A crash course in machine learning

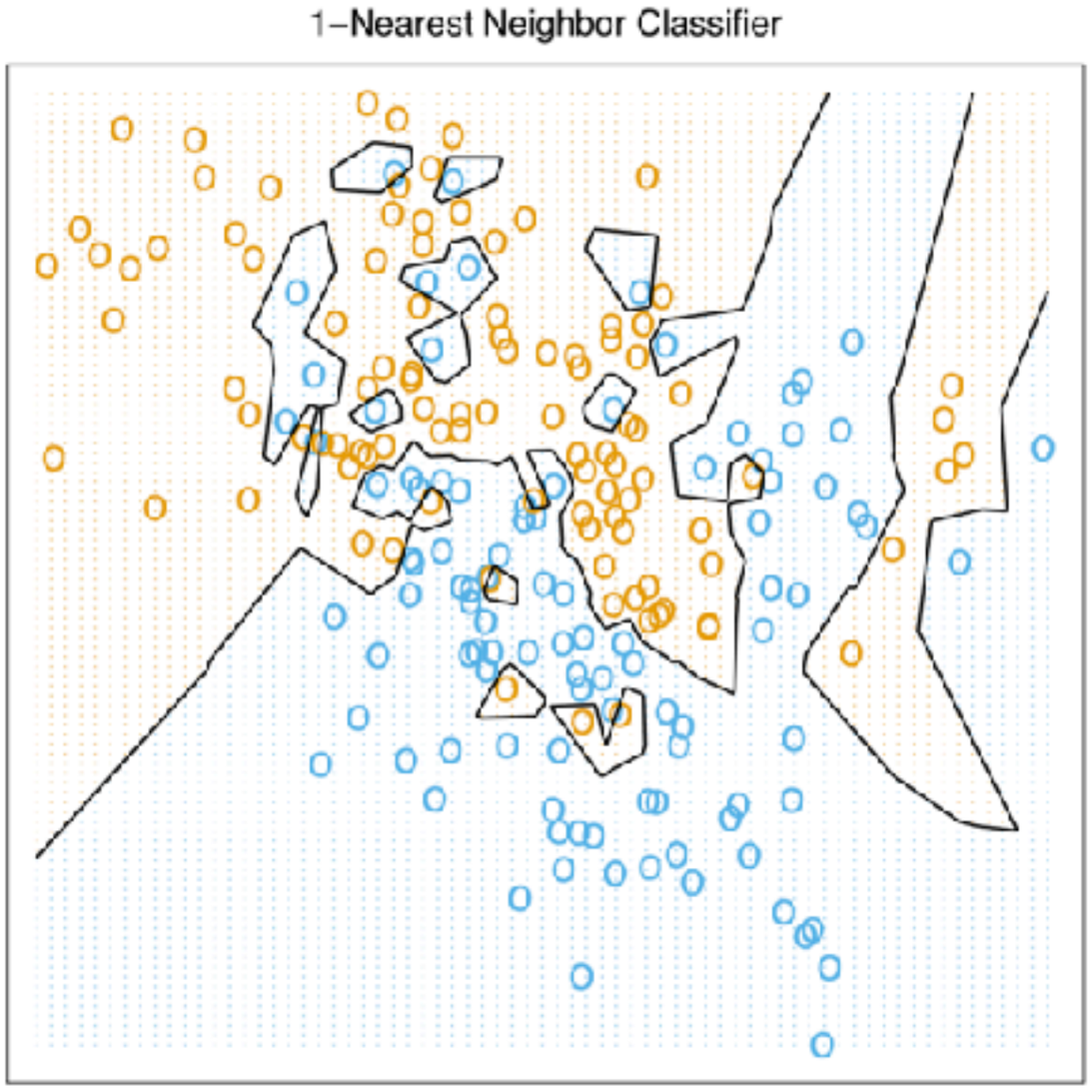
Florent Krzakala & Antoine Baker

https://sphinxteam.github.io/mlcourse_2019/

florent.krzakala@gmail.com **antoine.baker@gmail.com**

So far…

Knn and linear methods

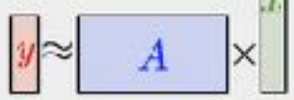


Solving $y \underset{\approx}{=} A\,x \in \mathbb{R}^m \qquad A \in \mathbb{R}^{m \times n}$

Determined $(m = n)$: $x = A^{-1}y$

Over-determined $(m > n)$: $\min_x \|Ax - y\|^2$

$x = (A^\top A)^{-1} A^\top y \overset{\text{def.}}{=} A^- y$



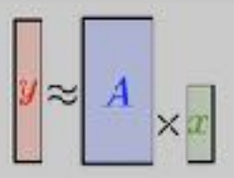
Under-determined $(m < n)$: $\min_x \{\|x\| \;;\; Ax = y\}$

$x = A^\top (AA^\top)^{-1} y \overset{\text{def.}}{=} A^- y$

A ill-posed and/or noise: $\min_x |Ax - y\|^2 + \lambda |x|^2$

$x = (A^\top A + \lambda \mathrm{Id}_n)^{-1} A^\top y \overset{\lambda \to 0}{\longrightarrow} A^+ y$

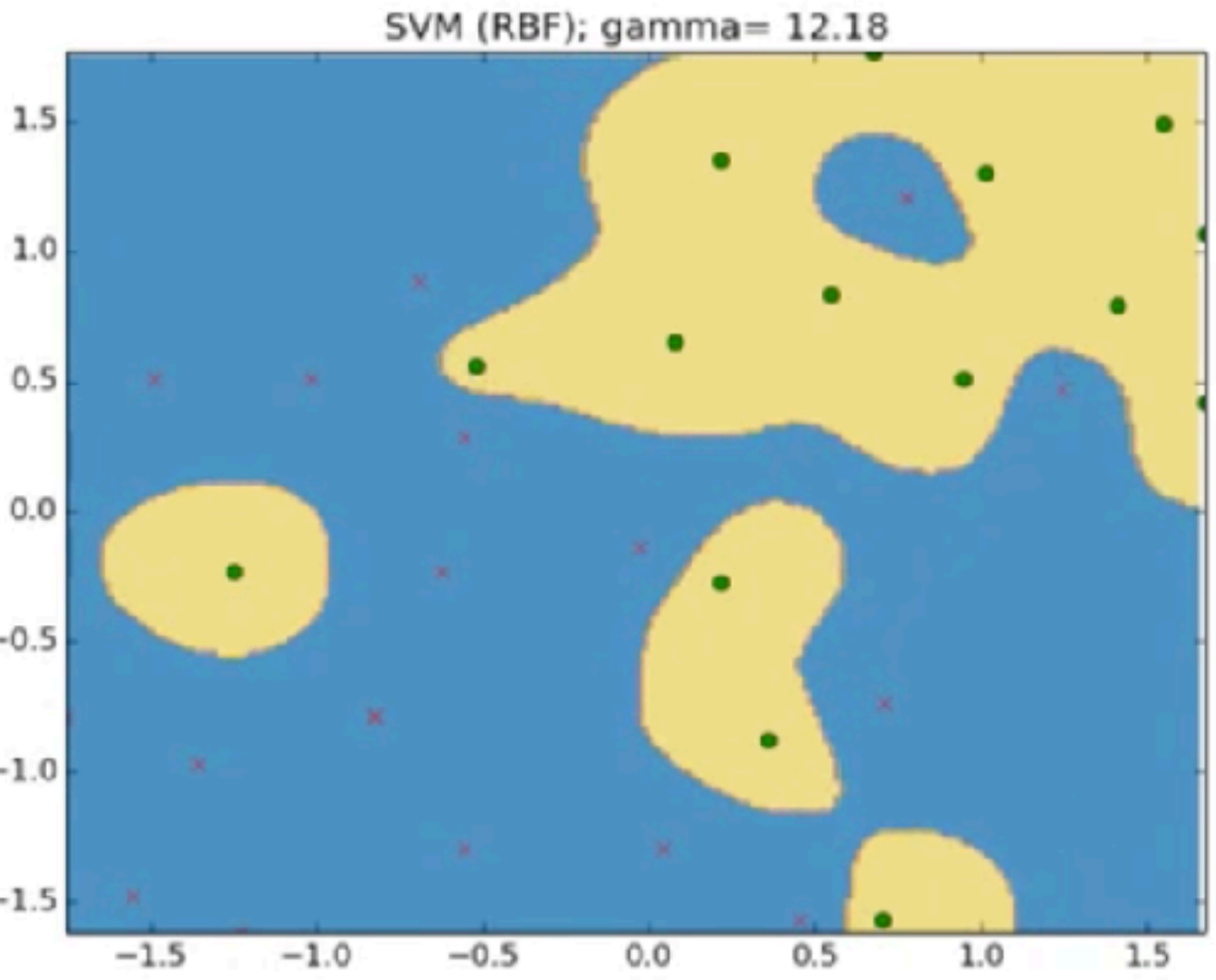
$= A^\top (AA^\top + \lambda \mathrm{Id}_m)^{-1} y$ (Woodbury identity)

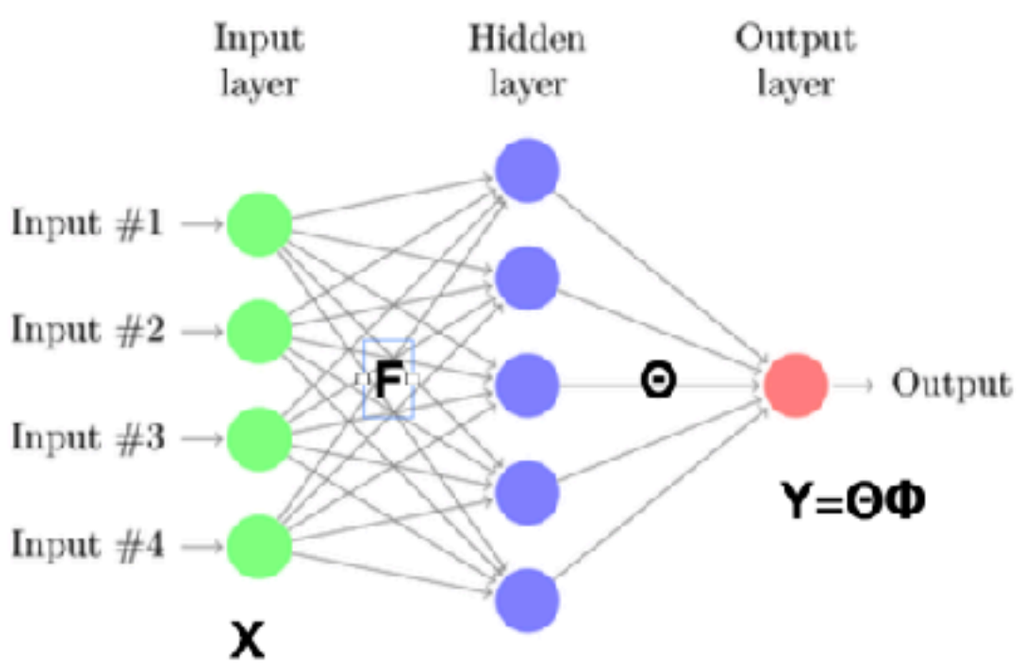


So far…

Kernels methods and random projections

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

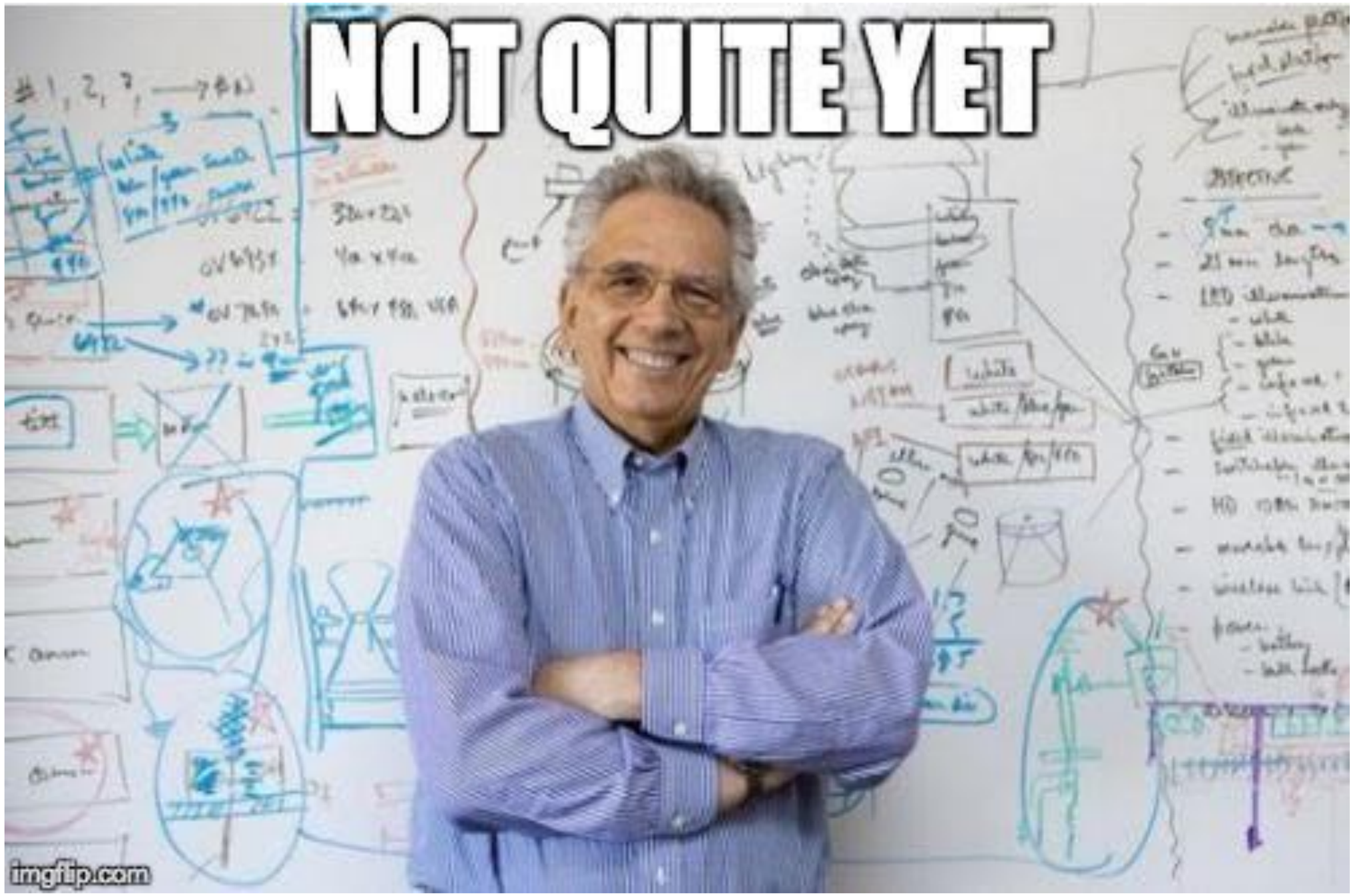




$$\Phi = g(Fx)$$

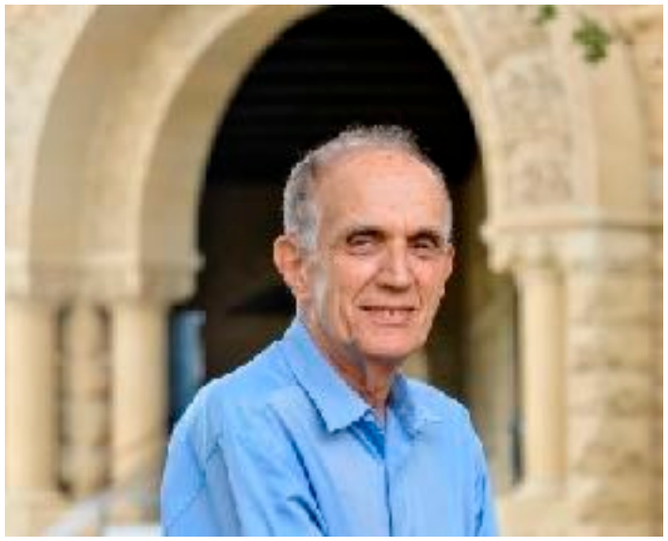
Hidden dimension
=infty (kernel)
= D with random projection


That’s it?
Are we finally going to speak about deep learning?
47


NOT QUITE YET
imgflip.com

Ensemble methods

Bagging
(Bootstrap Aggregating)



Bradley Efron (1979)

Leo Breiman (1994)

Bootstrap

Why working with a single data set?

```
[[ 0.524]
 [ 0.4  ]
 [ 0.361]
 [ 0.078]
 [ 0.859]
 [ 0.654]
 [ 0.384]
 [ 0.738]
 [ 0.156]
 [ 0.572]]
```

Bootstrap

Why working with a single data set?

Create many data sets by sampling with replacement:

```
[[ 0.524]
 [ 0.4  ]
 [ 0.361]
 [ 0.078]
 [ 0.859]
 [ 0.654]
 [ 0.384]
 [ 0.738]
 [ 0.156]
 [ 0.572]]
```

Bootstrap

Why working with a single data set?

Create many data sets by sampling with replacement:

```
[[ 0.524]
 [ 0.4  ]
 [ 0.361]
 [ 0.078]
 [ 0.859]
 [ 0.654]
 [ 0.384]
 [ 0.738]
 [ 0.156]
 [ 0.572]]
```

```
[[ 0.361]
 [ 0.572]
 [ 0.156]
 [ 0.361]
 [ 0.4  ]
 [ 0.384]
 [ 0.572]
 [ 0.654]
 [ 0.4  ]
 [ 0.384]]
```

Bootstrap

Why working with a single data set?

Create many data sets by sampling with replacement:

```
[[ 0.524]
 [ 0.4  ]
 [ 0.361]
 [ 0.078]
 [ 0.859]
 [ 0.654]
 [ 0.384]
 [ 0.738]
 [ 0.156]
 [ 0.572]]
```

```
[[ 0.361]
 [ 0.572]
 [ 0.156]
 [ 0.361]
 [ 0.4  ]
 [ 0.384]
 [ 0.572]
 [ 0.654]
 [ 0.4  ]
 [ 0.384]]
```

```
[[ 0.361]
 [ 0.156]
 [ 0.4  ]
 [ 0.654]
 [ 0.654]
 [ 0.859]
 [ 0.384]
 [ 0.4  ]
 [ 0.524]
 [ 0.859]]
```

Bootstrap

Why working with a single data set?

Create many data sets by sampling with replacement:

```
[[ 0.524]
 [ 0.4  ]
 [ 0.361]
 [ 0.078]
 [ 0.859]
 [ 0.654]
 [ 0.384]
 [ 0.738]
 [ 0.156]
 [ 0.572]]
```

```
[[ 0.361]
 [ 0.572]
 [ 0.156]
 [ 0.361]
 [ 0.4  ]
 [ 0.384]
 [ 0.572]
 [ 0.654]
 [ 0.4  ]
 [ 0.384]]
```

```
[[ 0.361]
 [ 0.156]
 [ 0.4  ]
 [ 0.654]
 [ 0.654]
 [ 0.859]
 [ 0.384]
 [ 0.4  ]
 [ 0.524]
 [ 0.859]]
```

```
[[ 0.156]
 [ 0.156]
 [ 0.524]
 [ 0.078]
 [ 0.524]
 [ 0.859]
 [ 0.361]
 [ 0.078]
 [ 0.654]
 [ 0.4  ]]
```

Bootstrap

Why working with a single data set?

Create many data sets by sampling with replacement:

```
[[ 0.524]
 [ 0.4  ]
 [ 0.361]
 [ 0.078]
 [ 0.859]
 [ 0.654]
 [ 0.384]
 [ 0.738]
 [ 0.156]
 [ 0.572]]
```

```
[[ 0.361]
 [ 0.572]
 [ 0.156]
 [ 0.361]
 [ 0.4  ]
 [ 0.384]
 [ 0.572]
 [ 0.654]
 [ 0.4  ]
 [ 0.384]]
```

```
[[ 0.361]
 [ 0.156]
 [ 0.4  ]
 [ 0.654]
 [ 0.654]
 [ 0.859]
 [ 0.384]
 [ 0.4  ]
 [ 0.524]
 [ 0.859]]
```

```
[[ 0.156]
 [ 0.156]
 [ 0.524]
 [ 0.078]
 [ 0.524]
 [ 0.859]
 [ 0.361]
 [ 0.078]
 [ 0.654]
 [ 0.4  ]]
```

```
[[ 0.078]
 [ 0.524]
 [ 0.859]
 [ 0.384]
 [ 0.859]
 [ 0.361]
 [ 0.361]
 [ 0.738]
 [ 0.572]
 [ 0.156]]
```

Bootstrap

Why working with a single data set?

Create many data sets by sampling with replacement:

```
[[ 0.524]
 [ 0.4  ]
 [ 0.361]
 [ 0.078]
 [ 0.859]
 [ 0.654]
 [ 0.384]
 [ 0.738]
 [ 0.156]
 [ 0.572]]
```

```
[[ 0.361]
 [ 0.572]
 [ 0.156]
 [ 0.361]
 [ 0.4  ]
 [ 0.384]
 [ 0.572]
 [ 0.654]
 [ 0.4  ]
 [ 0.384]]
```

```
[[ 0.361]
 [ 0.156]
 [ 0.4  ]
 [ 0.654]
 [ 0.654]
 [ 0.859]
 [ 0.384]
 [ 0.4  ]
 [ 0.524]
 [ 0.859]]
```

```
[[ 0.156]
 [ 0.156]
 [ 0.524]
 [ 0.078]
 [ 0.524]
 [ 0.859]
 [ 0.361]
 [ 0.078]
 [ 0.654]
 [ 0.4  ]]
```

```
[[ 0.078]
 [ 0.524]
 [ 0.859]
 [ 0.384]
 [ 0.859]
 [ 0.361]
 [ 0.361]
 [ 0.738]
 [ 0.572]
 [ 0.156]]
```

Consider these datasets are equally valid

Bootstrap

Why working with a single data set?

Create many data sets by sampling with replacement:

```
[[ 0.524]
 [ 0.4  ]
 [ 0.361]
 [ 0.078]
 [ 0.859]
 [ 0.654]
 [ 0.384]
 [ 0.738]
 [ 0.156]
 [ 0.572]]
```

```
[[ 0.361]
 [ 0.572]
 [ 0.156]
 [ 0.361]
 [ 0.4  ]
 [ 0.384]
 [ 0.572]
 [ 0.654]
 [ 0.4  ]
 [ 0.384]]
```

```
[[ 0.361]
 [ 0.156]
 [ 0.4  ]
 [ 0.654]
 [ 0.654]
 [ 0.859]
 [ 0.384]
 [ 0.4  ]
 [ 0.524]
 [ 0.859]]
```

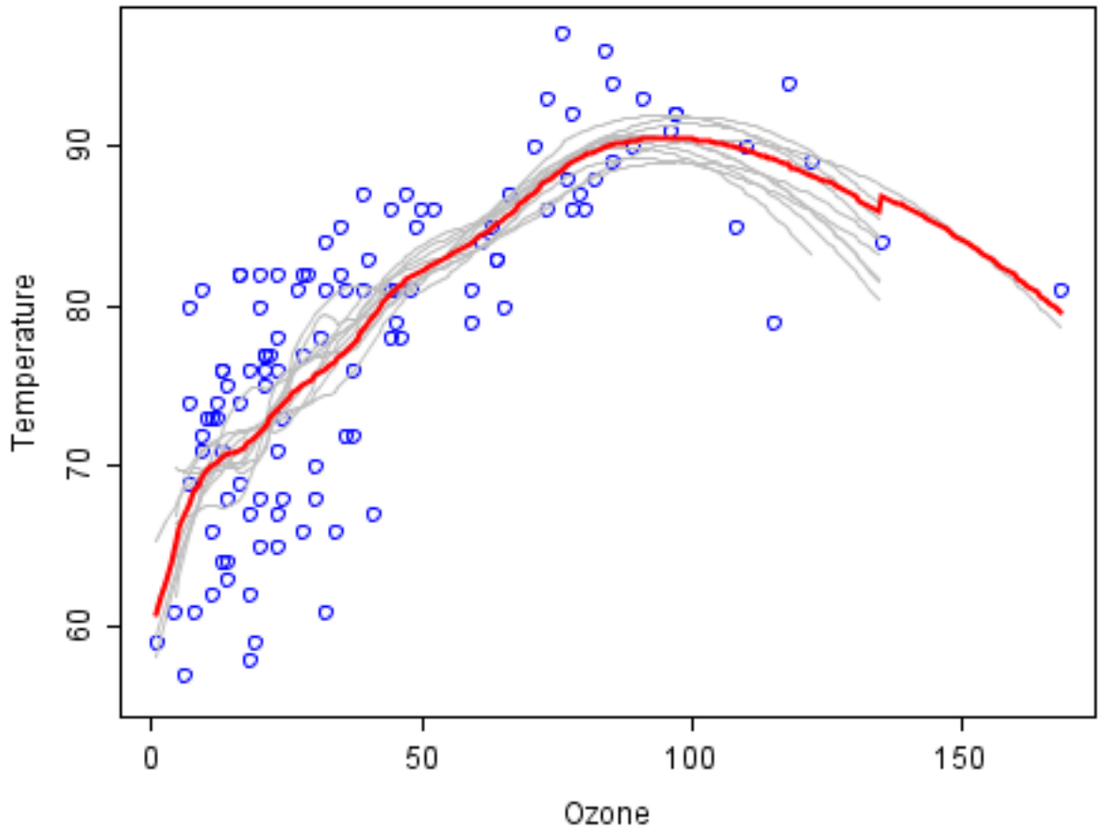
```
[[ 0.156]
 [ 0.156]
 [ 0.524]
 [ 0.078]
 [ 0.524]
 [ 0.859]
 [ 0.361]
 [ 0.078]
 [ 0.654]
 [ 0.4  ]]
```

```
[[ 0.078]
 [ 0.524]
 [ 0.859]
 [ 0.384]
 [ 0.859]
 [ 0.361]
 [ 0.361]
 [ 0.738]
 [ 0.572]
 [ 0.156]]
```

Consider these datasets are equally valid

Many uses: error analysis, statistical tests, …

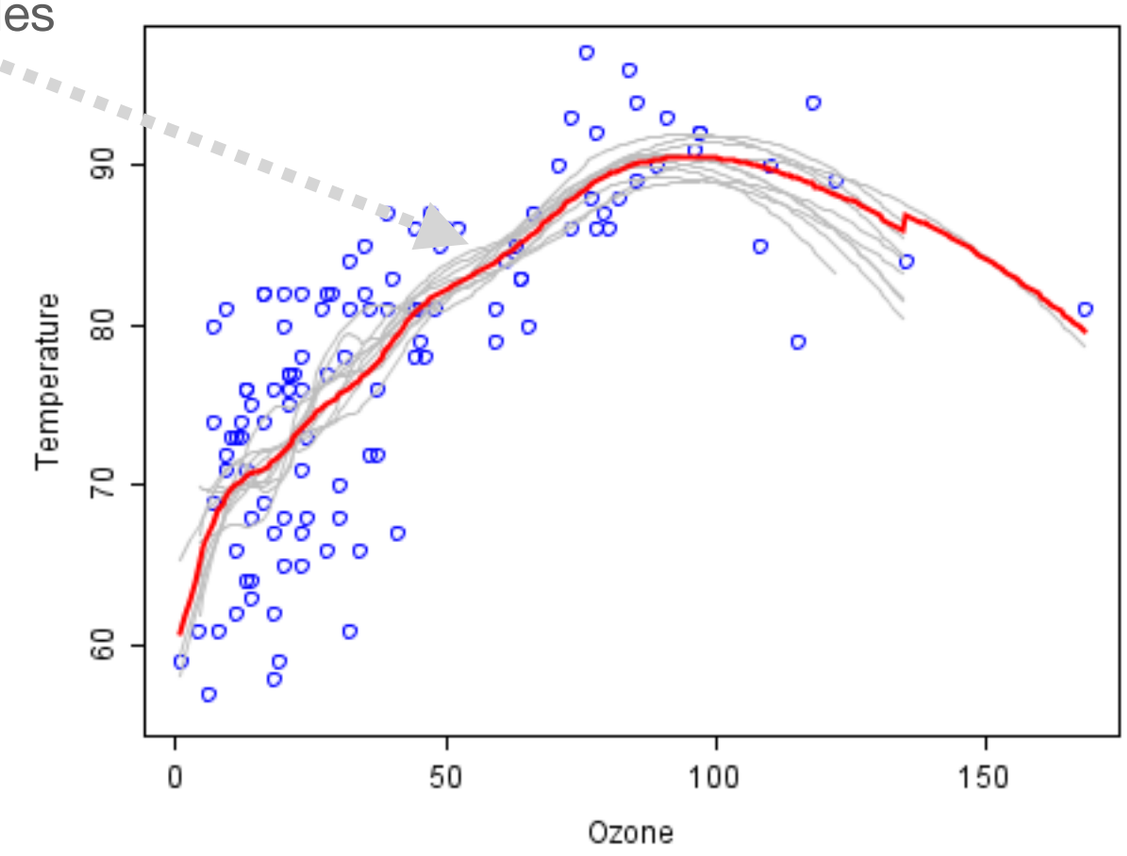
Bagging
(Bootstrap Aggregating)



Ozone data

Bagging
(Bootstrap Aggregating)

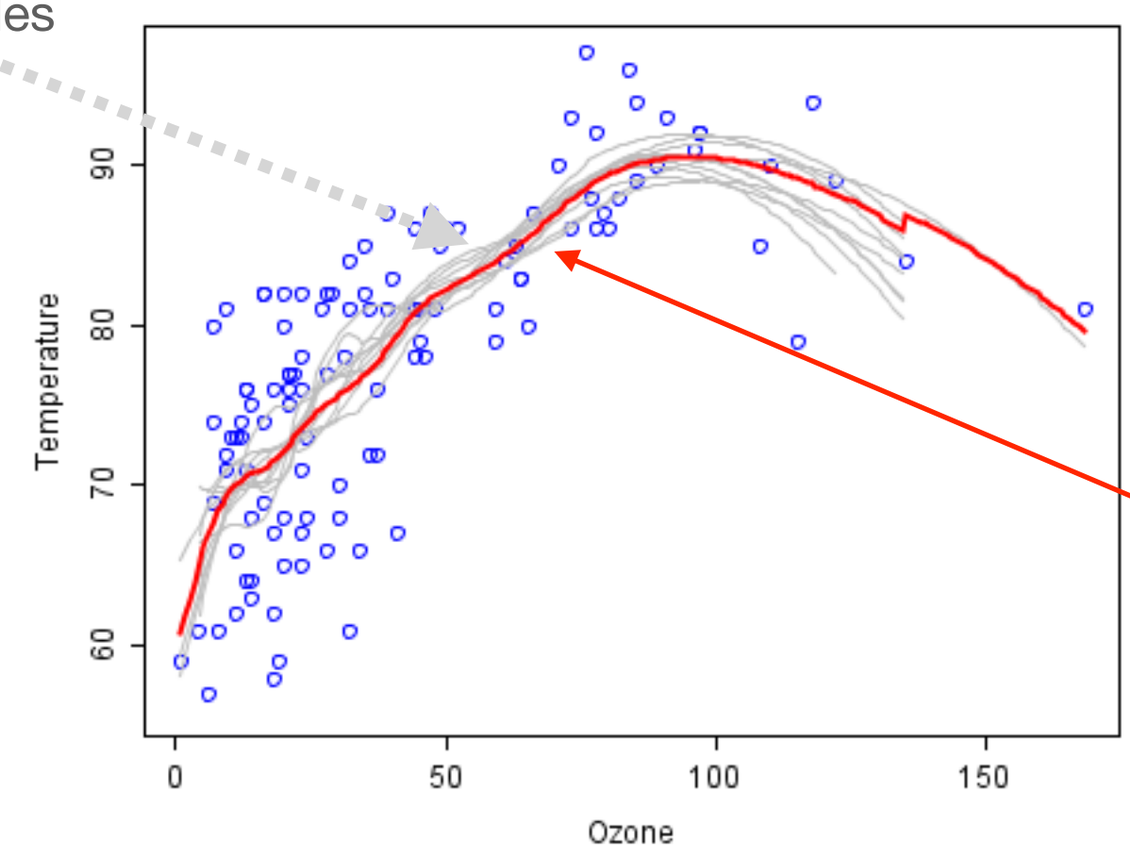
Fits on bootstrap samples



Ozone data

Bagging
(Bootstrap Aggregating)

Fits on bootstrap samples

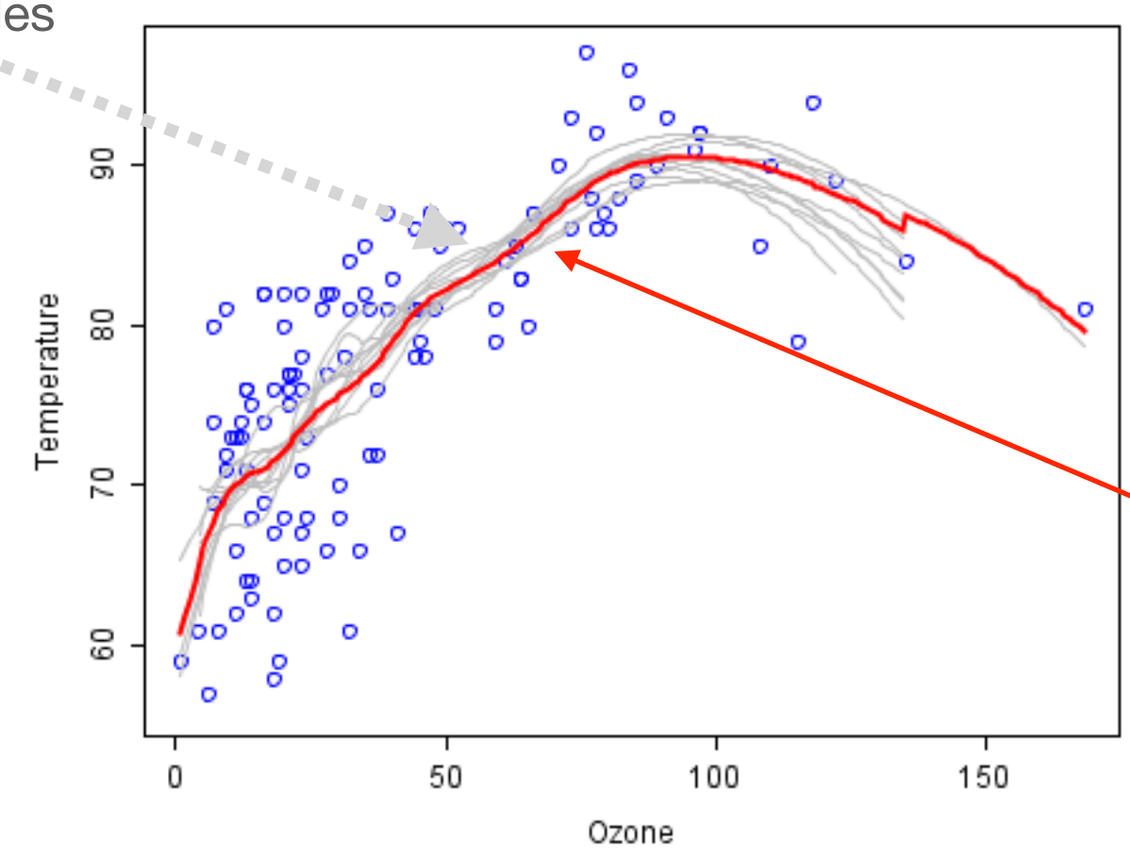


Ozone data

Averaged model

Bagging
(Bootstrap Aggregating)

Fits on bootstrap samples

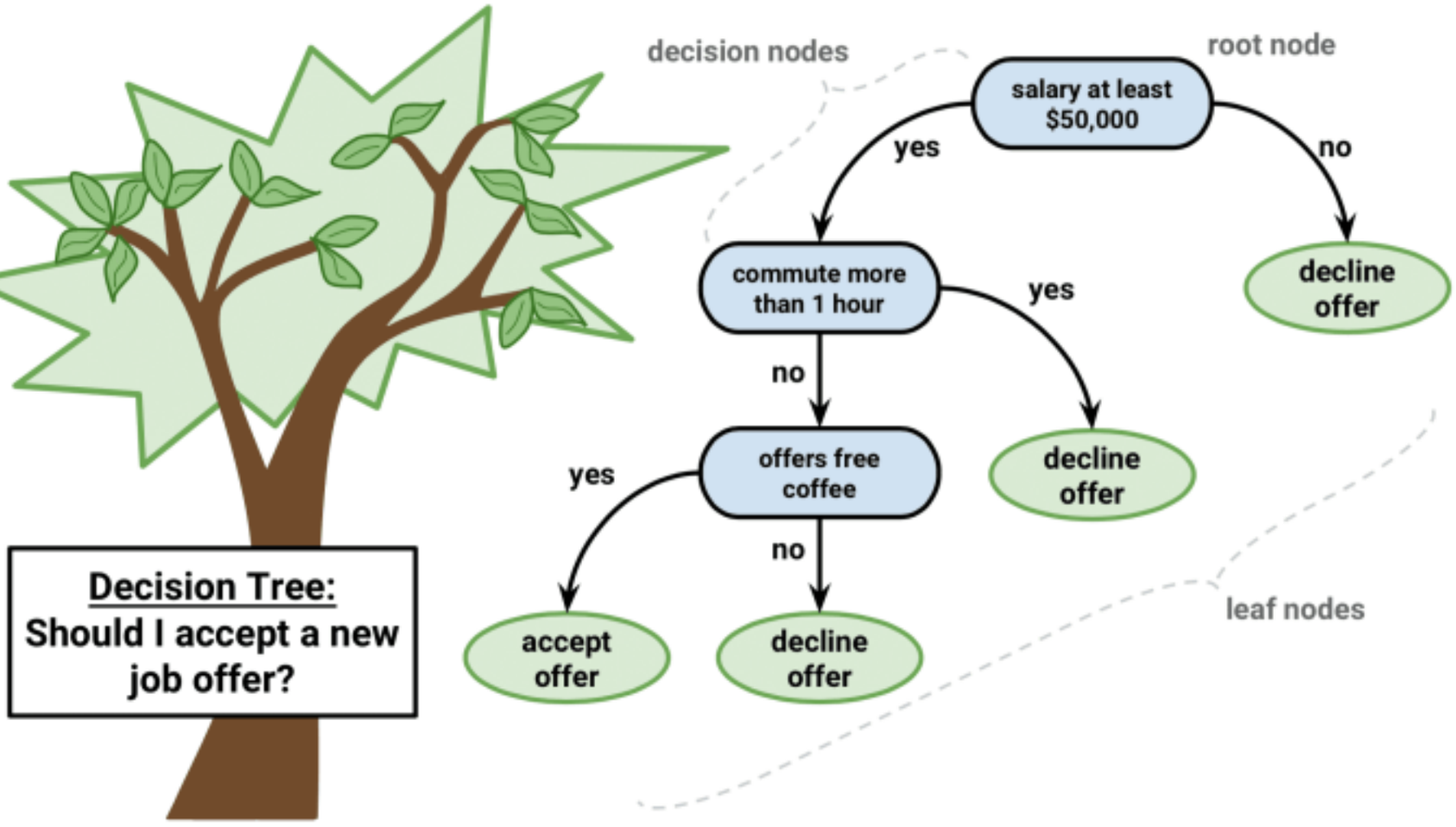


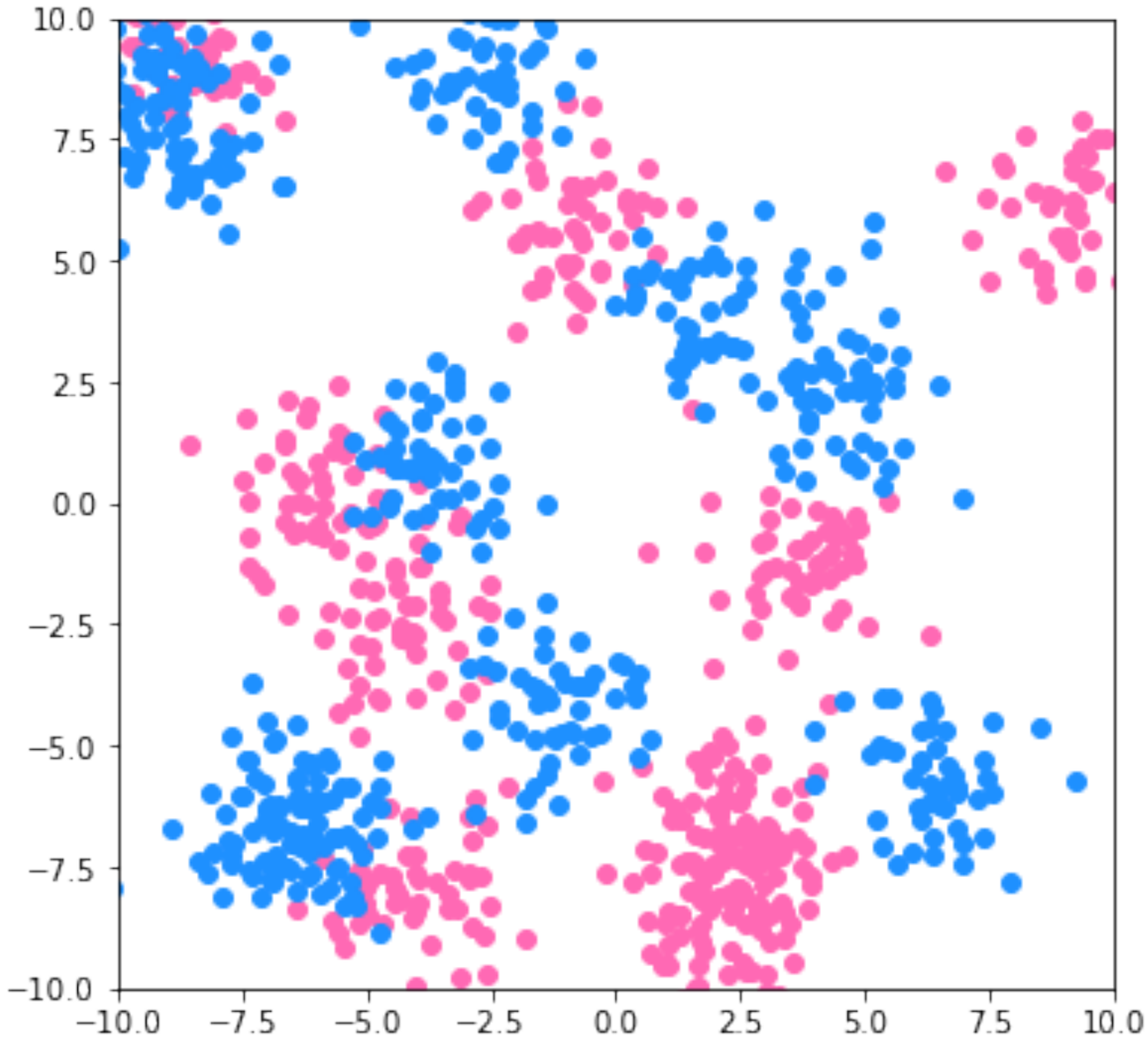
Ozone data

Averaging models reduces overfitting!

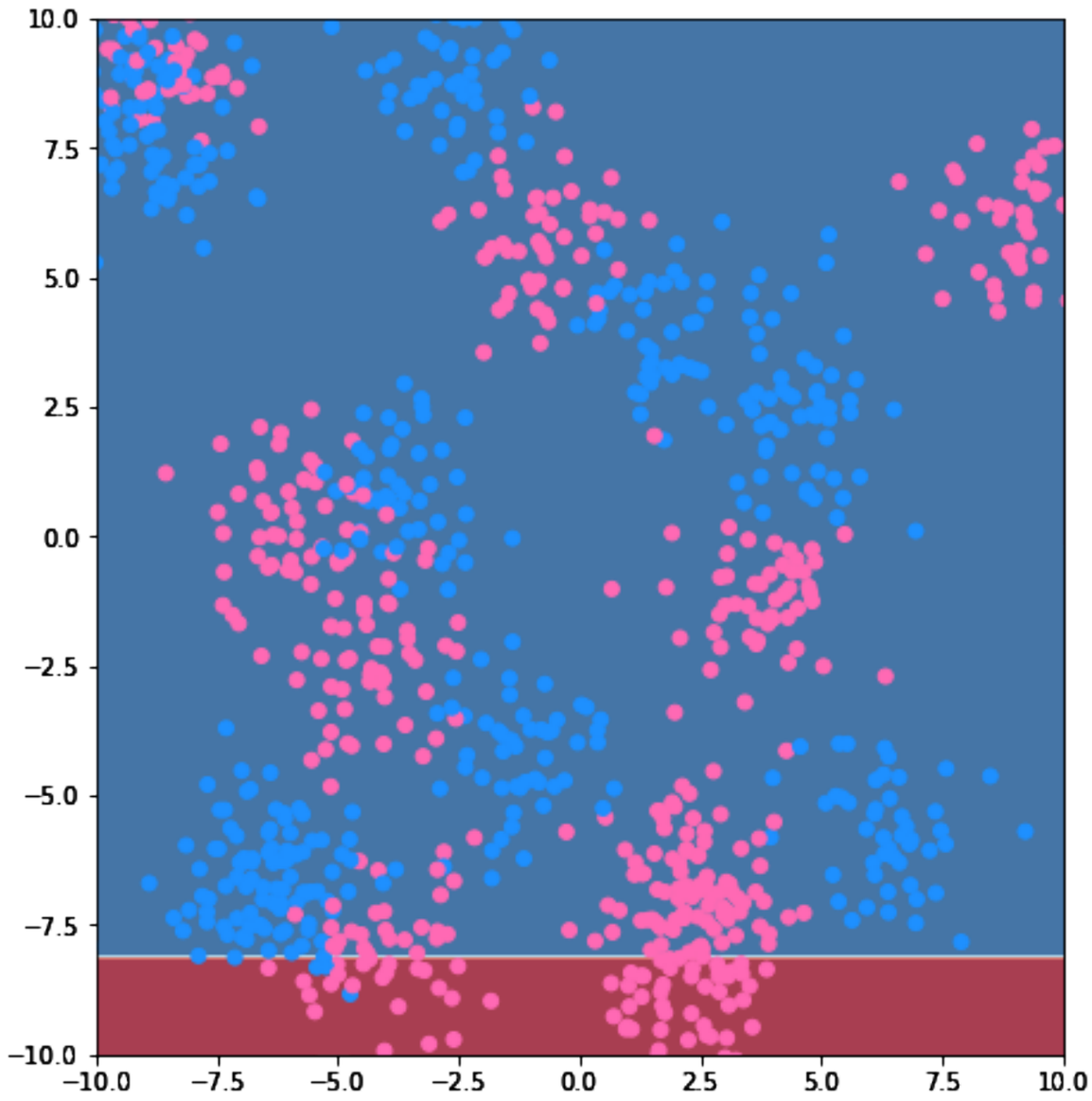
And for classification?

Decision trees!

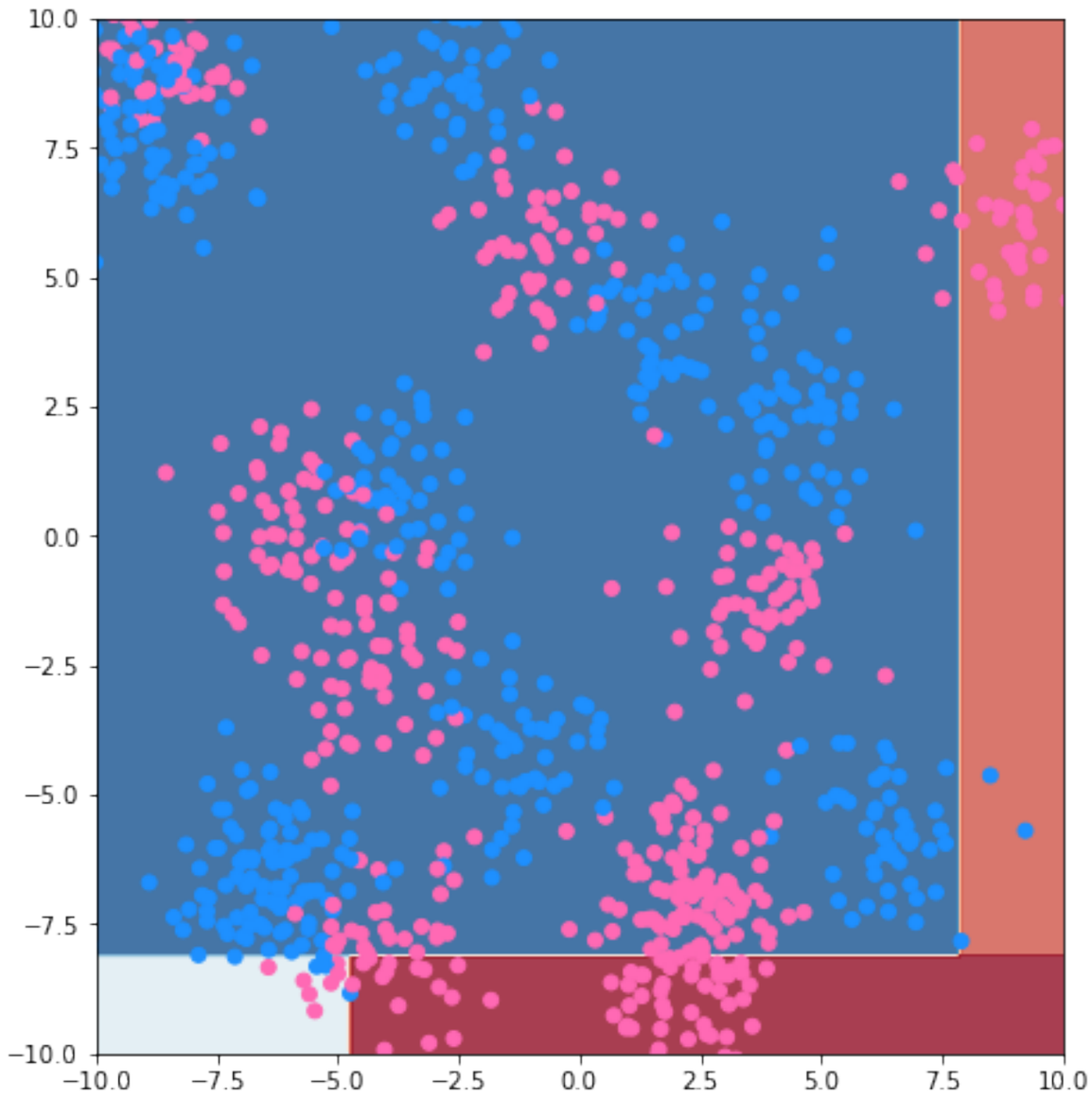



10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

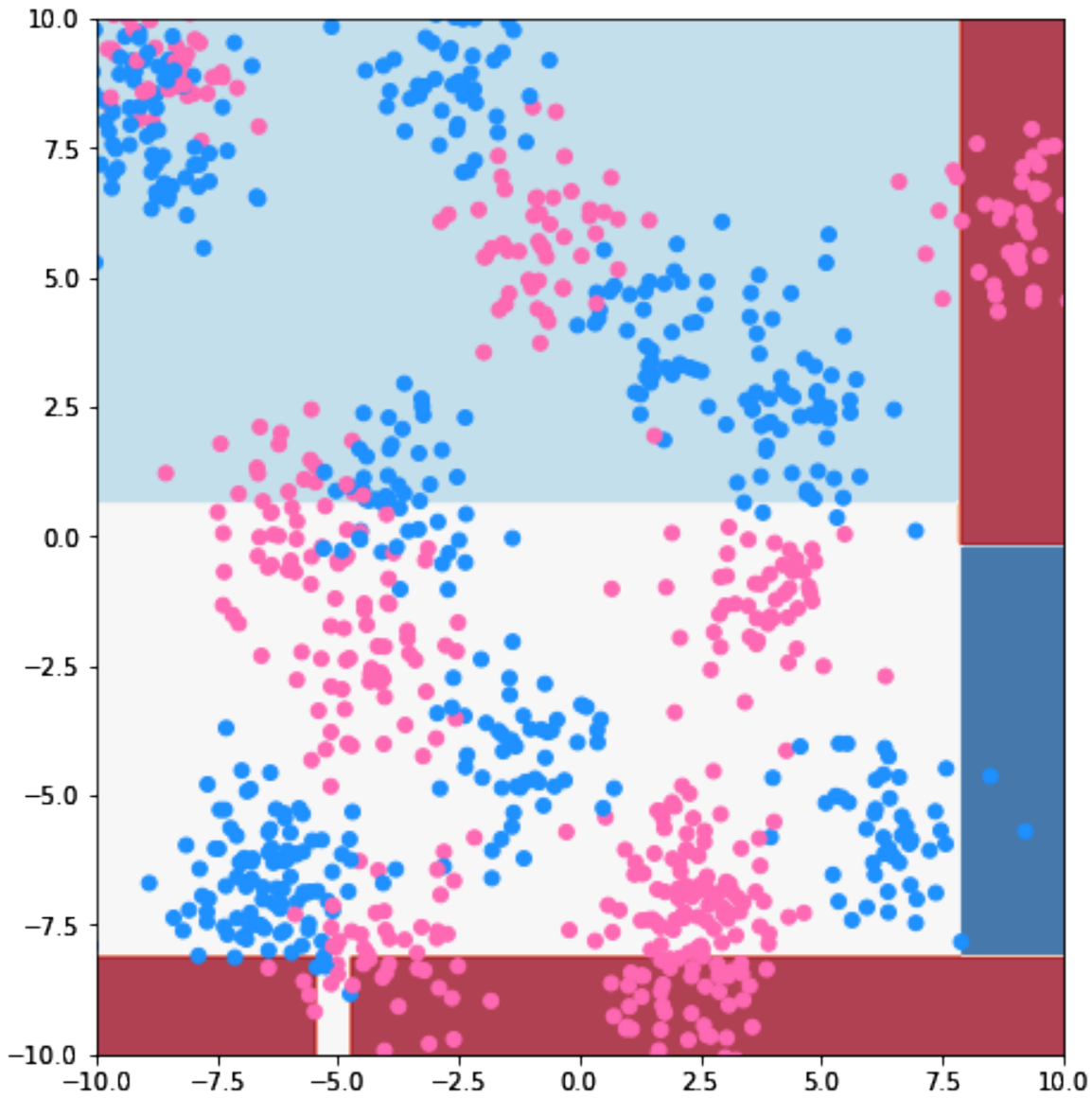
Decision tree, depth=1



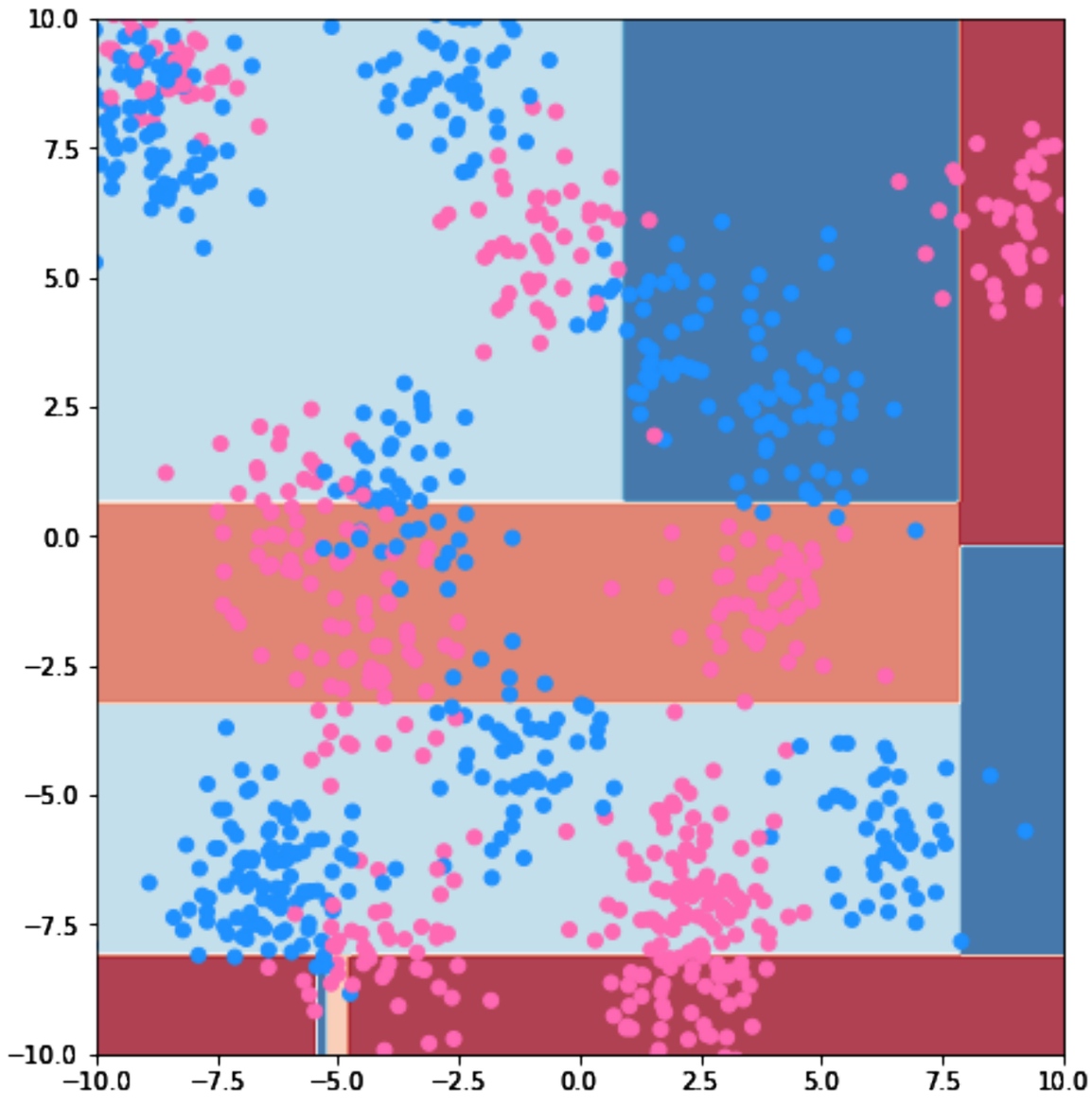
Decision tree, depth=2



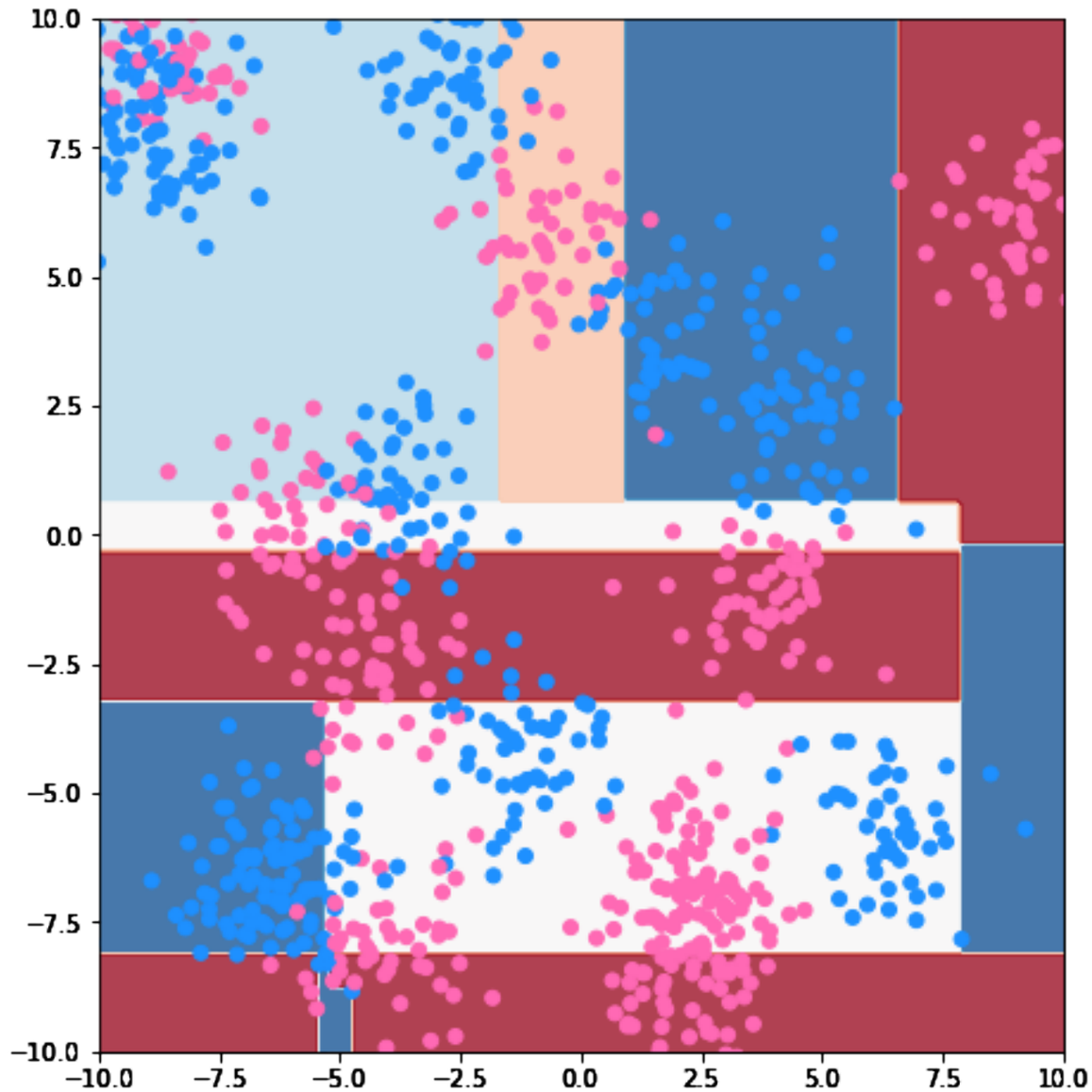
Decision tree, depth=3



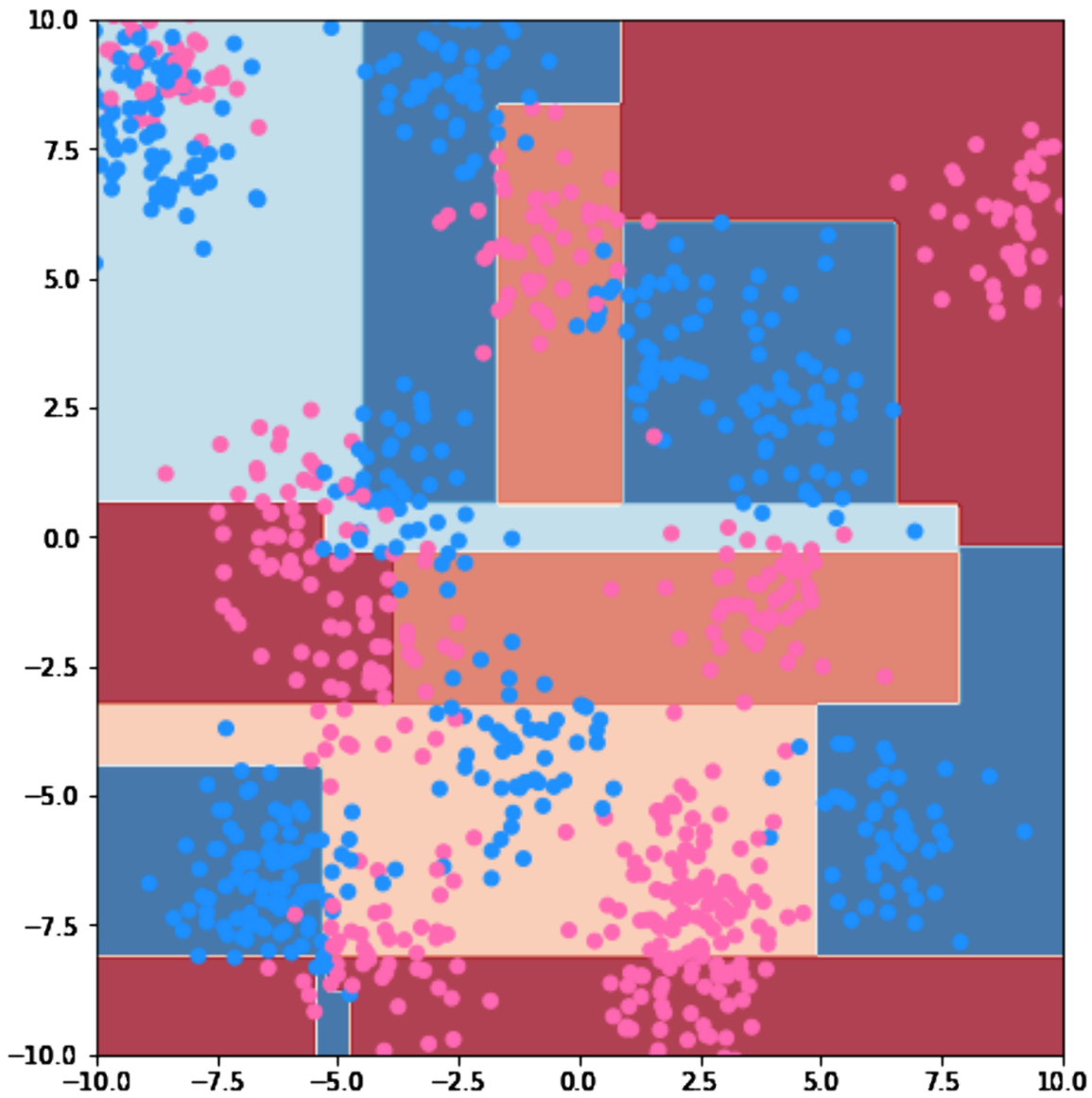
Decision tree, depth=4



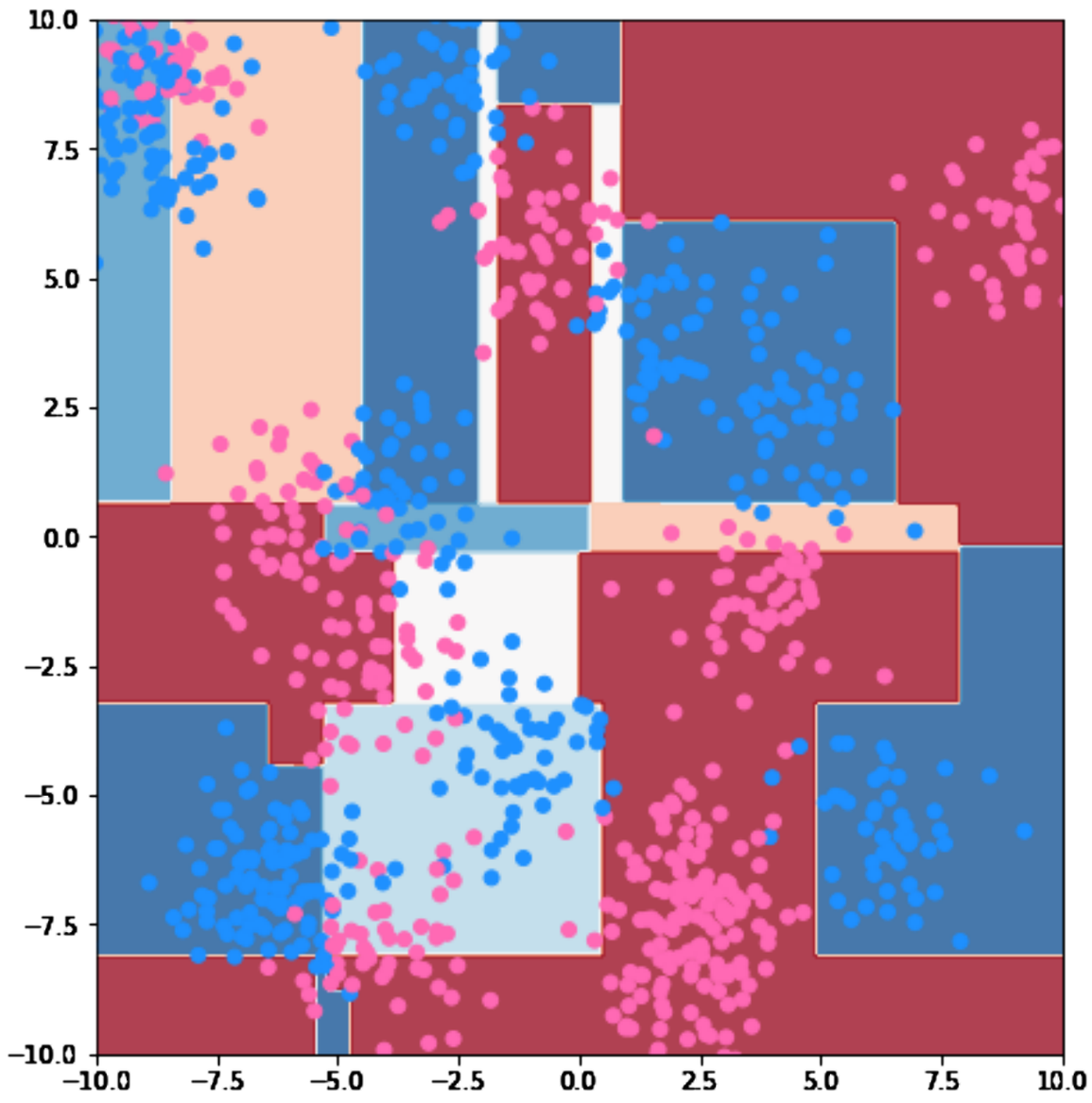
Decision tree, depth=5



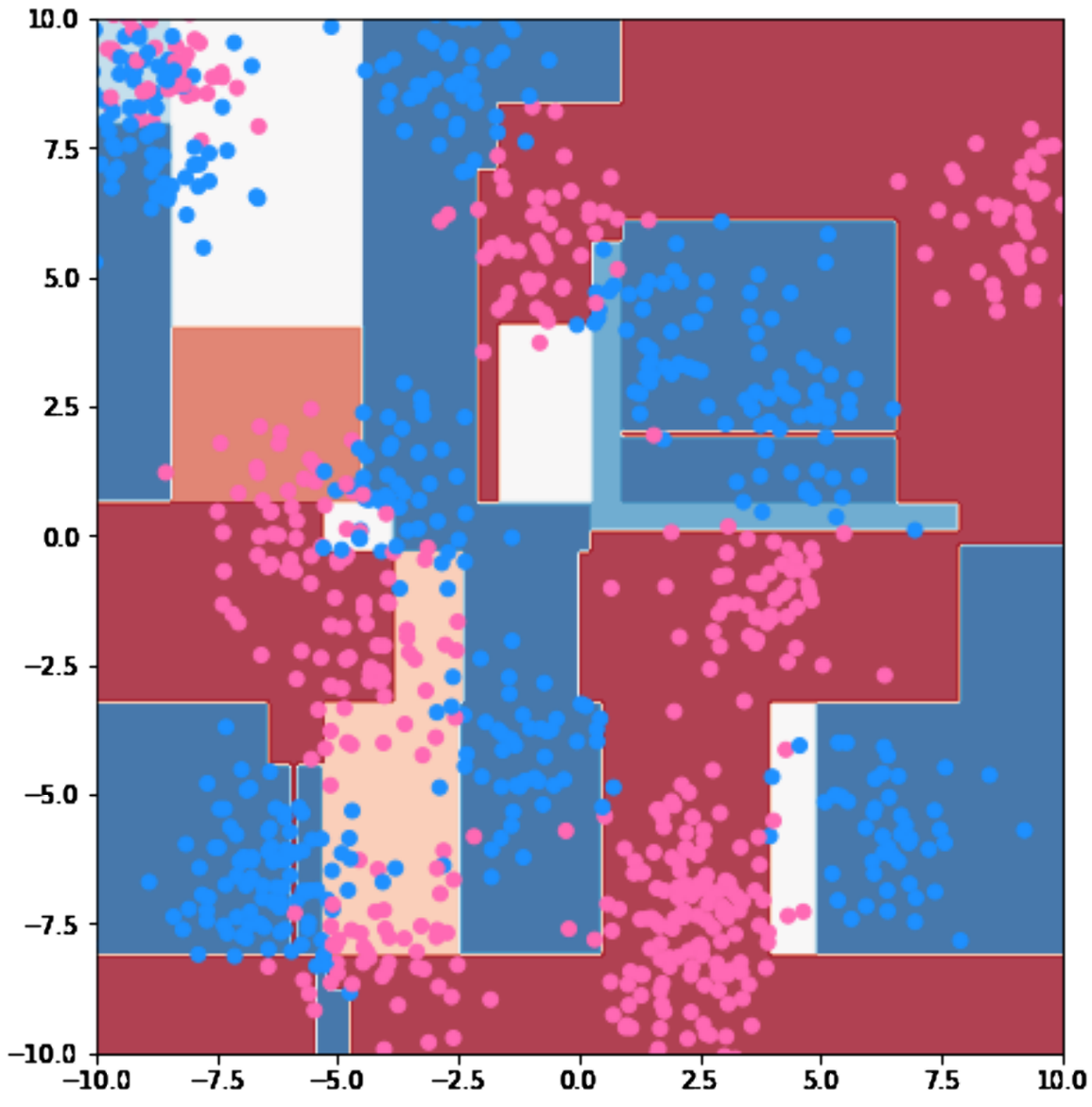
Decision tree, depth=6



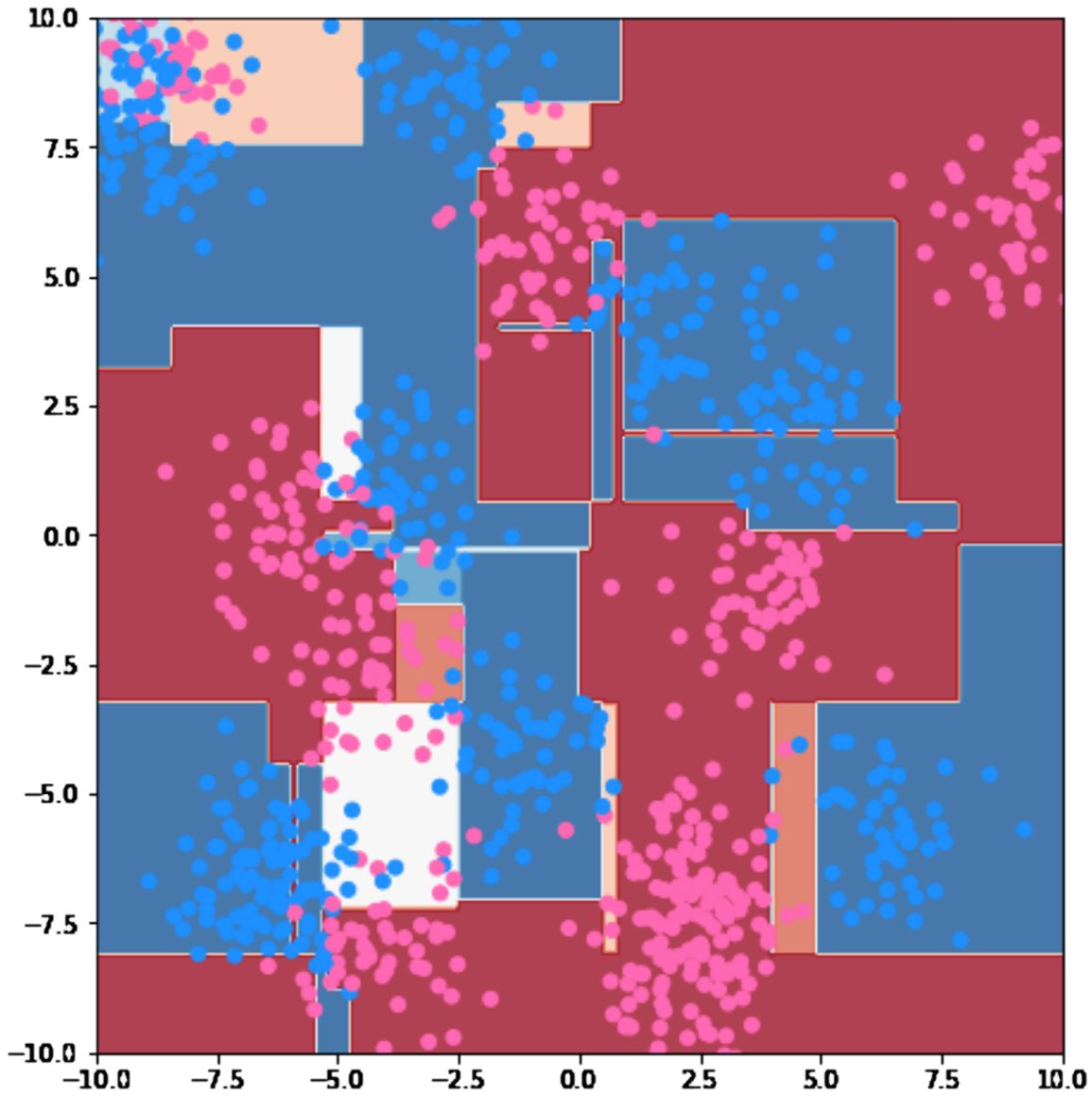
Decision tree, depth=7

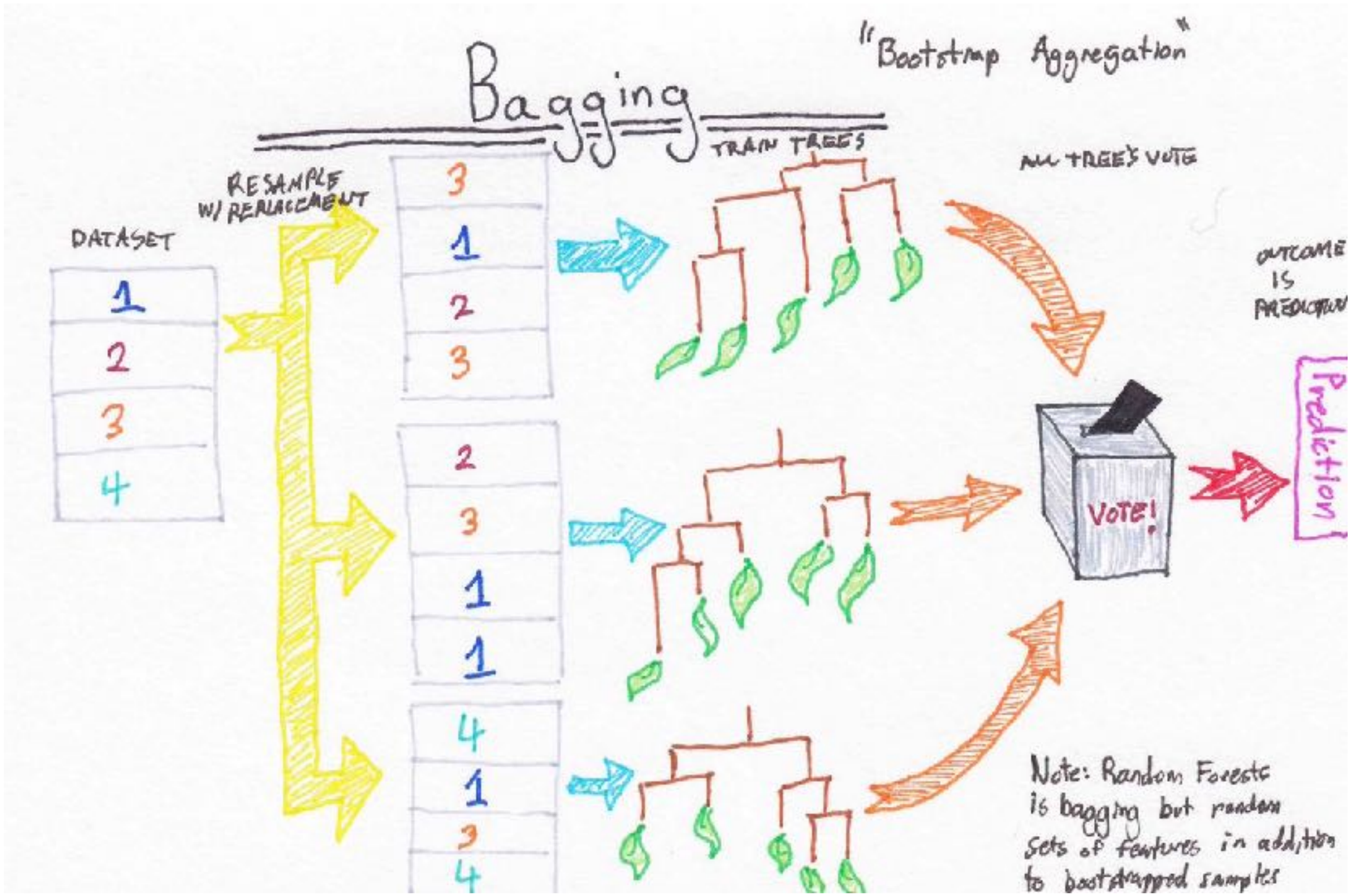


Decision tree, depth=8

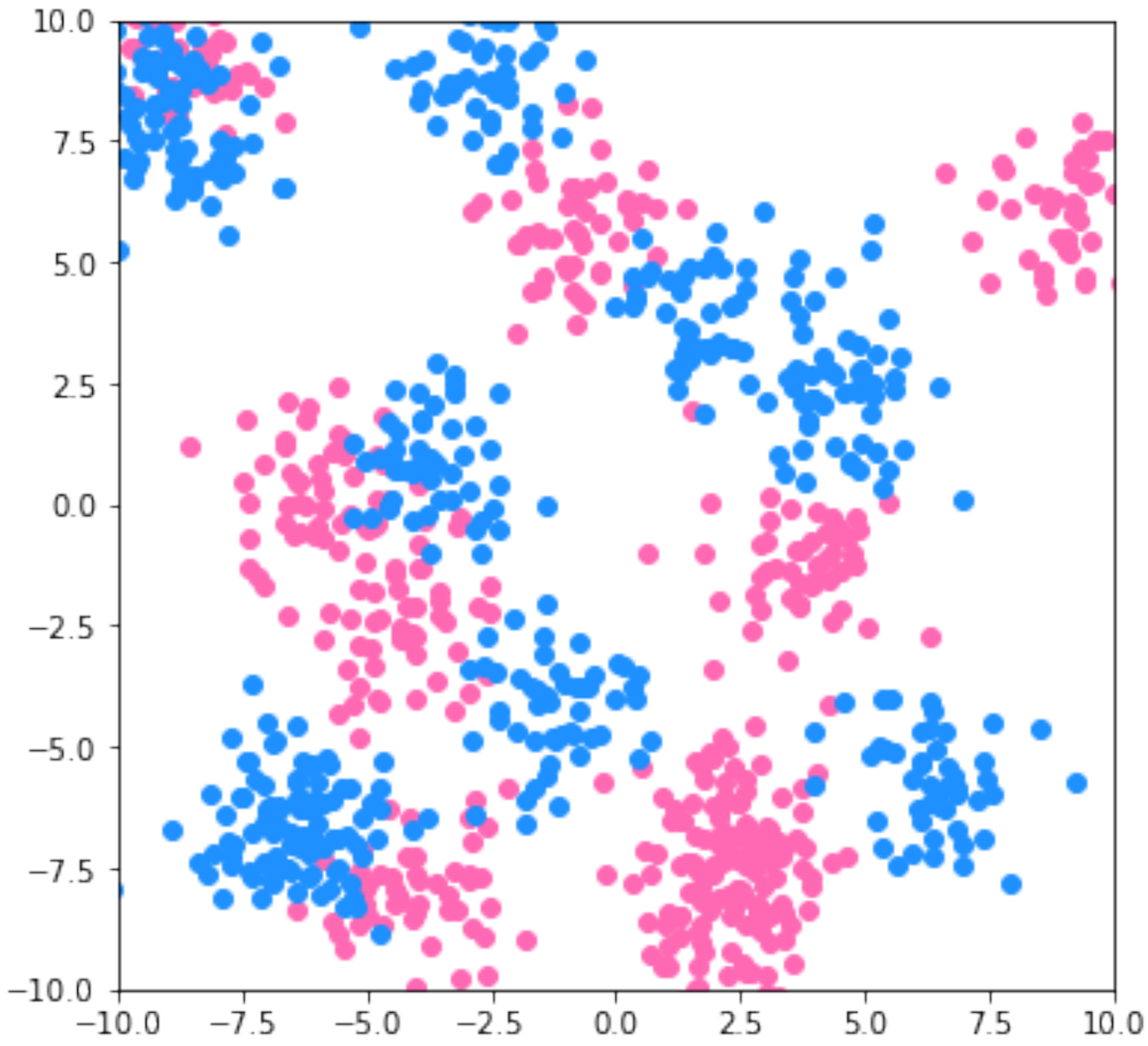


Decision tree, depth=9

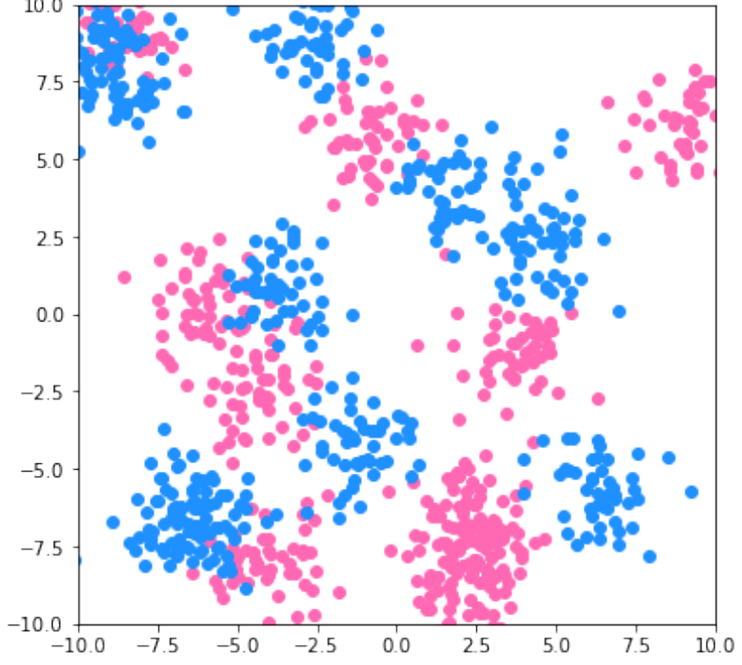



Bagging
"Bootstrap Aggregation"
DATASET
1
2
3
4
RESAMPLE W/ REPLACEMENT
3
1
2
3
2
3
1
1
4
1
3
4
TRAIN TREES
ALL TREES VOTE
VOTE!
OUTCOME IS PREDICTION
Prediction
Note: Random Forests is bagging but random sets of features in addition to bootstrapped samples

The original set

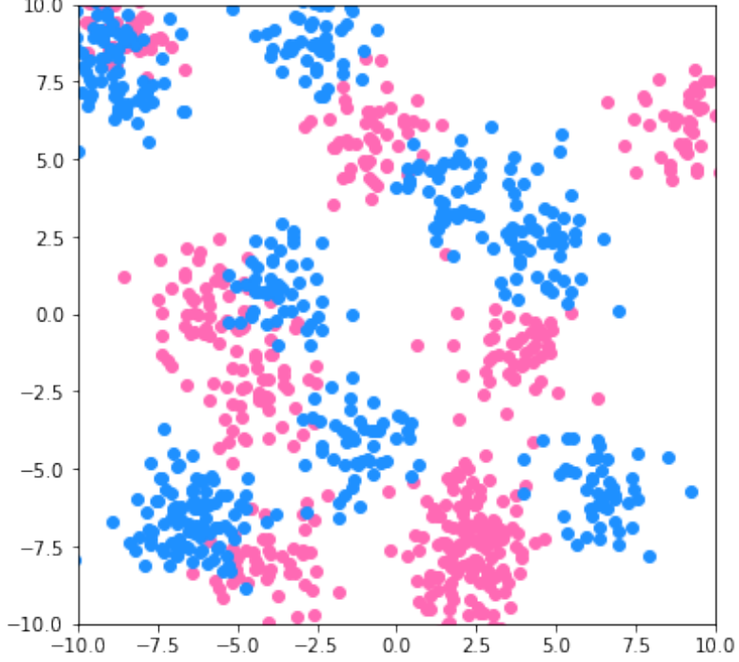


BOOSTRAP!!

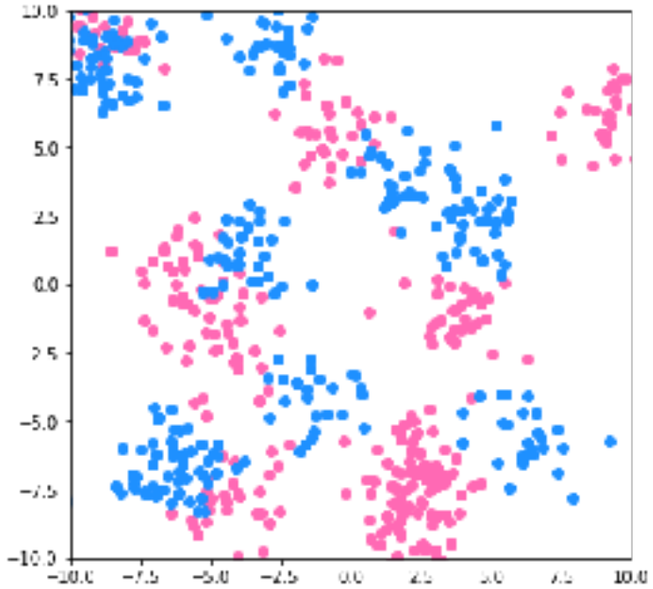


BOOSTRAP!!

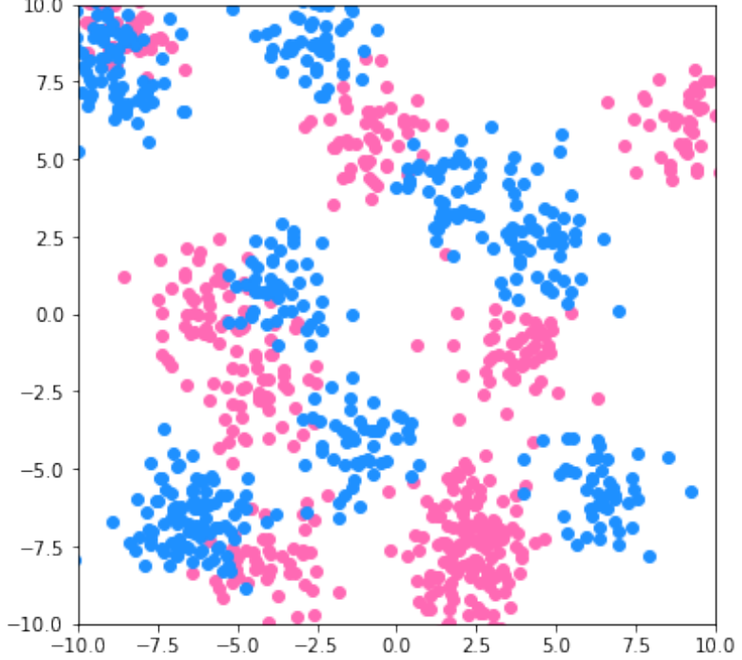
BOOSTRAP!!



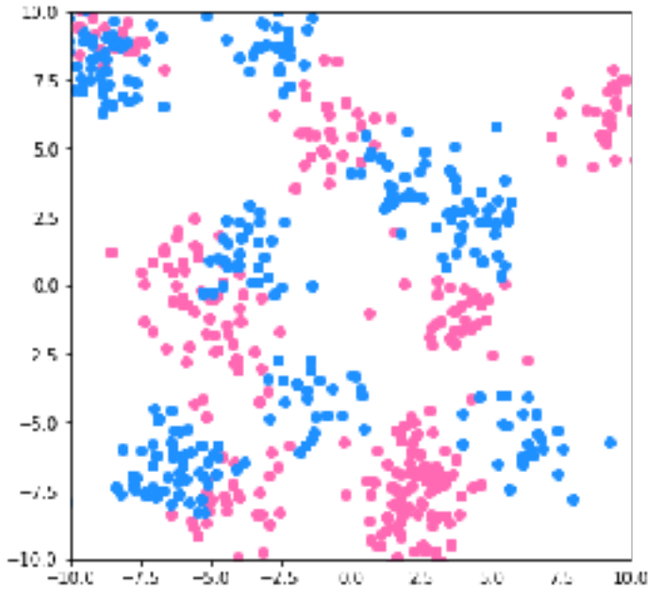
BOOSTRAP!!

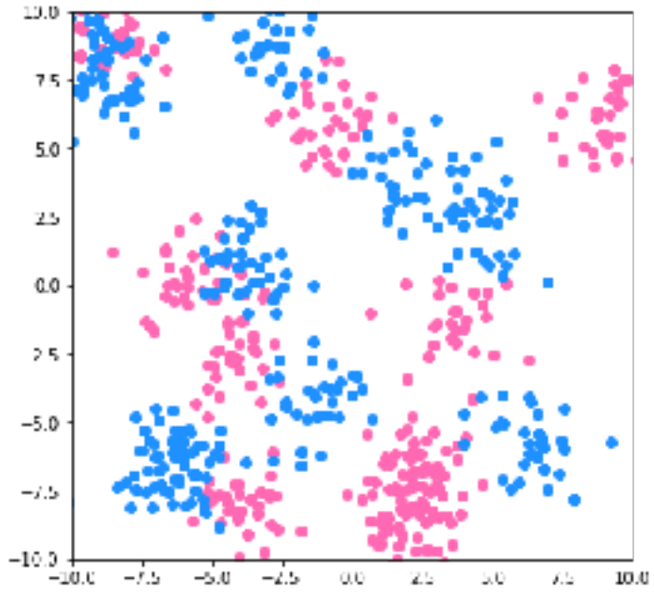


BOOSTRAP!!

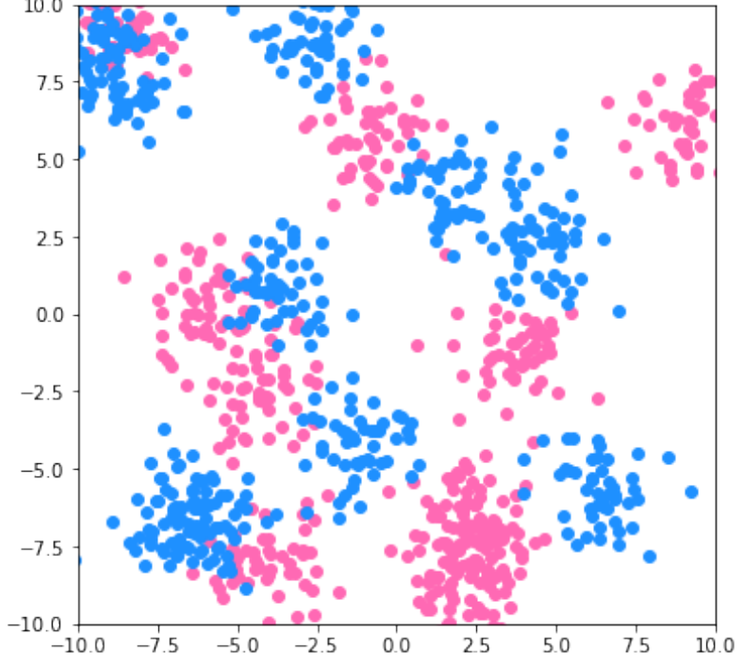


BOOSTRAP!!

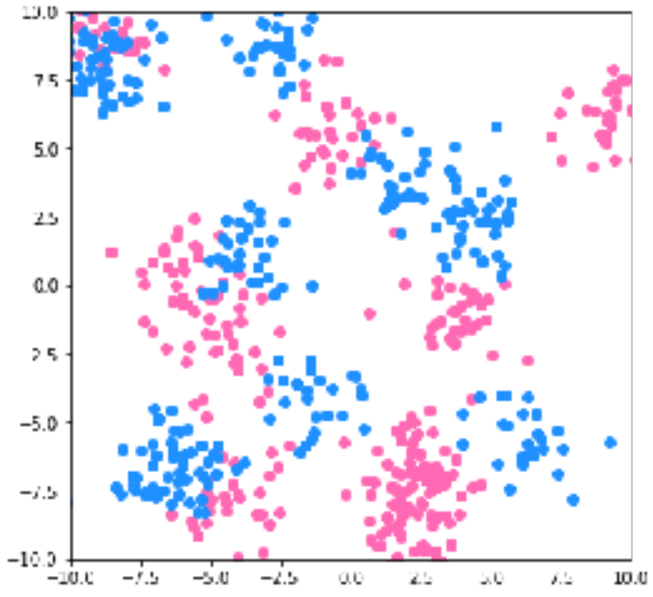


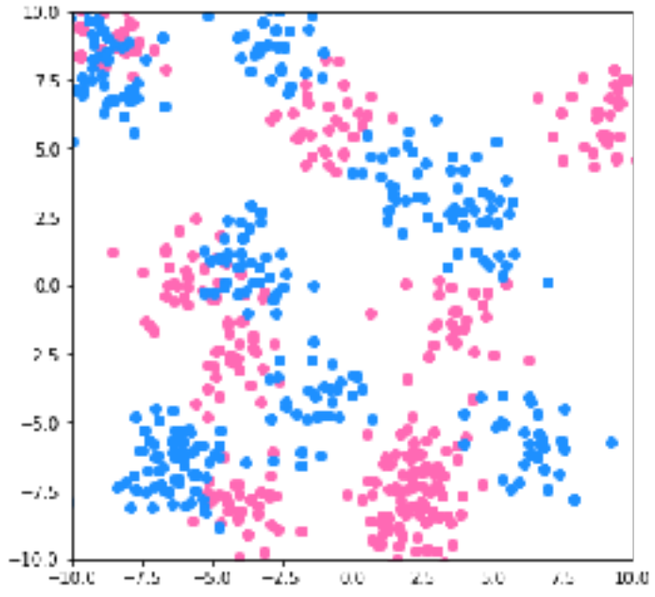


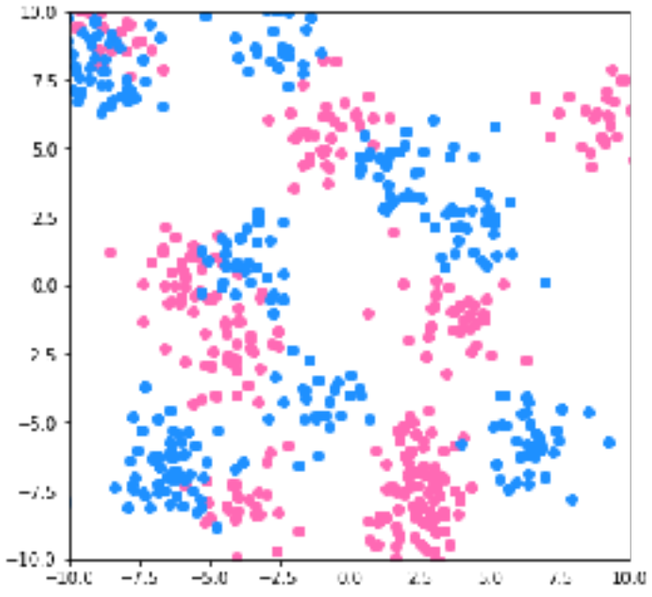
BOOSTRAP!!



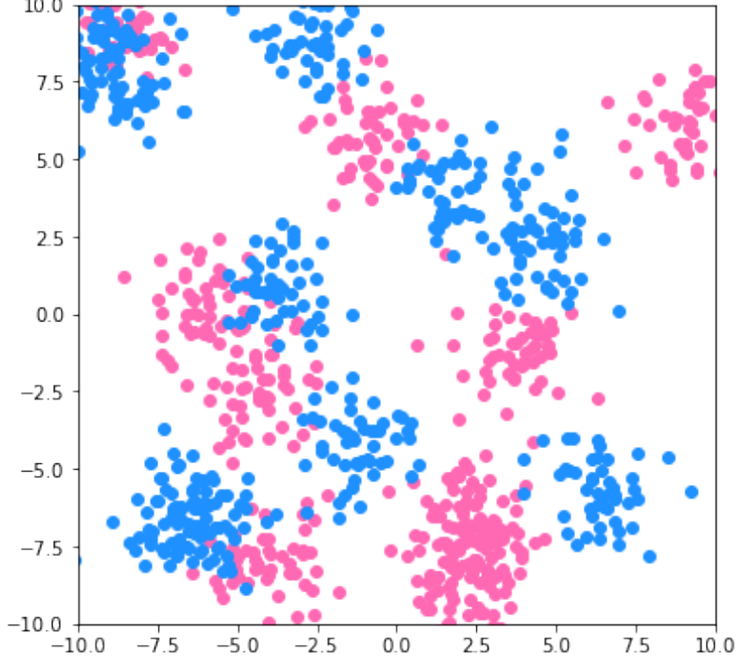
BOOSTRAP!!



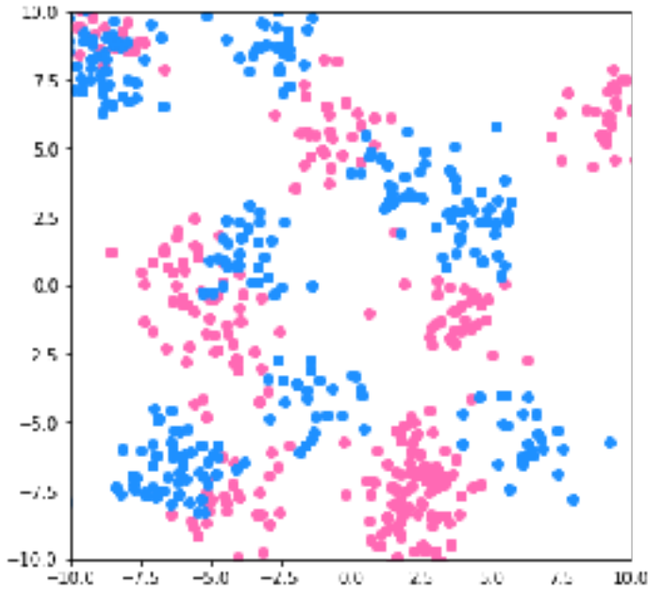


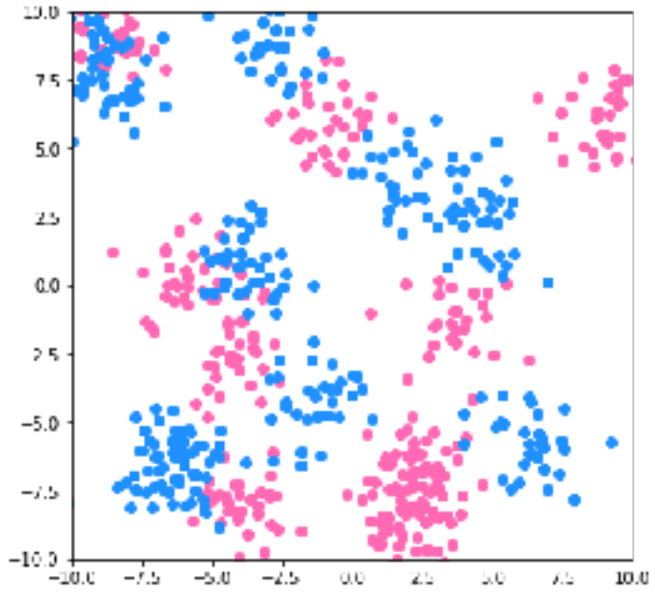


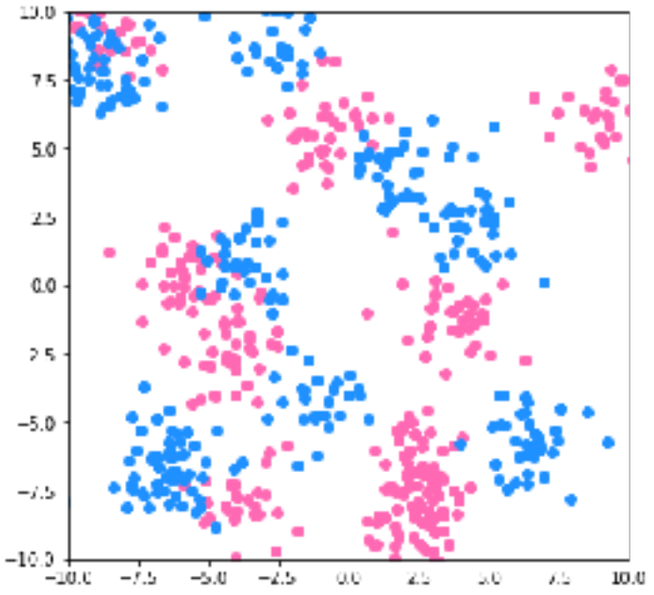
BOOSTRAP!!

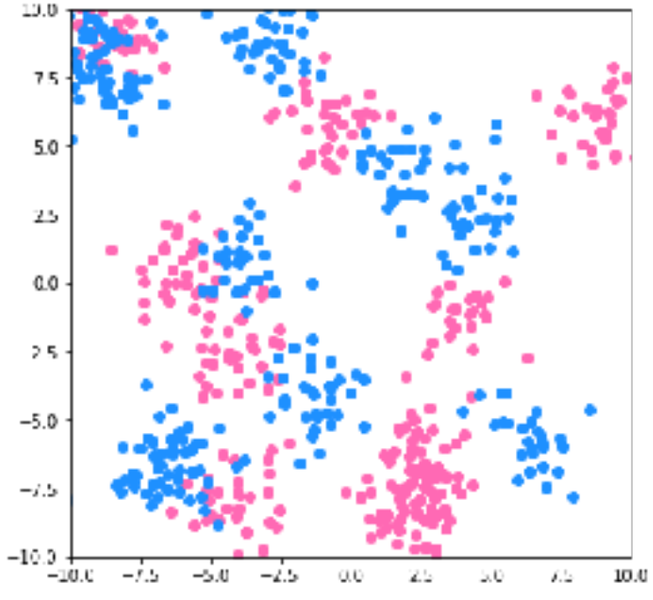


BOOSTRAP!!

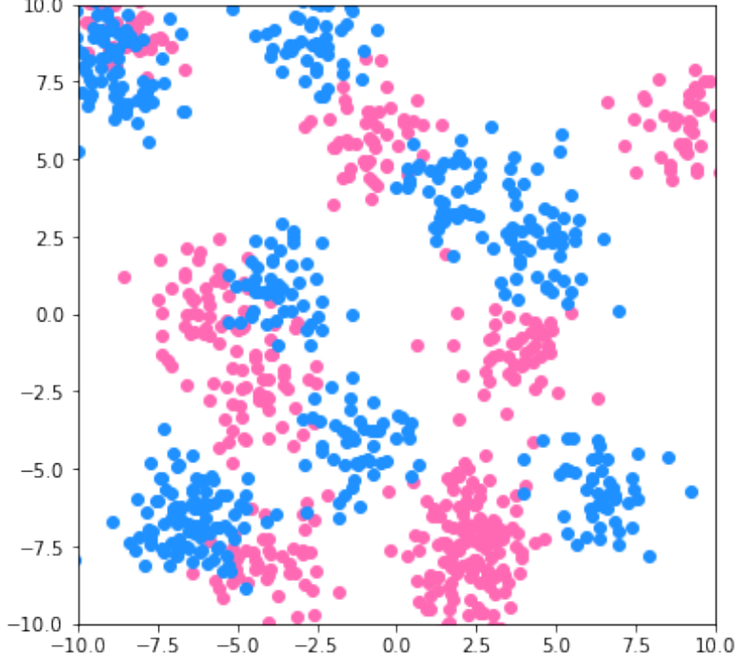




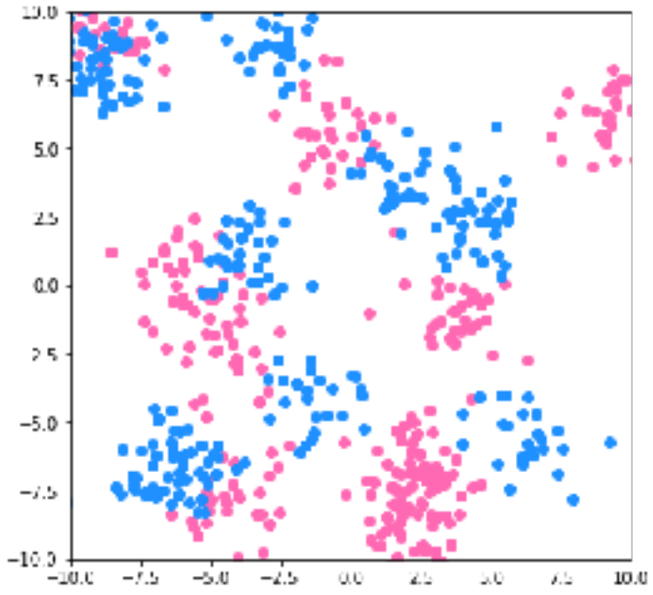


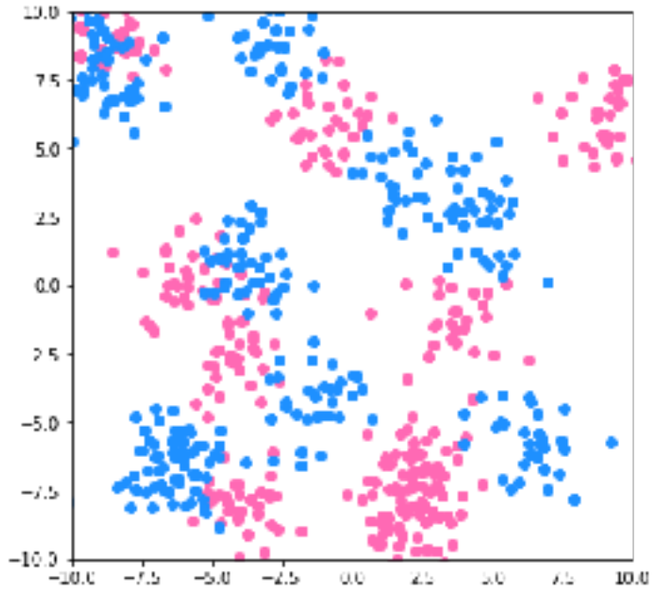


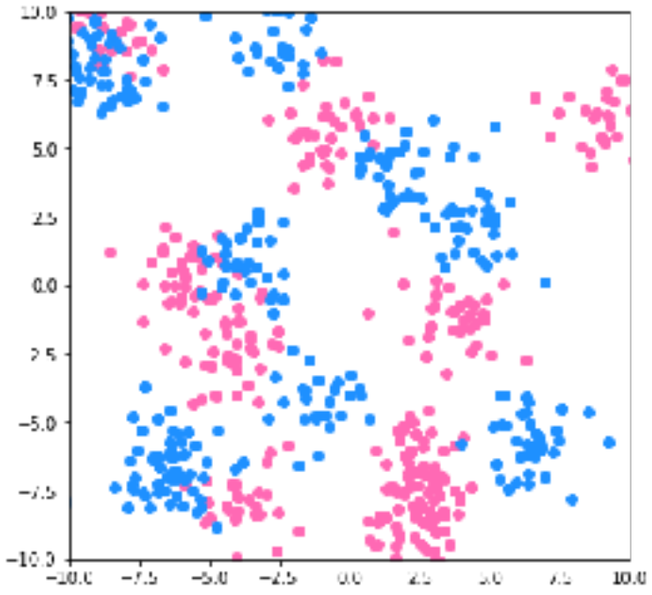
BOOSTRAP!!

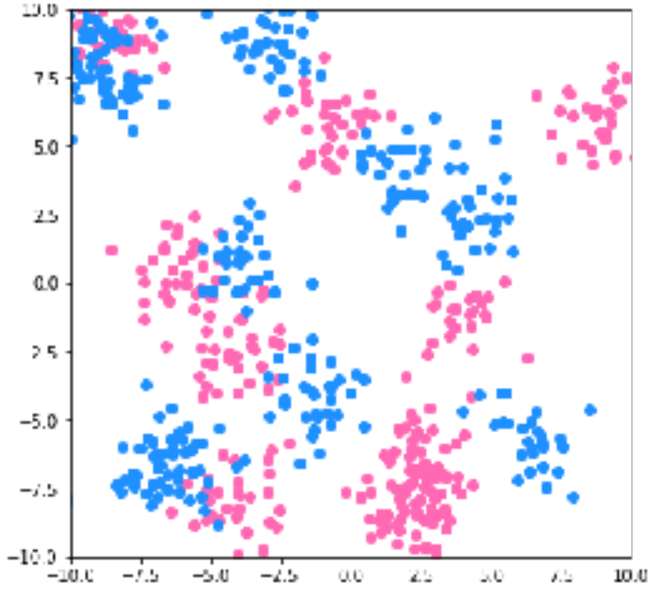


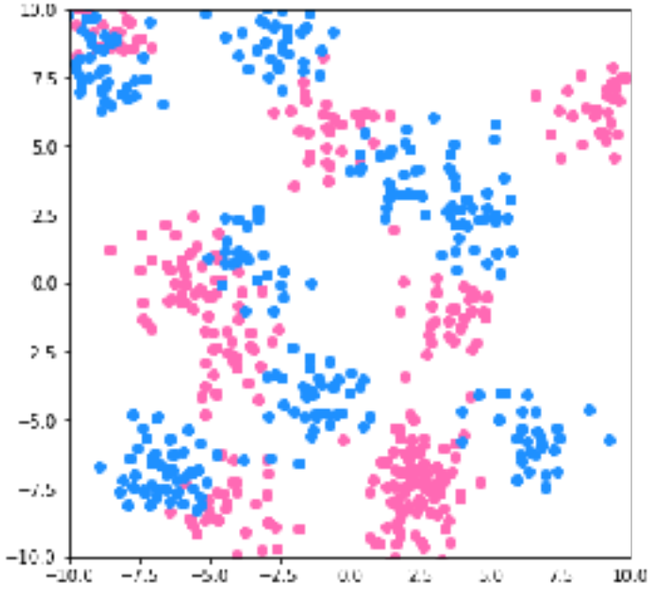
BOOSTRAP!!



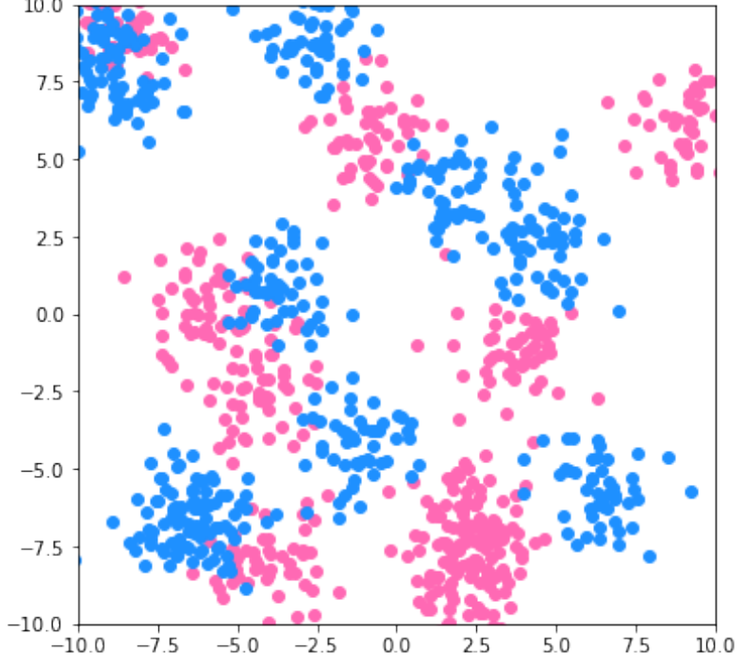




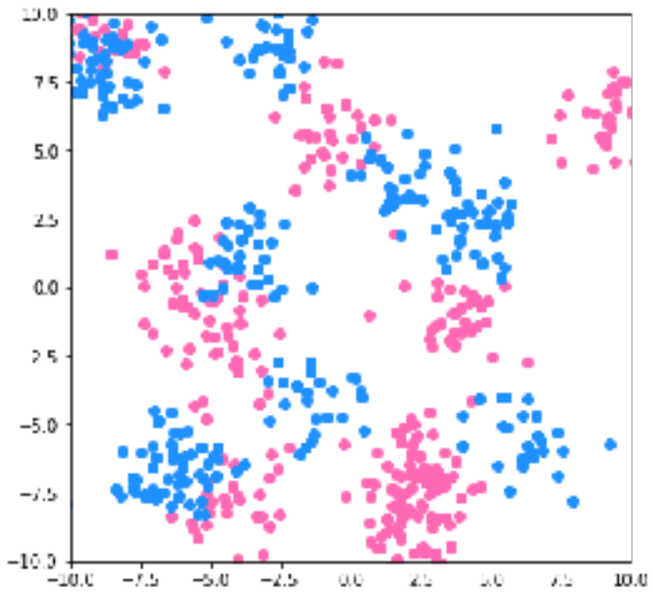


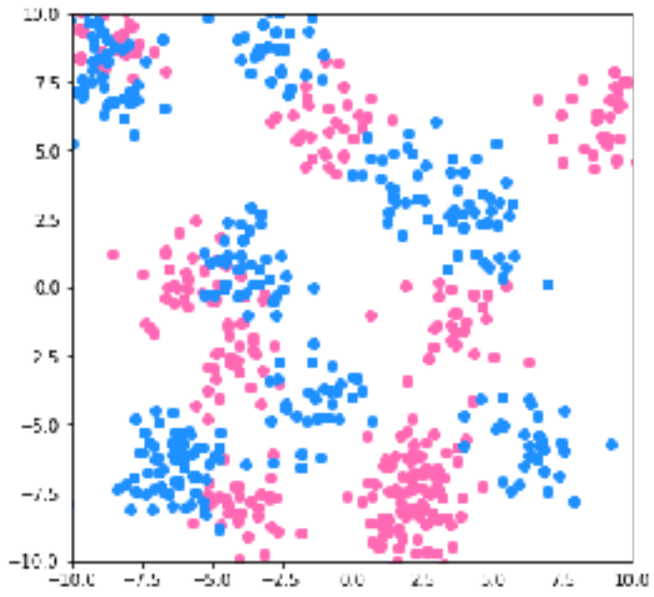


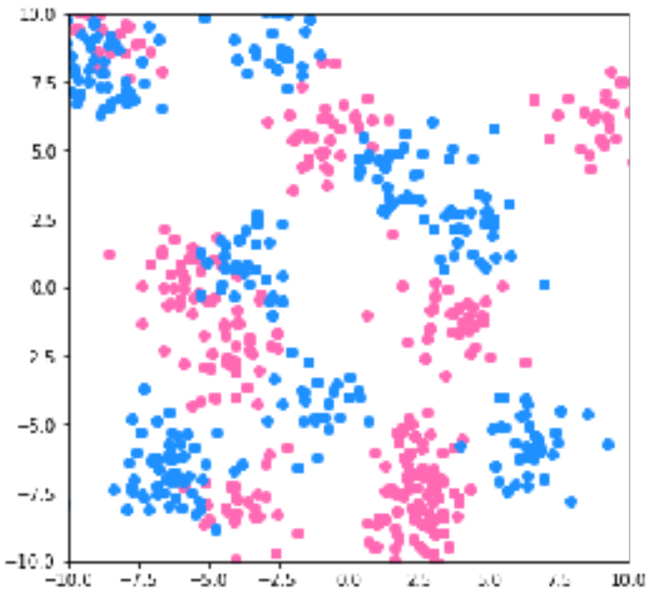
BOOSTRAP!!

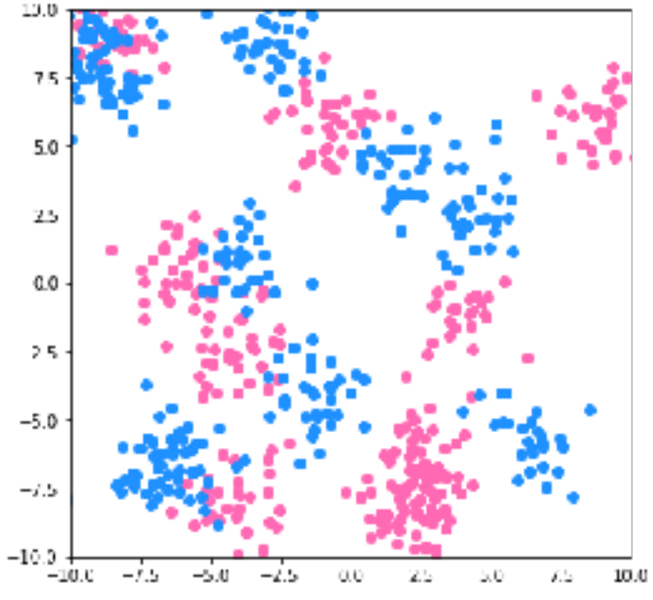


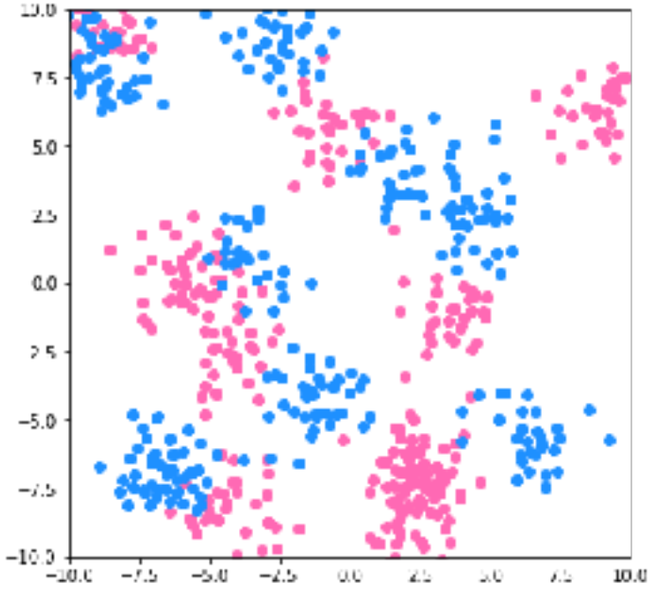
BOOSTRAP!!

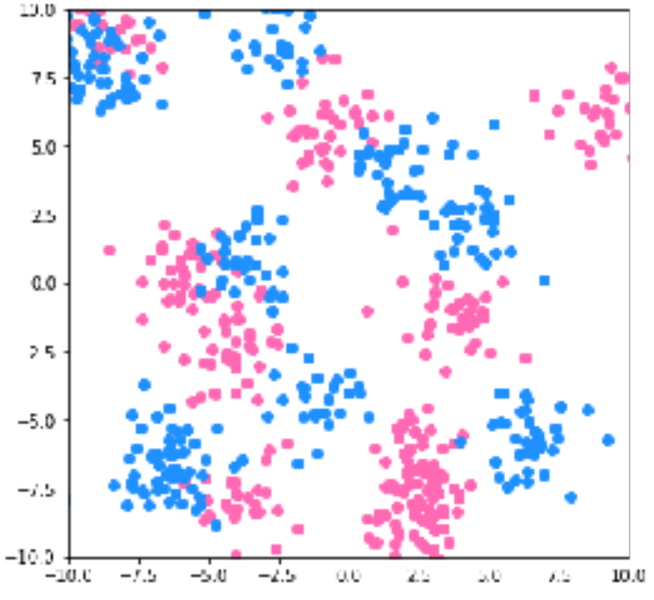




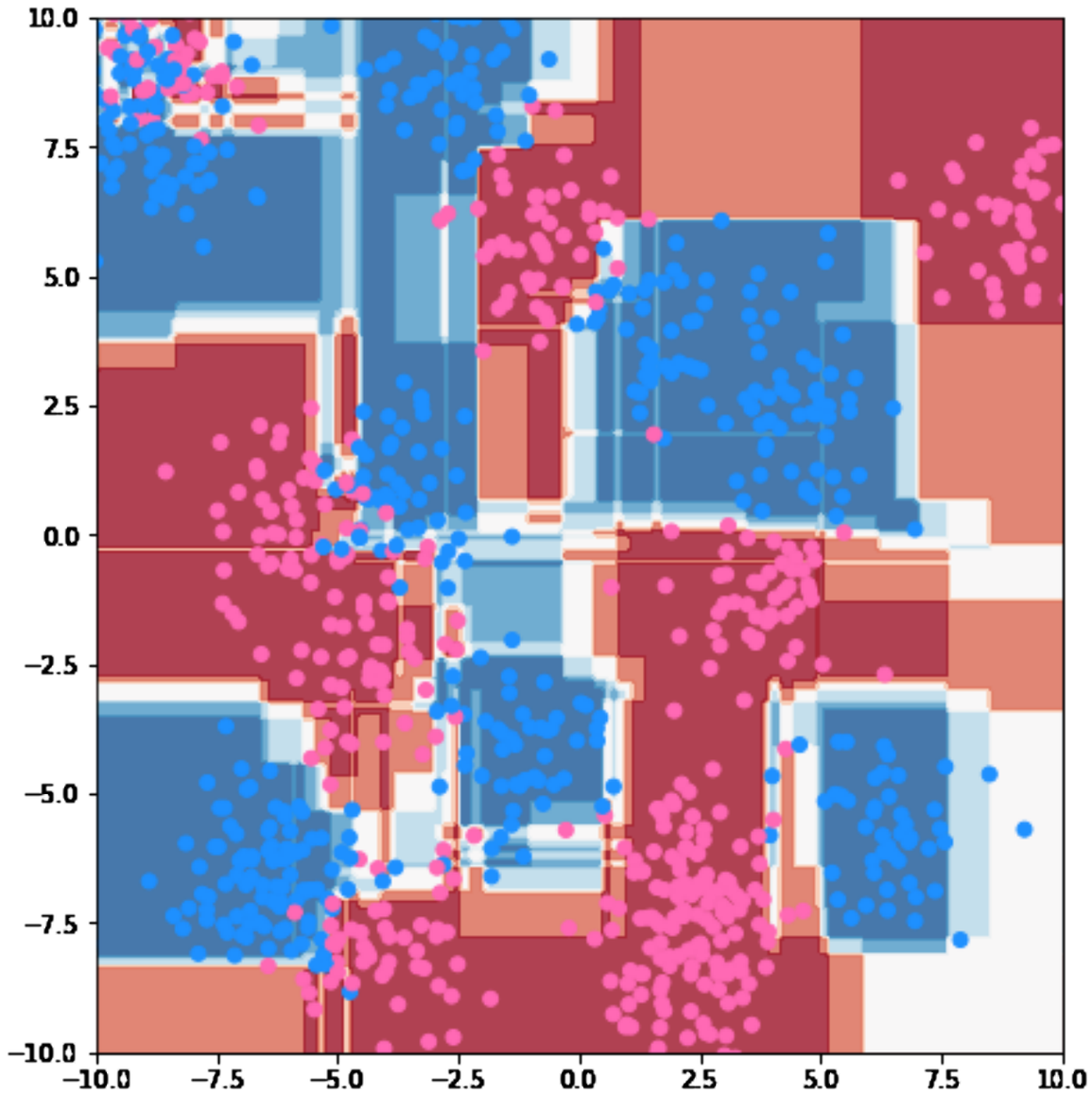




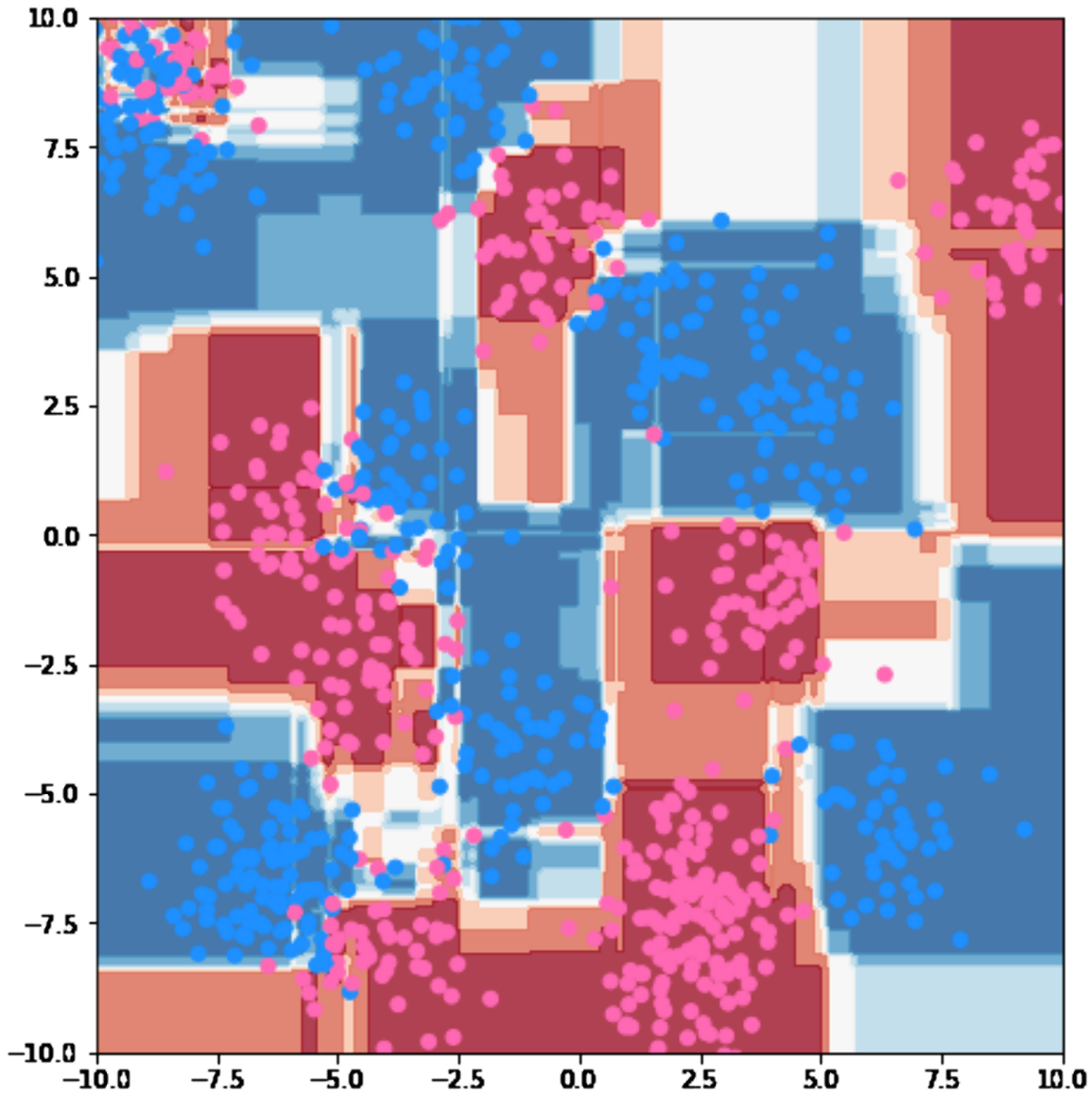




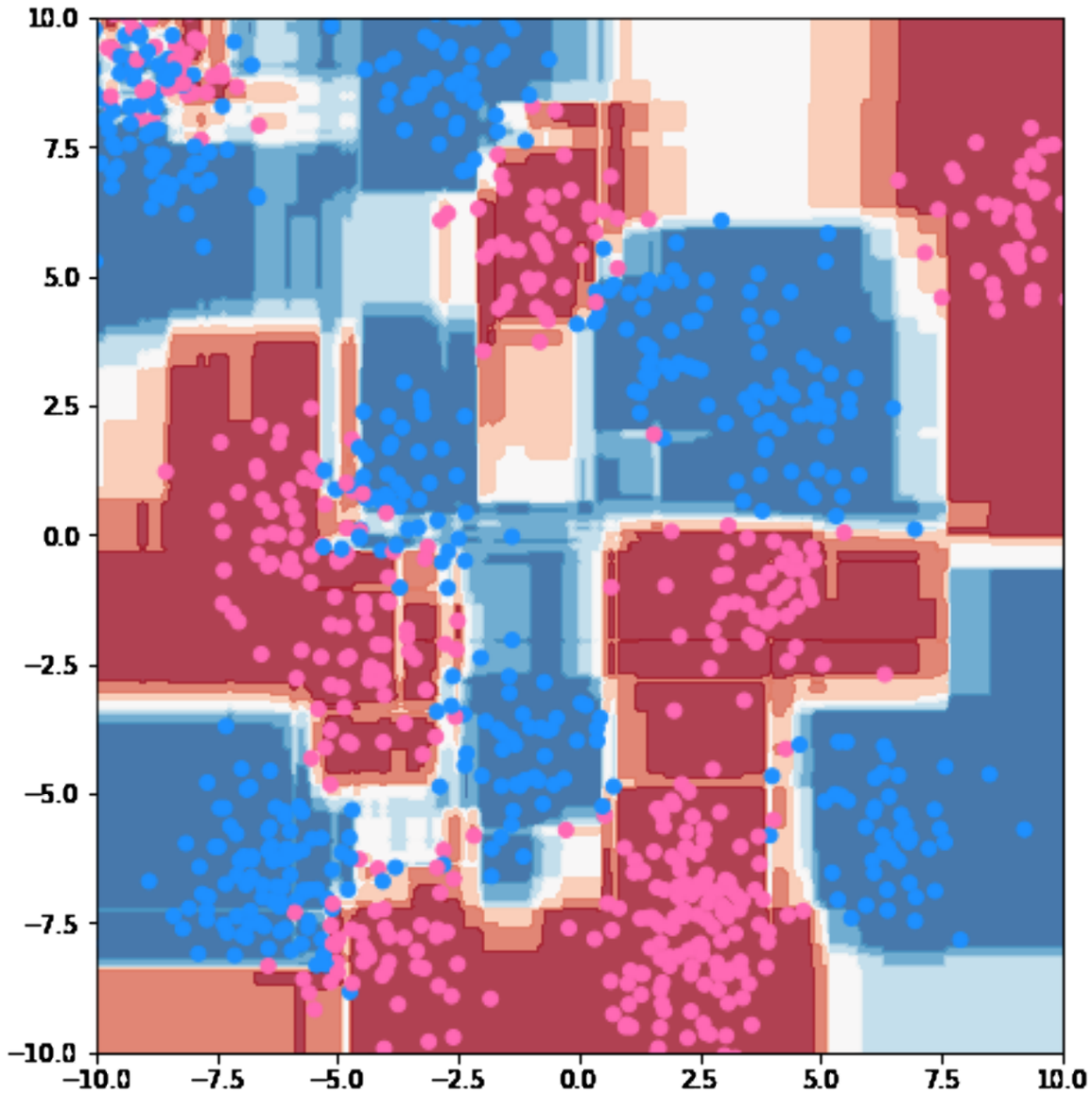
Bagging 5 Decision tree!



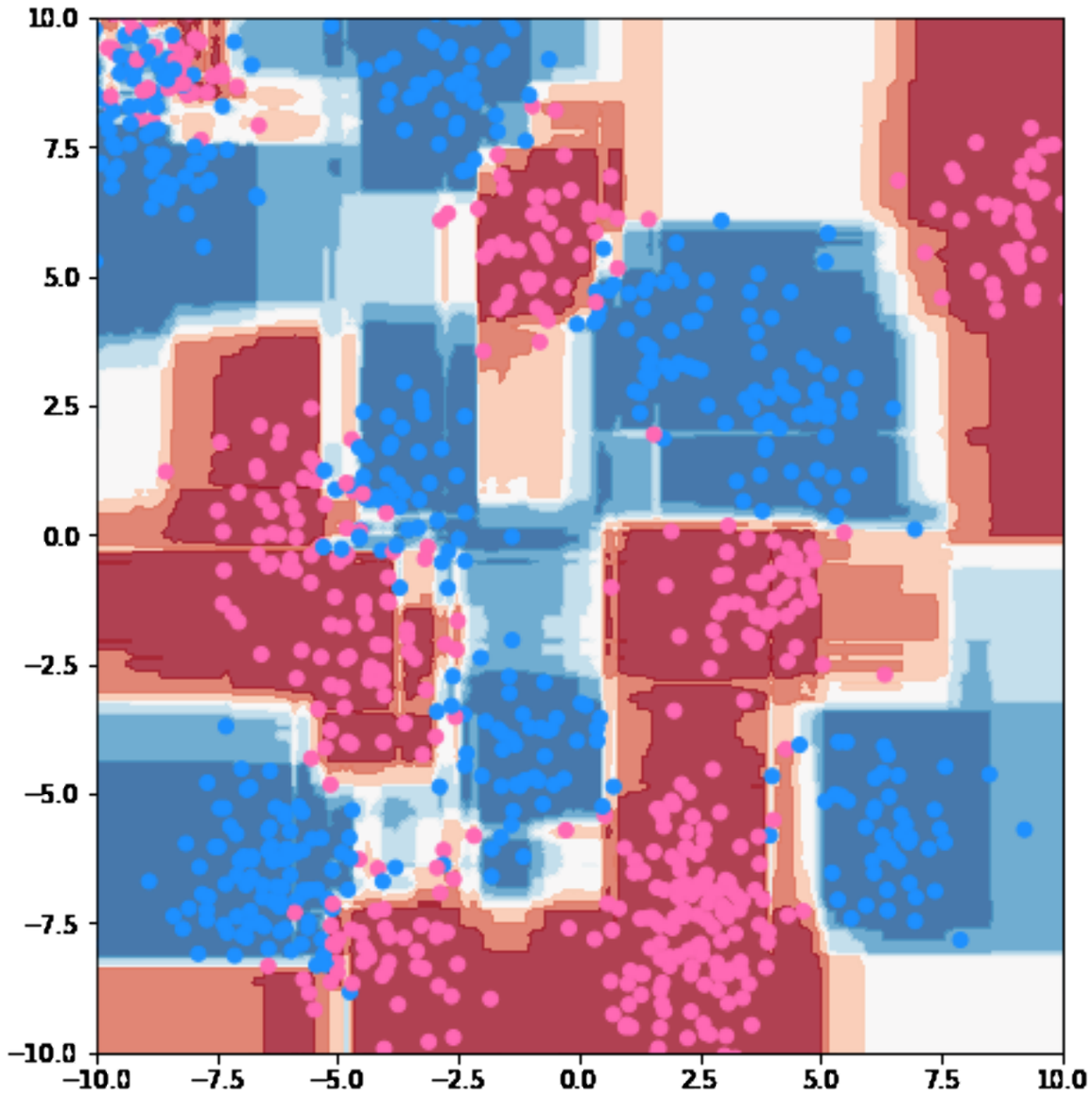
Bagging 10 Decision tree!



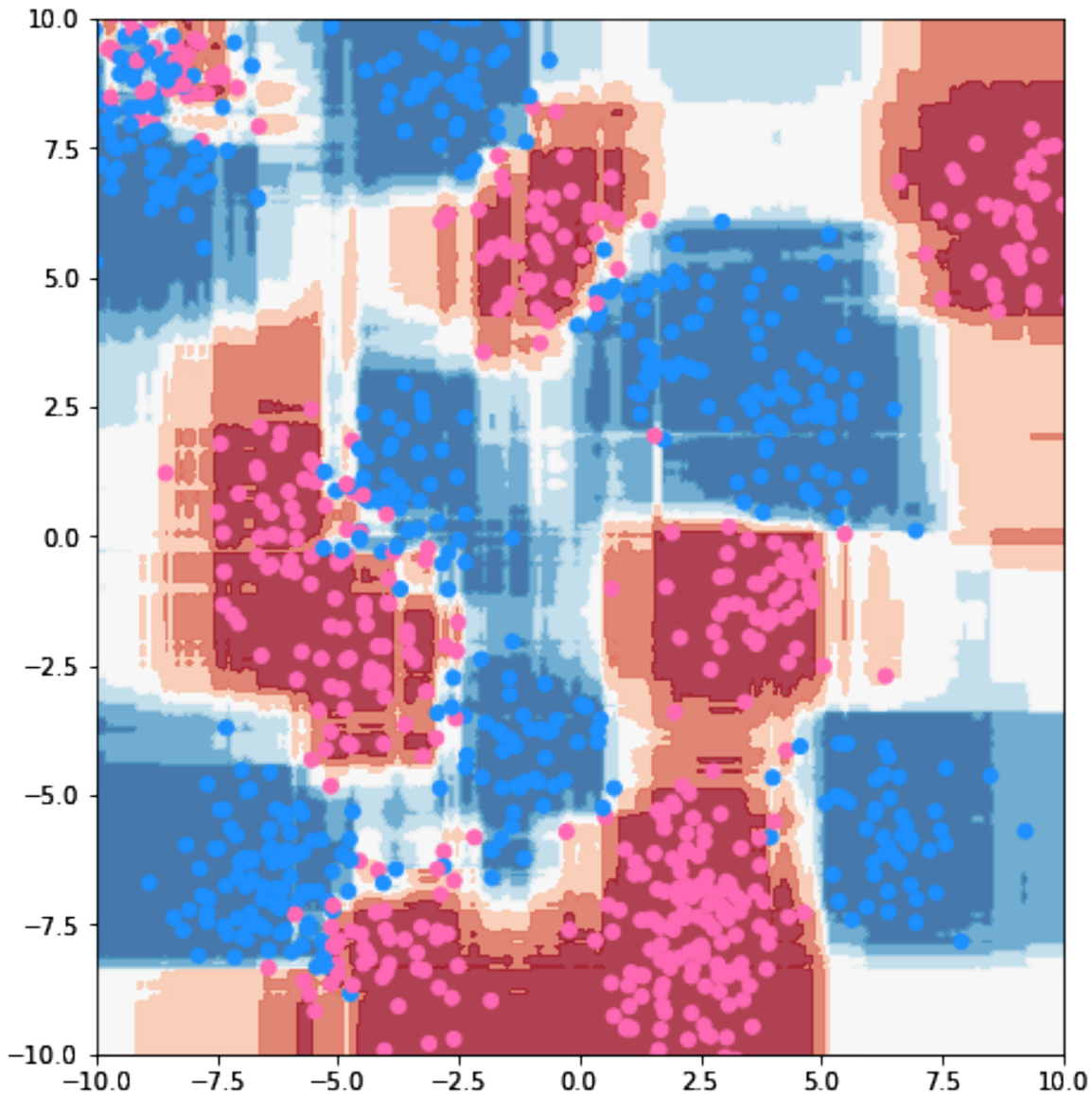
Bagging 20 Decision tree!



Bagging 20 Decision tree!



In practice: Random Forrest



Further randomization: at each candidate split in the learning process, a random subset of the features. For a classification problem with p features, $\sqrt{p}$ **(rounded down)** features are used in each split

Boosting

Can we make many dumb learners smart?

Boosting

Can we make many dumb learners smart?

Yes (but you need a smart leader!)

Can we make many dumb learners smart?

Can we make many dumb learners smart?

Kearns and Valiant ’88:

- Does weak learnability imply strong learnability?

In other words, can we boost from weak to strong?

Can we make many dumb learners smart?

Kearns and Valiant ’88:
- Does weak learnability imply strong learnability?
In other words, can we boost from weak to strong?

Schapire ’89
Freund and Schapire ’95
- Yes, with adaBoost

How to combine many weak classifiers?

Ingredients:

(i) T classifiers $h_t(.)$, each of them slightly better than random
(ii) A training set with N labeled examples

Adaptive linear combination of classifiers

$$H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

ADABOOST

Exponential loss:

$$\mathcal{R} = \sum_i e^{-Y_i H(\vec{x}_i)}$$

Goal: start from t=0, add new classifiers and

(i) Adapt the weight alpha at each steps
(ii) re-weight the instances in the training set : more weight to ill-classified instances
(each new classifier concentrate on badly classified examples)

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i e^{-Y_i H(\vec{x}_i)}$$

$$H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i e^{-Y_i H(\vec{x}_i)} \qquad H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

$$\mathcal{R} = \sum_i e^{-Y_i \sum_{t=1}^{\tau-1} \alpha_t h_t(\vec{x}_i) - Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i e^{-Y_i H(\vec{x}_i)} \qquad H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

$$\mathcal{R} = \sum_i e^{-Y_i \overbrace{\sum_{t=1}^{\tau-1} \alpha_t h_t(\vec{x}_i)}^{\lambda_i^\tau} - Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i e^{-Y_i H(\vec{x}_i)} \qquad H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

$$\mathcal{R} = \sum_i e^{-Y_i \overbrace{\sum_{t=1}^{\tau-1} \alpha_t h_t(\vec{x}_i)}^{\lambda_i^\tau} - Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = \sum_i e^{-Y_i \lambda_i^\tau} e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i e^{-Y_i H(\vec{x}_i)} \qquad H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

$$\mathcal{R} = \sum_i e^{-Y_i \overbrace{\sum_{t=1}^{\tau-1} \alpha_t h_t(\vec{x}_i)}^{\lambda_i^\tau} - Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = \sum_i \underbrace{e^{-Y_i \lambda_i^\tau}}_{\omega_i^\tau} e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i e^{-Y_i H(\vec{x}_i)} \qquad H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

$$\mathcal{R} = \sum_i e^{-Y_i \overbrace{\sum_{t=1}^{\tau-1} \alpha_t h_t(\vec{x}_i)}^{\lambda_i^\tau} - Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = \sum_i \underbrace{e^{-Y_i \lambda_i^\tau}}_{\omega_i^\tau} e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i e^{-Y_i H(\vec{x}_i)} \qquad H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

$$\mathcal{R} = \sum_i e^{-Y_i \overbrace{\sum_{t=1}^{\tau-1} \alpha_t h_t(\vec{x}_i)}^{\lambda_i^\tau} - Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = \sum_i \underbrace{e^{-Y_i \lambda_i^\tau}}_{\omega_i^\tau} e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

ω_i^τ **= weights for each instances at time τ**

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i \omega_i^{\tau} e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_{i \in \mathrm{OK}} \omega_i^\tau + e^{\alpha_\tau} \sum_{i \in \mathrm{NOT\ OK}} \omega_i^\tau$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_{i \in \mathrm{OK}} \omega_i^\tau + e^{\alpha_\tau} \sum_{i \in \mathrm{NOT\ OK}} \omega_i^\tau = e^{-\alpha_\tau} \sum_i \mathbf{1}(Y_i = h_i)\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_{i \in \mathrm{OK}} \omega_i^\tau + e^{\alpha_\tau} \sum_{i \in \mathrm{NOT\ OK}} \omega_i^\tau = e^{-\alpha_\tau} \sum_i \mathbf{1}(Y_i = h_i)\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_i (1 - \mathbf{1}(Y_i \neq h_i))\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_{i \in \mathrm{OK}} \omega_i^\tau + e^{\alpha_\tau} \sum_{i \in \mathrm{NOT\ OK}} \omega_i^\tau = e^{-\alpha_\tau} \sum_i \mathbf{1}(Y_i = h_i)\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_i (1 - \underbrace{\mathbf{1}(Y_i \neq h_i)}_{\epsilon_t})\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_{i \in \text{OK}} \omega_i^\tau + e^{\alpha_\tau} \sum_{i \in \text{NOT OK}} \omega_i^\tau = e^{-\alpha_\tau} \sum_i \mathbf{1}(Y_i = h_i)\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_i (1 - \underbrace{\mathbf{1}(Y_i \neq h_i)}) \omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

$$\epsilon_t$$

ϵ_t is what the classifier h_t(.) is trying to minimise

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_{i \in \text{OK}} \omega_i^\tau + e^{\alpha_\tau} \sum_{i \in \text{NOT OK}} \omega_i^\tau = e^{-\alpha_\tau} \sum_i \mathbf{1}(Y_i = h_i)\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_i (1 - \underbrace{\mathbf{1}(Y_i \neq h_i)}_{\epsilon_t})\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

ϵ_t is what the classifier $h_t(.)$ is trying to minimise

Let us choose α in order to minize the global risk

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_{i \in \text{OK}} \omega_i^\tau + e^{\alpha_\tau} \sum_{i \in \text{NOT OK}} \omega_i^\tau = e^{-\alpha_\tau} \sum_i \mathbf{1}(Y_i = h_i)\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_i (1 - \underbrace{\mathbf{1}(Y_i \neq h_i)}_{\epsilon_t})\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

ϵ_t **is what the classifier $h_t(.)$ is trying to minimise**

Let us choose α in order to minize the global risk

$$\mathcal{R} = e^{-\alpha_\tau} - e^{-\alpha_\tau}\epsilon_t + e^{\alpha_\tau}\epsilon_t = e^{-\alpha_\tau}(1 - \epsilon_t) + e^{\alpha_\tau}\epsilon_t$$

How does one set the weight and α at each time steps?

Assume we have done the job until time τ !

$$\mathcal{R} = \sum_i \omega_i^\tau e^{-Y_i \alpha_\tau h_\tau(\vec{x}_i)}$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_{i \in \mathrm{OK}} \omega_i^\tau + e^{\alpha_\tau} \sum_{i \in \mathrm{NOT\ OK}} \omega_i^\tau = e^{-\alpha_\tau} \sum_i \mathbf{1}(Y_i = h_i)\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

$$\mathcal{R} = e^{-\alpha_\tau} \sum_i (1 - \underbrace{\mathbf{1}(Y_i \neq h_i)}_{\epsilon_t})\omega_i^\tau + e^{\alpha_\tau} \sum_i \omega_i^\tau \mathbf{1}(Y_i \neq h_i)$$

ϵ_t **is what the classifier h_t(.) is trying to minimise**

Let us choose α in order to minize the global risk

$$\mathcal{R} = e^{-\alpha_\tau} - e^{-\alpha_\tau}\epsilon_t + e^{\alpha_\tau}\epsilon_t = e^{-\alpha_\tau}(1 - \epsilon_t) + e^{\alpha_\tau}\epsilon_t$$

$$\partial_{\alpha_t}\mathcal{R} = 0 \rightarrow \alpha_t = \frac{1}{2}\log\frac{1 - \epsilon_t}{\epsilon_t}$$

ADABOOST TRAINING

$$\forall\ i : \omega_i^{t=1} = \frac{1}{n}$$

$$\text{for t} = 1, \ldots, \text{T}_{\max} \qquad \textbf{Do}$$

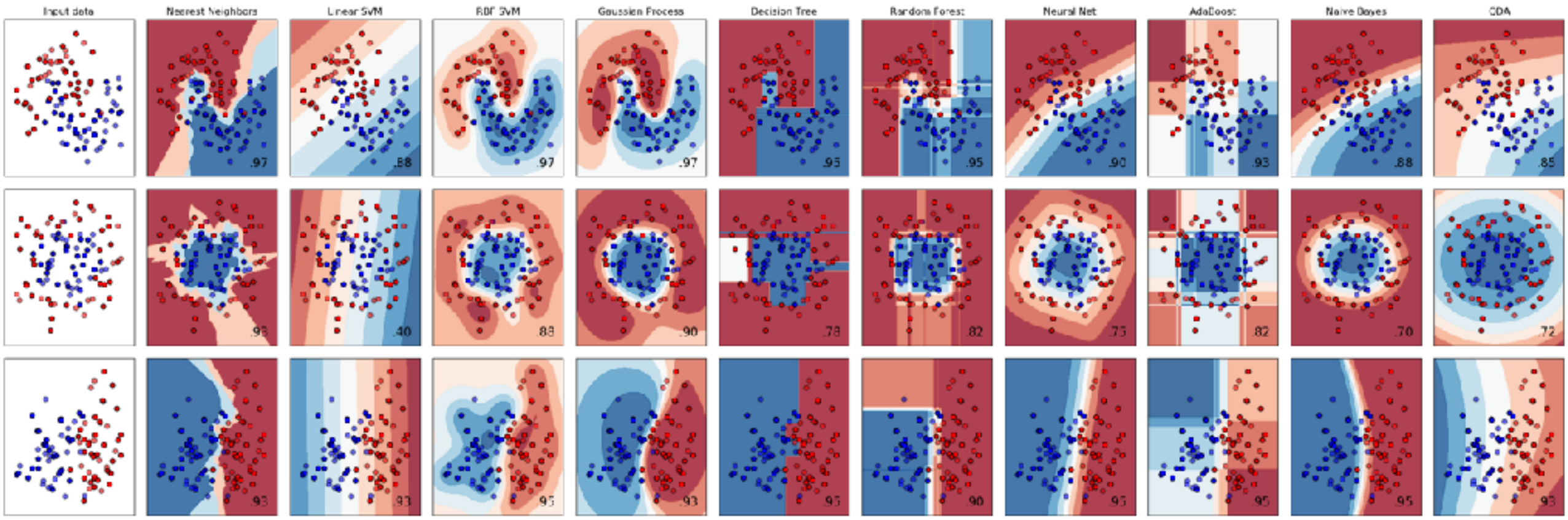
Run a classifier for : $\epsilon_t = \left[\sum_i \omega_i^{\tau} \mathbf{1}(Y_i \neq h_i))\right]$

Set: $\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$

Agregate classifier: $H(.) = H^{t-1}(.) + \alpha_t h_t(.)$

Update weights: $\omega_i^{t+1} = \frac{\omega_i^t e^{-Y_i \alpha_t h_t(\vec{x}_i)}}{\sum_i \omega_i^t e^{-Y_i \alpha_t h_t(\vec{x}_i)}}$

A tour on standard machine learning classifiers



From sk-learn

PCA and SVD

Blackboard lecture