Comparing Classifiers

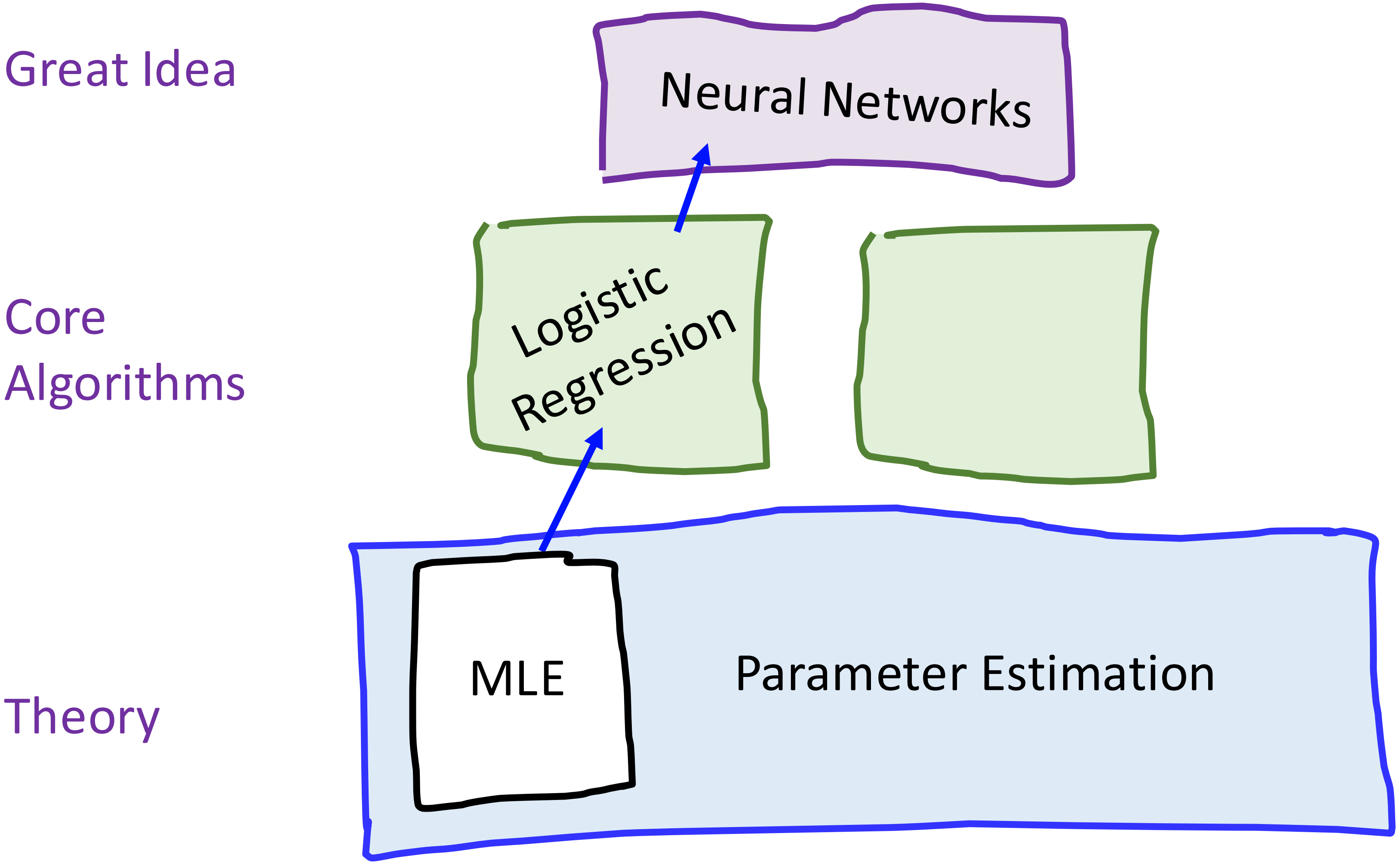
CS109, Stanford University

Learning Goals

1. Know the derivations behind logistic regression
2. Learn about other models for classification
3. Learn proper ML etiquette for comparing datasets

Review

Machine Learning in CS109



Great Idea

Core Algorithms

Theory

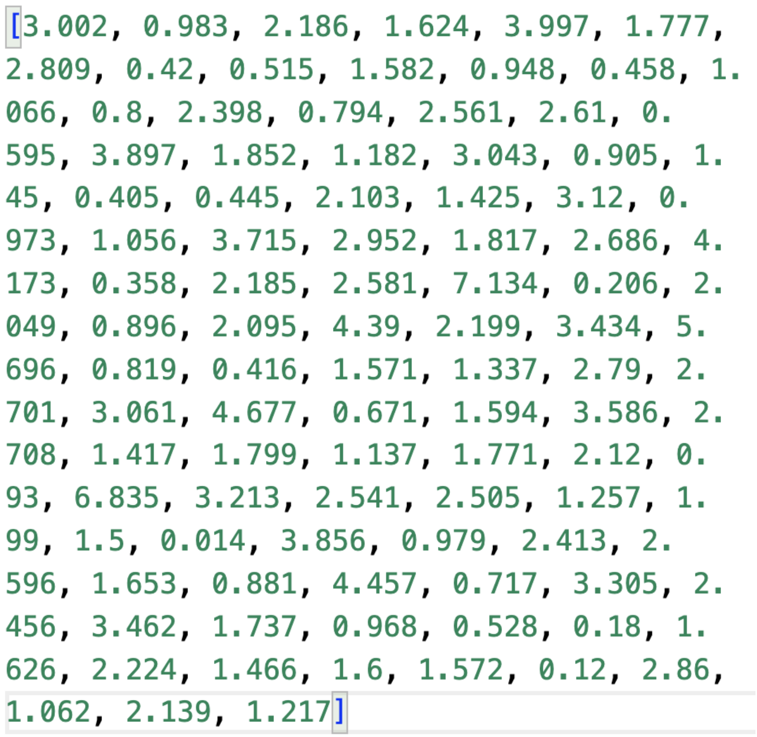
Neural Networks

Logistic Regression

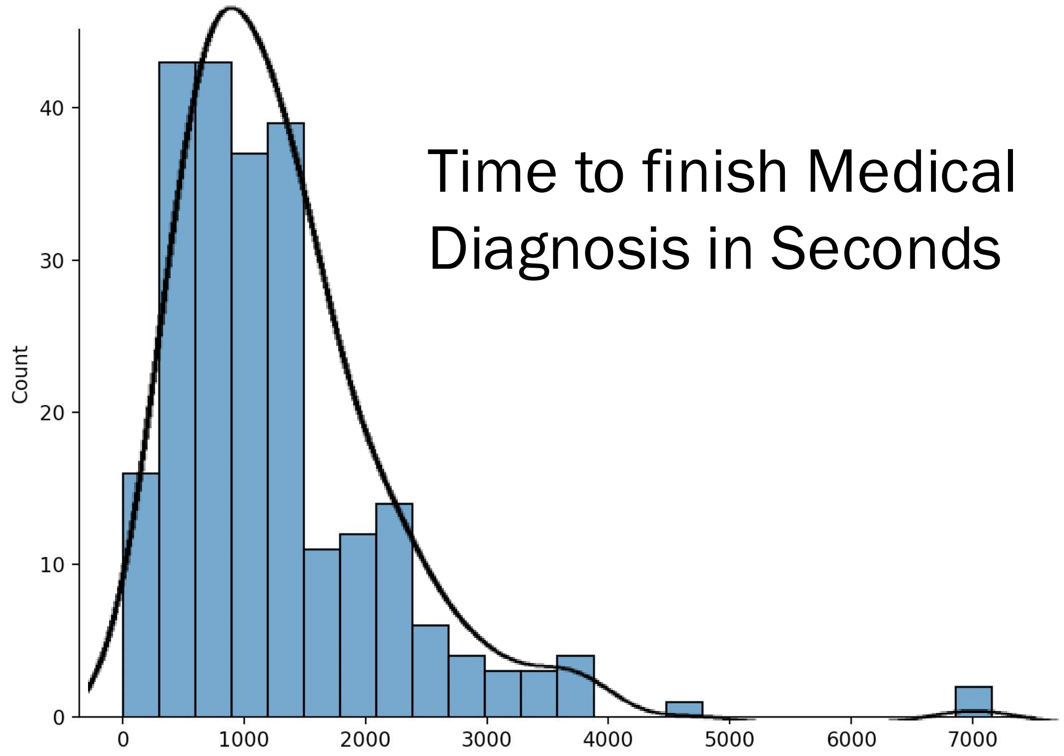
MLE

Parameter Estimation

MLE of Erlang



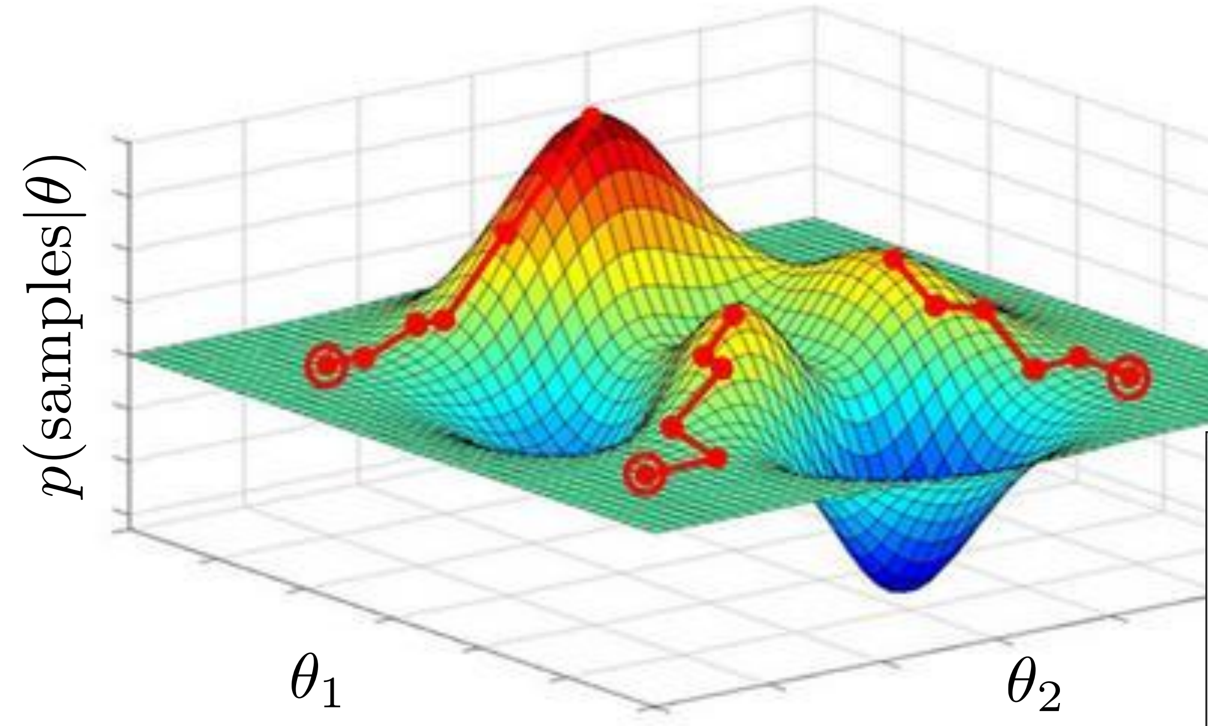
```
[3.002, 0.983, 2.186, 1.624, 3.997, 1.777,
2.809, 0.42, 0.515, 1.582, 0.948, 0.458, 1.
066, 0.8, 2.398, 0.794, 2.561, 2.61, 0.
595, 3.897, 1.852, 1.182, 3.043, 0.905, 1.
45, 0.405, 0.445, 2.103, 1.425, 3.12, 0.
973, 1.056, 3.715, 2.952, 1.817, 2.686, 4.
173, 0.358, 2.185, 2.581, 7.134, 0.206, 2.
049, 0.896, 2.095, 4.39, 2.199, 3.434, 5.
696, 0.819, 0.416, 1.571, 1.337, 2.79, 2.
701, 3.061, 4.677, 0.671, 1.594, 3.586, 2.
708, 1.417, 1.799, 1.137, 1.771, 2.12, 0.
93, 6.835, 3.213, 2.541, 2.505, 1.257, 1.
99, 1.5, 0.014, 3.856, 0.979, 2.413, 2.
596, 1.653, 0.881, 4.457, 0.717, 3.305, 2.
456, 3.462, 1.737, 0.968, 0.528, 0.18, 1.
626, 2.224, 1.466, 1.6, 1.572, 0.12, 2.86,
1.062, 2.139, 1.217]
```



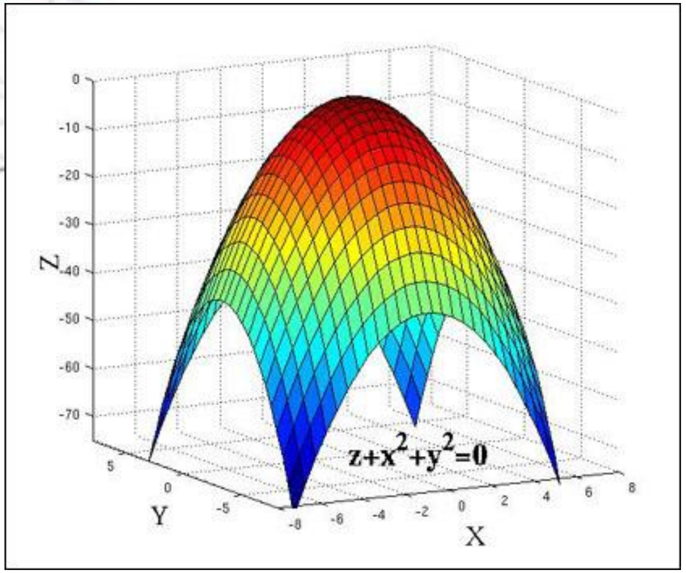
Time to finish Medical Diagnosis in Seconds

$$f(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{\Gamma(k)}$$

Gradient Ascent



Especially good if function is convex

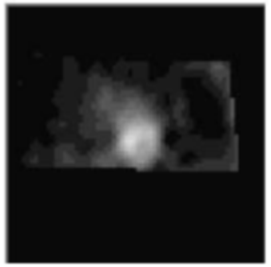


Walk uphill and you will find a local maxima
(if your step size is small enough)

Gradient Ascent

```
Initialize: $\theta_j$ = random for all $0 \le j \le m$

Repeat many times:

    Calculate all gradient[j]'s based on data

    $\theta_j$ += $\eta$ * gradient[j] for all $0 \le j \le m$
```

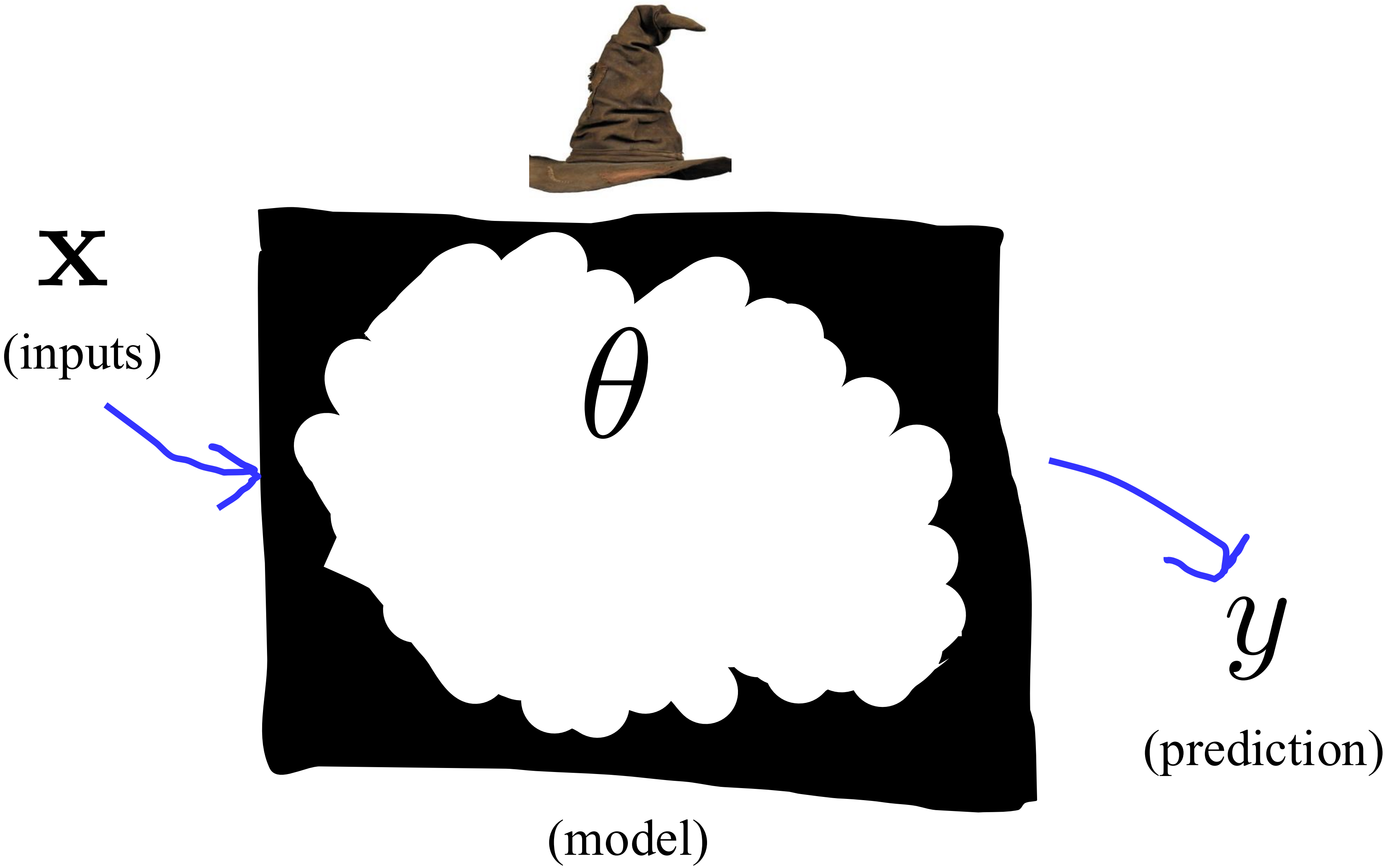
Healthy Heart Classifier

	ROI 1	ROI 2		ROI *m*	Output	
			...			
Heart 1	0	1		1	0	
Heart 2	1	1		1	0	$(\mathbf{x}^{(2)}, y^{(2)})$
			⋮		⋮	
Heart *n*	0	0		0	1	

Classification is Building a Harry Potter Hat

$$\mathbf{x} = [0, 1, \ldots, 1]$$

Machine Learning for Classification



$\mathbf{x}$

(inputs)

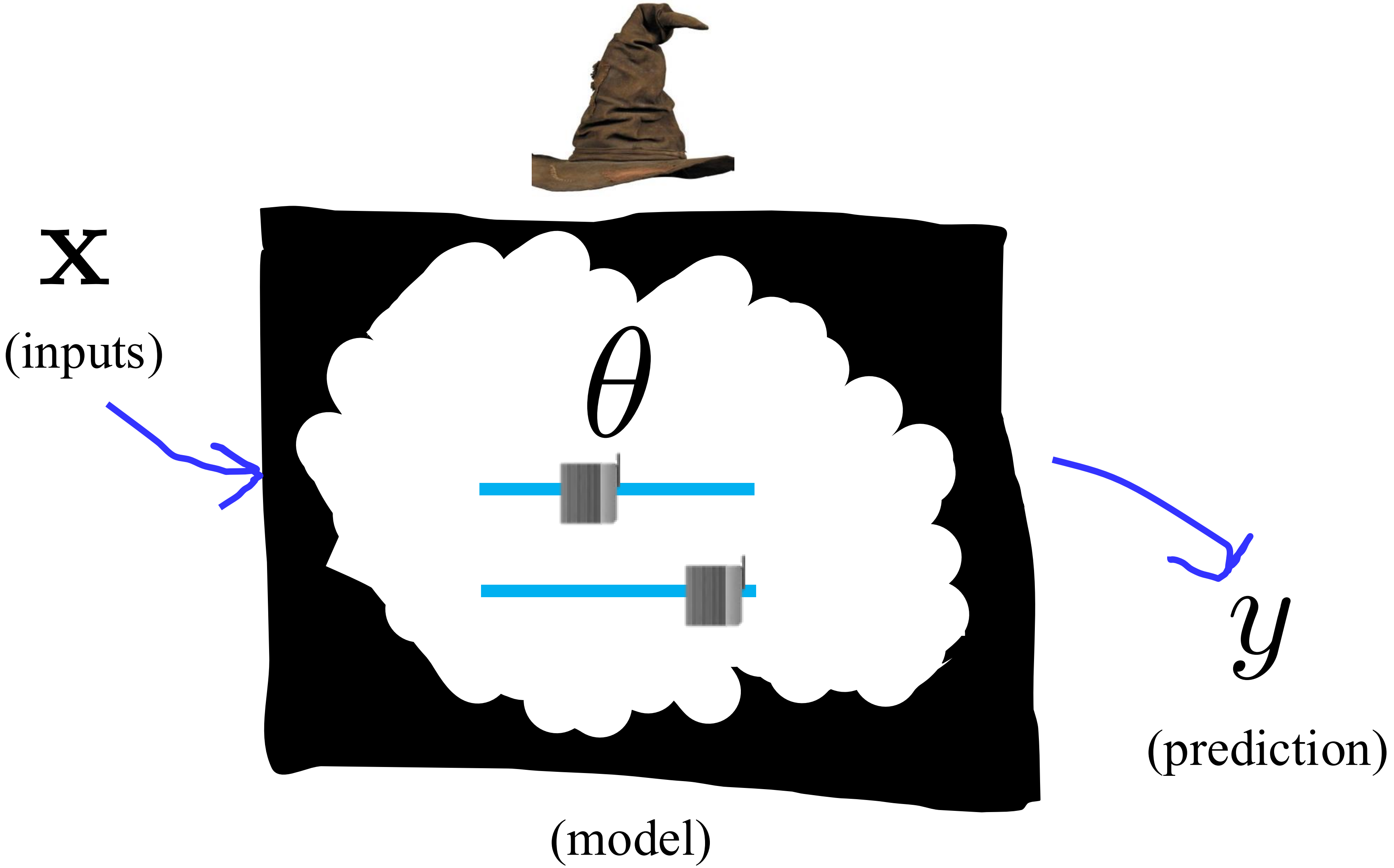
θ

(model)

y

(prediction)

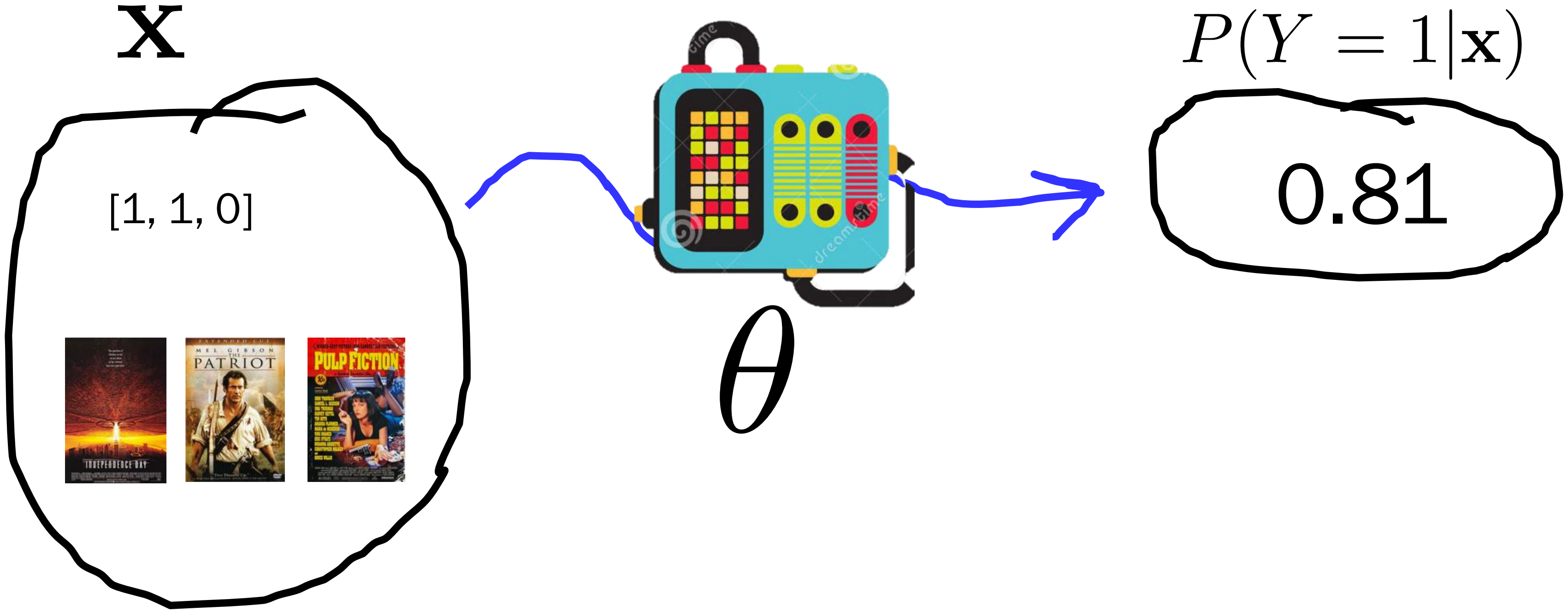
Machine Learning for Classification



Logistic Regression Assumption

Could we compute $P(Y = 1|\mathbf{X} = \mathbf{x})$ via a machine?

Welcome our friend: logistic regression!



$\mathbf{x}$

[1, 1, 0]

θ

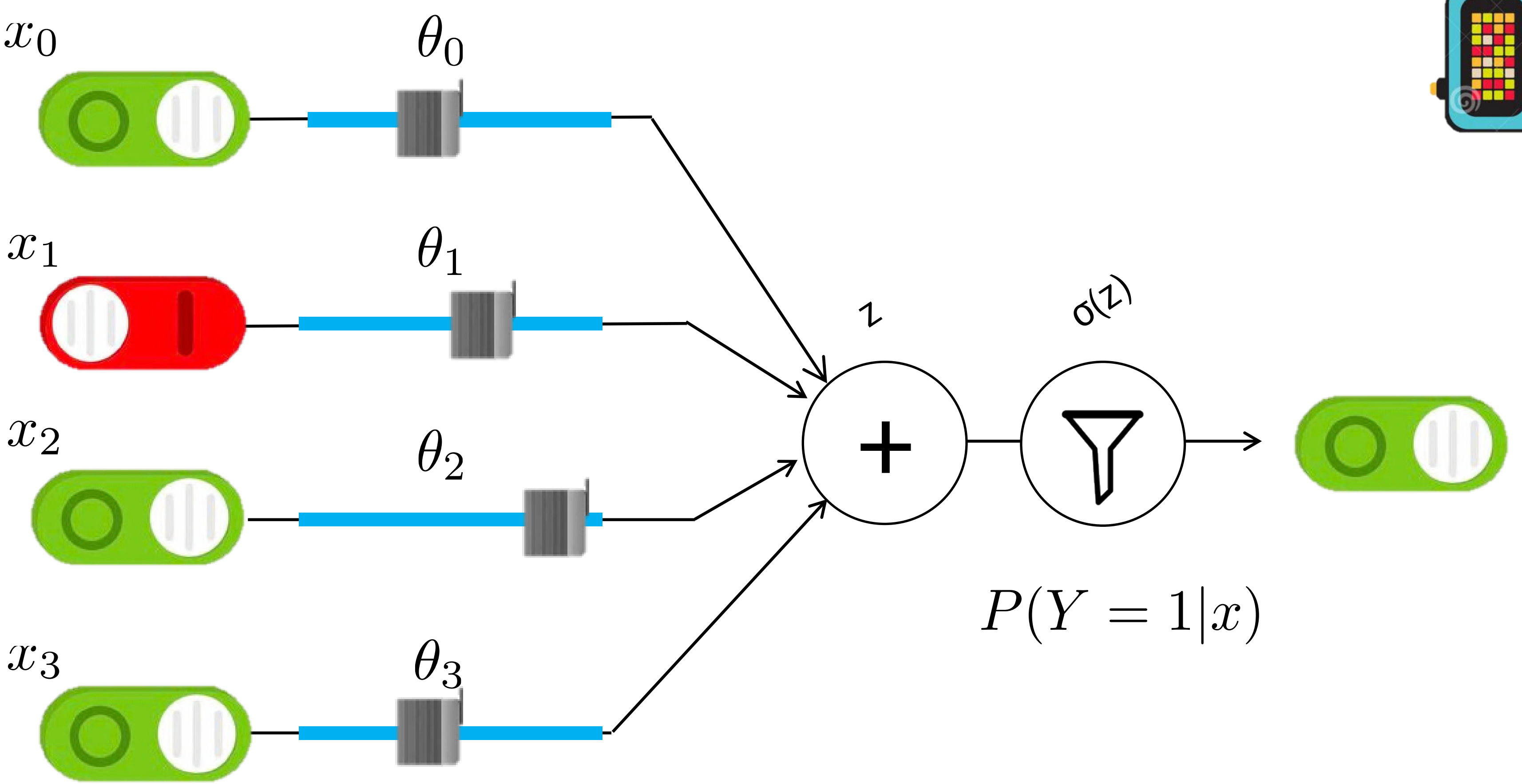
$P(Y = 1|\mathbf{x})$

0.81

Logistic Regression Assumption

$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Logistic Regression



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Stanford University

Math for Logistic Regression

1 Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$
$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

Often call this $\hat{y}$

2 Calculate the log likelihood for all data

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

3 Get derivative of log likelihood with respect to thetas

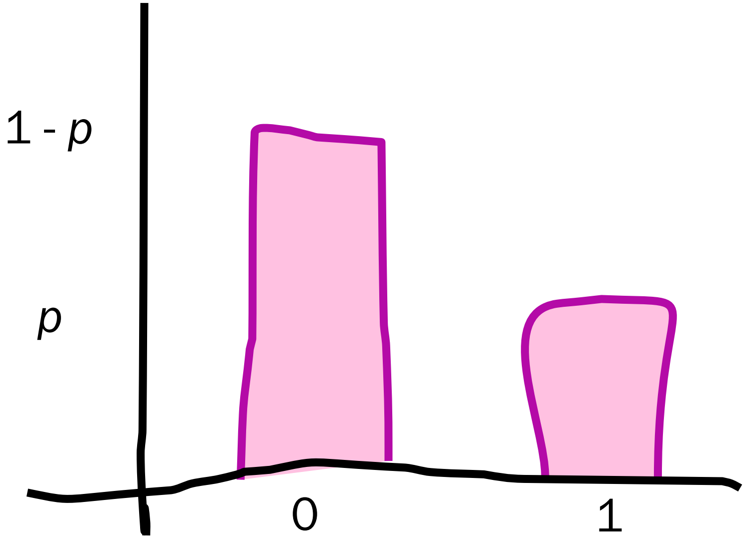
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Recall: PMF of Bernoulli

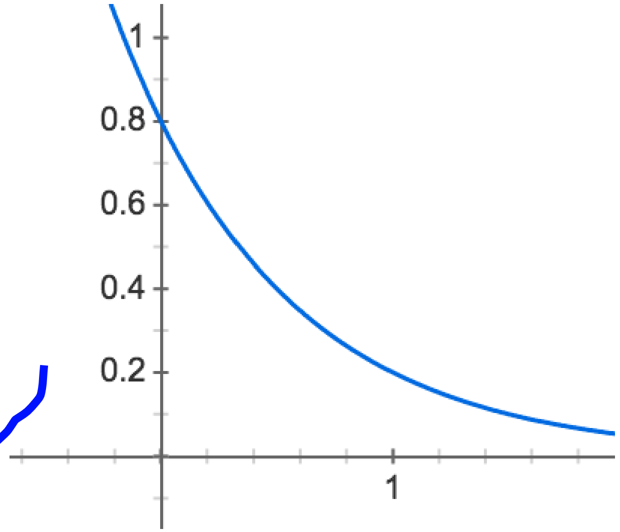
$$Y \sim \text{Bern}(p)$$

Probability mass function: $P(Y = y)$

PMF of Bernoulli



PMF of Bernoulli (p = 0.2)



$$P(Y = y) = p^y(1-p)^{1-y} \qquad P(Y = y) = 0.2^y(0.8)^{1-y}$$

Recall:

$Y \sim \text{Bern}(p)$

$P(Y = y) = p^y(1-p)^{1-y}$

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T\mathbf{x})$$

$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T\mathbf{x})$$

Implies

$$P(Y = y|X = \mathbf{x}) = \sigma(\theta^T\mathbf{x})^y \cdot \left[1 - \sigma(\theta^T\mathbf{x})\right]^{(1-y)}$$

Ask Juliette:
Why not

$P(Y = y^{(i)}, X = x^{(i)})$

For IID data

$$L(\theta) = \prod_{i=1}^{n} P(Y = y^{(i)}|X = \mathbf{x}^{(i)})$$

$$= \prod_{i=1}^{n} \sigma(\theta^T\mathbf{x}^{(i)})^{y^{(i)}} \cdot \left[1 - \sigma(\theta^T\mathbf{x}^{(i)})\right]^{(1-y^{(i)})}$$

Take the log

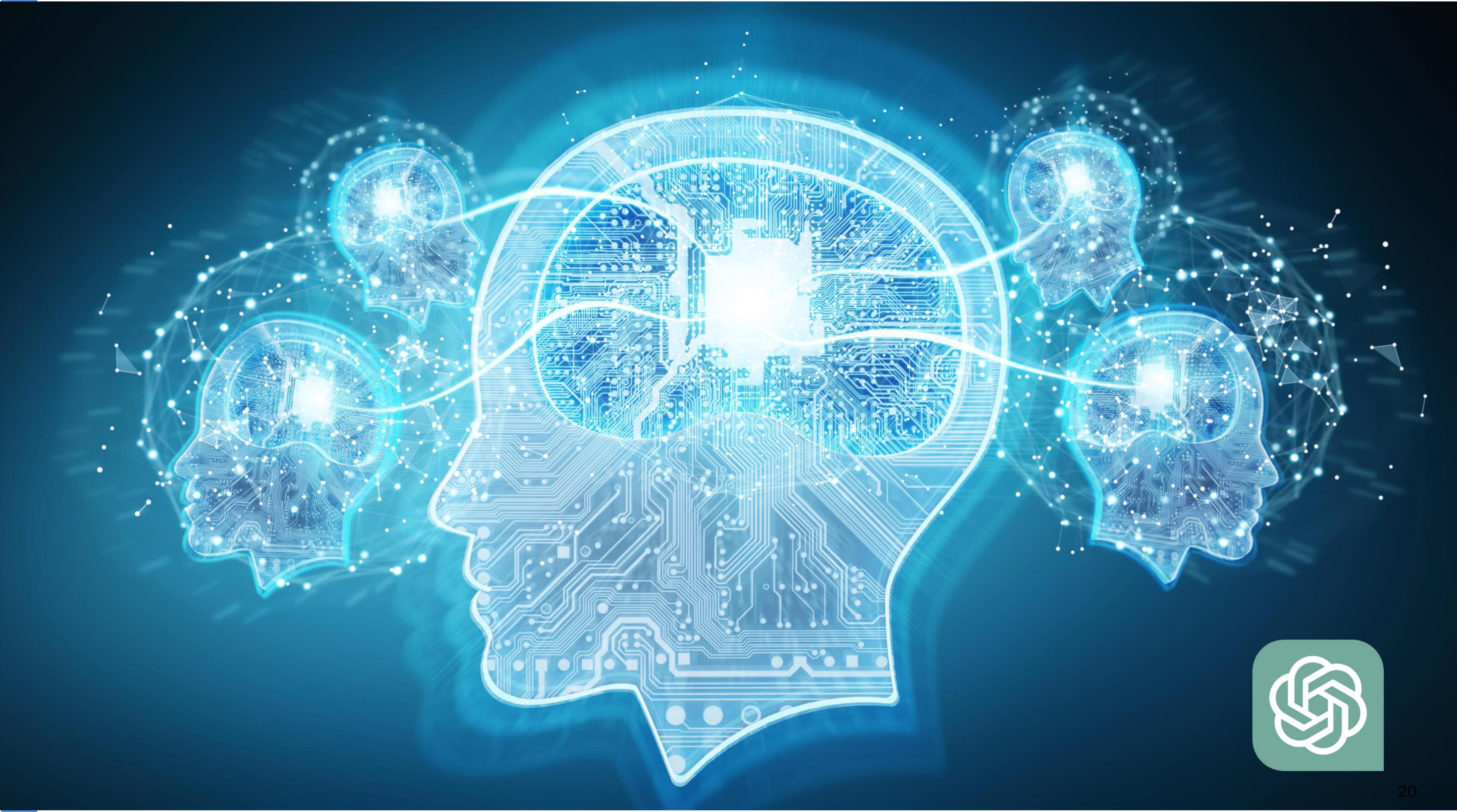
$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma(\theta^T\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T\mathbf{x}^{(i)})]$$

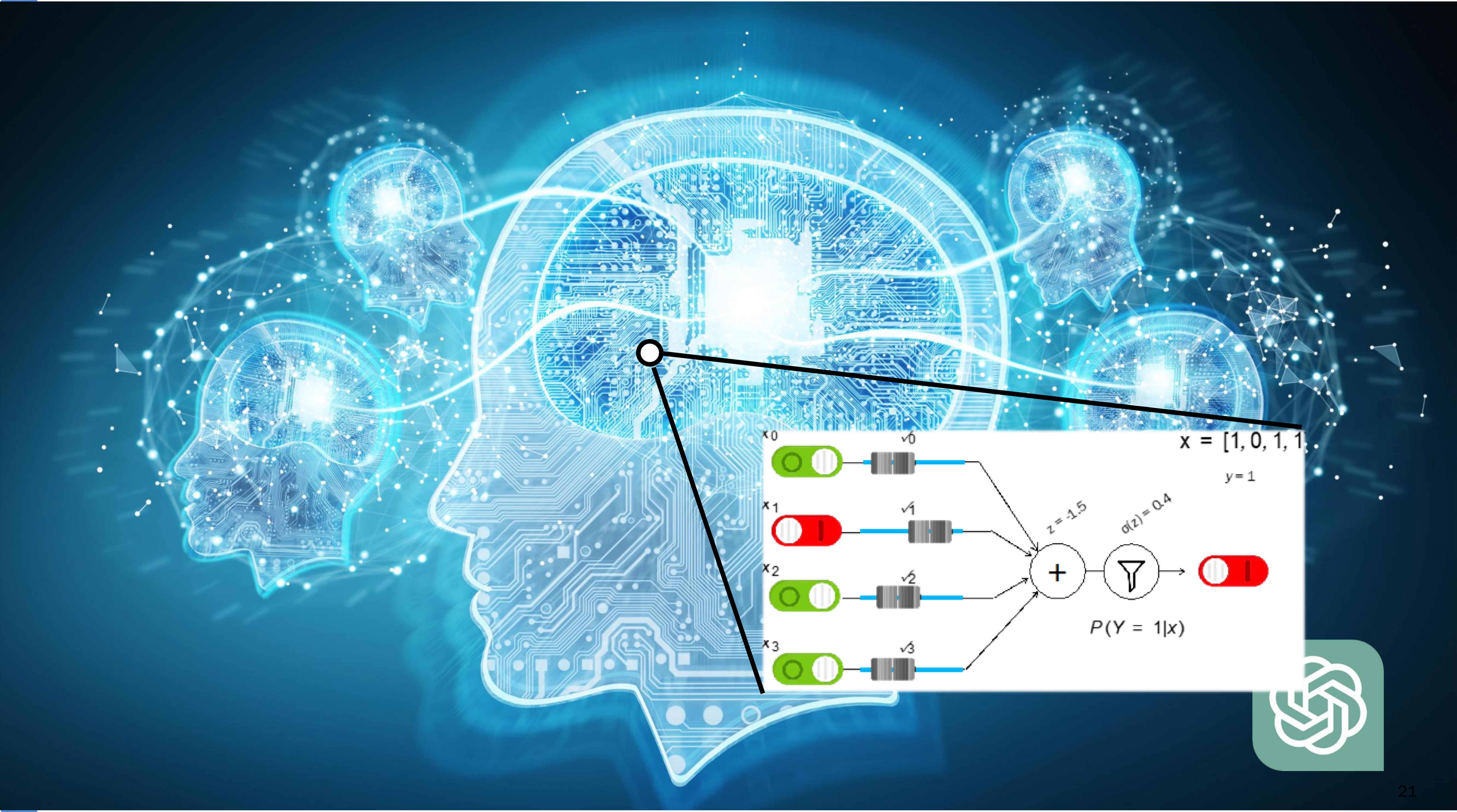
Logistic Regression Training

```
Initialize: $\theta_j$ = random for all $0 \le j \le m$

Repeat many times:
    Calculate all gradient[j]'s based on data
    $\theta_j$ += $\eta$ * gradient[j] for all $0 \le j \le m$
```

Logistic Regression Training

```
Initialize: $\theta_j$ = 0 for all 0 ≤ $j$ ≤ $m$

Repeat many times:
    gradient[j] = 0 for all 0 ≤ $j$ ≤ $m$
    For each training example (x, y):
        For each parameter j:
            gradient[j] $\mathrel{+}= x_j\left(y - \frac{1}{1+e^{-\theta^{\mathrm{T}}\mathbf{x}}}\right)$
    $\theta_j$ += η * gradient[j] for all 0 ≤ $j$ ≤ $m$
```




x 0
√0
x 1
√1
x 2
√2
x 3
√3
x = [1, 0, 1, 1
y = 1
z = -1.5
σ(z) = 0.4
+
P(Y = 1|x)

End Review

Three Guiding Questions

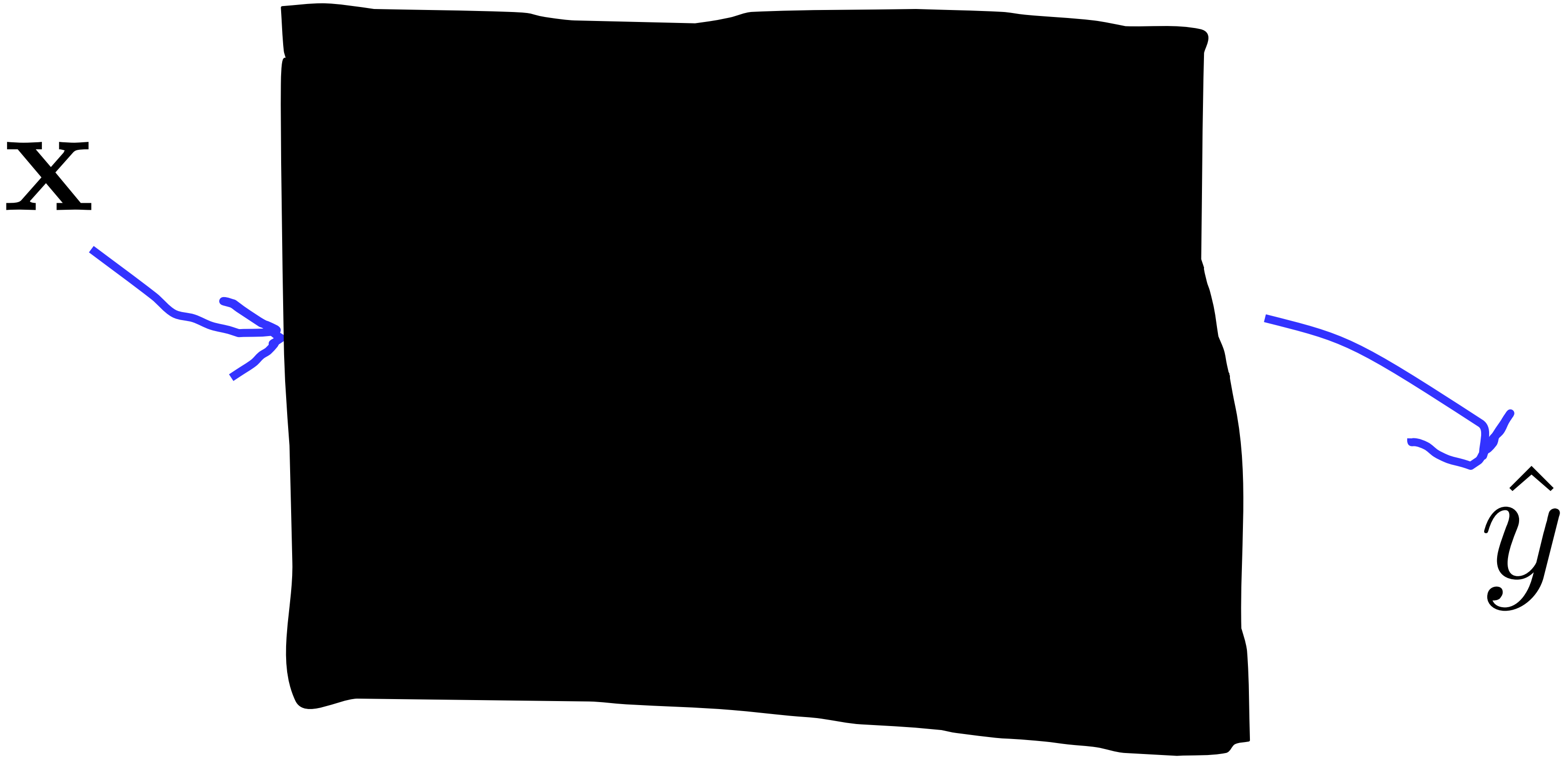
1. What are other models for classification?

2. For a dataset, how do you tell which model is best?

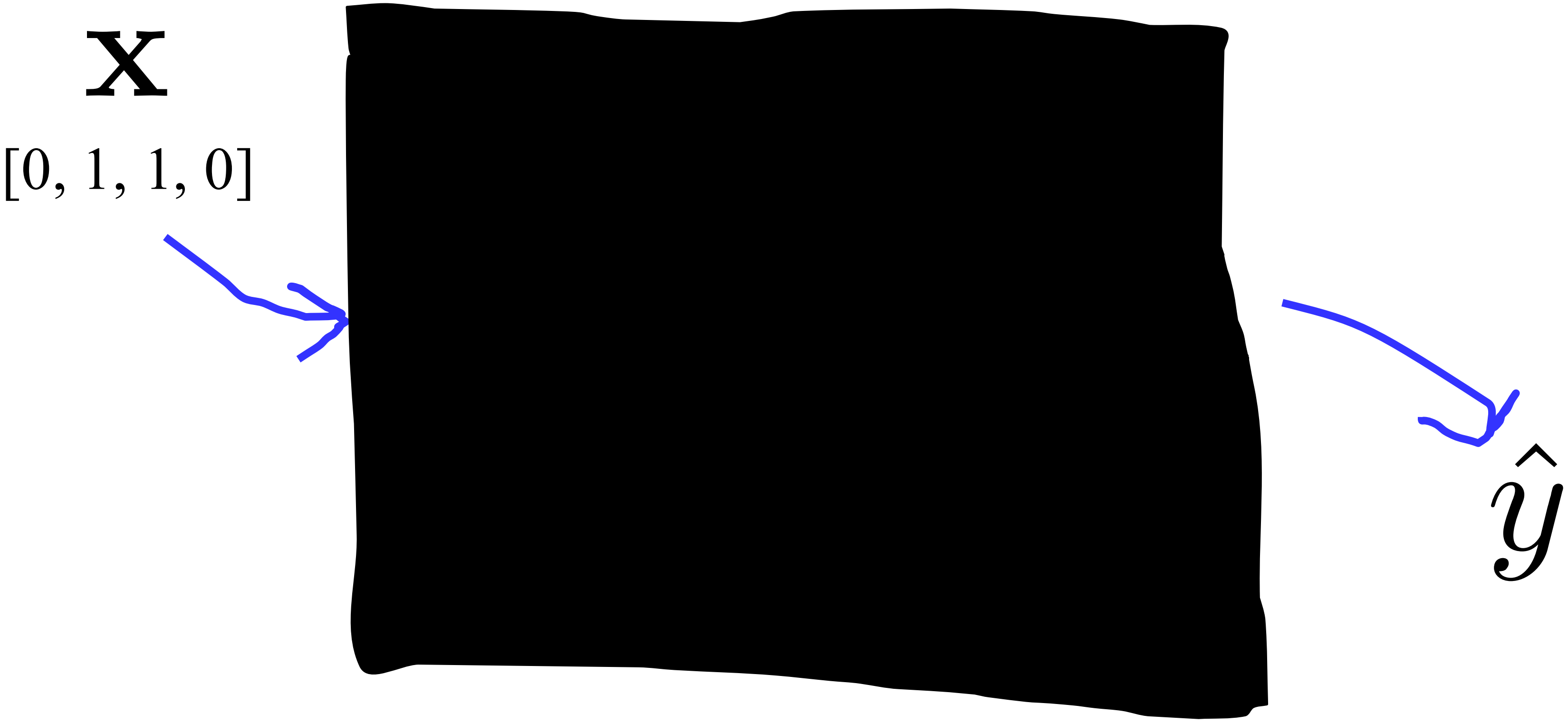
3. What are other validation metrics I might care about?

Fake Algorithm:
Brute Bayes Classifier

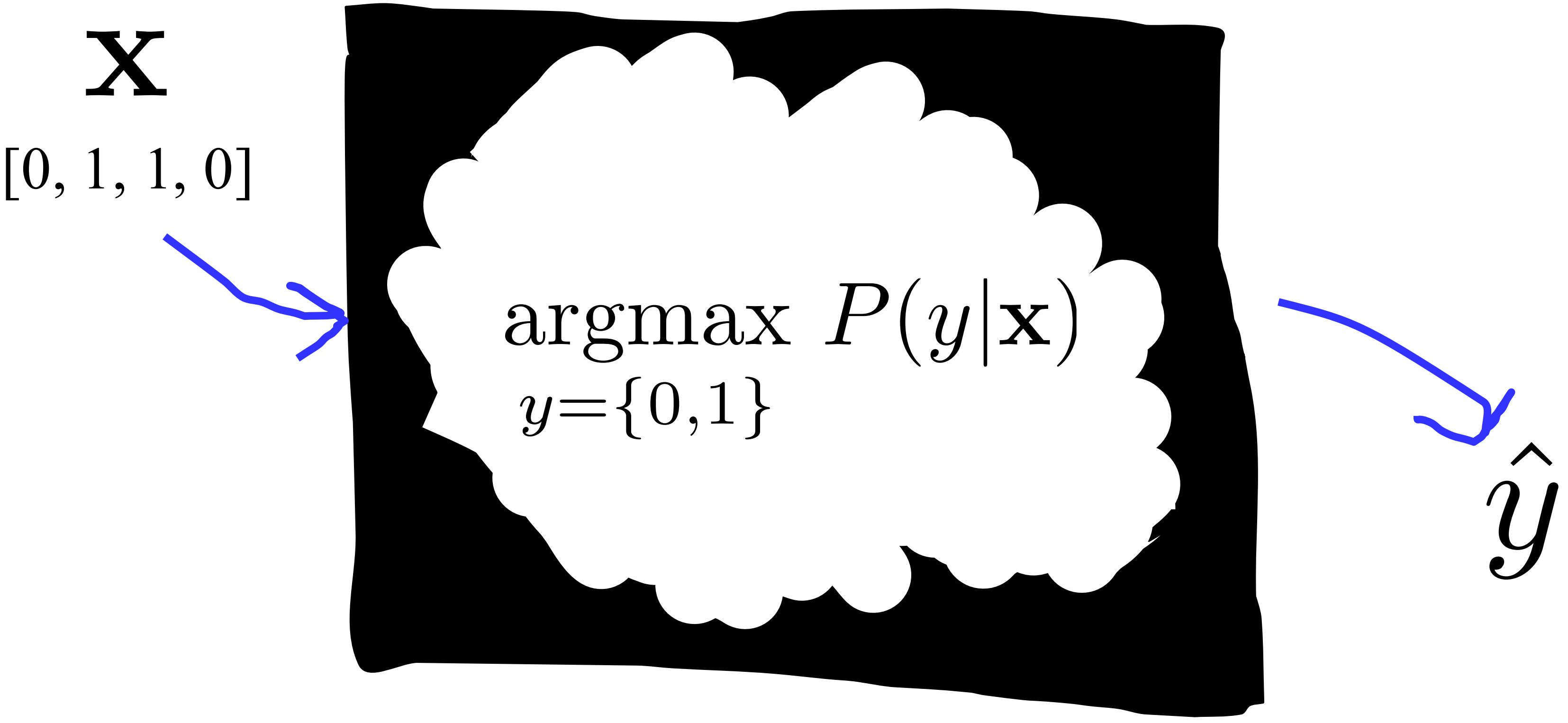
Brute Force Bayes



Brute Force Bayes



Brute Force Bayes



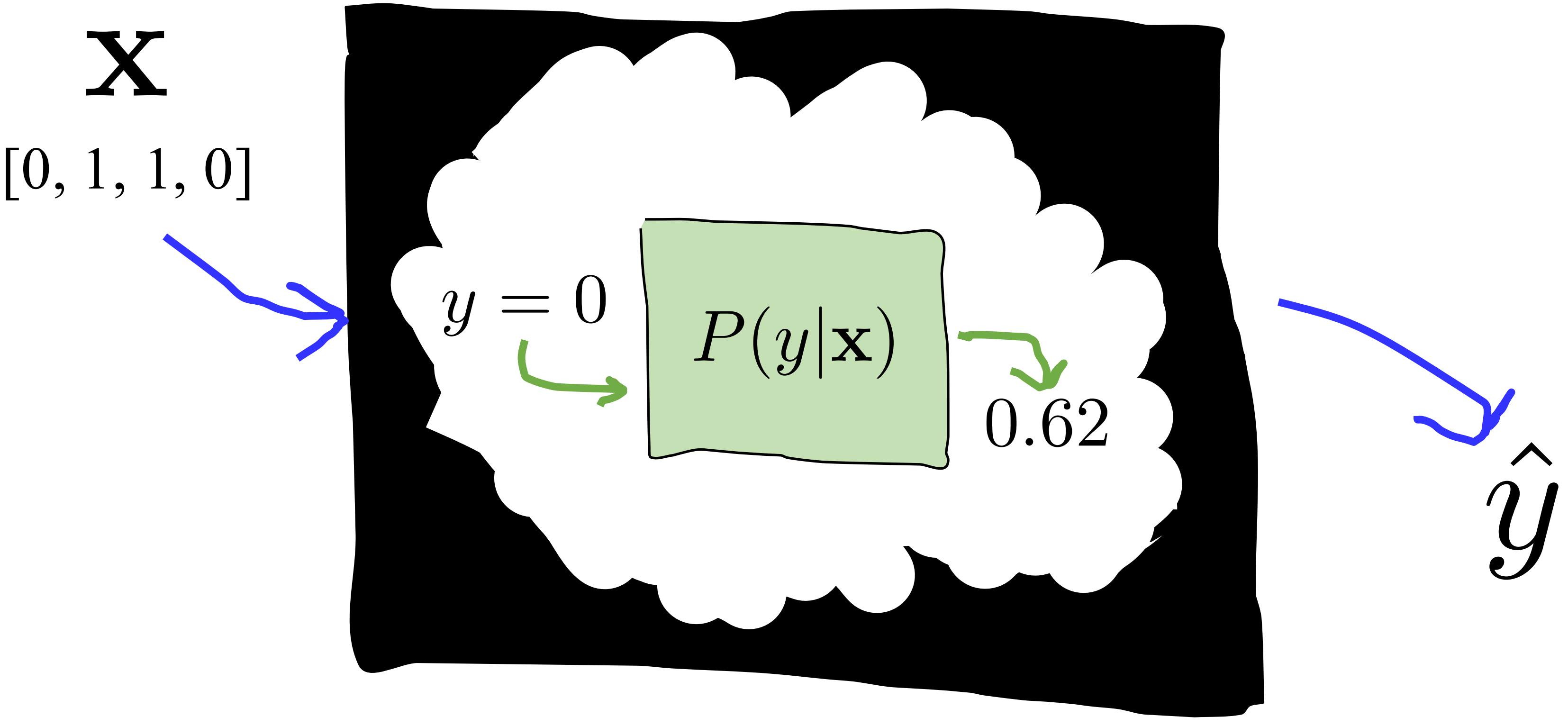
$\mathbf{x}$

[0, 1, 1, 0]

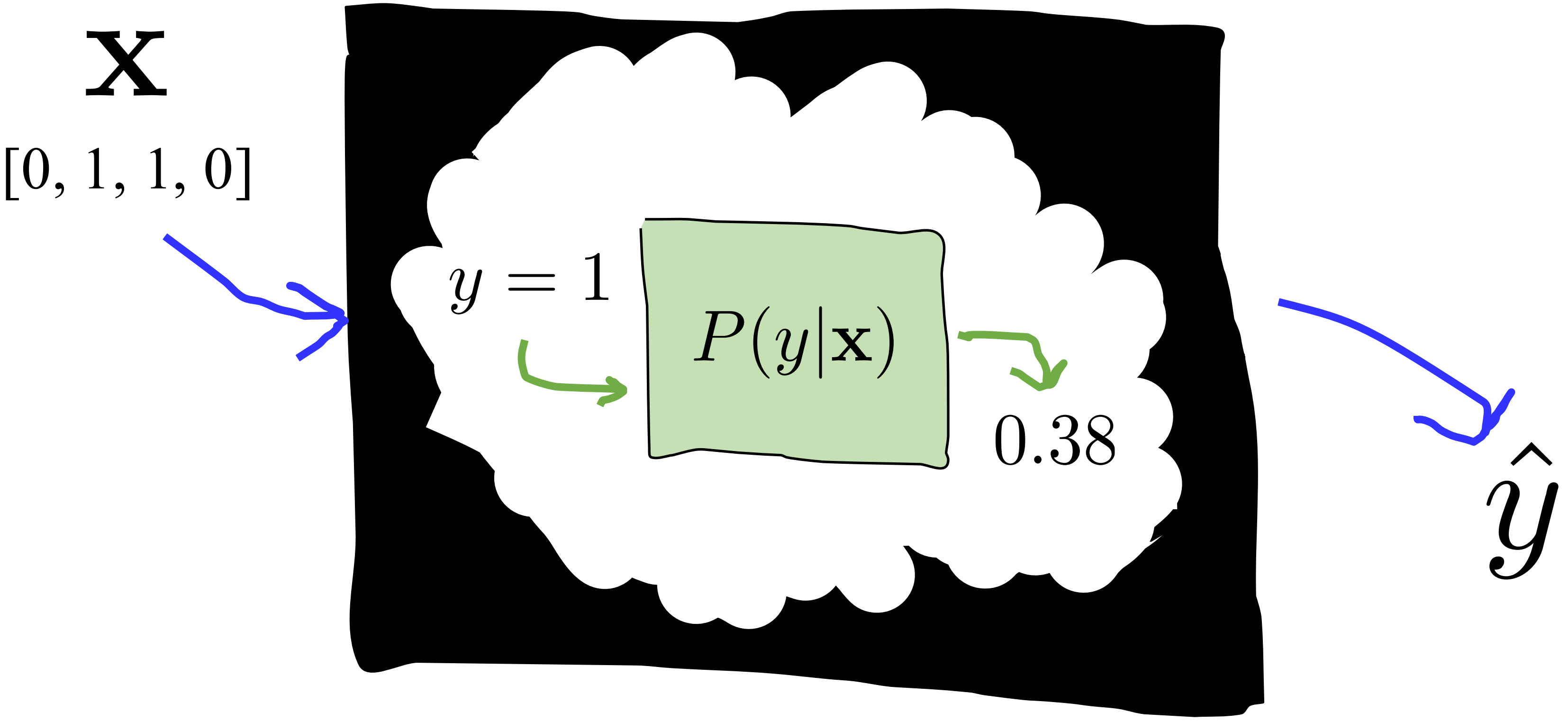
$$\underset{y=\{0,1\}}{\text{argmax}}\ P(y|\mathbf{x})$$

$\hat{y}$

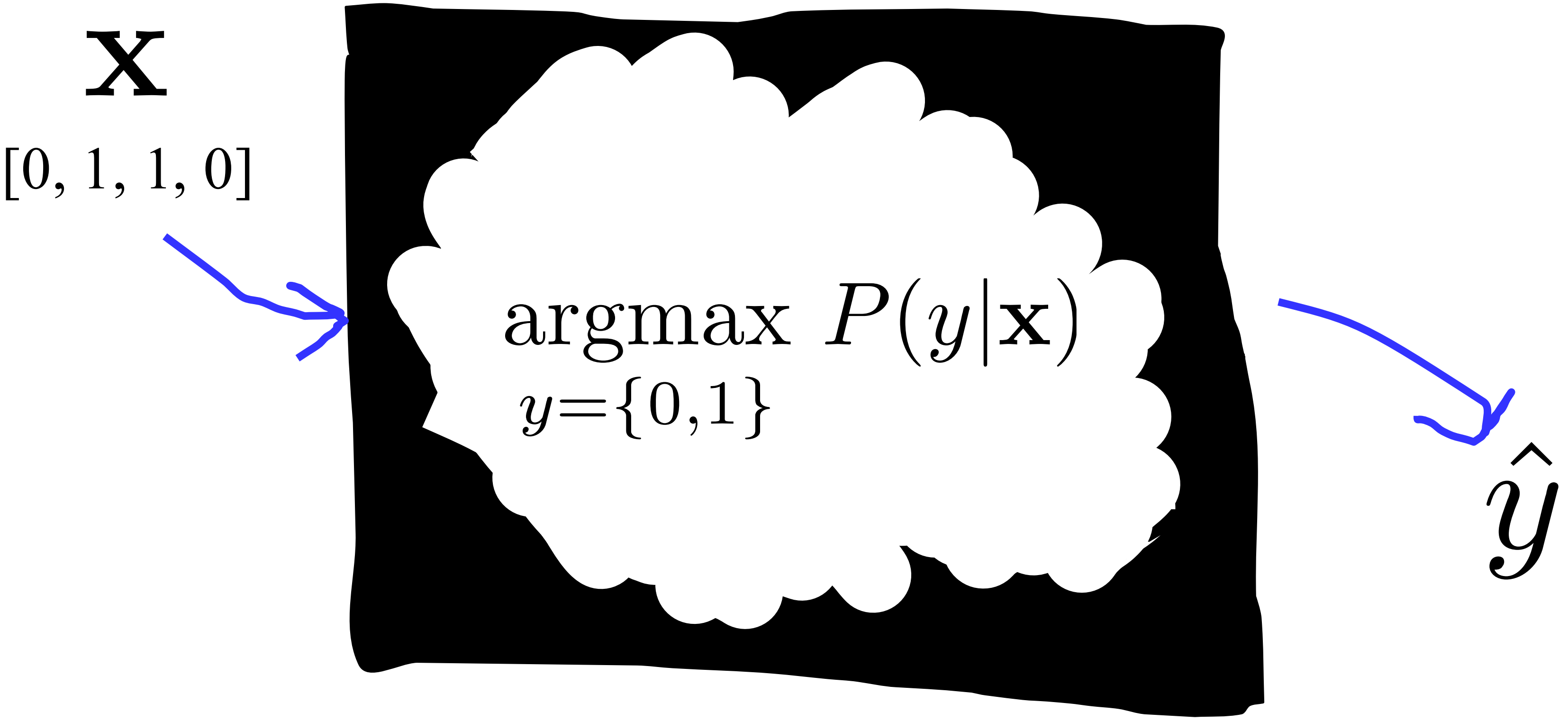
Brute Force Bayes



Brute Force Bayes



Brute Force Bayes



$\mathbf{x}$

[0, 1, 1, 0]

$$\underset{y=\{0,1\}}{\text{argmax}}\ P(y|\mathbf{x})$$

$\hat{y}$

Brute Force Bayes

Prediction: will they like L.I.B.?

Whether or not they liked Independence day

$$\hat{y} = \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(y|\mathbf{x})$$

If y = 1, they like L.I.B.?

Simply chose the class label that is the most likely given the data

This is for one user

Brute Force Bayes

$$\hat{y} = \underset{y=\{0,1\}}{\text{argmax}} \; P(y|\mathbf{x})$$

Simply chose the class label that is the most likely given the data

This is for one user

Brute Force Bayes

$$\hat{y} = \underset{y=\{0,1\}}{\operatorname{argmax}}\ P(y|\mathbf{x})$$

$$= \underset{y=\{0,1\}}{\operatorname{argmax}}\ \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

$$= \underset{y=\{0,1\}}{\operatorname{argmax}}\ \underline{P(\mathbf{x}|y)}\underline{P(y)}$$

Simply chose the class label that is the most likely given the data

This is for one user

* Note how similar this is to Hamilton example ☺

What are the Parameters?

Brute Force Bayes

$$\hat{y} = \underset{y=\{0,1\}}{\operatorname{argmax}} \ \underline{P(\mathbf{x}|y)}\,\underline{P(y)}$$

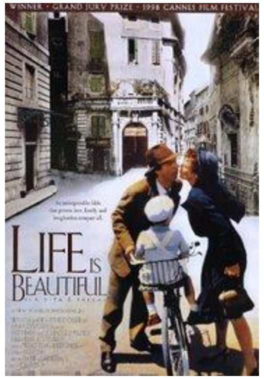
Conditional probability table

Y = 0

X_1 = 0	θ_0
X_1 = 1	θ_1

Y = 1

X_1 = 0	θ_2
X_1 = 1	θ_3



Y = 0	θ_4
Y = 1	θ_5

Learn these during training

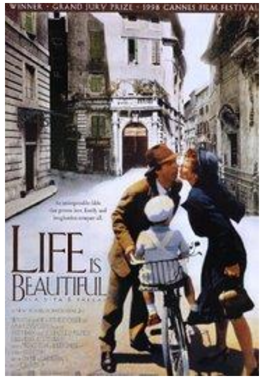
Brute Force Bayes

$$\hat{y} = \underset{y=\{0,1\}}{\operatorname{argmax}}\ \underline{P(\mathbf{x}|y)}\,\underline{P(y)}$$

The question of whether or not we are alone in the universe has been answered.
INDEPENDENCE DAY

Conditional probability table

X_1 \ Y	0	1
0	θ_0	θ_2
1	θ_1	θ_3

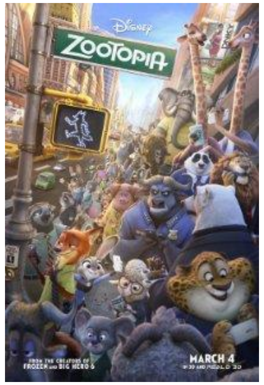


Y = 0	θ_4
Y = 1	θ_5

Learn these during training

Seems pretty good!

Brute Force Bayes m = 2

	x_1	x_2	y
			
User 1	1	0	1
User 2	1	0	0
			⋮
User *n*	0	1	1

Brute Force Bayes m = 2

Simply chose the class label that is the most likely given the data

$$\begin{aligned}\hat{y} &= \underset{y=\{0,1\}}{\operatorname{argmax}}\ P(y|\mathbf{x}) \\ &= \underset{y=\{0,1\}}{\operatorname{argmax}}\ \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\ &= \underset{y=\{0,1\}}{\operatorname{argmax}}\ \underline{P(\mathbf{x}|y)}\,\underline{P(y)}\end{aligned}$$

$$P(x_1, x_2|y)$$

Brute Force Bayes

$$\hat{y} = \underset{y=\{0,1\}}{\operatorname{argmax}} \boxed{P(\mathbf{x}|y)} P(y)$$

Y = 0

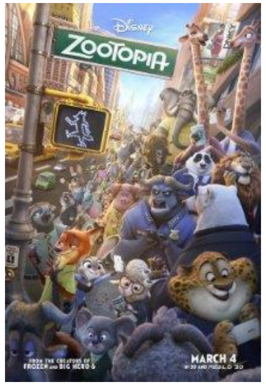
X_2 \ X_1	0	1
0	θ_0	θ_1
1	θ_2	θ_3

Y = 1

X_2 \ X_1	0	1
0	θ_4	θ_5
1	θ_6	θ_7

x_1

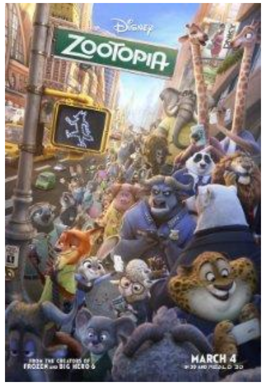
x_2



y

Fine

Brute Force Bayes m = 3

	x_1	x_2	x_3	y
				
User 1	1	0	1	1
User 2	1	0	1	0
				⋮
User *n*	0	1	1	1

Brute Force Bayes m = 3

Simply chose the class label that is the most likely given the data

$$\begin{aligned}
\hat{y} &= \underset{y=\{0,1\}}{\operatorname{argmax}}\ P(y|\mathbf{x}) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}}\ \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}}\ P(\mathbf{x}|y)P(y)
\end{aligned}$$

$$P(x_1, x_2, x_3|y)$$

Brute Force Bayes

$$\hat{y} = \underset{y=\{0,1\}}{\operatorname{argmax}} \boxed{P(\mathbf{x}|y)} P(y)$$

Y = 0

X_3 = 0

X_2 \ X_1	0	1
0	θ_0	θ_1
1	θ_2	θ_3

X_3 = 1

X_2 \ X_1	0	1
0	θ_4	θ_5
1	θ_6	θ_7

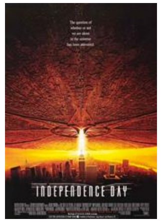
Y = 1

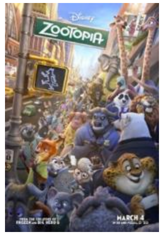
X_3 = 0

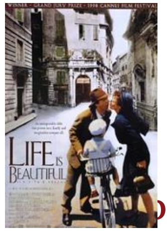
X_2 \ X_1	0	1
0	θ_8	θ_9
1	θ_{10}	θ_{11}

X_3 = 1

X_2 \ X_1	0	1
0	θ_{12}	θ_{13}
1	θ_{14}	θ_{15}







And if $m=100$?

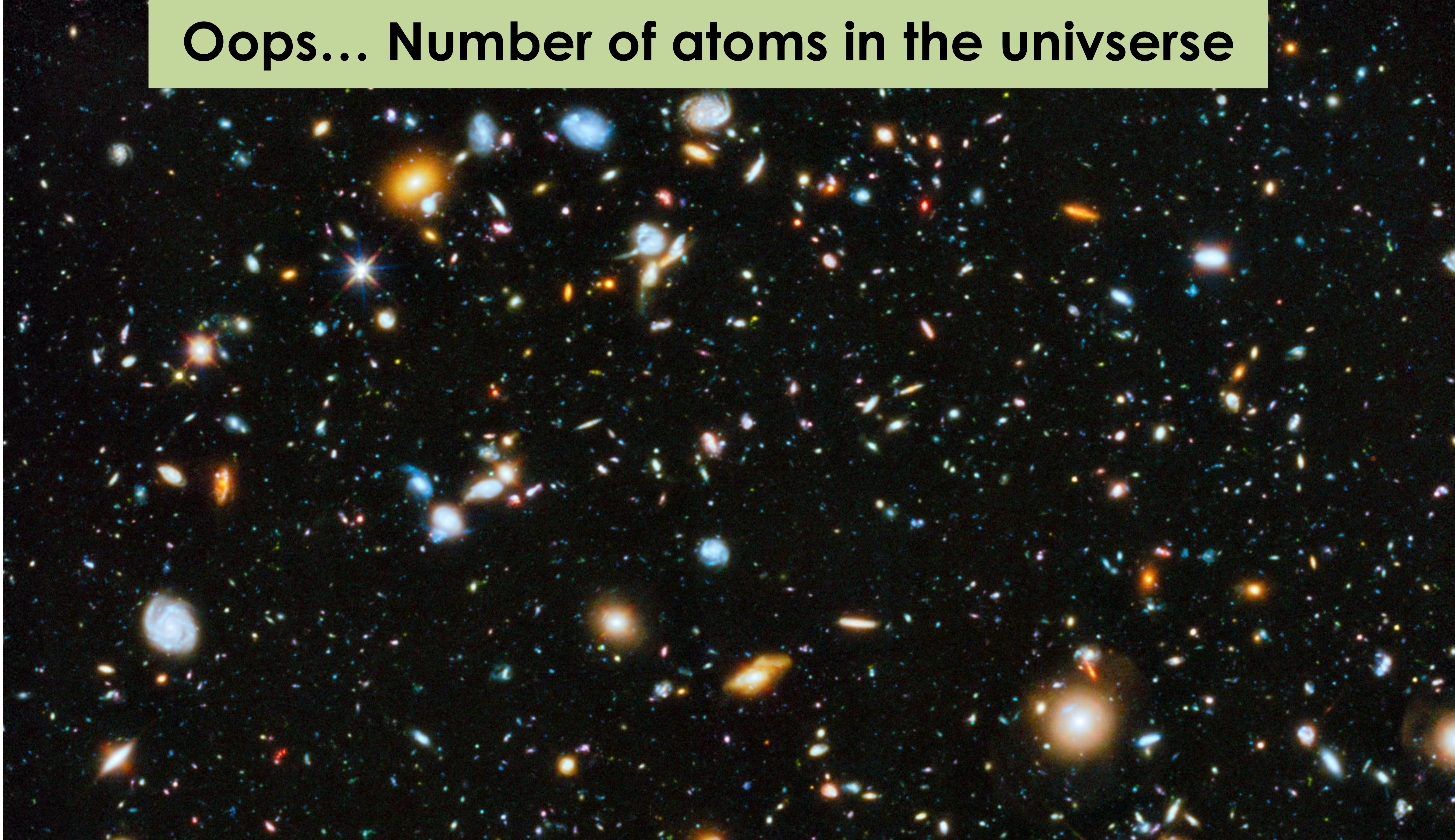
Brute Force Bayes m = 100

Simply chose the class label that is the most likely given the data

$$\begin{aligned}\hat{y} &= \underset{y=\{0,1\}}{\text{argmax}}\ P(y|\mathbf{x}) \\ &= \underset{y=\{0,1\}}{\text{argmax}}\ \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\ &= \underset{y=\{0,1\}}{\text{argmax}}\ P(\mathbf{x}|y)P(y)\end{aligned}$$

$$P(x_1, x_2, x_3, \ldots, x_{100}|y)$$

Oops… Number of atoms in the univserse



What is the big O for # parameters?
m = # features.

Big O of Brute Force Joint

What is the big O for # parameters?
m = # features.

$$O(2^m)$$

Assuming each feature is binary…

Not going to cut it!

What is the problem here?

$$\hat{y} = \underset{y=\{0,1\}}{\operatorname{argmax}}\ P(y|\mathbf{x})$$

$$= \underset{y=\{0,1\}}{\operatorname{argmax}}\ \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

$$= \underset{y=\{0,1\}}{\operatorname{argmax}}\ \boxed{P(\mathbf{x}|y)}P(y)$$

$$P(\mathbf{x}|y) = P(x_1, x_2, \ldots, x_m|y)$$

Naïve Bayes Assumption

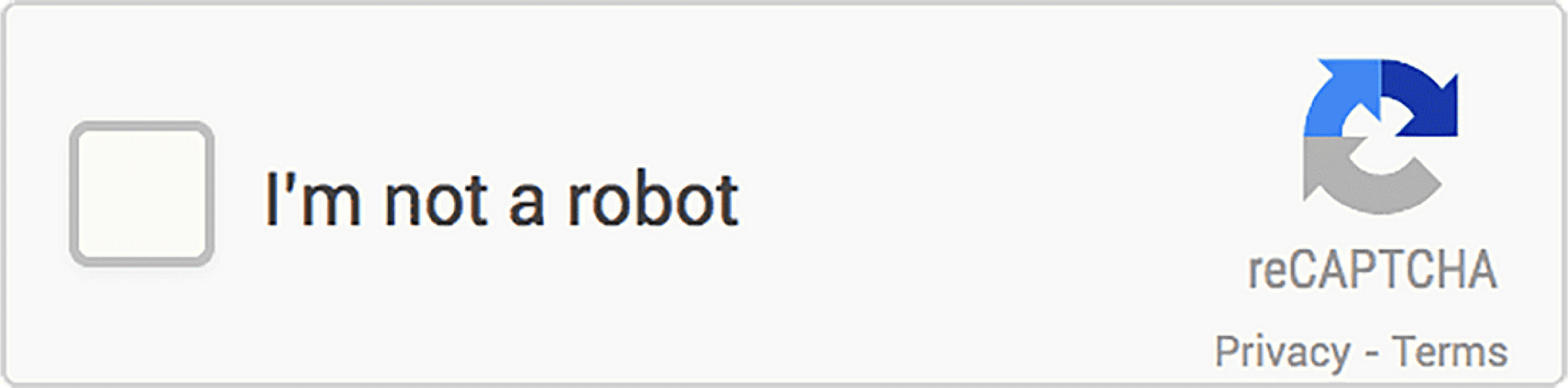
$$\begin{aligned}\hat{y} &= \underset{y=\{0,1\}}{\operatorname{argmax}}\ P(y|\mathbf{x}) \\ &= \underset{y=\{0,1\}}{\operatorname{argmax}}\ \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\ &= \underset{y=\{0,1\}}{\operatorname{argmax}}\ P(\mathbf{x}|y)P(y)\end{aligned}$$

$$\begin{aligned}P(\mathbf{x}|y) &= P(x_1, x_2, \ldots, x_m|y) \\ &= \prod_i P(x_i|y)\end{aligned}$$

The Naïve Bayes assumption

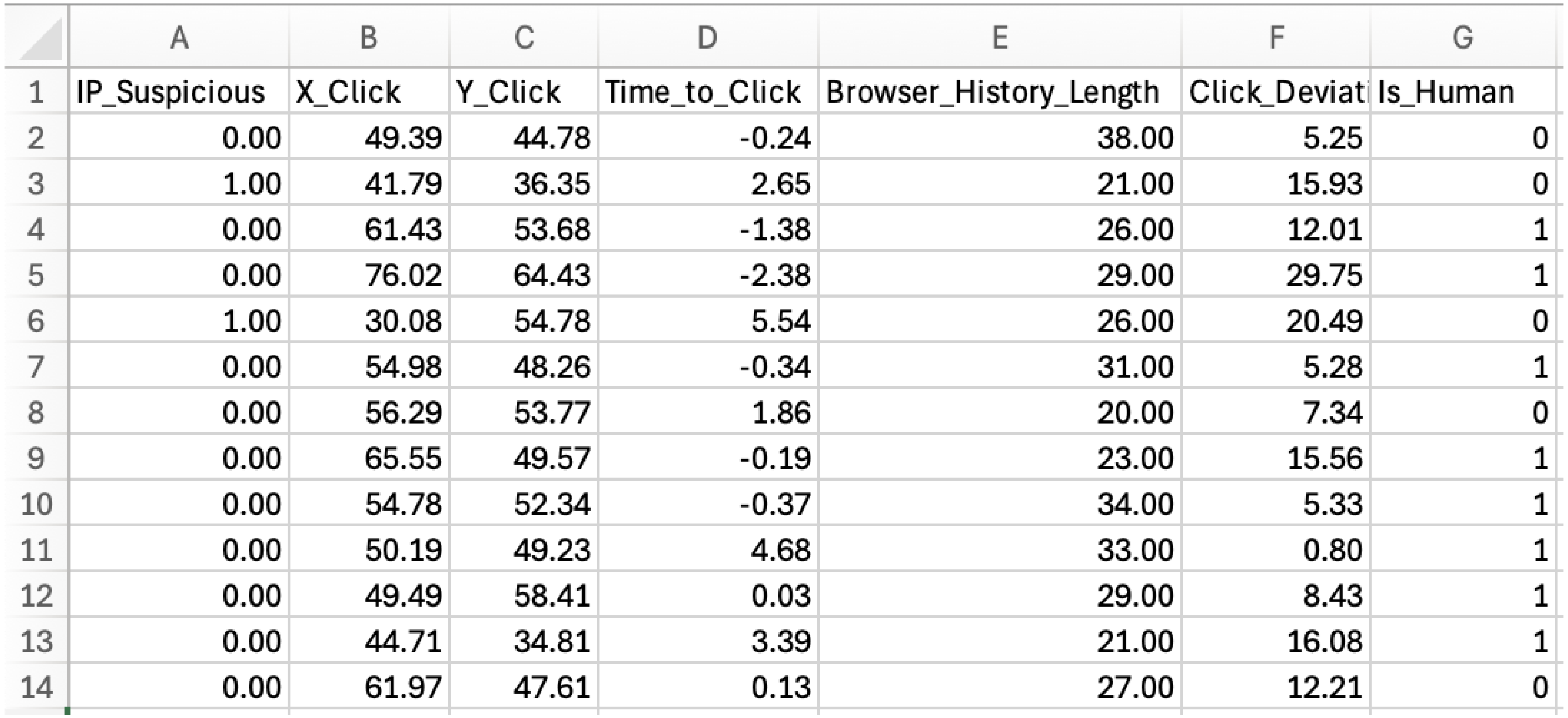
How Can We Compare?

New Challenge: Captcha



□ I'm not a robot

New Challenge: Captcha



	A	B	C	D	E	F	G
1	IP_Suspicious	X_Click	Y_Click	Time_to_Click	Browser_History_Length	Click_Deviati	Is_Human
2	0.00	49.39	44.78	-0.24	38.00	5.25	0
3	1.00	41.79	36.35	2.65	21.00	15.93	0
4	0.00	61.43	53.68	-1.38	26.00	12.01	1
5	0.00	76.02	64.43	-2.38	29.00	29.75	1
6	1.00	30.08	54.78	5.54	26.00	20.49	0
7	0.00	54.98	48.26	-0.34	31.00	5.28	1
8	0.00	56.29	53.77	1.86	20.00	7.34	0
9	0.00	65.55	49.57	-0.19	23.00	15.56	1
10	0.00	54.78	52.34	-0.37	34.00	5.33	1
11	0.00	50.19	49.23	4.68	33.00	0.80	1
12	0.00	49.49	58.41	0.03	29.00	8.43	1
13	0.00	44.71	34.81	3.39	21.00	16.08	1
14	0.00	61.97	47.61	0.13	27.00	12.21	0

This makes it a classification task

Comparing Algorithms: Buggy Metric

Algorithm	Accuracy on the Data
Logistic Regression	0.7
Naïve Bayes	0.6
Nearest Neighbor	

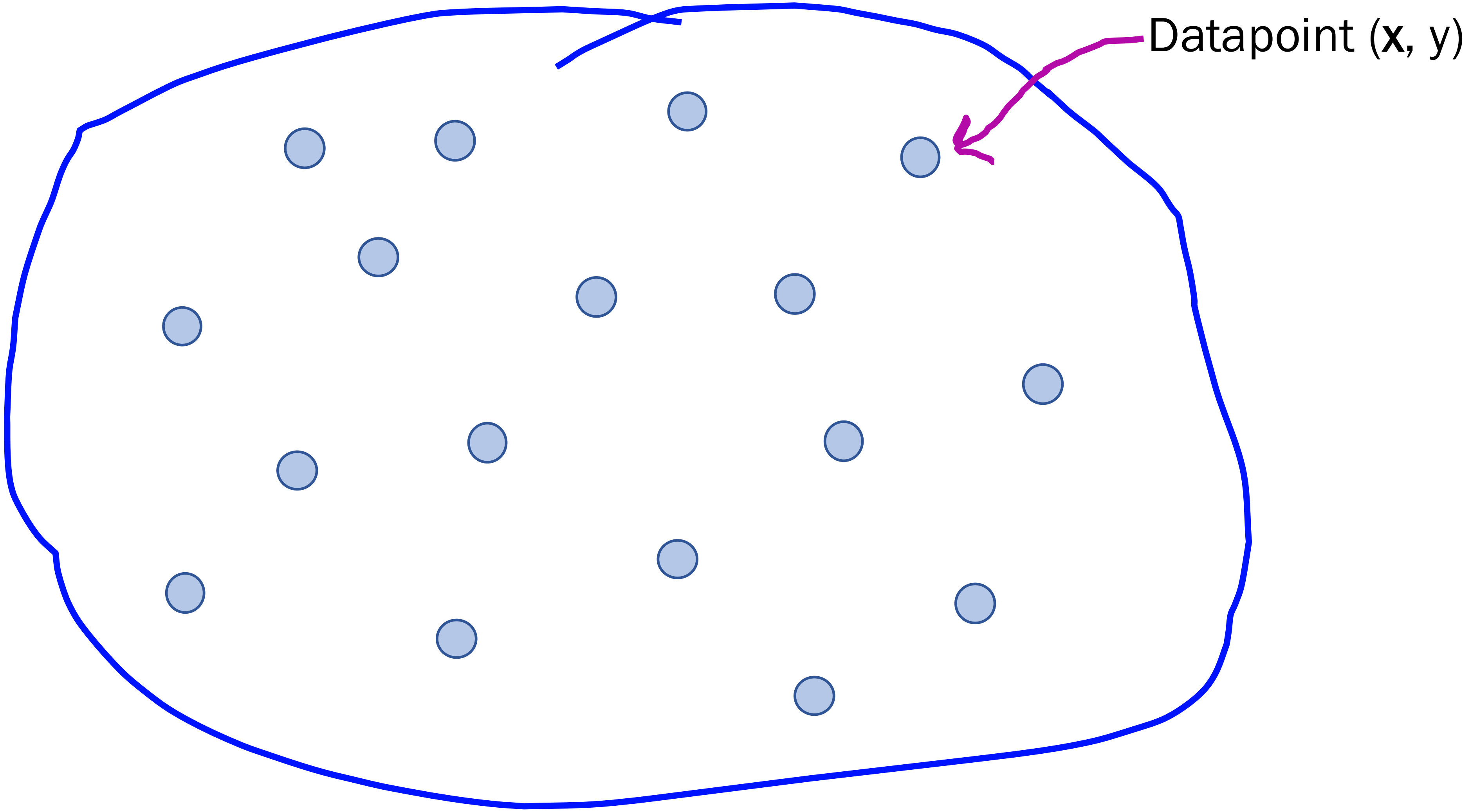
Comparing Algorithms: Buggy Metric

Algorithm	Accuracy on the Data
Logistic Regression	0.7
Naïve Bayes	0.6
Nearest Neighbor	**1.0**

Comparing Algorithms: Better Metric

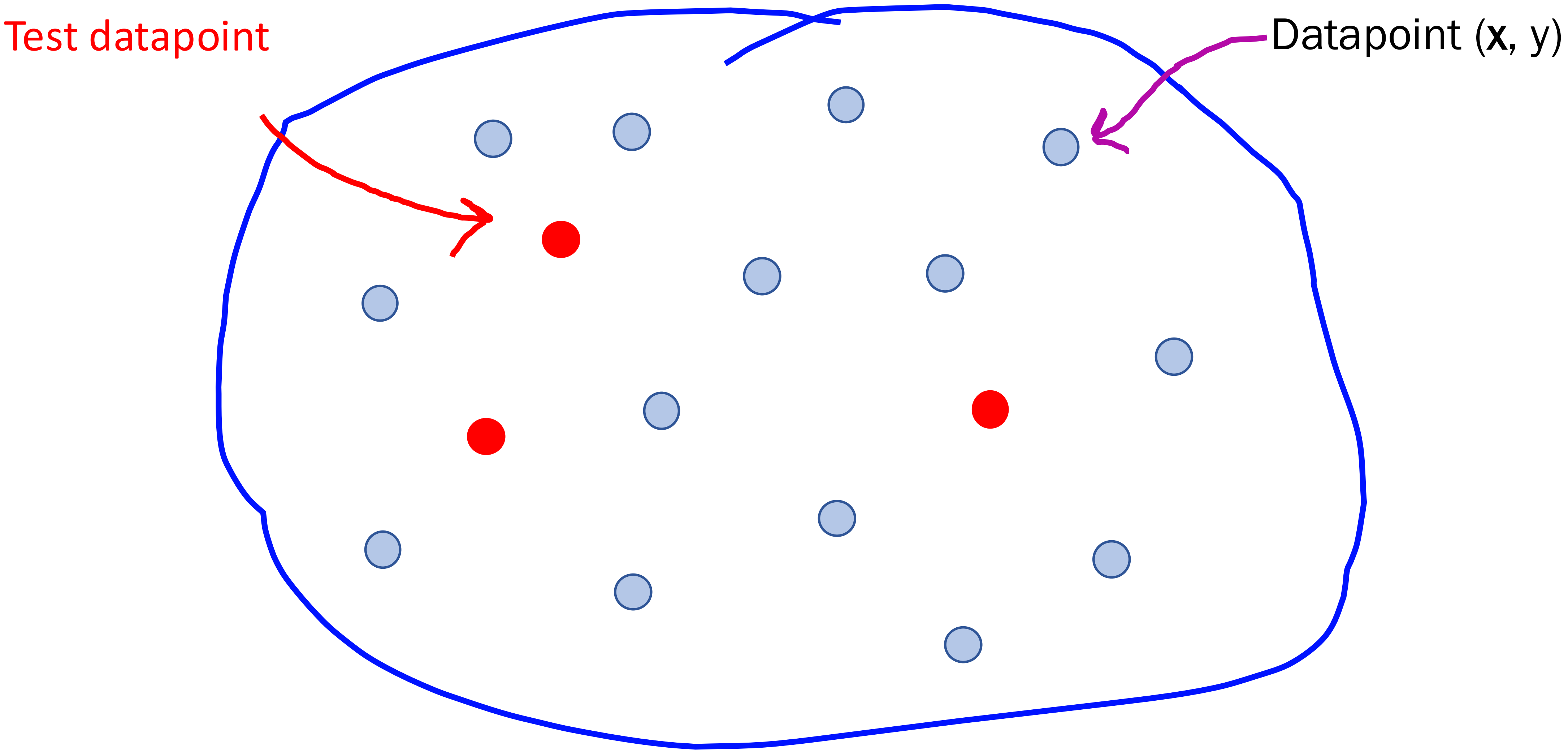
Algorithm	Accuracy on Train Data	Accuracy on Test Data
Logistic Regression	0.7	0.65
Naïve Bayes	0.6	0.60
Nearest Neighbor	1.0	0.56

Train / Test Split



Datapoint (**x**, y)

Train / Test Split

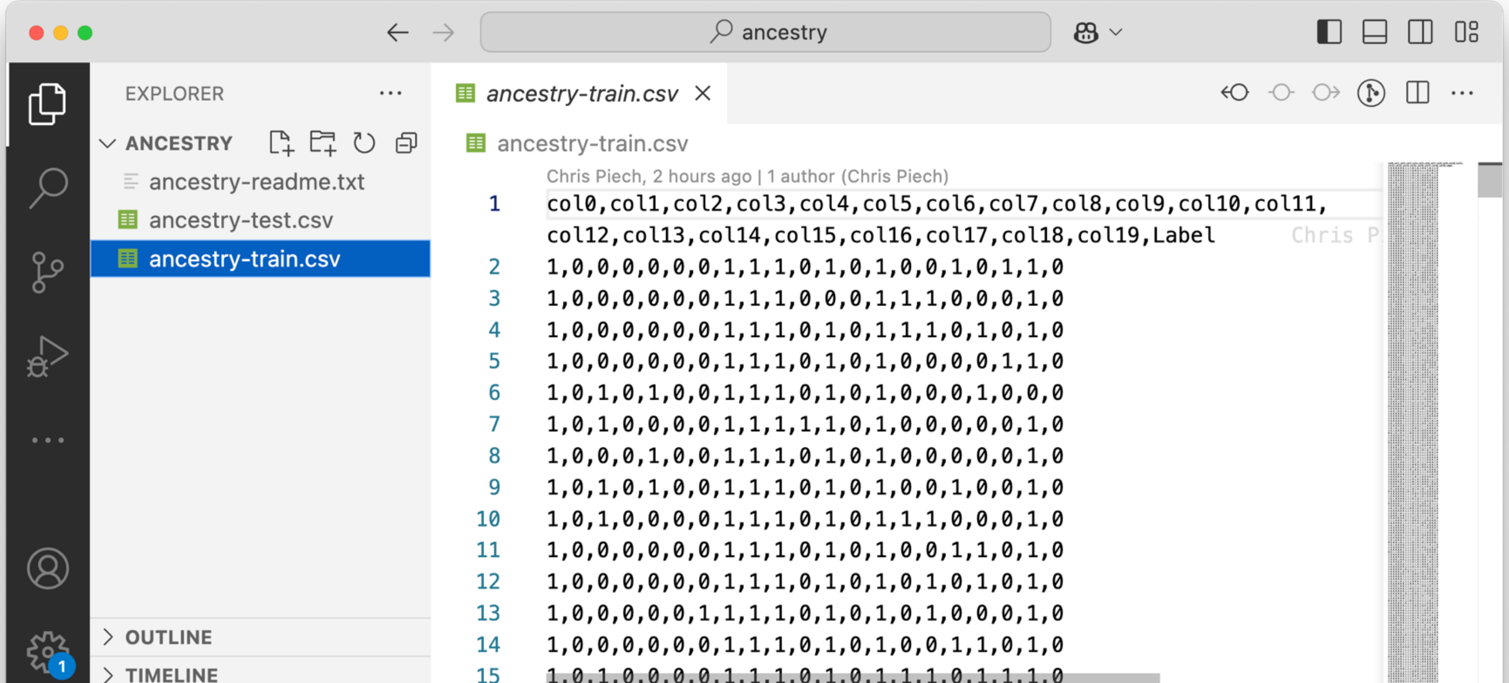


Train / Test Split

```
# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train dataset
print(X_train, y_train)

# Test dataset
print(X_test, y_test)
```

In your pset we will give you these pre-split

train.csv



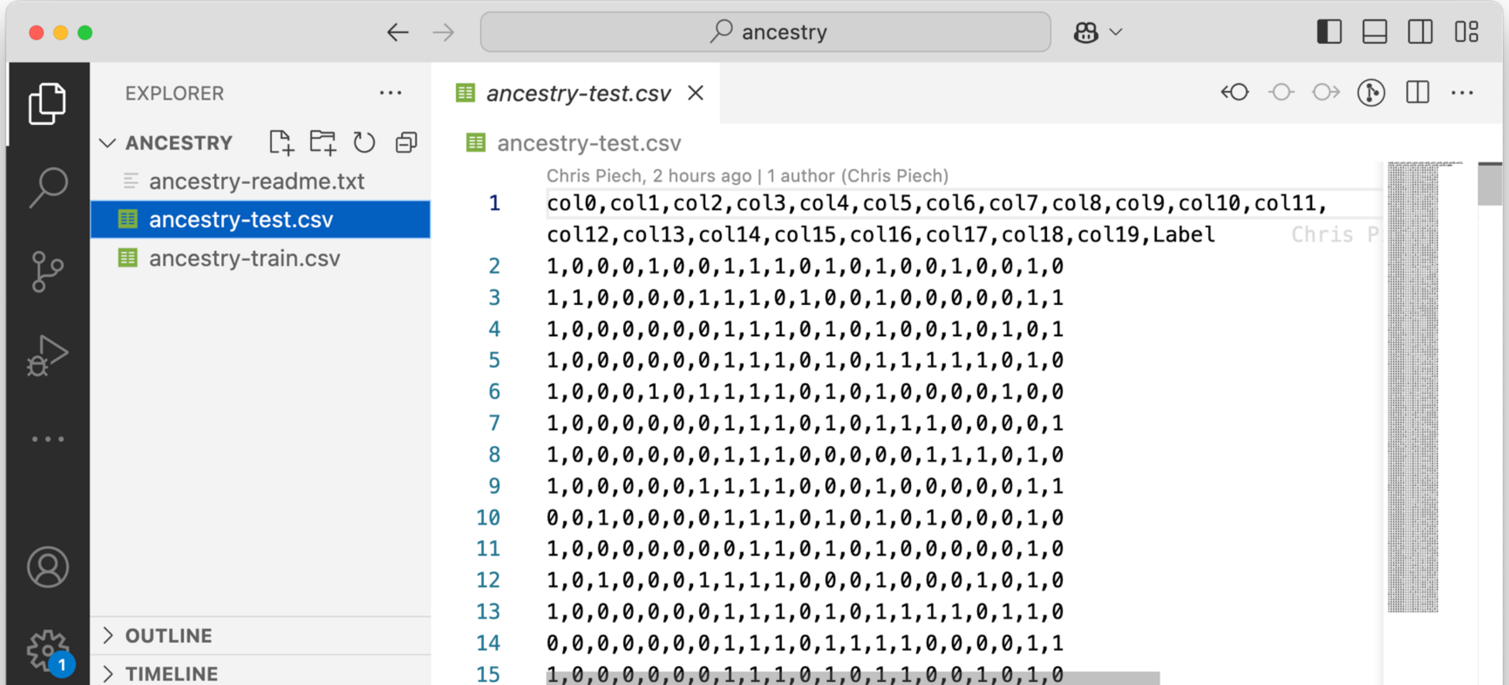
```
col0,col1,col2,col3,col4,col5,col6,col7,col8,col9,col10,col11,
col12,col13,col14,col15,col16,col17,col18,col19,Label
1,0,0,0,0,0,0,1,1,1,0,1,0,1,0,0,1,0,1,1,0
1,0,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,0
1,0,0,0,0,0,0,1,1,1,0,1,0,1,1,1,0,1,0,1,0
1,0,0,0,0,0,0,1,1,1,0,1,0,1,0,0,0,0,1,1,0
1,0,1,0,1,0,0,1,1,1,0,1,0,1,0,0,0,1,0,0,0
1,0,1,0,0,0,0,1,1,1,1,1,0,1,0,0,0,0,0,1,0
1,0,0,0,1,0,0,1,1,1,0,1,0,1,0,0,0,0,0,1,0
1,0,1,0,1,0,0,1,1,1,0,1,0,1,0,0,1,0,0,1,0
1,0,1,0,0,0,0,1,1,1,0,1,0,1,1,1,0,0,0,1,0
1,0,0,0,0,0,0,1,1,1,0,1,0,1,0,0,1,1,0,1,0
1,0,0,0,0,0,0,1,1,1,0,1,0,1,0,1,0,1,0,1,0
1,0,0,0,0,0,1,1,1,1,0,1,0,1,0,1,0,0,0,1,0
1,0,0,0,0,0,0,1,1,1,0,1,0,1,0,0,1,1,0,1,0
1,0,1,0,0,0,0,1,1,1,0,1,0,1,1,1,0,1,1,1,0
```

Train / Test Split

```
# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train dataset
print(X_train, y_train)

# Test dataset
print(X_test, y_test)
```

In your pset we will give you these pre-split

test.csv



```
col0,col1,col2,col3,col4,col5,col6,col7,col8,col9,col10,col11,
col12,col13,col14,col15,col16,col17,col18,col19,Label
1,0,0,0,1,0,0,1,1,1,0,1,0,1,0,0,1,0,0,1,0
1,1,0,0,0,0,1,1,1,0,1,0,0,1,0,0,0,0,0,1,1
1,0,0,0,0,0,0,1,1,1,0,1,0,1,0,0,1,0,1,0,1
1,0,0,0,0,0,0,1,1,1,0,1,0,1,1,1,1,1,0,1,0
1,0,0,0,1,0,1,1,1,1,0,1,0,1,0,0,0,0,1,0,0
1,0,0,0,0,0,0,1,1,1,0,1,0,1,1,1,0,0,0,0,1
1,0,0,0,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,0
1,0,0,0,0,0,1,1,1,1,0,0,0,1,0,0,0,0,0,1,1
0,0,1,0,0,0,0,1,1,1,0,1,0,1,0,1,0,0,0,1,0
1,0,0,0,0,0,0,0,1,1,0,1,0,1,0,0,0,0,0,1,0
1,0,1,0,0,0,1,1,1,1,0,0,0,1,0,0,0,1,0,1,0
1,0,0,0,0,0,0,1,1,1,0,1,0,1,1,1,1,0,1,1,0
0,0,0,0,0,0,0,1,1,1,0,1,1,1,1,0,0,0,0,1,1
1,0,0,0,0,0,0,1,1,1,0,1,0,1,1,0,0,1,0,1,0
```

To the Code

Results

```
Model                    Train Accuracy  Test Accuracy
-----------------------------------------------------------
Baseline                 0.6031          0.5887
Naive Bayes              0.7909          0.8067
Logistic Regression      0.8169          0.8307
Decision Tree            0.8514          0.8307
Random Forest            0.8726          0.8500
Gradient Boosting        0.8611          0.8440
AdaBoost                 0.8334          0.8353
BayesNet                 0.8320          0.8507
```

Results

Model	Train Accuracy	Test Accuracy
Baseline	**0.6031**	**0.5887**
Naive Bayes	0.7909	0.8067
Logistic Regression	0.8169	0.8307
Decision Tree	0.8514	0.8307
Random Forest	0.8726	0.8500
Gradient Boosting	0.8611	0.8440
AdaBoost	0.8334	0.8353
BayesNet	0.8320	0.8507

Always predict 1

Results

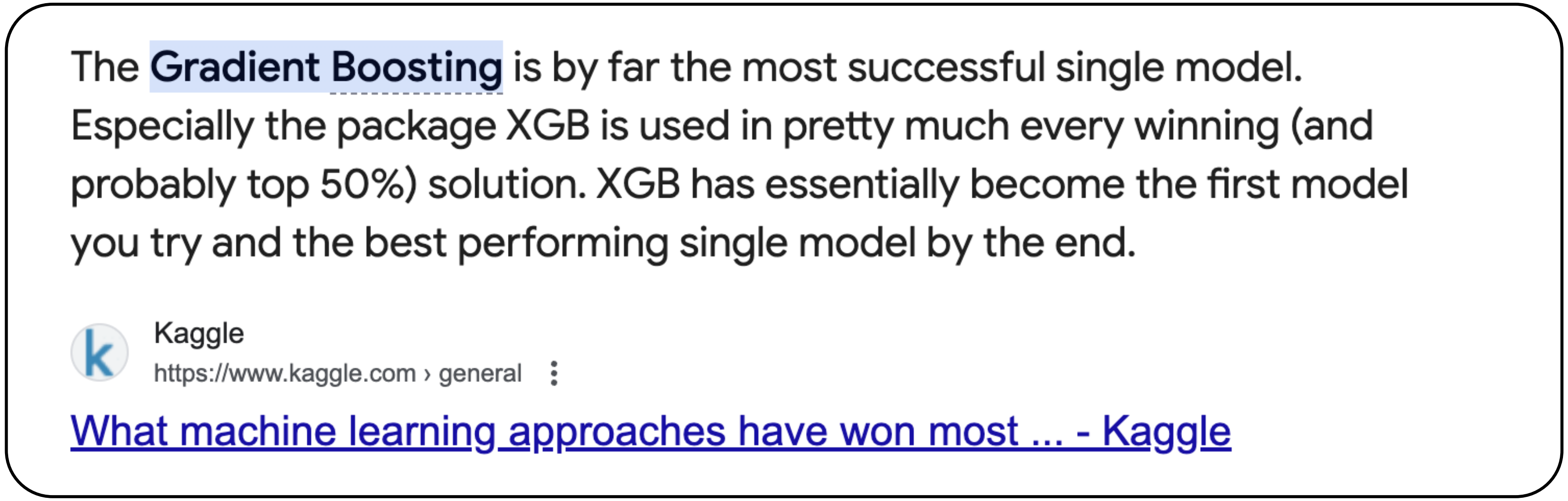
Model	Train Accuracy	Test Accuracy
Baseline	0.6031	0.5887
Naive Bayes	0.7909	0.8067
Logistic Regression	0.8169	0.8307
Decision Tree	**0.8514**	**0.8307**
Random Forest	**0.8726**	**0.8500**
Gradient Boosting	**0.8611**	**0.8440**
AdaBoost	0.8334	0.8353
BayesNet	0.8320	0.8507

overfitting

Results

Model	Train Accuracy	Test Accuracy
Baseline	0.6031	0.5887
Naive Bayes	0.7909	0.8067
Logistic Regression	0.8169	0.8307
Decision Tree	0.8514	0.8307
Random Forest	0.8726	0.8500
Gradient Boosting	**0.8611**	**0.8440**
AdaBoost	0.8334	0.8353
BayesNet	0.8320	0.8507

The Kaggle Champion



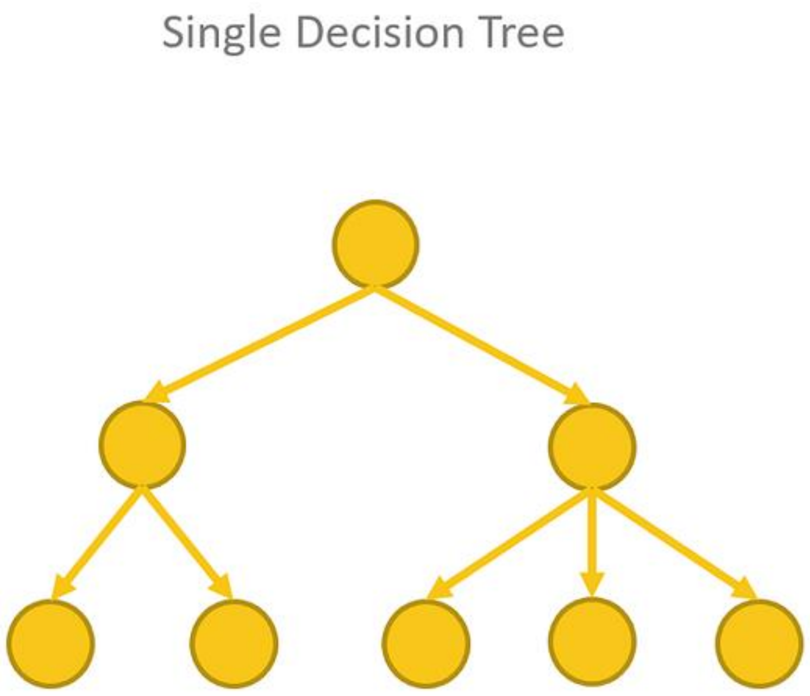
The **Gradient Boosting** is by far the most successful single model. Especially the package XGB is used in pretty much every winning (and probably top 50%) solution. XGB has essentially become the first model you try and the best performing single model by the end.

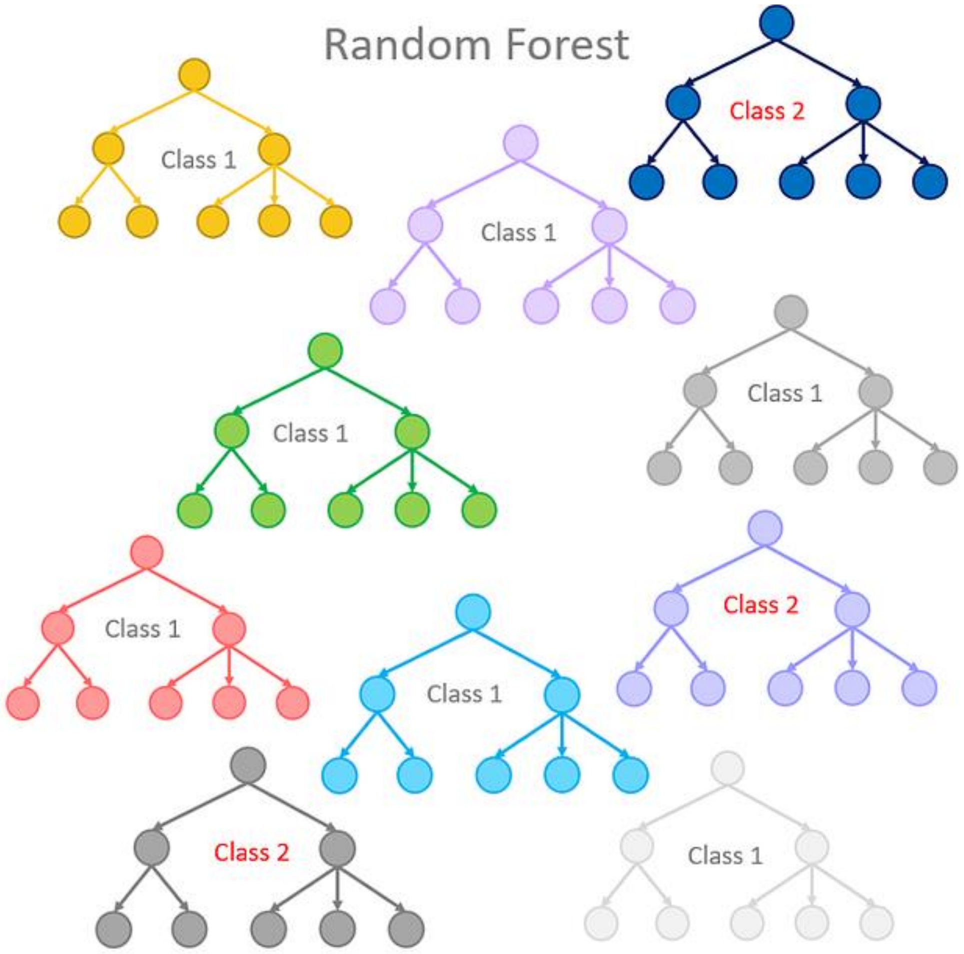
Kaggle
https://www.kaggle.com › general

What machine learning approaches have won most ... - Kaggle

XGB is for extreme gradient boosting

Decision Trees vs Random Forests

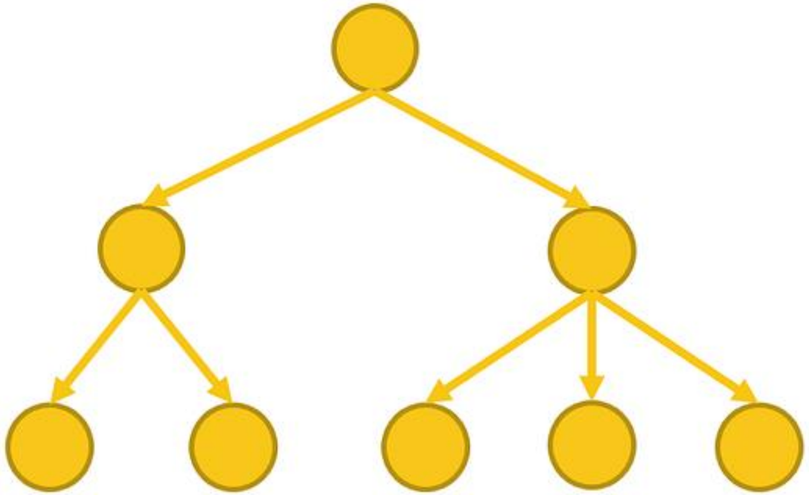




Each tree is trained on a bootstrap sampling of the data

Gradient Boosting

Successive small decision trees

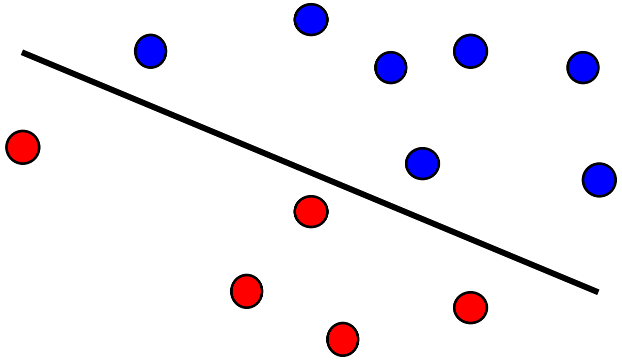


Are predicting the "residual"

$$\frac{\partial LL(\theta)}{\partial \hat{y}^{(i)}} = \underline{y^{(i)} - \hat{y}^{(i)}}$$

Discrimination Intuition

- Logistic regression is trying to fit a **line** that separates data instances where *y* = 1 from those where *y* = 0



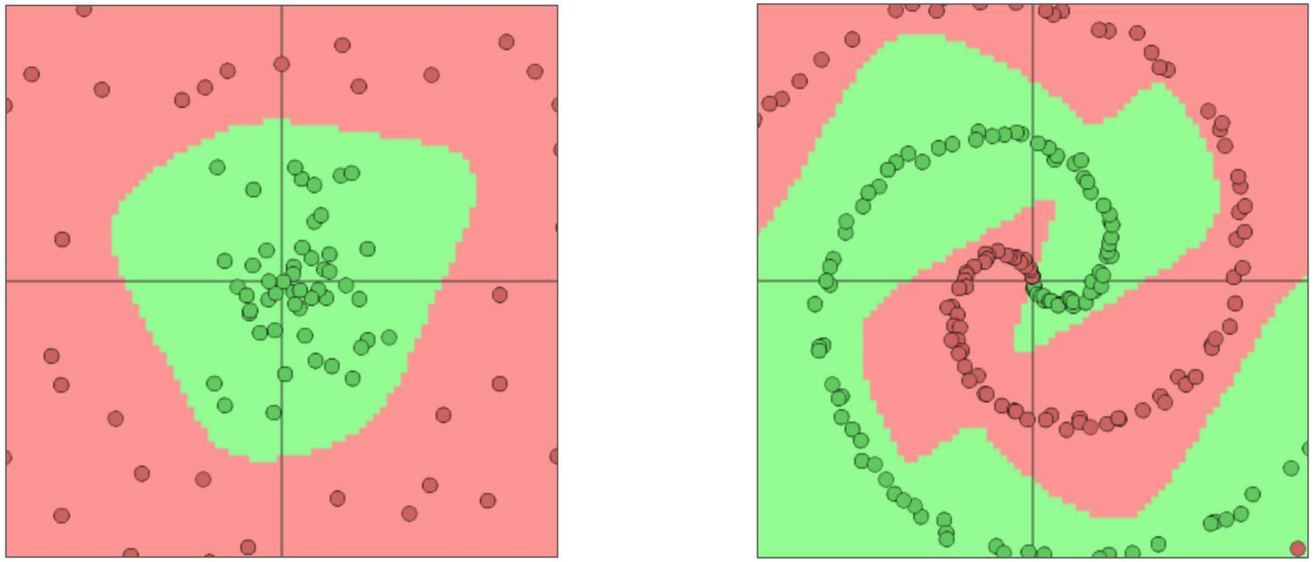
$$\theta^T \mathbf{x} = 0$$

$$\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_m x_m = 0$$

- We call such data (or the functions generating the data) "**linearly separable**"
- **Naïve bayes is linear too** as there is no interaction between different features.

Some Data Not Linearly Seperable

- Some data sets/functions are not separable



  - Not possible to draw a line that successfully separates all the *y* = 1 points (green) from the *y* = 0 points (red)
  - Despite this fact, logistic regression and Naive Bayes still often work well in practice

Three Guiding Questions

1. What are other models for classification? ✓

2. For a dataset, how do you tell which one is best? ✓

3. What are other validation metrics I might care about?

How **well calibrated** are my probabilities?

Measuring if Probabilities are Calibrated

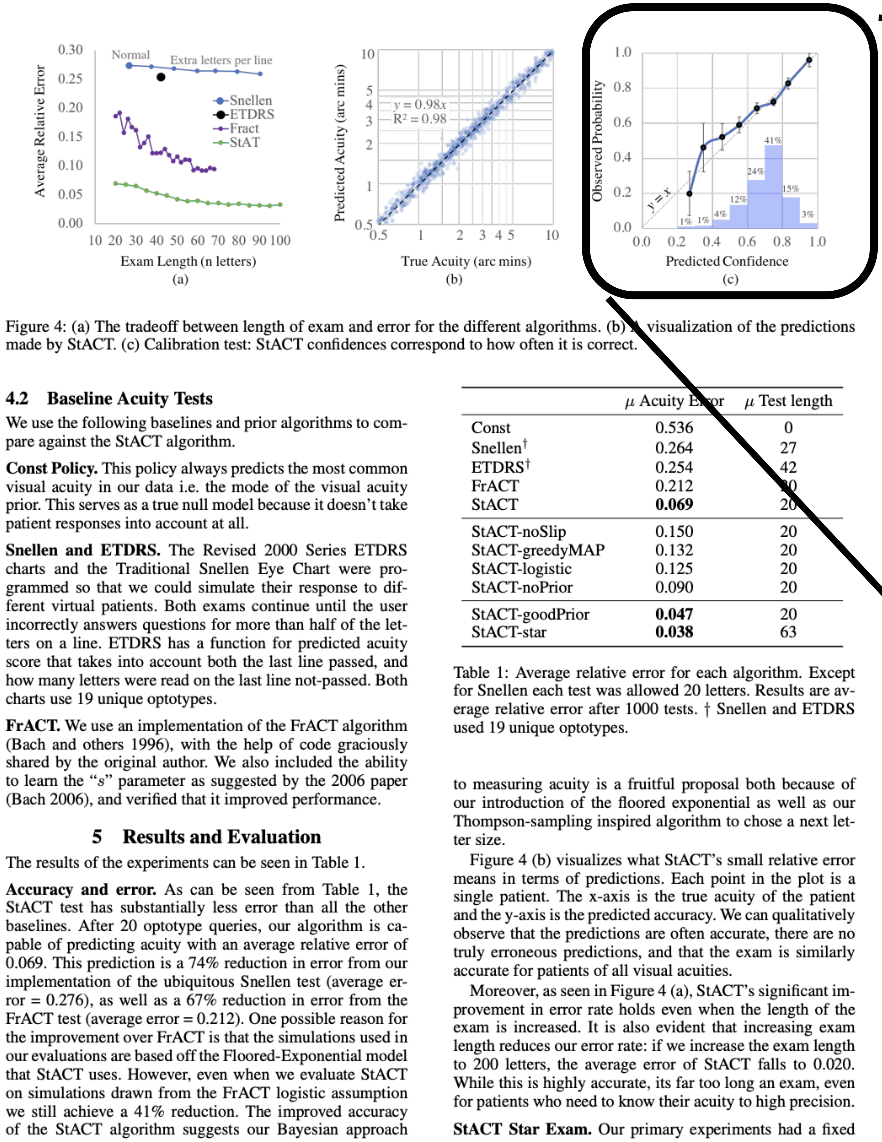


Figure 4: (a) The tradeoff between length of exam and error for the different algorithms. (b) A visualization of the predictions made by StACT. (c) Calibration test: StACT confidences correspond to how often it is correct.

4.2 Baseline Acuity Tests

We use the following baselines and prior algorithms to compare against the StACT algorithm.

Const Policy. This policy always predicts the most common visual acuity in our data i.e. the mode of the visual acuity prior. This serves as a true null model because it doesn't take patient responses into account at all.

Snellen and ETDRS. The Revised 2000 Series ETDRS charts and the Traditional Snellen Eye Chart were programmed so that we could simulate their response to different virtual patients. Both exams continue until the user incorrectly answers questions for more than half of the letters on a line. ETDRS has a function for predicted acuity score that takes into account both the last line passed, and how many letters were read on the last line not-passed. Both charts use 19 unique optotypes.

FrACT. We use an implementation of the FrACT algorithm (Bach and others 1996), with the help of code graciously shared by the original author. We also included the ability to learn the "*s*" parameter as suggested by the 2006 paper (Bach 2006), and verified that it improved performance.

5 Results and Evaluation

The results of the experiments can be seen in Table 1.

Accuracy and error. As can be seen from Table 1, the StACT test has substantially less error than all the other baselines. After 20 optotype queries, our algorithm is capable of predicting acuity with an average relative error of 0.069. This prediction is a 74% reduction in error from our implementation of the ubiquitous Snellen test (average error = 0.276), as well as a 67% reduction in error from the FrACT test (average error = 0.212). One possible reason for the improvement over FrACT is that the simulations used in our evaluations are based off the Floored-Exponential model that StACT uses. However, even when we evaluate StACT on simulations drawn from the FrACT logistic assumption we still achieve a 41% reduction. The improved accuracy of the StACT algorithm suggests our Bayesian approach to measuring acuity is a fruitful proposal both because of our introduction of the floored exponential as well as our Thompson-sampling inspired algorithm to chose a next letter size.

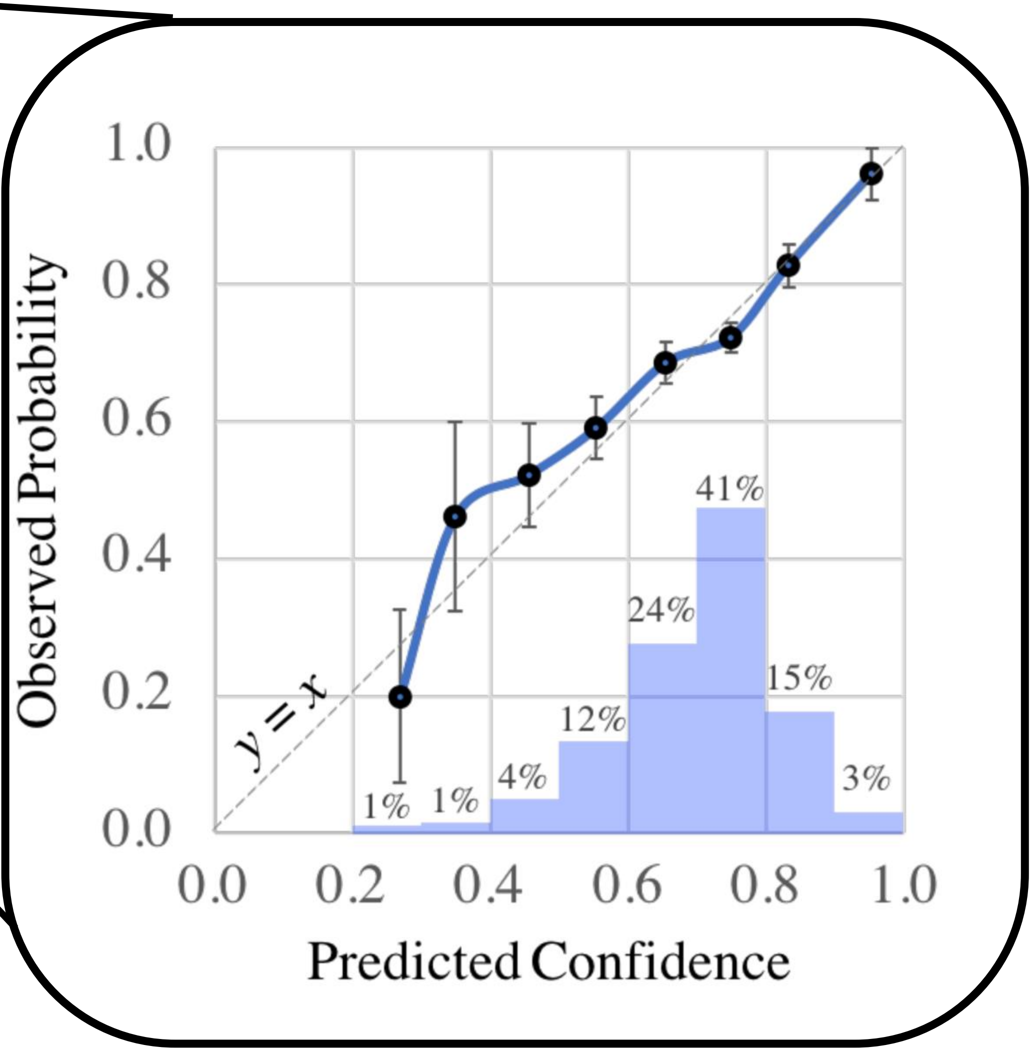
	μ Acuity Error	μ Test length
Const	0.536	0
Snellen[†]	0.264	27
ETDRS[†]	0.254	42
FrACT	0.212	20
StACT	**0.069**	20
StACT-noSlip	0.150	20
StACT-greedyMAP	0.132	20
StACT-logistic	0.125	20
StACT-noPrior	0.090	20
StACT-goodPrior	**0.047**	20
StACT-star	**0.038**	63

Table 1: Average relative error for each algorithm. Except for Snellen each test was allowed 20 letters. Results are average relative error after 1000 tests. † Snellen and ETDRS used 19 unique optotypes.

Figure 4 (b) visualizes what StACT's small relative error means in terms of predictions. Each point in the plot is a single patient. The x-axis is the true acuity of the patient and the y-axis is the predicted accuracy. We can qualitatively observe that the predictions are often accurate, there are no truly erroneous predictions, and that the exam is similarly accurate for patients of all visual acuities.

Moreover, as seen in Figure 4 (a), StACT's significant improvement in error rate holds even when the length of the exam is increased. It is also evident that increasing exam length reduces our error rate: if we increase the exam length to 200 letters, the average error of StACT falls to 0.020. While this is highly accurate, its far too long an exam, even for patients who need to know their acuity to high precision.

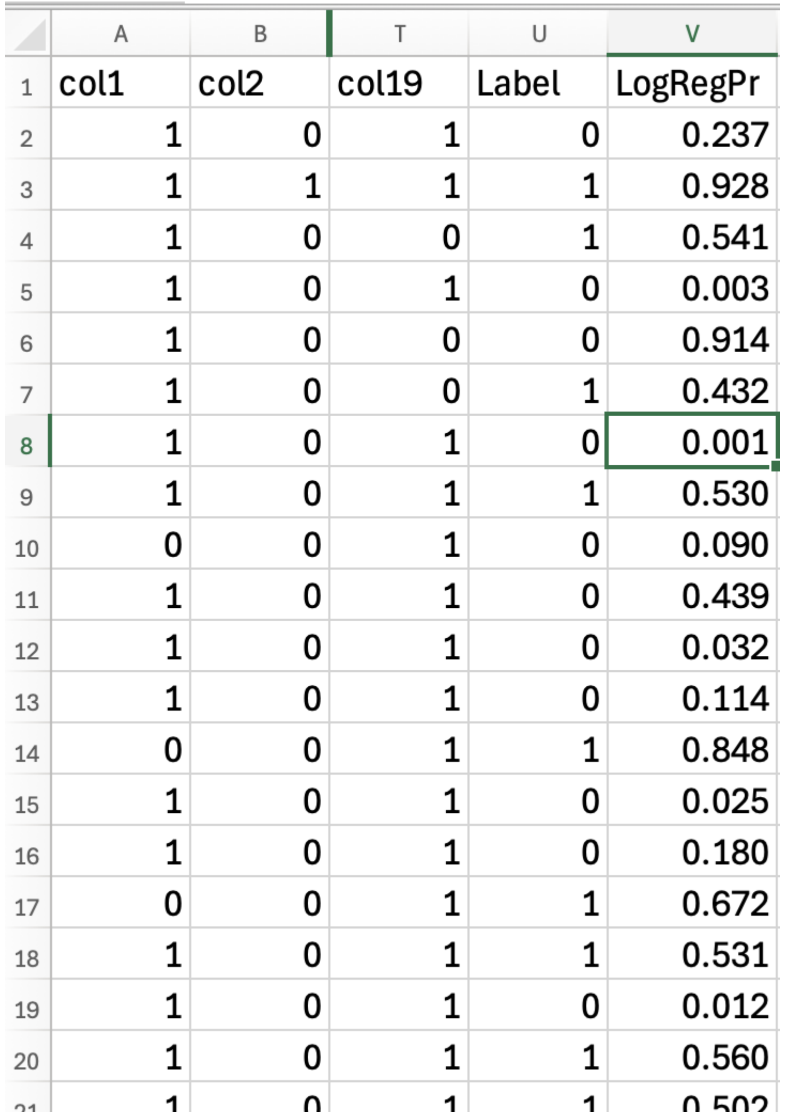
StACT Star Exam. Our primary experiments had a fixed



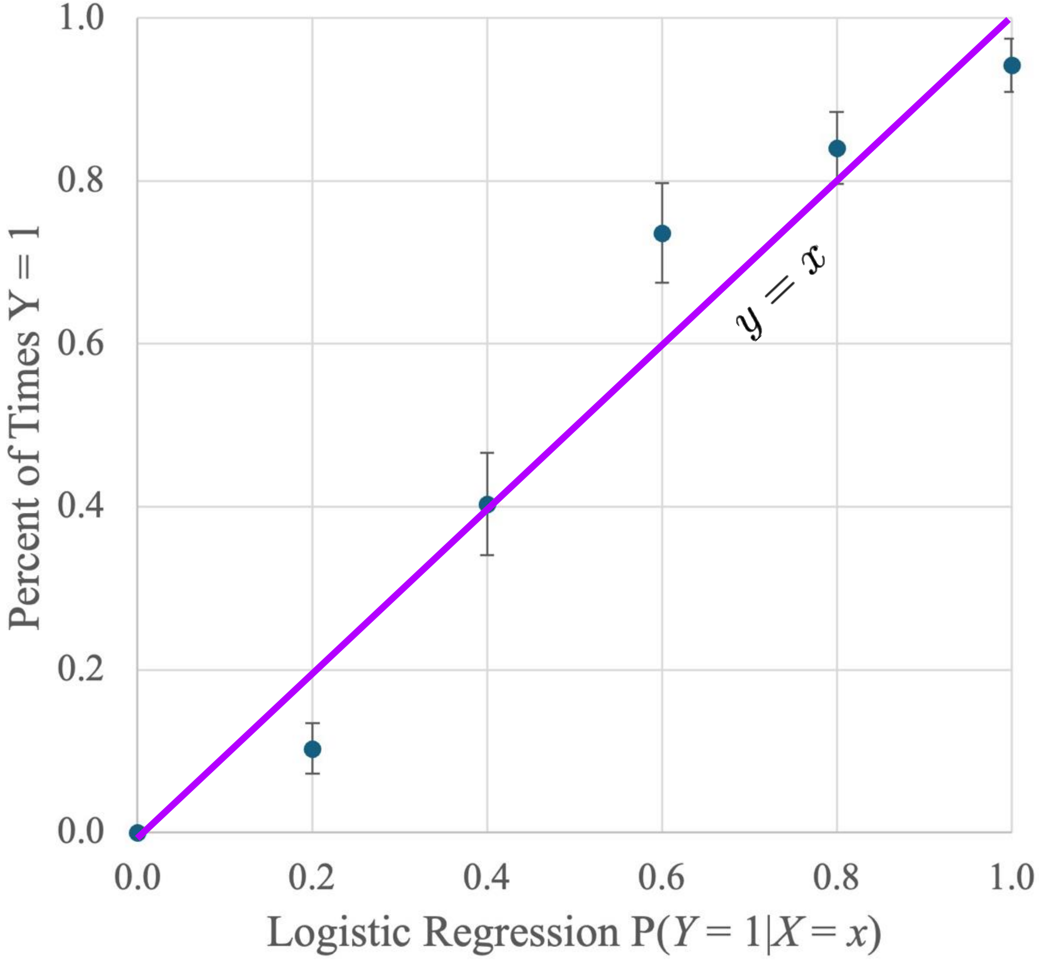
Calibration Curve

x_1 x_2 x_{19} y

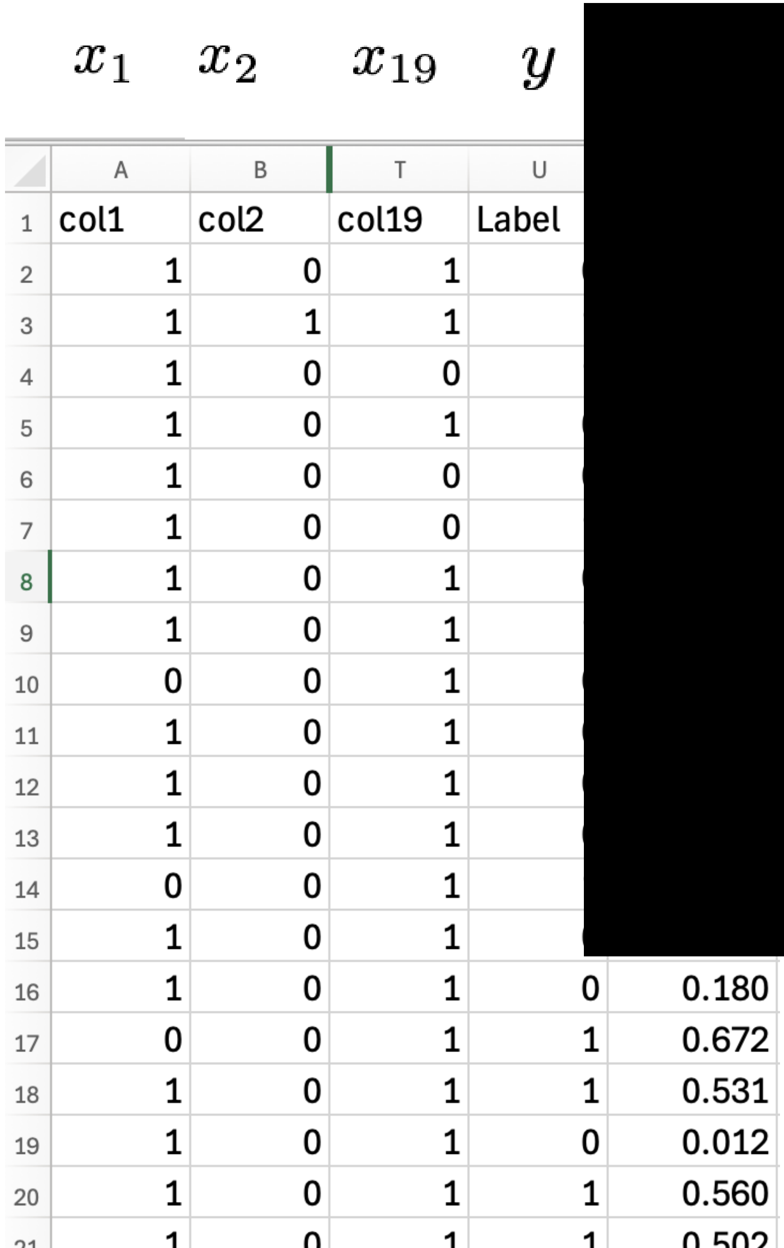
$$P(Y=1|\mathbf{X}=\mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$



	A	B	T	U	V
1	col1	col2	col19	Label	LogRegPr
2	1	0	1	0	0.237
3	1	1	1	1	0.928
4	1	0	0	1	0.541
5	1	0	1	0	0.003
6	1	0	0	0	0.914
7	1	0	0	1	0.432
8	1	0	1	0	0.001
9	1	0	1	1	0.530
10	0	0	1	0	0.090
11	1	0	1	0	0.439
12	1	0	1	0	0.032
13	1	0	1	0	0.114
14	0	0	1	1	0.848
15	1	0	1	0	0.025
16	1	0	1	0	0.180
17	0	0	1	1	0.672
18	1	0	1	1	0.531
19	1	0	1	0	0.012
20	1	0	1	1	0.560
21	1	0	1	1	0.502

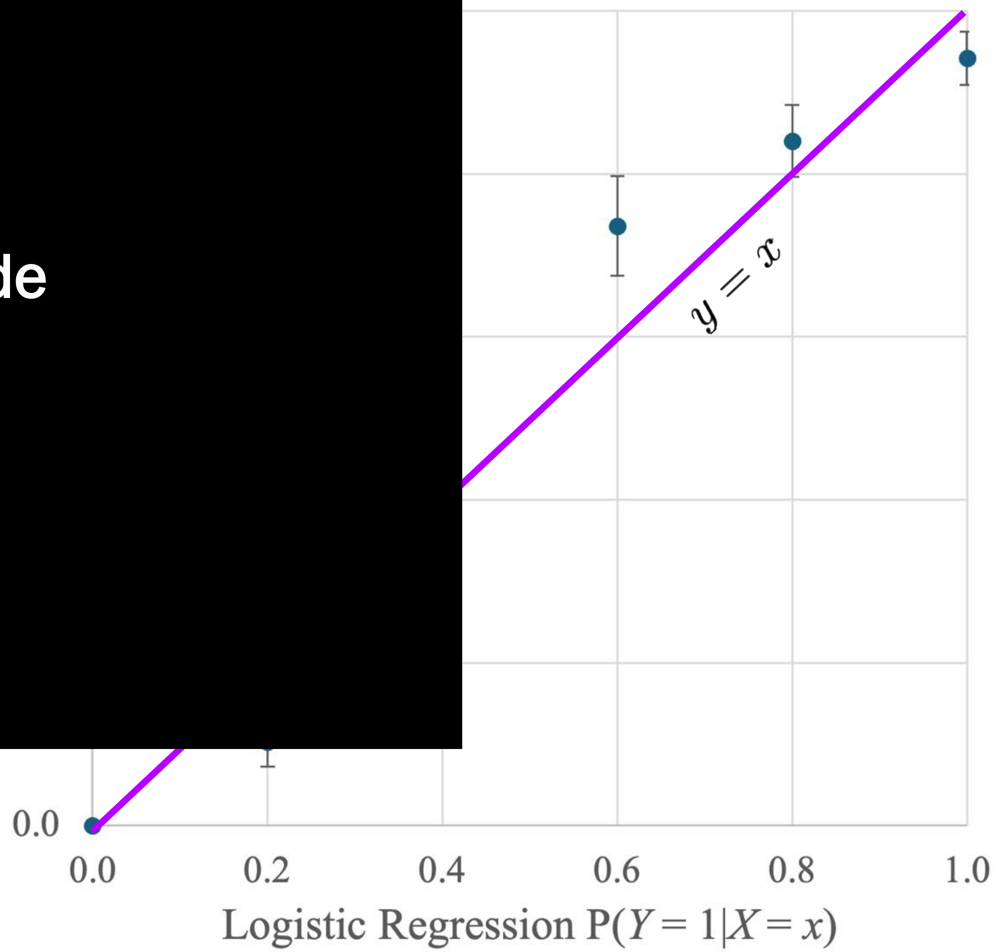


Calibration Curve



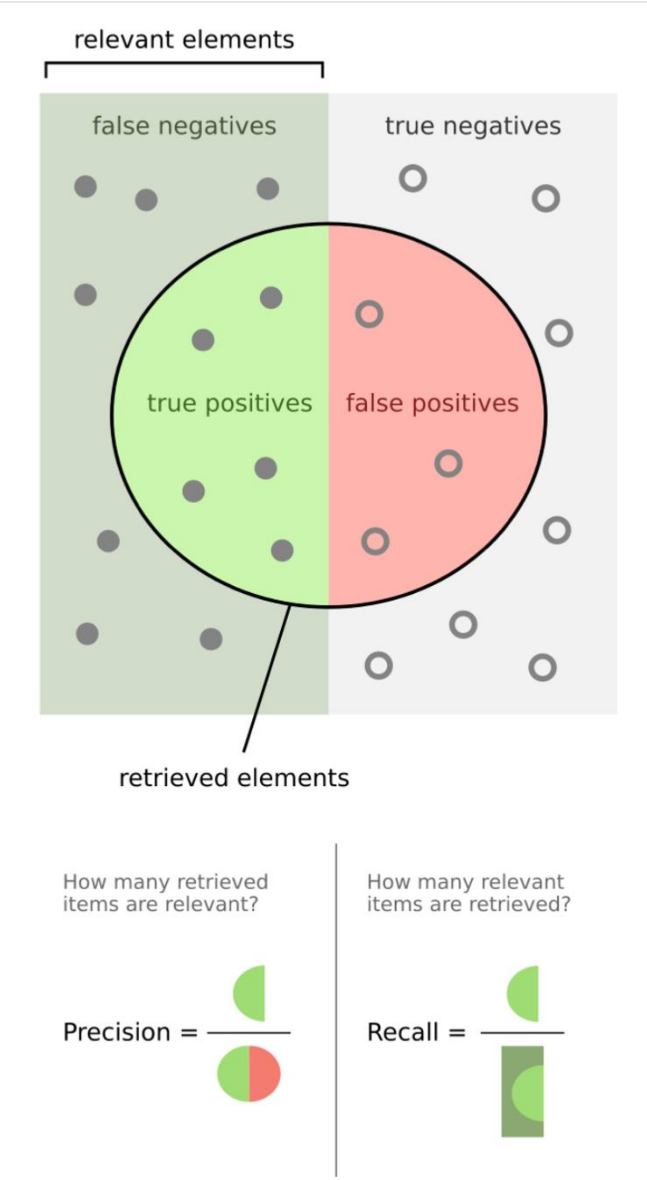
	x_1	x_2	x_{19}	y	
	A	B	T	U	
1	col1	col2	col19	Label	
2	1	0	1		
3	1	1	1		
4	1	0	0		
5	1	0	1		
6	1	0	0		
7	1	0	0		
8	1	0	1		
9	1	0	1		
10	0	0	1		
11	1	0	1		
12	1	0	1		
13	1	0	1		
14	0	0	1		
15	1	0	1		
16	1	0	1	0	0.180
17	0	0	1	1	0.672
18	1	0	1	1	0.531
19	1	0	1	0	0.012
20	1	0	1	1	0.560
21	1	0	1	1	0.502

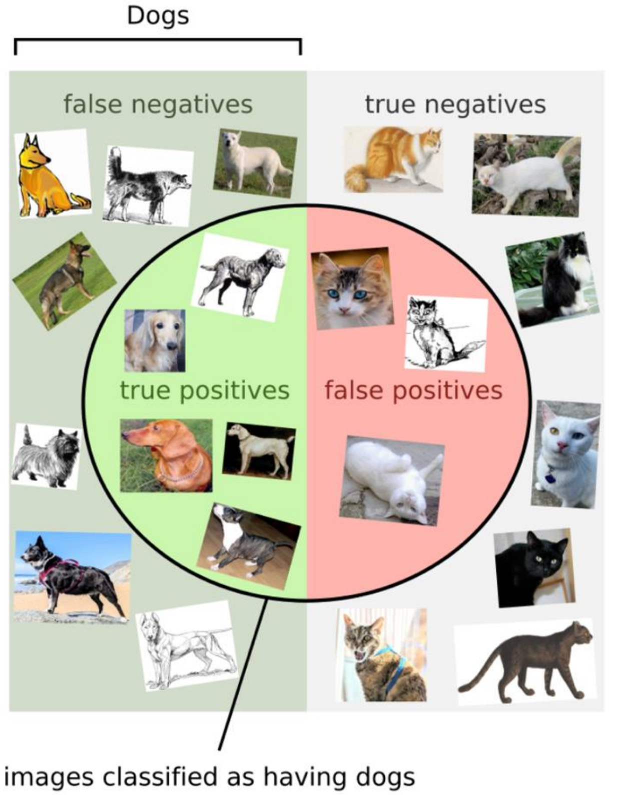
The the code



Is it worse to be wrong when you predict a 1?

Precision Recall Curve

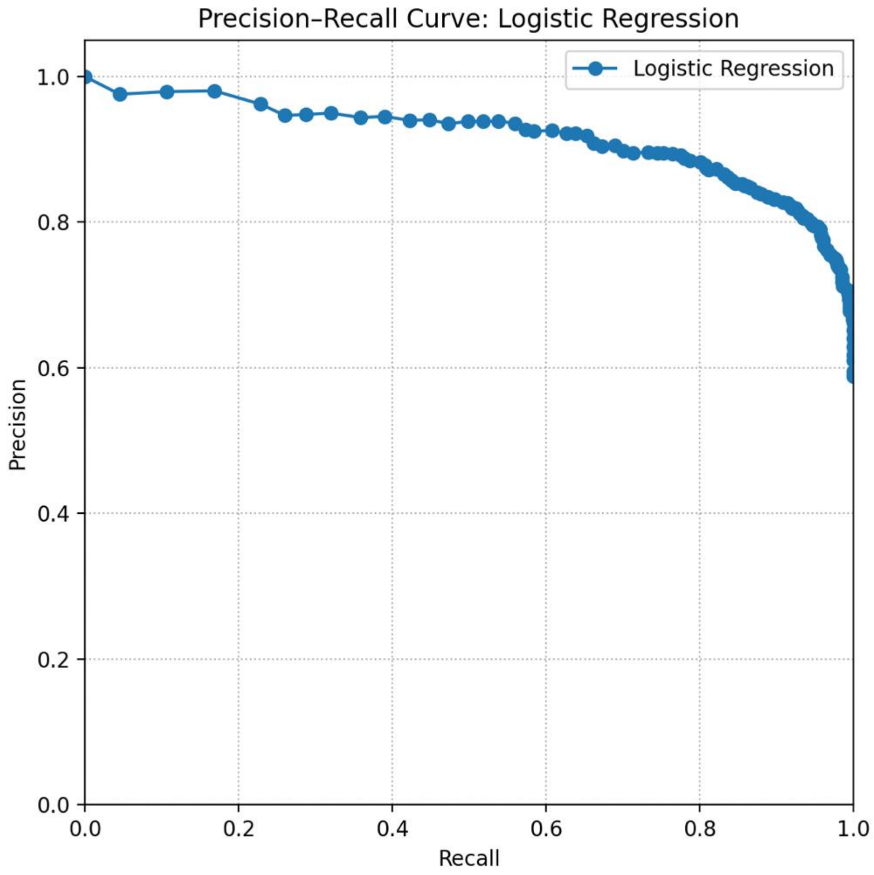




$$\text{Precision} = \frac{\text{5 true pos.}}{\text{8 total pos.}} \qquad \text{Recall} = \frac{\text{5 true pos.}}{\text{12 total dogs}}$$

$$\text{Prevalence} = \frac{\text{12 total dogs}}{\text{22 total images}}$$

$$\text{Accuracy} = \frac{\text{5 true pos. + 7 true neg.}}{\text{22 total images}}$$



Dog classifier. Image credit: wikipedia

Is the Model Fair?

(pedagogical pause)

Two Philosophic Values of Fairness

Procedural Fairness:

Focuses on the decision-making or classification *process*, ensures that the algorithm does not rely on unfair features.

Distributive Fairness:

Focuses on the decision-making or classification *outcome*, ensures that the distribution of good and bad outcomes is equitable.

Two Philosophic Values of Fairness

Procedural Fairness:

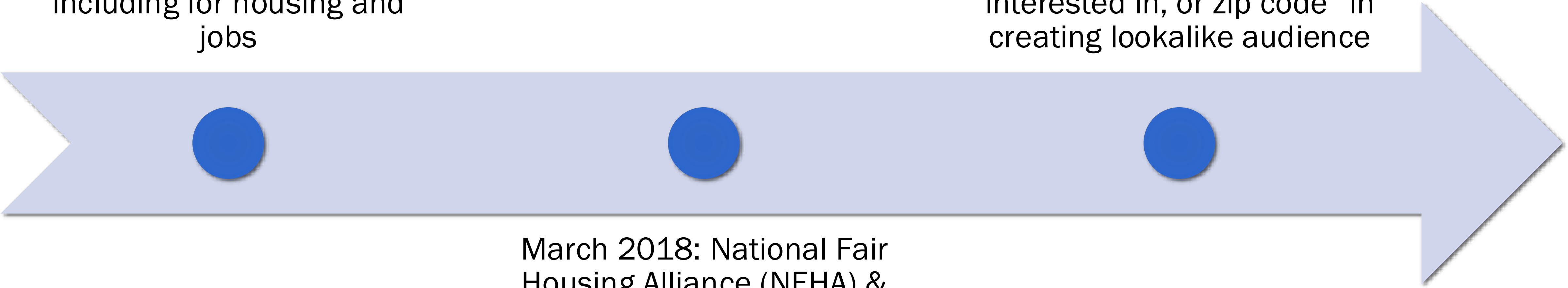
Focuses on the decision-making or classification *process*, ensures that the algorithm does not rely on unfair features.

Distributive Fairness:

Focuses on the decision-making or classification *outcome*, ensures that the distribution of good and bad outcomes is equitable.

Fairness through unawareness

Case Study: Facebook Ads & Job/Housing Recommendations

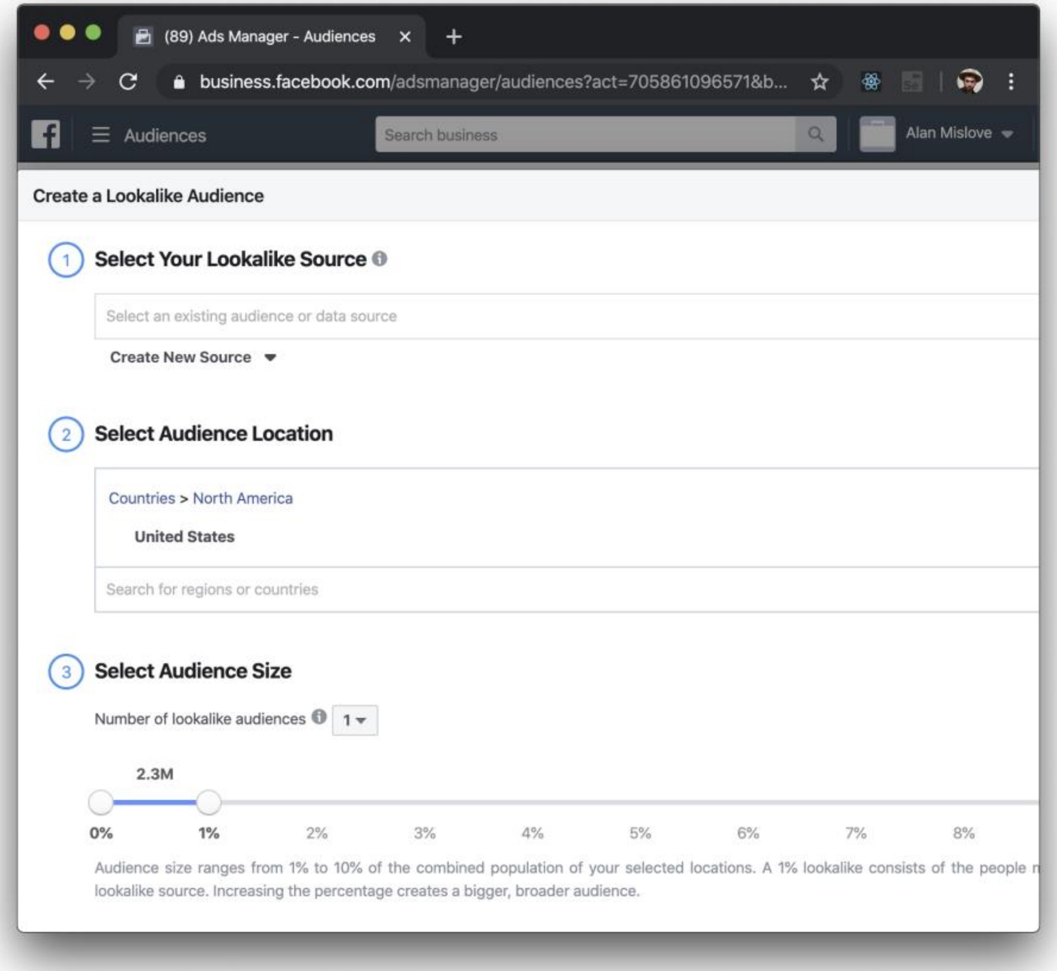


Facebook creates "Lookalike" feature for advertisers: upload a "source list" and find users with "common qualities" to target ads, including for housing and jobs

March 2018: National Fair Housing Alliance (NFHA) & other civil rights groups sue Facebook over violations of the Fair Housing Act

March 2019: As part of settlement, Facebook agrees not to use "age, gender, relationship status, religious views, school, political views, interested in, or zip code" in creating lookalike audience

Facebook Input Lookalikes


(89) Ads Manager - Audiences
business.facebook.com/adsmanager/audiences?act=705861096571&b...
Audiences
Search business
Alan Mislove

Create a Lookalike Audience

1 Select Your Lookalike Source

Select an existing audience or data source

Create New Source

2 Select Audience Location

Countries > North America

United States

Search for regions or countries

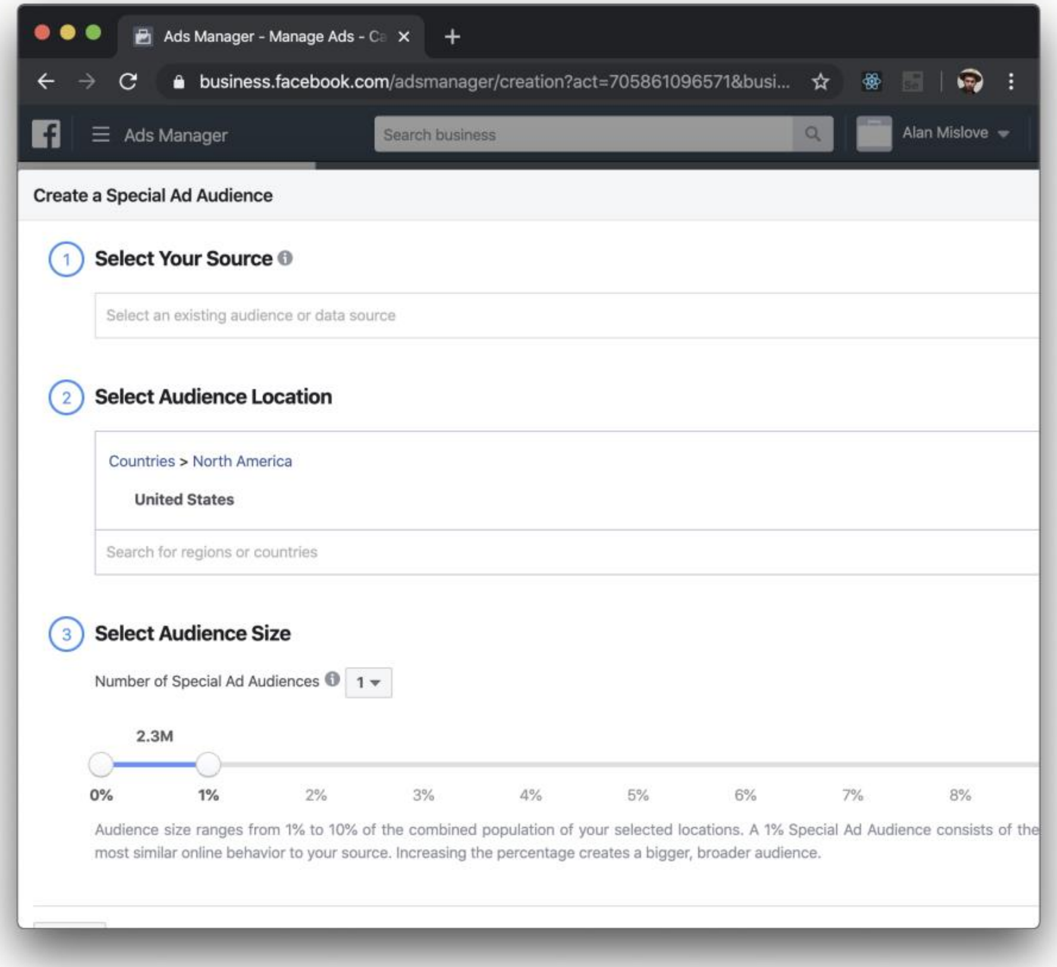
3 Select Audience Size

Number of lookalike audiences 1

2.3M

0% 1% 2% 3% 4% 5% 6% 7% 8%

Audience size ranges from 1% to 10% of the combined population of your selected locations. A 1% lookalike consists of the people r lookalike source. Increasing the percentage creates a bigger, broader audience.


Ads Manager - Manage Ads - Ca
business.facebook.com/adsmanager/creation?act=705861096571&busi...
Ads Manager
Search business
Alan Mislove

Create a Special Ad Audience

1 Select Your Source

Select an existing audience or data source

2 Select Audience Location

Countries > North America

United States

Search for regions or countries

3 Select Audience Size

Number of Special Ad Audiences 1

2.3M

0% 1% 2% 3% 4% 5% 6% 7% 8%

Audience size ranges from 1% to 10% of the combined population of your selected locations. A 1% Special Ad Audience consists of the most similar online behavior to your source. Increasing the percentage creates a bigger, broader audience.

New "Special Ad" Audiences Still Biased

Gender: Equally Biased

Age: Almost as Biased

Race: more difficult to measure given the tools provided but still somewhat biased

Political Views: Less Biased

Sapiezynski et. al 2019,

https://sapiezynski.com/papers/sapiezynski2019algorithms.pdf

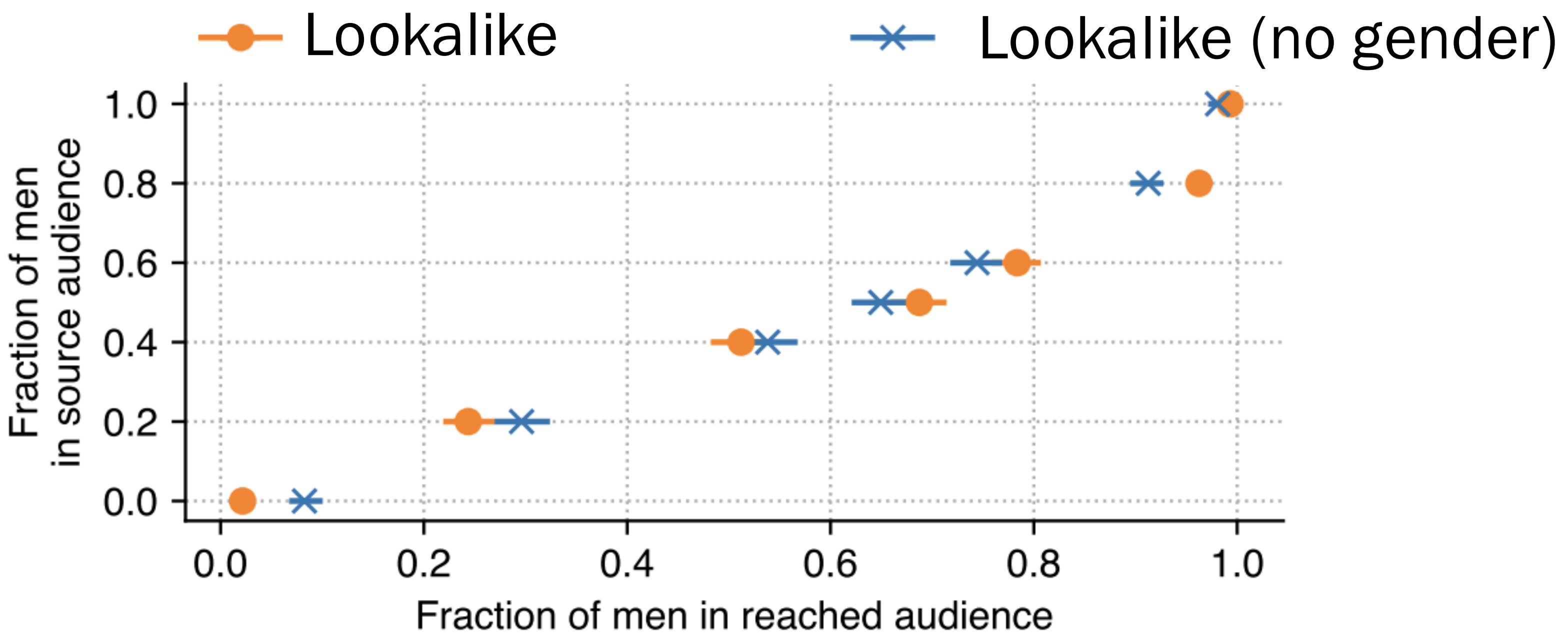
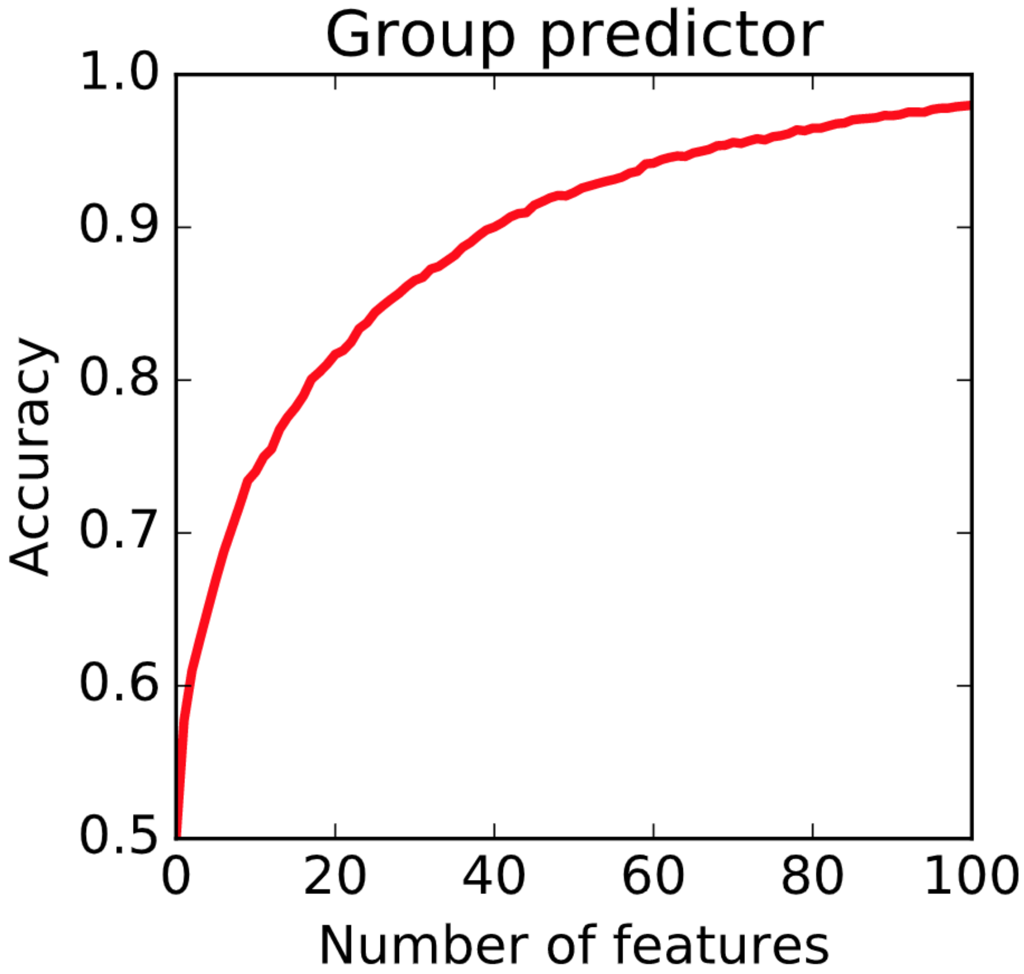


Figure 2: Gender breakdown of ad delivery to Lookalike and Special Ad audiences created from the same source audience with varying fraction of male users, using the same ad creative. We can observe that both Lookalike and Special Ad audiences reflect the gender distribution of the source audience, despite the lack of gender being provided as an input to Special Ad Audiences.

Yo, Piotr, you got your axis backwards ☺



Many Features = Accurate Group Prediction

Sensitive attributes are often “redundantly encoded” in the dataset

Many of the features or datapoints are correlated with the sensitive attribute

Two Philosophic Values of Fairness

Procedural Fairness:

Focuses on the decision-making or classification *process*, ensures that the algorithm does not rely on unfair features.

Distributive Fairness:

Focuses on the decision-making or classification *outcome*, ensures that the distribution of good and bad outcomes is equitable.

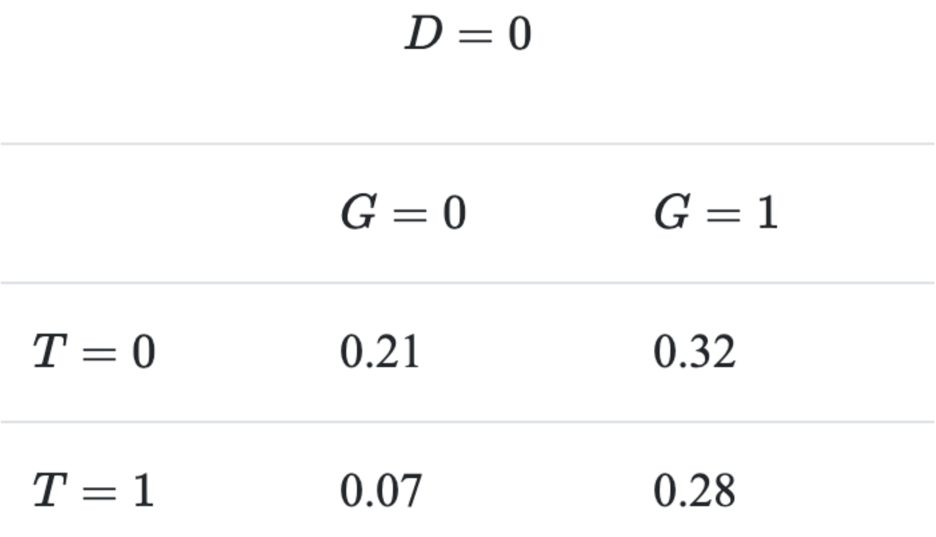
Fairness through unawareness
(facebook example shows this is hard)

Let’s Try Fairness Through Awareness!

Awareness of what?

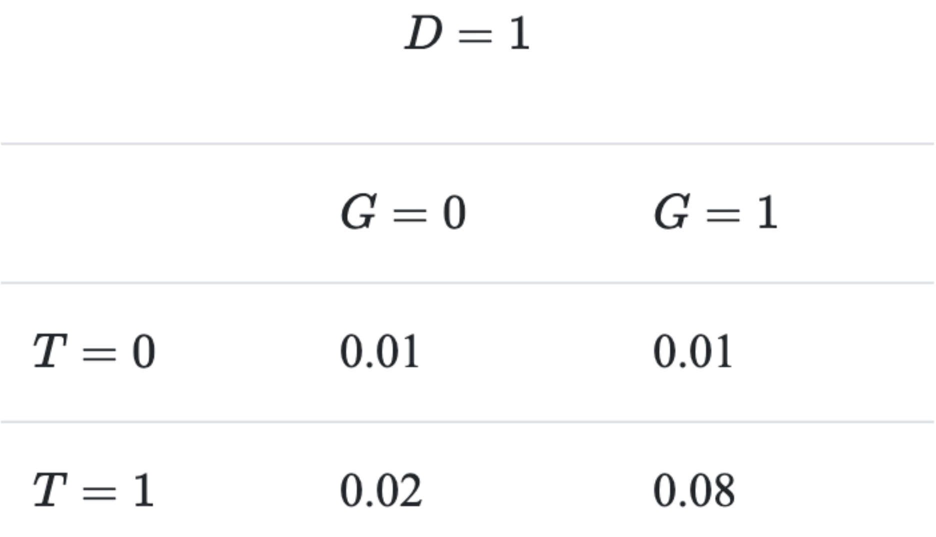
Fairness Through Awareness Terms

D: protected demographic
G: guess of your model (aka y hat)
T: the true value (aka y)



$D = 0$

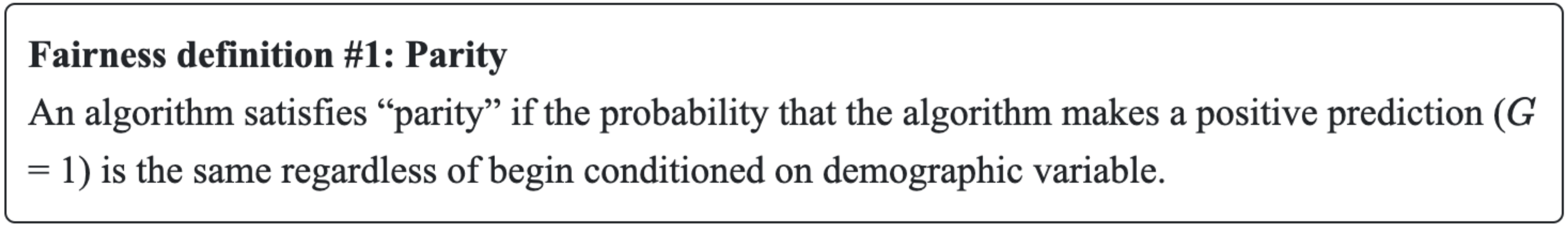
	$G = 0$	$G = 1$
$T = 0$	0.21	0.32
$T = 1$	0.07	0.28



$D = 1$

	$G = 0$	$G = 1$
$T = 0$	0.01	0.01
$T = 1$	0.02	0.08

Distributive Fairness #1: Parity



Fairness definition #1: Parity

An algorithm satisfies “parity” if the probability that the algorithm makes a positive prediction (G = 1) is the same regardless of begin conditioned on demographic variable.

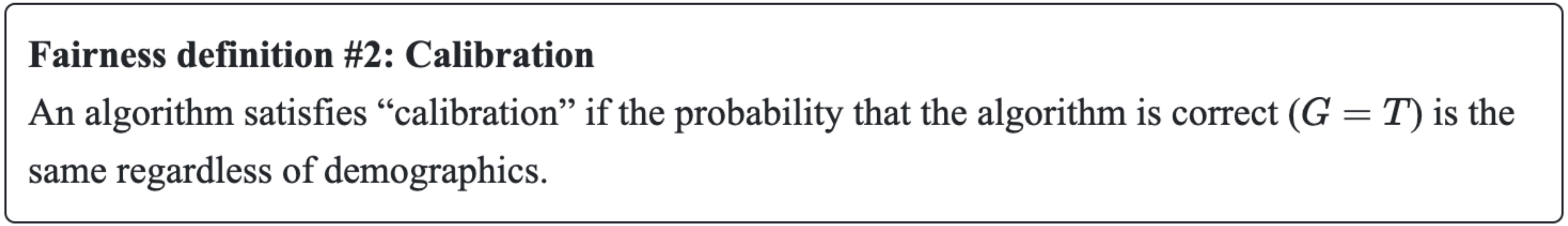
D: protected demographic

G: guess of your model (aka y hat)

T: the true value (aka y)

$$P(G=1|D=1) = P(G = 1 \mid D = 0)$$

Distributive Fairness #2: Calibration



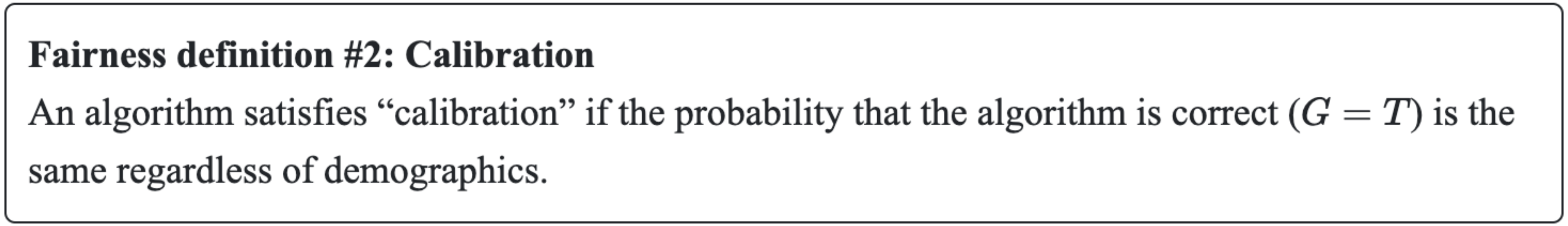
Fairness definition #2: Calibration

An algorithm satisfies "calibration" if the probability that the algorithm is correct ($G = T$) is the same regardless of demographics.

D: protected demographic
G: guess of your model (aka y hat)
T: the true value (aka y)

$$P(G = T | D = 0) = P(G = T | D = 1)$$

Calibration (Relaxed)



Fairness definition #2: Calibration

An algorithm satisfies "calibration" if the probability that the algorithm is correct ($G = T$) is the same regardless of demographics.

D: protected demographic
G: guess of your model (aka y hat)
T: the true value (aka y)

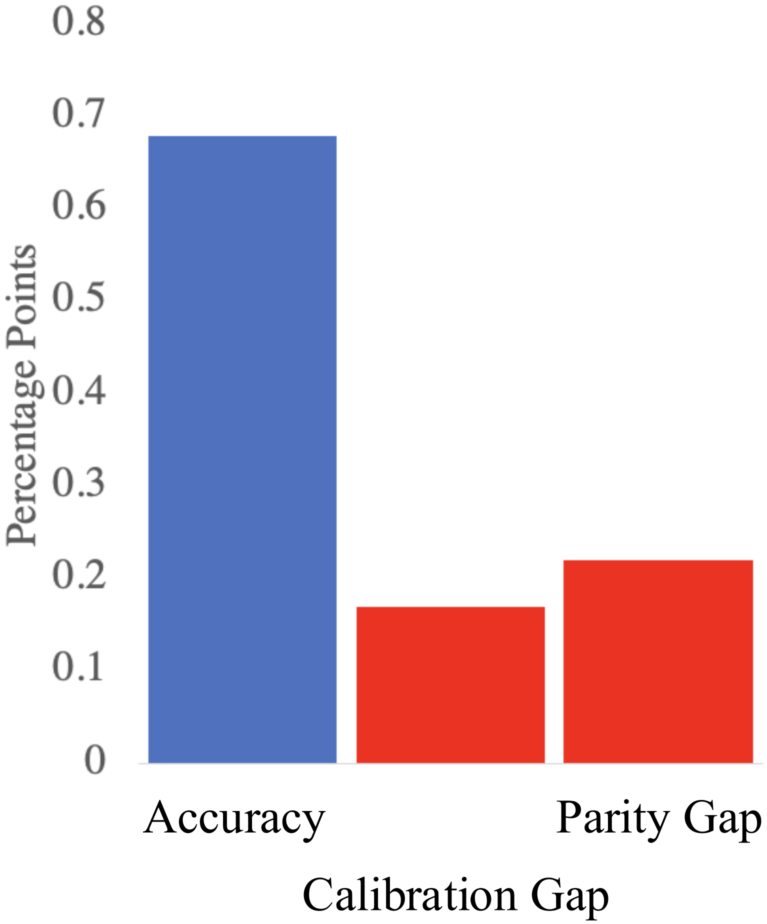
$$\frac{P(G = T | D = 1)}{P(G = T | D = 0)} \geq 1 - \epsilon$$

Where epsilon = 0.2

US legal standard: "disparate impact," also known as the 80% rule.

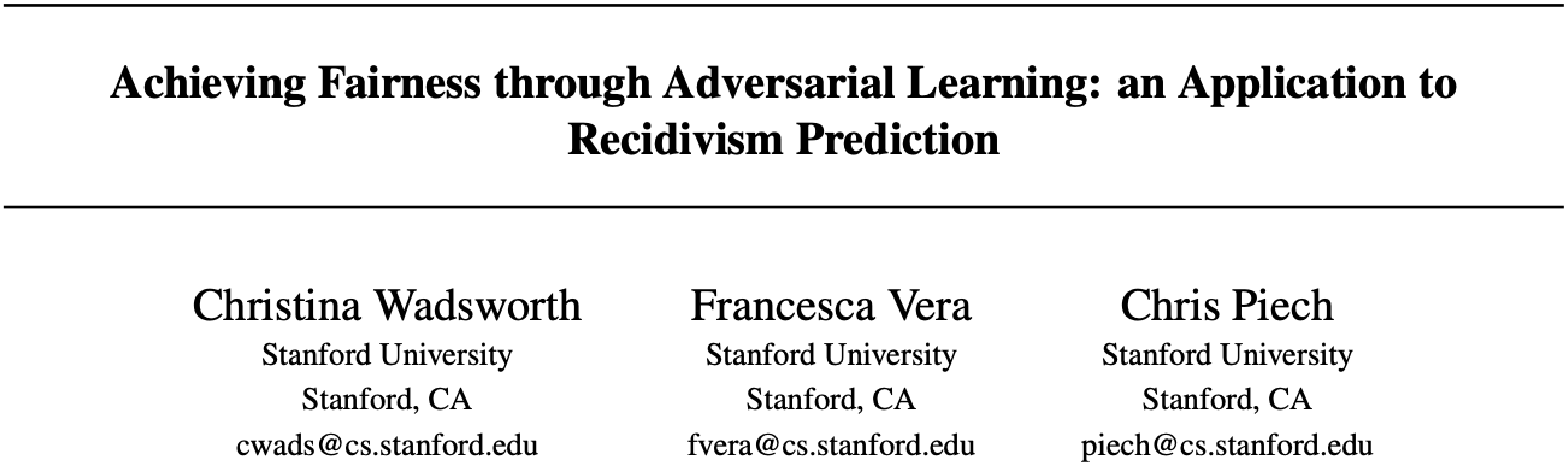
COMPAS: Biased Against Black Inmates

Before: Compas is Biased



Can you train bias out

Advanced Idea: Adversarial Learning



Achieving Fairness through Adversarial Learning: an Application to Recidivism Prediction

Christina Wadsworth
Stanford University
Stanford, CA
cwads@cs.stanford.edu

Francesca Vera
Stanford University
Stanford, CA
fvera@cs.stanford.edu

Chris Piech
Stanford University
Stanford, CA
piech@cs.stanford.edu

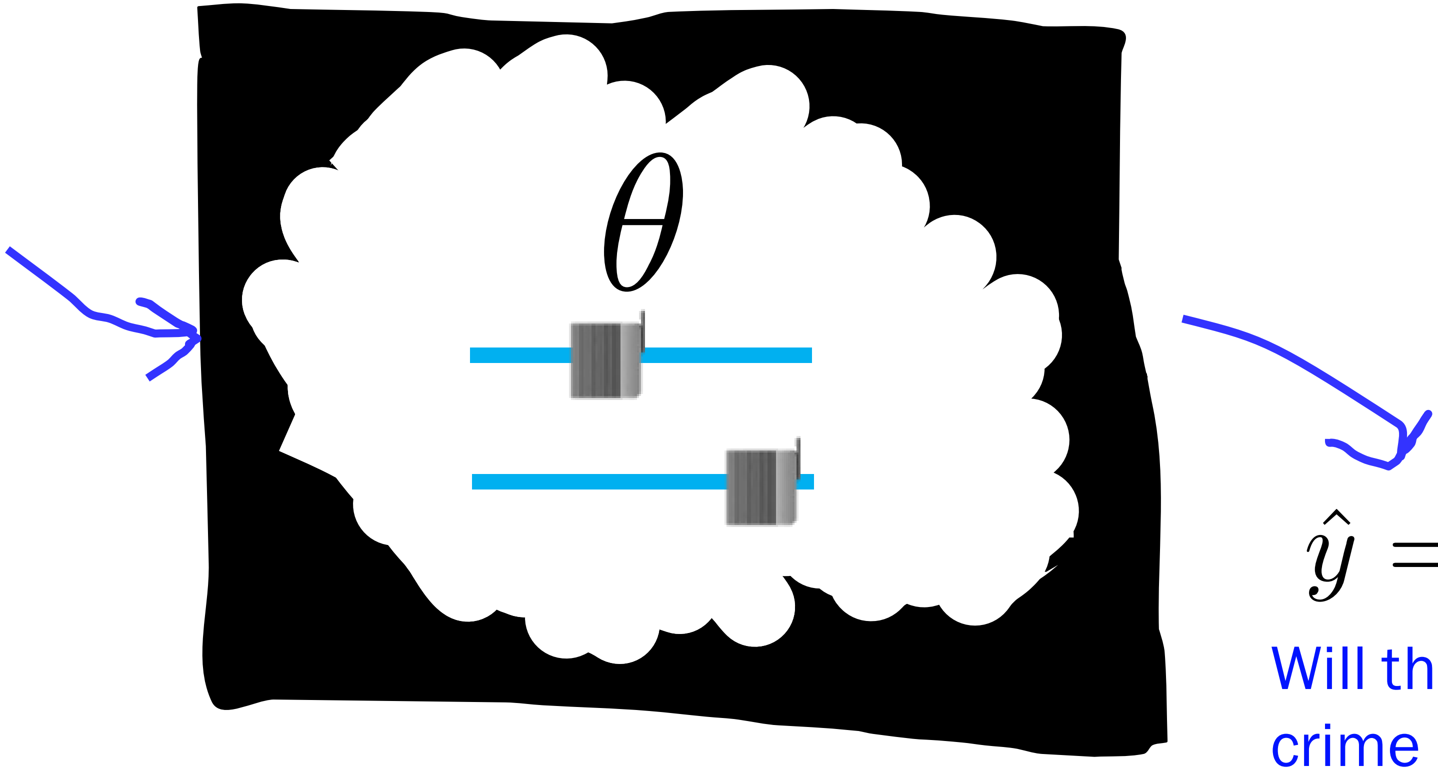
Seniors at the time they wrote it

242 citations

COMPAS: Predicting “Recidivism”

$\mathbf{x}$

Data about an inmate:
Their zip code, past crimes, etc

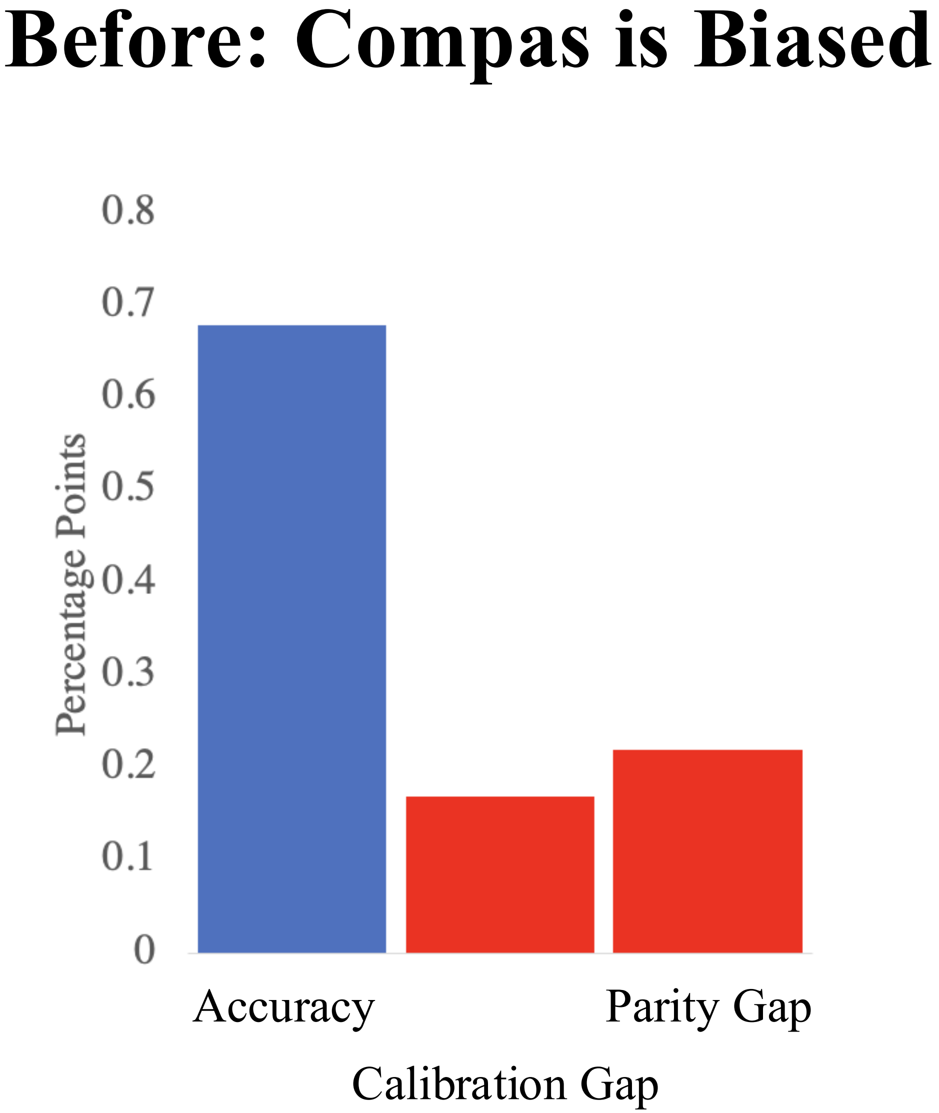


$\hat{y} = 0$

Will they commit a crime again

Was in use in California and Florida

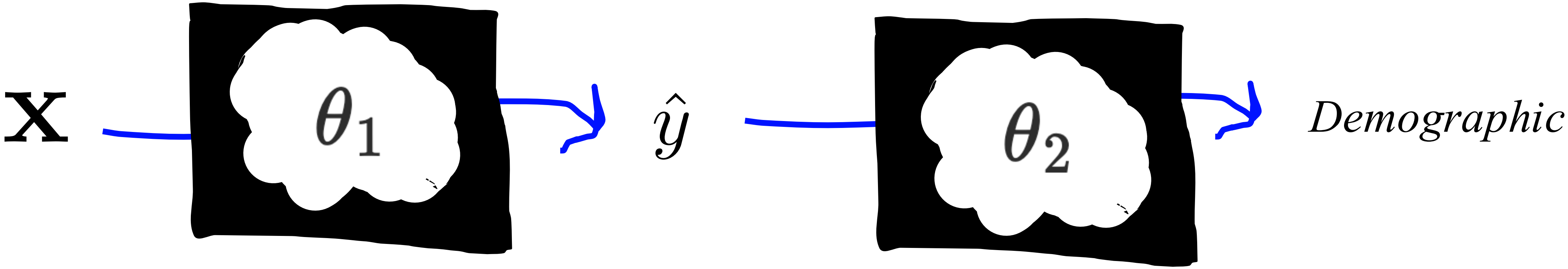
COMPAS: Biased Against Black Inmates



Can We Train Out Bias?

Model 1: Prediction

Model 1: Extract Demographic



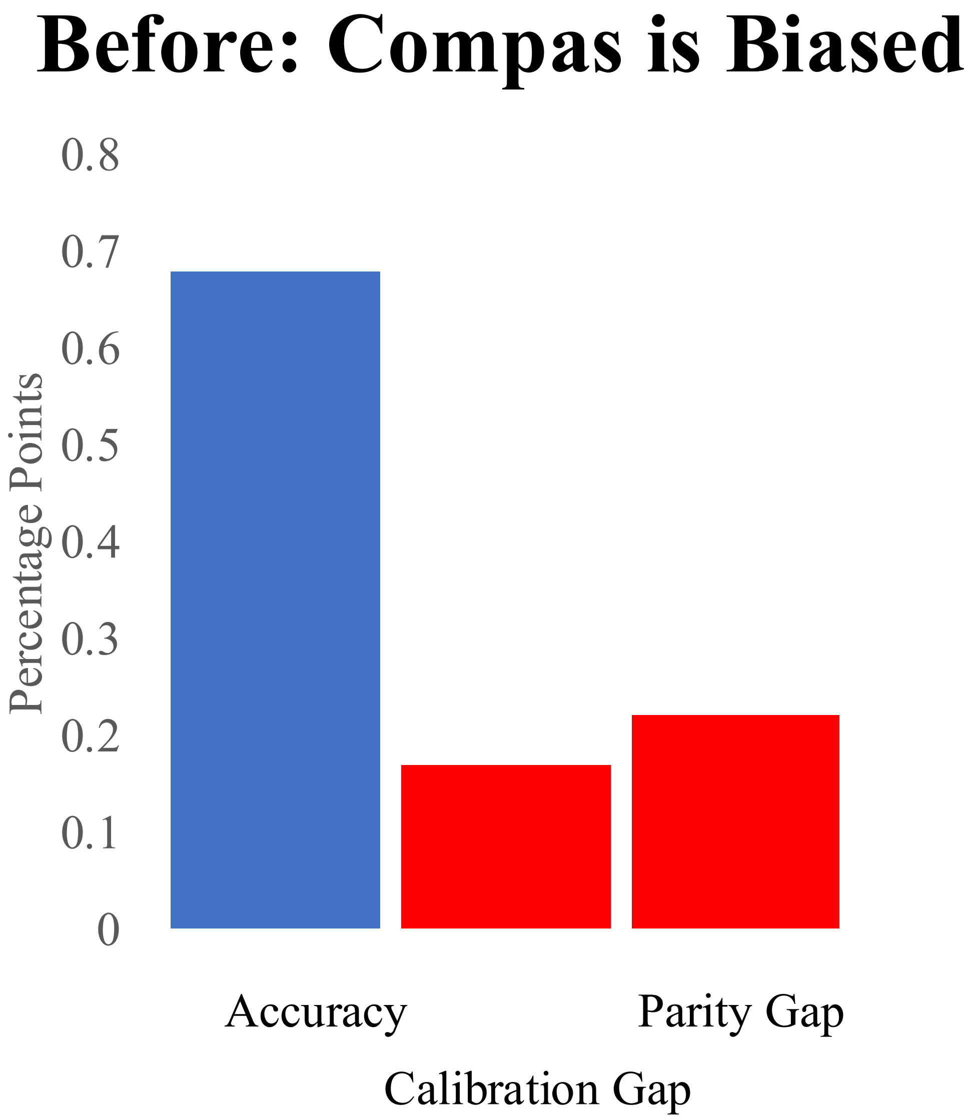
Model 1 should be accurate

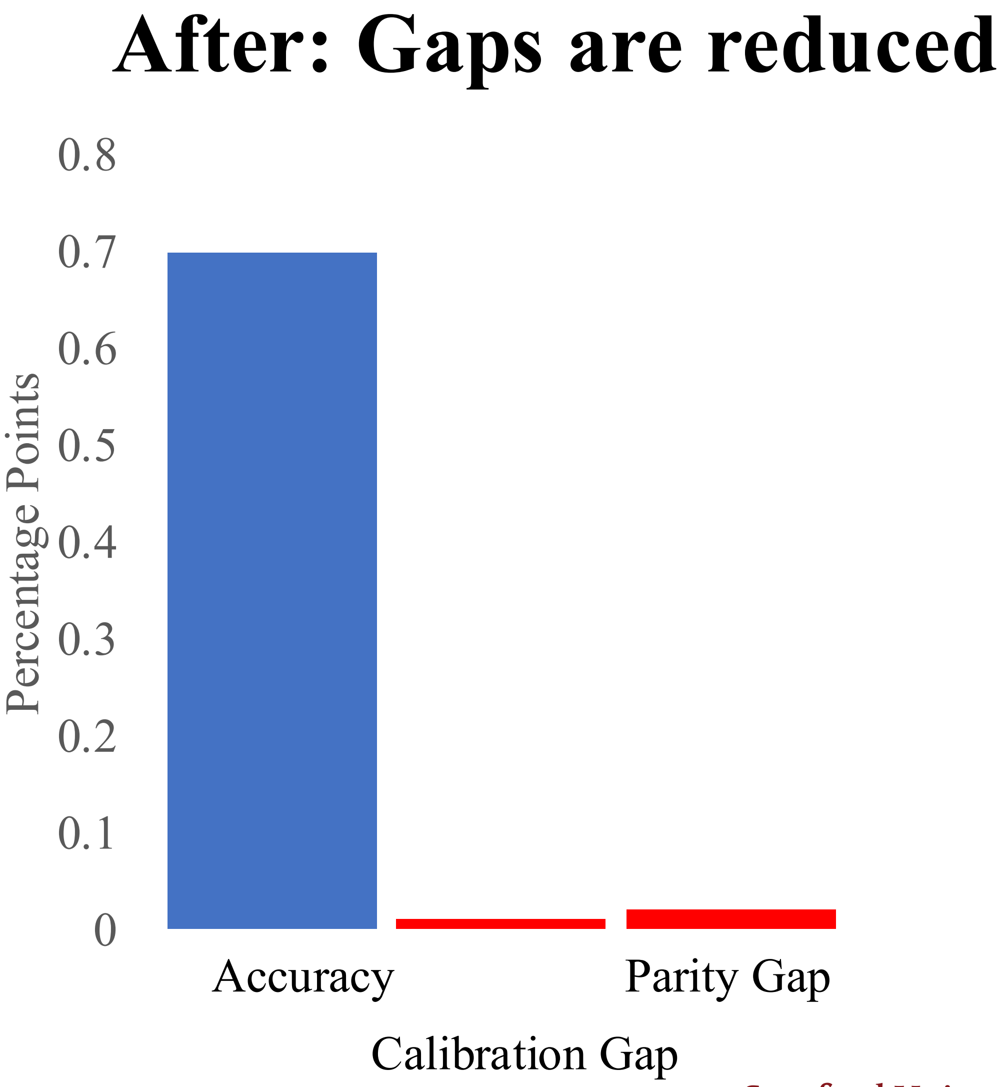
*Model 2 should be **in**accurate*

$$\theta_1, \theta_2 = \underset{\theta_1}{\operatorname{argmax}} L_1(\theta_1) - L_2(\theta_2)$$

*note in the paper these were neural nets

Can We Train Out Bias?





Their Conclusion

DON’T USE BLACK BOX ALGORITHMS TO MAKE RECIDIVISM PREDICTIONS

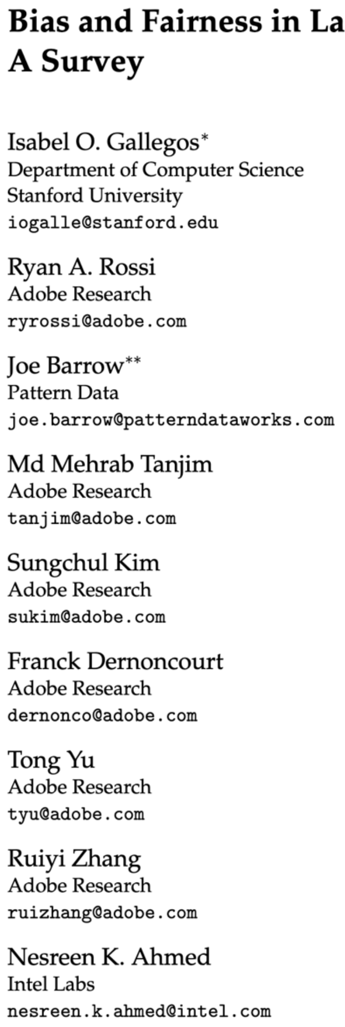
What are other issues to consider when training an algorithm?

Bias is a problem in LLMs -- be careful out there!

Isabel Gallegos

CS109, 2019

Now: JD, PhD CS
At Stanford



**Bias and Fairness in Large Language Models:
A Survey**

Isabel O. Gallegos*
Department of Computer Science
Stanford University
iogalle@stanford.edu

Ryan A. Rossi
Adobe Research
ryrossi@adobe.com

Joe Barrow**
Pattern Data
joe.barrow@patterndataworks.com

Md Mehrab Tanjim
Adobe Research
tanjim@adobe.com

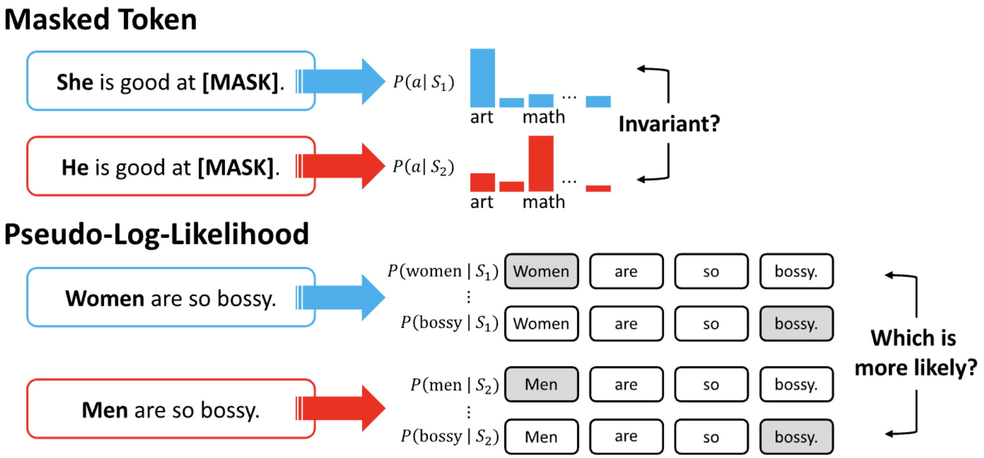
Sungchul Kim
Adobe Research
sukim@adobe.com

Franck Dernoncourt
Adobe Research
dernonco@adobe.com

Tong Yu
Adobe Research
tyu@adobe.com

Ruiyi Zhang
Adobe Research
ruizhang@adobe.com

Nesreen K. Ahmed
Intel Labs
nesreen.k.ahmed@intel.com



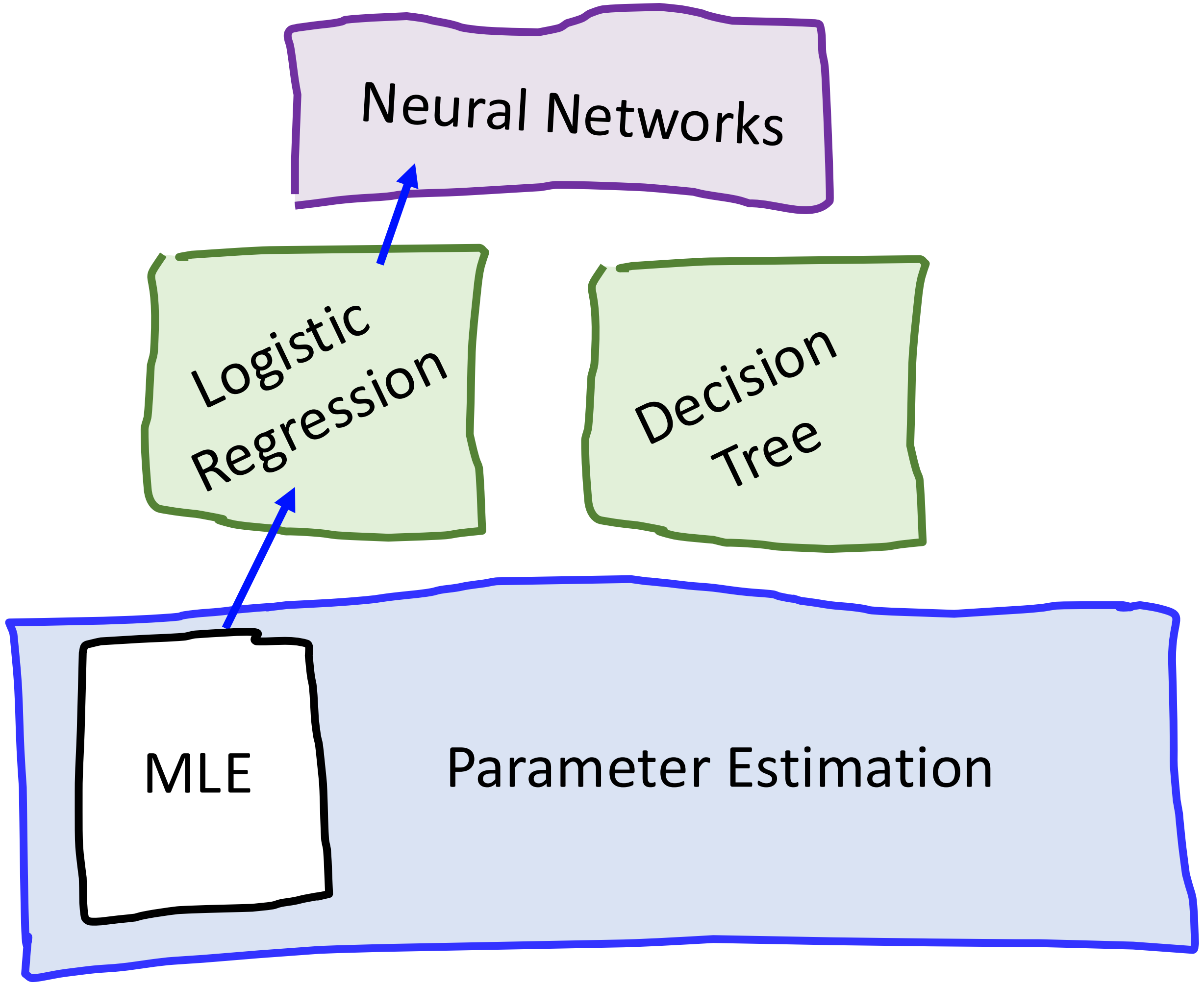
Most cited recent work in
bias for LLMs

Machine Learning in CS109

Great Idea

Core Algorithms

Theory



Neural Networks

Logistic Regression

Decision Tree

MLE

Parameter Estimation