ML and Society

Feb 24, 2022

Please have a face mask on

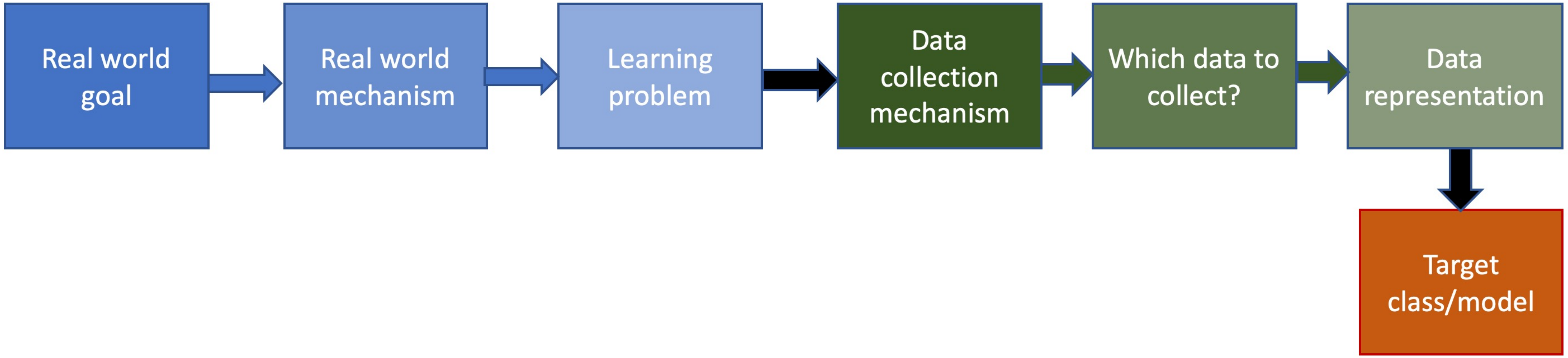
Masking requirement

Your face mask **must cover** your nose and mouth at all times.

UB requires all students, employees and visitors – regardless of their vaccination status – to wear face coverings while inside campus buildings.

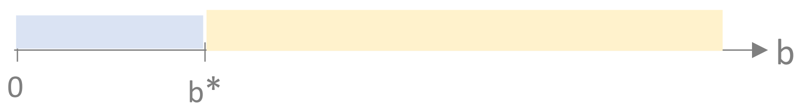
https://www.buffalo.edu/coronavirus/health-and-safety/health-safety-guidelines.html

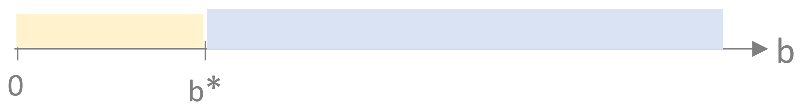
ML model classes



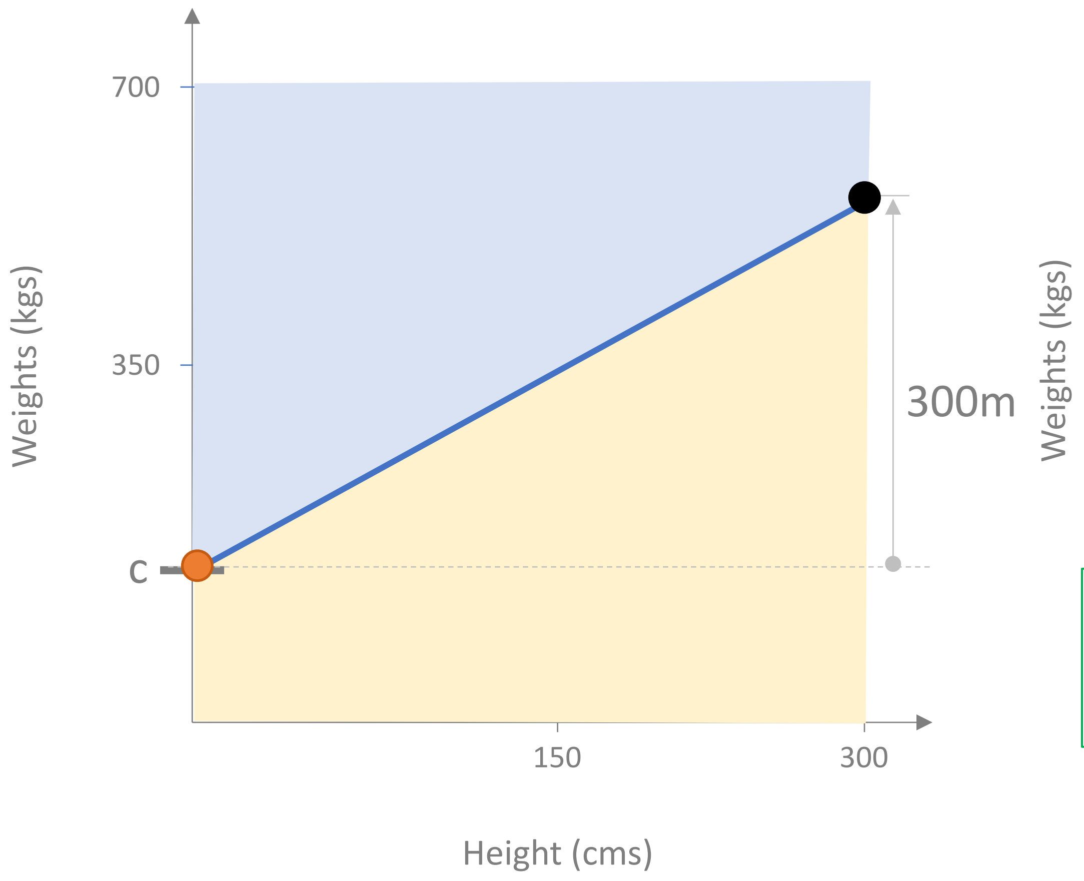
Linear model for one variable

$$\ell(b) = \begin{cases} 1 & \text{if } b < b^* \\ -1 & \text{if } b \geq b^* \end{cases} .$$





Linear model in two variables



$$\ell(w,h) = \begin{cases} 1 & \text{if } w \geq m \cdot h + c \\ -1 & \text{if } w < m \cdot h + c \end{cases}$$

Properties

$$\ell(w,h) = \begin{cases} 1 & \text{if } w \geq m \cdot h + c \\ -1 & \text{if } w < m \cdot h + c \end{cases}$$

Efficient training ✅

Efficient prediction ✅

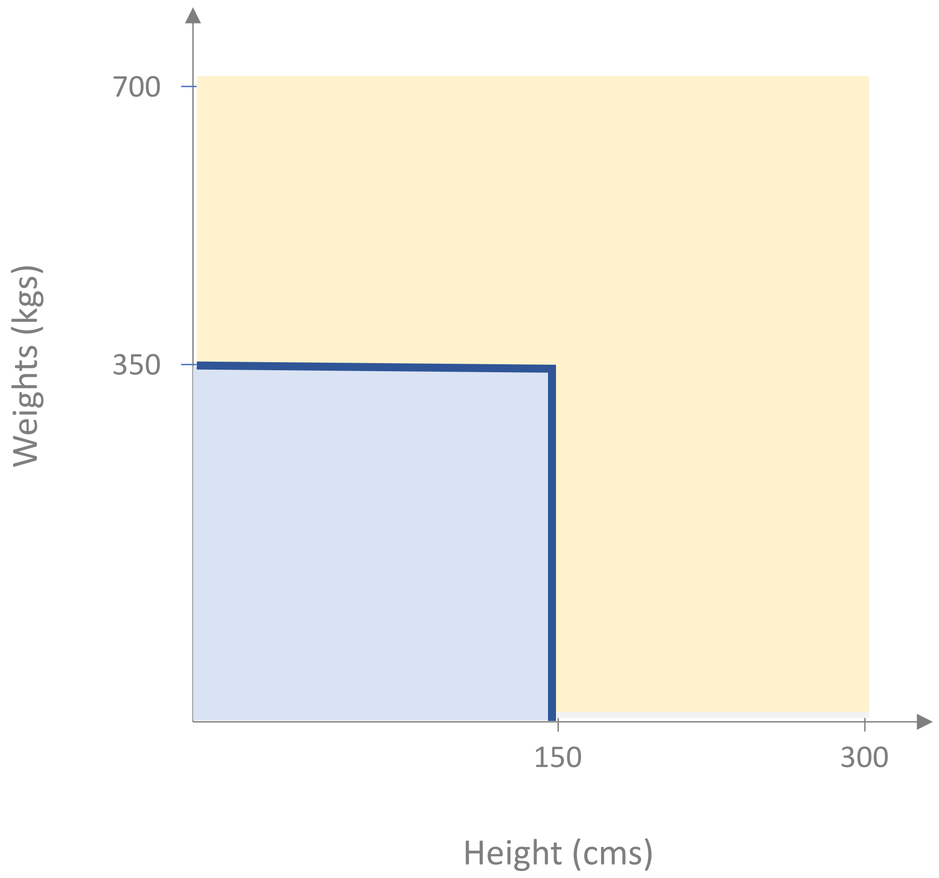
Expressivity ❌

Explainability ✅

Decision tree models

Heard of 20 Questions?

We already have seen a decision tree

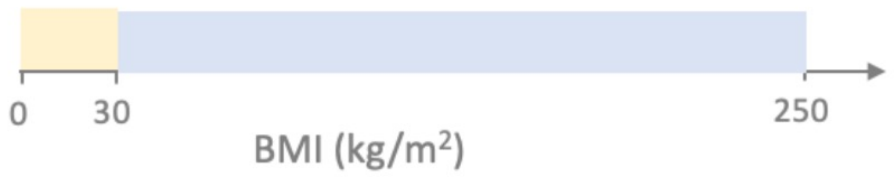


$$f(w, h) = \begin{cases} 1 & \text{if } w < w^* \text{ and } h < h^* \\ -1 & \text{otherwise} \end{cases}$$

Decision trees in one variable

All questions of the form $b \geq b^*$?

To focus our discussion, we will only consider the questions of the form-- $b \geq b^*$? I.e. for the single input variable b we can ask whether is is smaller than a fixed value b^* that we can choose as part of designing the question? Also we will allow the answers to be either 1 (for the case that $b \geq b^*$), or -1 (for the case $b < b^*$). For example, the result of the query $b \geq 30$ can be represented in a form that we have seen before:

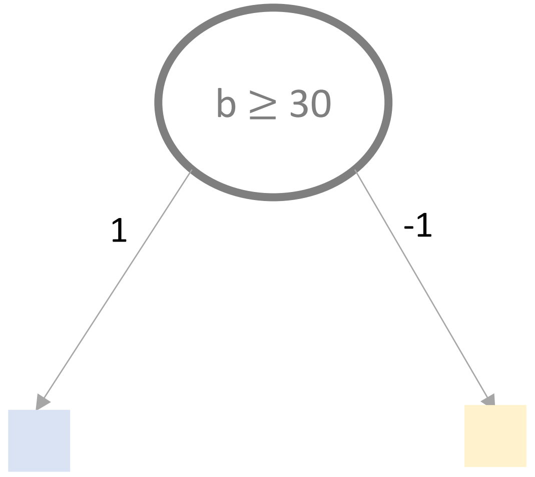


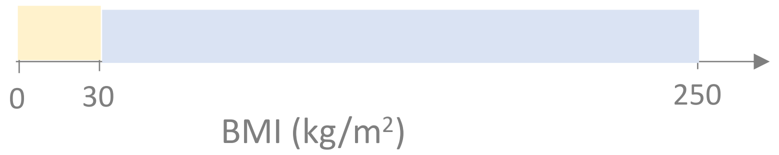
where the blue part represent the values b in which the question $b \geq 30$ will return 1 and the yellow part returns all values b for which the question $b \geq 30$ returns -1.

Decision trees in one variable (informal-ish definition)

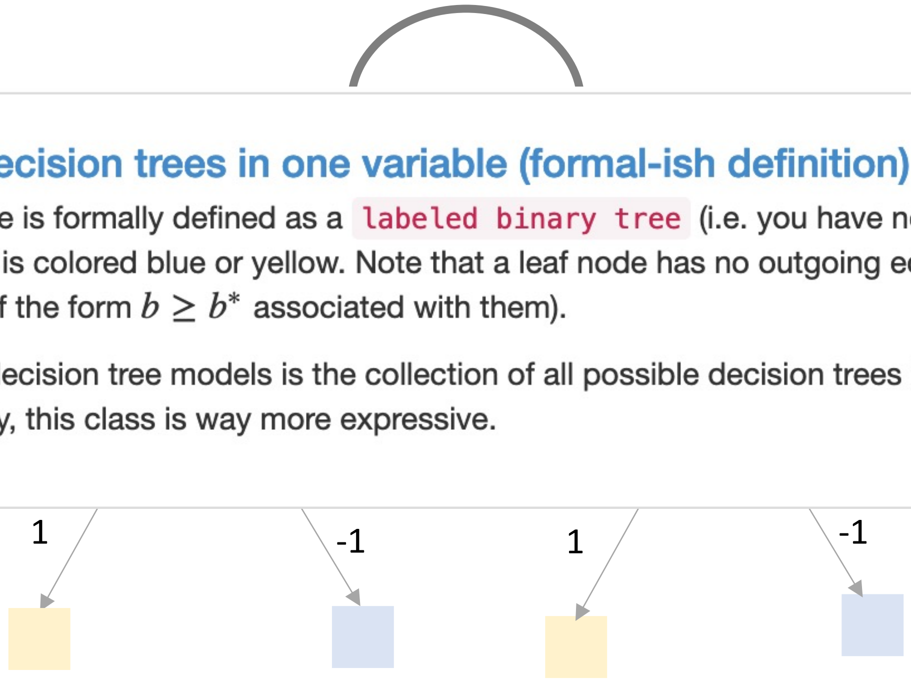
A decision tree (model) informally is a set of questions of the form $b \geq b^*$ and the model decides to color a point b blue or yellow based on the answers to the questions that were asked.

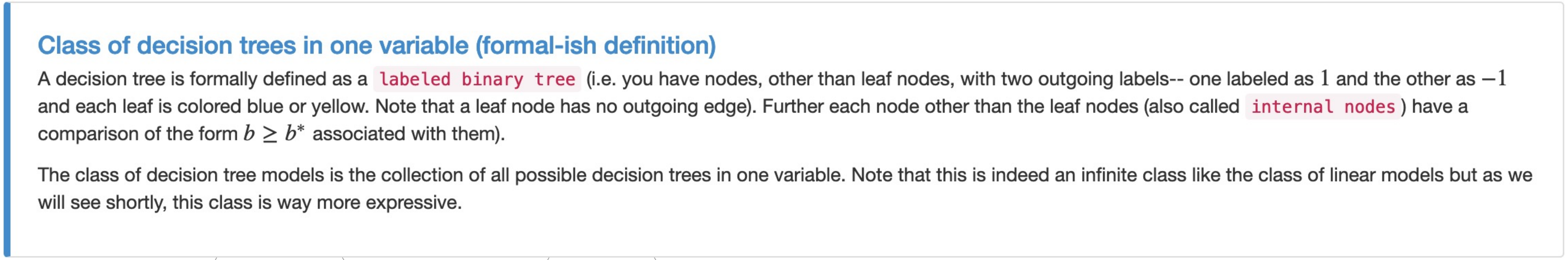
A different representation





Asking more questions



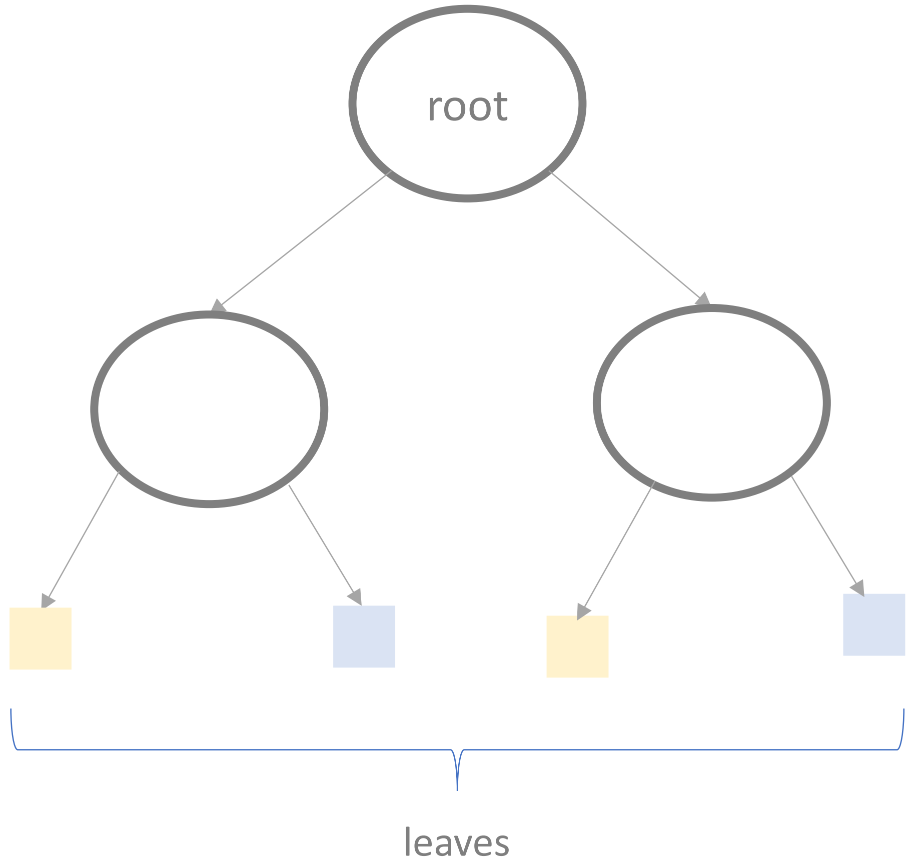


Class of decision trees in one variable (formal-ish definition)

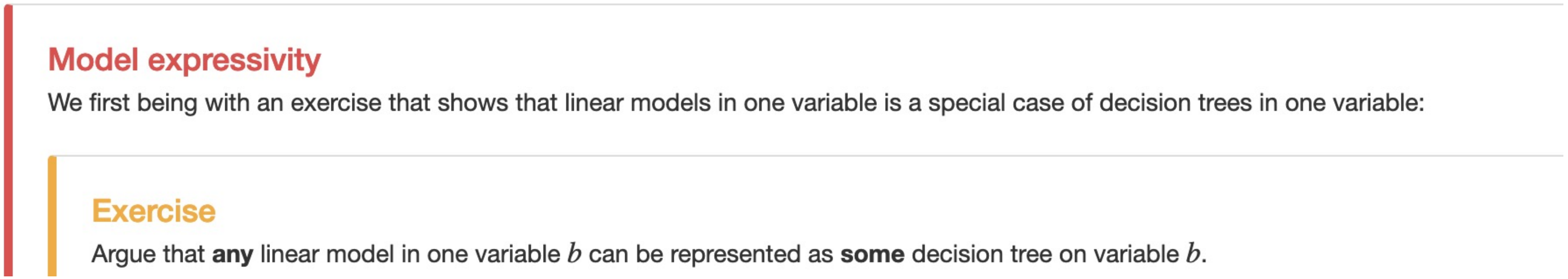
A decision tree is formally defined as a `labeled binary tree` (i.e. you have nodes, other than leaf nodes, with two outgoing labels-- one labeled as 1 and the other as -1 and each leaf is colored blue or yellow. Note that a leaf node has no outgoing edge). Further each node other than the leaf nodes (also called `internal nodes`) have a comparison of the form $b \geq b^*$ associated with them).

The class of decision tree models is the collection of all possible decision trees in one variable. Note that this is indeed an infinite class like the class of linear models but as we will see shortly, this class is way more expressive.

Why is this a tree?



Model expressivity



Model expressivity

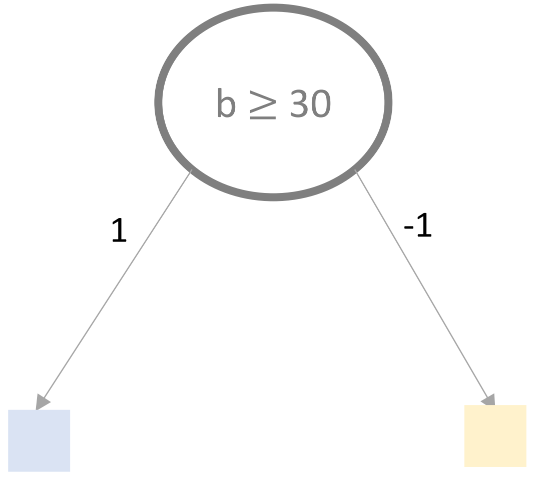
We first being with an exercise that shows that linear models in one variable is a special case of decision trees in one variable:

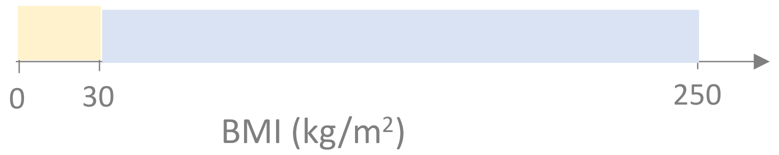
Exercise

Argue that **any** linear model in one variable b can be represented as **some** decision tree on variable b.

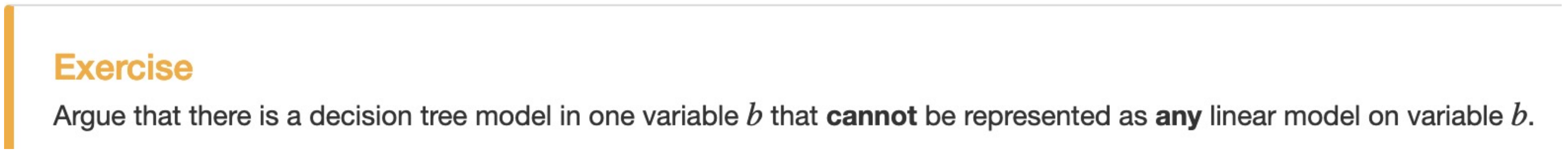
Why?

Remember this?





Strictly more powerful than linear models

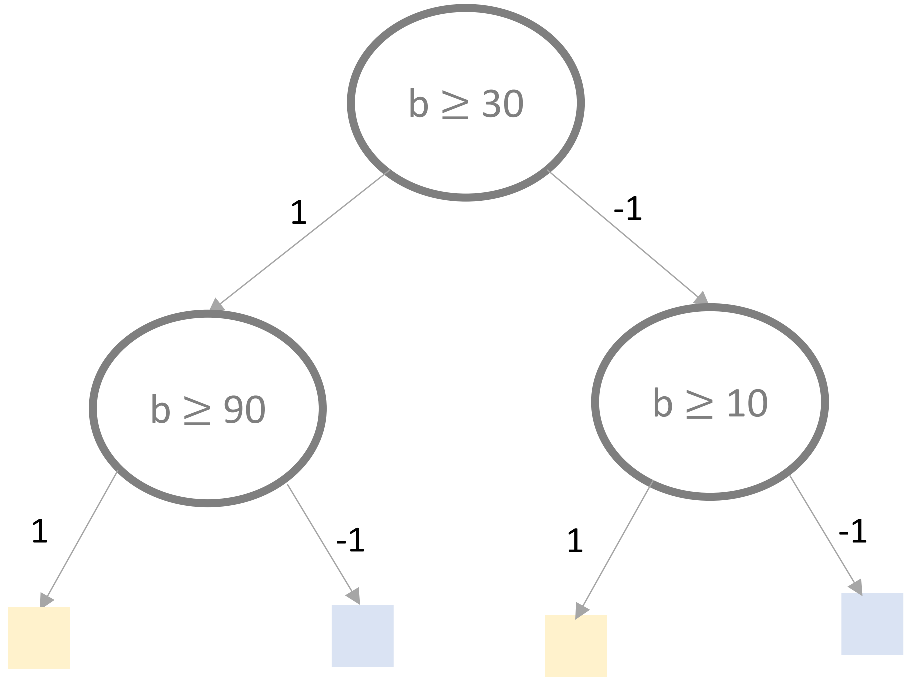


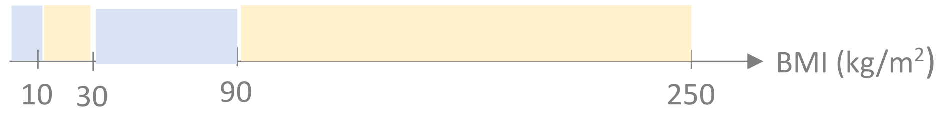
Exercise

Argue that there is a decision tree model in one variable b that **cannot** be represented as **any** linear model on variable b.

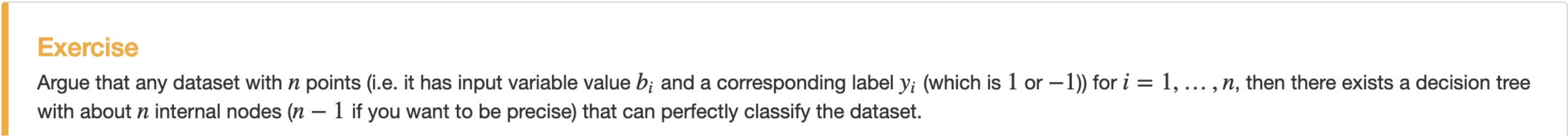
Why?

Remember this?





General result

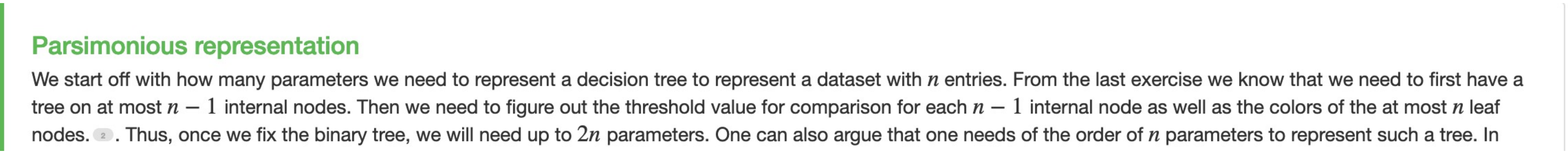


Exercise

Argue that any dataset with n points (i.e. it has input variable value b_i and a corresponding label y_i (which is 1 or -1)) for $i = 1, \ldots, n$, then there exists a decision tree with about n internal nodes ($n - 1$ if you want to be precise) that can perfectly classify the dataset.

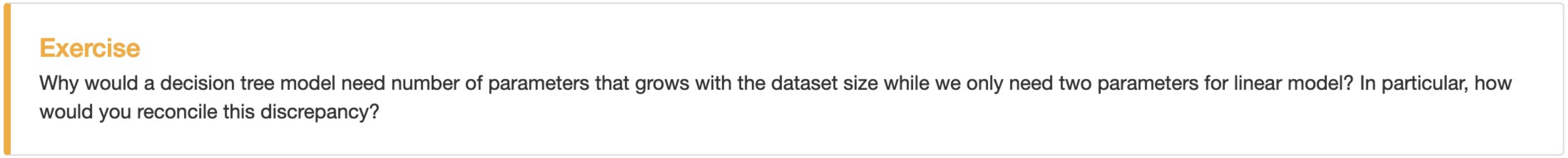
Model training

Parsimonious representation



Parsimonious representation

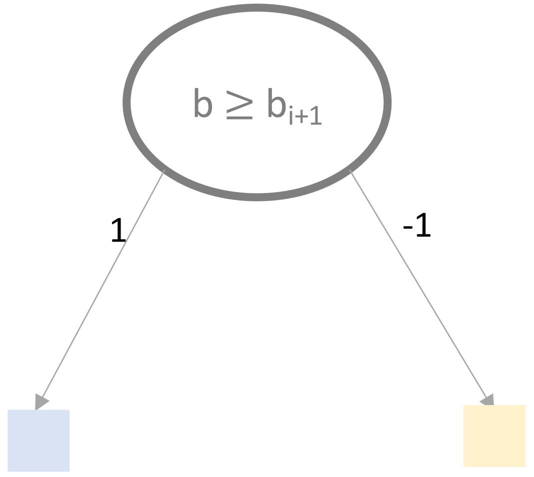
We start off with how many parameters we need to represent a decision tree to represent a dataset with n entries. From the last exercise we know that we need to first have a tree on at most $n-1$ internal nodes. Then we need to figure out the threshold value for comparison for each $n-1$ internal node as well as the colors of the at most n leaf nodes.[2]. Thus, once we fix the binary tree, we will need up to $2n$ parameters. One can also argue that one needs of the order of n parameters to represent such a tree. In

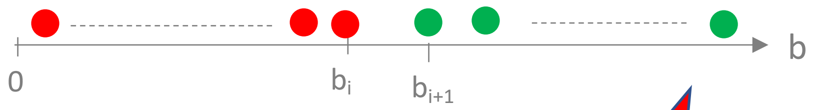


Exercise

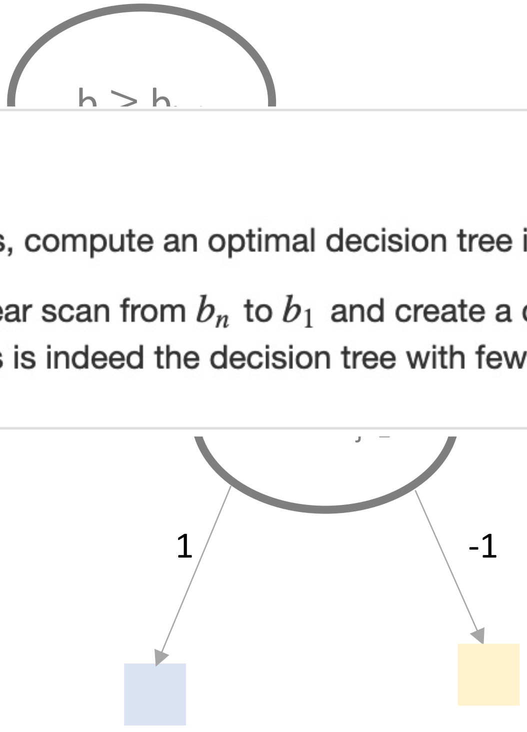
Why would a decision tree model need number of parameters that grows with the dataset size while we only need two parameters for linear model? In particular, how would you reconcile this discrepancy?

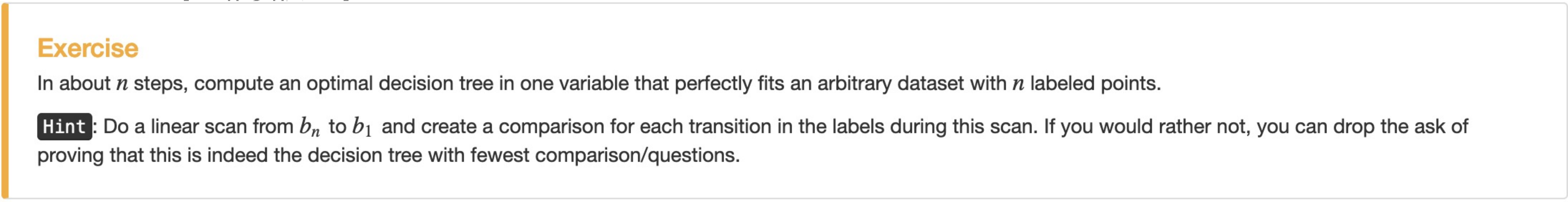
Model training





Two transition points





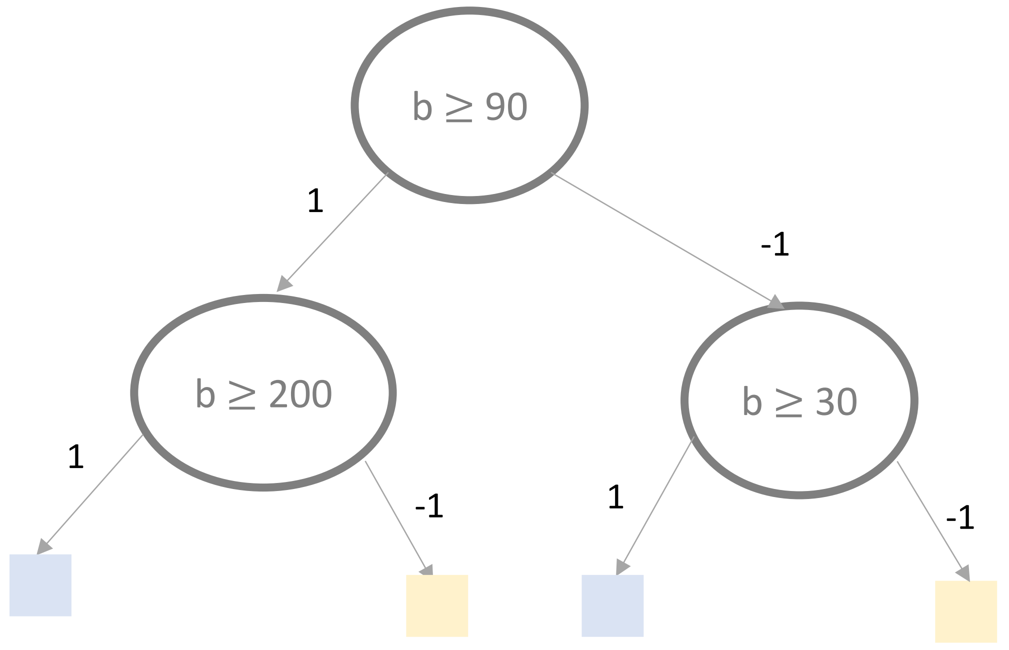
Exercise

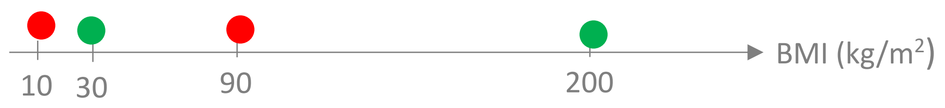
In about n steps, compute an optimal decision tree in one variable that perfectly fits an arbitrary dataset with n labeled points.

`Hint`: Do a linear scan from b_n to b_1 and create a comparison for each transition in the labels during this scan. If you would rather not, you can drop the ask of proving that this is indeed the decision tree with fewest comparison/questions.

Decision tree?

Prediction

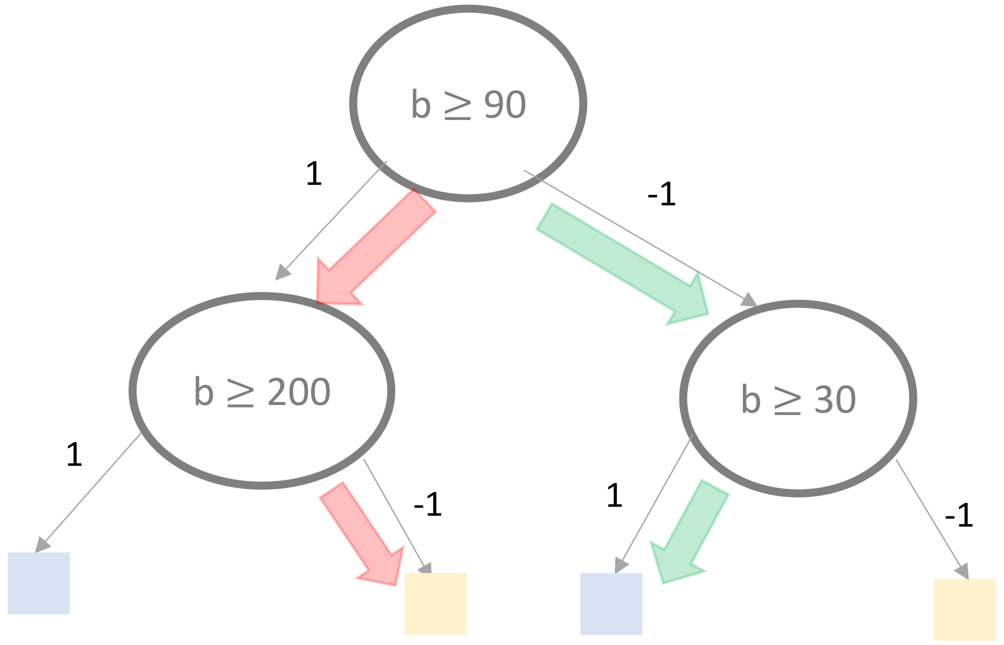


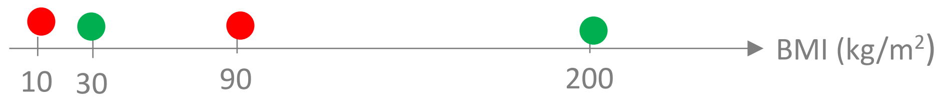


Efficient prediction

As with the linear model, the prediction algorithm naturally follows from the definition of the decision tree. In particular, given a value b, start with the root see if doing the comparison with value b gives an answer of 1 or -1 and based on the answer the correspondingly labeled edge. If you reach a leaf node, then you just output the color of the leaf otherwise you repeat the whole procedure you did at the root.

$b = 150$ and $b = 70$

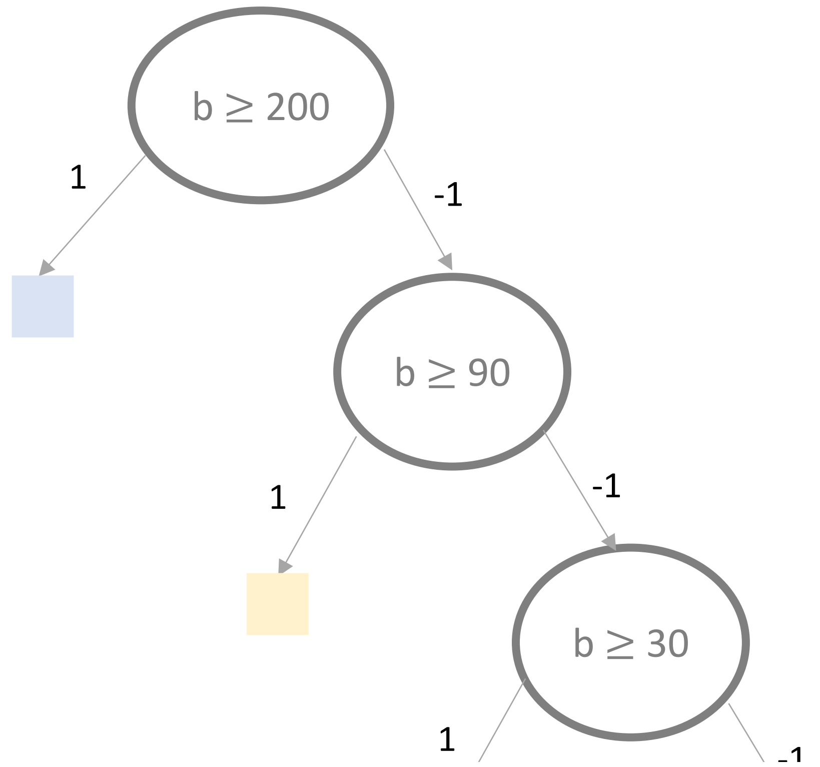


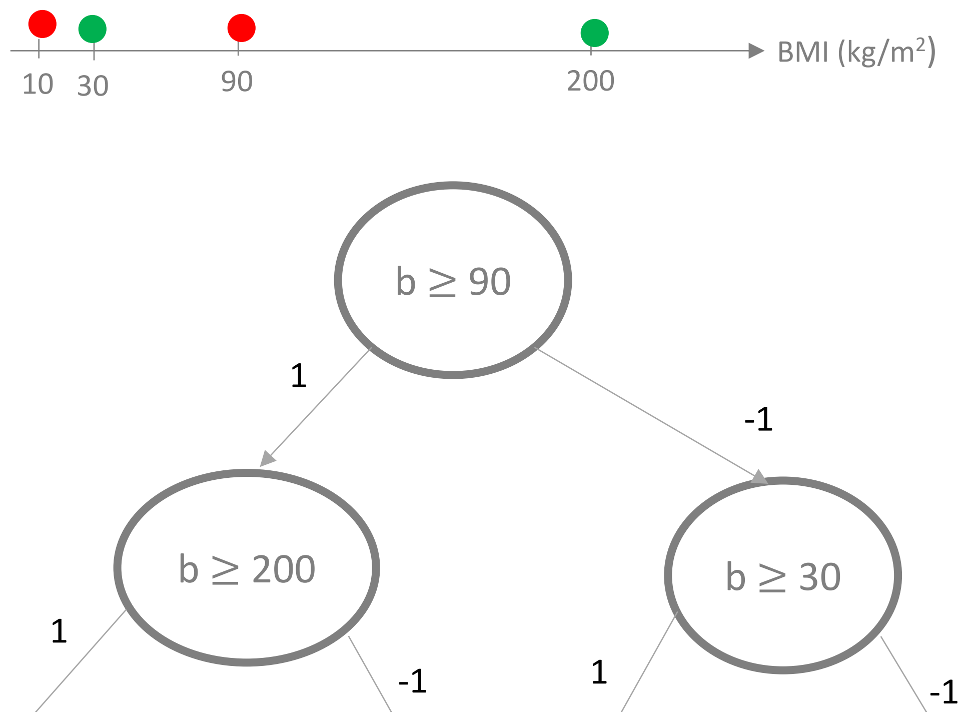


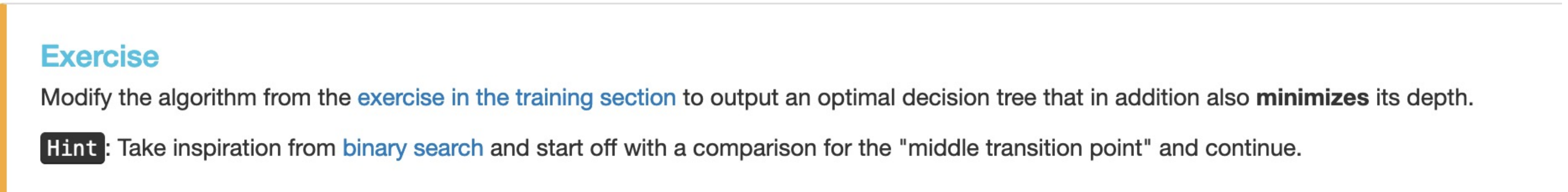
Exercise

The number of steps needed to compute the decision tree models prediction on b is the number of comparisons made in the algorithm above. In particular, this is the same as the number of edges needed to go from the root node to the relevant leaf node.

Multiple decisions trees with different depth





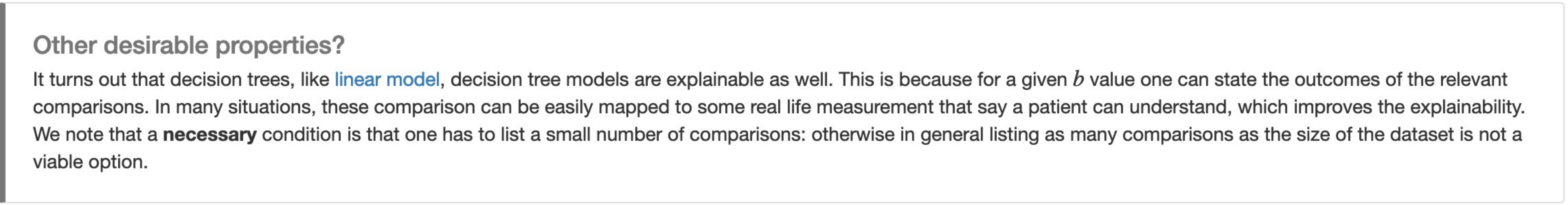


Exercise

Modify the algorithm from the exercise in the training section to output an optimal decision tree that in addition also **minimizes** its depth.

`Hint`: Take inspiration from binary search and start off with a comparison for the "middle transition point" and continue.

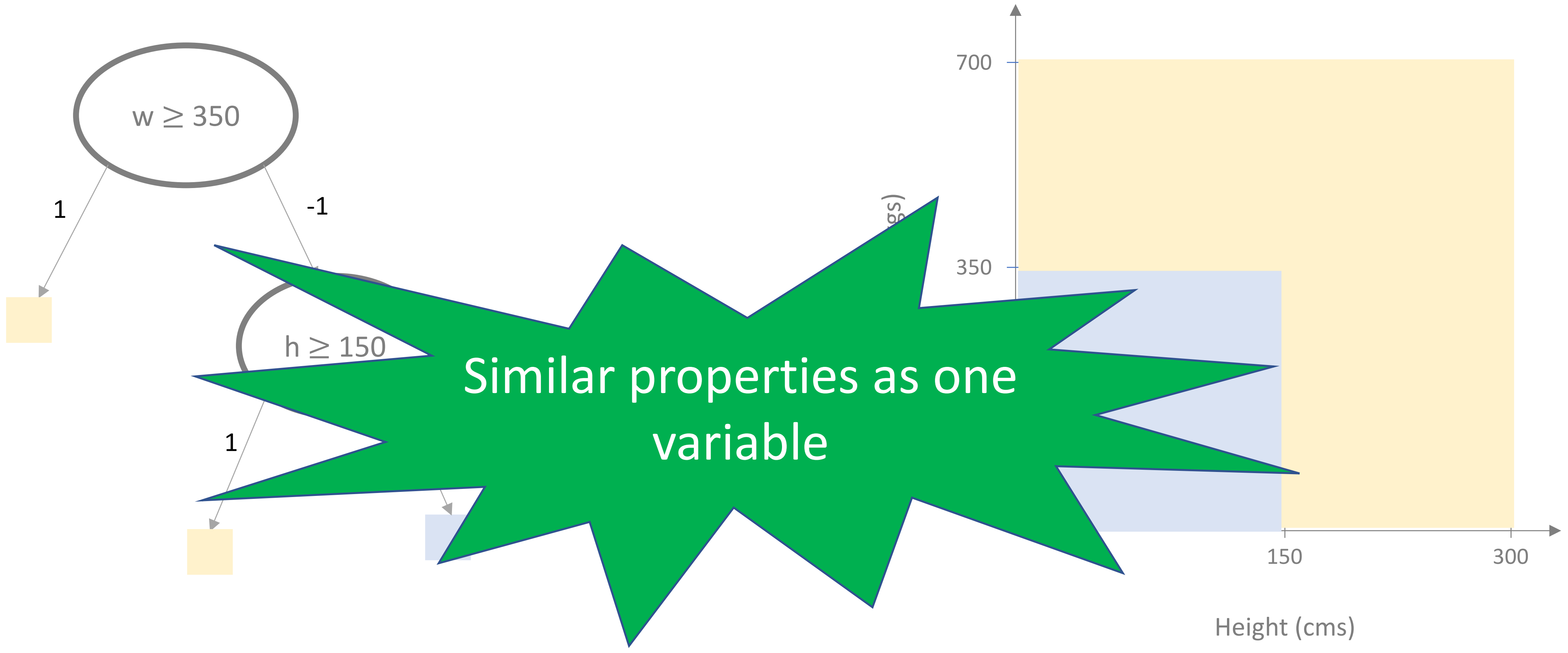
Other properties?



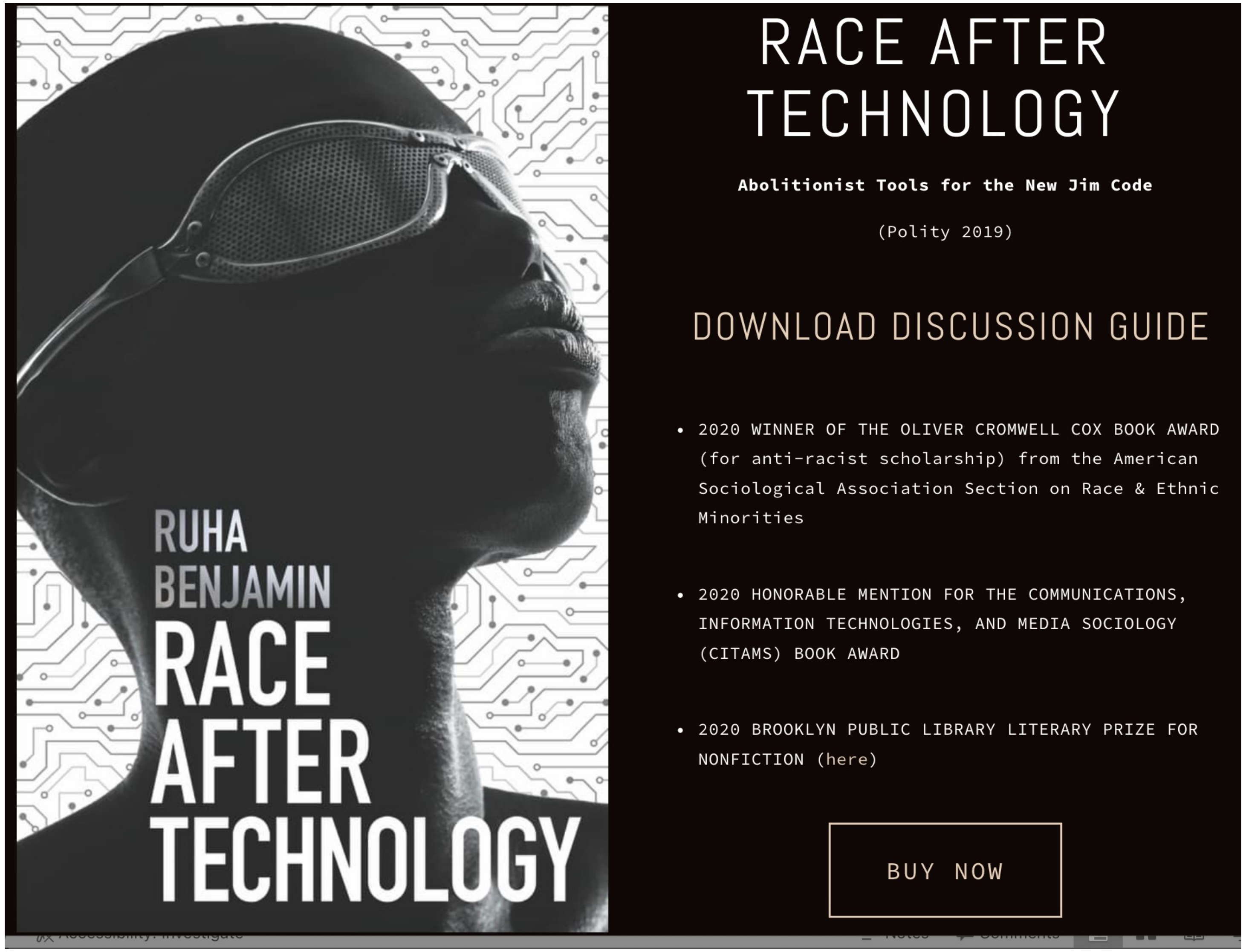
Other desirable properties?

It turns out that decision trees, like linear model, decision tree models are explainable as well. This is because for a given b value one can state the outcomes of the relevant comparisons. In many situations, these comparison can be easily mapped to some real life measurement that say a patient can understand, which improves the explainability. We note that a **necessary** condition is that one has to list a small number of comparisons: otherwise in general listing as many comparisons as the size of the dataset is not a viable option.

Decision trees in two variables



Today’s pass phrase: Ruha Benjamin



RUHA BENJAMIN
RACE AFTER TECHNOLOGY

RACE AFTER TECHNOLOGY

Abolitionist Tools for the New Jim Code

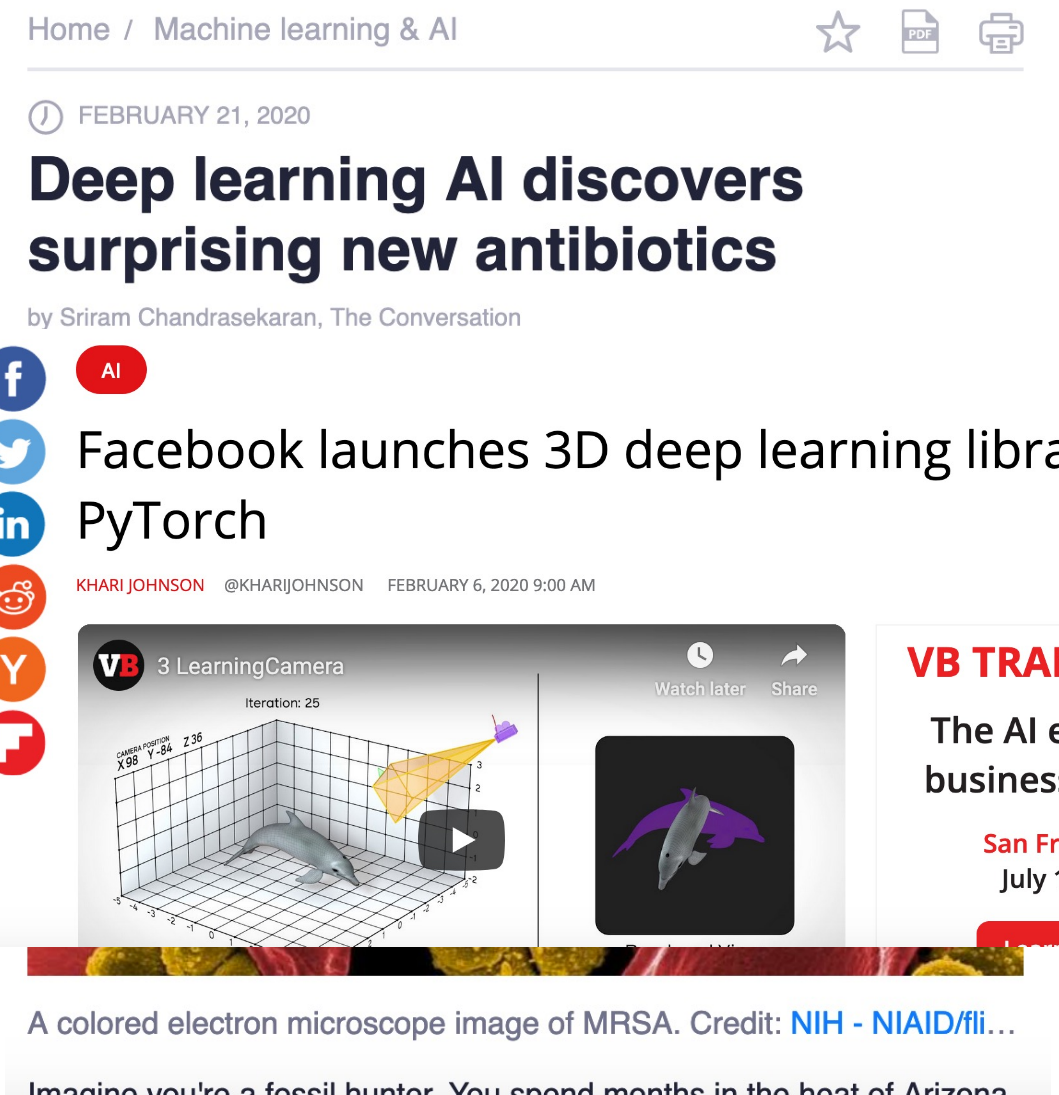
(Polity 2019)

DOWNLOAD DISCUSSION GUIDE

- 2020 WINNER OF THE OLIVER CROMWELL COX BOOK AWARD (for anti-racist scholarship) from the American Sociological Association Section on Race & Ethnic Minorities
- 2020 HONORABLE MENTION FOR THE COMMUNICATIONS, INFORMATION TECHNOLOGIES, AND MEDIA SOCIOLOGY (CITAMS) BOOK AWARD
- 2020 BROOKLYN PUBLIC LIBRARY LITERARY PRIZE FOR NONFICTION (here)

BUY NOW

Last model class: Neural networks (NNs)



Home / Machine learning & AI

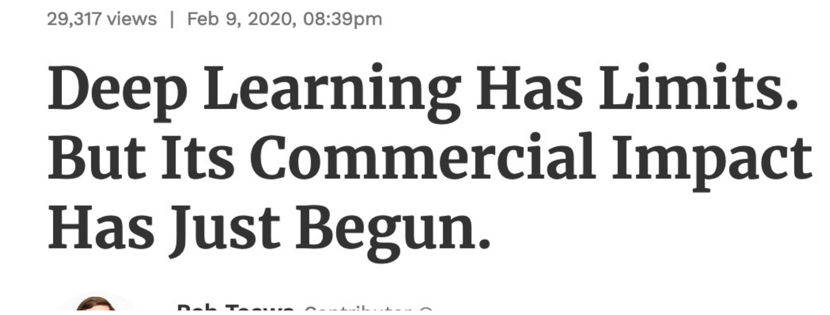
FEBRUARY 21, 2020

Deep learning AI discovers surprising new antibiotics

by Sriram Chandrasekaran, The Conversation

A colored electron microscope image of MRSA. Credit: NIH - NIAID/fli…

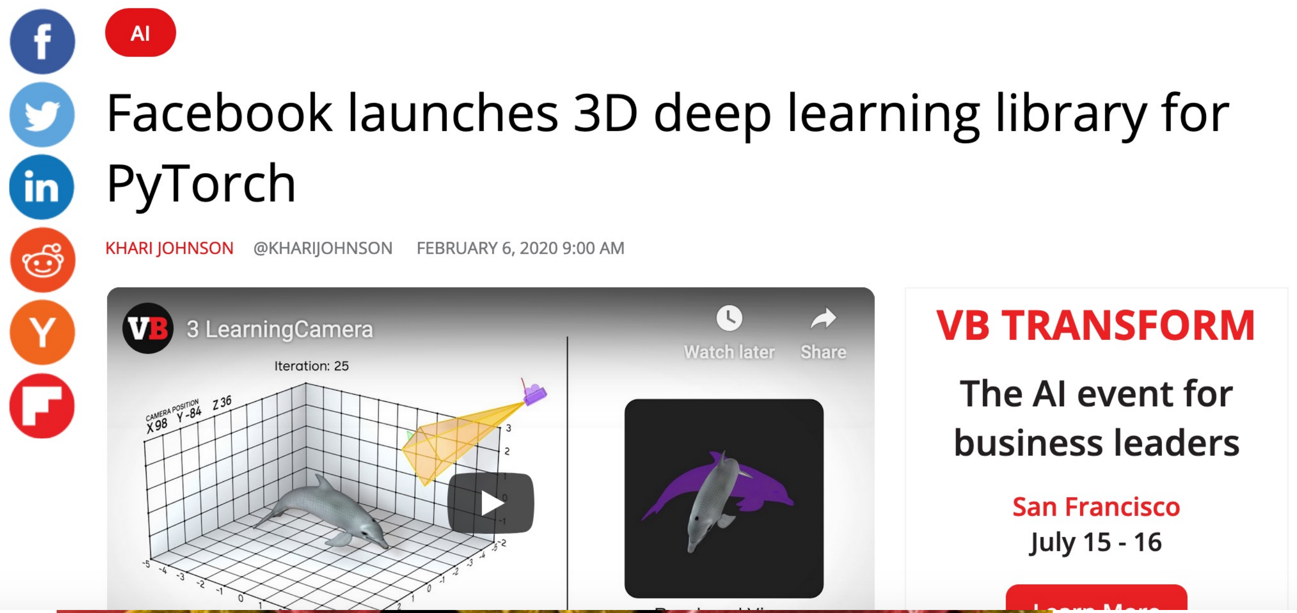
Imagine you're a fossil hunter. You spend months in the heat of Arizona



29,317 views | Feb 9, 2020, 08:39pm

Deep Learning Has Limits. But Its Commercial Impact Has Just Begun.

Rob Toews Contributor



AI

Facebook launches 3D deep learning library for PyTorch

KHARI JOHNSON @KHARIJOHNSON FEBRUARY 6, 2020 9:00 AM

3 LearningCamera

Iteration: 25

Watch later

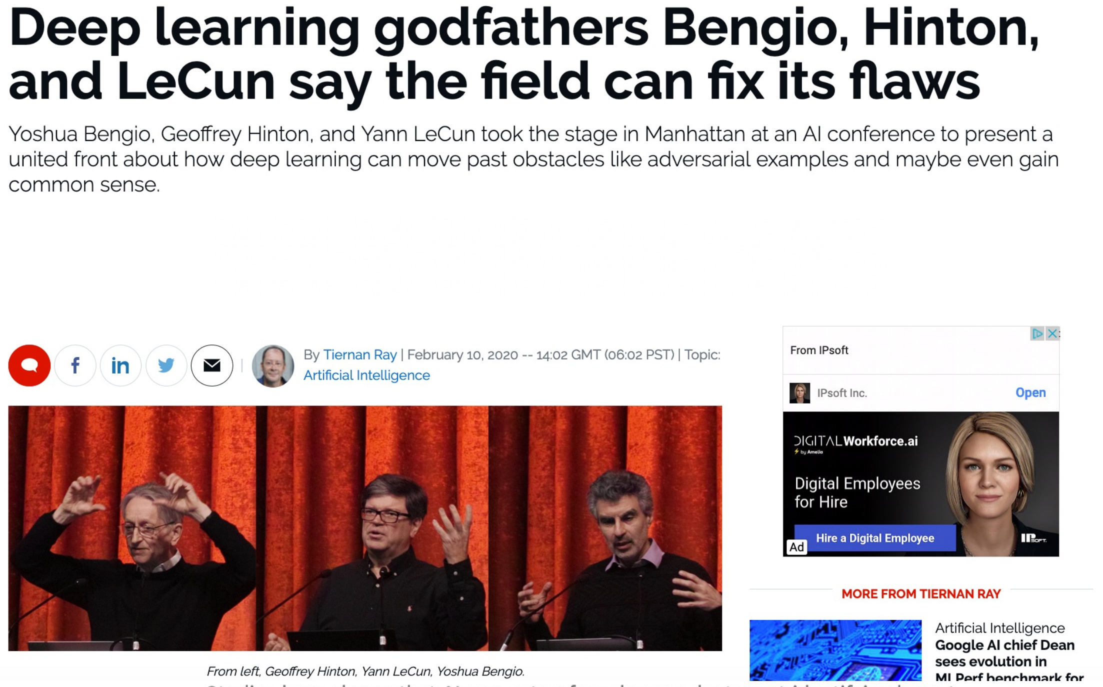
Share

VB TRANSFORM

The AI event for business leaders

San Francisco

July 15 - 16



Deep learning godfathers Bengio, Hinton, and LeCun say the field can fix its flaws

Yoshua Bengio, Geoffrey Hinton, and Yann LeCun took the stage in Manhattan at an AI conference to present a united front about how deep learning can move past obstacles like adversarial examples and maybe even gain common sense.

By Tiernan Ray | February 10, 2020 -- 14:02 GMT (06:02 PST) | Topic: Artificial Intelligence

From IPsoft

IPsoft Inc. Open

DIGITAL Workforce.ai

Digital Employees for Hire

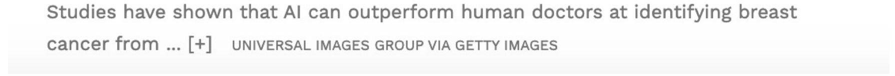
Hire a Digital Employee

MORE FROM TIERNAN RAY

Artificial Intelligence

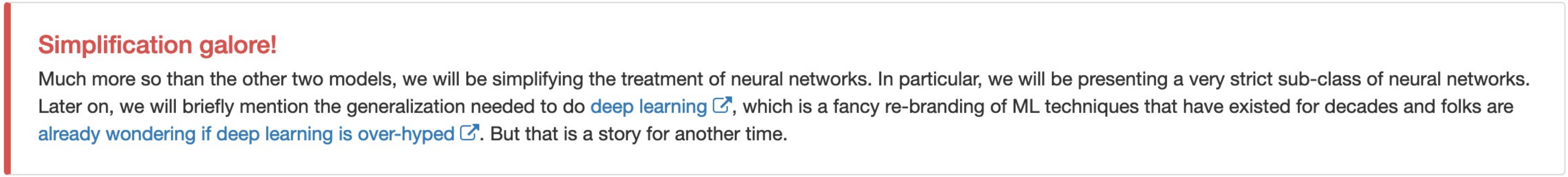
Google AI chief Dean sees evolution in ML Perf benchmark for

From left, Geoffrey Hinton, Yann LeCun, Yoshua Bengio.



Studies have shown that AI can outperform human doctors at identifying breast cancer from ... [+] UNIVERSAL IMAGES GROUP VIA GETTY IMAGES

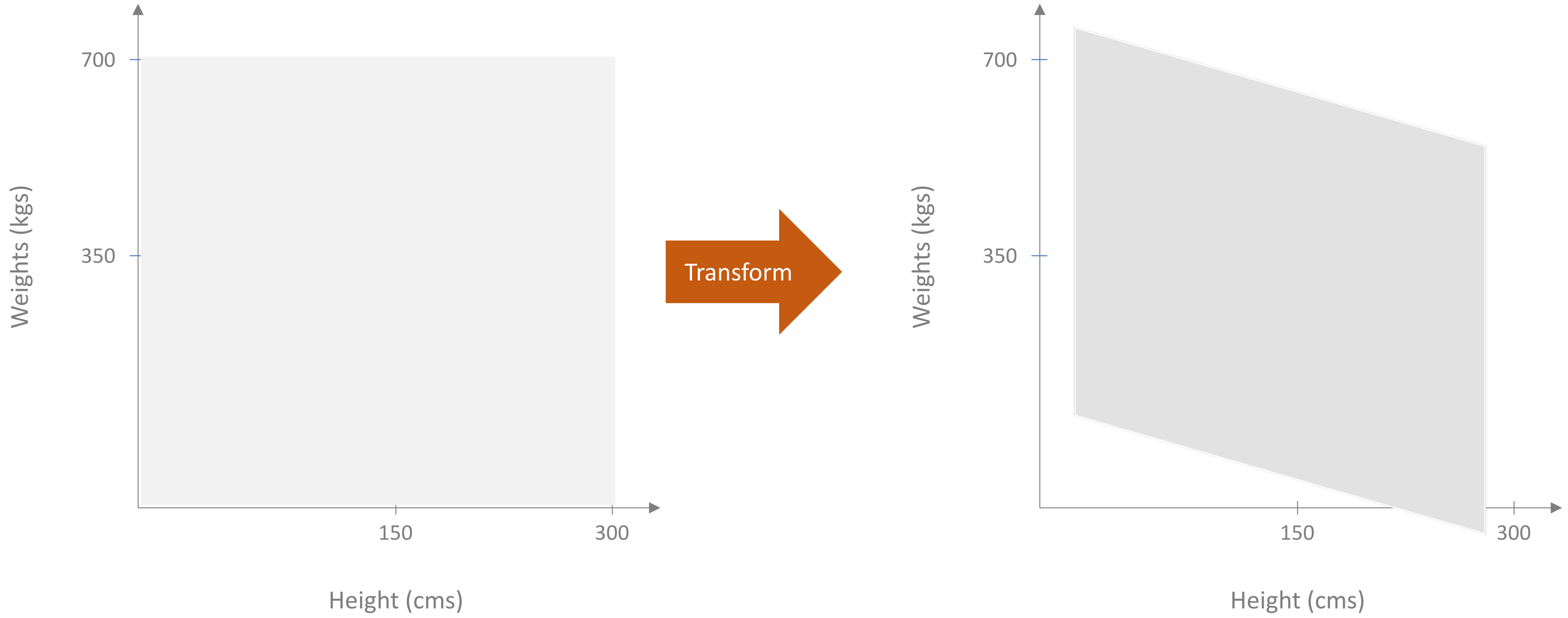
Away from the hype: non-deep NNs



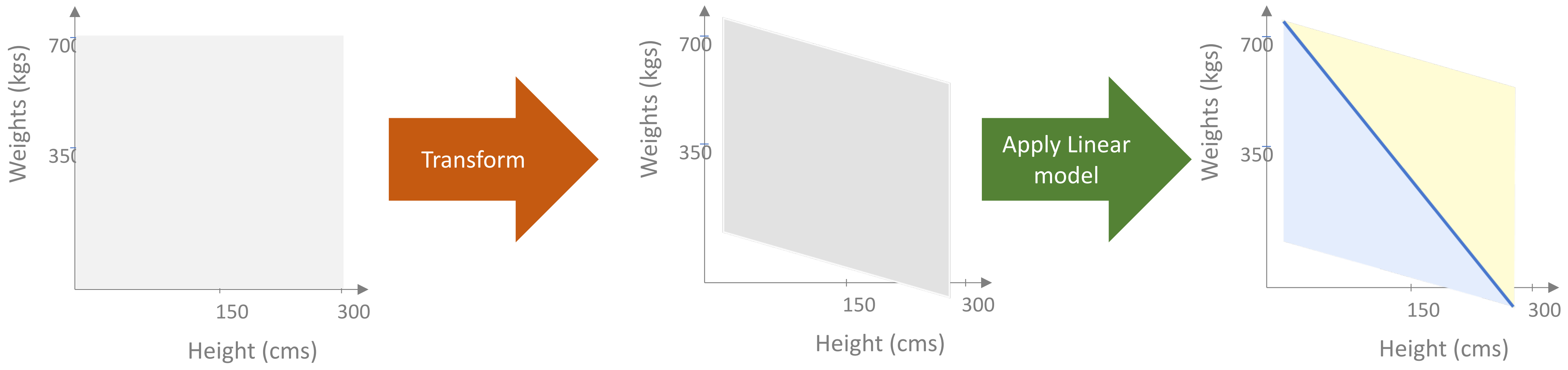
Simplification galore!

Much more so than the other two models, we will be simplifying the treatment of neural networks. In particular, we will be presenting a very strict sub-class of neural networks. Later on, we will briefly mention the generalization needed to do deep learning, which is a fancy re-branding of ML techniques that have existed for decades and folks are already wondering if deep learning is over-hyped. But that is a story for another time.

NN Idea 1: Transform the underlying space

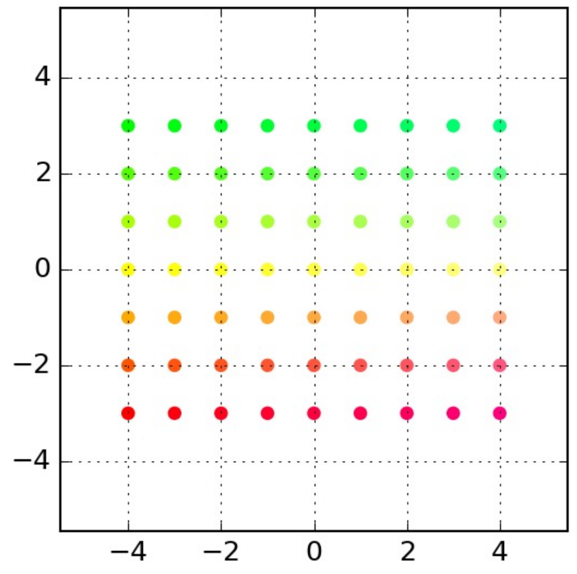


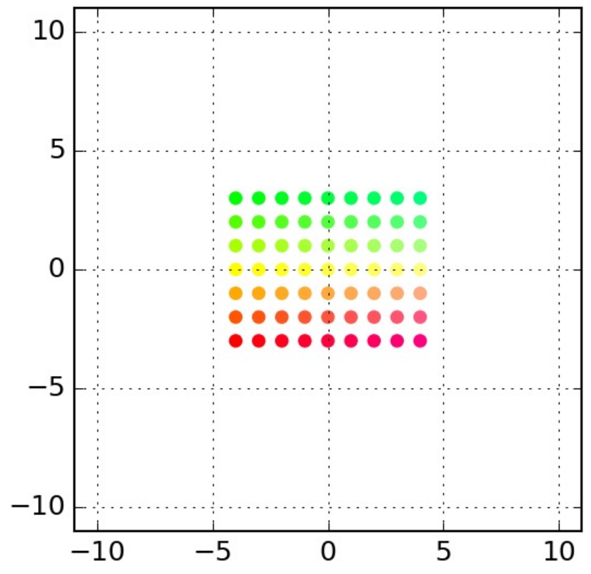
NN Idea 2: Apply linear model AFTER transform

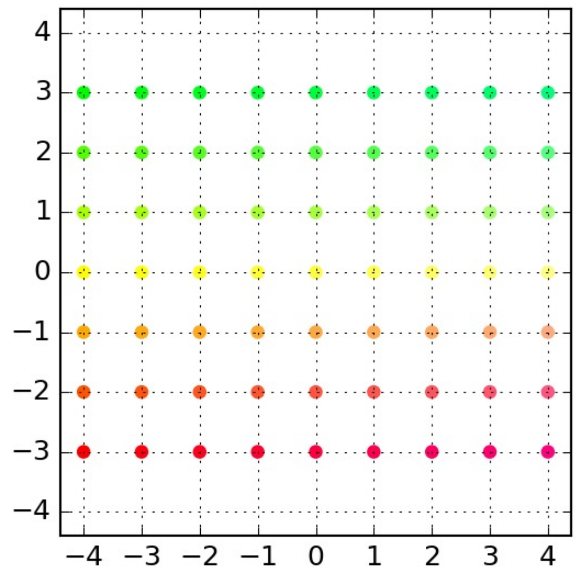


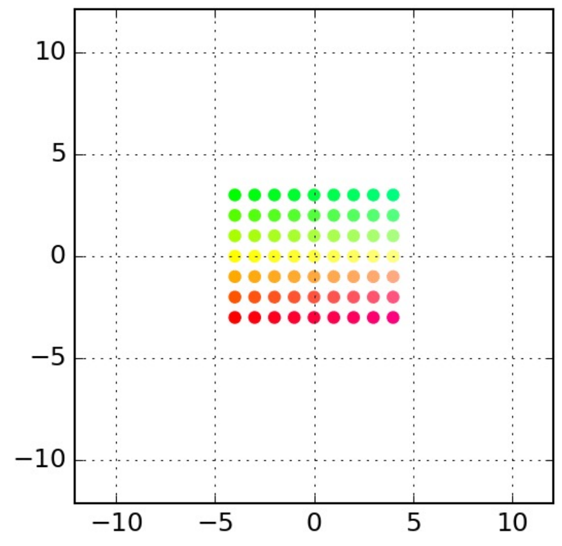
Whither the transforms?

One possibility: Linear transforms


Rotation

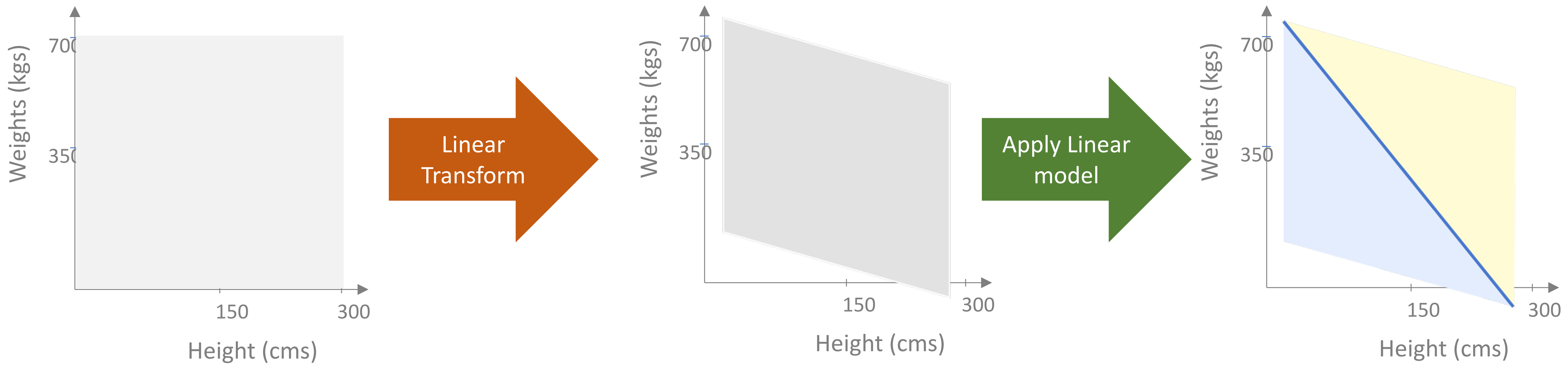

Shear


Permutation

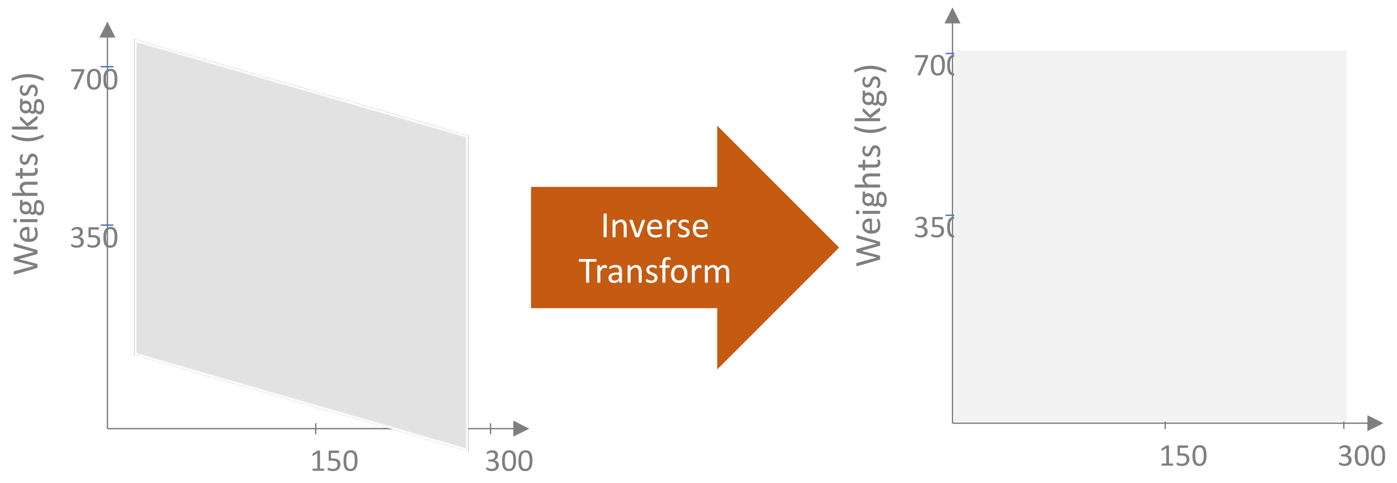

General Linear

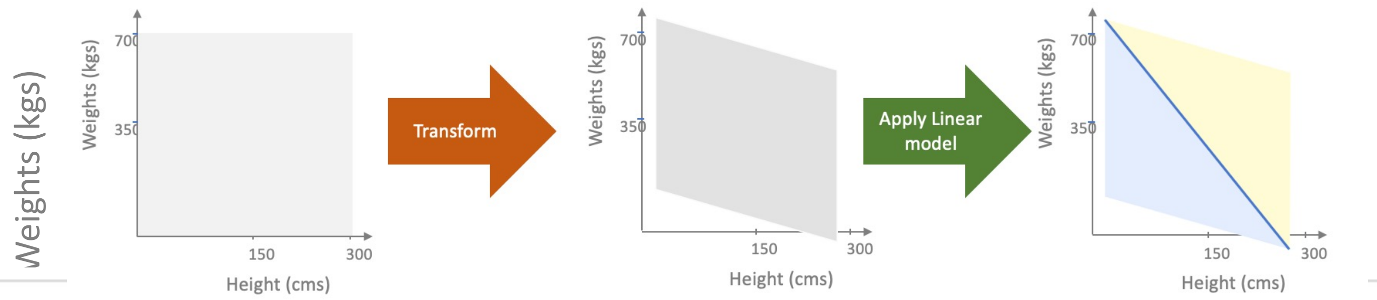
https://dododas.github.io/linear-algebra-with-python/posts/16-12-29-2d-transformations.html

Current overall scheme



Linear transforms* are invertible





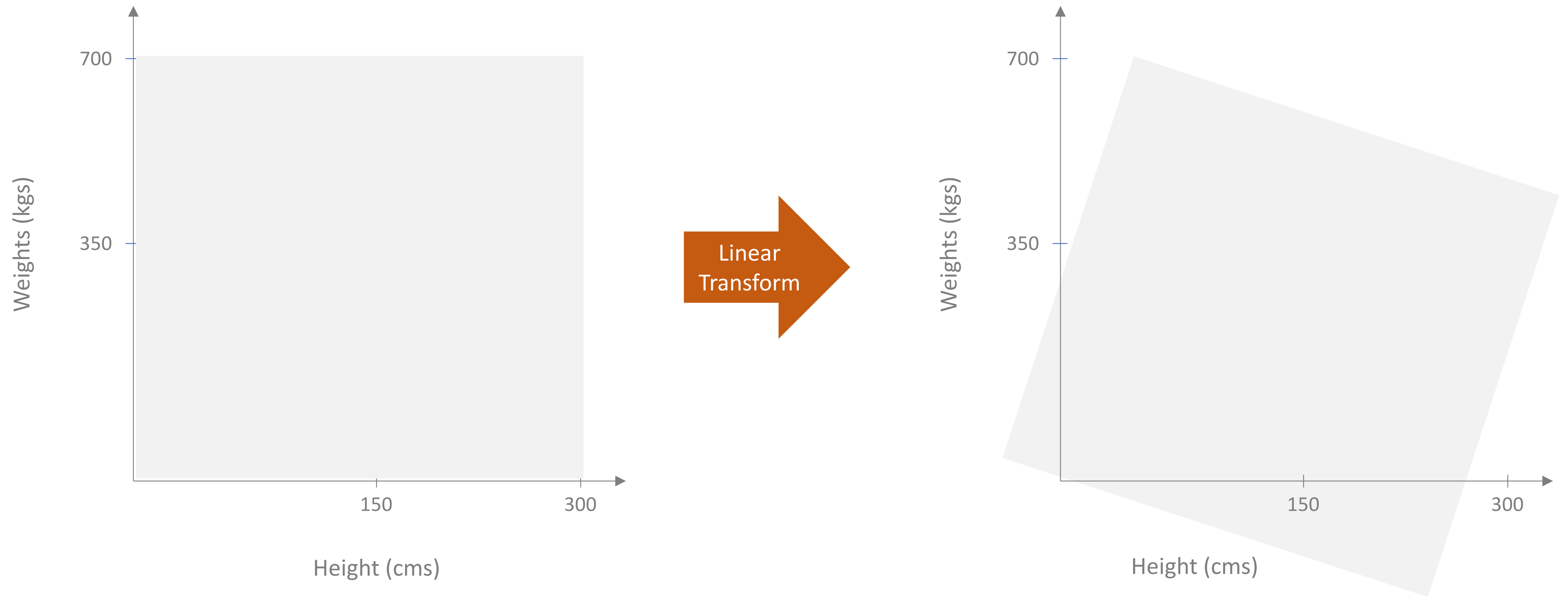
Exercise

Argue that a linear transform followed by a linear model is **equivalent** to (another) linear model in the original space.

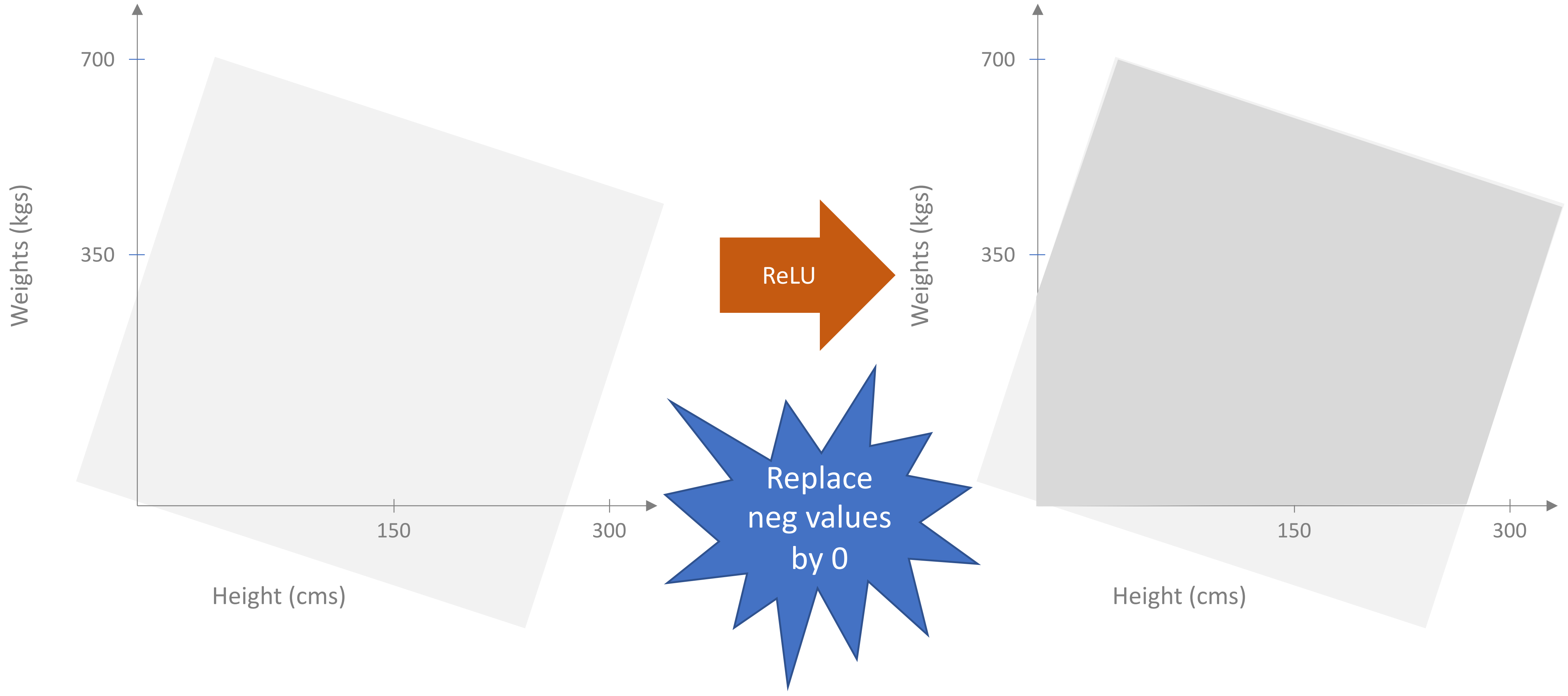
`Hint`: Can you "reverse" the whole process?

Height (cms)

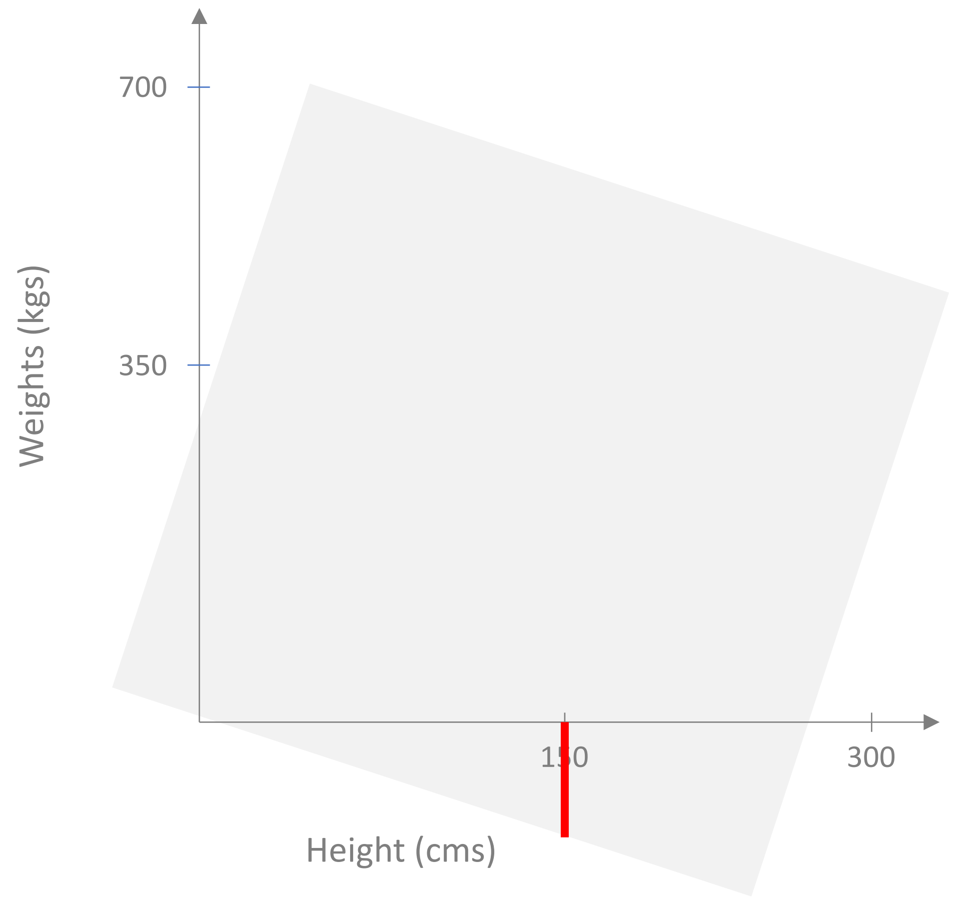
NN Idea 1’: Use a non-linear transform



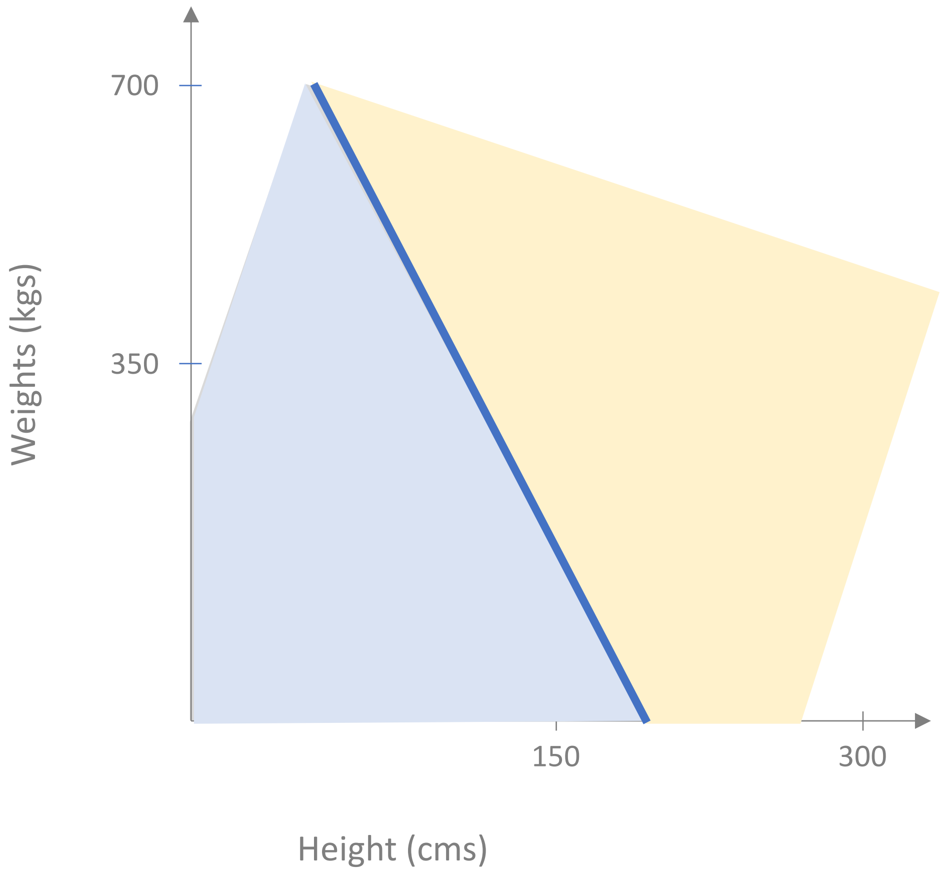
NN Idea 1’: Use a non-linear transform



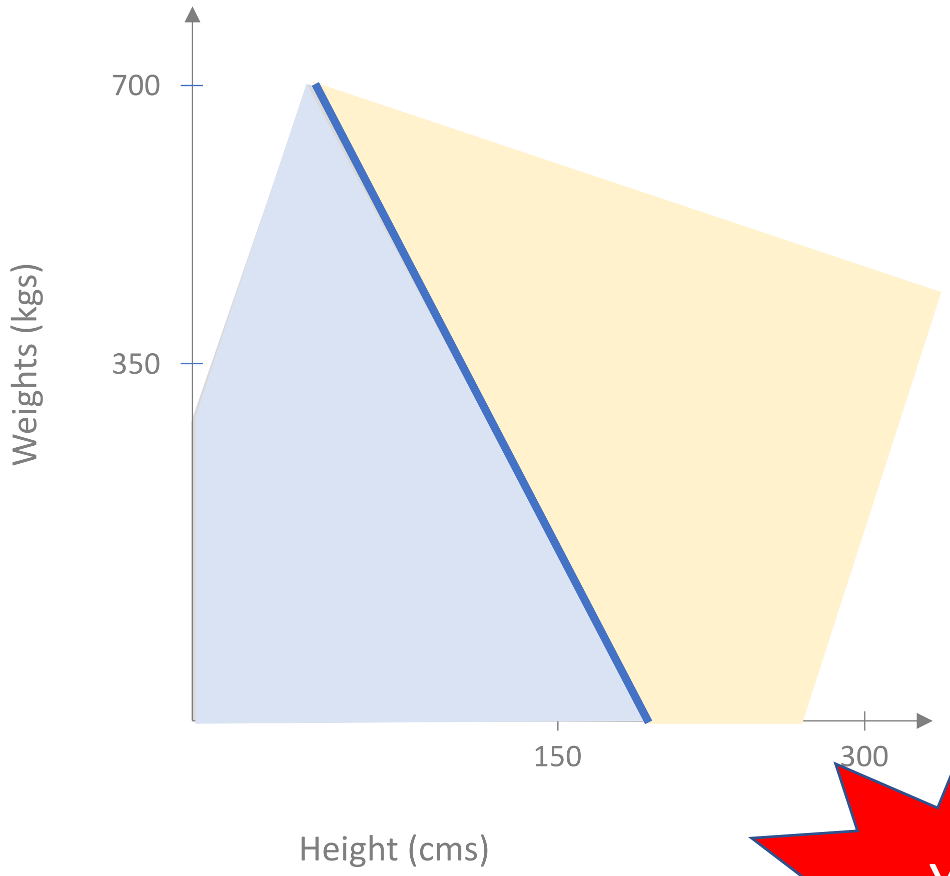
Bunch of values get zeroed out!

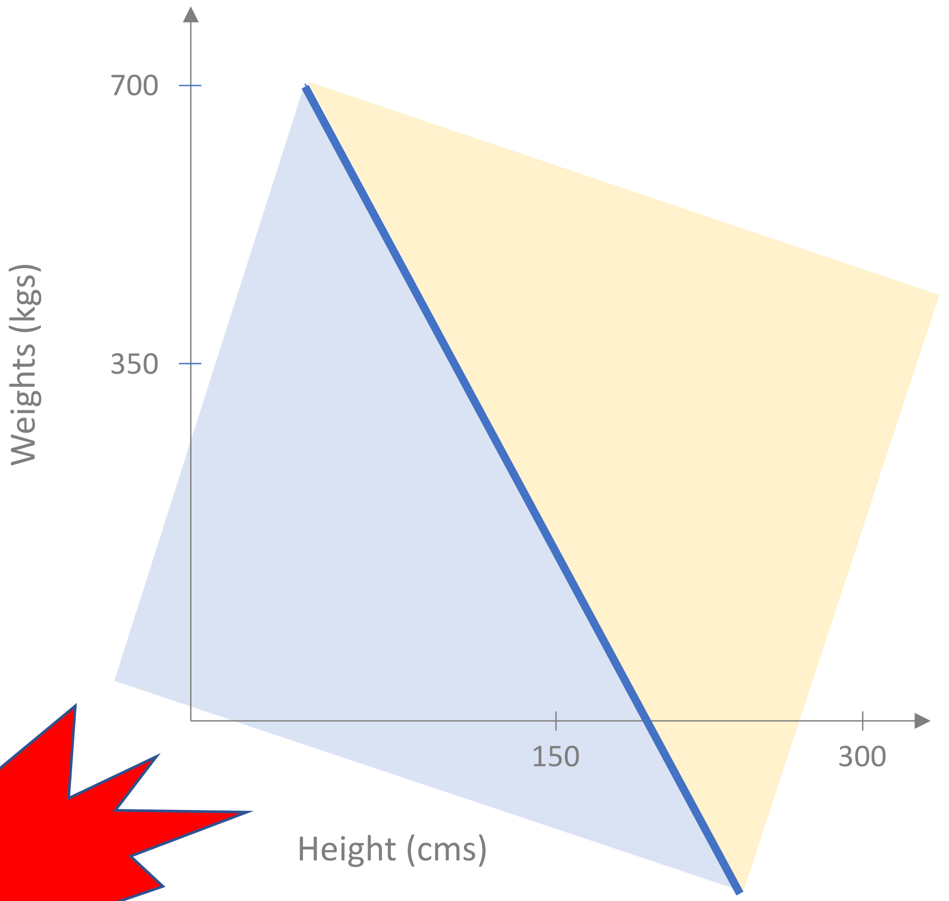


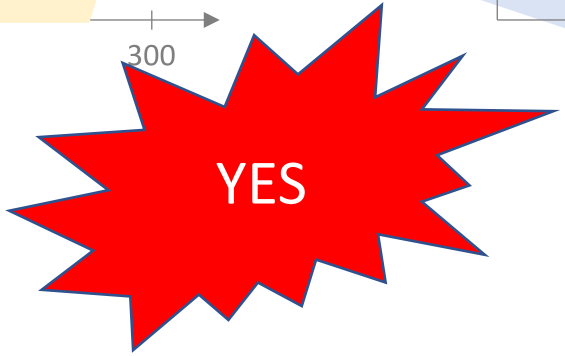
Let’s apply linear model to transformed space



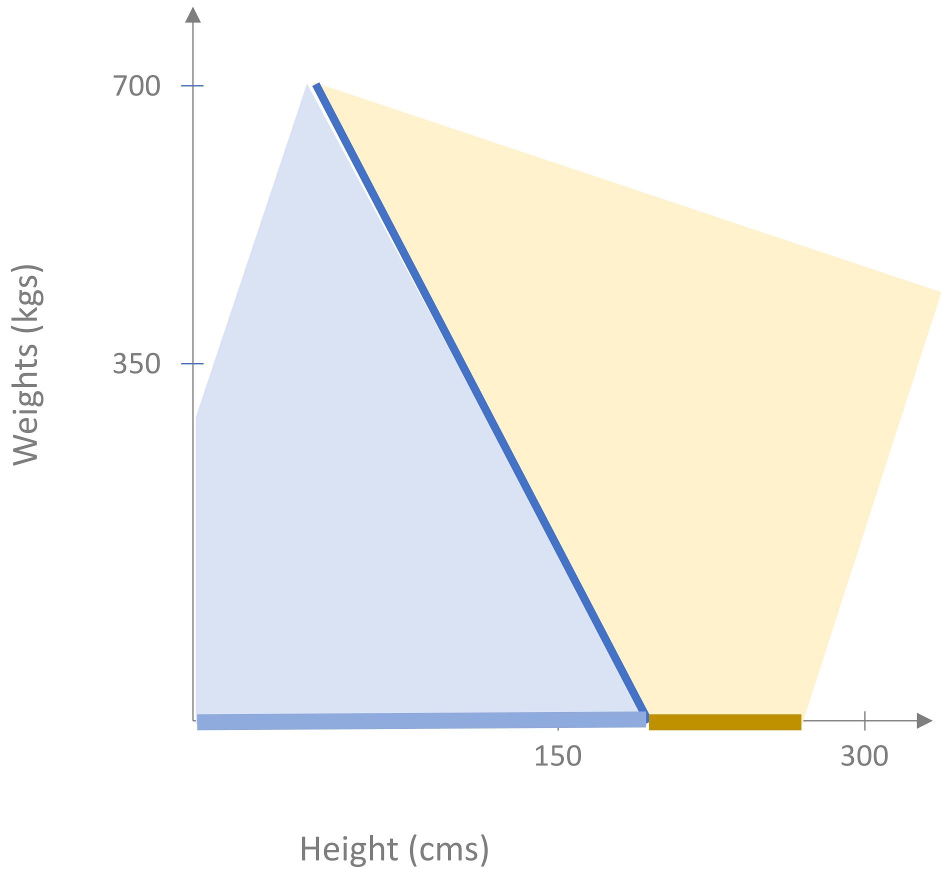
Did we gain over linear models?



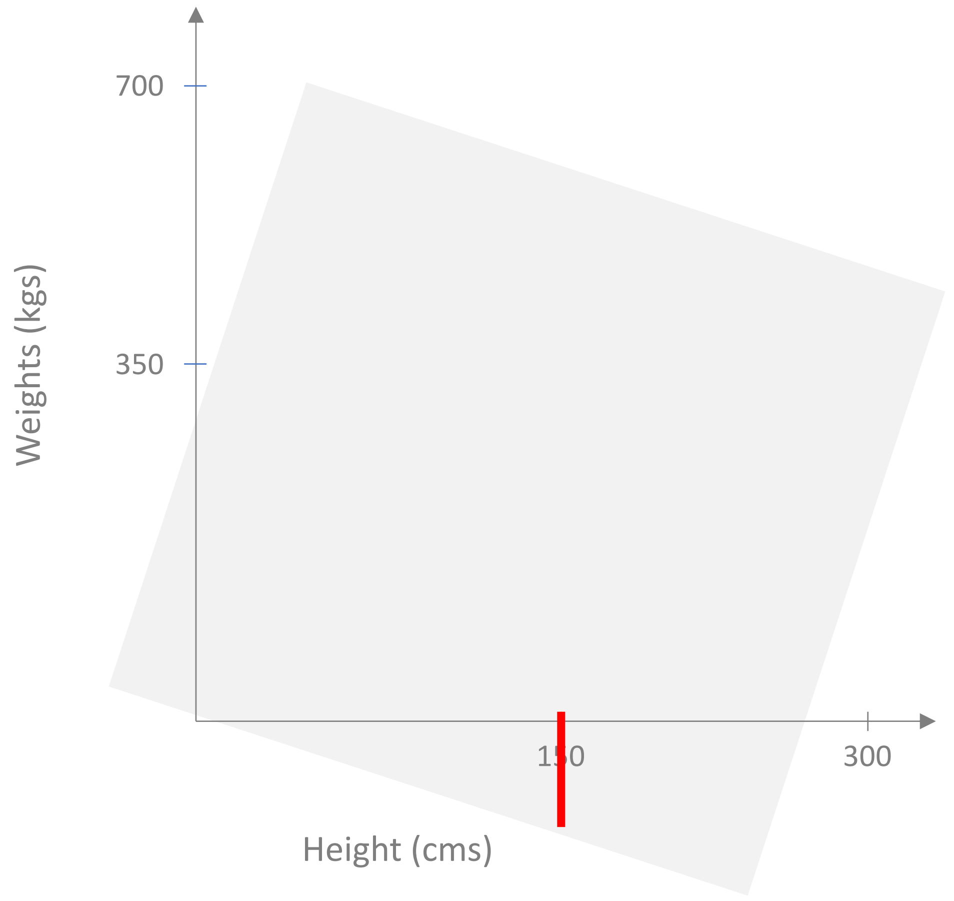




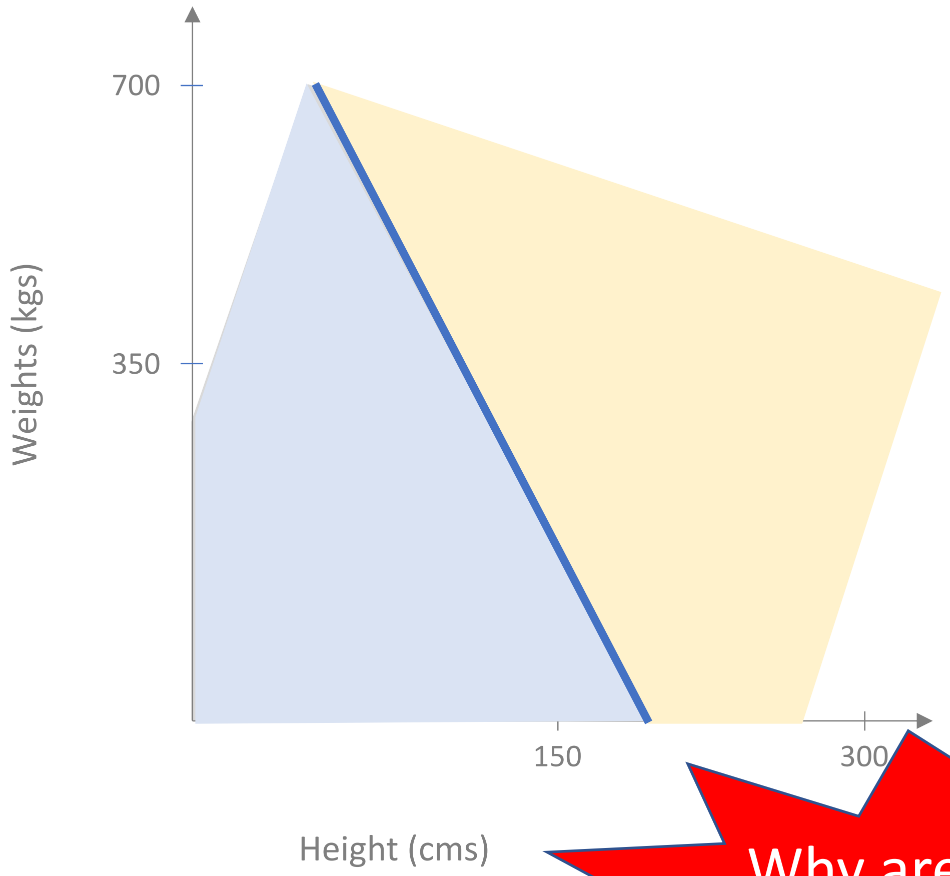
Reminder # 1

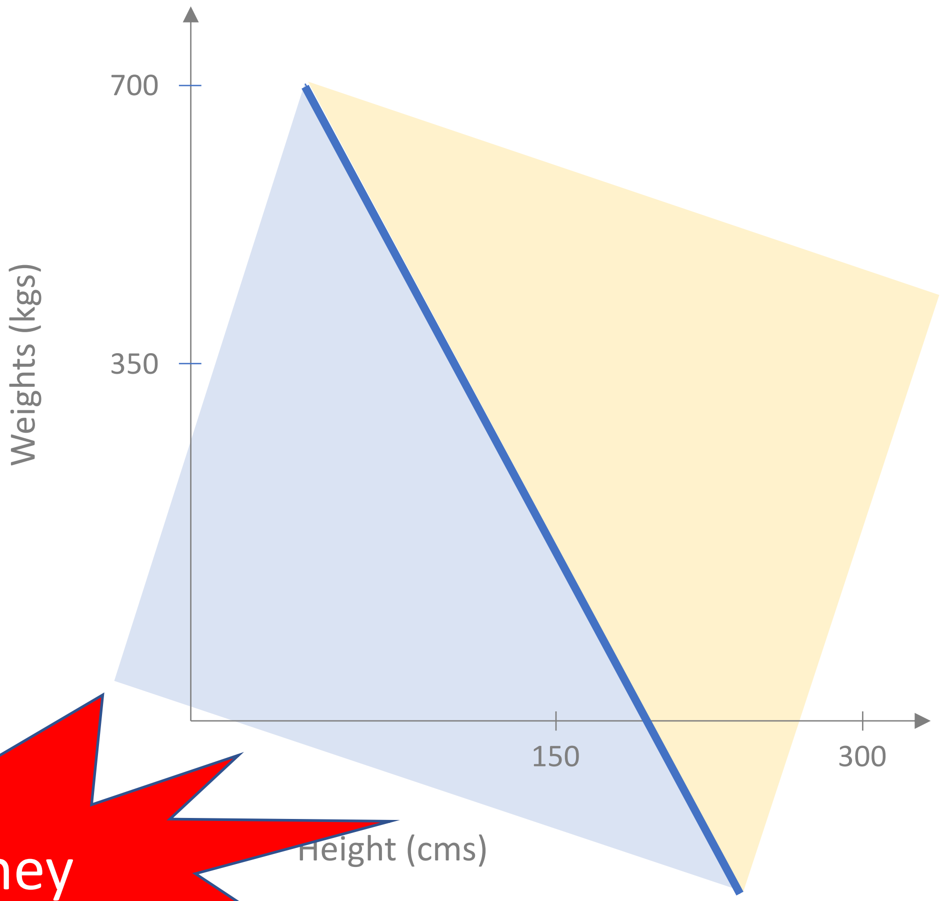


Reminder # 2



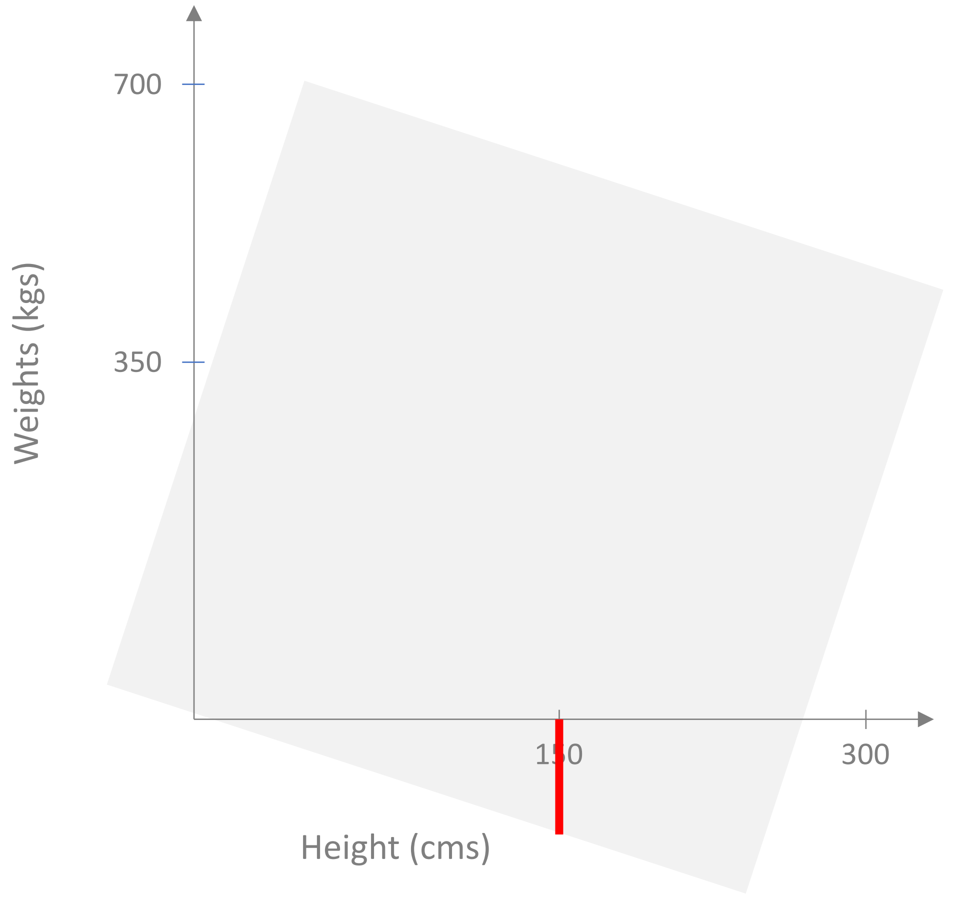
Did we gain over linear models?

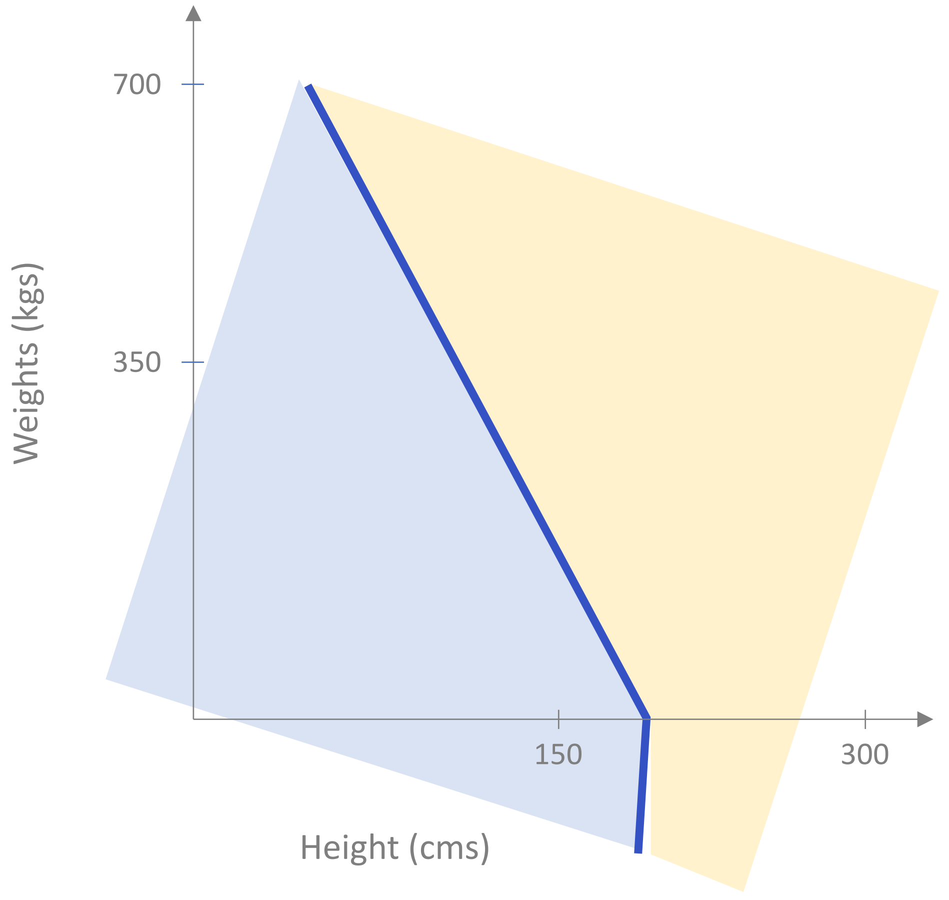




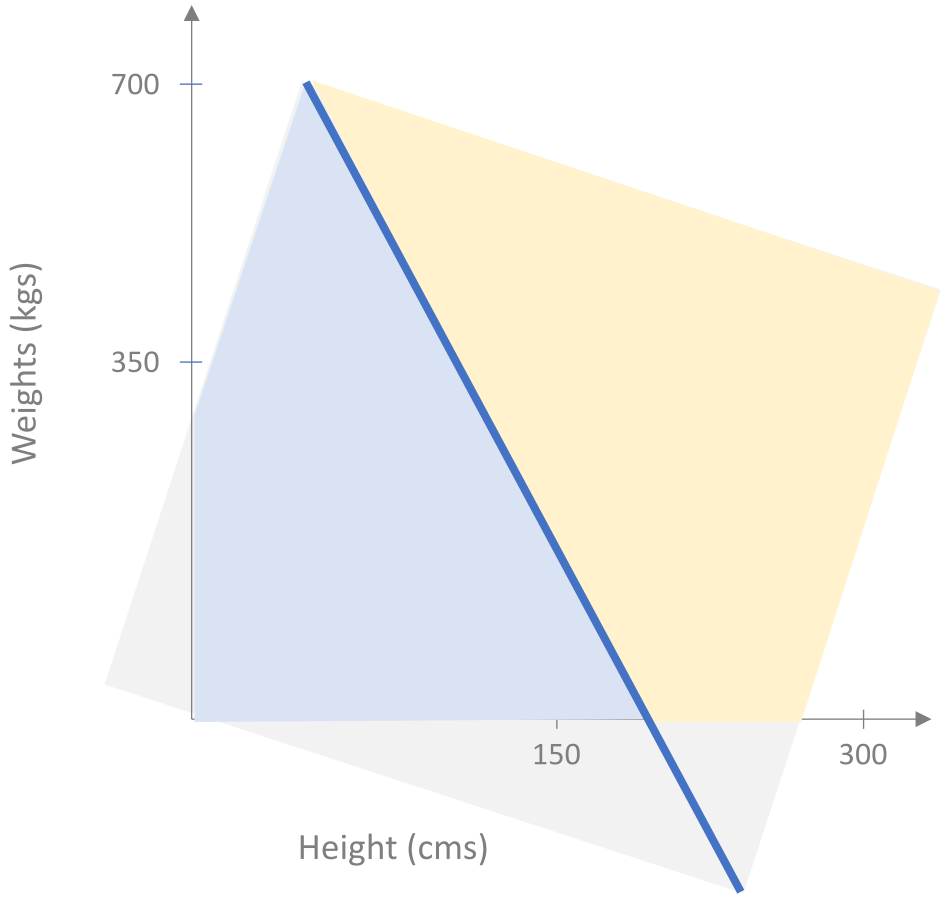
Why are they different?

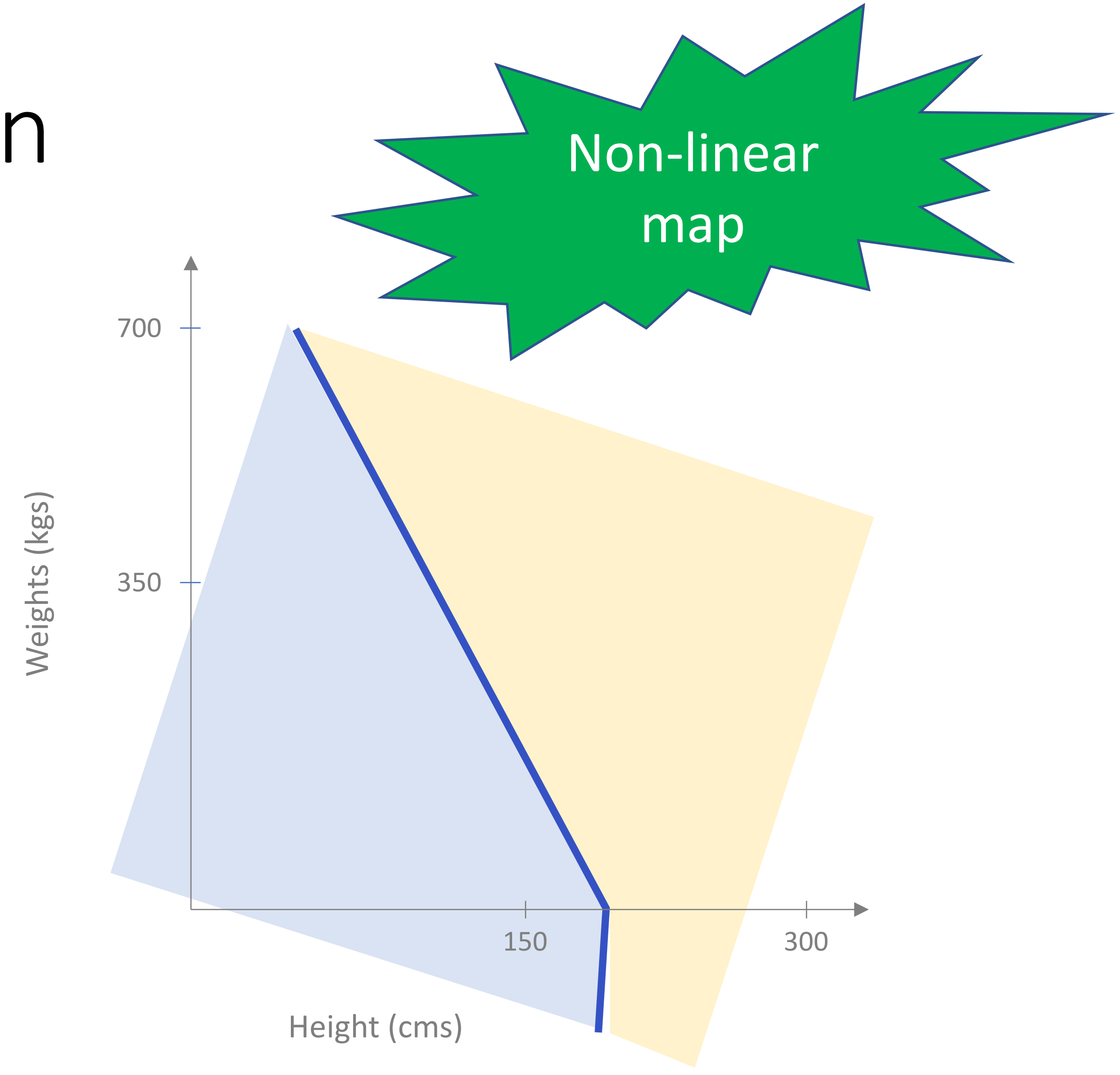
What is really happening....



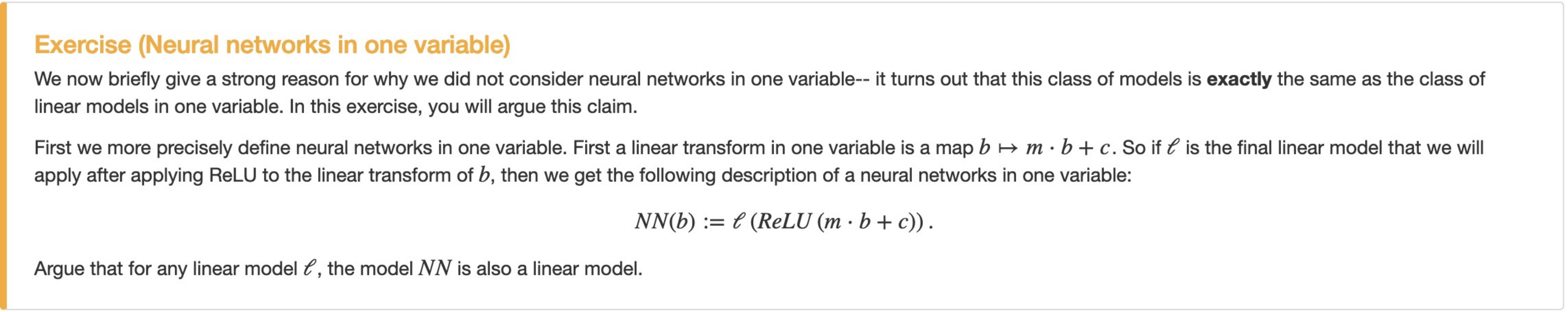


Side by side comparison





An exercise to do at home (if you so choose)



Exercise (Neural networks in one variable)

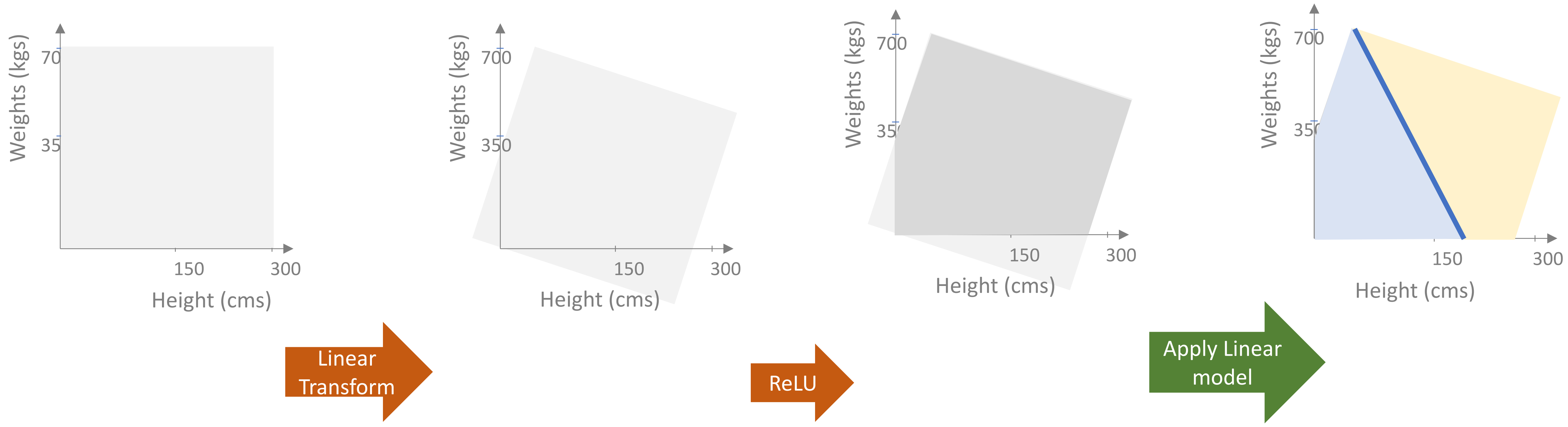
We now briefly give a strong reason for why we did not consider neural networks in one variable-- it turns out that this class of models is **exactly** the same as the class of linear models in one variable. In this exercise, you will argue this claim.

First we more precisely define neural networks in one variable. First a linear transform in one variable is a map $b \mapsto m \cdot b + c$. So if ℓ is the final linear model that we will apply after applying ReLU to the linear transform of b, then we get the following description of a neural networks in one variable:

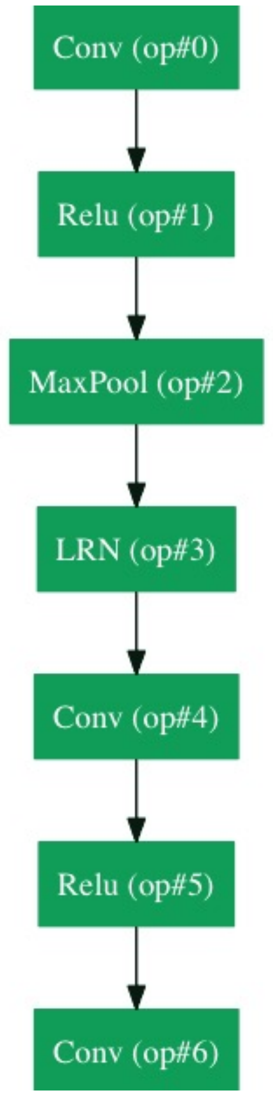
$$NN(b) := \ell\left(ReLU\left(m \cdot b + c\right)\right).$$

Argue that for any linear model ℓ, the model NN is also a linear model.

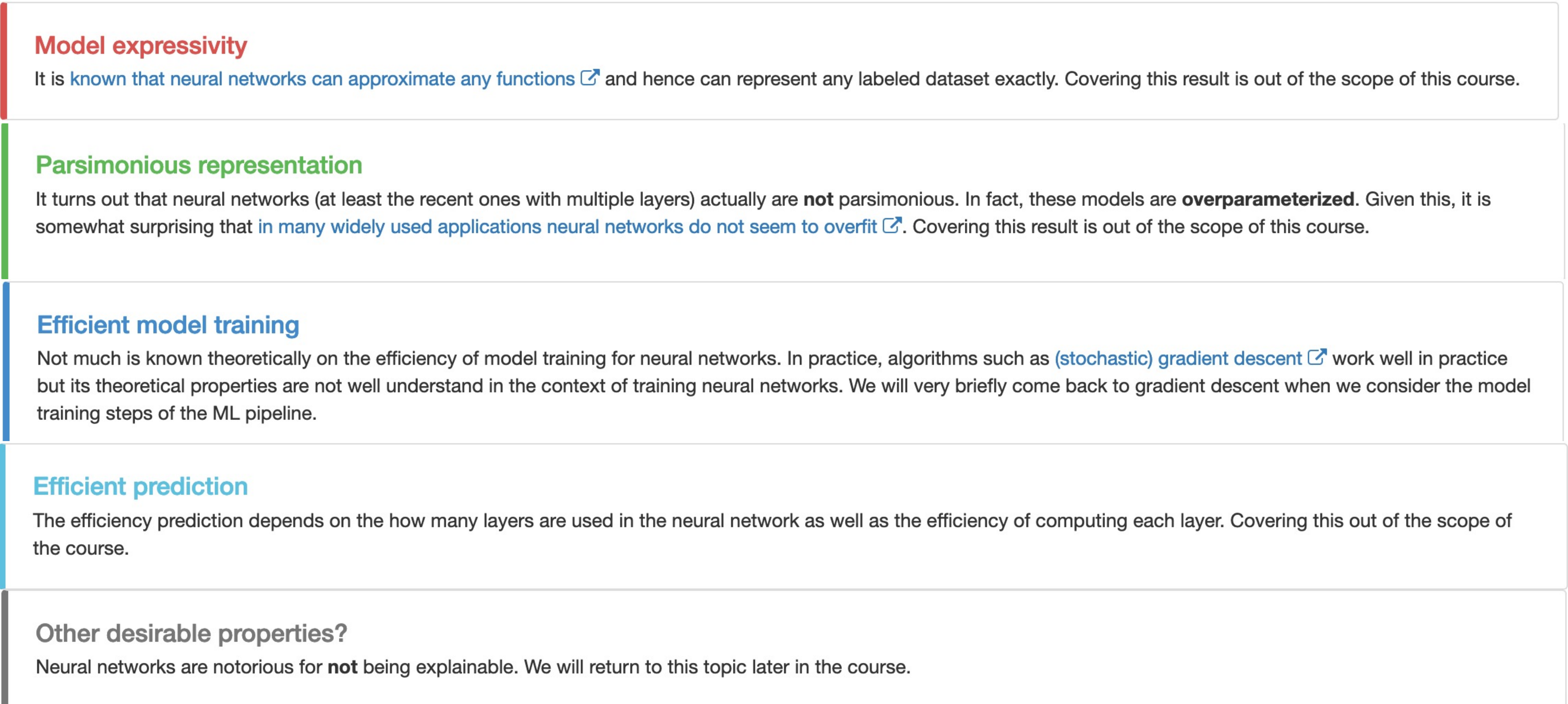
What is so “deep” about this?



Use many non-linear transforms!



Desired properties of NNs



Model expressivity

It is known that neural networks can approximate any functions and hence can represent any labeled dataset exactly. Covering this result is out of the scope of this course.

Parsimonious representation

It turns out that neural networks (at least the recent ones with multiple layers) actually are **not** parsimonious. In fact, these models are **overparameterized**. Given this, it is somewhat surprising that in many widely used applications neural networks do not seem to overfit. Covering this result is out of the scope of this course.

Efficient model training

Not much is known theoretically on the efficiency of model training for neural networks. In practice, algorithms such as (stochastic) gradient descent work well in practice but its theoretical properties are not well understand in the context of training neural networks. We will very briefly come back to gradient descent when we consider the model training steps of the ML pipeline.

Efficient prediction

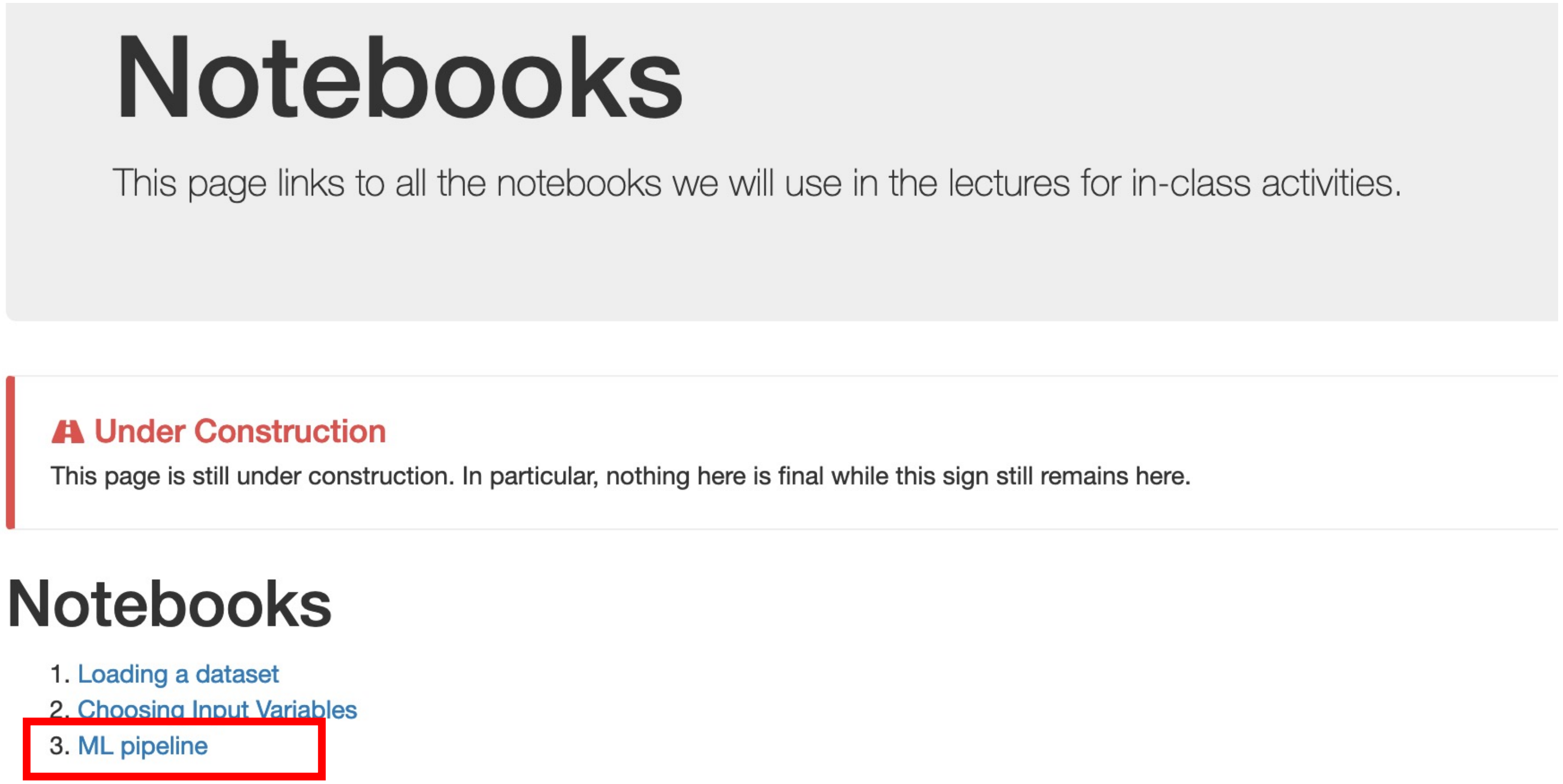
The efficiency prediction depends on the how many layers are used in the neural network as well as the efficiency of computing each layer. Covering this out of the scope of the course.

Other desirable properties?

Neural networks are notorious for **not** being explainable. We will return to this topic later in the course.

Another Jupyter exercise

https://colab.research.google.com/



Notebooks

This page links to all the notebooks we will use in the lectures for in-class activities.

Under Construction

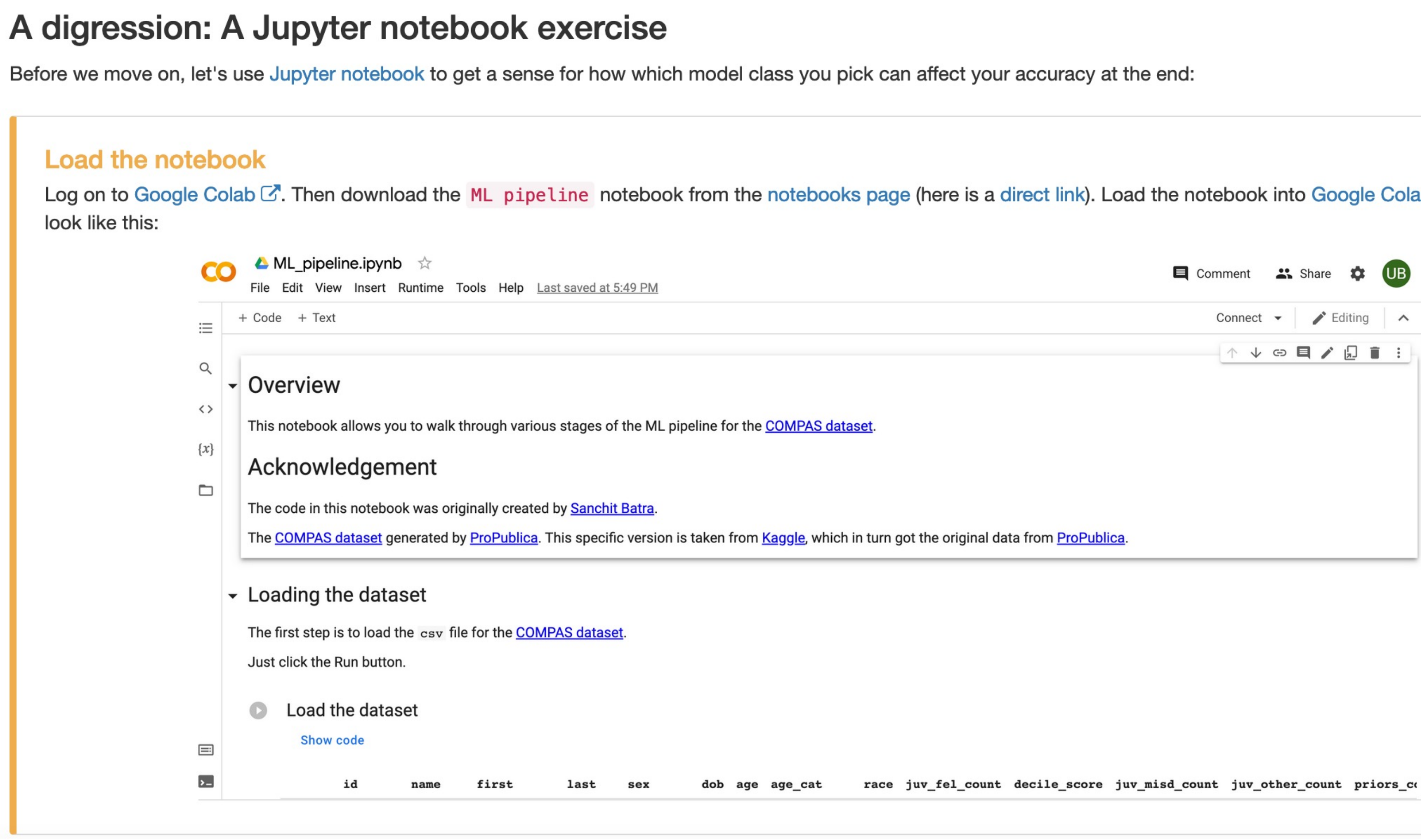
This page is still under construction. In particular, nothing here is final while this sign still remains here.

Notebooks

1. Loading a dataset
2. Choosing Input Variables
3. ML pipeline

Another Jupyter exercise

https://colab.research.google.com/



A digression: A Jupyter notebook exercise

Before we move on, let's use Jupyter notebook to get a sense for how which model class you pick can affect your accuracy at the end:

Load the notebook

Log on to Google Colab. Then download the `ML pipeline` notebook from the notebooks page (here is a direct link). Load the notebook into Google Cola
look like this:

ML_pipeline.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5:49 PM
Comment Share UB
+ Code + Text
Connect Editing

Overview

This notebook allows you to walk through various stages of the ML pipeline for the COMPAS dataset.

Acknowledgement

The code in this notebook was originally created by Sanchit Batra.

The COMPAS dataset generated by ProPublica. This specific version is taken from Kaggle, which in turn got the original data from ProPublica.

Loading the dataset

The first step is to load the `csv` file for the COMPAS dataset.

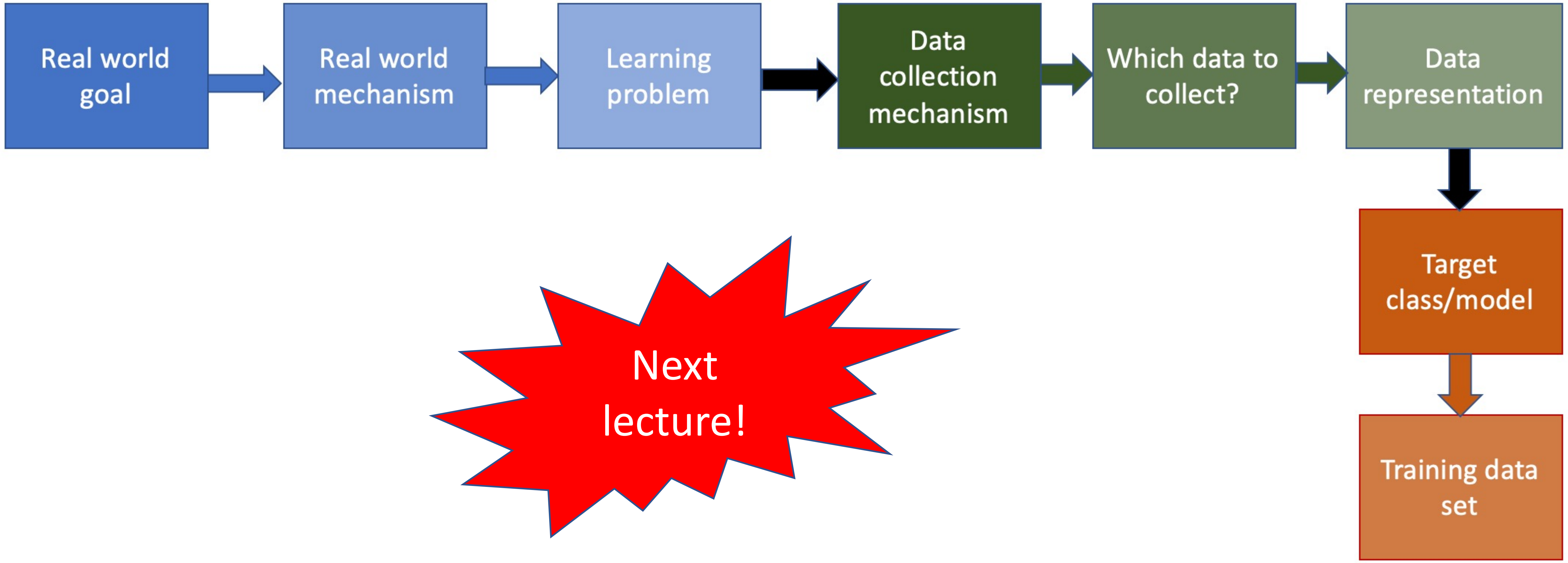
Just click the Run button.

Load the dataset

Show code

```
   id    name    first    last    sex    dob  age  age_cat    race  juv_fel_count  decile_score  juv_misd_count  juv_other_count  priors_c
```

Finally to next step in the ML pipeline!


Real world goal
Real world mechanism
Learning problem
Data collection mechanism
Which data to collect?
Data representation
Target class/model
Training data set
Next lecture!