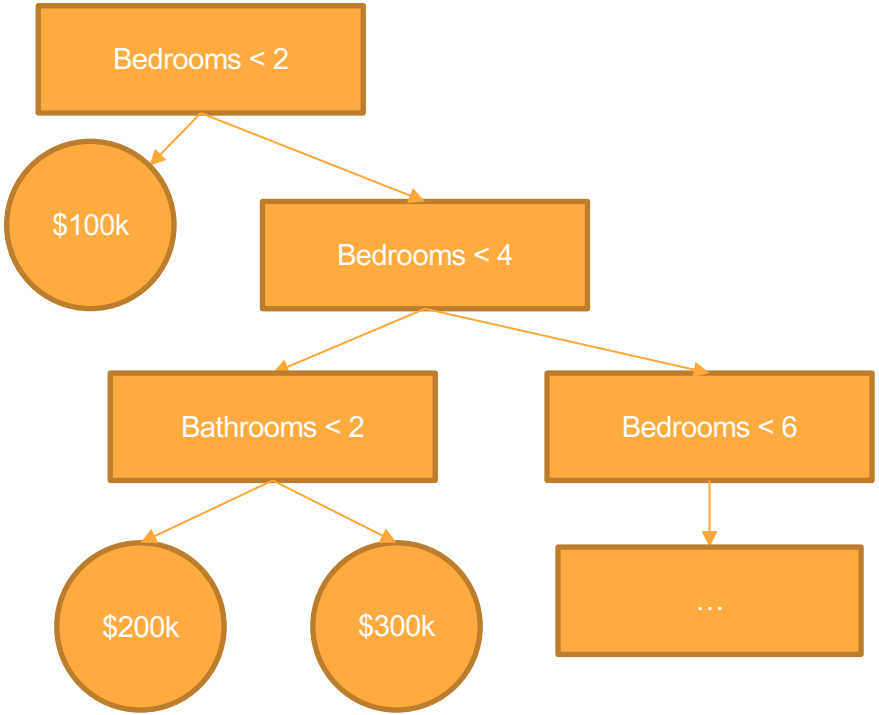
Step 1) Find your nearest neighbors

Step 2) Compute the mean

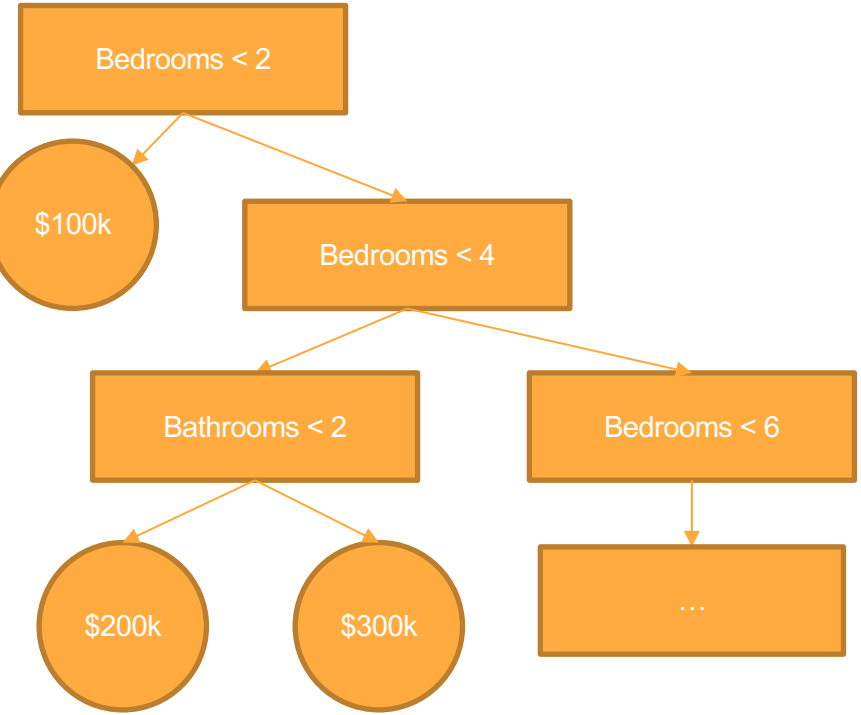


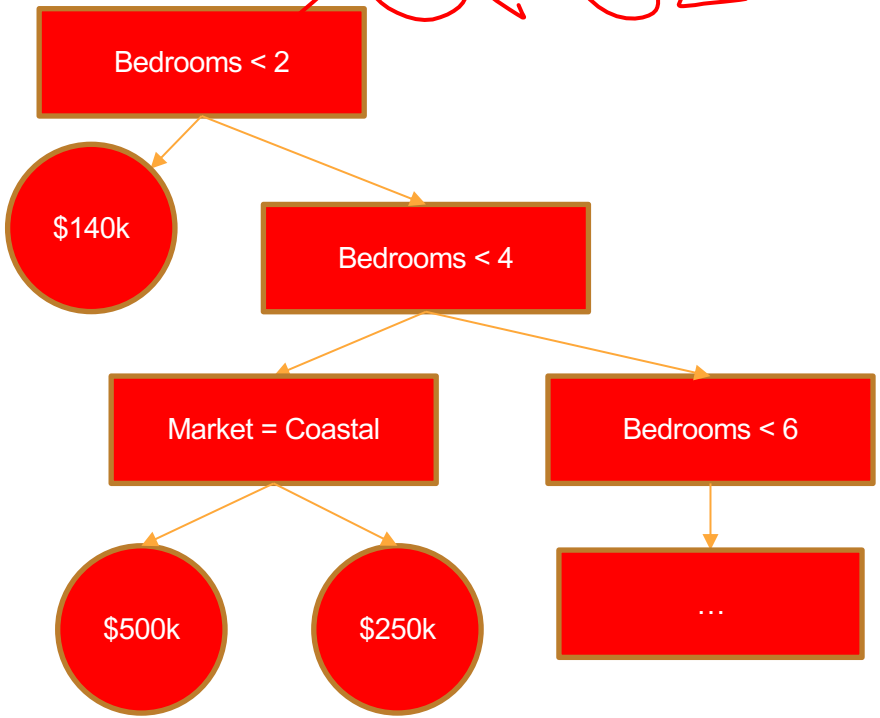
Housing Market

Housing Market – Decision Tree



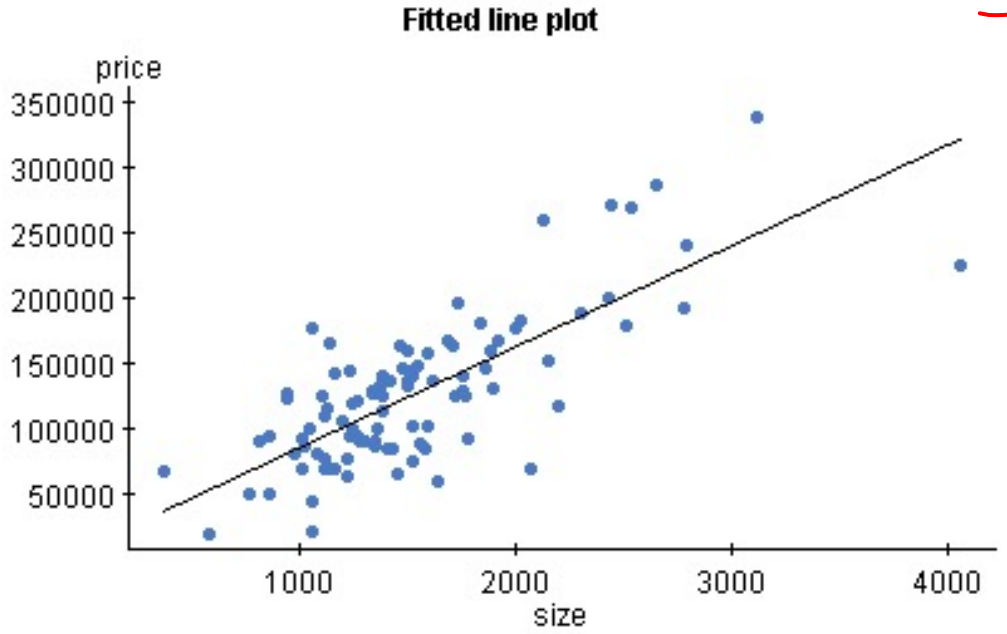
Housing Market – Decision Tree forest





Linear Regression – The Basics

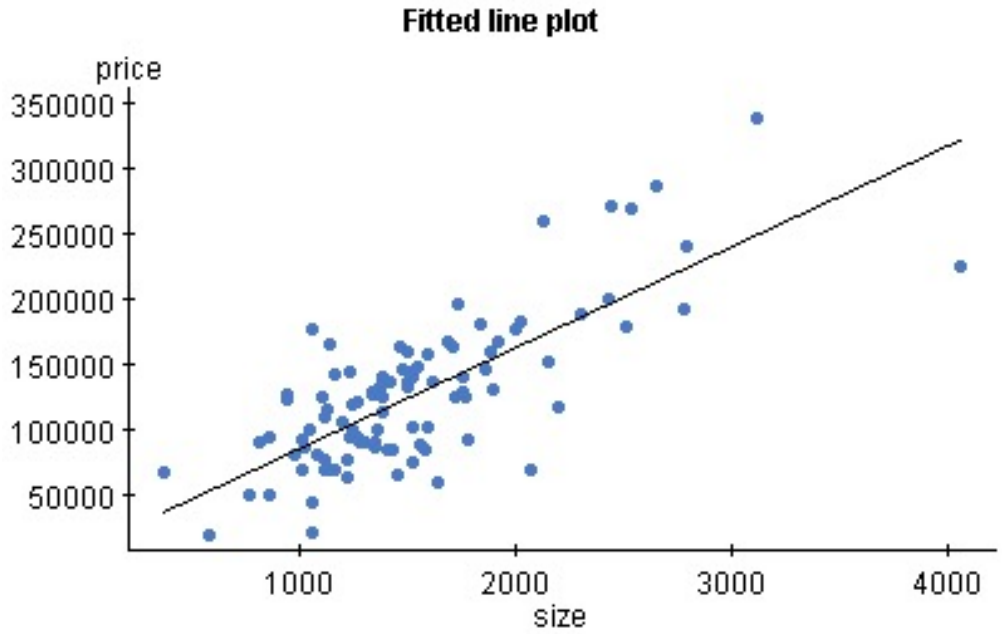
END



Linear Regression – The Basics

$Y \approx \beta_0 + \beta_1 X.$

$y = w_0 + w_1 * x$

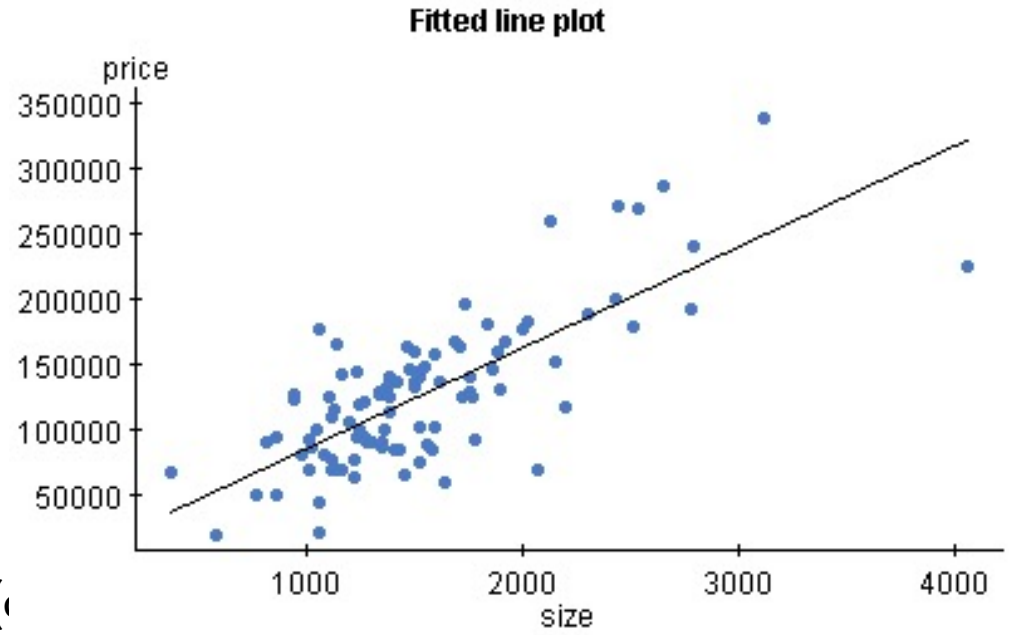


Linear Regression – The Basics

$Y \approx \beta_0 + \beta_1 X.$

$y = w_0 + w_1 * x$

How do I find (



Linear Regression – Multiple Dimensions

$X = [x_1]$, we care about w_0 (bias, intercept) + w_1(weight, slope) * x_1

What about multiple dimensions?

Linear Regression – Multiple Dimensions

$X = [x_1]$, we care about w_0 (bias, intercept) + w_1(weight, slope) * x_1

What about multiple dimensions? *Each Dimension Gets a Weight!*

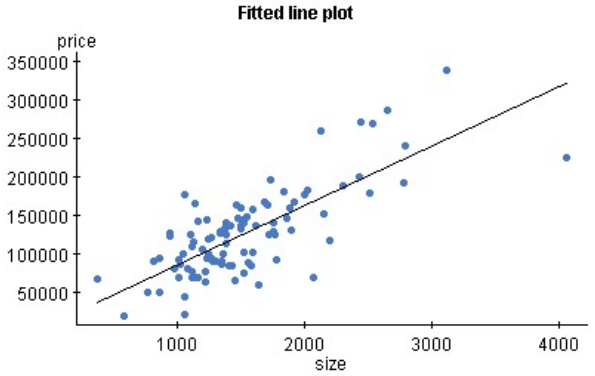
$X = [x_1, x_2, x_3, \ldots, x_n]$

$w_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + \ldots + w_n * x_n = W \cdot X$

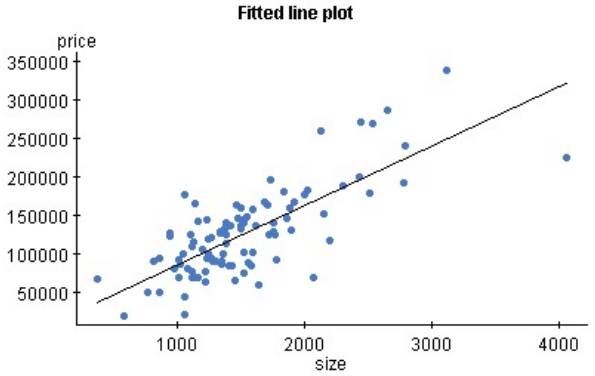
(NOTE: For dot product to work, X needs to be prepended, $X = [1, x_1, \ldots]$

Linear Regression – The Details

First, let's define "how good" a line is.
- How well does it define our points?



Linear Regression – The Details



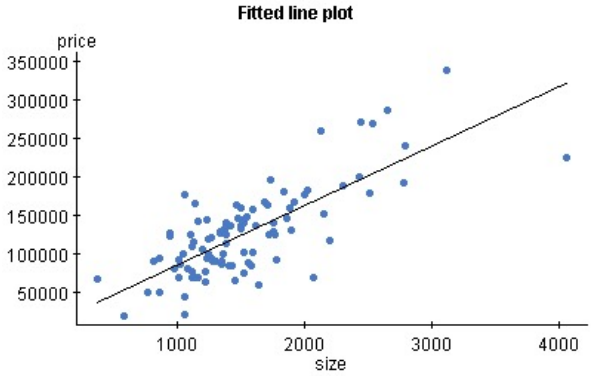
First, let's define "how good" a line is.
- How well does it define our points?

How far away, on average, is each point from the line?

The *Mean Squared Error*

1) The difference between each point and the line along the output axis
    1) (square this value to get it to always be a positive value)
2) Take the mean of this value

Linear Regression – The Details



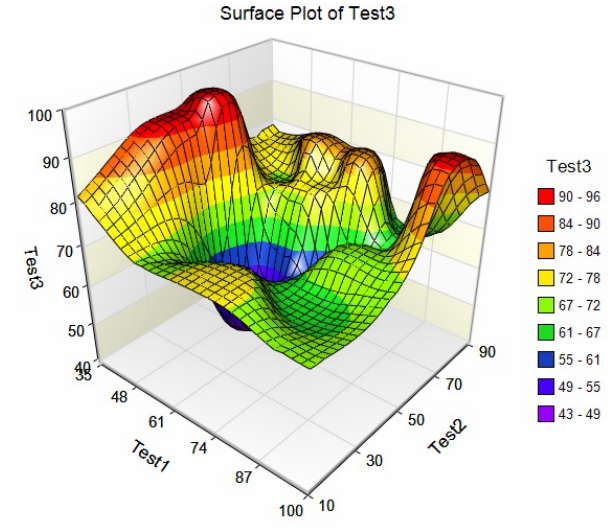
First, let's define "how good" a line is.
- How well does it define our points?

How far away, on average, is each point from the line?
The *Mean Squared Error*

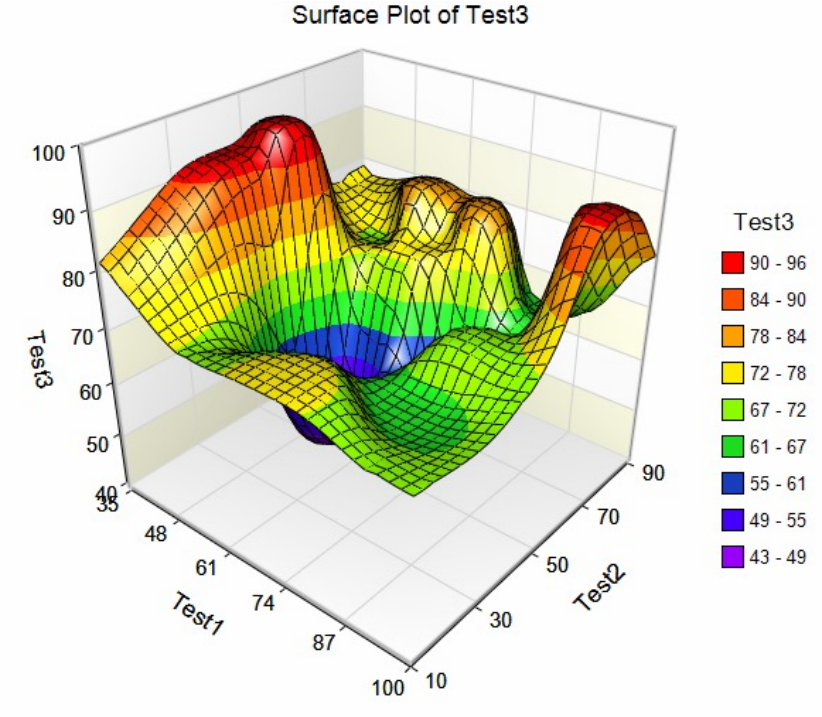
We want the minimum distance from the points to the line.

Minimizing a function...

Assume Test1 and Test2 are our w values, we want to minimize Test3. How would you do it?



Optimize by checking all points, choosing min

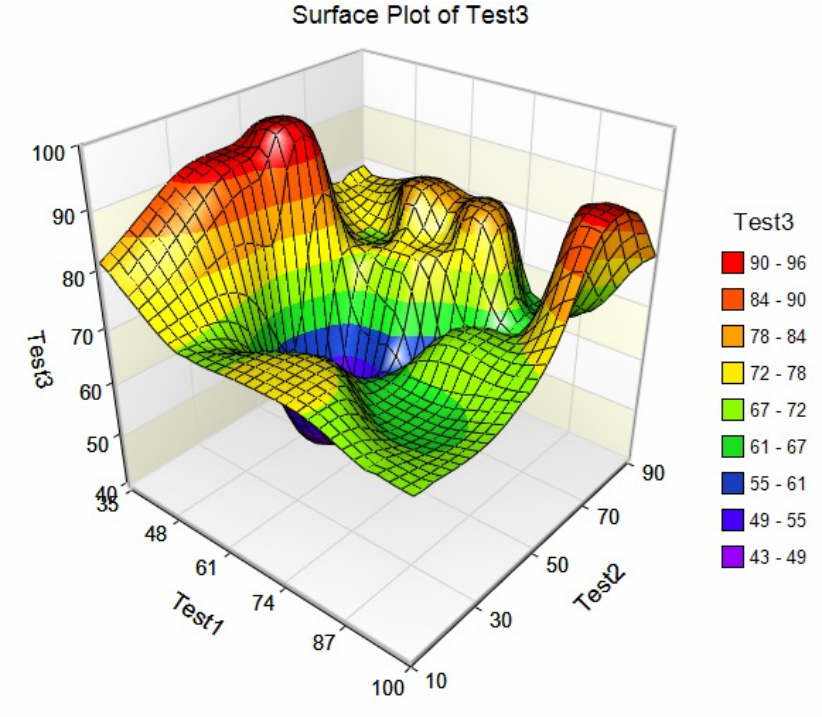


Optimizing using the gradient

For any function f, there exists a gradient.

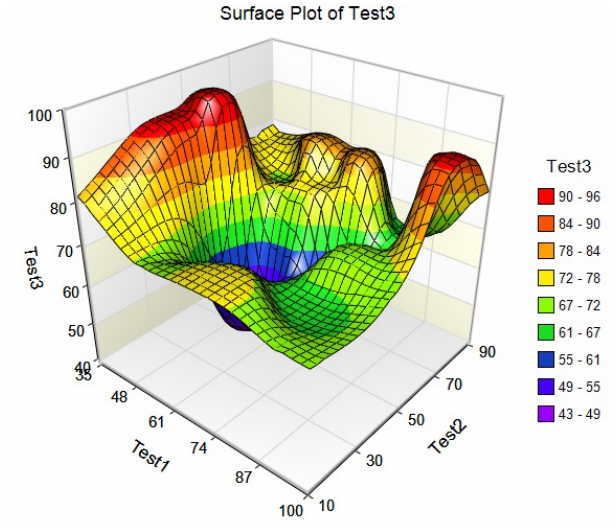
The gradient is the vector that points in the direction in which f is increasing the fastest.

Optimize using the gradient



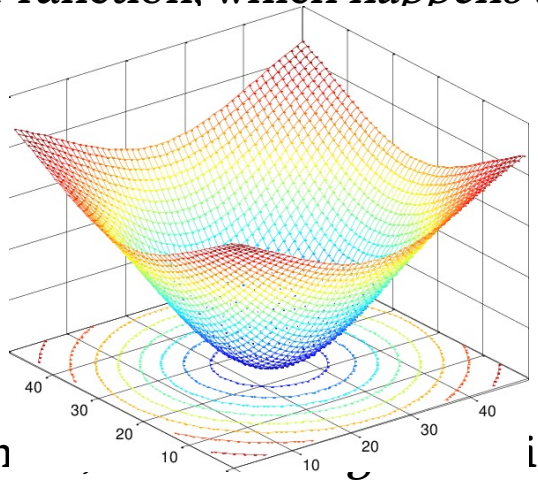
Optimizing argmin(MSE(w))

Gradient descent would not work for any given function...



Optimizing argmin(MSE(w))

But our function is a *linear* function, which *happens* to be concave!



i.e. If we're at a *local* minim inimum

https://homes.cs.washington.edu/~marcotcr/blog/concavity/

Optimizing using Gradient Descent

We can think of this in terms of *updating our weights*:

$w \leftarrow w - \nabla_w \text{MSE}(w)$

$\nabla_w \text{MSE}(w) = [\partial \text{MSE}(w) / \partial w_0, \partial \text{MSE}(w) / \partial w_1, \dots, \partial \text{MSE}(w) / \partial w_n]$

The weights are updated by removing the gradient
Each weight is updated by subtracting its partial derivative

Optimizing using Gradient Descent

We can think of this in terms of *a rate of change*:

$$w \leftarrow w - \lambda * \nabla_w \text{MSE}(w)$$

Optimizing using Gradient Descent

We can think of this in terms of *each feature weight's update*:

$w_k \leftarrow w_k - \lambda * \partial MSE(w) / \partial w_k$ for each $k = 0 \rightarrow D$ (dimensions)

Optimizing using Gradient Descent

We can think of this in terms of *each feature weight's update*:

$w_k \leftarrow w_k - \lambda * \partial MSE(w) / \partial w_k$ for each $k = 0 \rightarrow D$ (dimensions)

We can think of the partial derivative* of w_k in terms of the k^{th} feature of the i^{th} training example

*The proof of this fact can be found in readings... (i.e. it's really hard math!)