

# ML and Society

Feb 17, 2022

# Please have a face mask on

## Masking requirement



*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>

# All Jupyter notebooks have been added

CSE 440/441/540 Resources ▾

## 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](#)

## Datasets

Below are some of the datasets that we will be used by the notebooks we use in class:

- [COMPAS dataset](#). This is a dataset generated by [ProPublica](#). This specific version is taken from [Kaggle](#), which in turn got the original data from [ProPublica](#).
- [Adult dataset](#). This is a dataset from [UCI ML repository: Adult dataset](#). The local file has the headers for each column as well.

# First Progress Summary due 5pm on Mon

note @16 stop following 6 views

## Autolab accepting first progress summary

Autolab is now accepting submission for the first progress summary. See the [syllabus for the details](#) on what is needed in the summary.

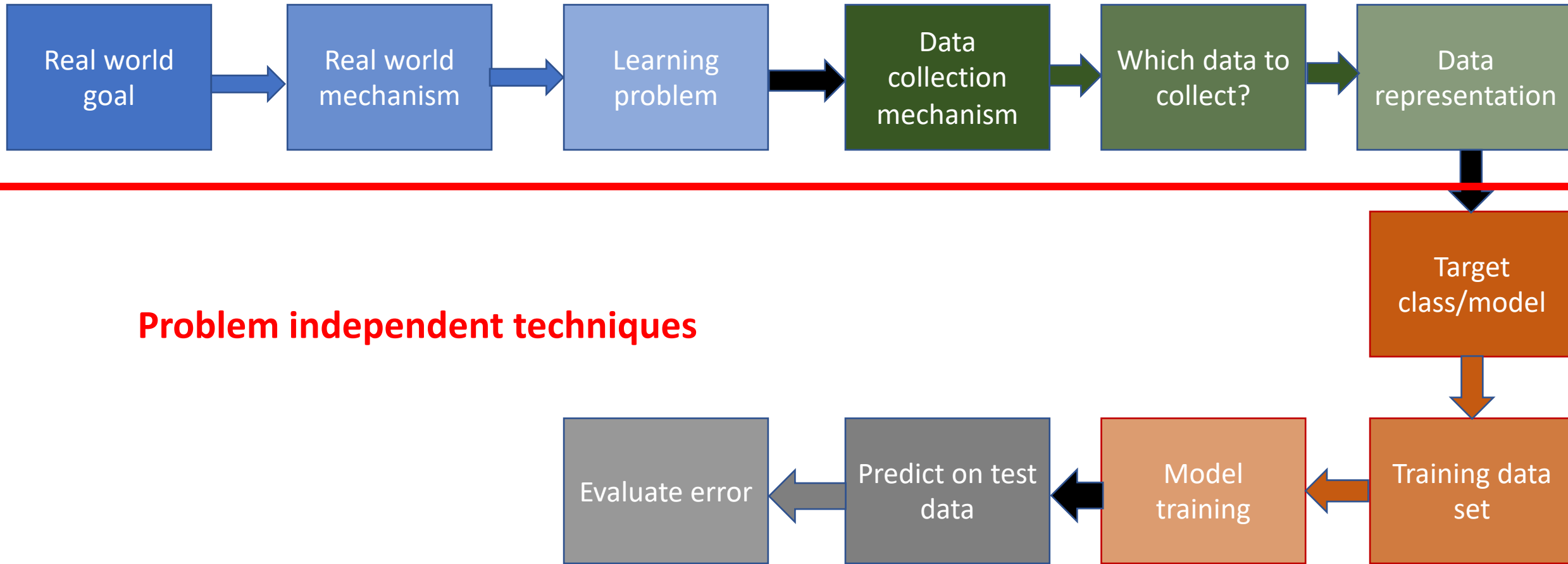
I created the groups (@15) for y'all on Autolab so only one person from the group needs to submit the summary by 5pm on Monday, Feb 21.

[project](#) [autolab](#)

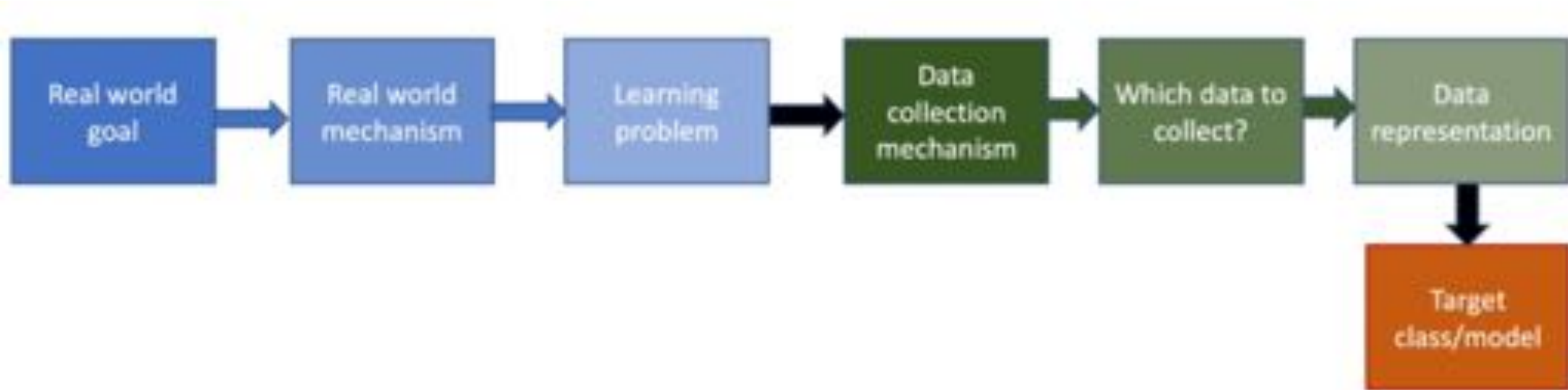
[edit](#) good note 0

Updated 1 day ago by Atri Rudra

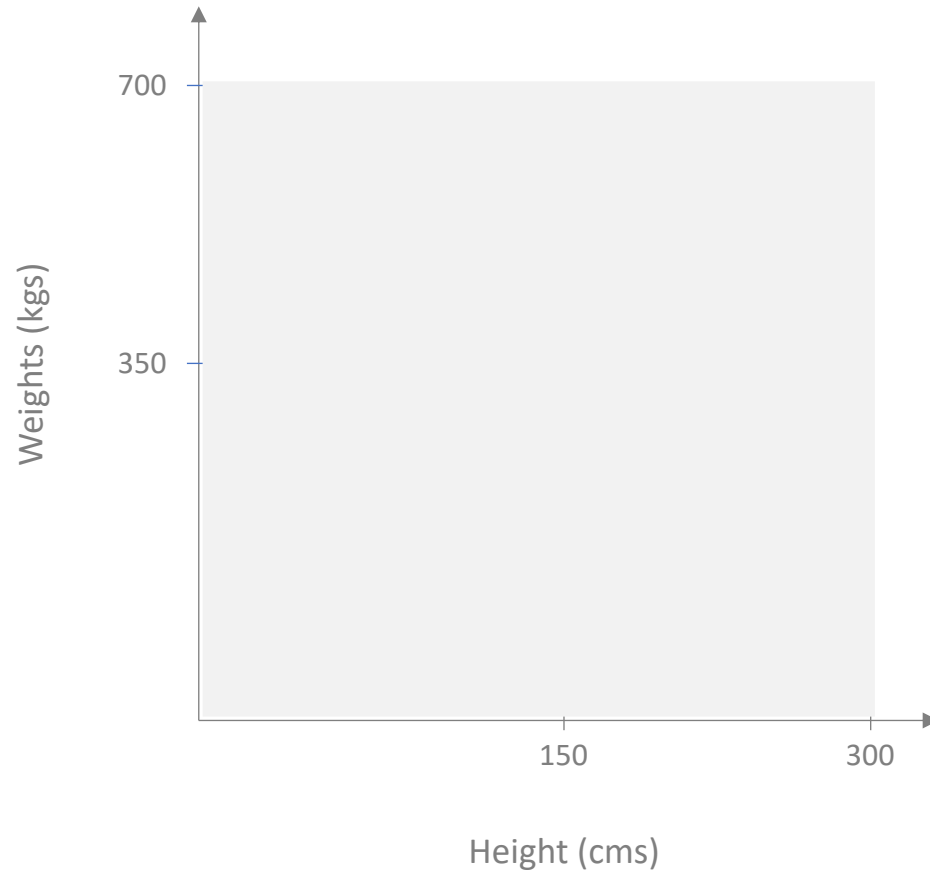
# Relation to problem statement



# ML model classes

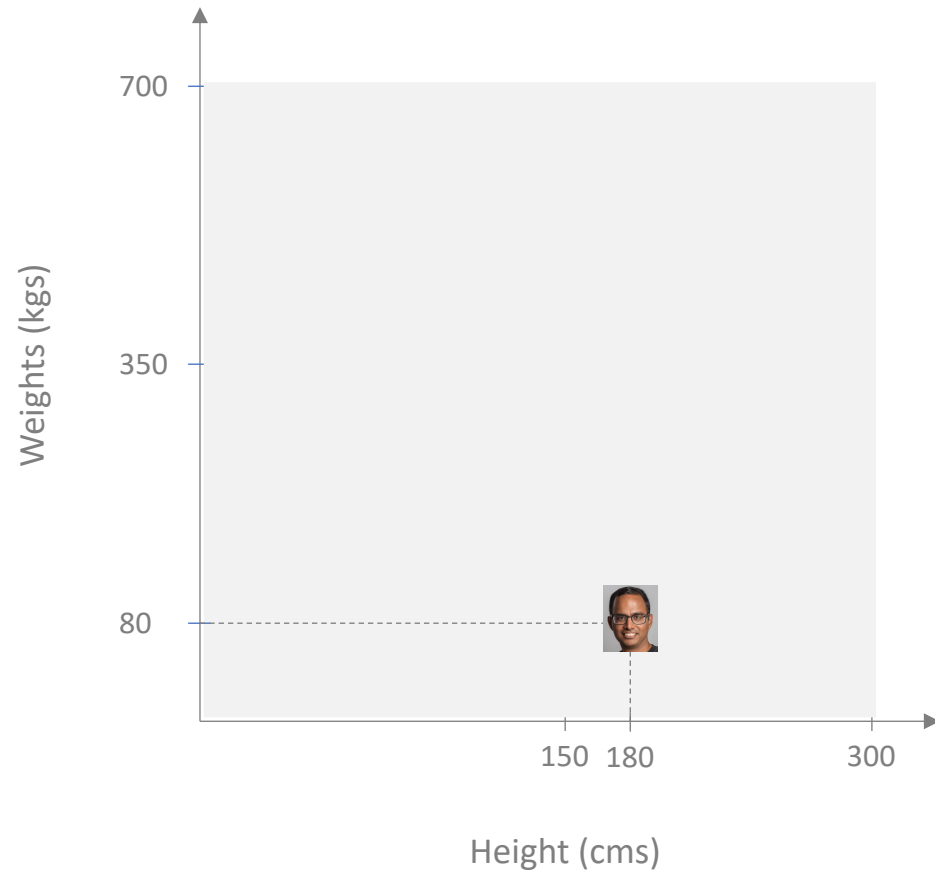


# Restrict to two input variables



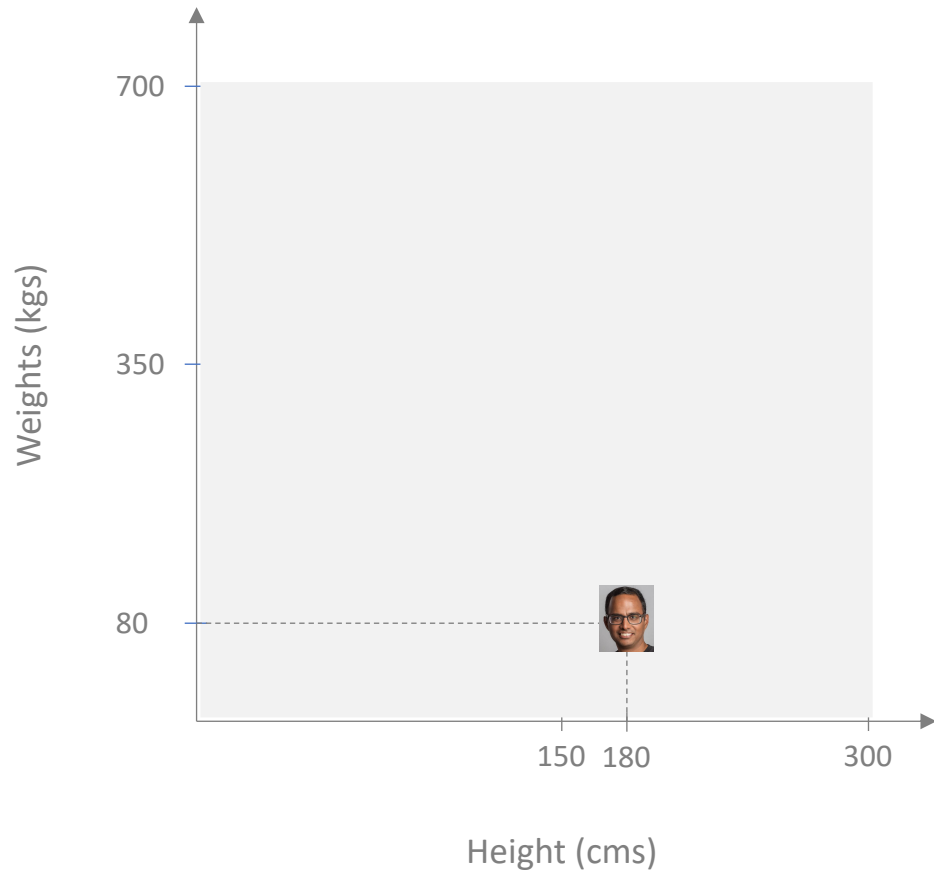
Predict risk of heart disease

# For example...

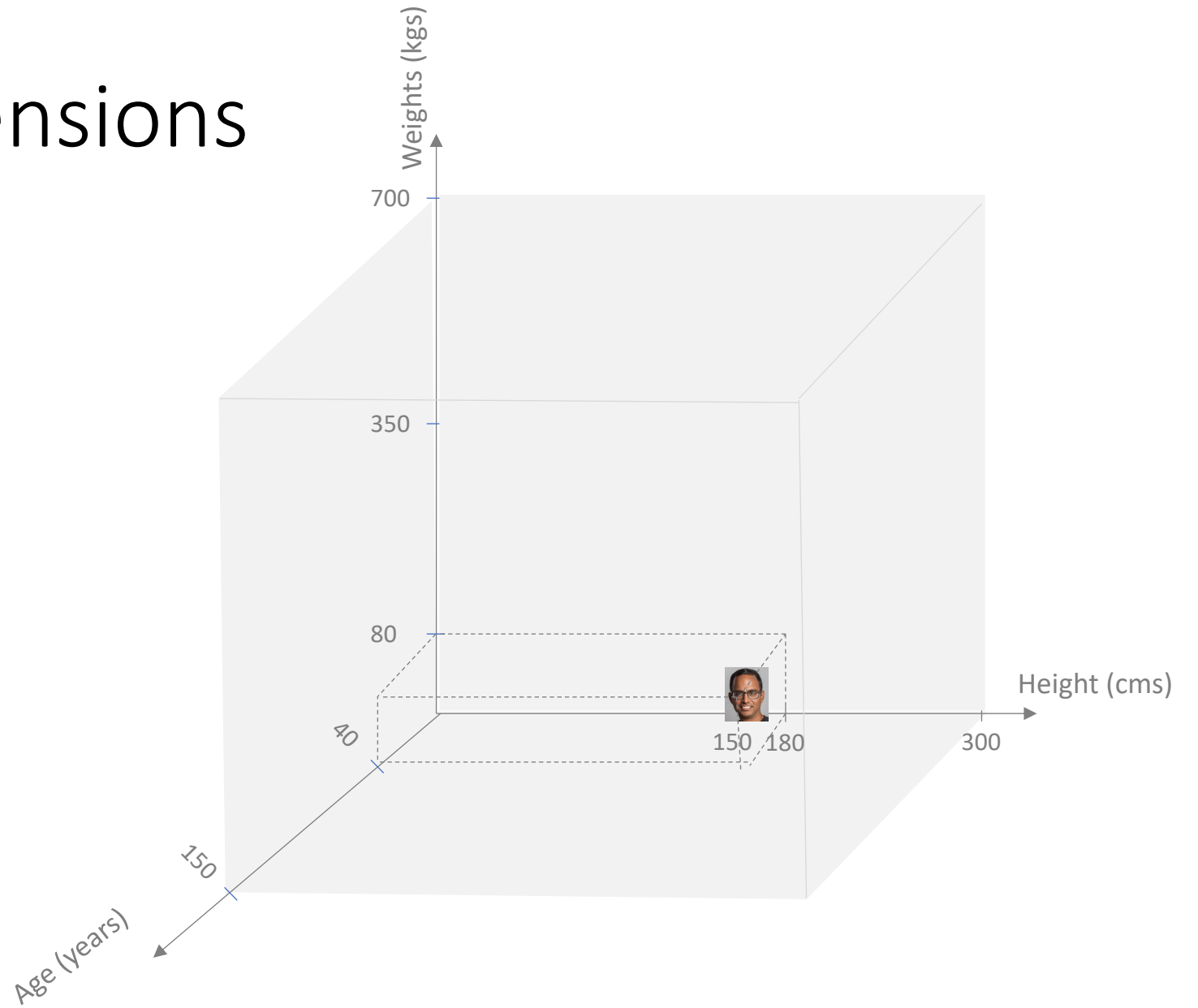




# Does every human get their unique point?



# Need more dimensions





# Focus on binary classification

## Binary Classification is the name of the game

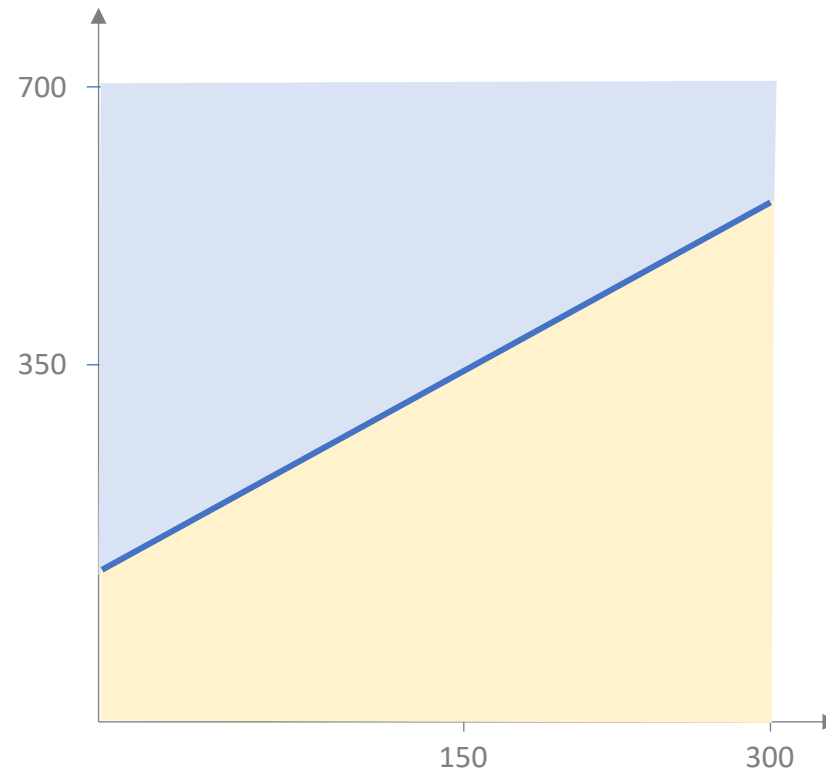
We had mentioned this in the passing *earlier*, that in this course we will focus on binary classification: i.e. the target variables that we will consider in this course only take two values: typically we will call one value *positive* and the other *negative*. For example, in the running example in this notes of predicting the risk of heart disease, *positive* would mean high risk of getting heart disease and *negative* would mean low risk of heart disease.

Of course, in many cases (including the heart disease risk prediction) it makes sense to have the target variable take many values (or even for the case when the target variable is inherently binary it makes sense to assign a probability of whether the target variable is positive-- in which case the target variable is a real number between 0 and 1 and thus takes on infinitely possible values).

We decided to focus on binary classification since it makes the exposition easier (including drawing figures of the kind y'all will see soon) but at the same time is complex enough to illustrate many of the technical details and nuances.

# What is a model?

A "curve"/function that labels every point either as positive (blue) or negative (yellow)



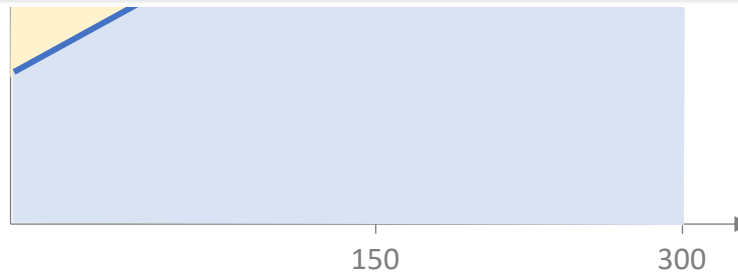
The line "defines" the model. Almost...

# The colors can be flipped!



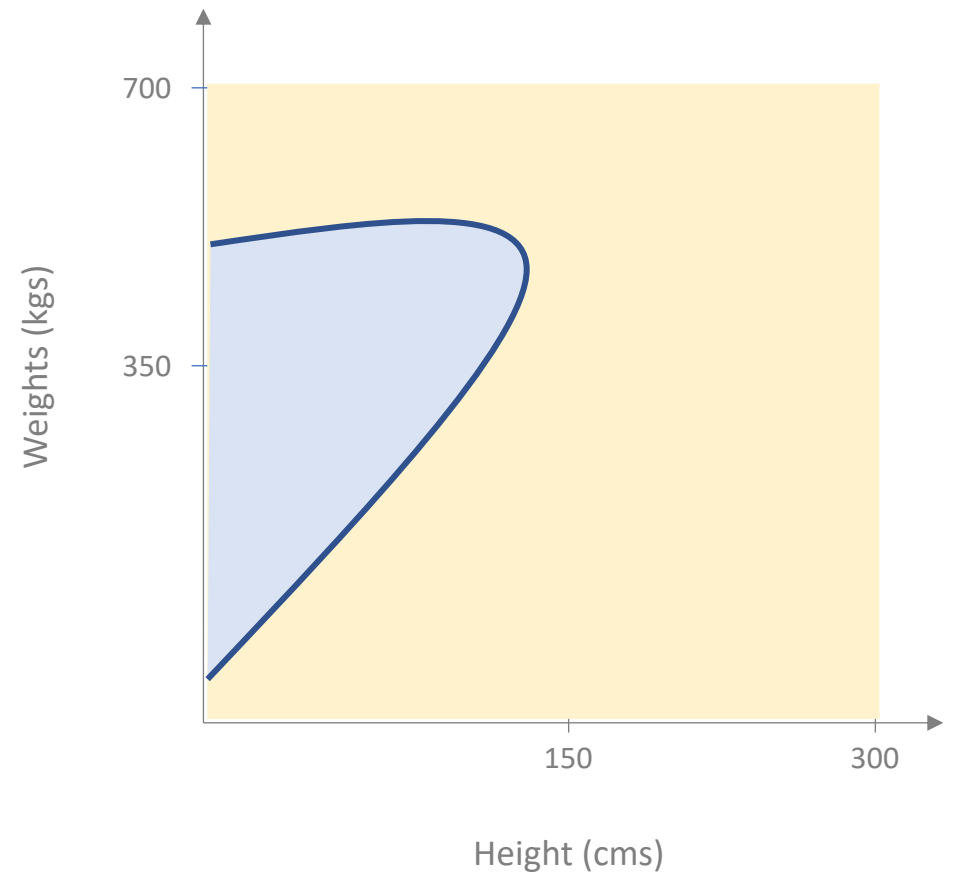
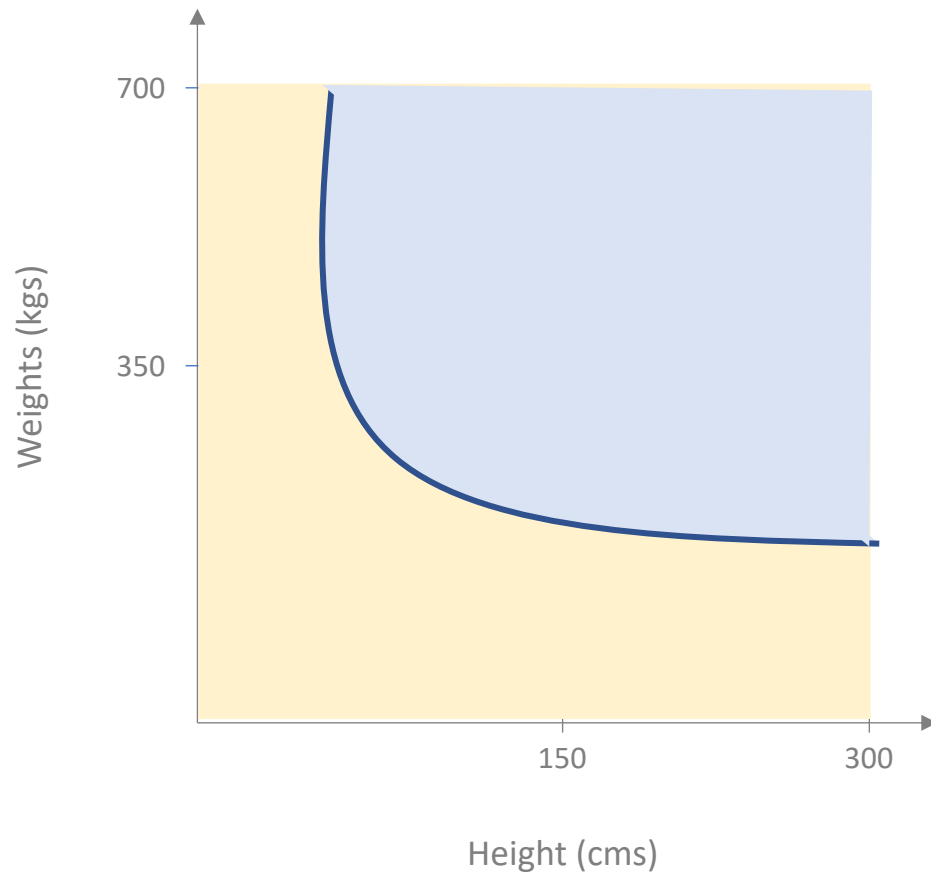
## Linear models

The models we considered above (as we have seen before) are called *linear models* (because you can literally draw a line to present them in 2D). In particular, in the case of two input variables, a linear model is **completely defined** once you specify the line as which of the two sides is the positive side (and the other side automatically becomes the negative side).

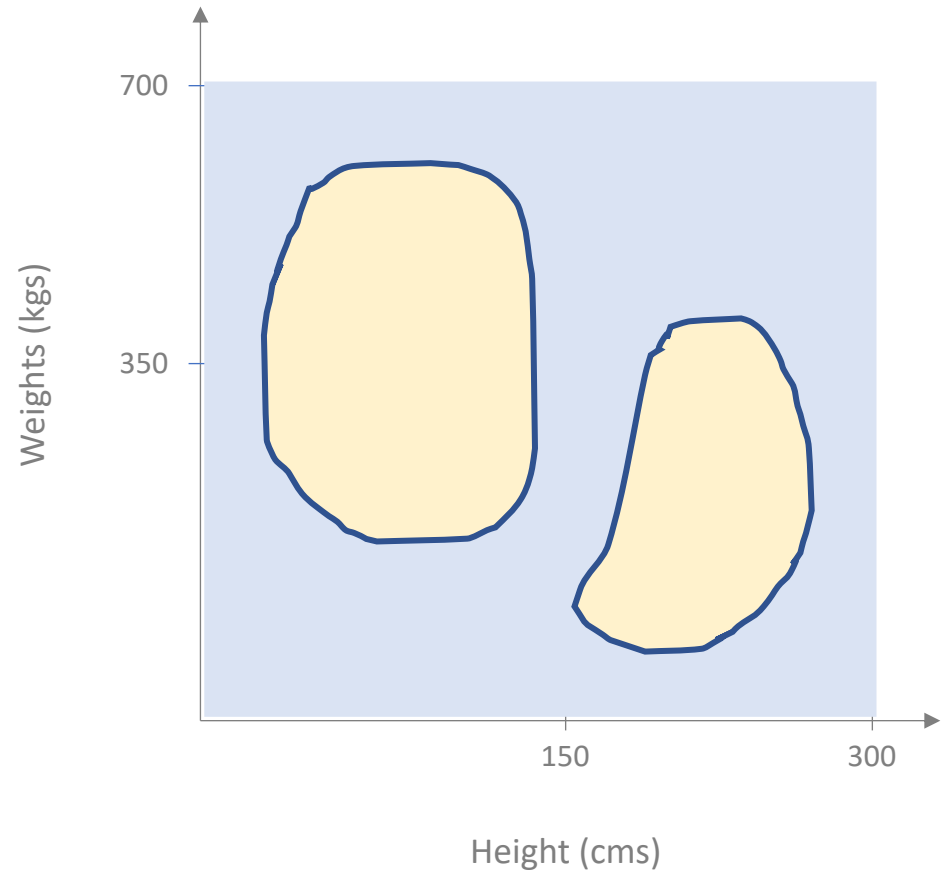
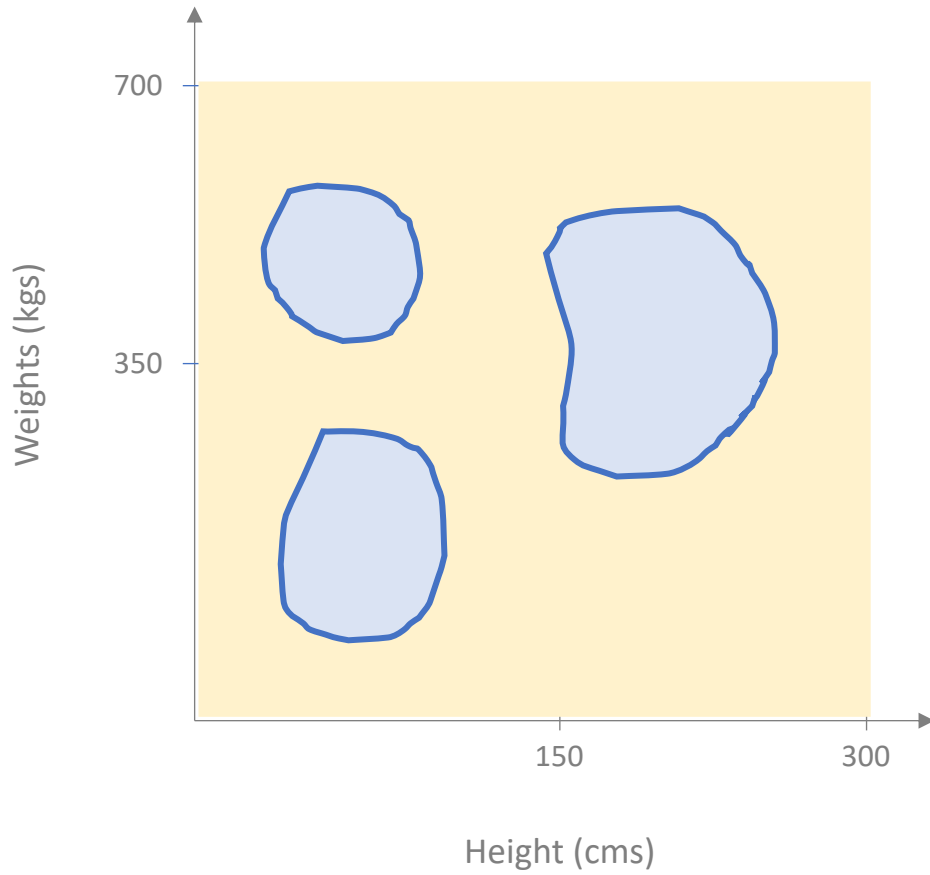


Height (cms)

# Can go beyond linear models: why not?

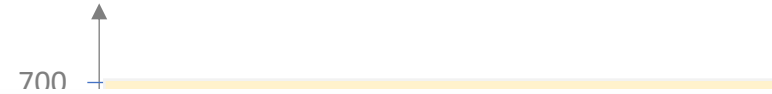


It can get crazier...





# And crazier!



## The figures above are not of real models

In case y'all were wondering the non-linear models were all drawn by hand and do not correspond to a model that has been used by someone in real life. However, except for perhaps the last figure, they are approximations of models that are used in real-life.



## Models for binary classification

A model is a function (or a procedure if you will) that divides up the ambient space (in the running example it is the gray area in the first figure) into disjoint regions where each region is colored blue (for what the model considers the positive points) and yellow (for what the model considers to be negative points).

Height (cms)

Height (cms)



# Model class is independent from earlier steps

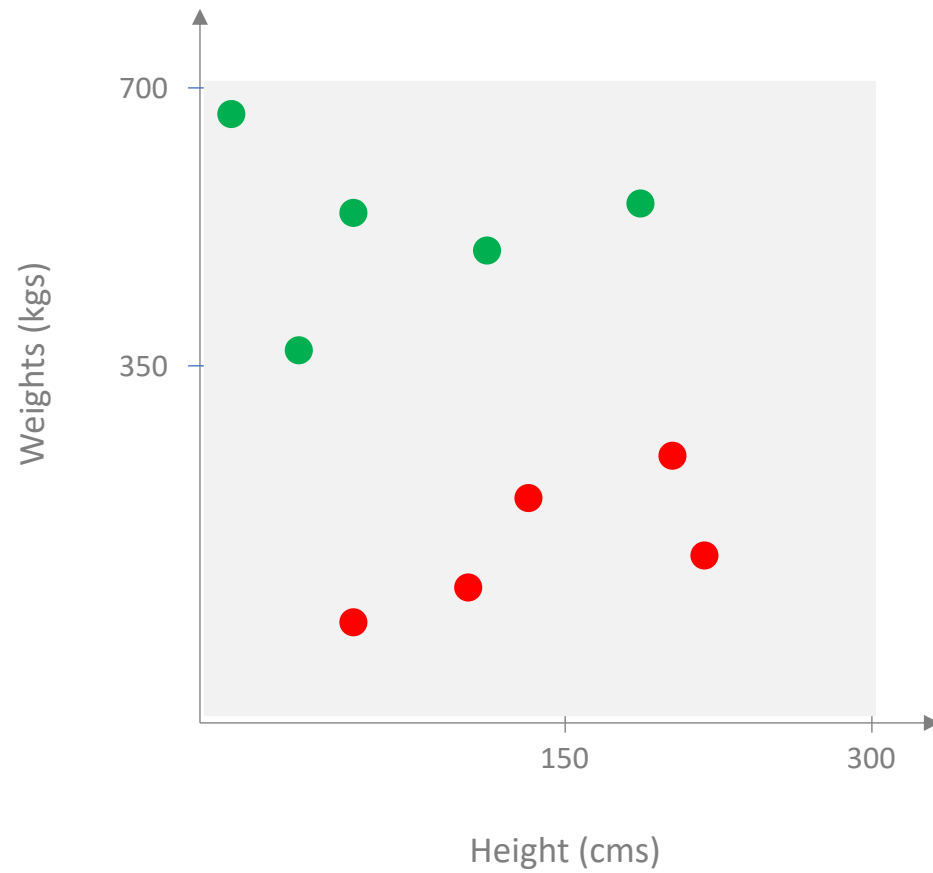
## Independence of model from problem/data

When we first looked at choosing the model class as part of our [walkthrough of the ML pipeline](#), by the time we looked at the linear model cartoon, we already had the labeled (red/green) points. However, so far we have not seen labeled points in our plots yet. Why so?

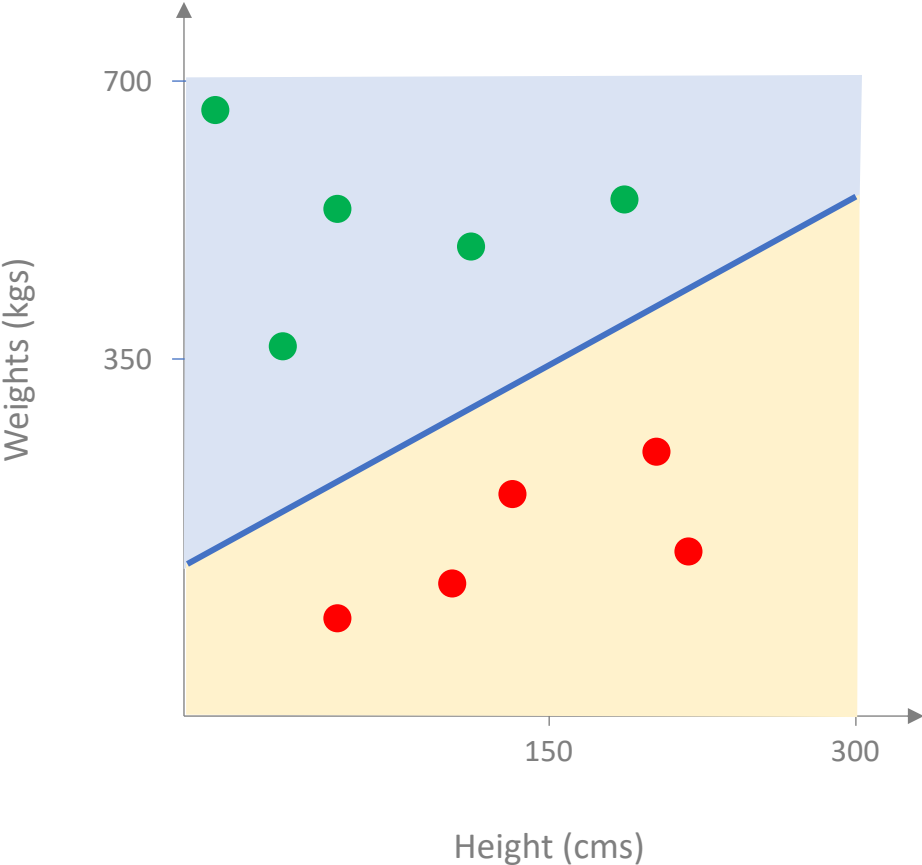
The fact that you can talk about a model independent of the training data is precisely the reason ML is so effective: i.e. traditionally it has been important for the models to be defined **independently of the problem/data** so that one can reason about them independently of the problem/data. This abstraction is what gives ML its power. In particular, this abstraction allows one to use the same technology for different problems/data and allows for the wide-spread use of ML in many areas. Basically if one can do the seventh step of the ML pipeline (and beyond) independently of the first six steps, then one can use "off-the-shelf" ML systems for the seventh step and beyond of the ML pipeline and for a particular application, one can just concentrate on the problem-specific tasks. In particular, any advance in the non-problem specific steps of the ML pipeline can then be used for any problem where one has handled the first six steps.

As mentioned above, the argument above is the current/traditional way ML systems are used. However, as we will see later in the course, such a clean separation between the first six steps of the ML pipeline and rest leads to situations with non-ideal outcome especially when it comes to questions of fairness or bias. The high-level reason for this is that even though we have drawn the ML pipeline as a sequential set of steps, in real life they are not so. Specifically, the fact that every step of the ML interacts with society means that the various are *not independent of each other* even though our flowchart might make it look otherwise.

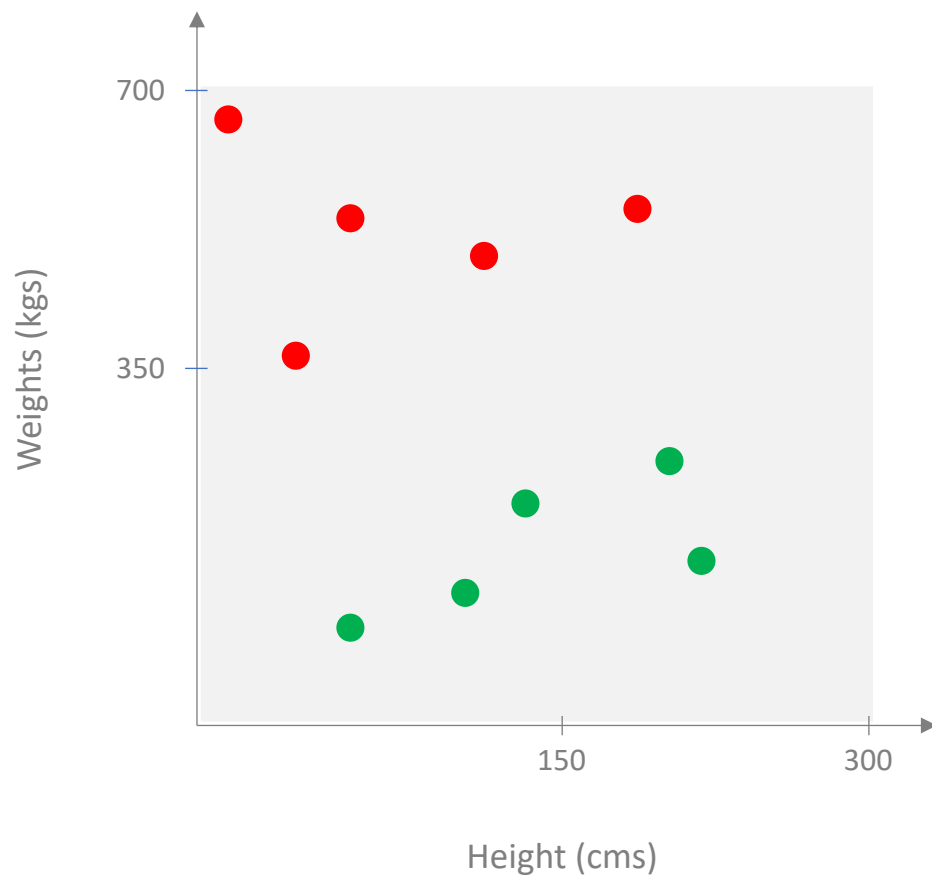
# Is one model enough?



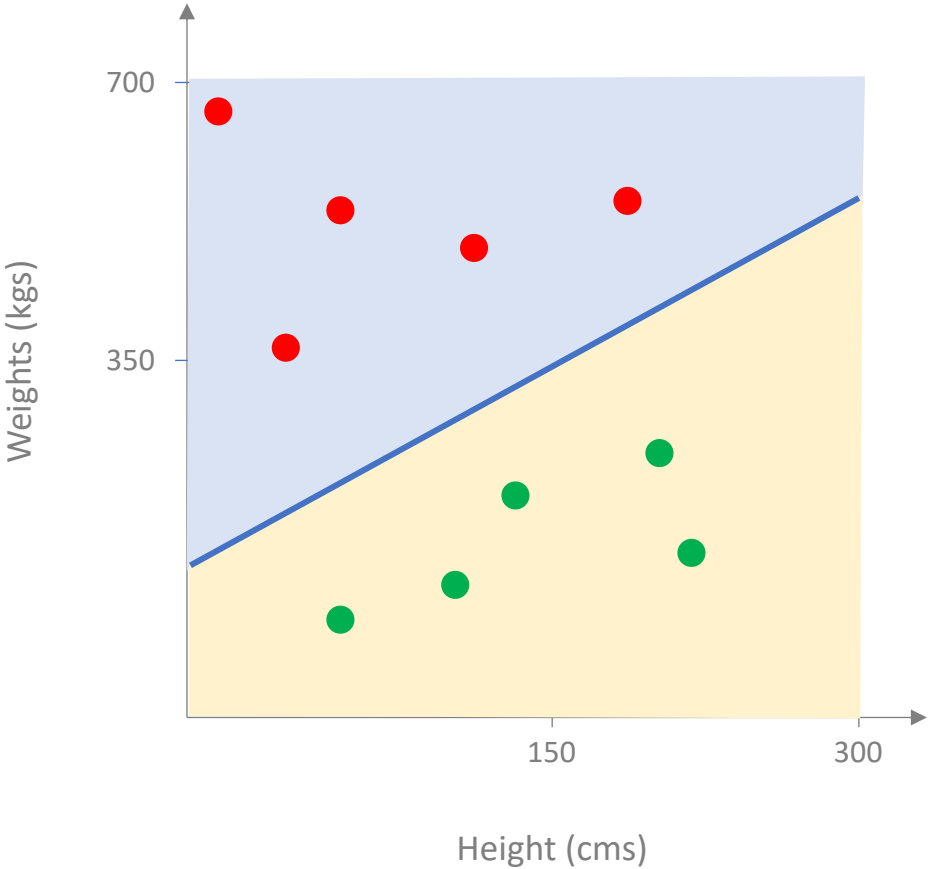
# The earlier linear model works!



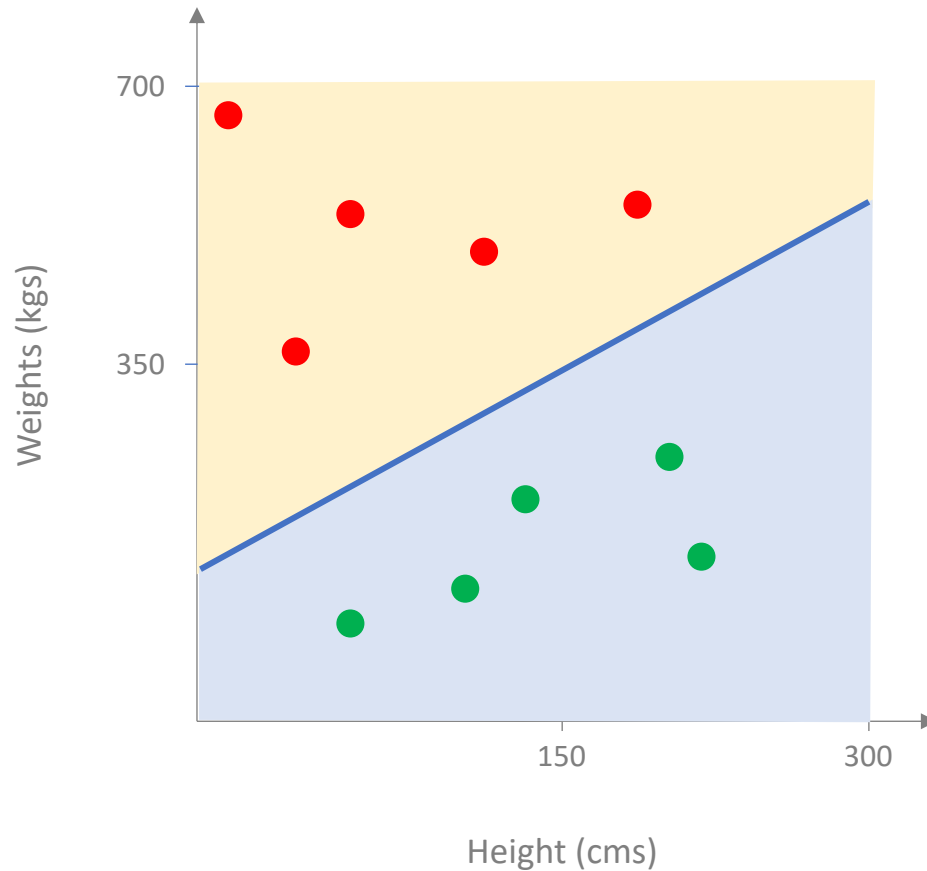
# What if the labels are flipped?



# The earlier linear model is terrible!

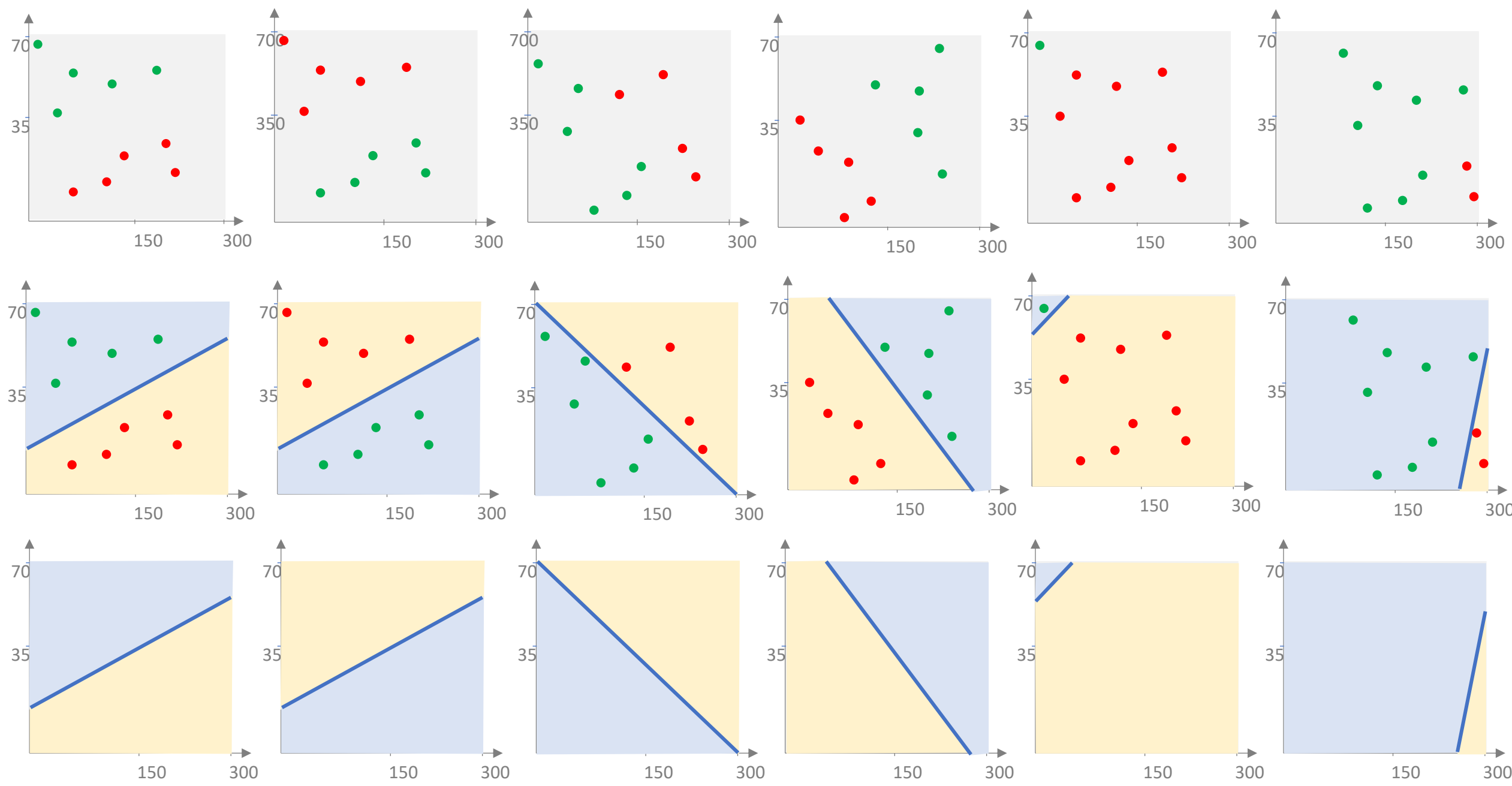


But the solution here is simple!



Is one linear model and its “complement” enough for all dataset?



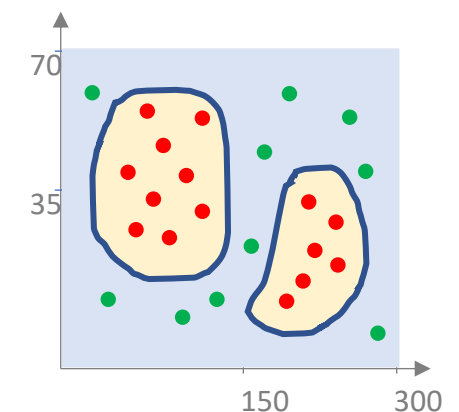
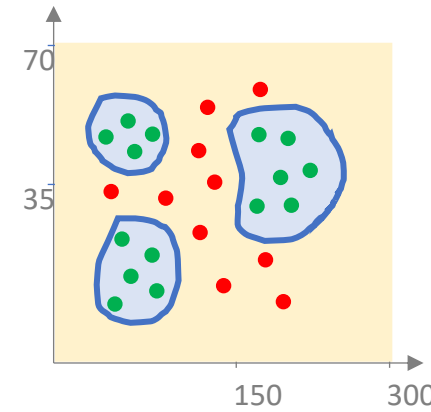
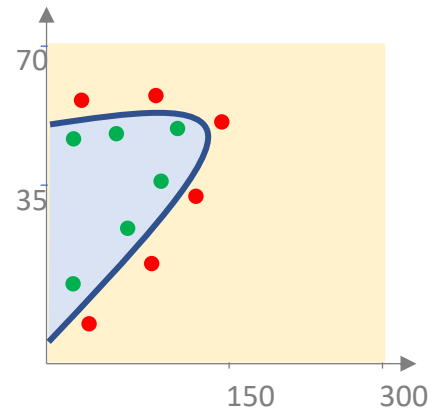
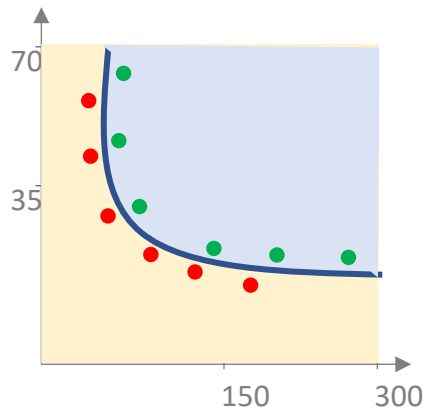
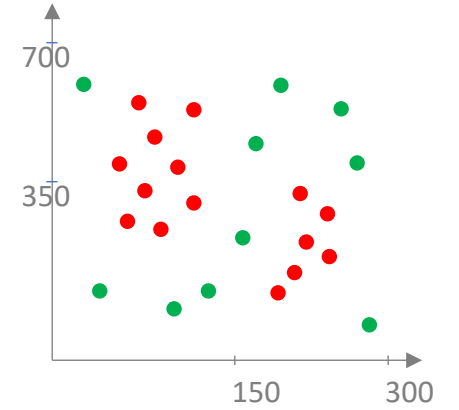
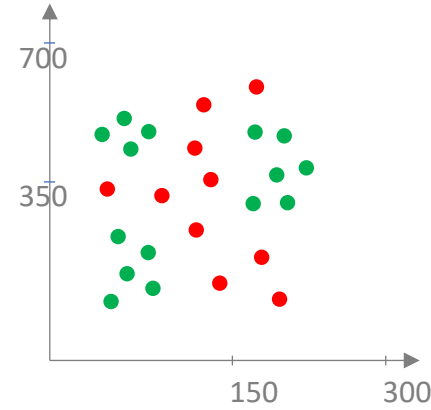
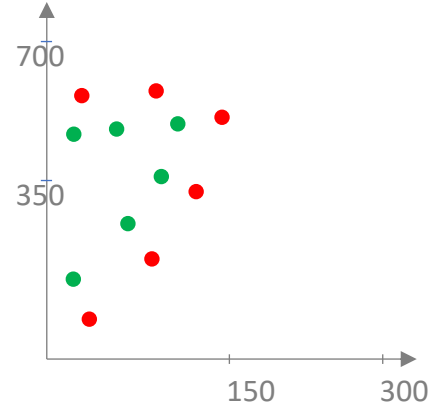
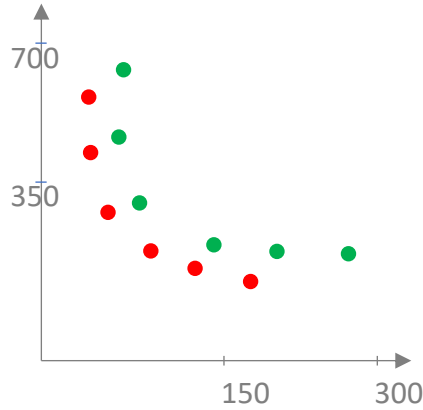


We need a *class* of models

Q1: Will linear models be enough for all datasets?

Q2: How should we define the *class* of linear models?

# Can a linear model explain any of these?



# Can a finite list of linear models work?

## Exercise

Convince yourself that having a *finite* set of linear models will not be able to represent all linear models (without ignoring precision constraints: i.e. assume we can represent any number to arbitrary precision).

**Hint:** If you pick a finite set of linear models, since there are infinitely many possible linear models (see the [section on linear models](#)). This means that your finite class will not represent a linear model. Can you use this linear model to construct a linear model that does not fit with the finite number of linear models in your class.

How would you search among the infinitely many linear models (say in model training)?

Need to include ALL infinitely many linear models

# Does there always exist non-linear model?

## Why Yes?

Convince yourself that given **any** dataset there is **always** a (possibly non-linear) model that fits it perfectly.

**Hint**: Given a dataset, can you use the dataset itself to define the model fits it perfectly? (Do not worry about how complicated the resulting model will be-- you just need to argue that such a model exists.) And do not peek below before you have spent some time thinking about the answer :-)

**NO: model training will not work**



# What properties should a model class have?

## Parsimonious representation

We would like to our model class such that its representation size does not depend on the size of your dataset (or has a more reasonable dependence than having to store the entire dataset to figure out the best fitting model).


In addition to representation size, models with small representation size can be considered to be the "right" model based on the [Occam's razor](#), which says that the simplest explanation is probably the best solution. This is also related to the notion of [generalization](#) (which we have [briefly seen before](#)).

It turns out that all the model classes that are used in an ML pipeline has this properties (though the notion of "small" changes from class to class).

## Efficient model training

In the later step in the ML pipeline of model training, the technical task is the following: given a class, figure out the best fit model from the given class. Of course, we would like to be able to do this step *efficiently*.

And it turns out that while sometimes it is possible to figure out the best fit model (e.g. [linear models](#)), this is not always possible (e.g. for [neural networks](#)). In the latter case, we often try to compute an approximation. If we are given a specific point and a model that has the smallest possible misclassification error, we might be able to find a model that makes



Are these  
properties  
enough?

## Efficient prediction

In the later step in the ML pipeline of prediction, the technical task is the following: given a specific model, figure out the label the model will assign to the point. Of course, we would like to be able to do this step *efficiently*.

We would like to point out that model training is basically done once while prediction is done over and over again once the training is done. Hence, while we want both efficient training and prediction, the notion of "efficient" is different for both tasks. E.g. in some cases, it is not unreasonable spend days trying to do model training but the corresponding prediction has to be done in (fraction of) a second.

# Other desirable properties

## Model expressivity

What is missing from the above properties is any constraint on how good the model class is at predicting datasets? In other words, for "real life" data, how accurately can this model class predict the correct labels?

The quotes around "real life" data is on purpose: it is a fool's errand to try and precisely define what real life data looks like. There are two ways around this: to propose certain natural conditions on the data and then theoretically show that under these theoretical conditions on the data, the best model in your class will predict the data labels with small error (and presumably you also experimentally show that at least some representative datasets satisfy the theoretical conditions). Alternatively, you run experiments and show that on some representative datasets the empirical prediction errors of the best model in your class is small.

## Other desirable properties?

As we progress in the course, we will see that there are properties other than the ones above (which are the primary focus of traditional ML) that could be useful. E.g. does the model make it easy to "explain" its decision to a human? Can we argue that in some precise sense the model is "fair" or not "biased."

Can you think of other useful properties?





# Today's pass phrase: **Daphne Koller**

## Daphne Koller

From Wikipedia, the free encyclopedia

**Daphne Koller** (Hebrew: דפנה קולר; born August 27, 1968) is an Israeli computer scientist, a professor at Stanford University and a MacArthur Fellowship recipient. Her research is in artificial intelligence and its applications in the field of medicine. She is the author of the book *Emerging Technologies That Will Change Your World*.

### Contents

- Education
- Career and research
  - Honors and awards
- Books
- Personal life
- References

## Education

Koller received a bachelor's degree from the Hebrew University of Jerusalem in 1988, at the age of 19. She completed her PhD at Stanford University in 1993.

## Career and research

After her PhD, Koller did postdoctoral research at the University of California, Berkeley in the Computer Science Department in 1993. She joined the faculty at Stanford University in 1995. She was elected a fellow of the Association for Computing Machinery in 2011 and was elected a fellow of the National Academy of Sciences in April 2016. Koller was awarded the first ever \$150,000

## Coursera Blog

Degrees For E

## Founders



## Daphne Koller

### CO-FOUNDER OF COURSERA

Daphne Koller co-founded Coursera in 2012. She served as the company's Co-CEO until 2014, and then as President until 2016.

Daphne has recently founded Insitro which applies machine learning

[Read More](#)

## Daphne Koller



Koller in 2009

<b>Born</b>	August 27, 1968 (age 51) Israel
<b>Nationality</b>	Israeli
<b>Alma mater</b>	Stanford University (PhD) Hebrew University of Jerusalem (BS, MS)
<b>Known for</b>	Machine learning Graphical models MOOCs

# Linear models in more detail

Assume only ONE input variable

## Exercise

Given a specific target variable (e.g. in our running example whether a human has high or low risk of heart disease) can we even hope to predict it with a *single* input variable?.

**Hint**: Given a target variable, can you think of a variable that can predict it (perfectly)? (If you think the answer is trivial, you are correct :-)). Think about it yourself first before peeking below!

# Body-Mass Index

## Body Mass Index

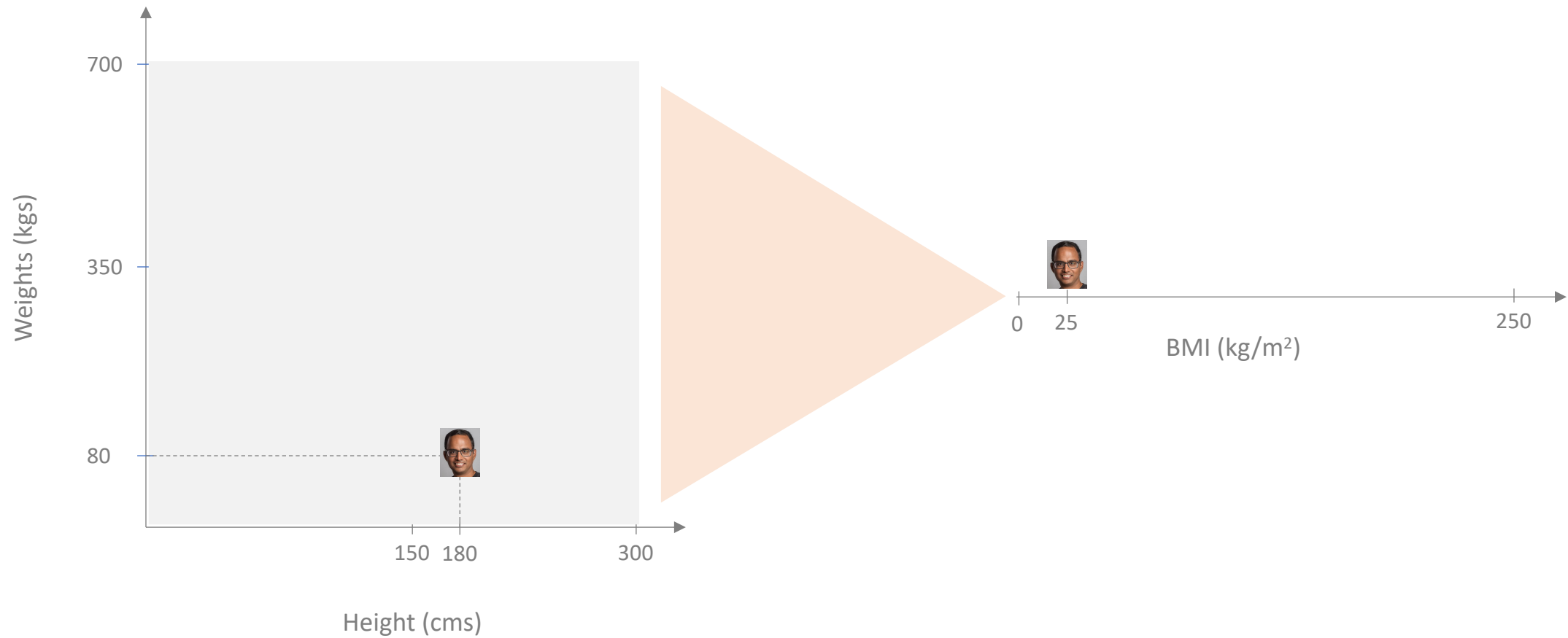
The [Body Mass Index](#) (or BMI) is defined as follows. Let  $w$  and  $h$  be the weight and height of someone in kilograms and centimeters respectively (as we have seen so far). Then their BMI  $b$  is defined as follows:

$$b = \frac{w}{\left(\frac{h}{100}\right)^2}.$$

This is equivalent to the more prevalent d

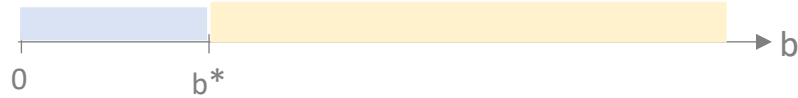


# Moving from 2 variables to 1



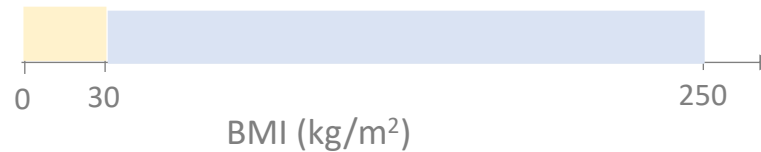
# Linear model for one variable

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



Why is this enough?

# Can one variable linear model be of any use?



## Healthy Weight

CDC | Healthy Weight | Assessing Your Weight



 Healthy Weight

Assessing Your Weight

## Body Mass Index (BMI)

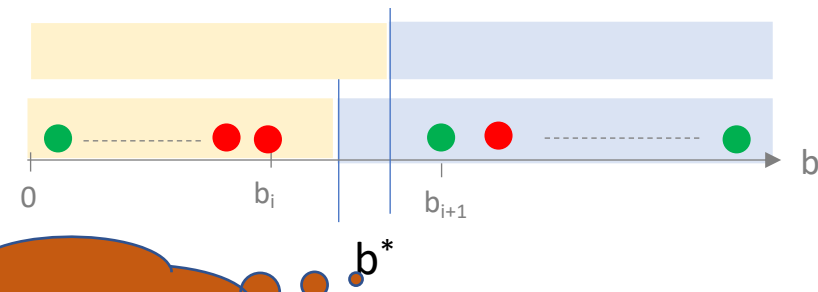
Español (Spanish)

# Parsimonious representation

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

How many parameters do we need to determine a linear model?

Infinitely many possible values for  $b^*$  !!!



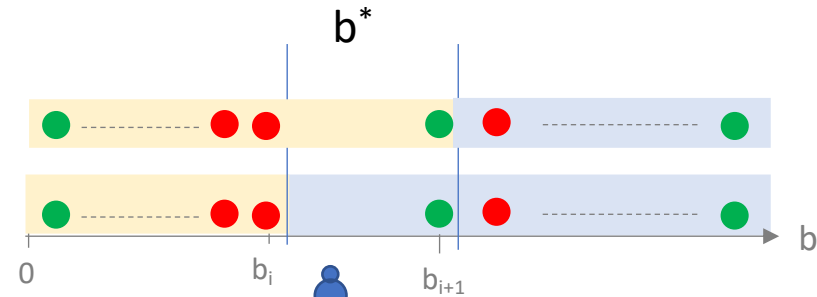
How many distinct values of  $b^*$  must be considered?



# Efficient model training

Given  $n$  choices for  $b^*$ , what is an algorithm to compute optimal linear model?

About  $n^2$   
steps



Update error values  
in constant steps?

# Other properties

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

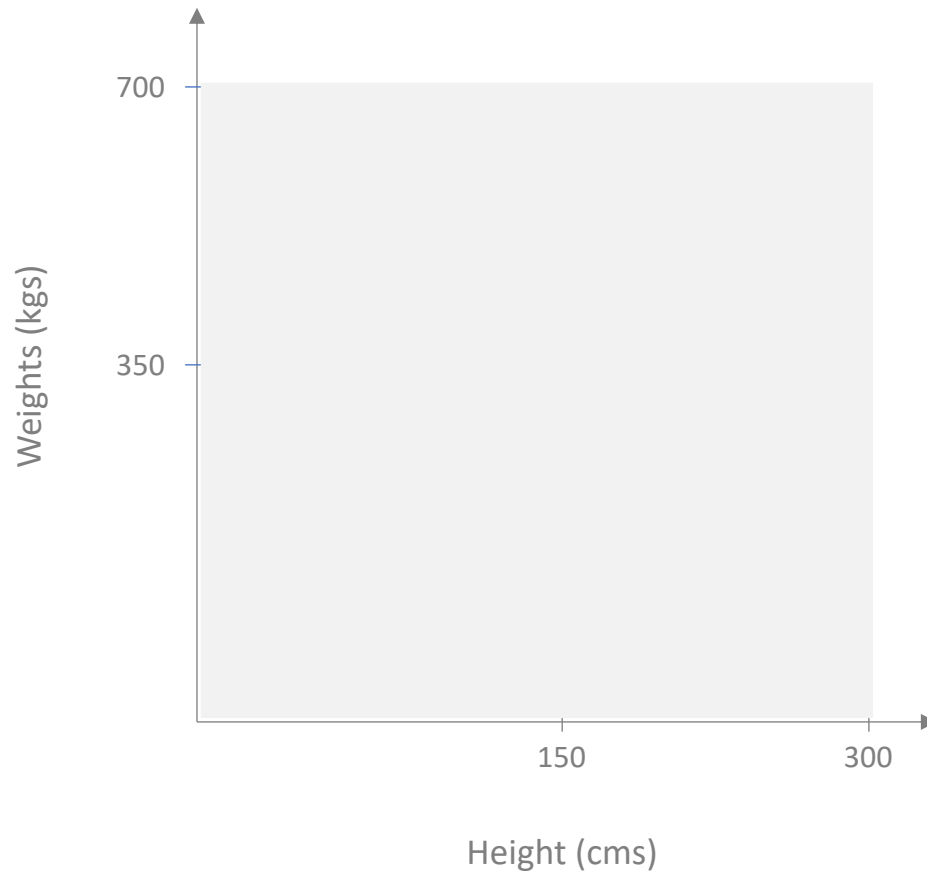
Efficient prediction?

Expressivity?

Explainability?



# Linear model in two variables

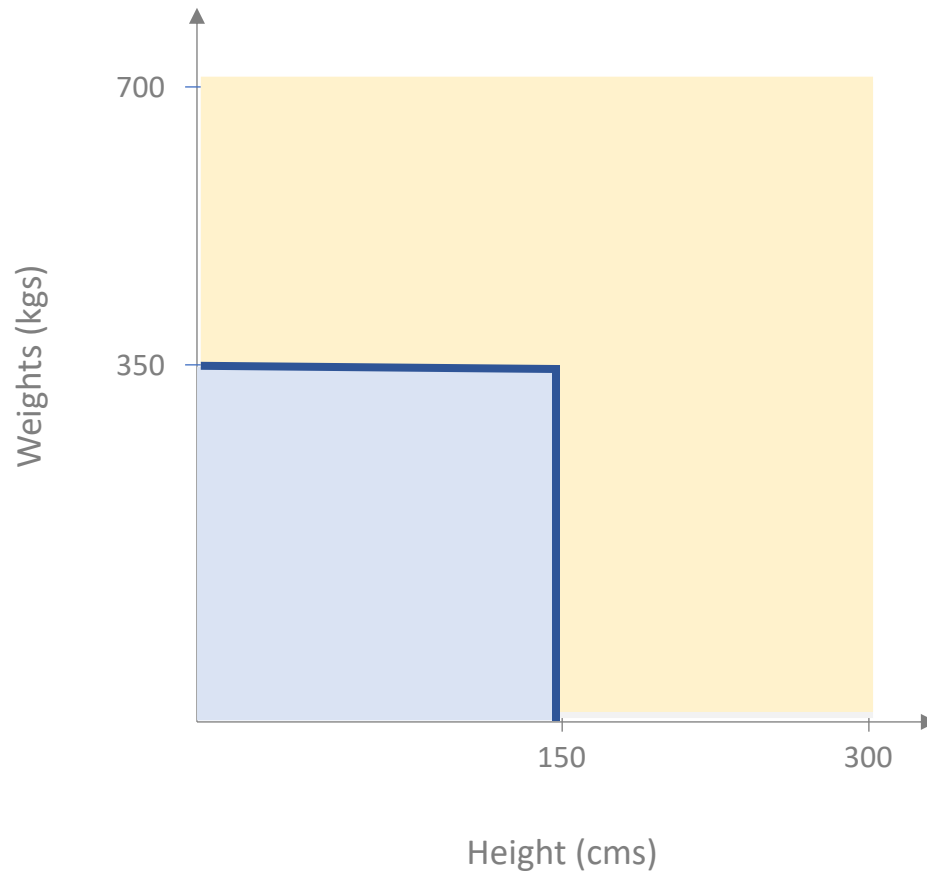


$$\ell(b) = \begin{cases} 1 & \text{if } b < b^* \\ -1 & \text{if } b \geq b^* \end{cases}$$



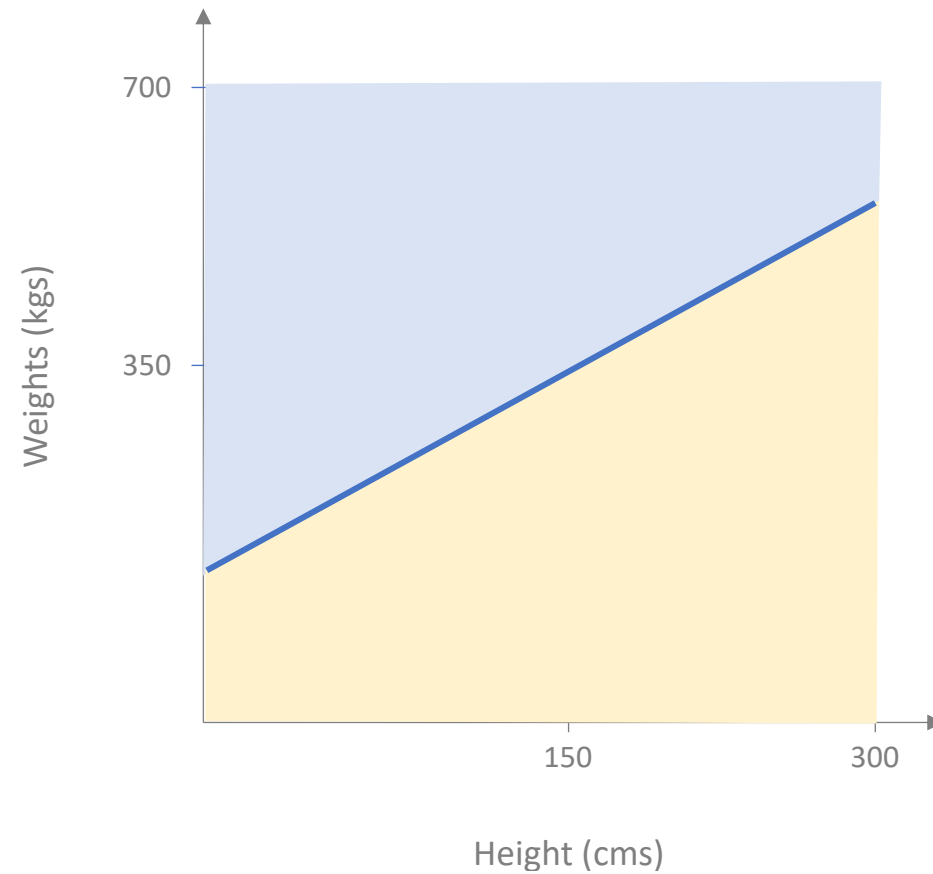
$$f(w, h) = \begin{cases} 1 & \text{if } w < w^* \text{ and } h < h^* \\ -1 & \text{otherwise} \end{cases}$$

# Is this a linear model?



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

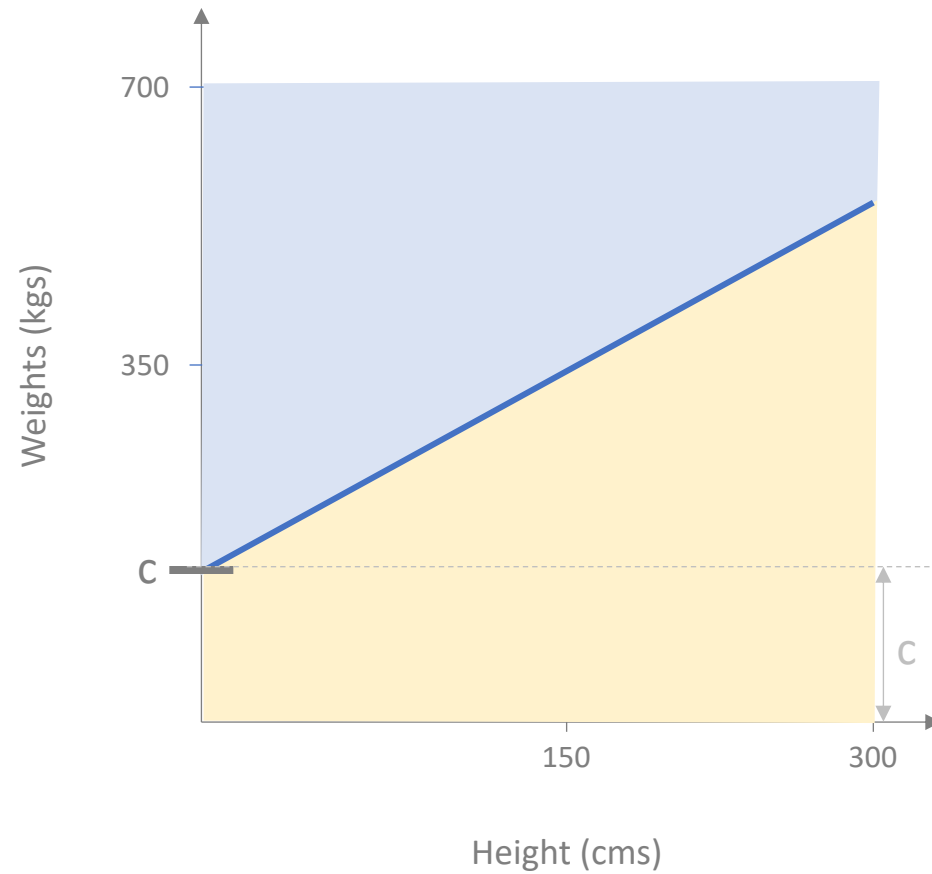
# Back to the real linear model



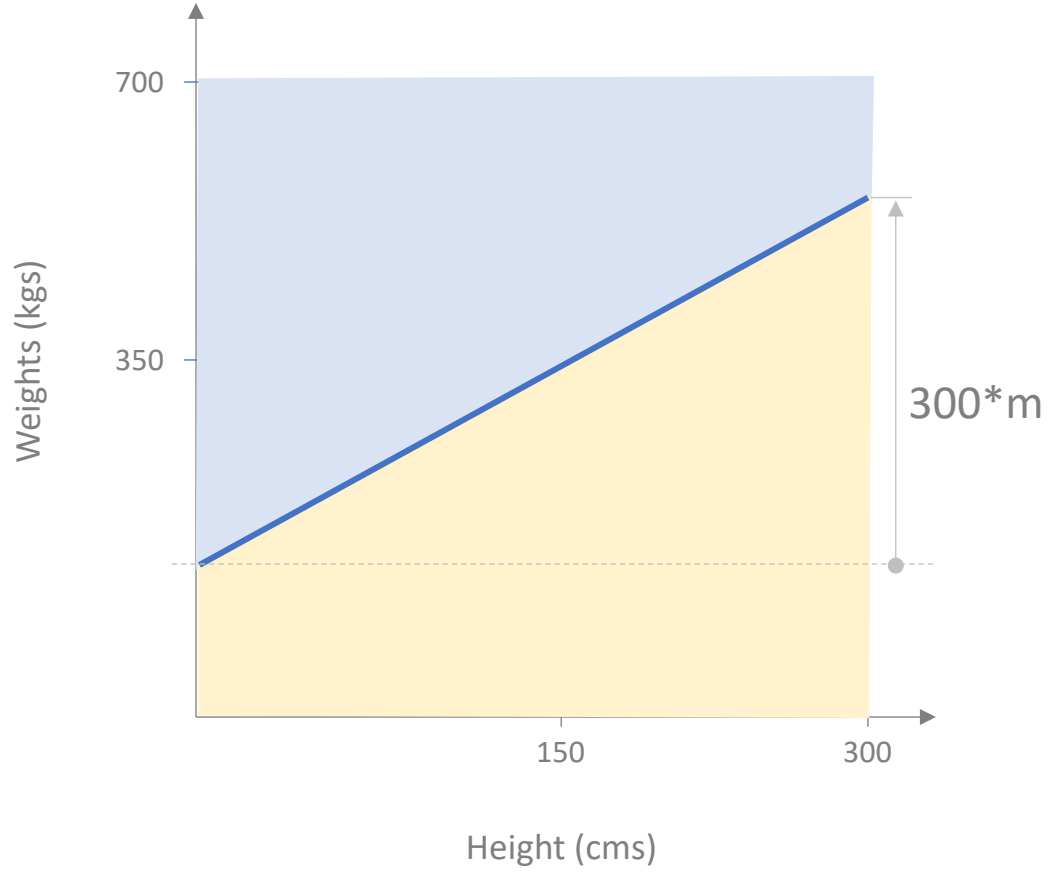
## Parsimonious representation

As in the case of one variable, which side of the line gets blue color is one parameter (or more precisely one [bit](#)). So the interesting question is how many parameters we need to represent the line itself.

# Intercept: parameter $c$

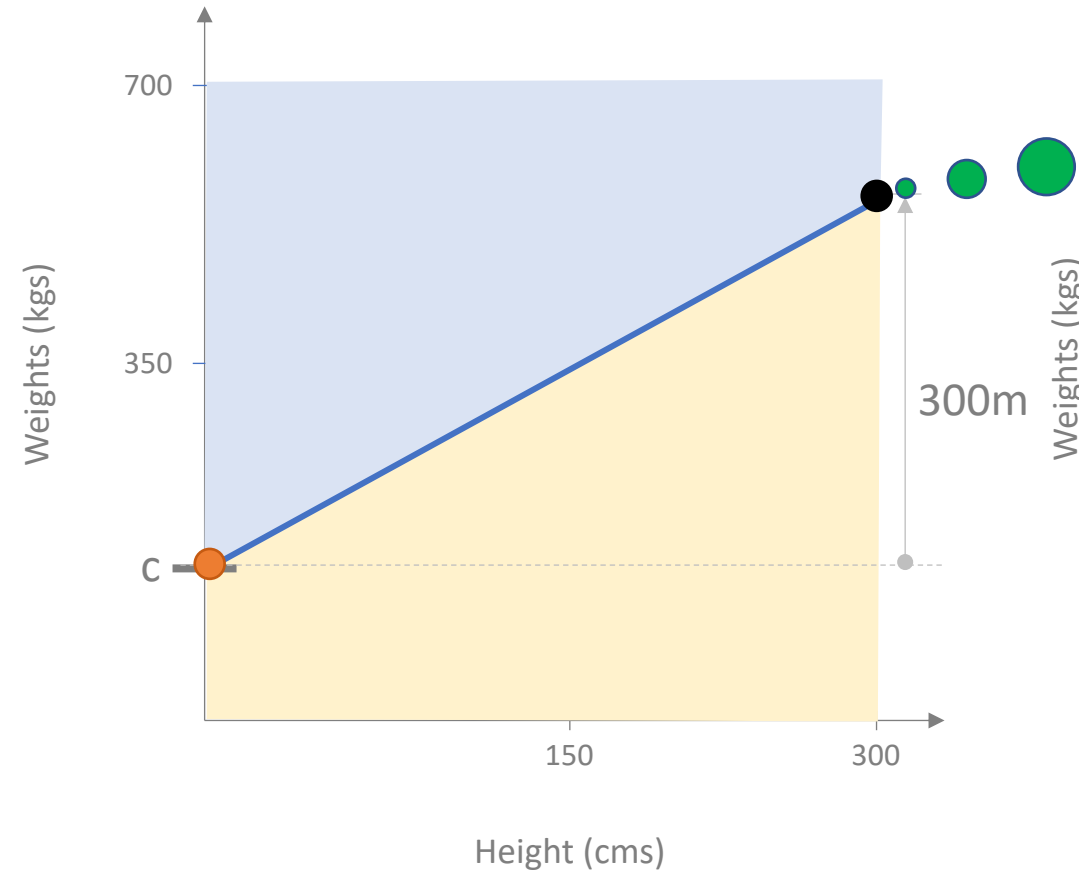


# Slope: parameter $m$





# We now have identified the line!



Total number of parameters?

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

# Efficient model training

## Efficient model training

It turns out that the situation for efficiently training a linear model in two variables is more tricky/complicated than the case in variables. In particular, as we have done so far, let us assume you want to find the optimal linear model for two variables for a dataset that minimizes the number of mis-classified data points. While there was a fairly simple/clean algorithm that was provably correct for the one variable case, there is simple algorithm for the two variable case (in other words, explaining those algorithms is out of scope for this course). However, if you are curious, there exists such an algorithm that can solve the model training problem on dataset with  $n$  2D points in roughly  $n^2$  steps (see the 2012 paper by Aronov et al. [1] for more on this, though the specific result I mentioned appeared 1993 paper by Houle [2]).

Unfortunately, unlike in the one variable case we cannot improve the runtime of this algorithm to scale linearly in  $n$  (doing so would improve the best known runtime for a large number of problems [3]). By contrast, checking if there is a linear model that has no mis-classification can be solved in linear in  $n$  number of steps [4].

Of course folks use linear models in real life and they are very efficient. So what gives? The idea is that these algorithms do not aim to get the optimal linear model and in fact for datasets that are not perfectly explained by a linear model, there are no known theoretical guarantees on how the quality of the approximation. However, these algorithms work pretty well on real-life datasets. We will come back to these algorithms when we study the model training part of the ML pipeline.

# Other properties

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

Expressivity?

Explainability?

Efficient prediction?

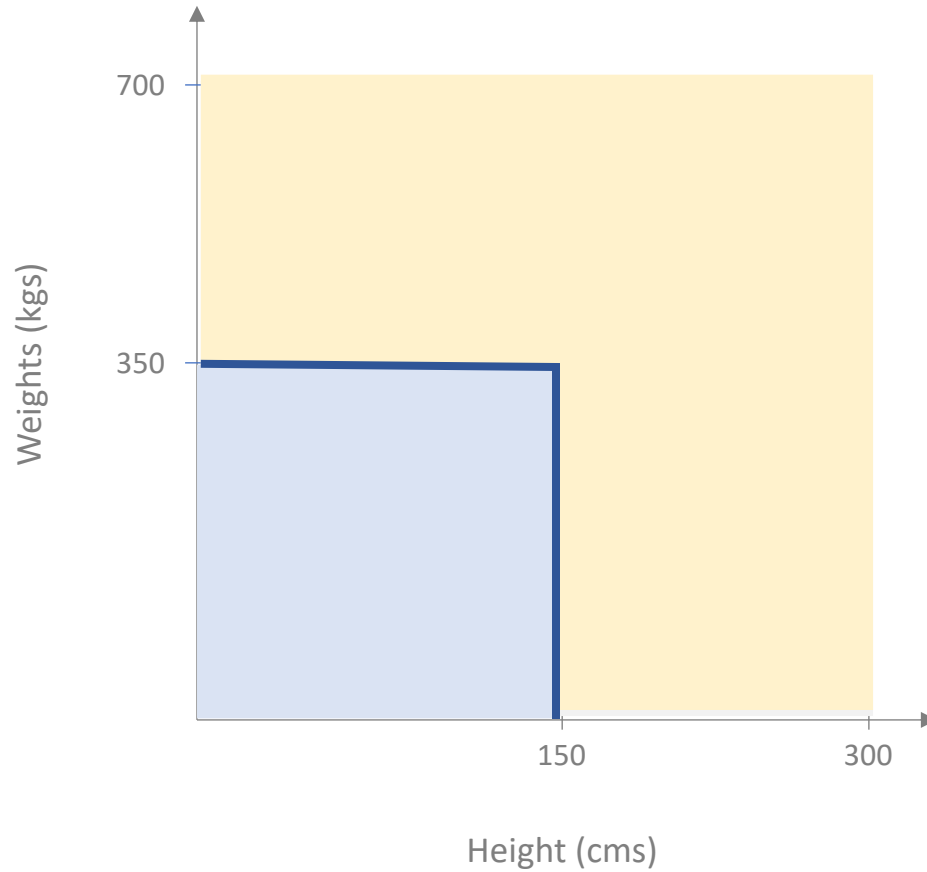
More than two variables?

# Decision tree models

Heard of 20 Questions?



# We already have seen a decision tree



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