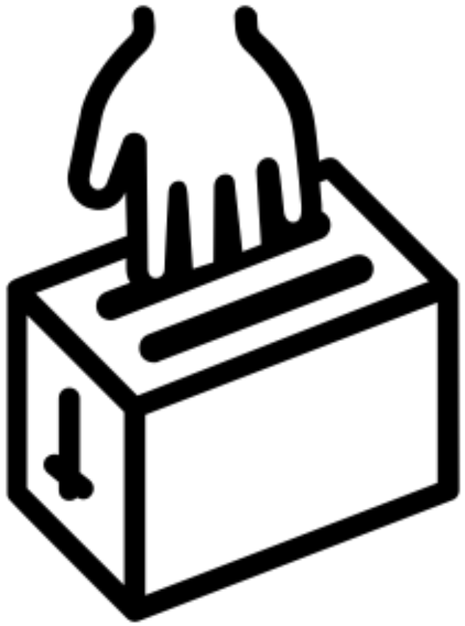
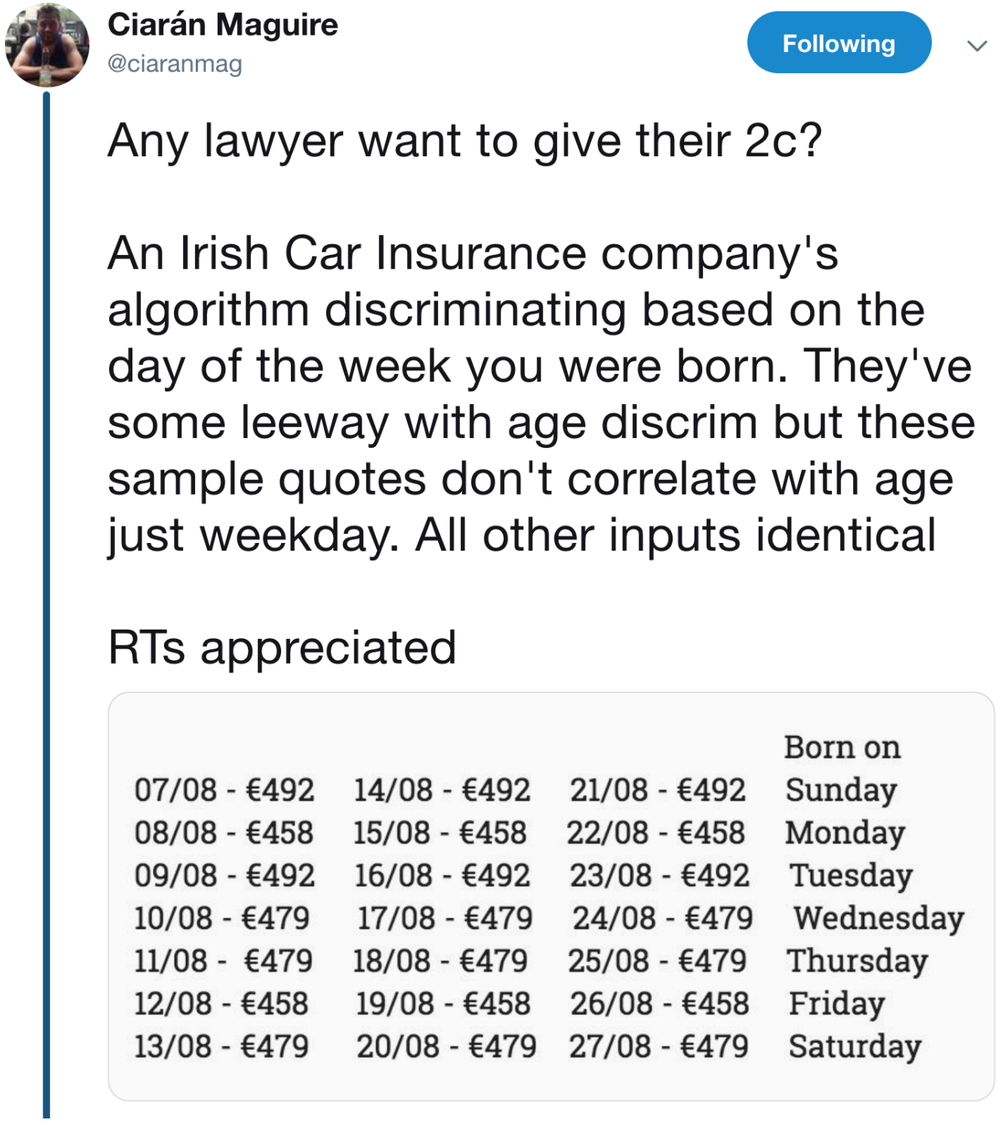
Constraining Artificial Stupidity



Attempting to prevent "DeepOops" at scale.

Artificial Intelligence is Amazing.

AI is Amazing.



Ciarán Maguire
@ciaranmag
Following

Any lawyer want to give their 2c?

An Irish Car Insurance company's algorithm discriminating based on the day of the week you were born. They've some leeway with age discrim but these sample quotes don't correlate with age just weekday. All other inputs identical

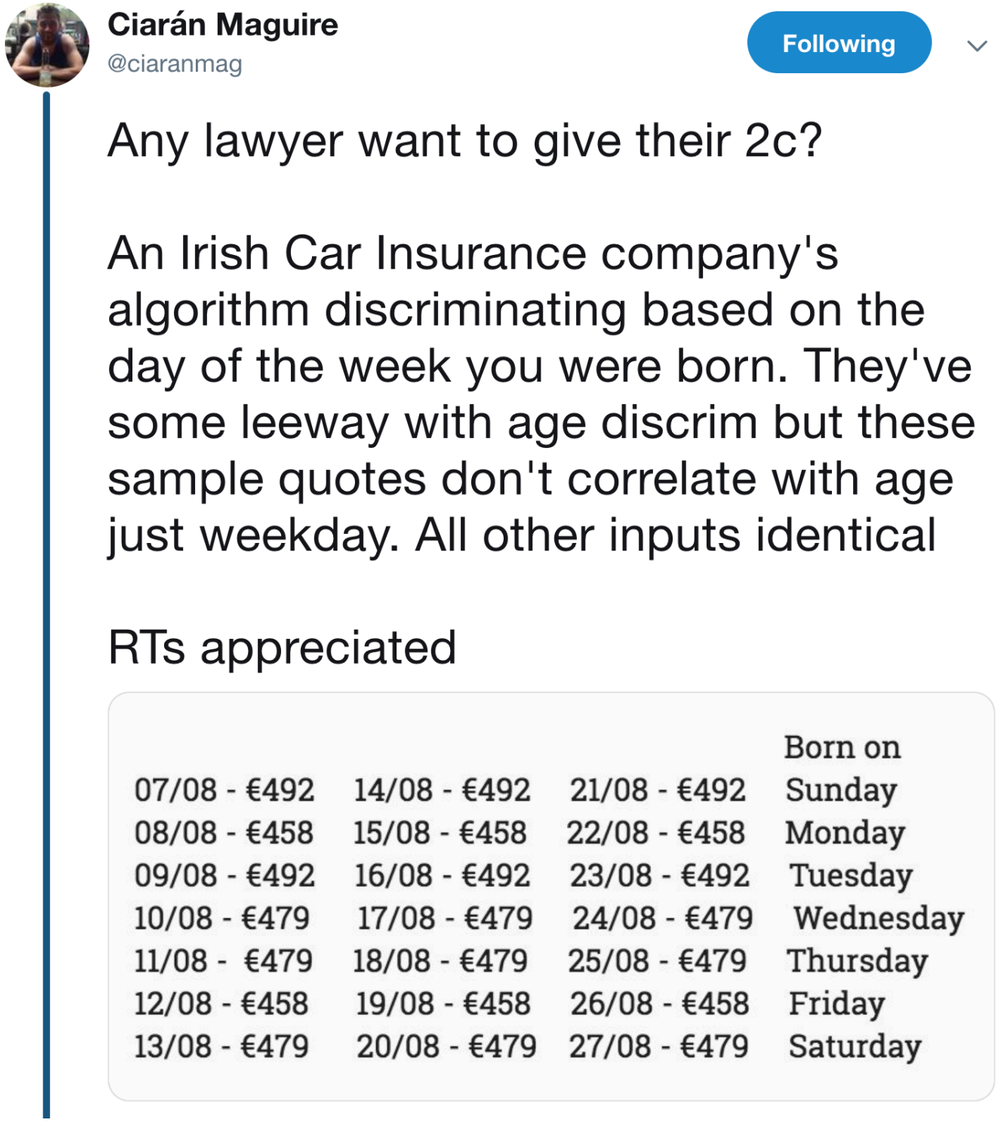
RTs appreciated

			Born on
07/08 - €492	14/08 - €492	21/08 - €492	Sunday
08/08 - €458	15/08 - €458	22/08 - €458	Monday
09/08 - €492	16/08 - €492	23/08 - €492	Tuesday
10/08 - €479	17/08 - €479	24/08 - €479	Wednesday
11/08 - €479	18/08 - €479	25/08 - €479	Thursday
12/08 - €458	19/08 - €458	26/08 - €458	Friday
13/08 - €479	20/08 - €479	27/08 - €479	Saturday

But not always in a good way.

AI can have an aura of authority that people put blind faith in and this leads to dangerous situations.

AI is Amazing.

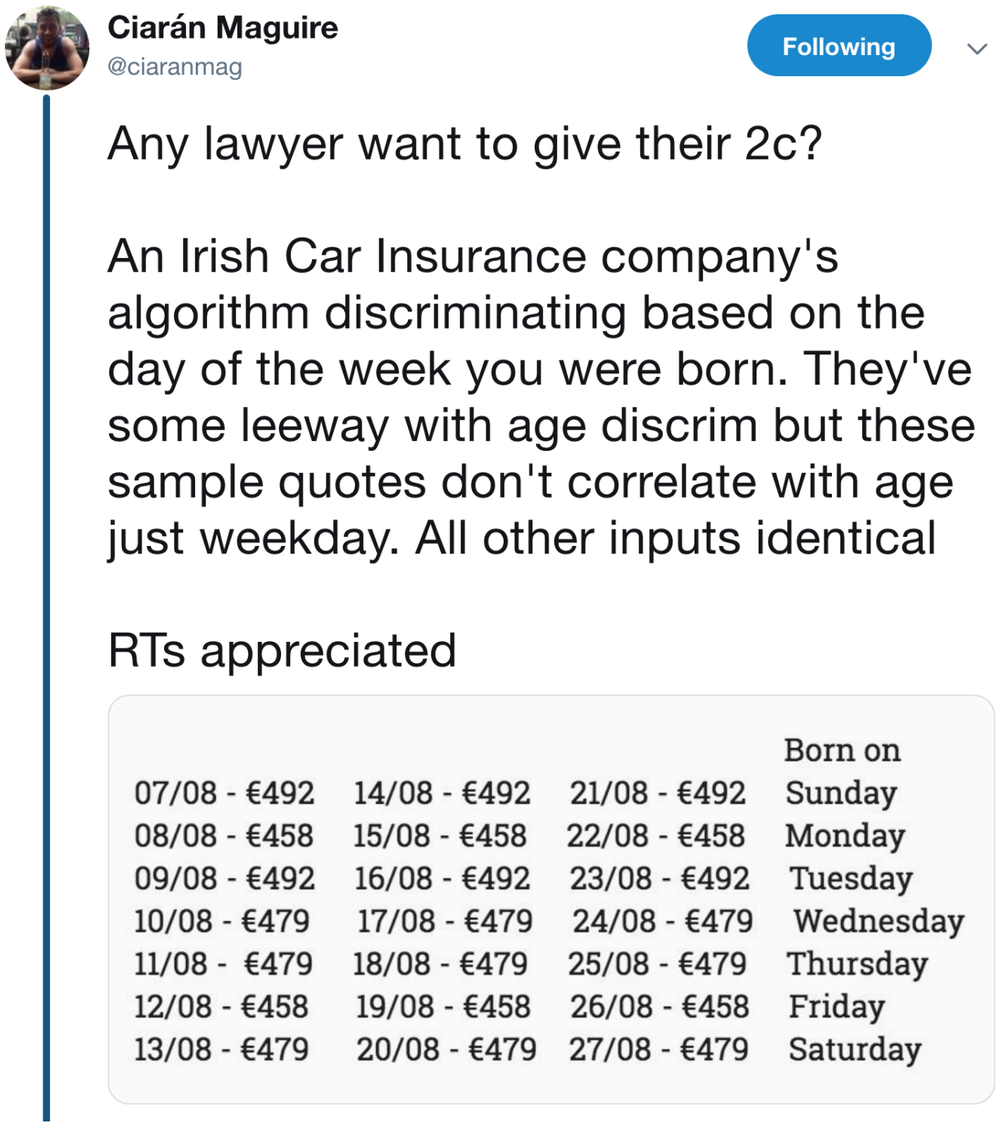


But not always in a good way.

AI can have an aura of authority that people put blind faith in and this leads to dangerous situations.

DeepOops[tm]

AI is Amazing.

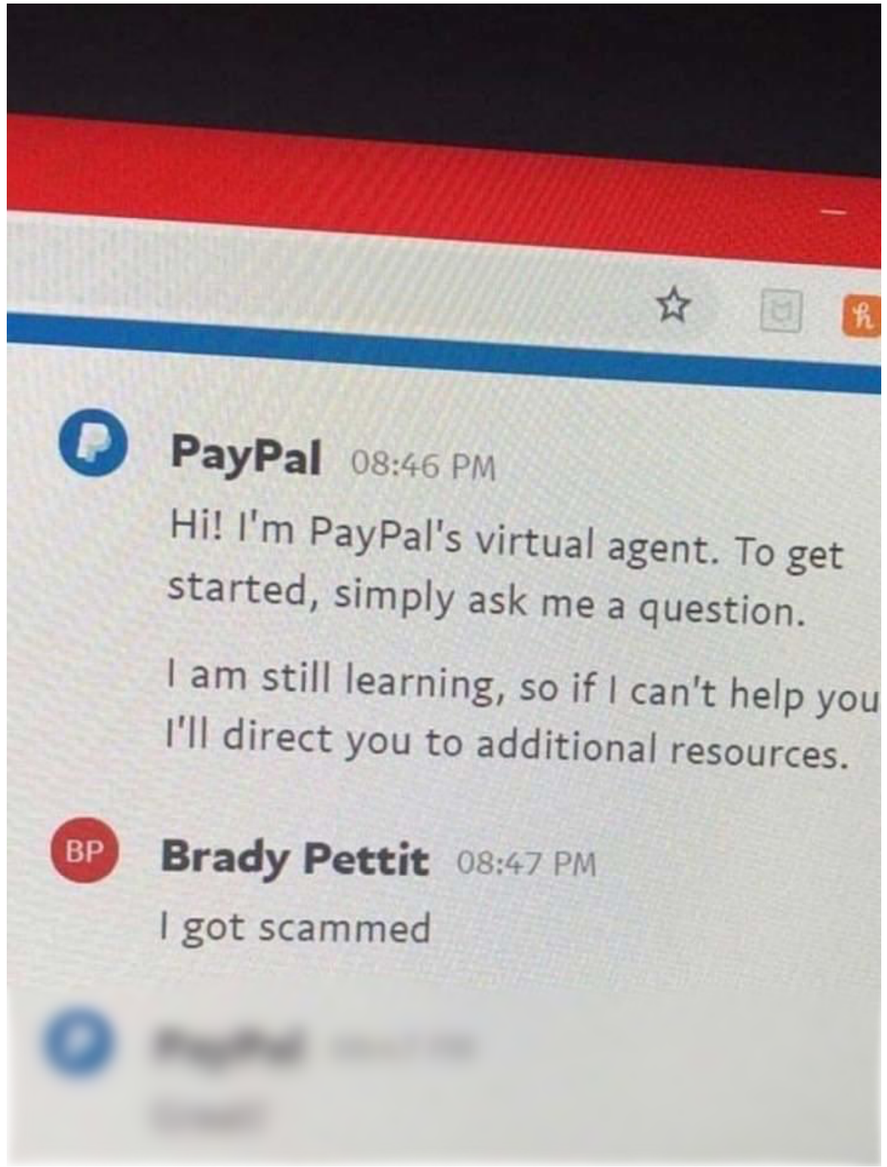


But not always in a good way.

AI can have an aura of authority that people put blind faith in and this leads to dangerous situations.

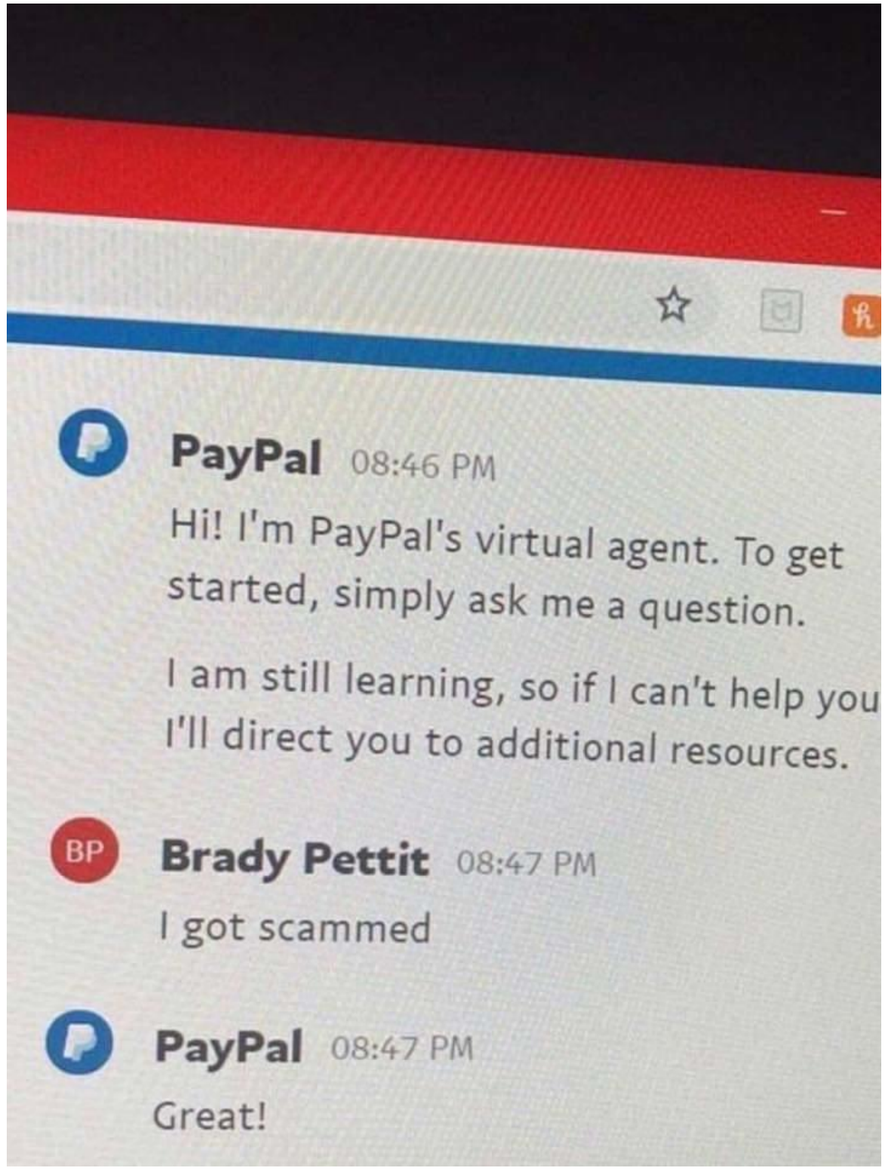
DeepOops[tm]: at Scale

AI is Amazing.



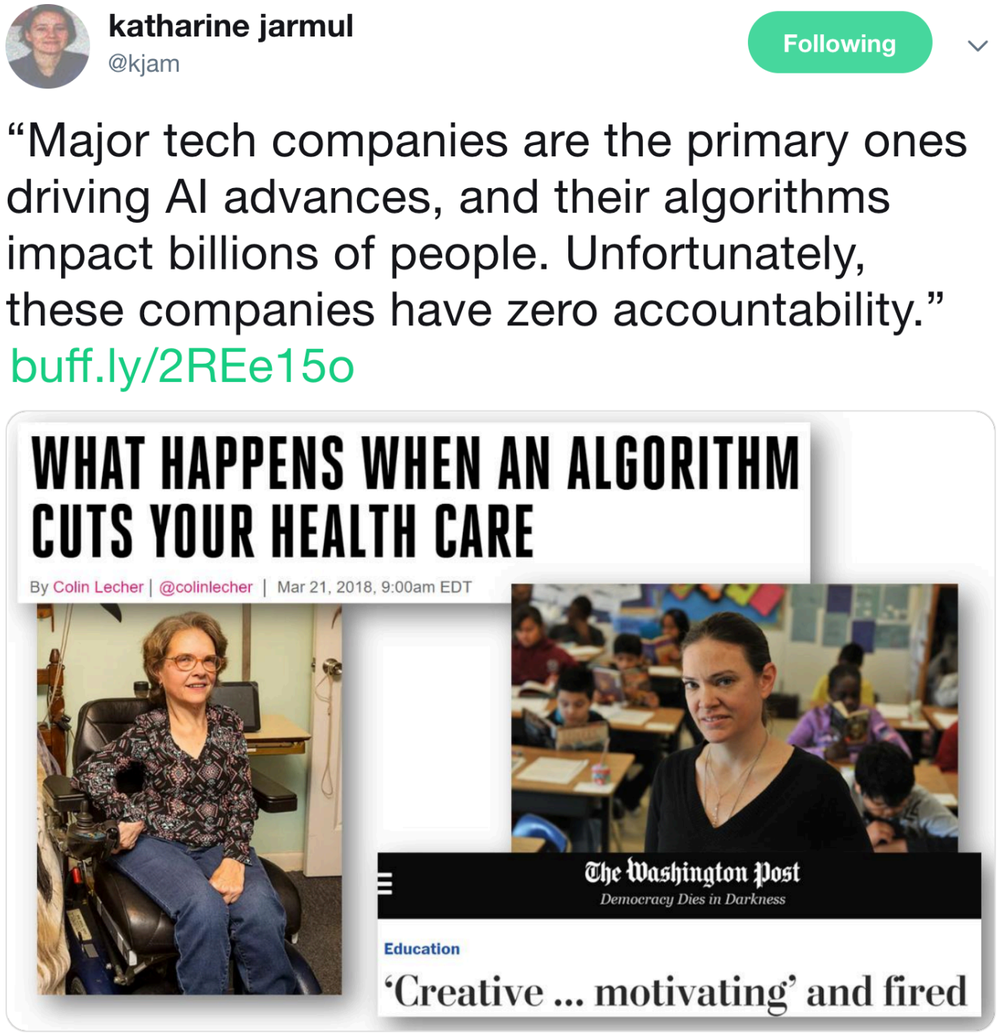
Sometimes it makes me giggle.

AI is Amazing.



Sometimes it makes me giggle.

AI is Amazing.



... but there's also moments where it is getting frightning.

As for the transparency of the system, he agrees that the algorithm is impossible for most to easily understand, but says that it's not a problem. "It's not simple," he says. "My washing machine isn't simple." But if you can capture complexity in more detail, Fries argues, this will ultimately serve the public better, and at some point, "you're going to have to trust me that a bunch of smart people determined this is the smart way to do it."

De Liban started keeping a list of what he thought of as "algorithmic absurdities." One variable in the assessment was foot problems. When an assessor visited a certain person, they wrote that the person didn't have any problems — because they were an amputee. Over time, De Liban says, they discovered wildly different scores when the same people were assessed, despite being in the same condition. (Fries says studies suggest this rarely happens.) De Liban also says negative changes, like a person contracting pneumonia, could counterintuitively lead them to receive fewer help hours because the flowchart-like algorithm would place them in a different category. (Fries denied this, saying the algorithm accounts for it.)

Definition

artificial
made or produced by human beings rather than occurring naturally, especially as a copy of something natural

stupidity
behaviour that shows a lack of good sense or judgement

Design and Fashion Statements

There are usually two things that can go **HorriblyWrong[tm]** with models.

1. Models don't perform well on a metric people like.
2. Models do something that you don't want them to.

My thesis is that the industry is too focussed on the former, we really need to worry about the latter.

Today

Remedies (not Solutions) for Artificial Stupidity

— Fix #1: Predict Less, but Carefully
— Fix #2: Constrain thy Features
— Fix #3: Constrain thy Model
— Fix #4: Modelling & **SomethingFundamental[tm]**
    — Demonstrate **the** unfortunate truth

Constraint #1: Predict Less

Classification models usually estimate a probability value, via `.predict_proba()`, and this proxies forward a predicted class, via `.predict()`.

In a two class case we usually:

$$\text{predict}(\mathbf{x}) = \begin{cases} \text{class}_A & \text{if } p(x) \leq 0.5 \\ \text{class}_B & \text{if } p(x) > 0.5 \end{cases}$$

Constraint #1: Predict Less

Classification models usually estimate a probability value, via `.predict_proba()`, and this proxies forward a predicted class, via `.predict()`.

But what if we did this?

$$\text{predict}(\mathbf{x}) = \begin{cases} \text{class}_A & \text{if } p(x) \leq 0.9 \\ \text{class}_B & \text{if } p(x) > 0.9 \end{cases}$$

Constraint #1: Predict Less

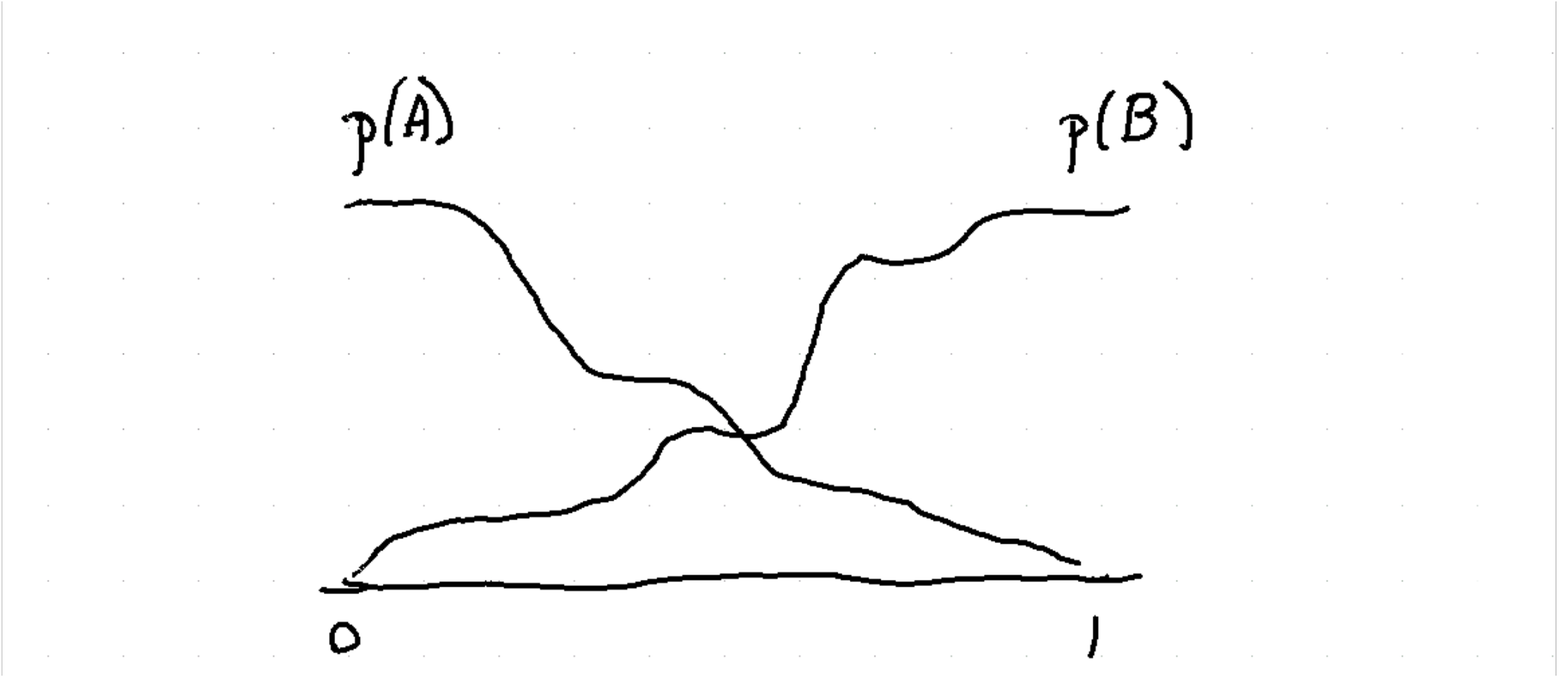
But what if we did this?

$$\text{predict}(\mathbf{x}) = \begin{cases} \text{class}_A & \text{if } p(x) \leq 0.2 \\ \text{class}_B & \text{if } p(x) > 0.8 \\ \text{won't predict} & \text{otherwise} \end{cases}$$

We will add a label: **won't predict**.

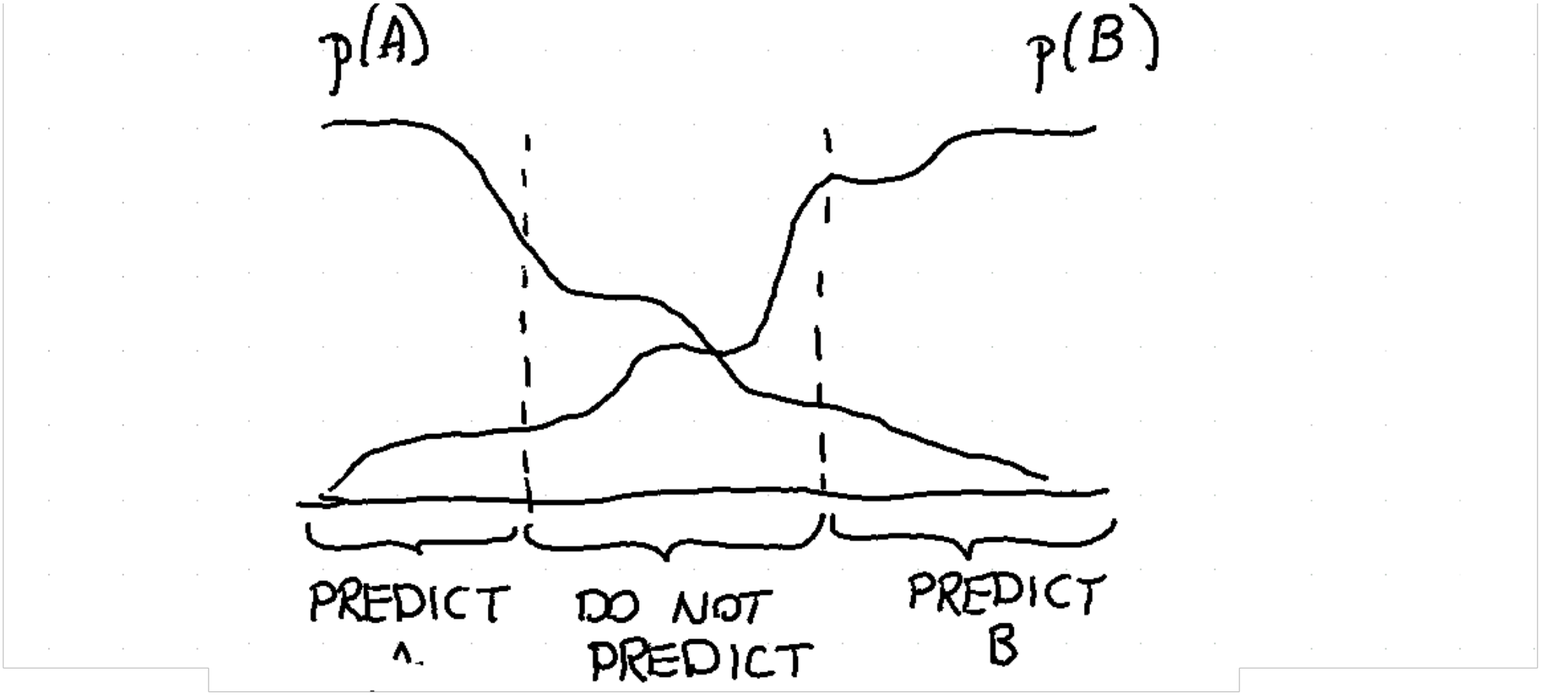
Constraint #1: Predict Less

Usually the odds of model selection have a shape

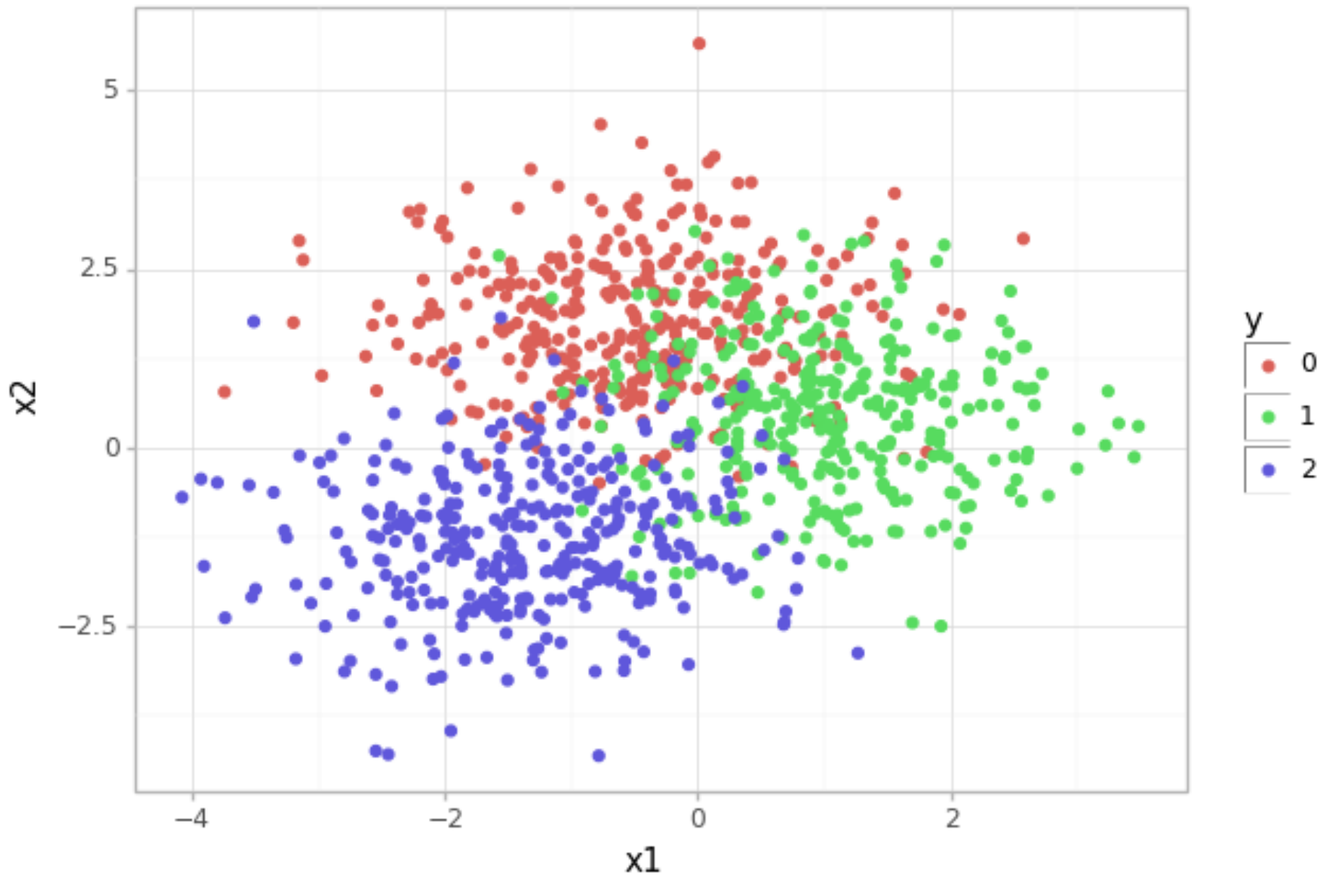


Constraint #1: Predict Less

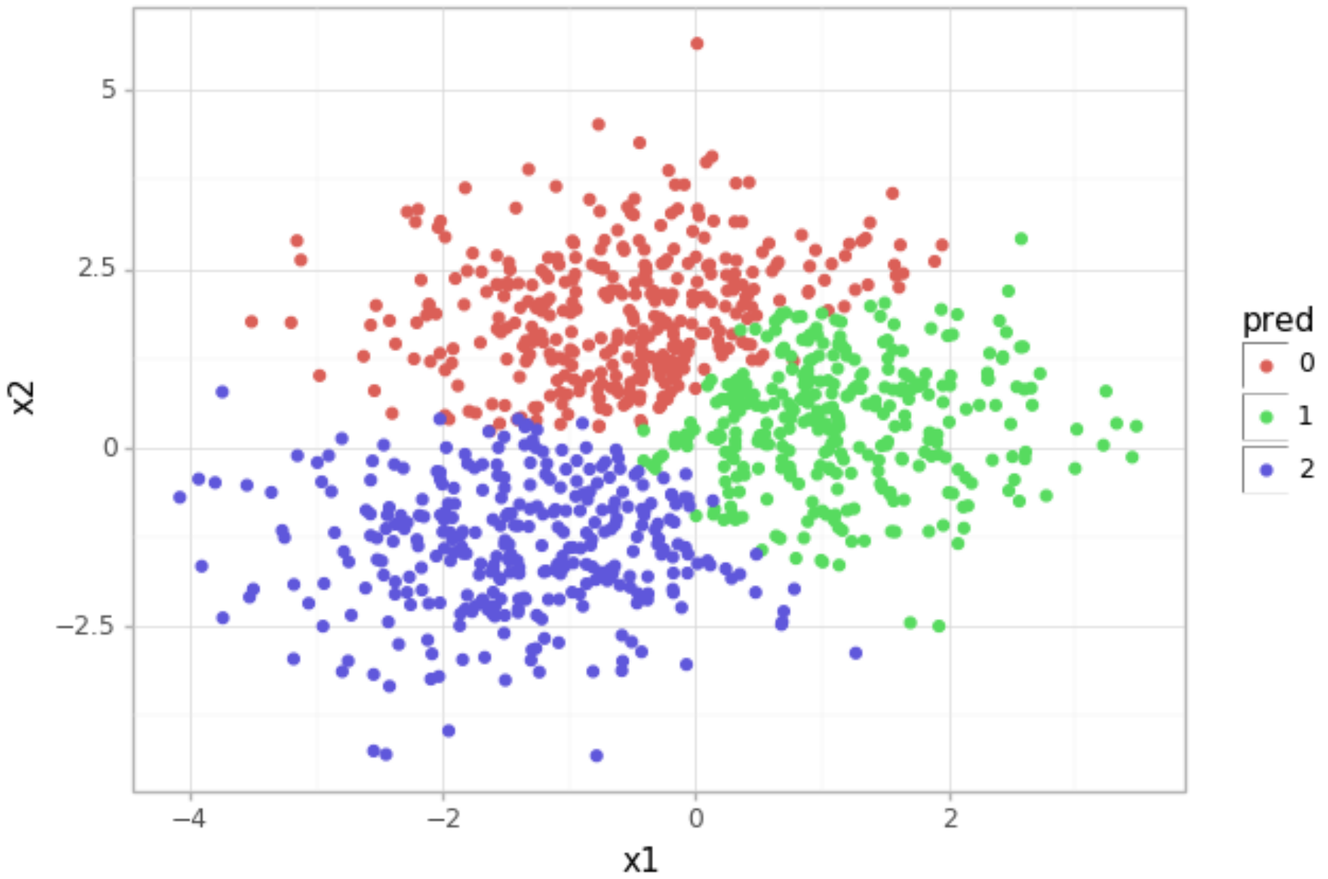
Let's test this idea



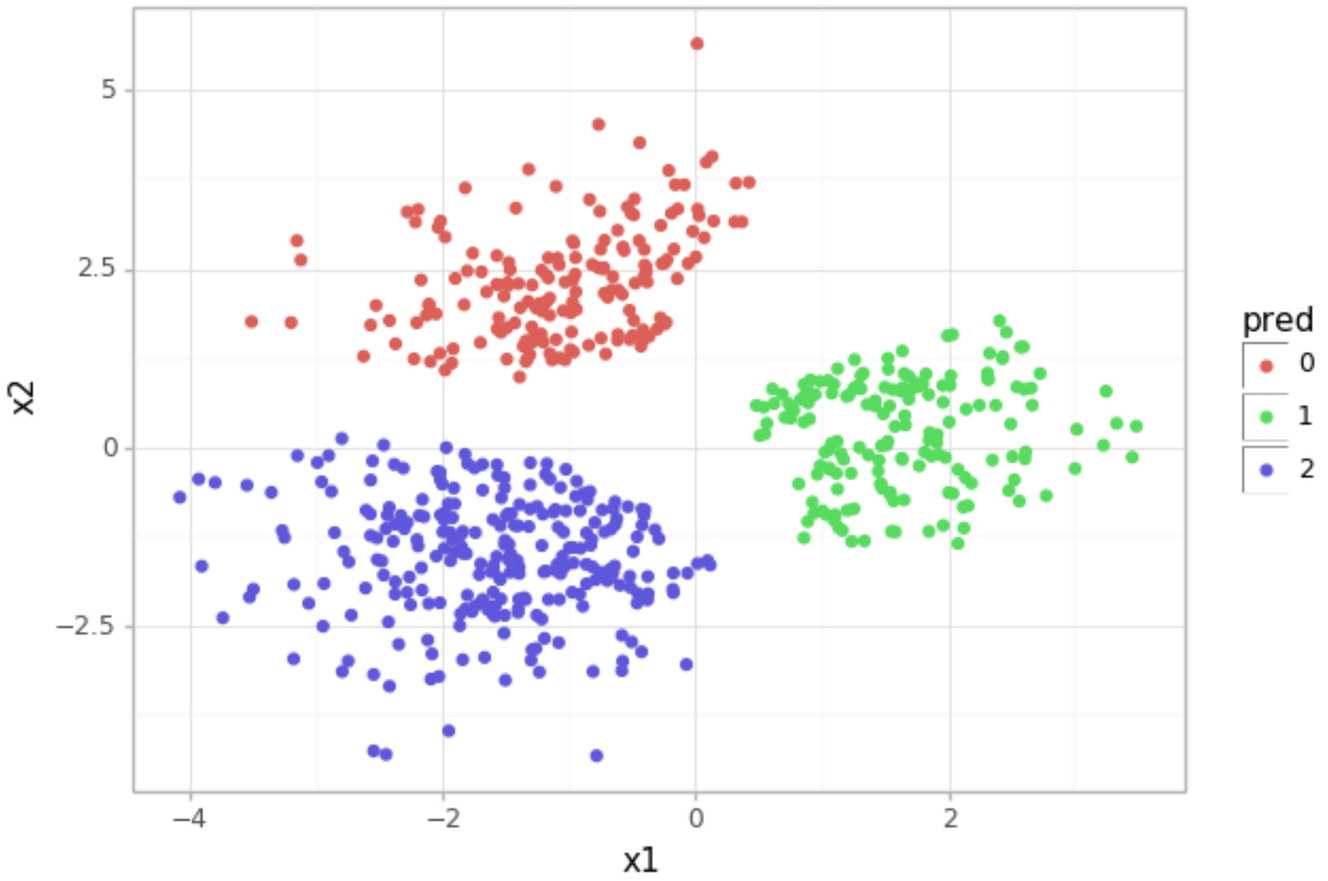
Example Dataset



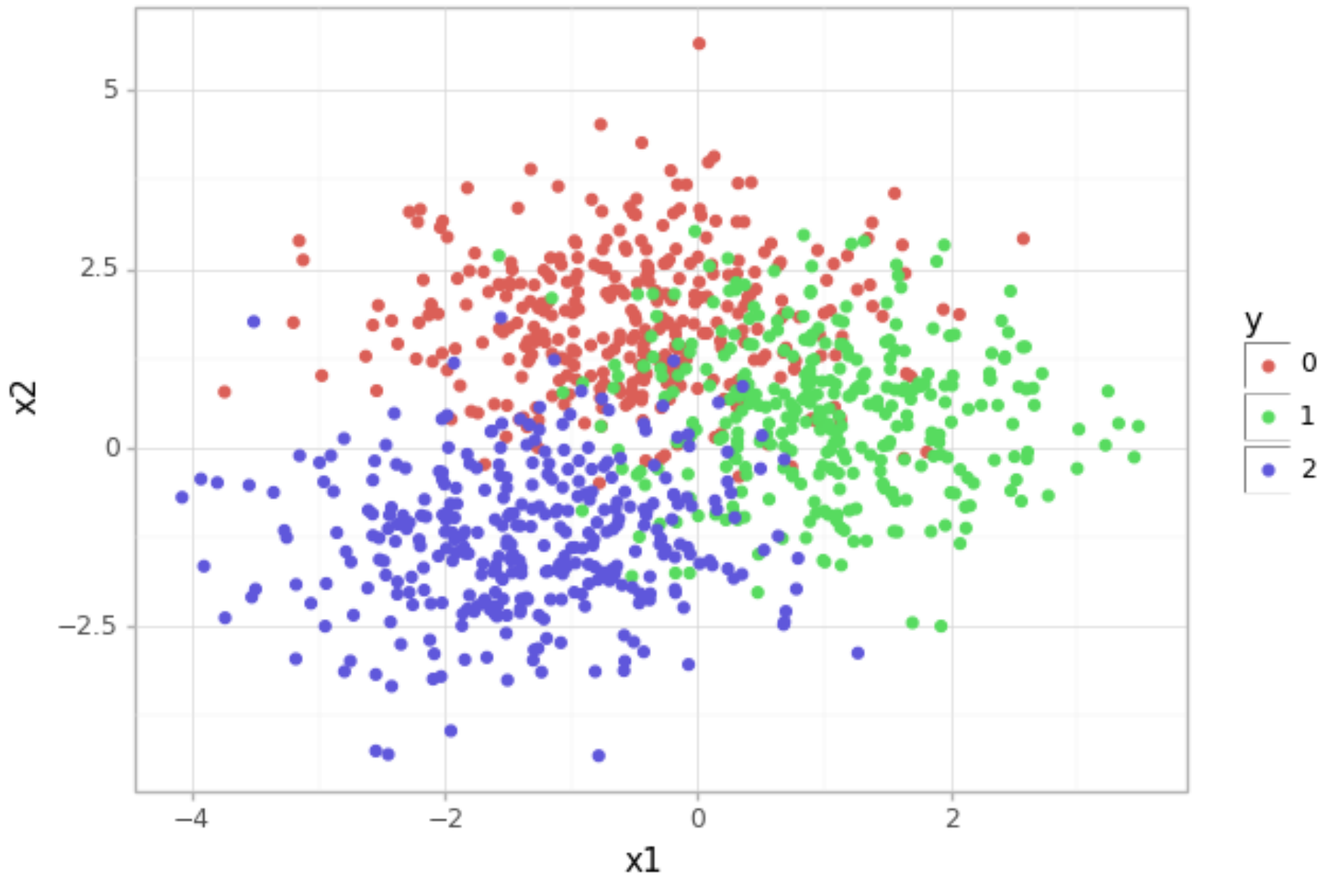
Example Predictions



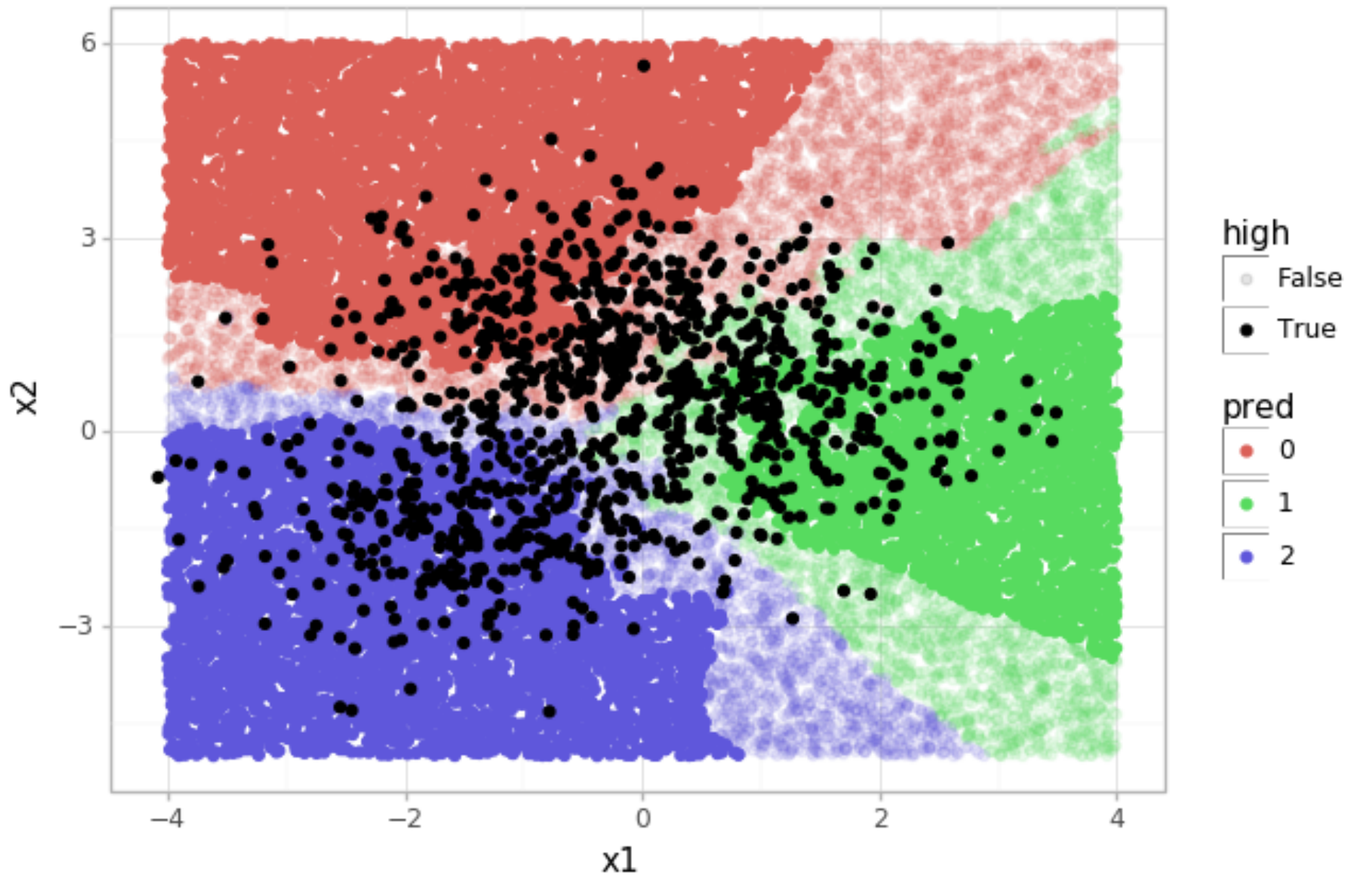
Example Prediction Prob > 0.8



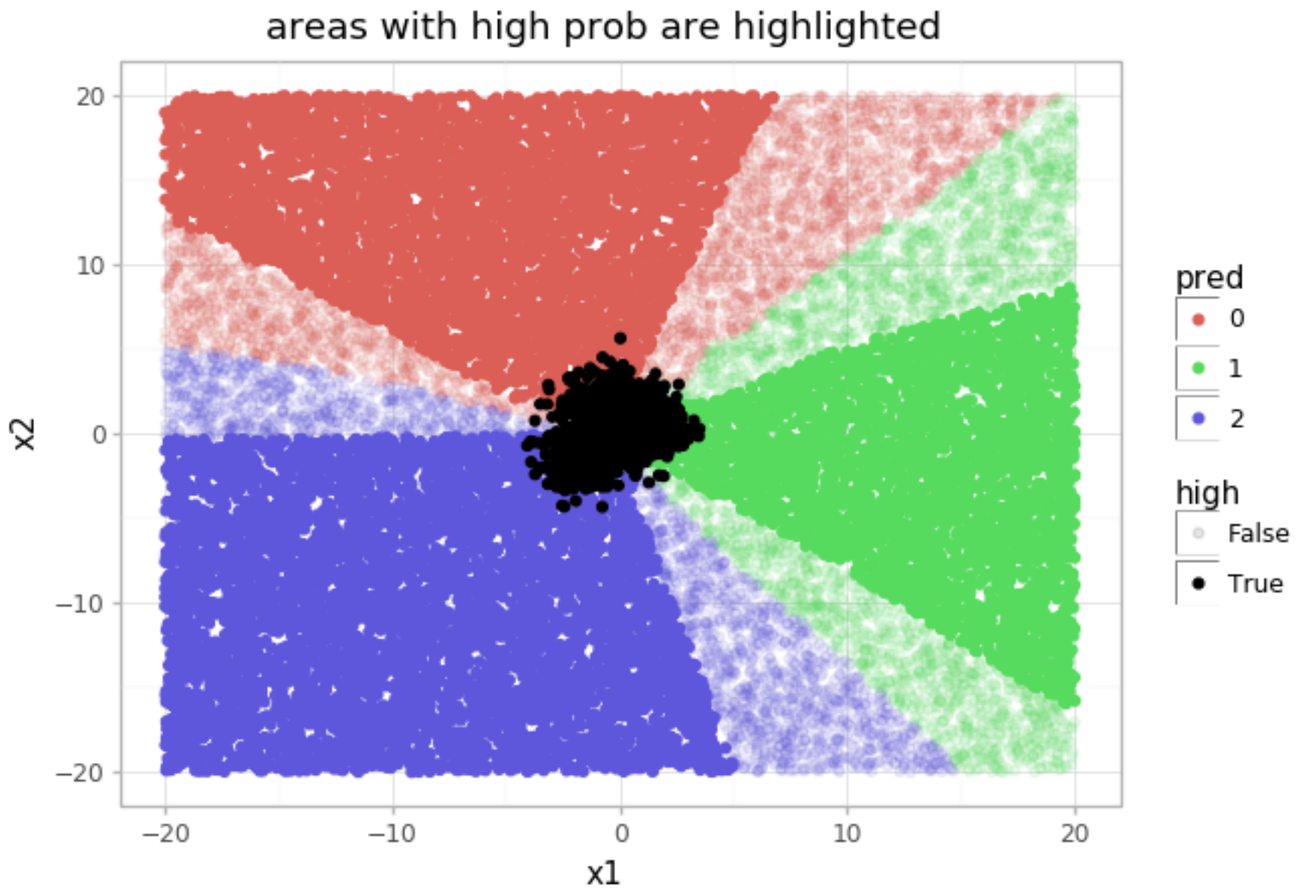
Remember Our Starting Point?

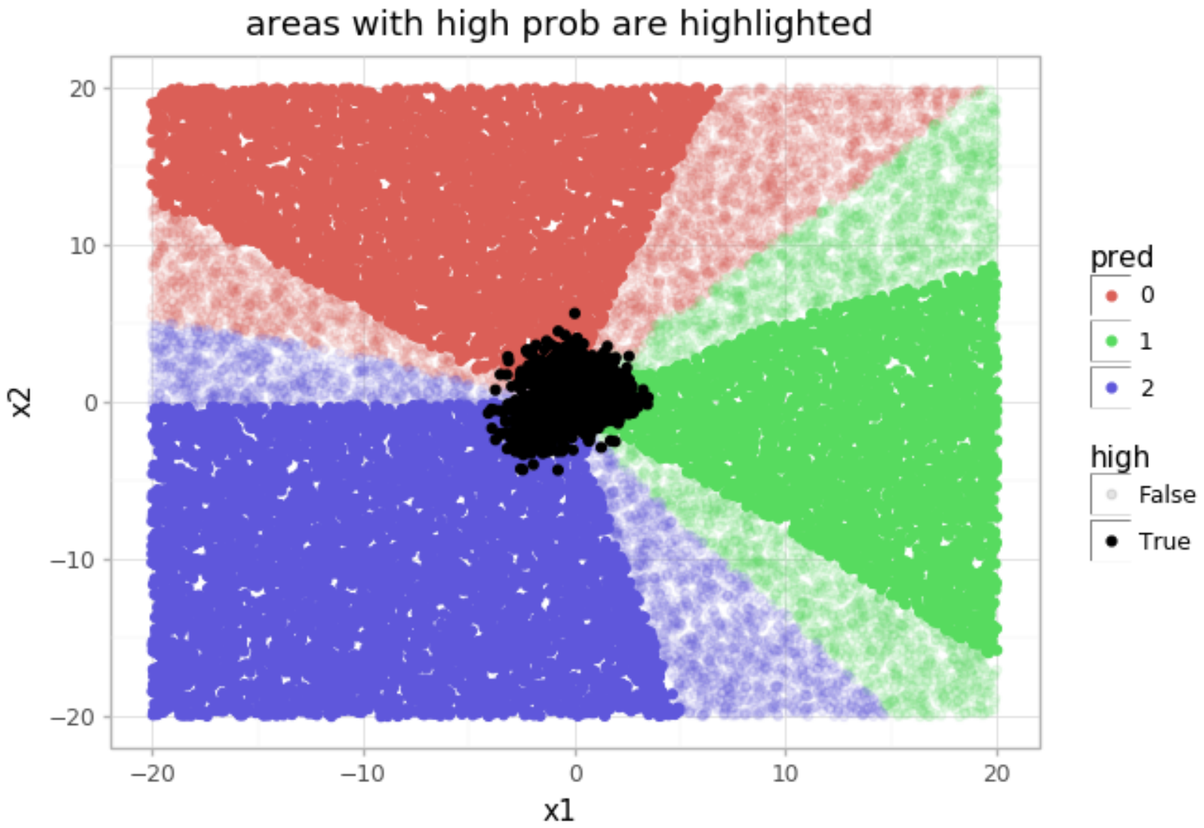


Let us now make predictions!



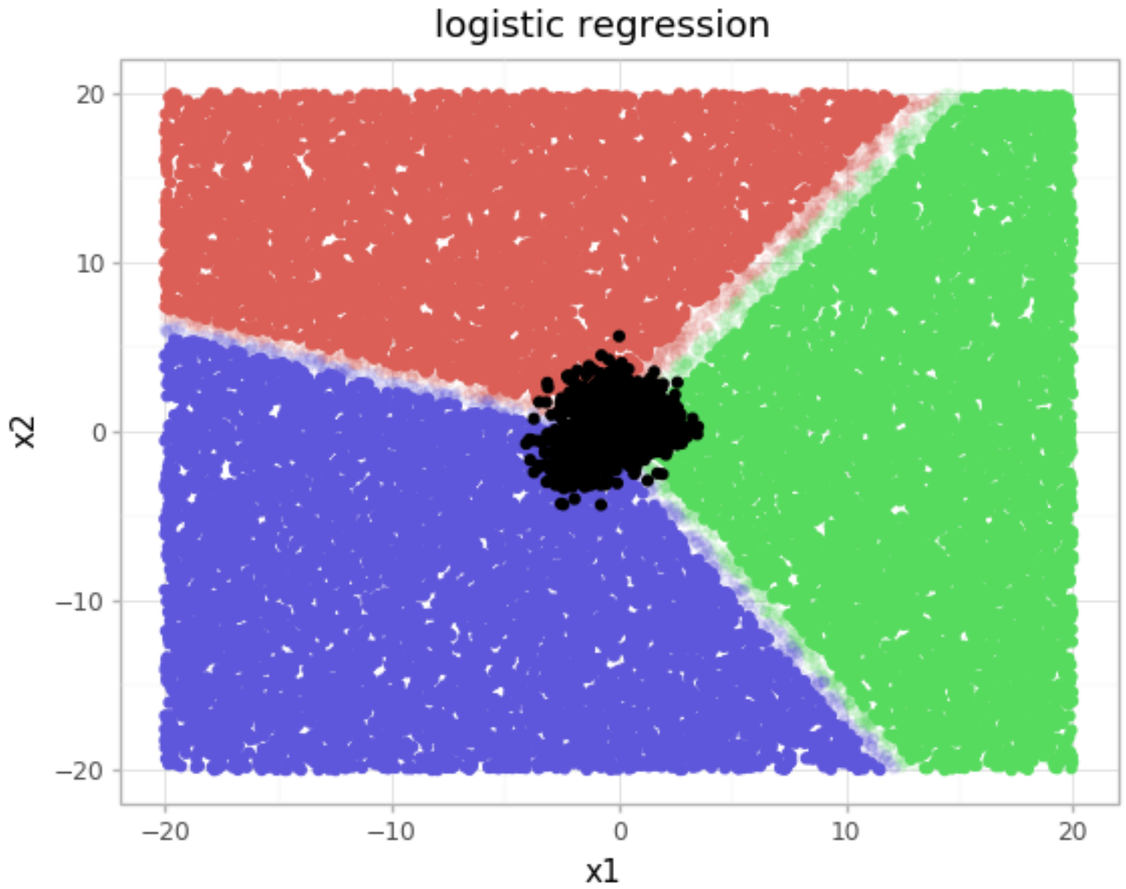
Let us now zoom out!



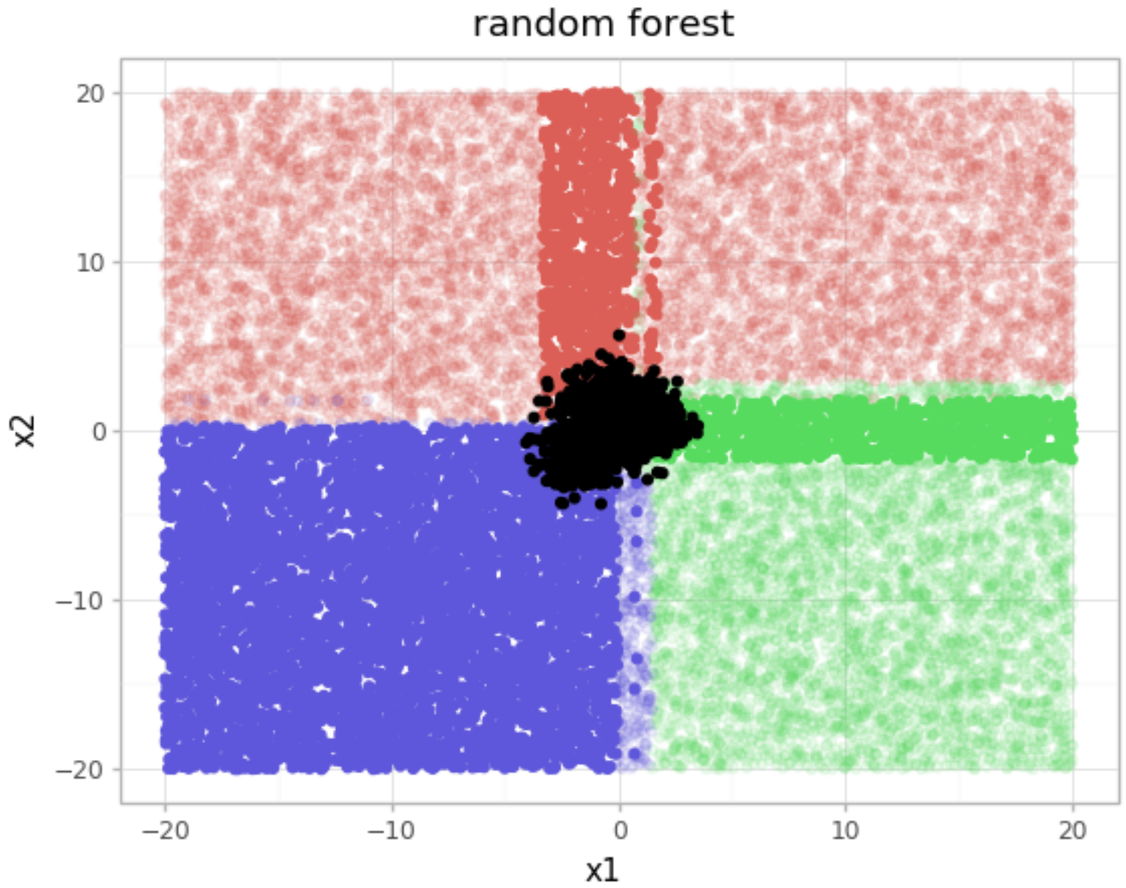


We're assigning super high probabilities to regions where we have seen no data. This feels wrong!

It's not the model: logistic regression



It's not the model: random forest

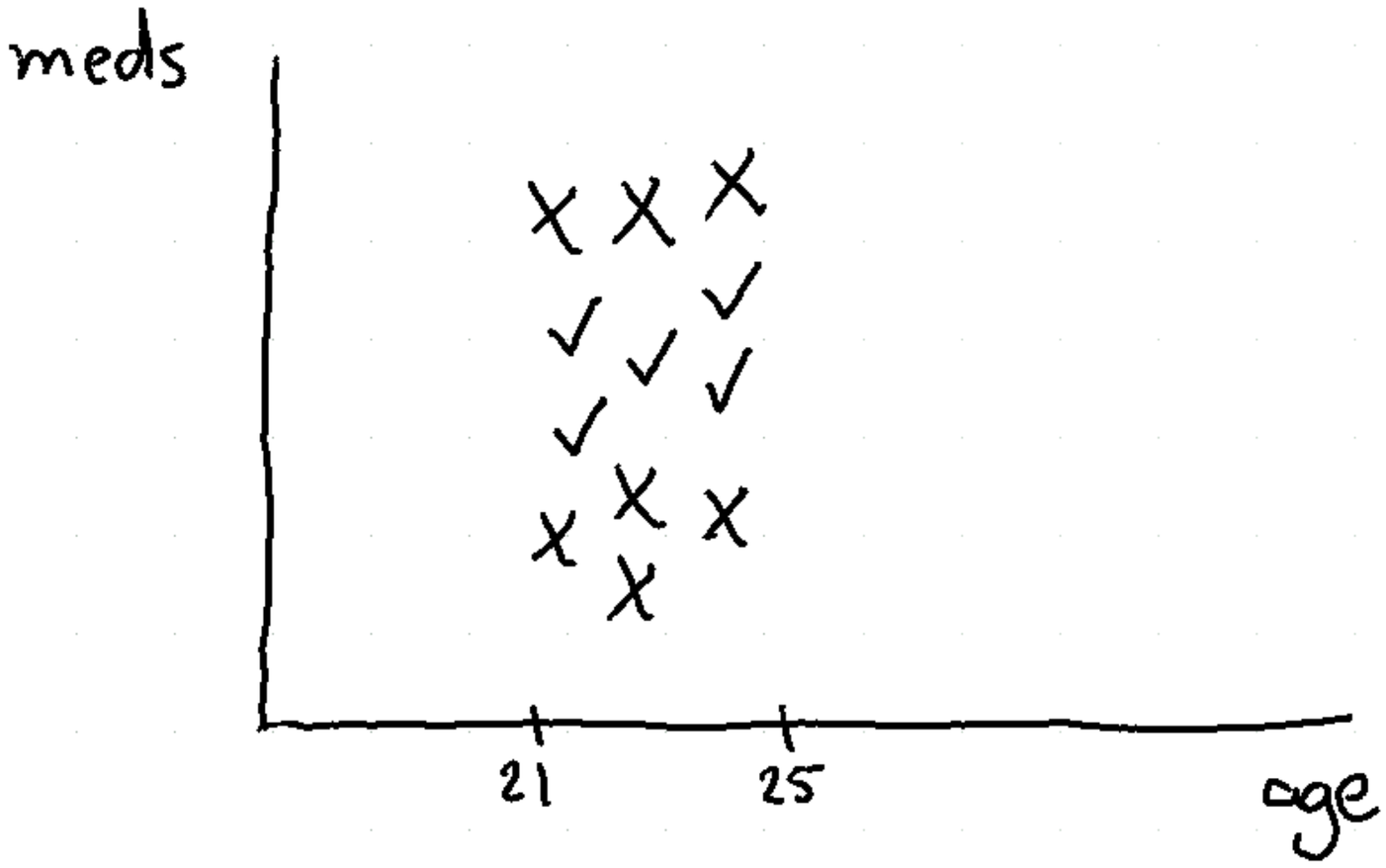


A Common Error

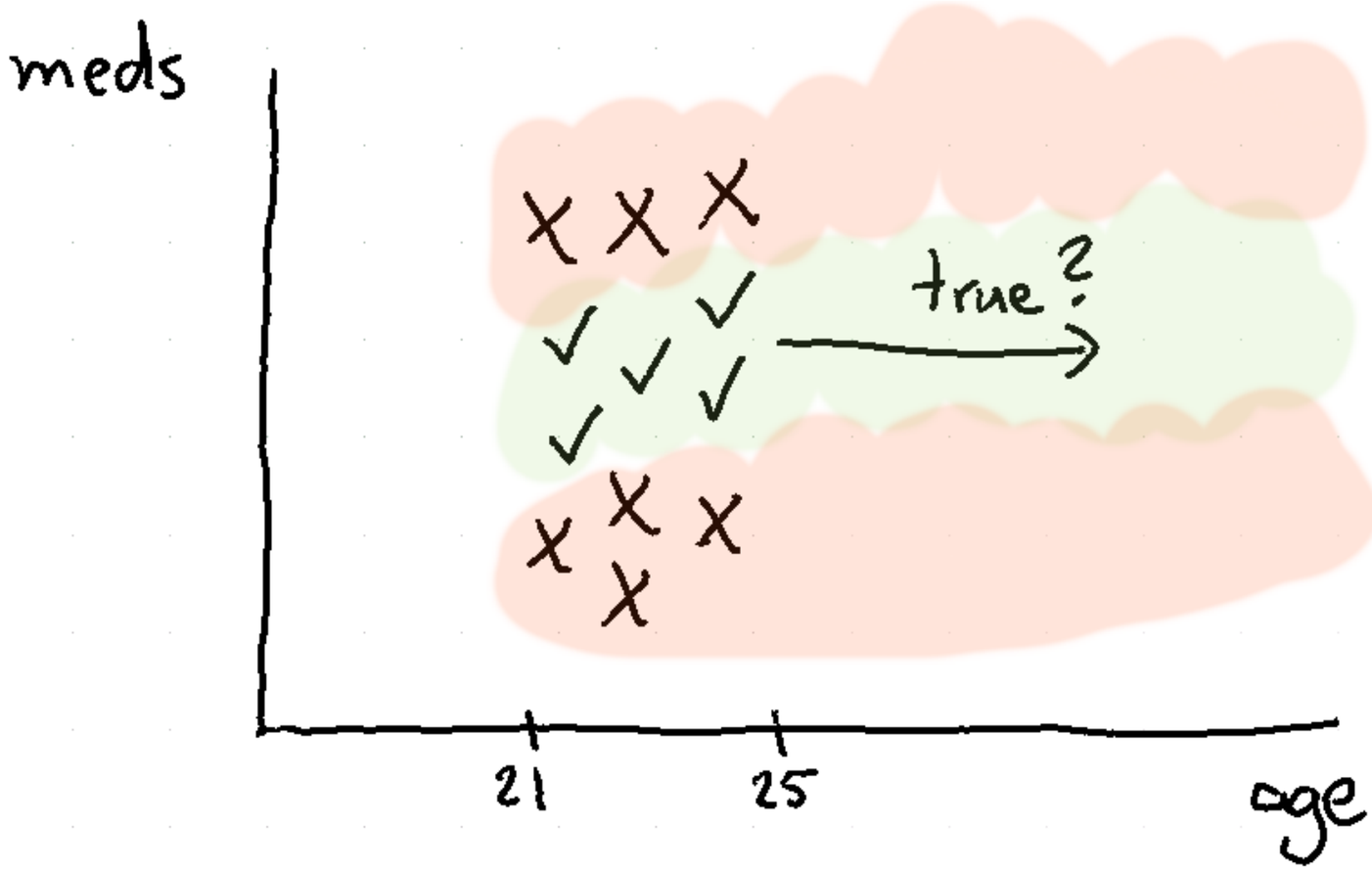
The output from `.predict_proba()` is an **approximation** of a probability. It might be a proxy but it is not a synonym of certainty.

This is because machine learning models are often designed for **interpolation, not extrapolation**. This means that applying them in an extrapolation usecase is tricky business.

A Common Error



A Common Error

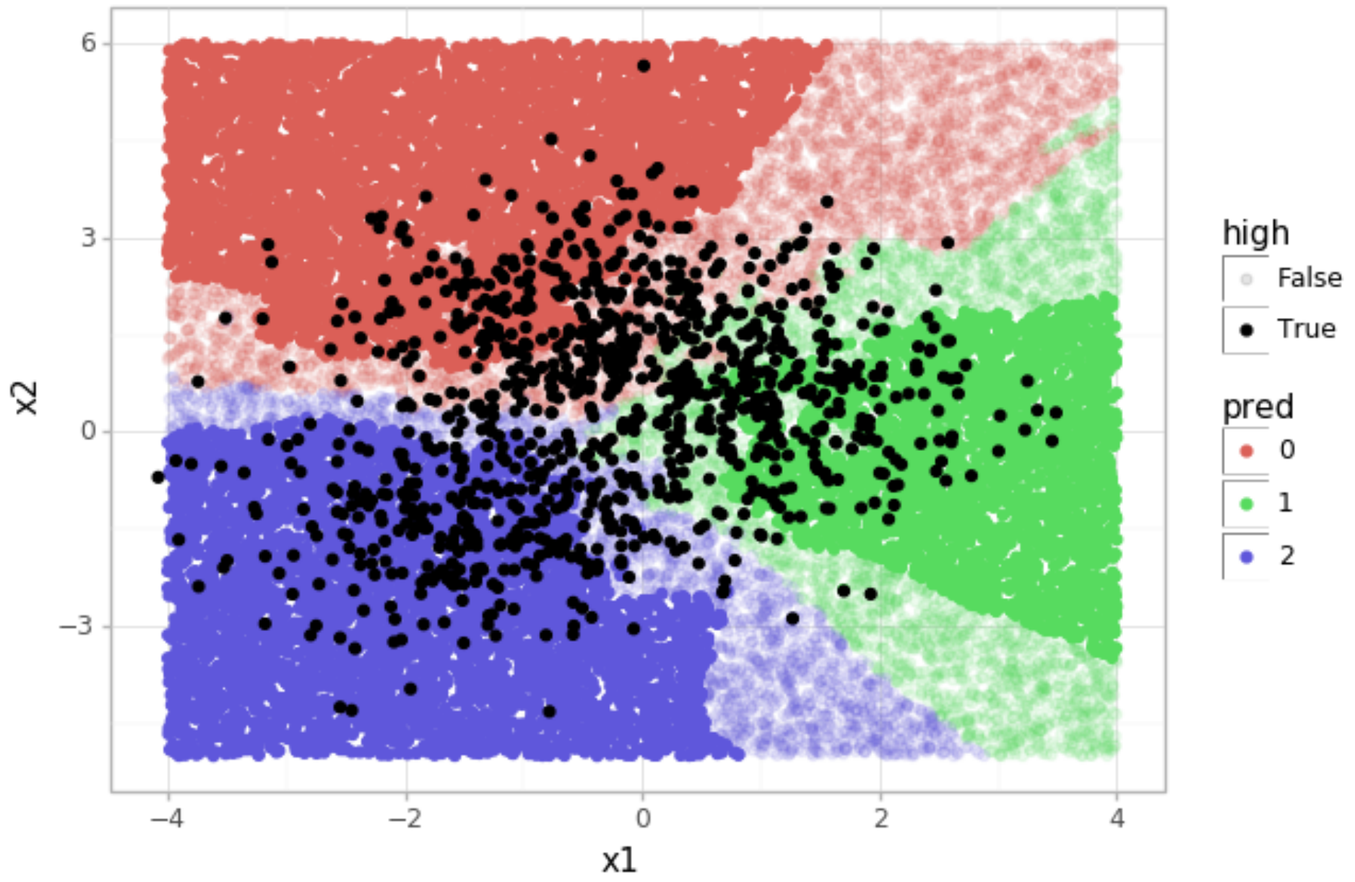


A Common Error

It would be less of an issue if machine learning models weren't designed to *always* give a result. There is usually no mechanism that assigns doubt to an outcome if a decision is made outside of a comfort zone.

Lucky for us, this last part of the problem we can be fixed with a little bit of probability glue.

A Common Error



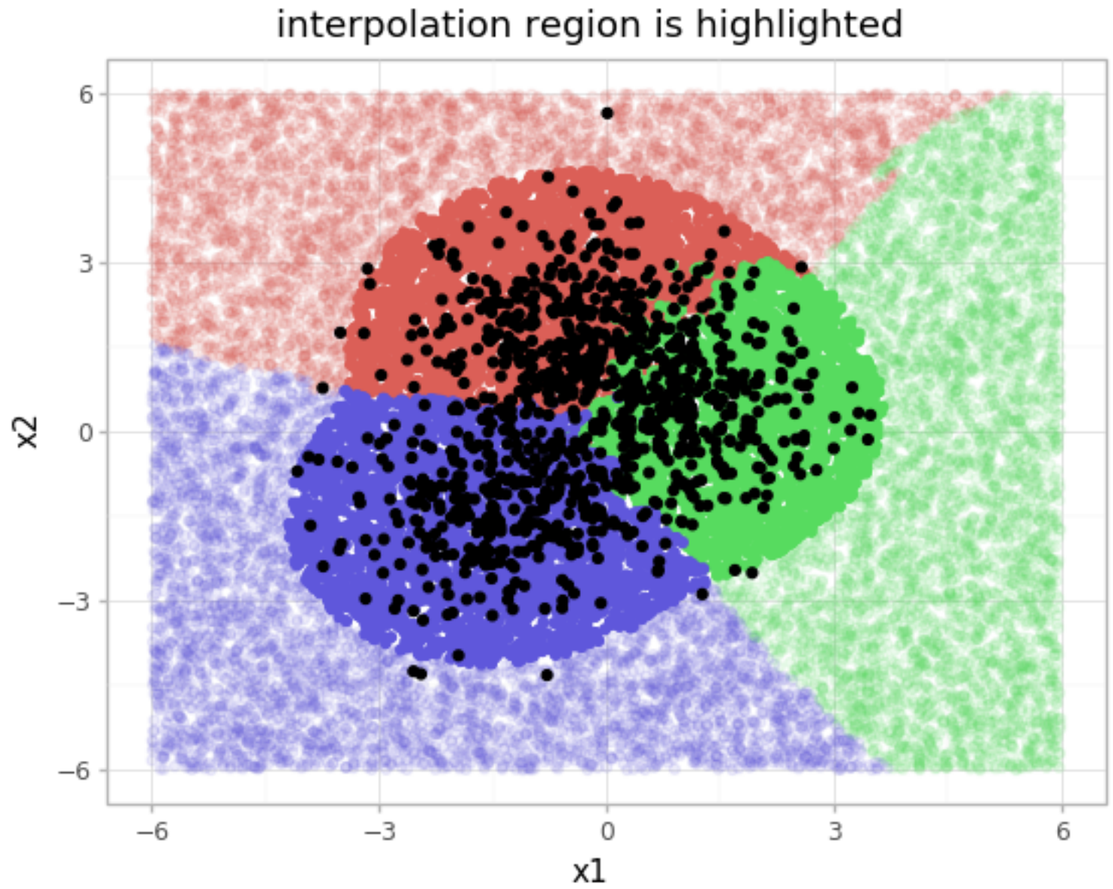
Gaussian Progress

We could train a Gaussian Mixture Model in this space for our original data and use this to detect if we're too far away from it.

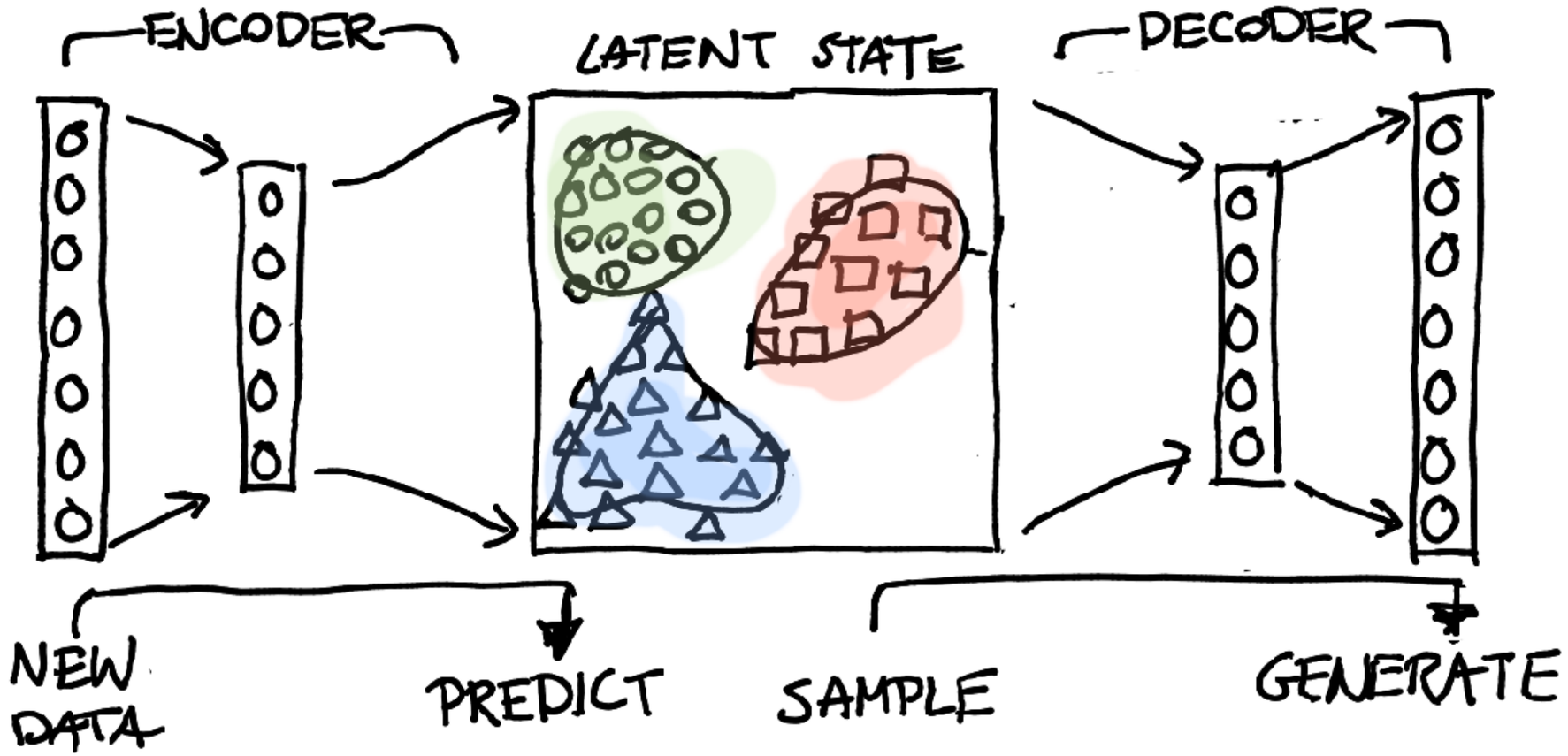
```
out = GaussianMixture(3).fit(X)
boundary = np.quantile(out.score_samples(X), 0.01)
```

It's my favorite outlier detection trick.

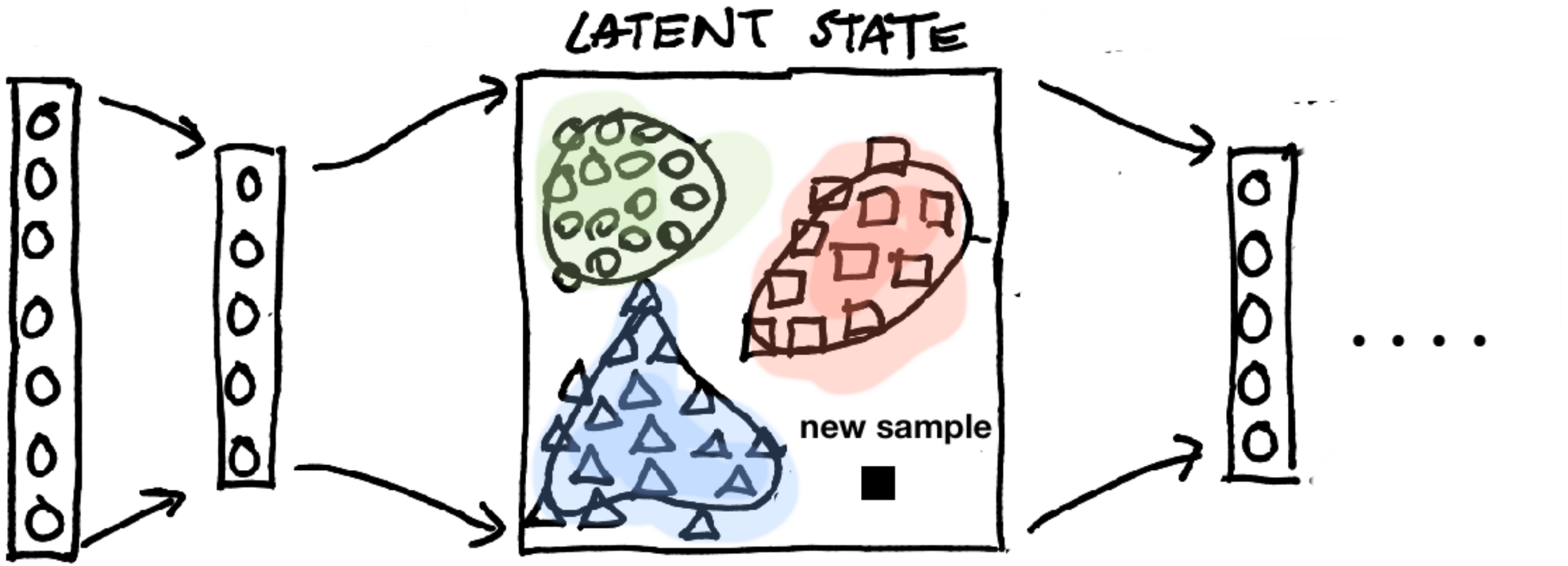
Safety



Generality

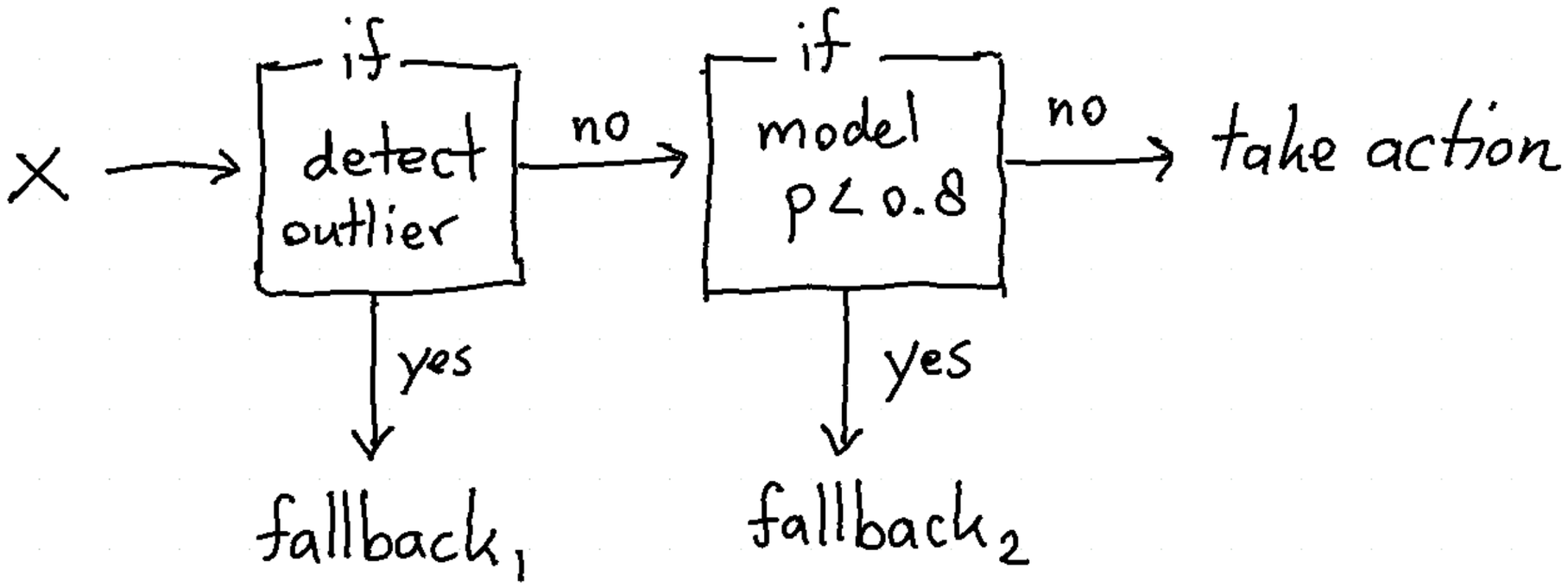

ENCODER
LATENT STATE
DECODER
NEW DATA
PREDICT
SAMPLE
GENERATE

Generality



Interesting Observation

Remember when people said that ML > rule based?



Lessons

The art is to fail gracefully. Not predicting given uncertainty is a **great idea**.

We can even design a fallback mechanism and unit-test in order to prevent mistakes.

But be careful! Algorithms merely automate, approximate and interpolate.

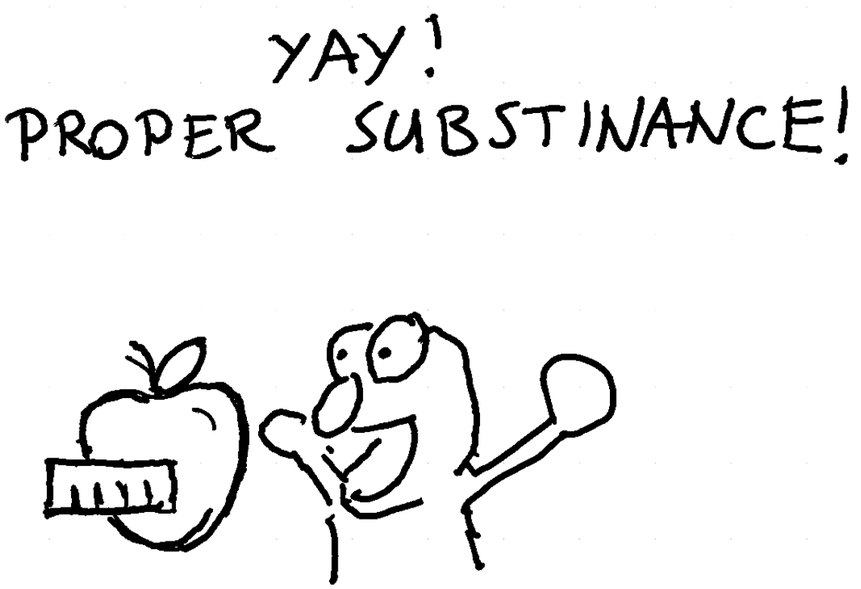
Lessons

Mere humans of management might **actually** think that our models are **actually** supplying us with an **actual** probability. This is wrong, it's an approximate proxy.

It is our job to care about the production layer. Try not to report on an A.I. miracle until you also understand when the model will fail.

Fix #2: Constrain thy Features

We can constraint the way we use the model but we might also constraint data we feed it.



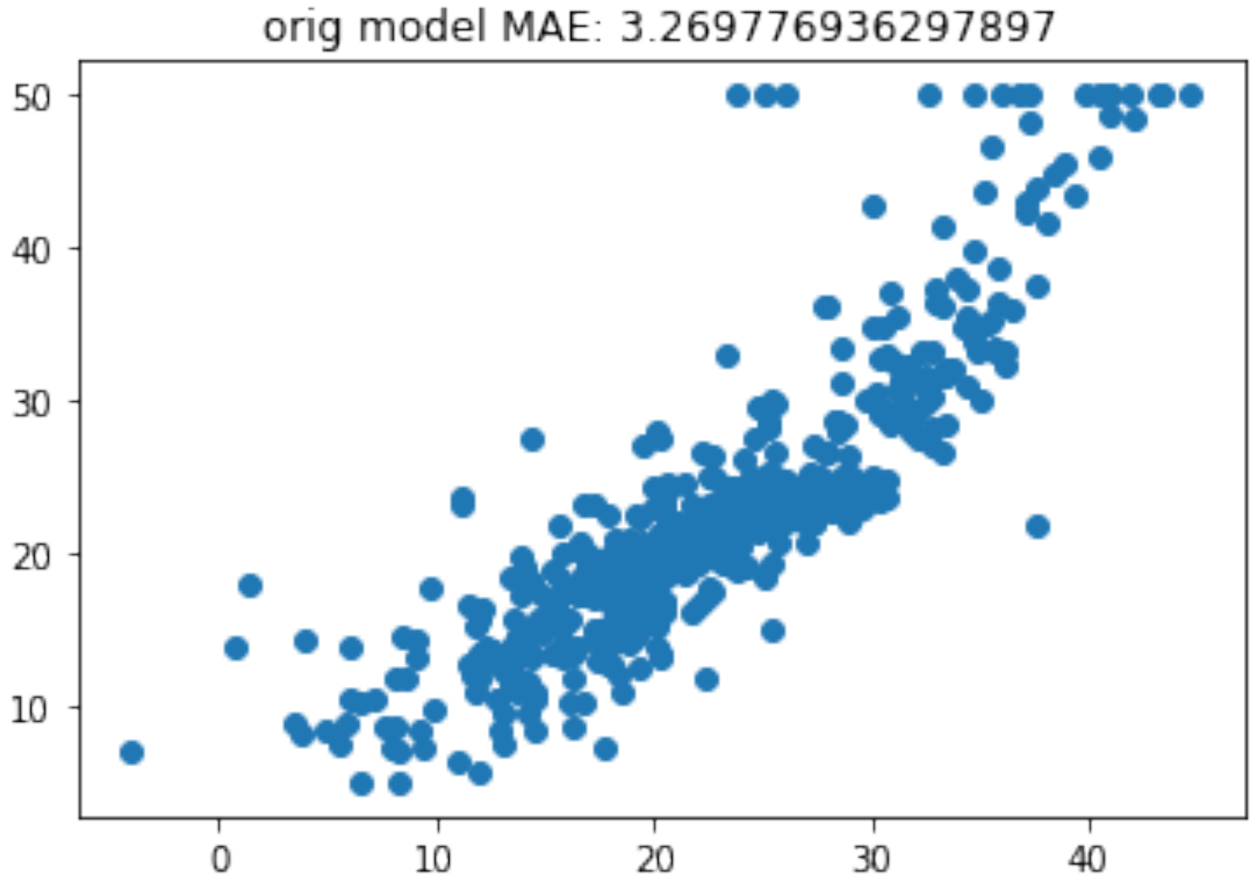
Fix #2: Constraint thy Features

We'll use a standard dataset and a standard pipeline to illustrate an issue.

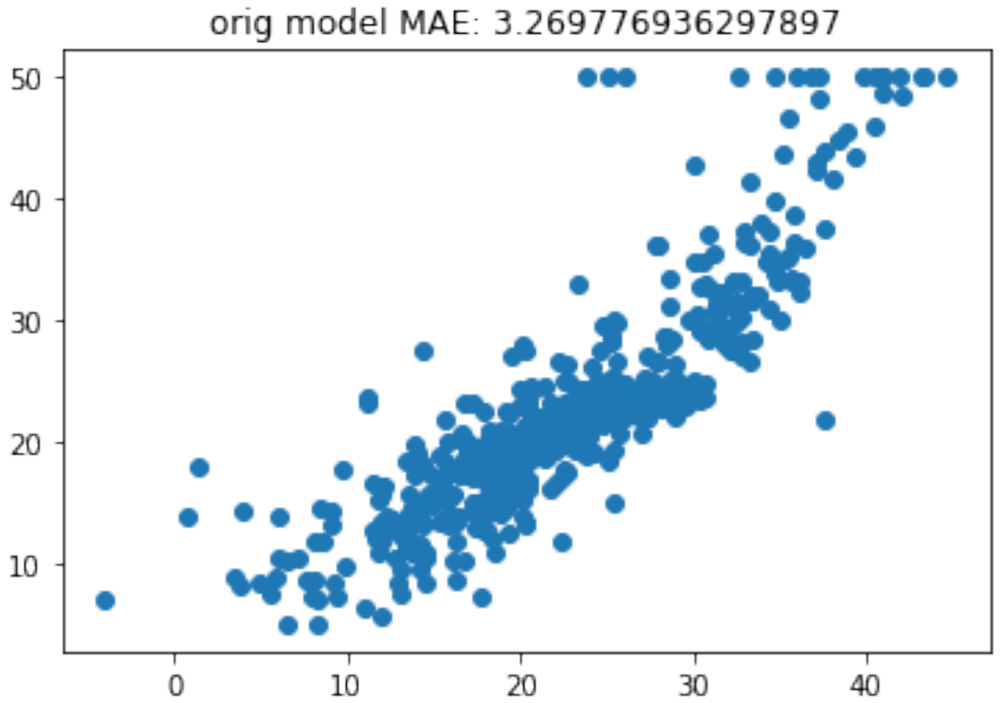
```
from sklearn.datasets import load_boston
from sklearn.pipeline import Pipeline

X, y = load_boston(return_X_y=True)
pipe = Pipeline([
    ("scale", StandardScaler()),
    ("model", LinearRegression())
])
```

Fix #2: Constraint thy Features



Fix #2: Constraint thy Features



Who can see the issue?

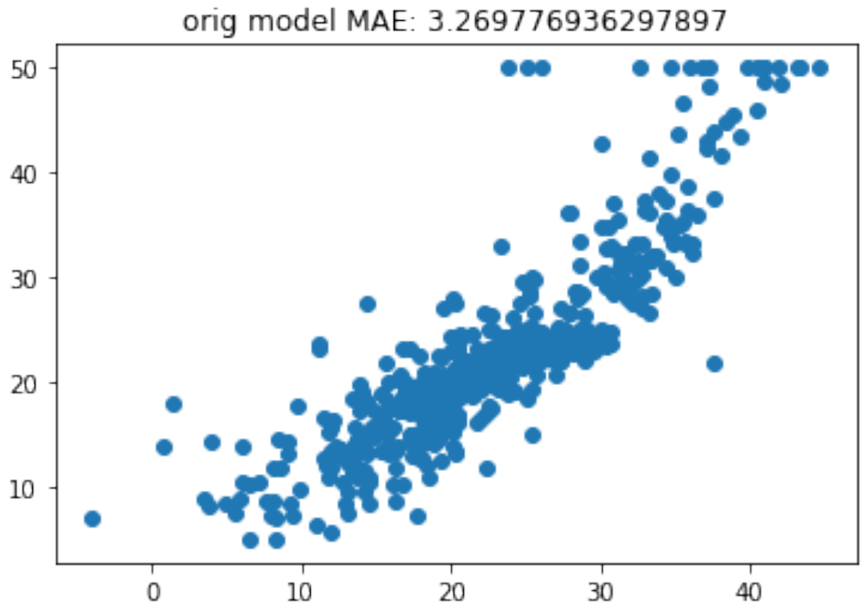
Fix #2: Constraint thy Features

What about now?

```
> load_boston()['DESCR']
...
- B       (Bk - 0.63)^2 where Bk is the % of blacks by town
- LSTAT   lower status of the population
```

I'm having a hard time coming up with a usecase for this dataset where these sensitive variables aren't data-laundering pre-existing bias.

This is an issue.



We're overfitting, not on a train set, but on the loss function. We should consider a fairness constraint.

Quantifying the Issue

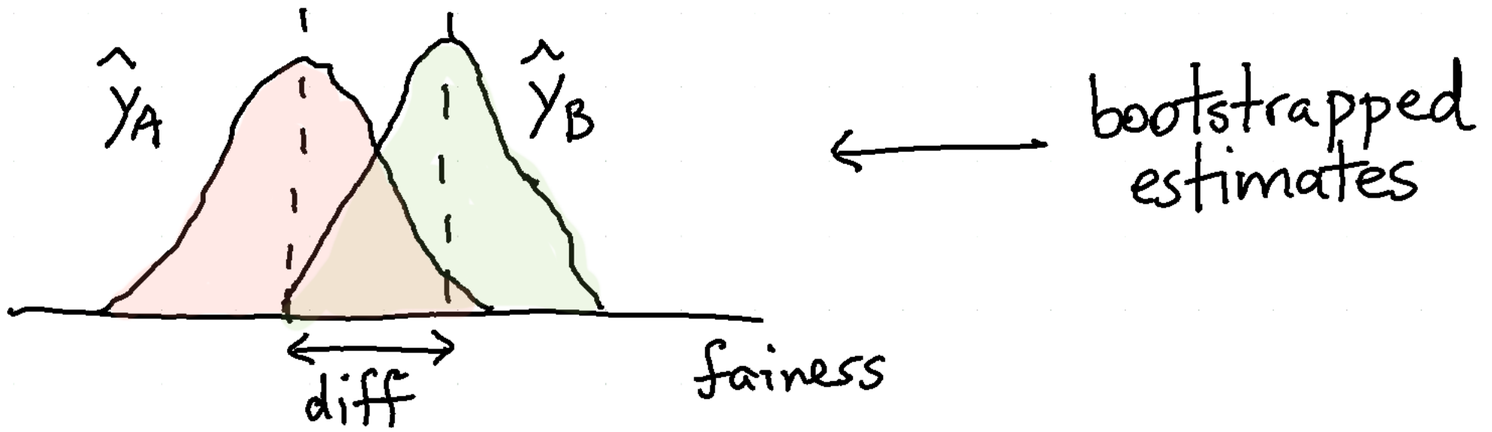
Say we have two prediction outputs, $\hat{y}_A$ and $\hat{y}_B$, one for each group in our sensitive attribute. I will consider one (there's many) definition of fairness;

$$\mathbf{E}[\hat{y}_A - \hat{y}_B] \approx 0$$

The sensitive attribute should not cause bias in the algorithm output. How to measure this?

Quantifying the Issue

Bootstrapping to the rescue.



From the predictions we try to quantify the bias between making a prediction between two groups.

Remove the column(s)

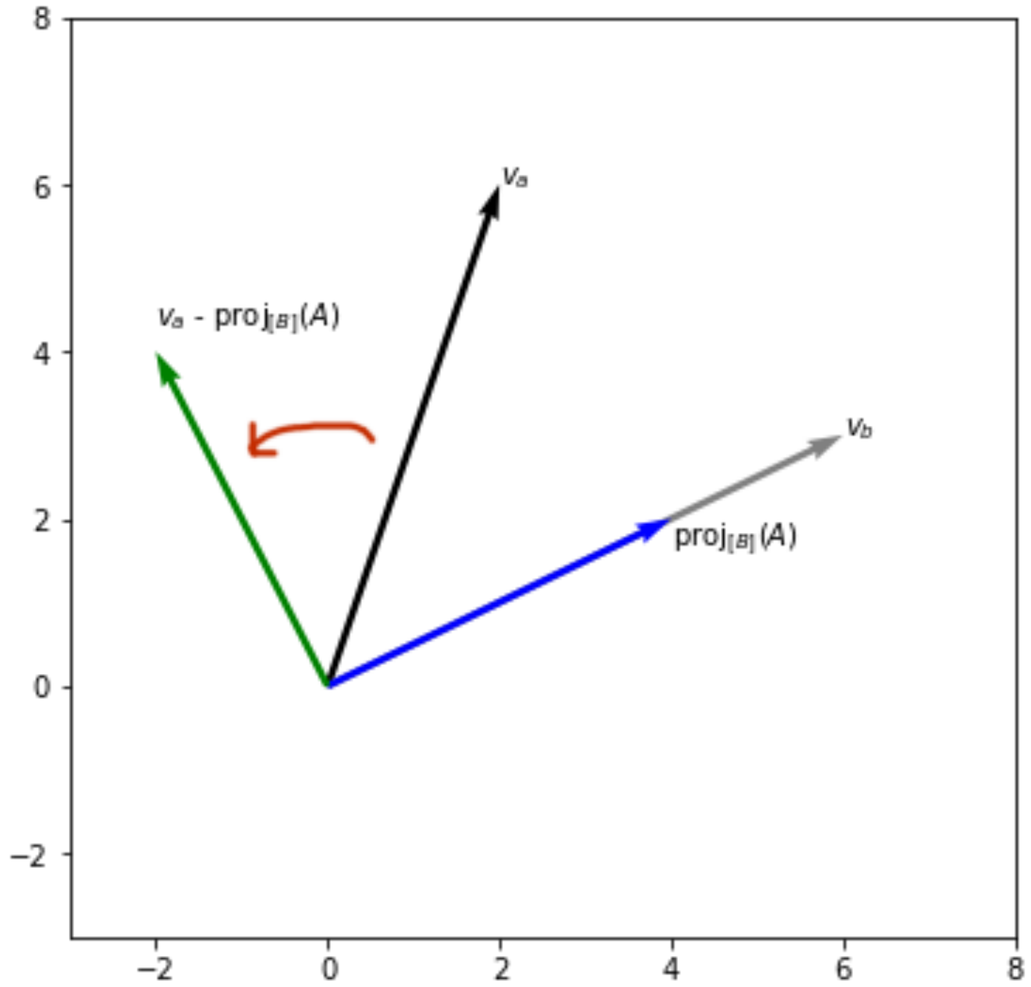
We've got our metric, now what?

We could remove the columns we don't like.

```
df_fair = df.drop(columns=<unfair col list>)
```

But there might be correlations between the other columns. We should probably do more.

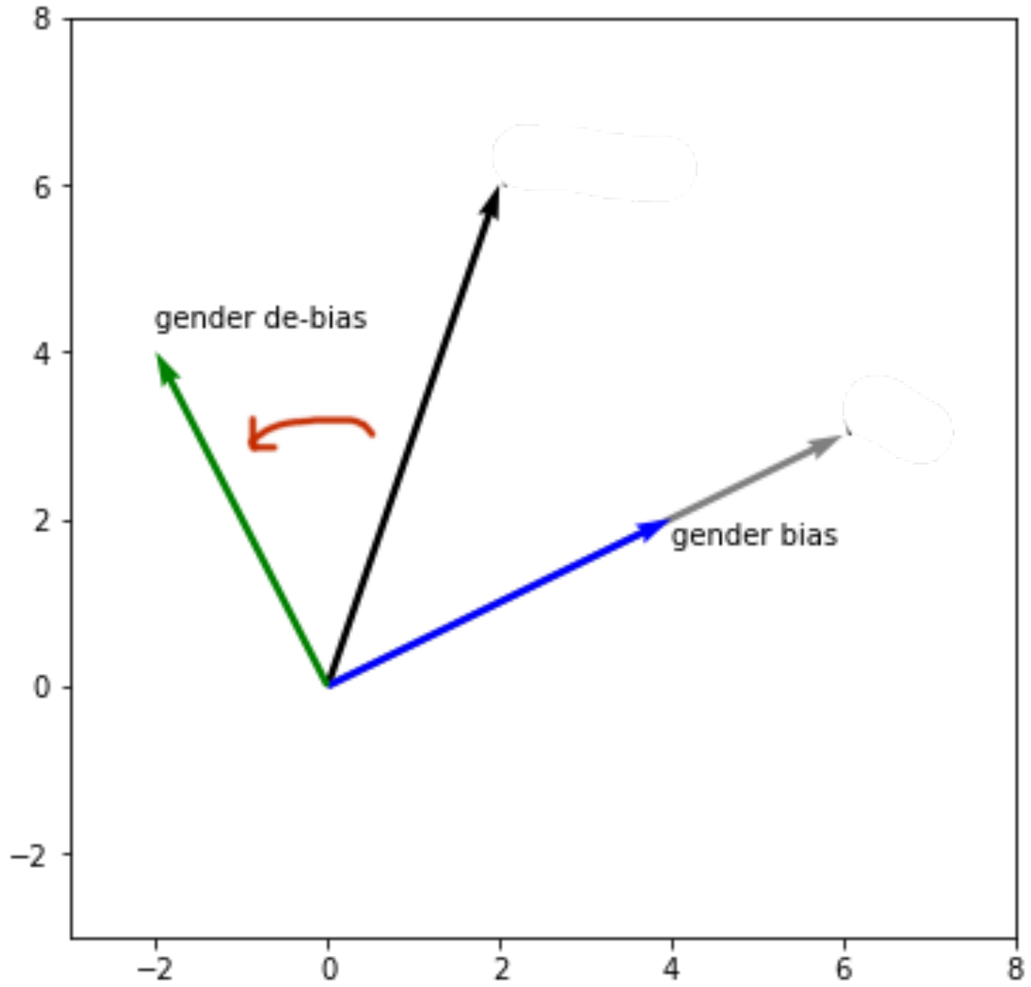
Remove the information(s)



There's a cool math trick we might do though.

$$
\begin{aligned}
v_1 &= x_1 \\
v_2 &= x_2 - \frac{x_2 v_1}{v_1 v_1} \\
v_3 &= x_3 - \frac{x_k v_1}{v_1 v_1} - \frac{x_2 v_2}{v_2 v_2} \\
&\dots \\
v_k &= x_k - \frac{x_k v_1}{v_1 v_1} - \frac{x_2 v_2}{v_2 v_2}
\end{aligned}
$$

Remove the information(s)



There's a cool math trick we might do though.

$$
\begin{aligned}
v_1 &= x_1 \\
v_2 &= x_2 - \frac{x_2 v_1}{v_1 v_1} \\
v_3 &= x_3 - \frac{x_k v_1}{v_1 v_1} - \frac{x_2 v_2}{v_2 v_2} \\
&\dots \\
v_k &= x_k - \frac{x_k v_1}{v_1 v_1} - \frac{x_2 v_2}{v_2 v_2}
\end{aligned}
$$

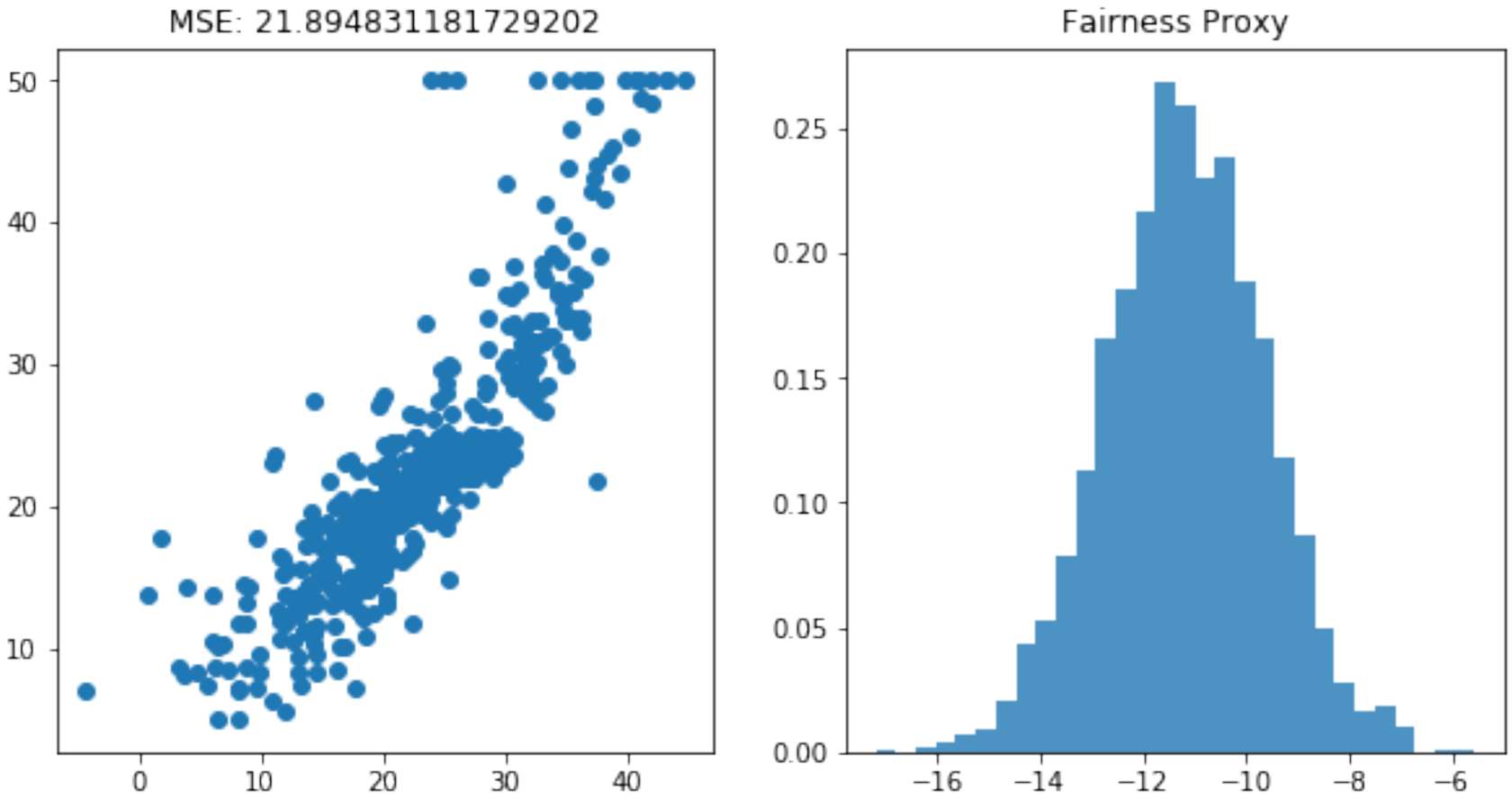
Let's compare!

We will demonstrate three scenarios:

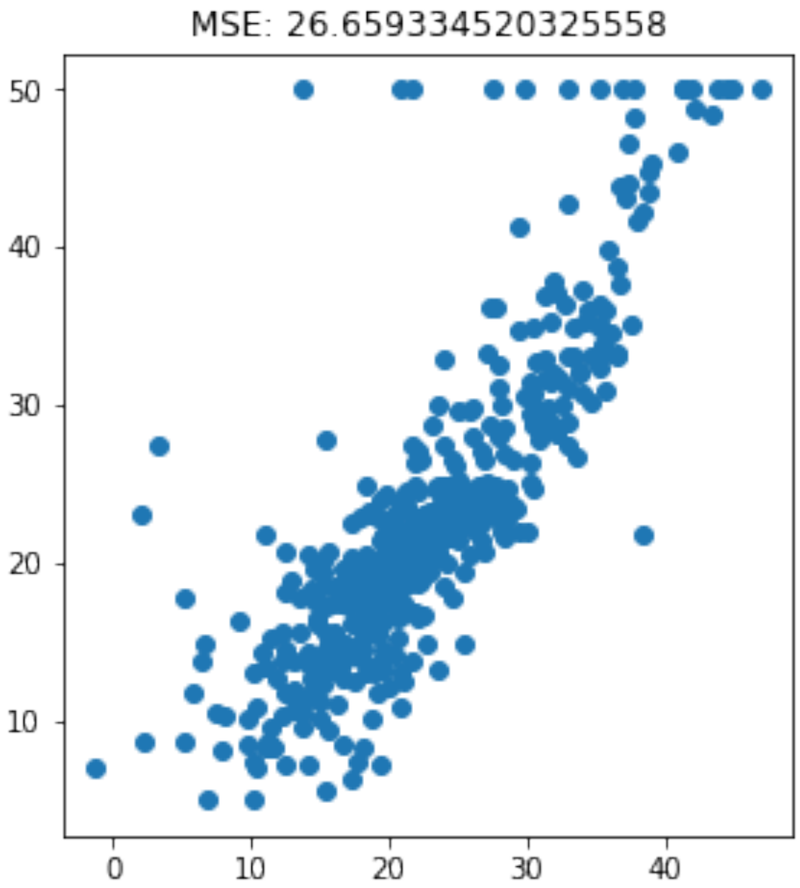
1. Normal situation
2. Situation with two dropped columns
3. Situation with information filter

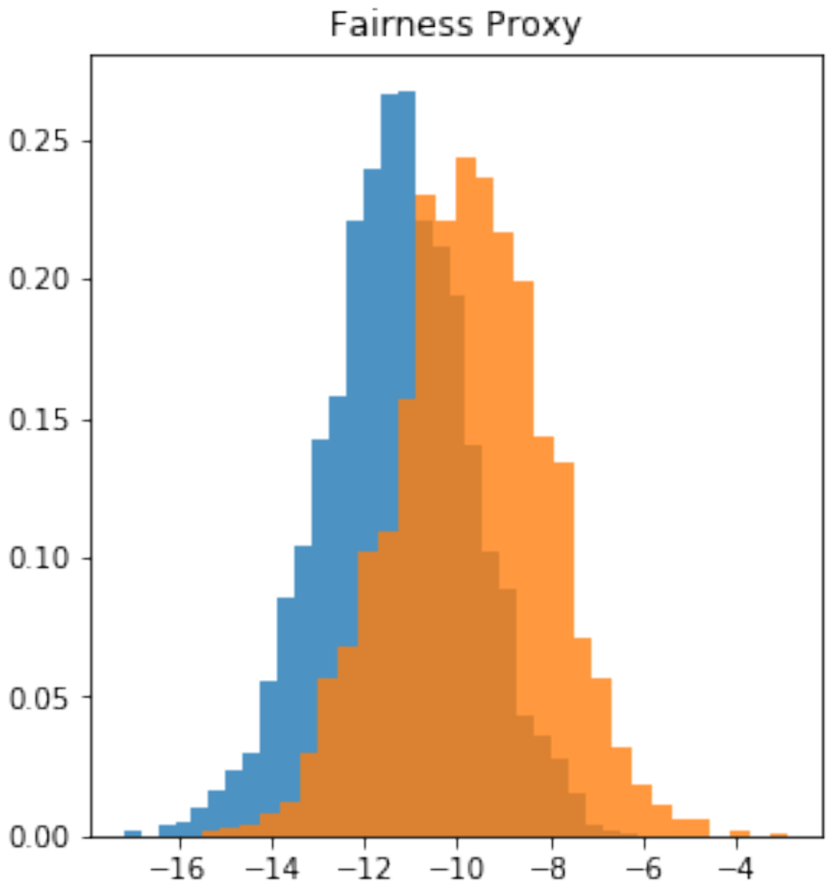
For each situation we will compare the fit of the model as well as the fairness distribution.

Let's compare! - Base

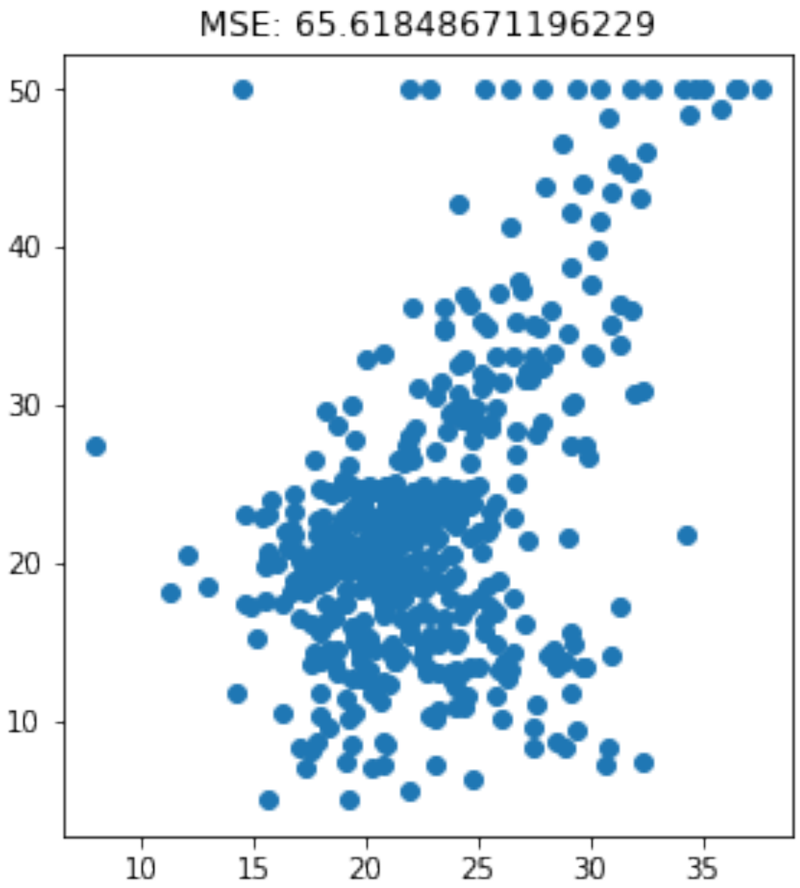


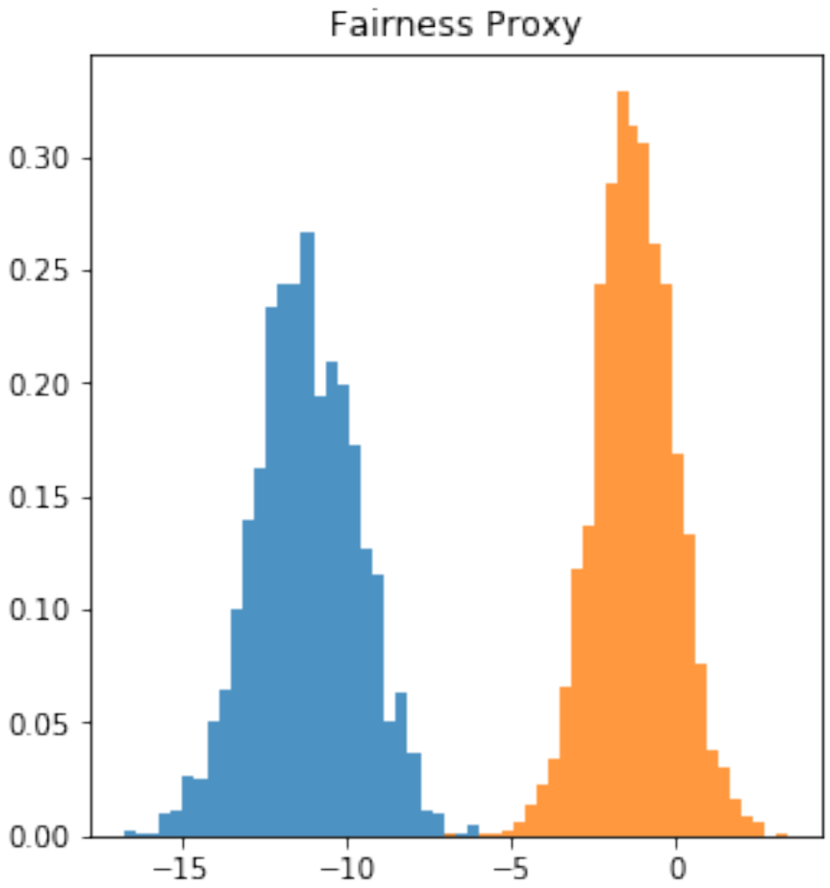
Let's compare! - Column Drop



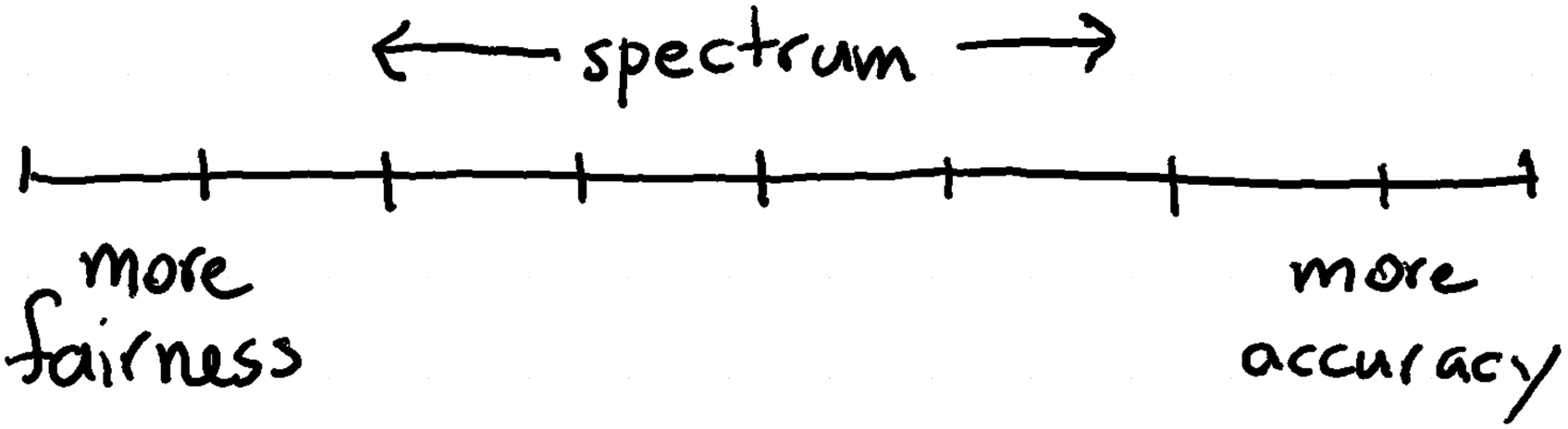


Let's compare! - Column Filter



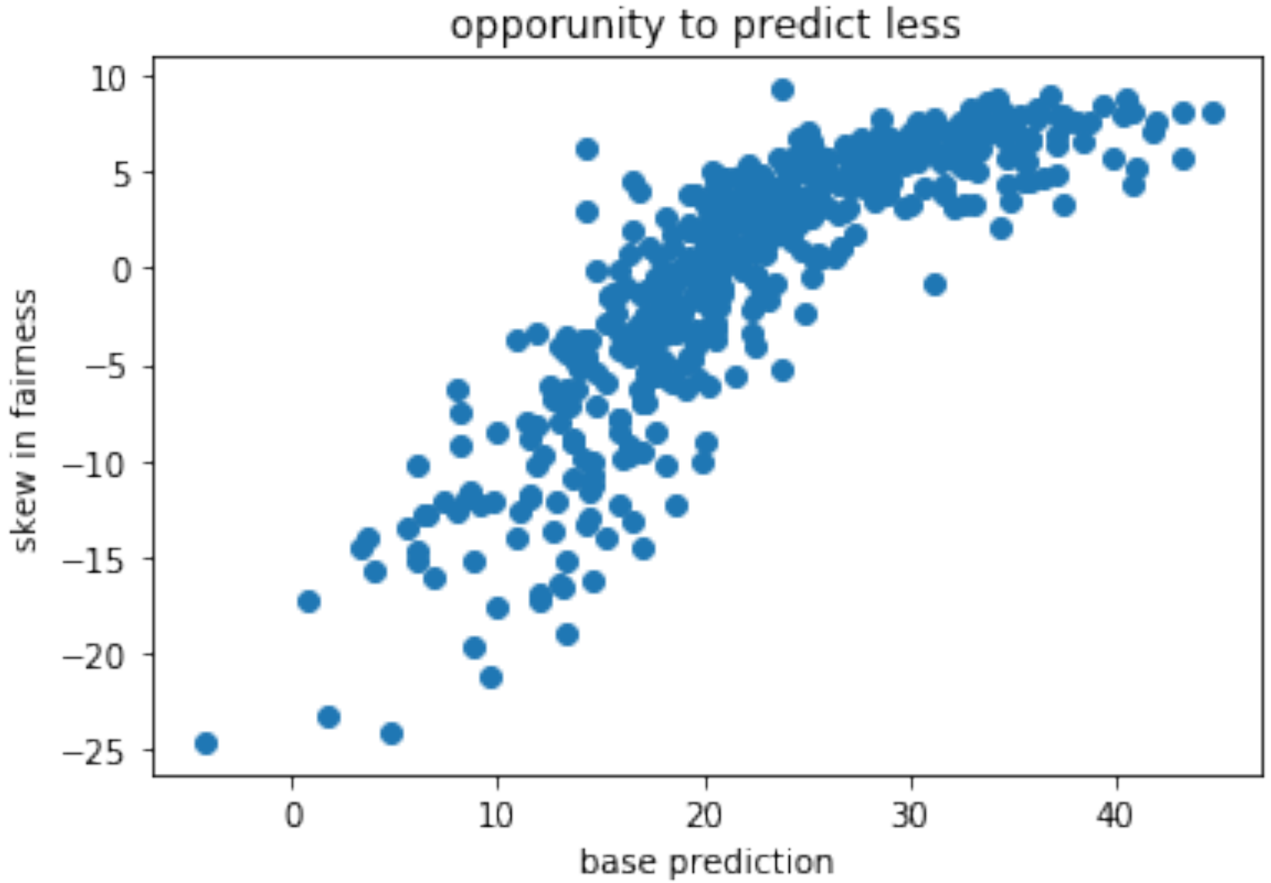


Spectrum



This is unfortunate because I don't expect corporate incentives to align here.

Predict Less!



Horrible Truth of All This

Fairness is in the eye of the beholder. There's an issue.

Horrible Truth of All This

I like the trick ... but ...

Horrible Truth of All This

I like the trick ... but ...

... in order to apply this trick to cause fairness ... with regards to income and skin color ...

Horrible Truth of All This

I like the trick ... but ...

... in order to apply this trick to cause fairness ... with regards to income and skin color ...

I need to **know** the income and skin color. What would have happened if we didn't have these columns?

Horrible Truth of All This

I like the trick ... but ...

... in order to apply this trick to cause fairness ... with regards to income and skin color ...

I need to know the income and skin color. What would have happened if we didn't have these columns?

Fairness might come at the cost of privacy.

Is this such a bad thing?

Fairness might come at the cost of privacy. Bargain?

Let's be synical; it is possible to use "we aren't allowed to know your gender" as an excuse to not measure fairness. As we saw before, not including a variable in your model **is not** the same as "being fair".

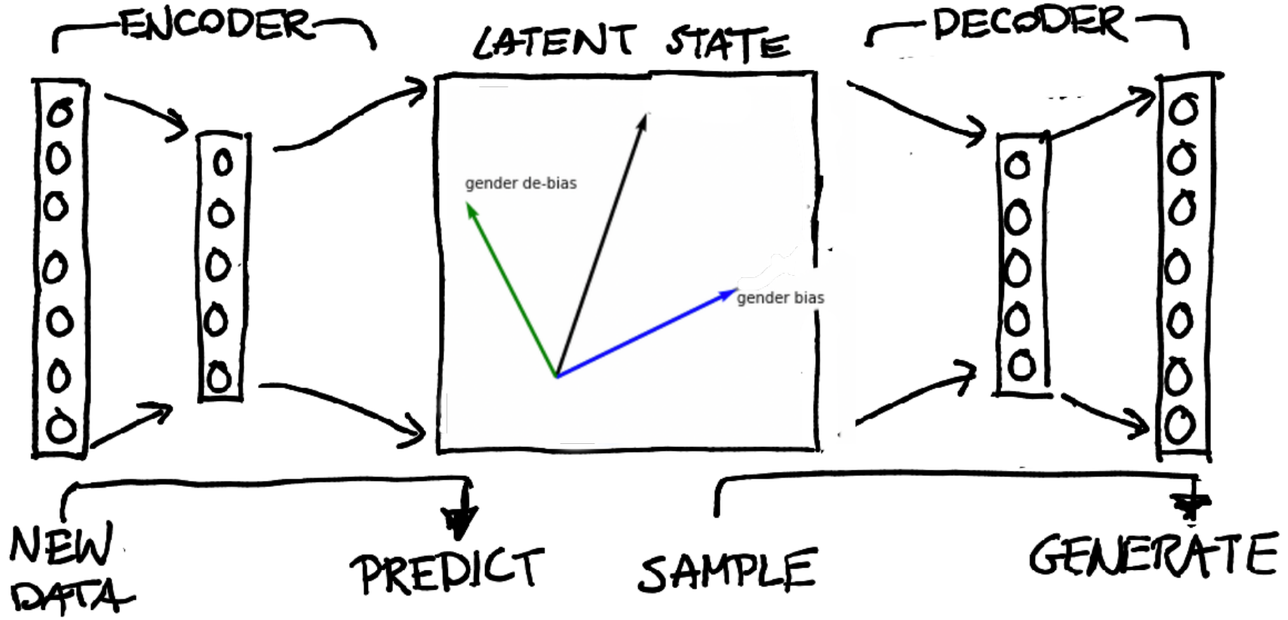
Is this such a bad thing?

Fairness might come at the cost of privacy. Bargain?

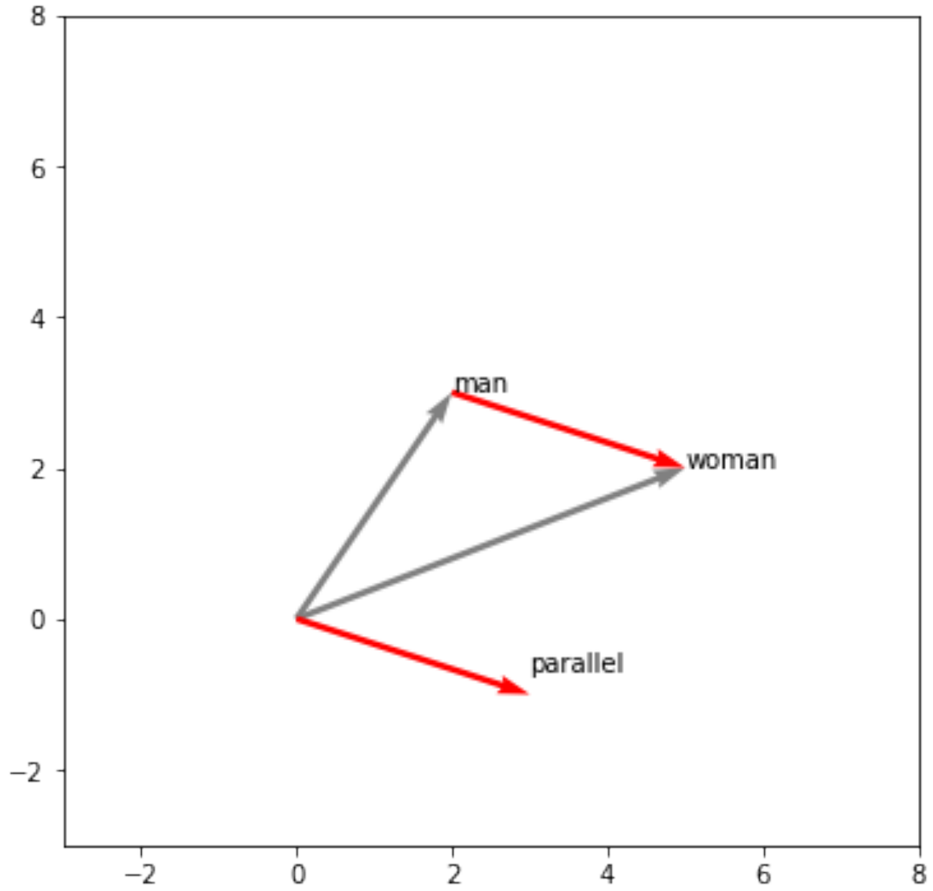
Let's be synical; it is possible to use "we aren't allowed to know your gender" as an excuse to not measure fairness. As we saw before, not including a variable in your model **is not** the same as "being fair".

A.I. Caramba!

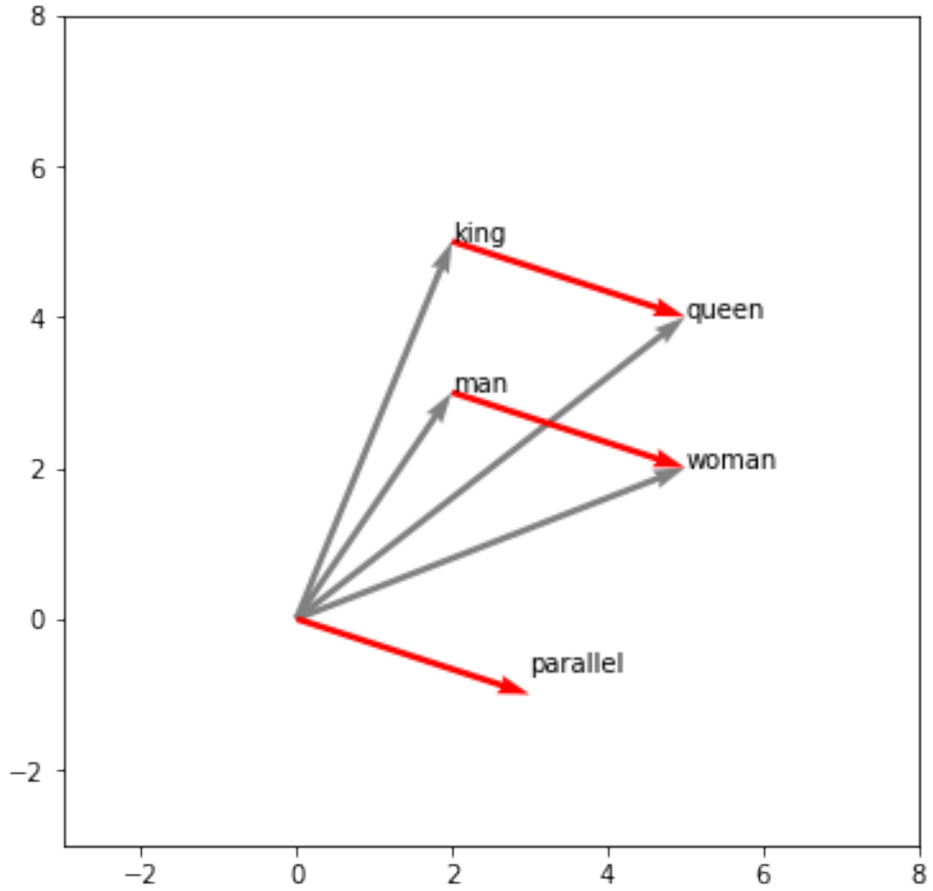
Our trick of "constraining features" is potentially useful in deep learning approaches too.



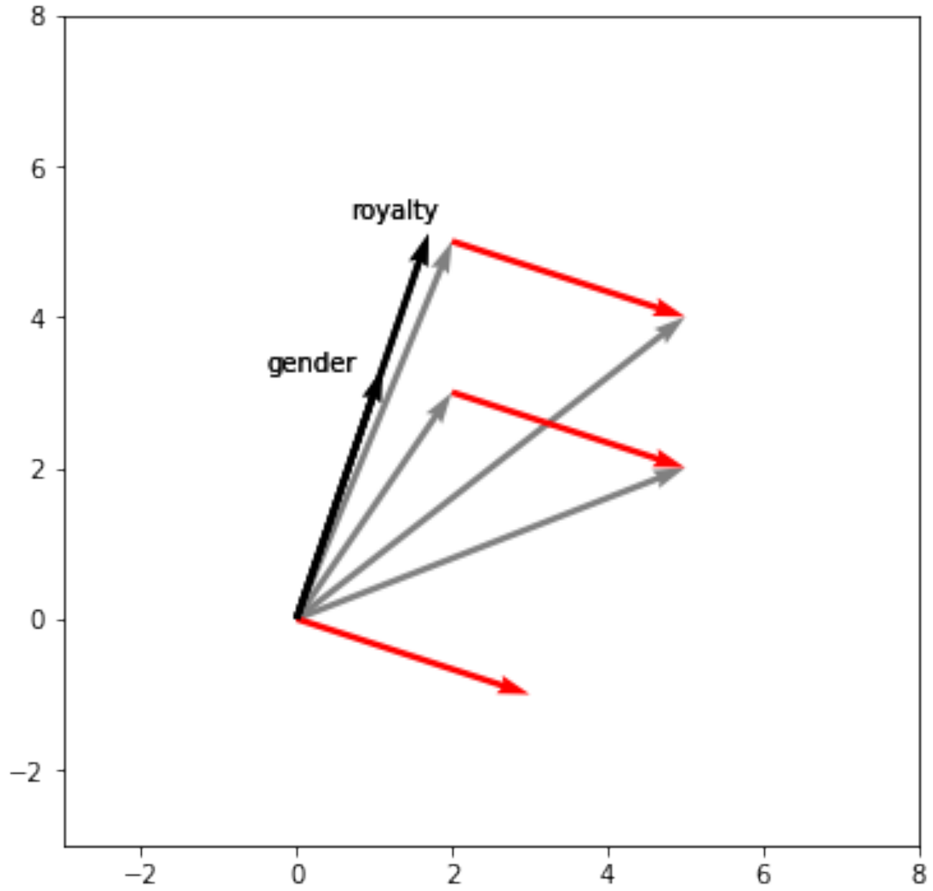
Not a Bad Trick (but beware analogies)



Not a Bad Trick (but beware analogies)



Not a Bad Trick (but beware analogies)



Such Importance, Variable Importance!

The variables that you put into your system and how they are preprocessed are **more** defining of the end result than the ML algorithm. I like seeing modelling pipelines where you **explicitly** select columns from a dataframe because it should be a fundamental part.

Understanding what goes into your model matters and everybody here should care. If the predictions goin' out is your responsibility, then so is the data goin' in.

Mathematical Constraints

We're moving in the area of constraining the model. If I want to do this properly I should discuss a bit of mathematics beforehand.

Observation

Suppose we want to optimise a portfolio of stocks. You could describe this mathematically via;

$$\mathbf{max\ profit}(\mathbf{x}) = r_1 x_1 + \ldots + r_k x_k$$

Observation

We should add a constraint.

$$\begin{aligned} \max \text{profit}(\mathbf{x}) &= r_1 x_1 + \ldots + r_k x_k \\ \text{subject to} & \\ \sum_i x_i &\leq \text{budget} \end{aligned}$$

Observation

But this constraint is even more important.

$$
\begin{aligned}
\max \text{profit}(\mathbf{x}) &= r_1 x_1 + \ldots + r_k x_k \\
\text{subject to} & \\
\sum_i x_i &\leq \text{budget} \\
\text{risk}(\mathbf{x}) &\leq \text{preference}
\end{aligned}
$$

What Constraints do we have on Models?

I'll discuss three:

— monotonicity
— fairness
— label guarantees

But there are many more, albeit numerically difficult.

Exibit A: Monotonic Features

$$\min \text{loss}(f(\theta|X), y))$$

$$\text{subject to}$$

$$\frac{df}{dx_i} > 0$$

$$\frac{df}{dx_j} < 0$$

Exibit A: Monotonic Features

For a single variable model this is well solvable.

$$\min \sum_i (\hat{y}_i - y_i)^2$$

subject to

$$\hat{y}_i < \hat{y}_j \qquad \forall i < j$$

Exibit A: Monotonic Features

For a single variable model this is well solvable.

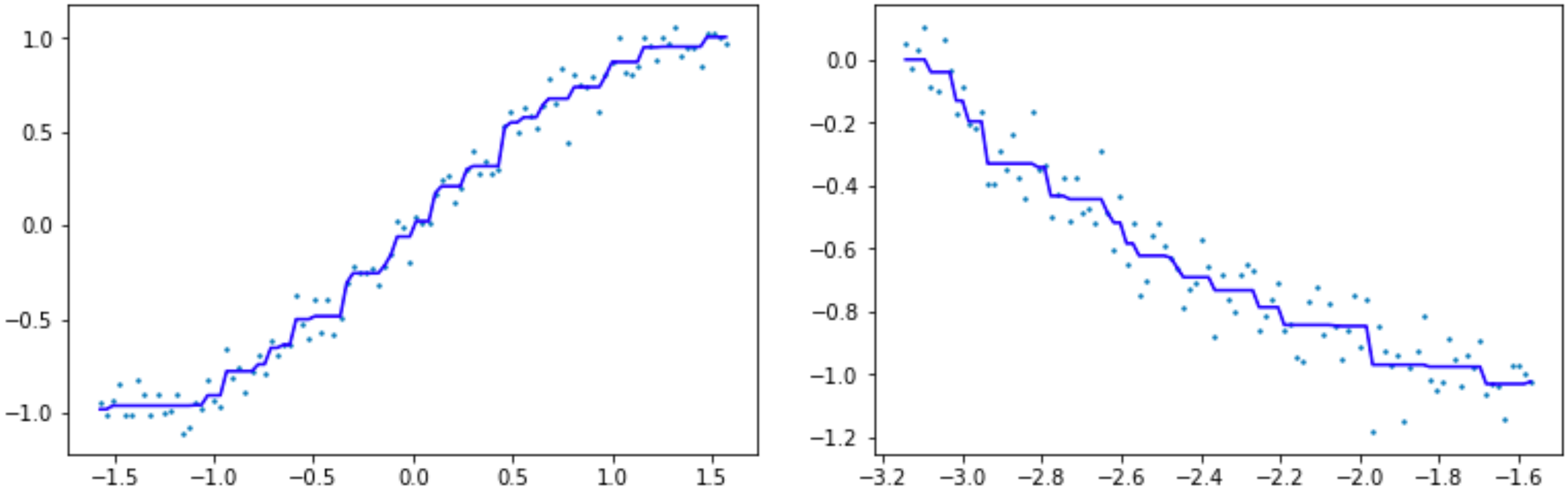
$$\min \sum_i (\hat{y}_i - y_i)^2$$

$$\text{subject to}$$

$$\hat{y}_i < \hat{y}_j \qquad \forall i < j$$

In fact, this is implemented in **sklearn** as `IsotonicRegression`.

You can expect relationships to look like this.

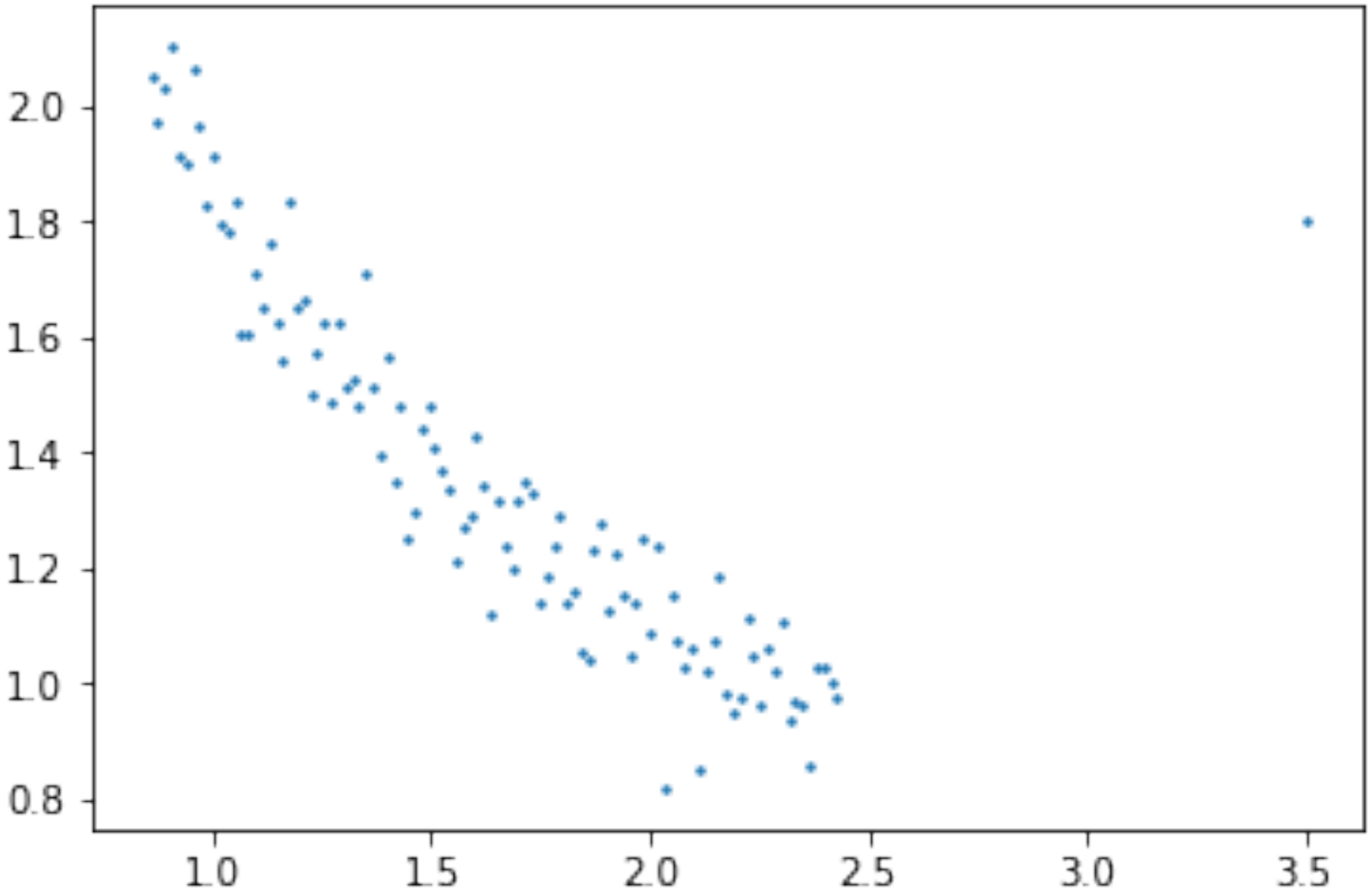


This is numerically tractable. Neeto.

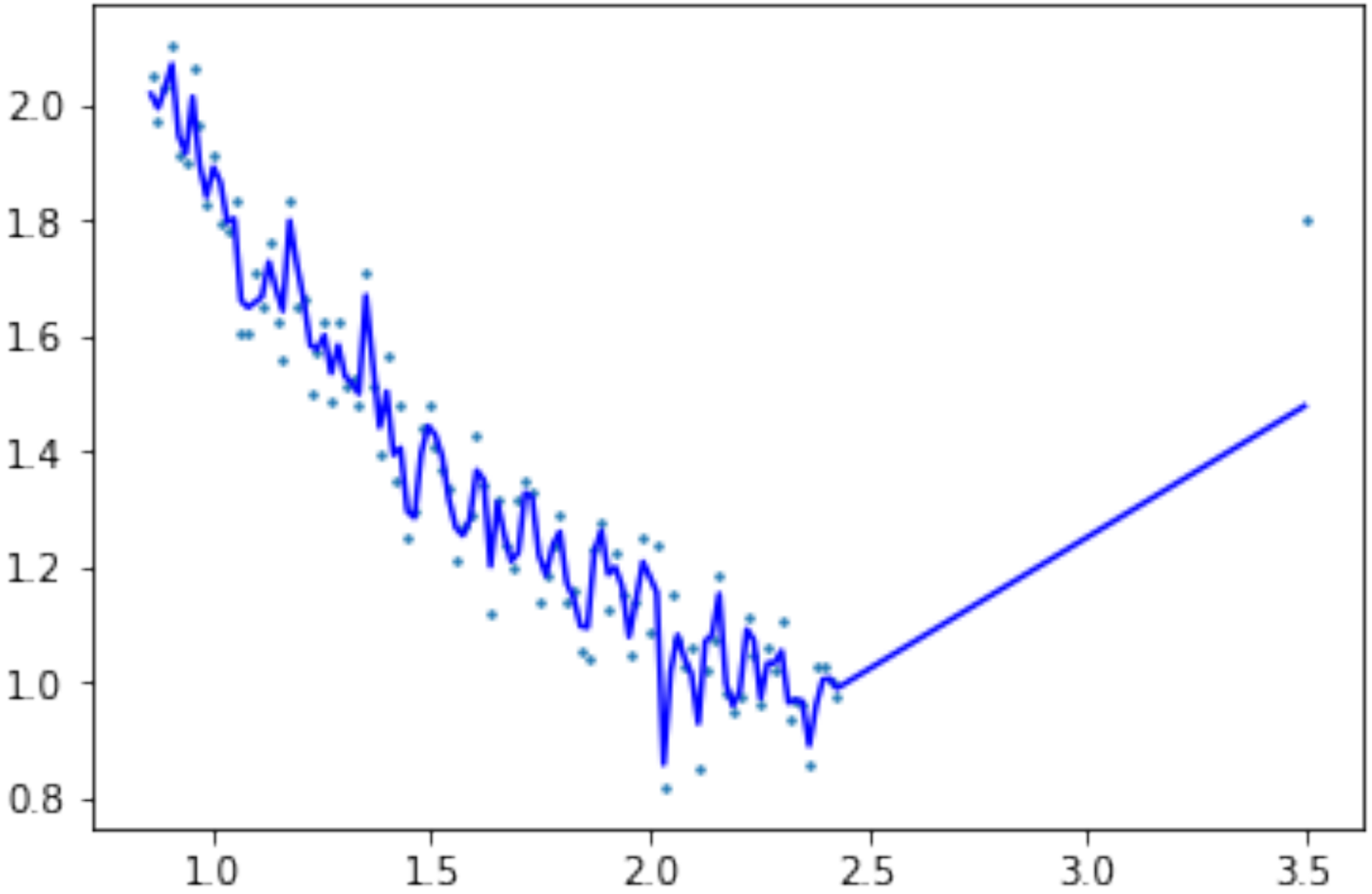
When

does this matter?

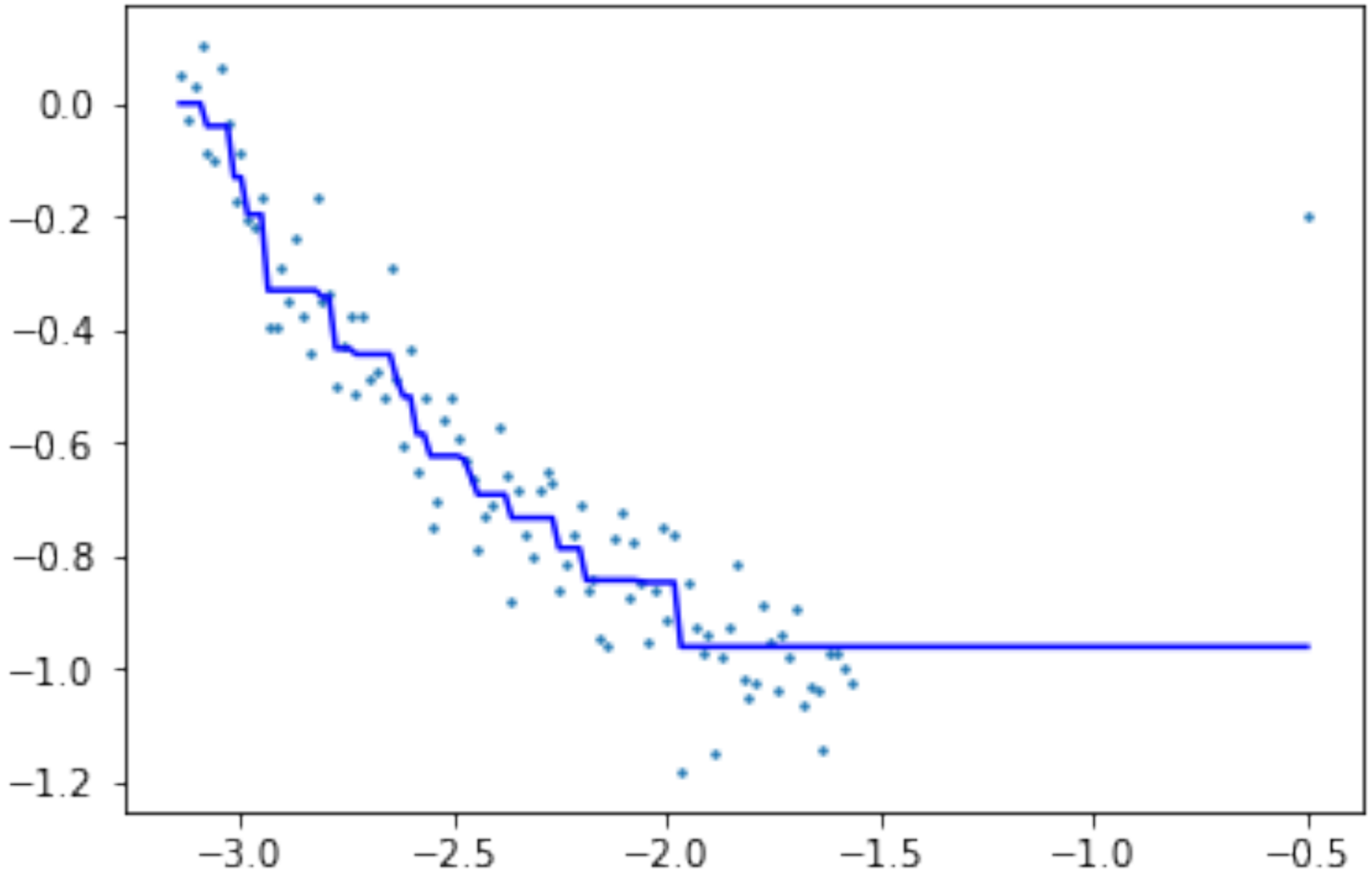
When your models merely interpolate.



This is the output of a random forest regressor.



This is the output of a monotonic regressor.



Are we limited to simple models?

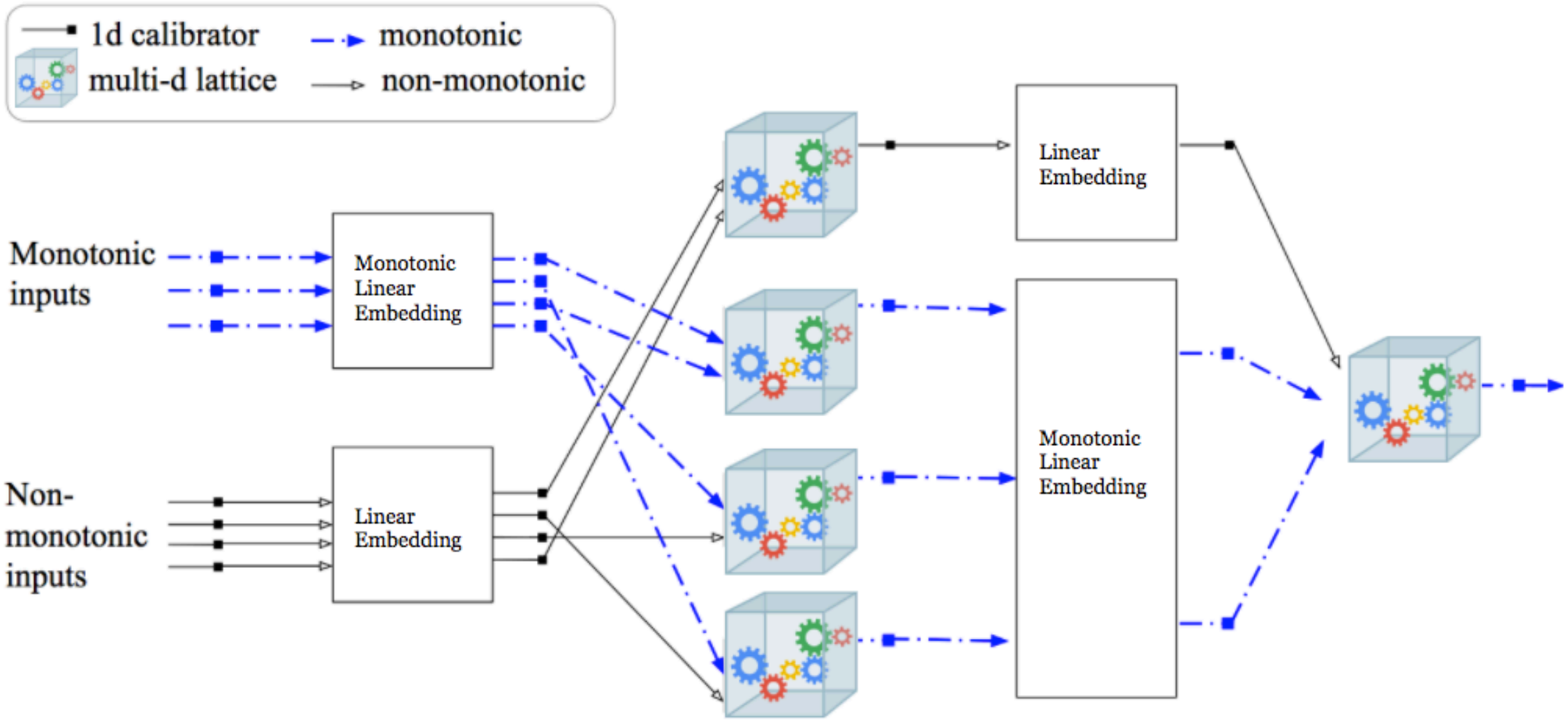
Are we limited to simple models?

1. First of all, simple models **aren't** limited.

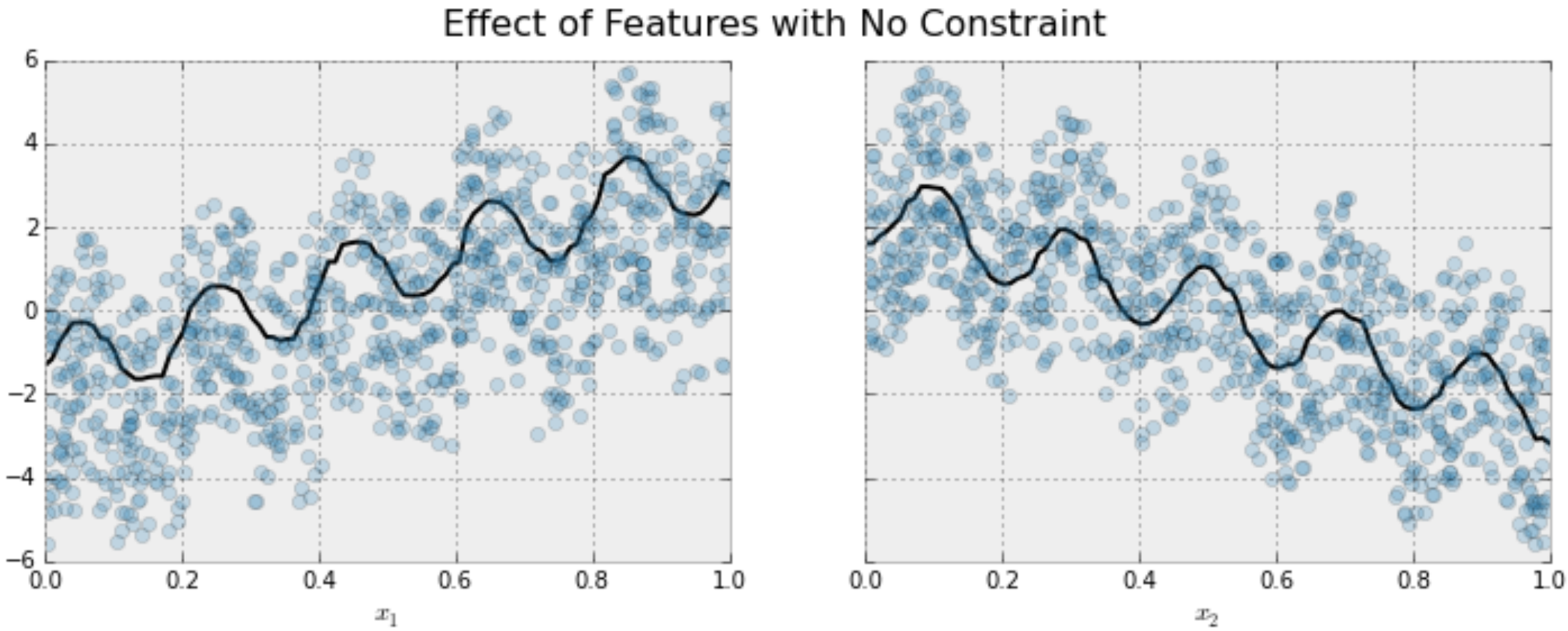
Are we limited to simple models?

1. First of all, simple models **aren't** limited.
2. Second, no.

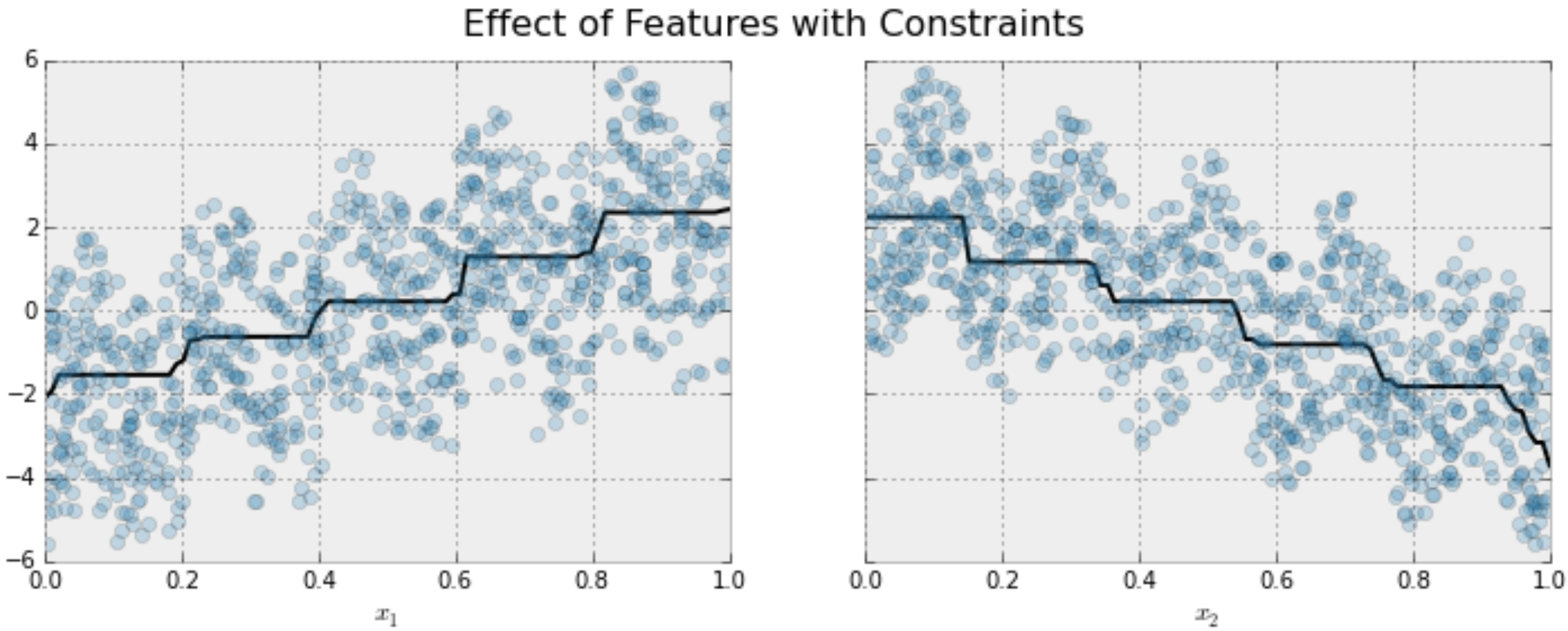
Neural Networks (TF Lattice)



XGBoost


Effect of Features with No Constraint
6
4
2
0
−2
−4
−6
0.0
0.2
0.4
0.6
0.8
1.0
x_1
0.0
0.2
0.4
0.6
0.8
1.0
x_2

XGBoost


Effect of Features with Constraints
6
4
2
0
−2
−4
−6
0.0
0.2
0.4
0.6
0.8
1.0
x_1
0.0
0.2
0.4
0.6
0.8
1.0
x_2

What else can we do?

We might add constraints for fairness!

$$\min \text{loss}(f(\theta|X), y))$$

$$\text{subject to}$$

$$\text{fairness}(\text{gender}) \geq \frac{p}{100}$$

$$\text{fairness}(\text{race}) \geq \frac{q}{100}$$

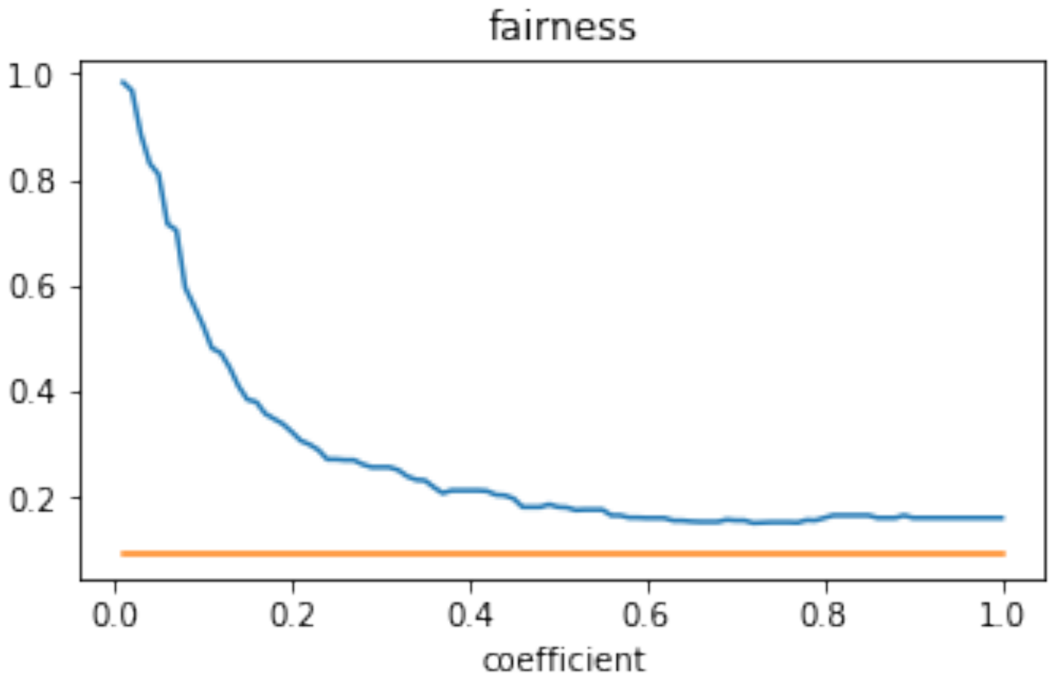
Fair Logistic Regression

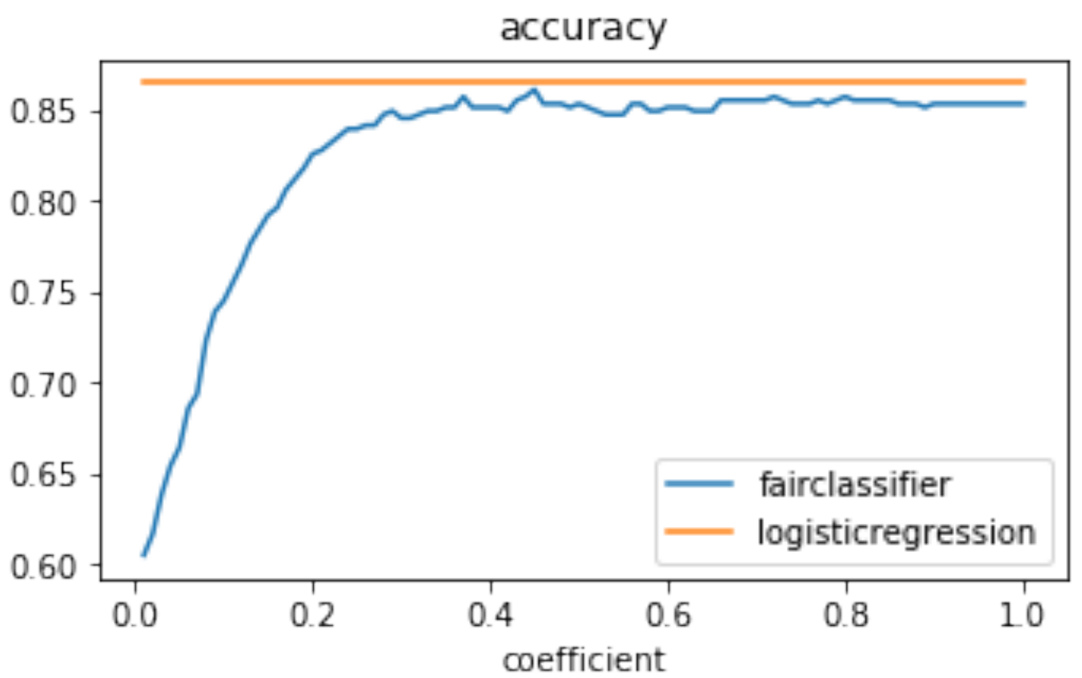
We might add constraints for fairness!

$$
\begin{aligned}
\text{minimize} \quad & -\sum_{i=1}^{N} \log p\left(y_i \middle| \mathbf{x}_i, \boldsymbol{\theta}\right) \\
\text{subject to} \quad & \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{z}_i - \overline{\mathbf{z}}\right) \boldsymbol{\theta}^T \mathbf{x}_i \leq \mathbf{c} \\
& \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{z}_i - \overline{\mathbf{z}}\right) \boldsymbol{\theta}^T \mathbf{x}_i \geq -\mathbf{c}
\end{aligned}
$$

The paper is a cool read.

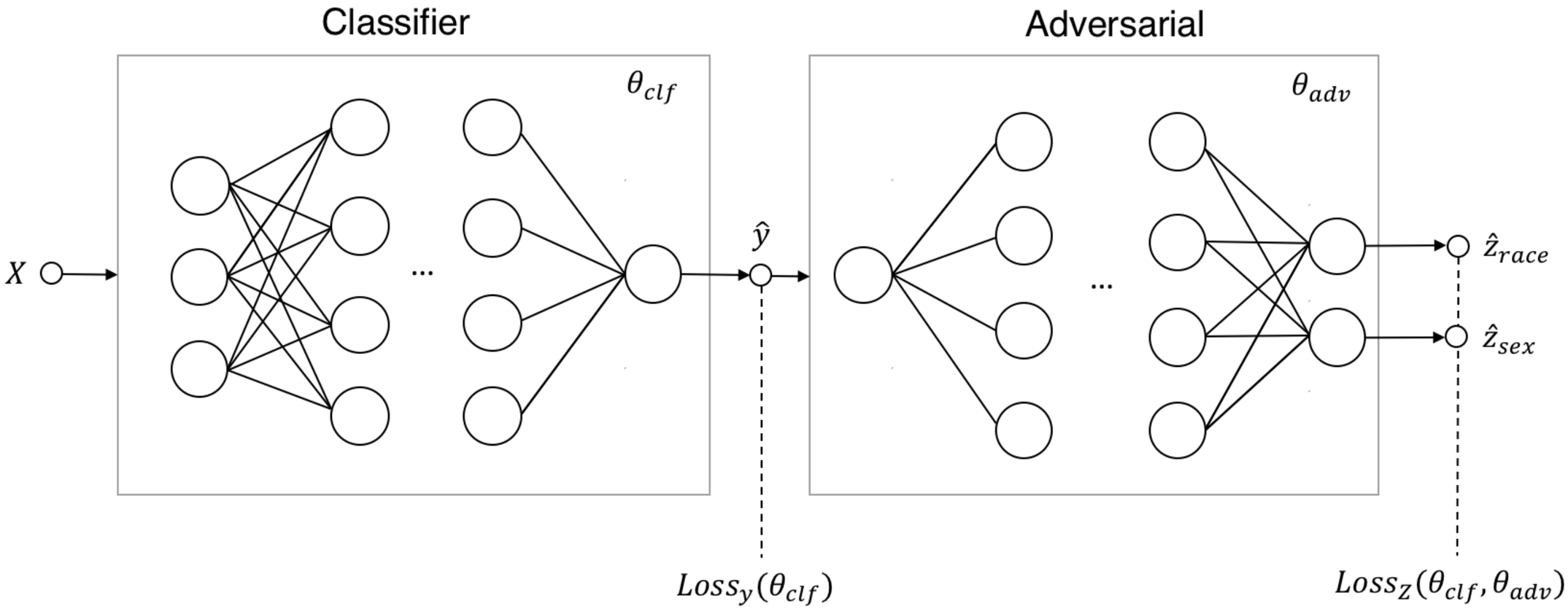
Internally **Matthijs Brouns** made an implementation.



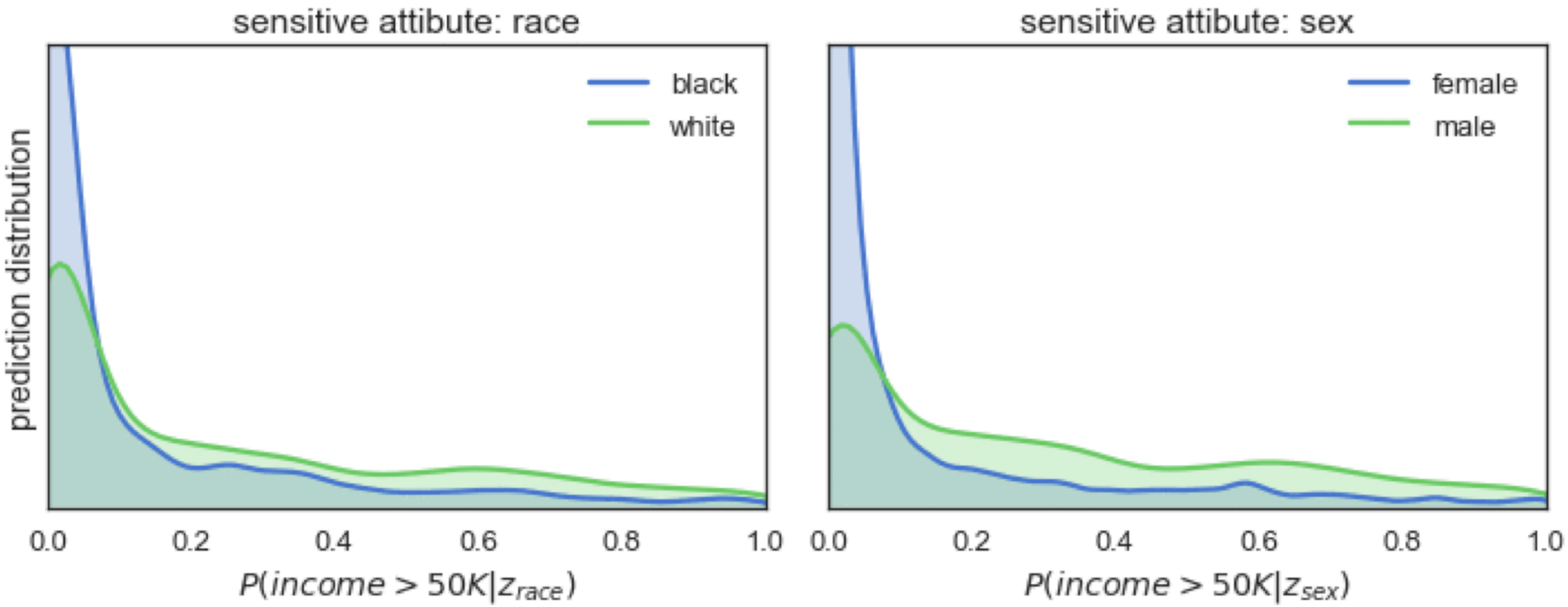


Stijn Tonk and **Henk Griffioen** made a neural variant.

Deep Learning Alternative via Cost Function



Deep Learning Alternative via Cost Function



Training iteration #1

Prediction performance:
- ROC AUC: 0.91
- Accuracy: 85.1

Satisfied p%-rules:
- race: 41%-rule
- sex: 36%-rule

What else can we do?

We might add constraints for detection!

$$\min \text{loss}(f(\theta|X), y))$$

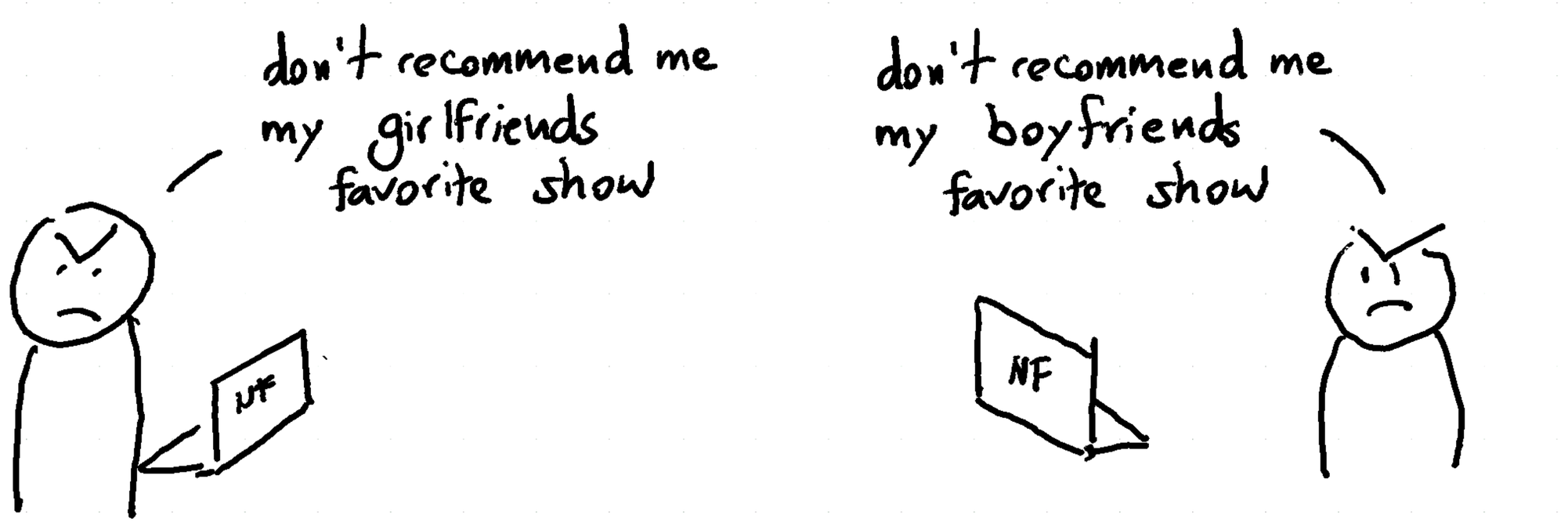
$$\text{subject to}$$

$$p(y = 1|x_i) \geq \ 0.9$$

$$p(y = 0|x_j) \geq \ 0.9$$

Imagine

Possible usecase.



The World Opens Up

Numeric issues aside; when you think about models this way a whole new world of modelling opens up.

Instead of overfitting our design concerns to AUC/ ROC we can actually design systems by thinking like this.

We get to actually **design** again. Fixing a problem is a lot easier when you can design a solution yourself.

Design and Fashion Statements

When I look for clothing, I try to find clothing that fits me. When I see a t-shirt that I really like it's pretty useless unless it fits me very well.

A bespoke suit needs to fit me. There is no way to get a "better" Vincent that fits a suit. This is why a proper tailor needs to spend time measuring me.

This is what `OneSizeForAll().fit()` feels like.

Design and Fashion Statements

I'll be making an argument for an attitude towards designing away from `model.fit()` to something more bespoke: `tailor.model()`.

When tailoring a model is there's less unexpected issues. We also have the opportunity to design a model that is **articulate**. You can easily answer questions about the assumptions without resorting to mere metrics.

Let's just ... take a step back.

There's a common theme to everything here.

$$\text{model} = \text{learn from data} \times \text{constraints}$$

Let's just ... take a step back.

There's a common theme to everything here.

$$\text{model} = \text{learn from data} \times \text{constraints}$$

Can you see it?

Let's just ... take a step back.

There's a common theme to everything here.

$$\text{model} = \text{learn from data} \times \text{constraints}$$

Can you see it?

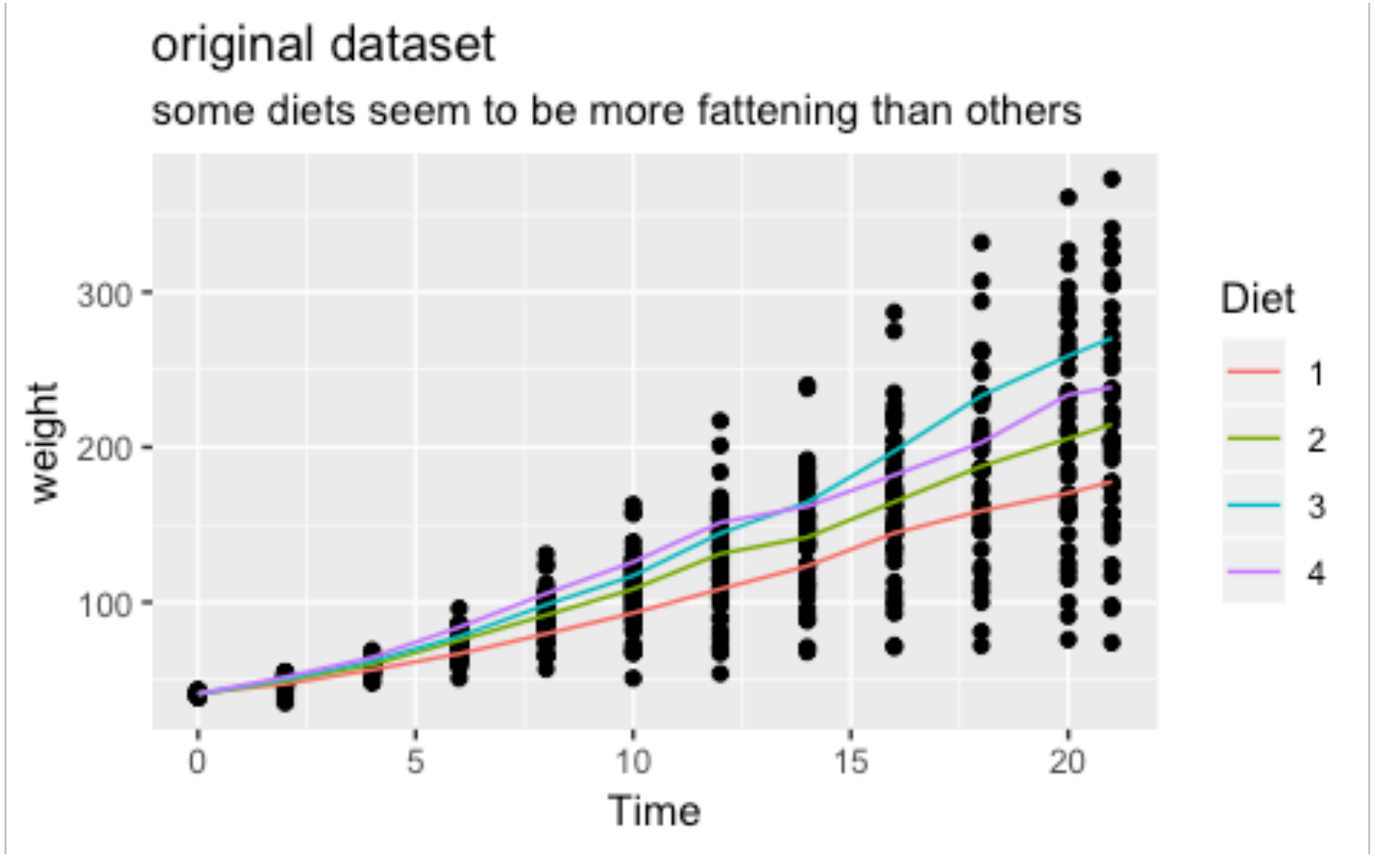
$$\text{model} = p(D|\theta)p(\theta)$$

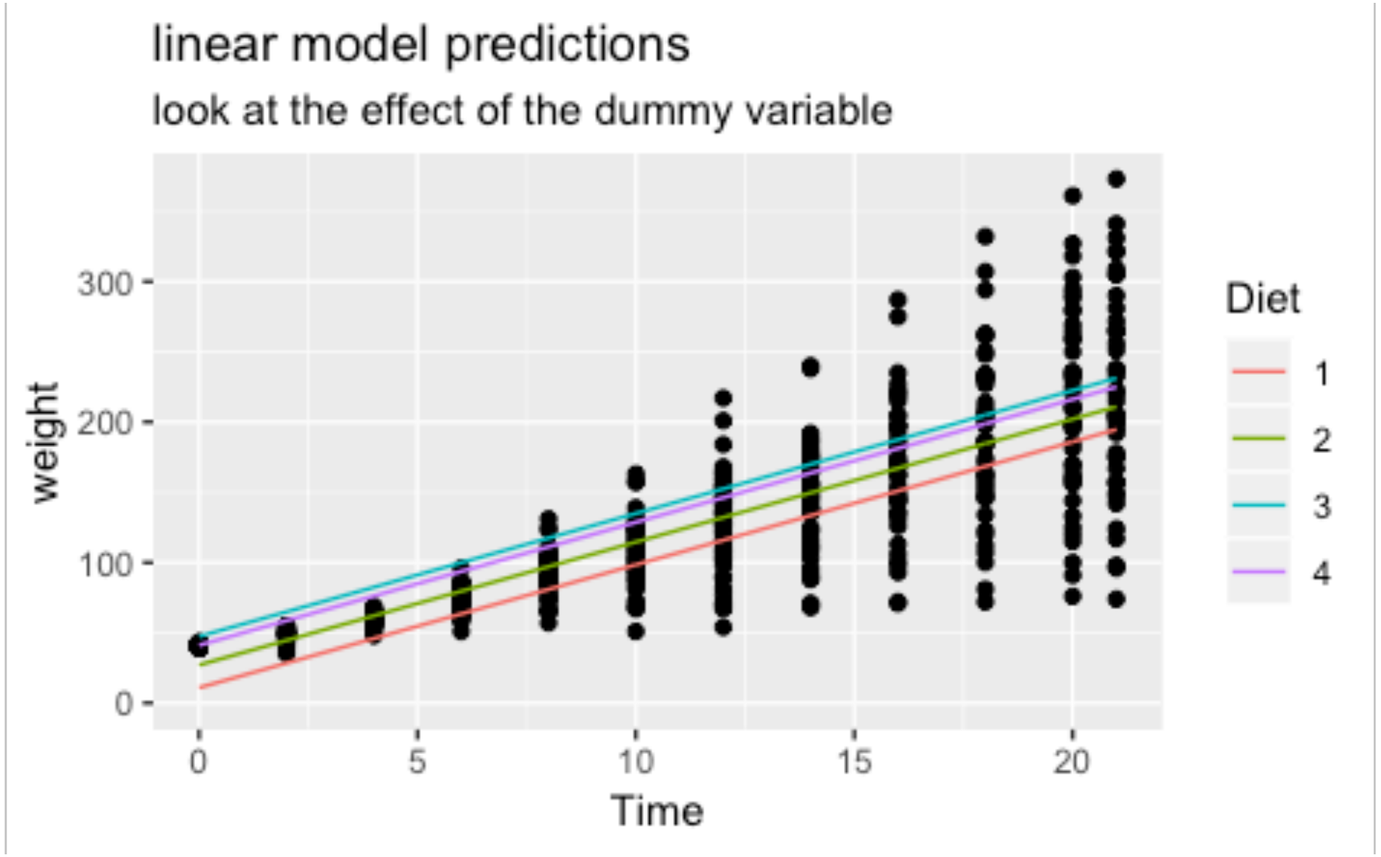
Maybe ... we need to ActuallyModel[tm]?

Just like that bespoke suit, maybe we need to accept that merely calling `model.fit()` is incredibly naive.

Perhaps we need to do what people in physics do. Try to study a system such that can come up with a proper model where we can explain everything.

I'll give one final example of this. It involves chickens.

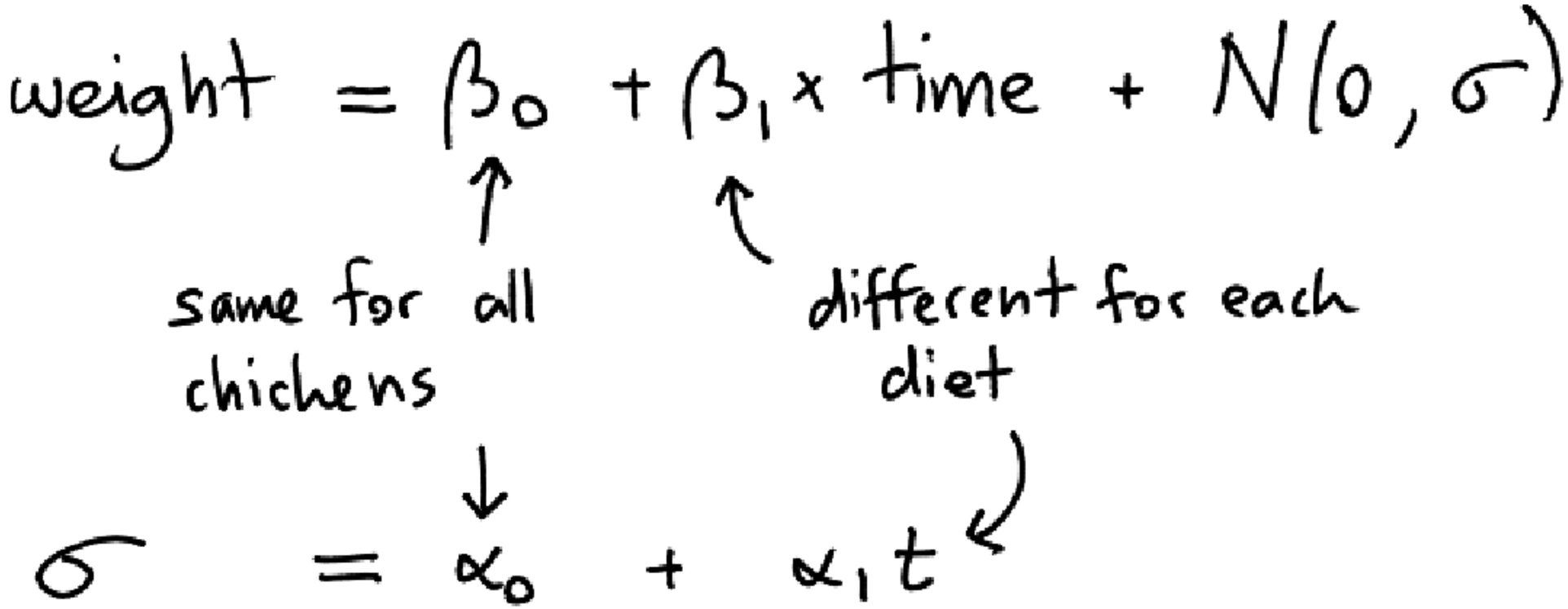

original dataset
some diets seem to be more fattening than others
weight
300
200
100
0
5
10
15
20
Time
Diet
1
2
3
4

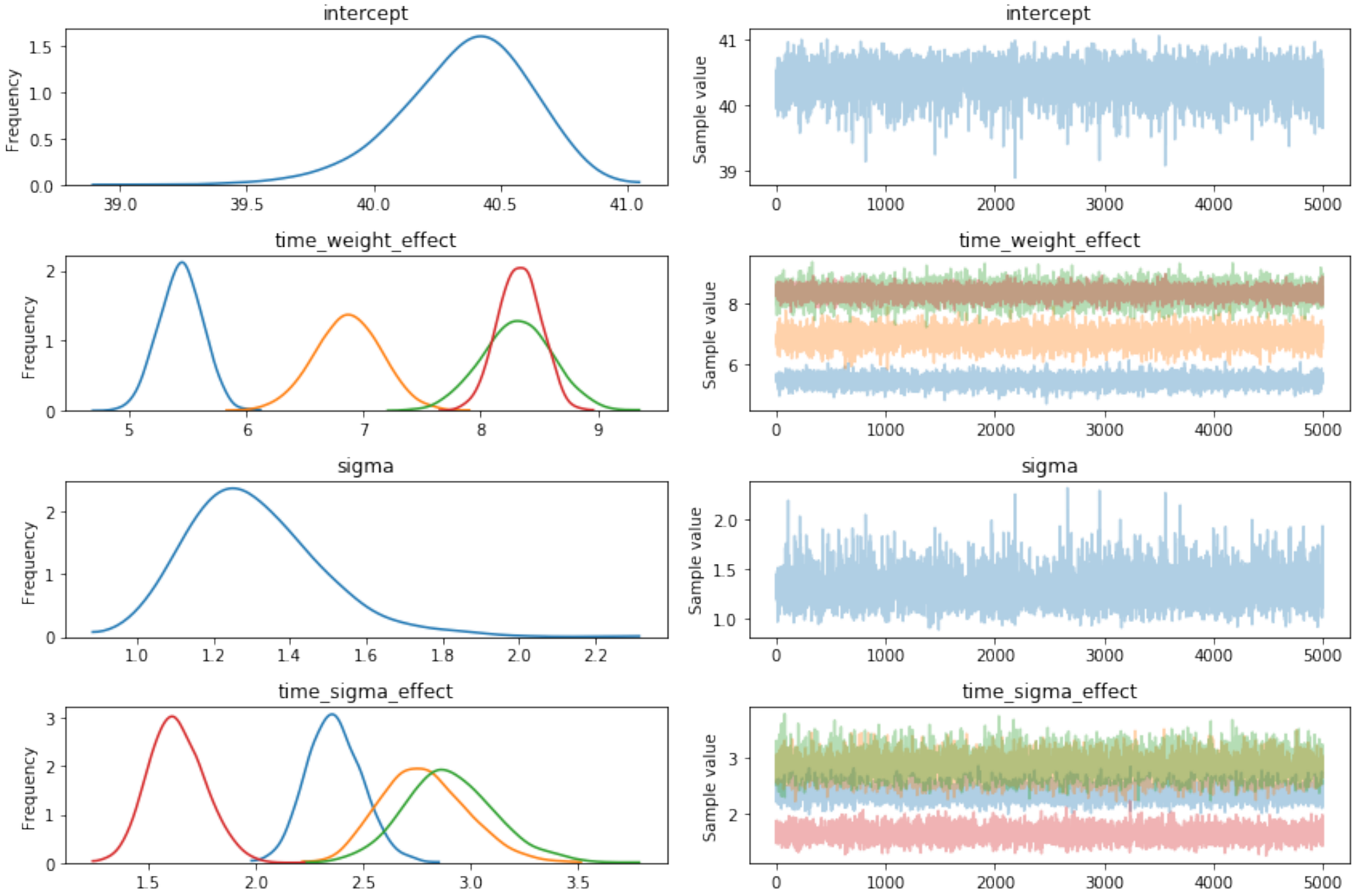

linear model predictions
look at the effect of the dummy variable
weight
300
200
100
0
0
5
10
15
20
Time
Diet
1
2
3
4

$$\text{weight} = \beta_0 + \beta_1 \times \text{time} + N(0, \sigma)$$

same for all chichens

different for each diet


weight = β0 + β1 x time + N(0, σ)
same for all chickens
different for each diet
σ = α0 + α1 t


intercept
Frequency
Sample value
time_weight_effect
sigma
time_sigma_effect

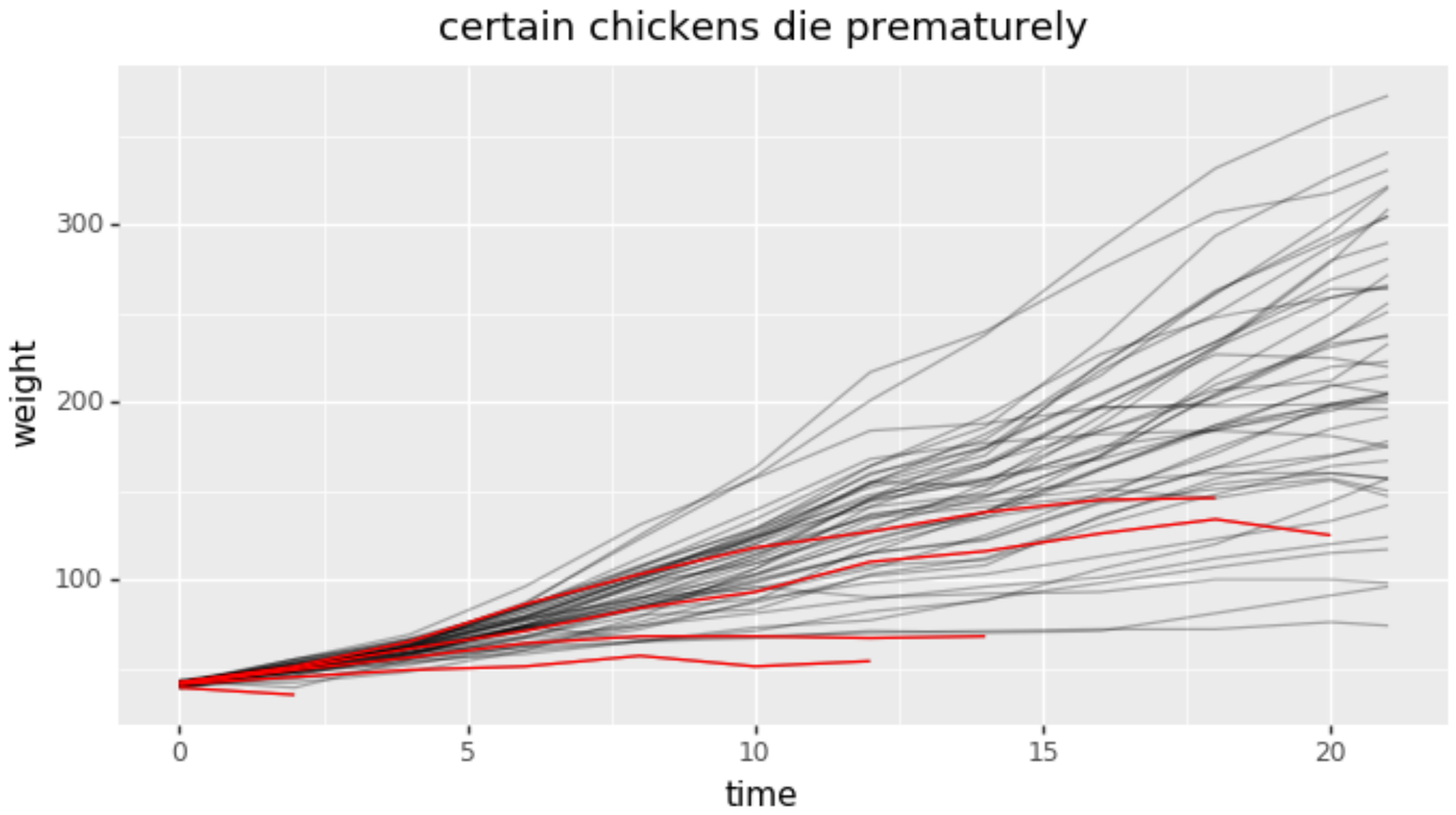
Benefits

— Perfect Explainability
— More Options for "What If"-type questions
— Useful for Hypothesis Tests

Downsides

— Human Bias? Something far worse?

But there is one problem



Models Won't Be Perfect

The entire talk I've spent talking about side effects of our models that go unnoticed if you're not careful.

But even the "best" model can fail if we don't understand the problem we are solving. We might be building a system that produced dead chickens.

But there are some remedies.

— Consider *not* predicting. It's a great way to remove risk, but be careful in your definition of certainty.

— Consider constraining features. Optimality goes further than ROC/AUC/MSE.

— Consider models that constrain, they are great.

— Bayesian systems are more articulate. Try them.

But most importantly.

— Take a step back and think about what you're doing.

— Consider *not* predicting. It's a great way to remove risk, but be careful in your definition of certainty.

— Consider constraining features. Optimality goes further than ROC/AUC/MSE.

— Consider models that constrain, they are great.

— Bayesian systems can give you freedom.

But most importantly.

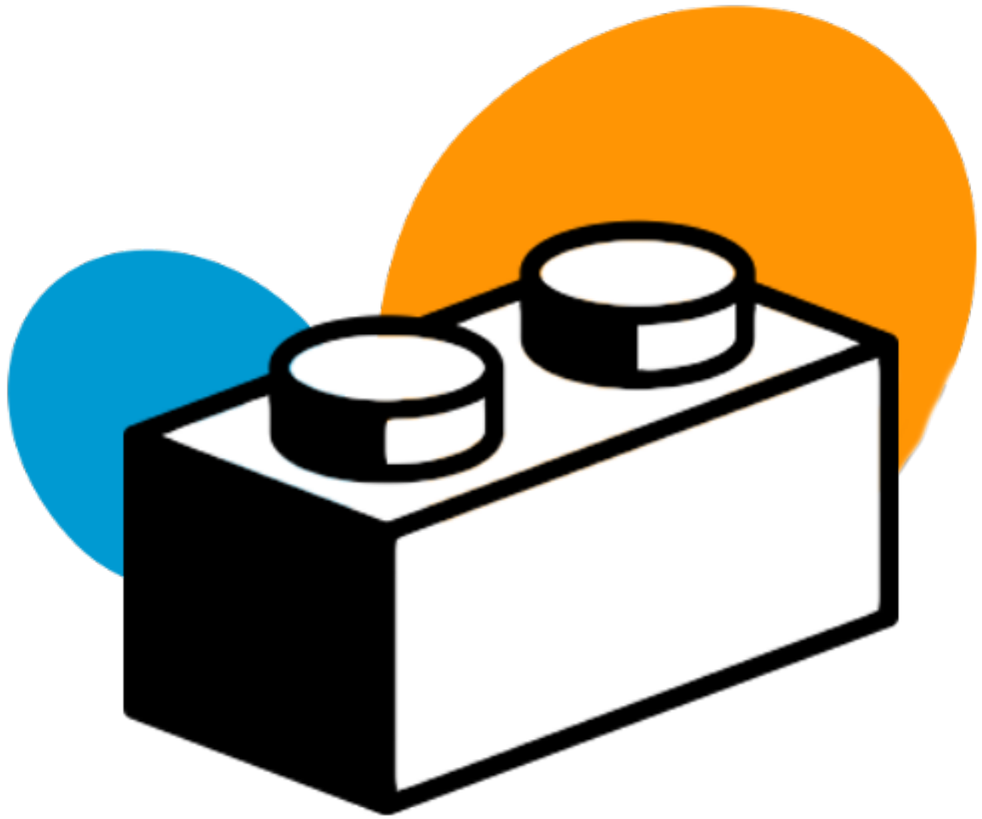
— Take a step back and think about what you're doing. *Beware the dead chickens.*

Natural Intelligence isn't such a bad idea.

Natural Intelligence isn't such a bad idea.

Please grant yourself the creative freedom to understand the problem. This will help you a great deal in designing the solution.

One more thing ...



Thanks!

Get 'yer stickers at NumFocus stand