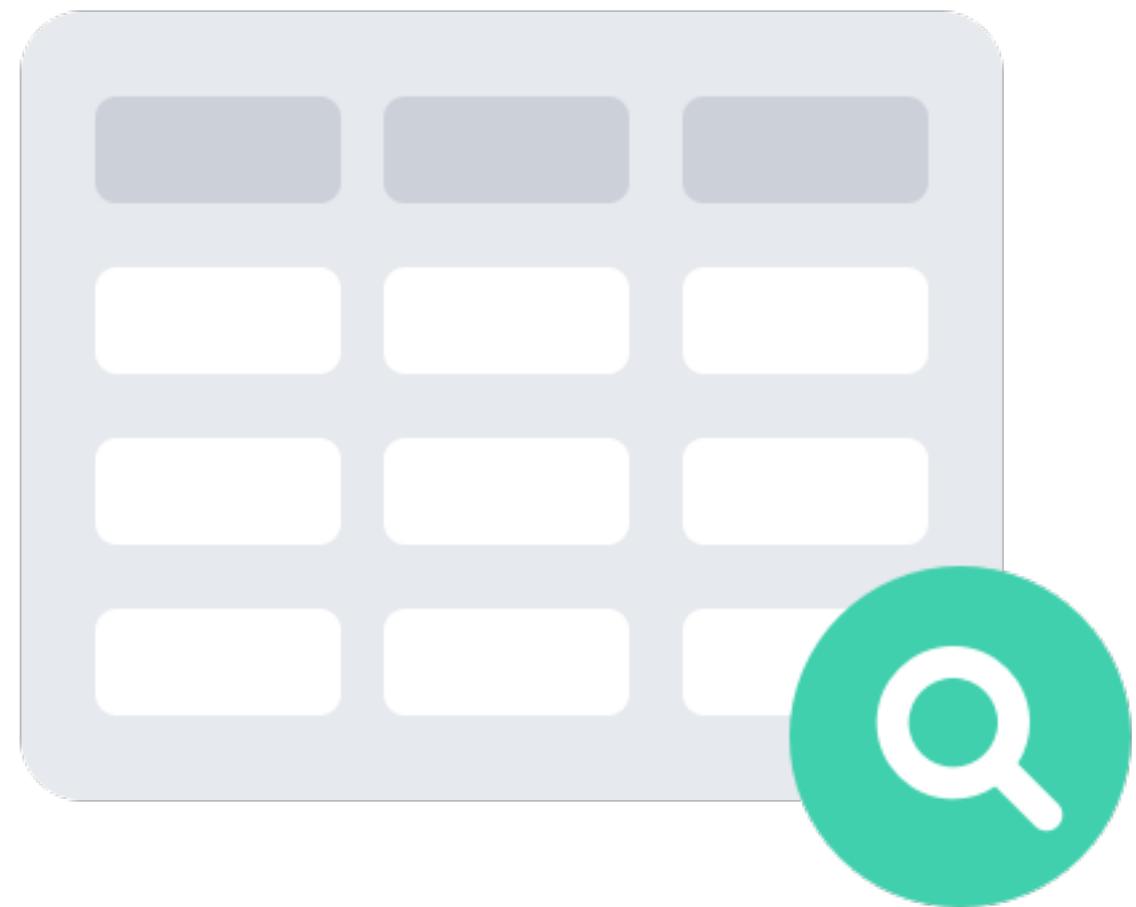
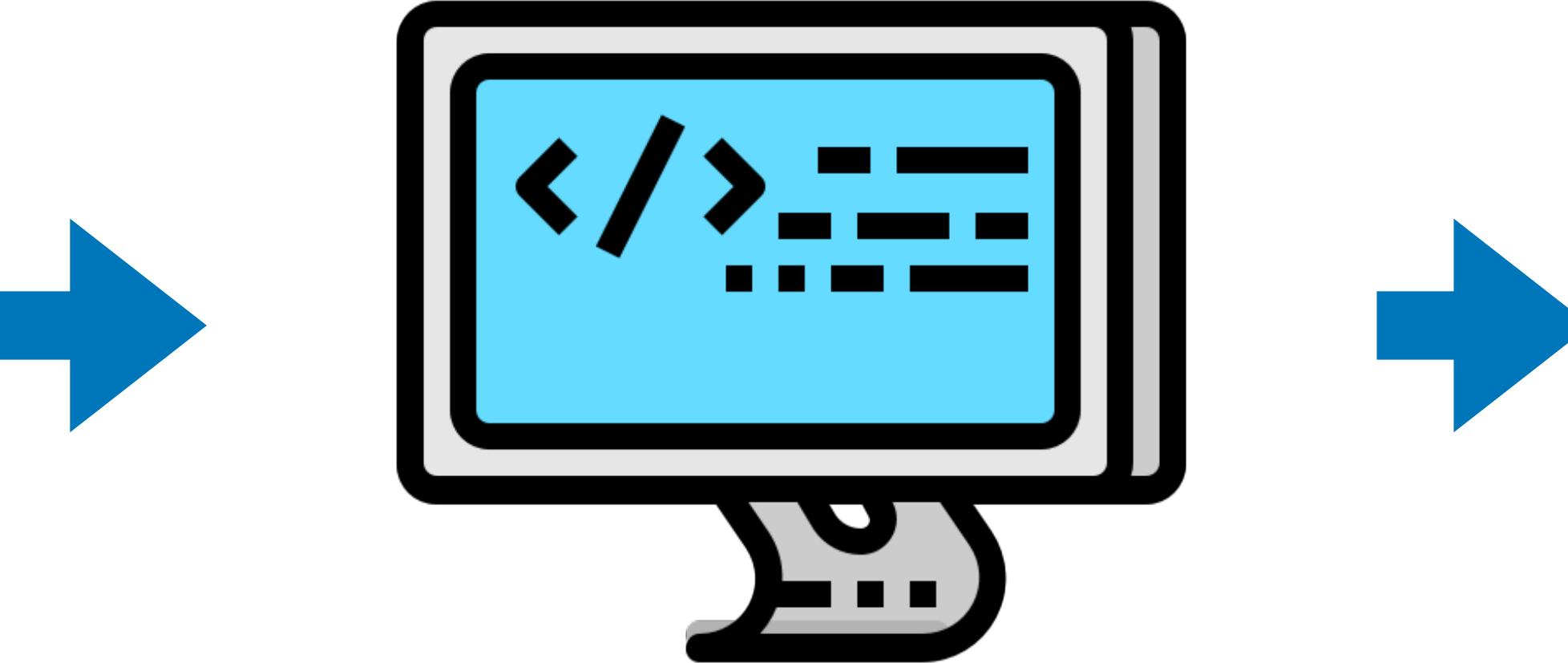


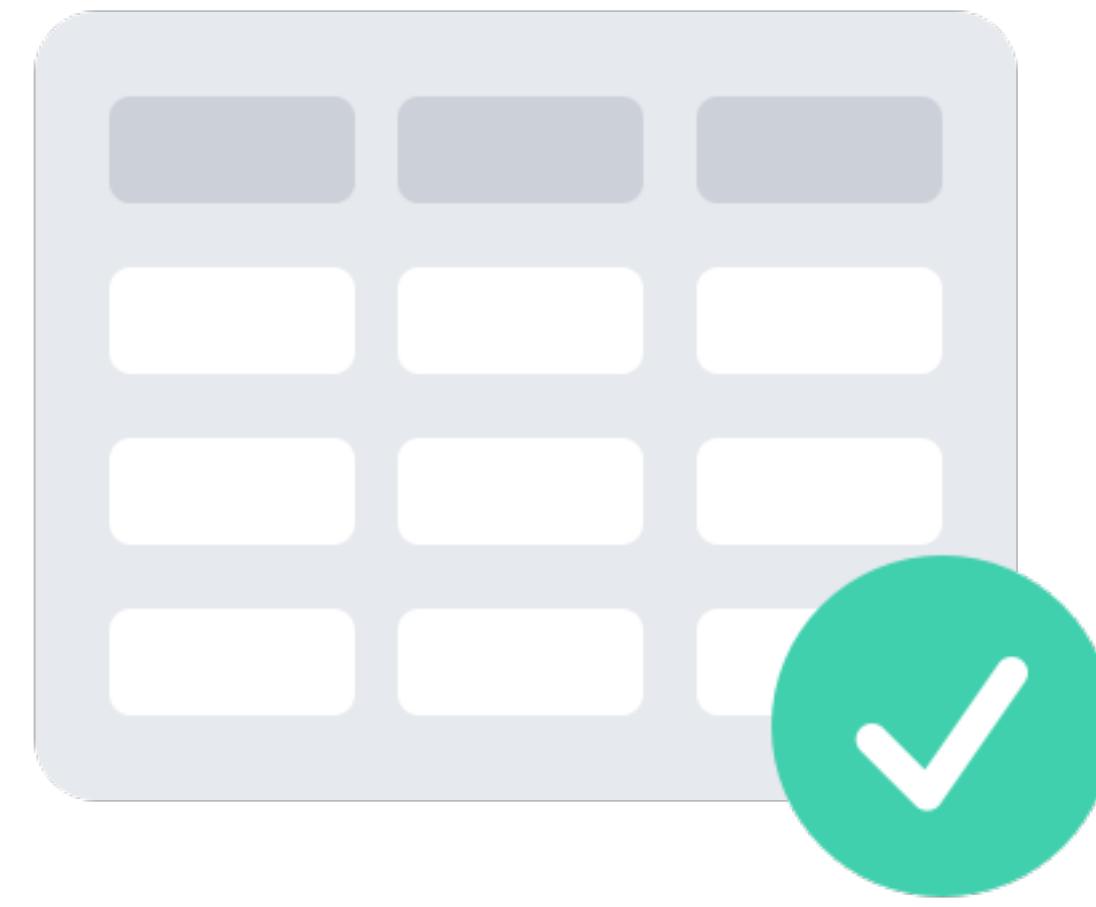
# **What is machine learning?**



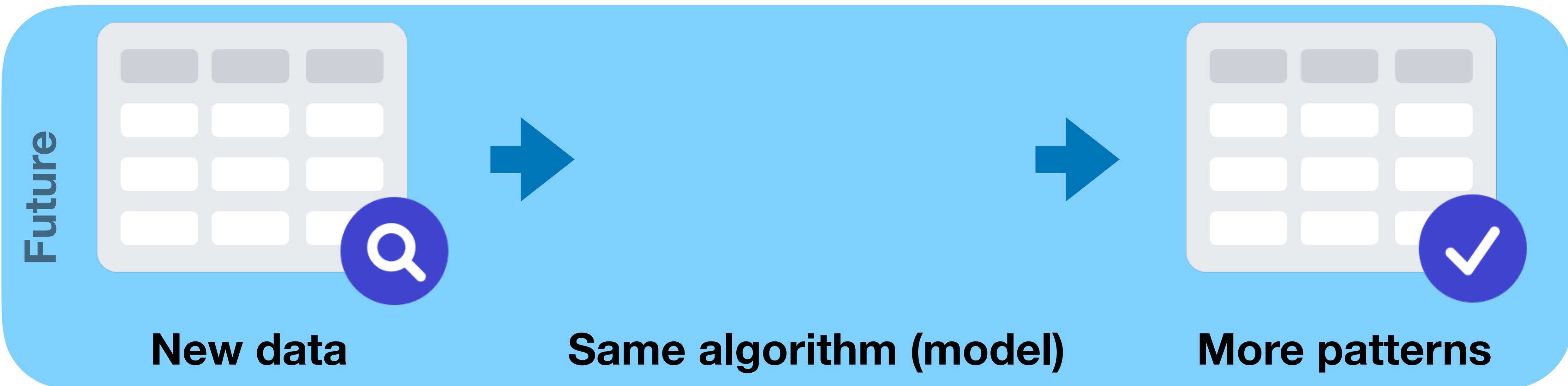
**Data**



**Machine learning  
algorithm**



**Patterns**



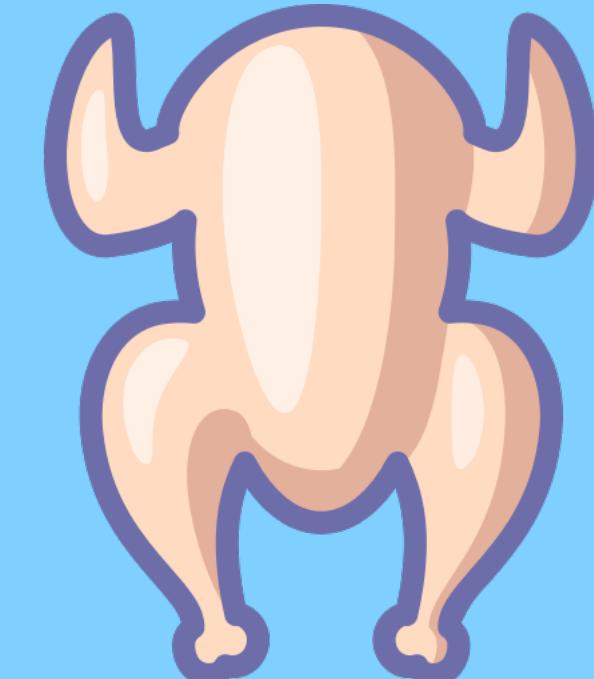
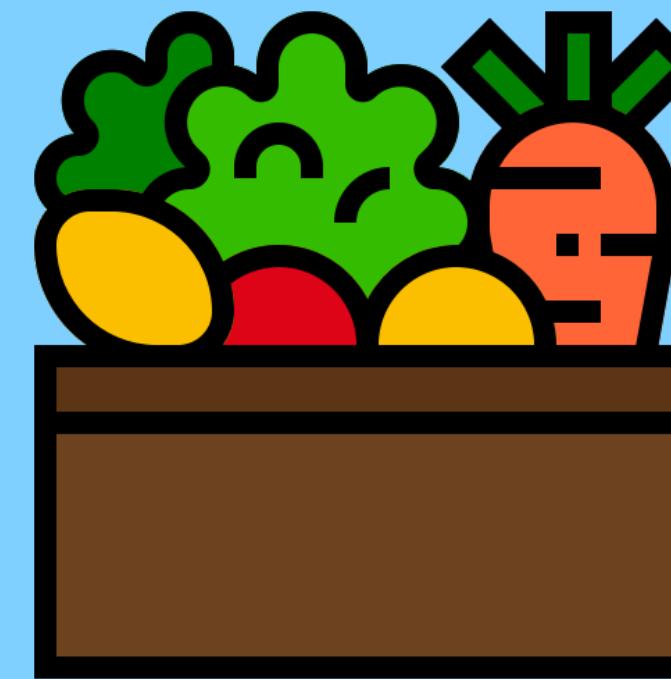
**Future**

**New data**

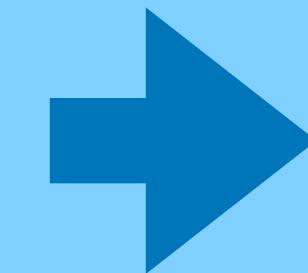
**Same algorithm (model)**

**More patterns**

## Normal algorithm



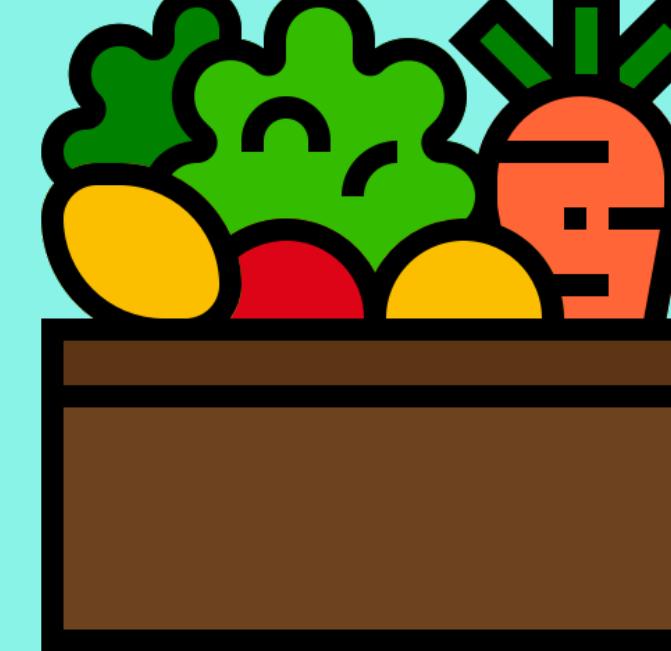
1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



**Starts with**

**Makes**

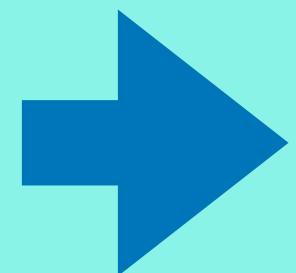
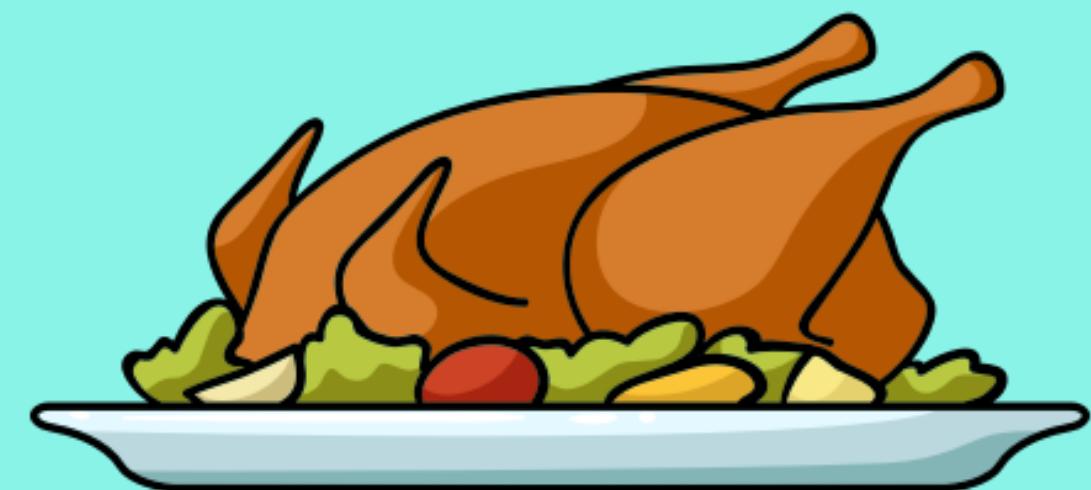
## Machine learning algorithm



**Inputs**



**Output**



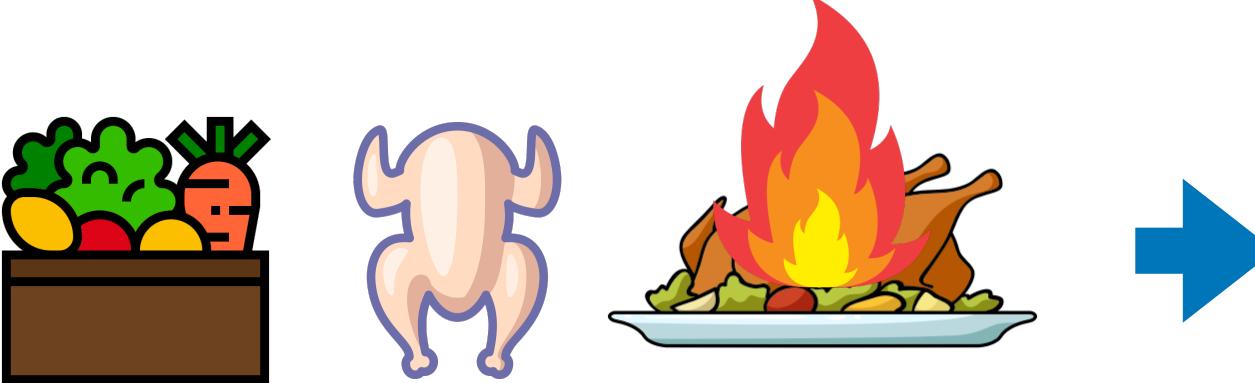
1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables

**Starts with**

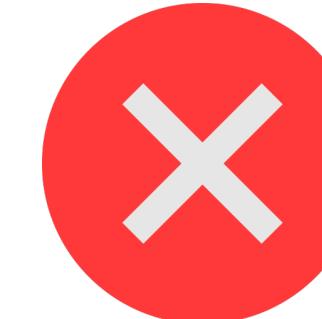
**Figures out**

# Attempt

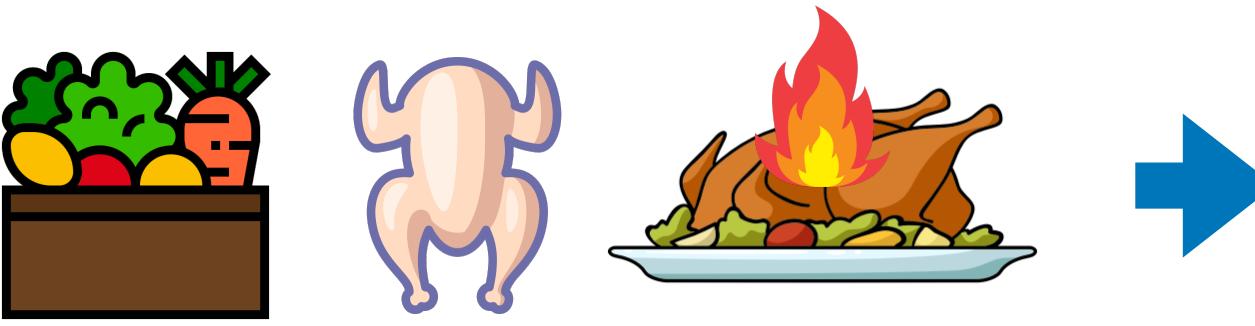
1



1. Cut vegetables
2. Season chicken with lots of spice
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



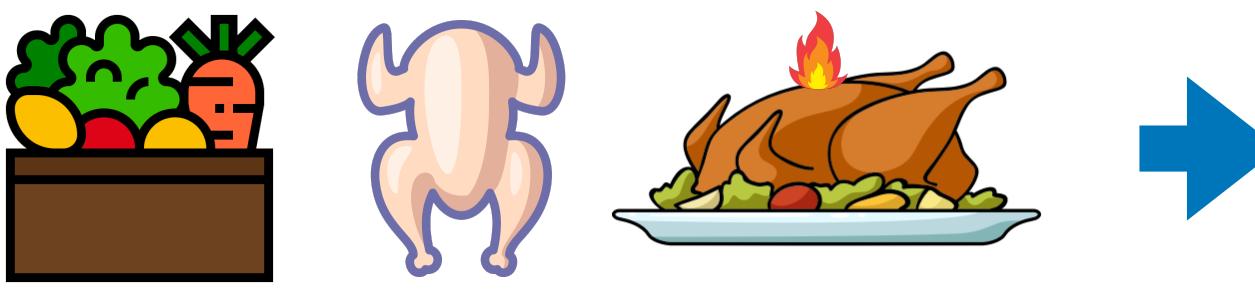
2



1. Cut vegetables
2. Season chicken with extra spice
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



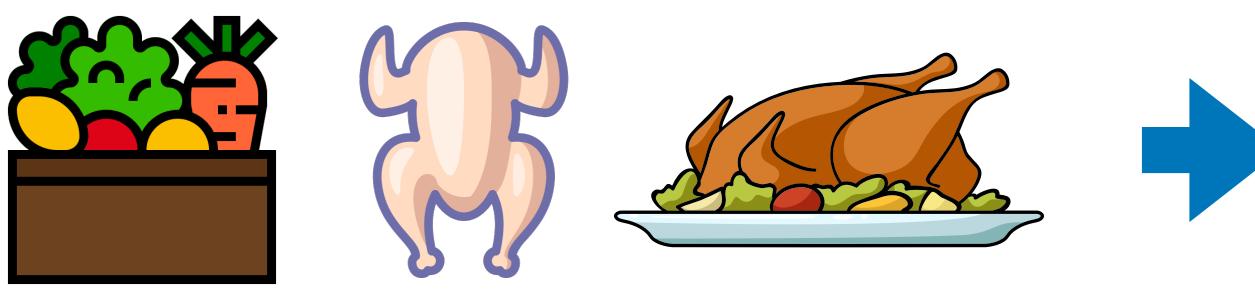
3



1. Cut vegetables
2. Season chicken a lil' extra spice
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



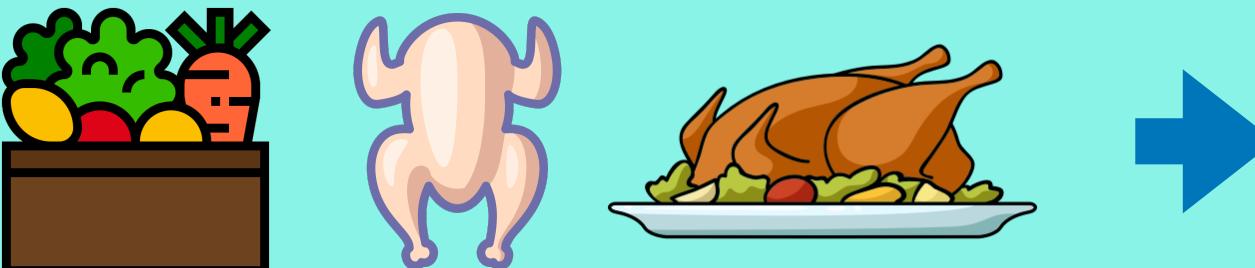
4



1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



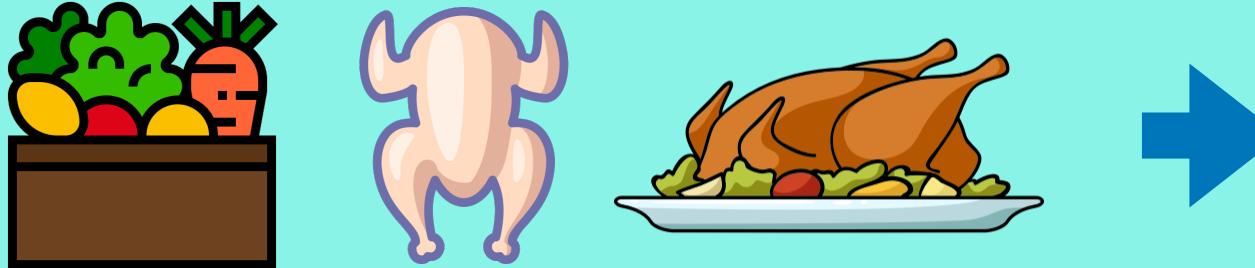
⋮



1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



100



1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables

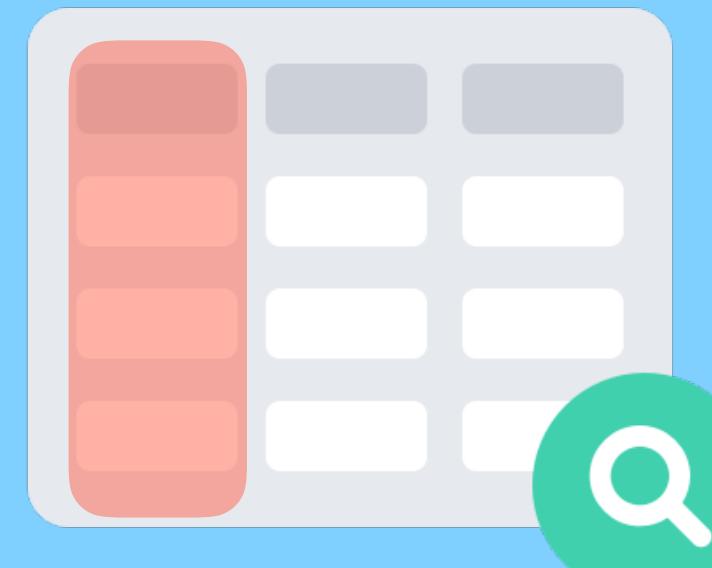
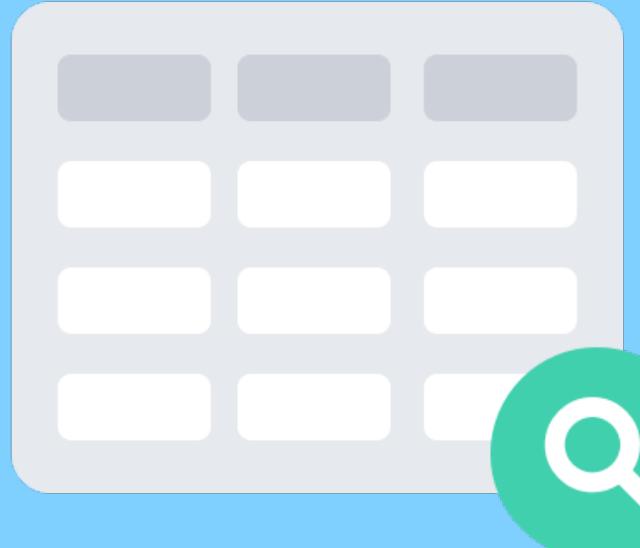


Machine learning algorithms  
may try 1000's of times to find  
the right instructions.

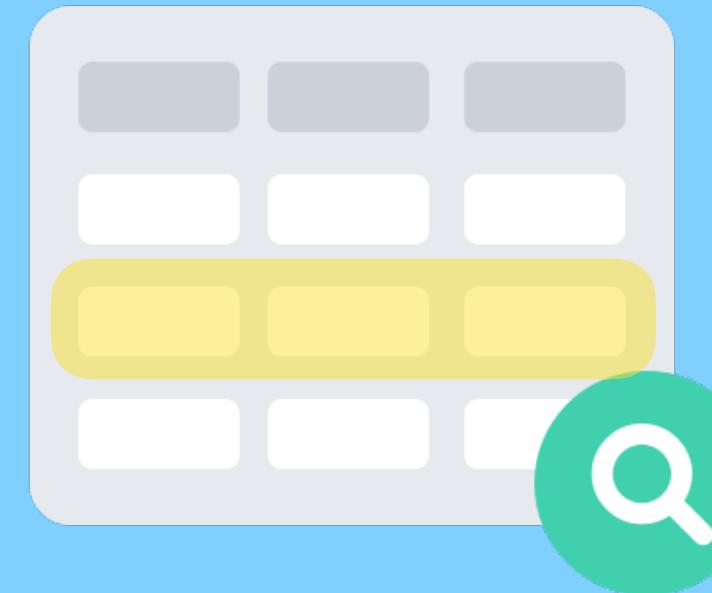
# Data science

## Data analysis

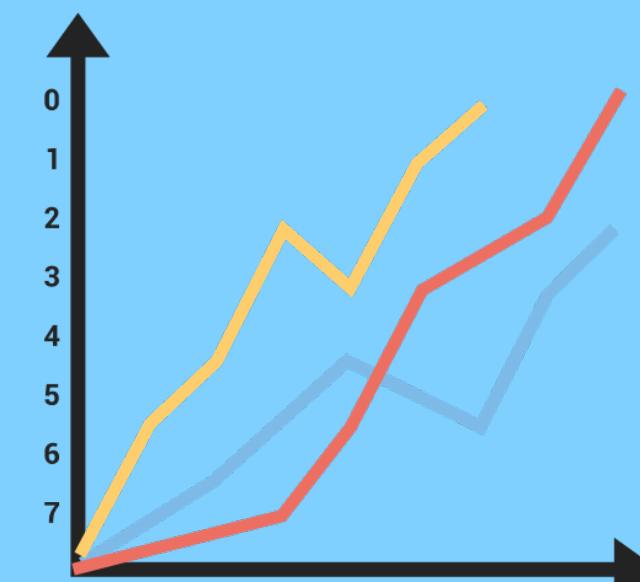
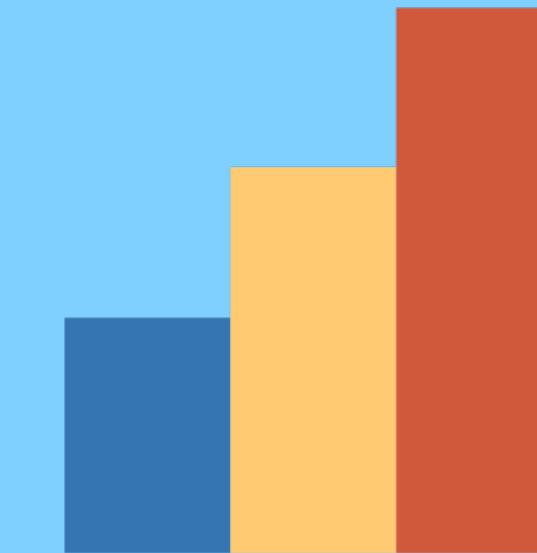
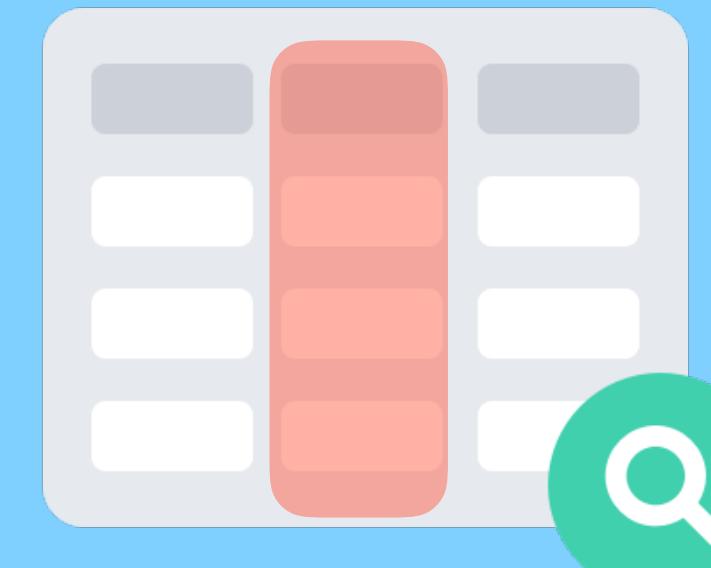
Data



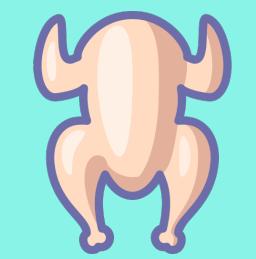
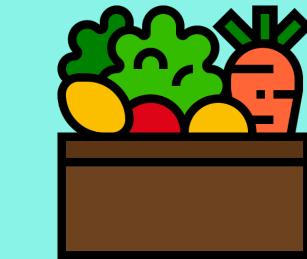
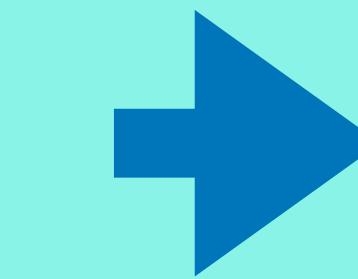
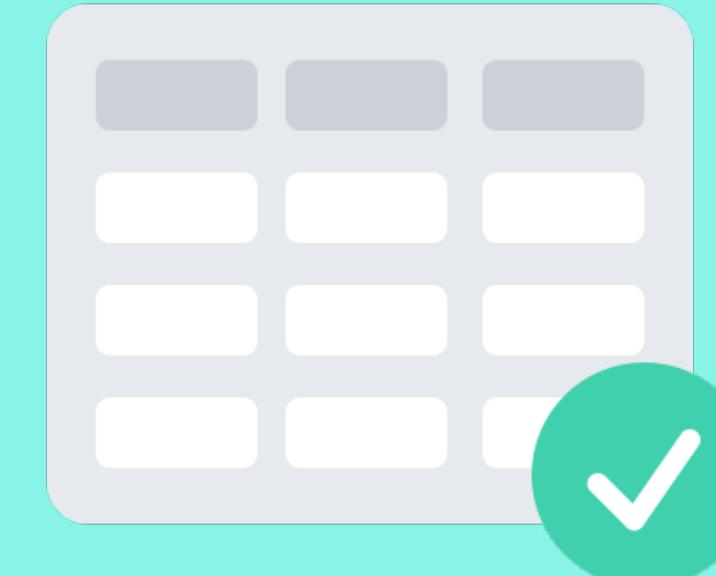
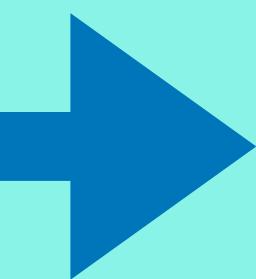
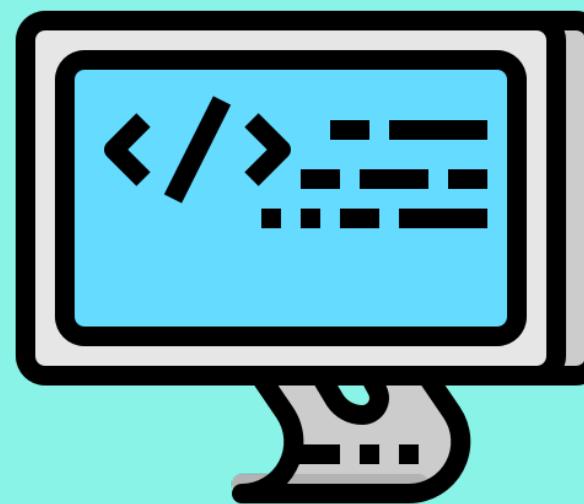
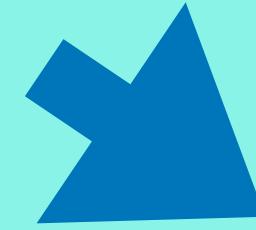
vs.



vs.



## Machine learning



1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables

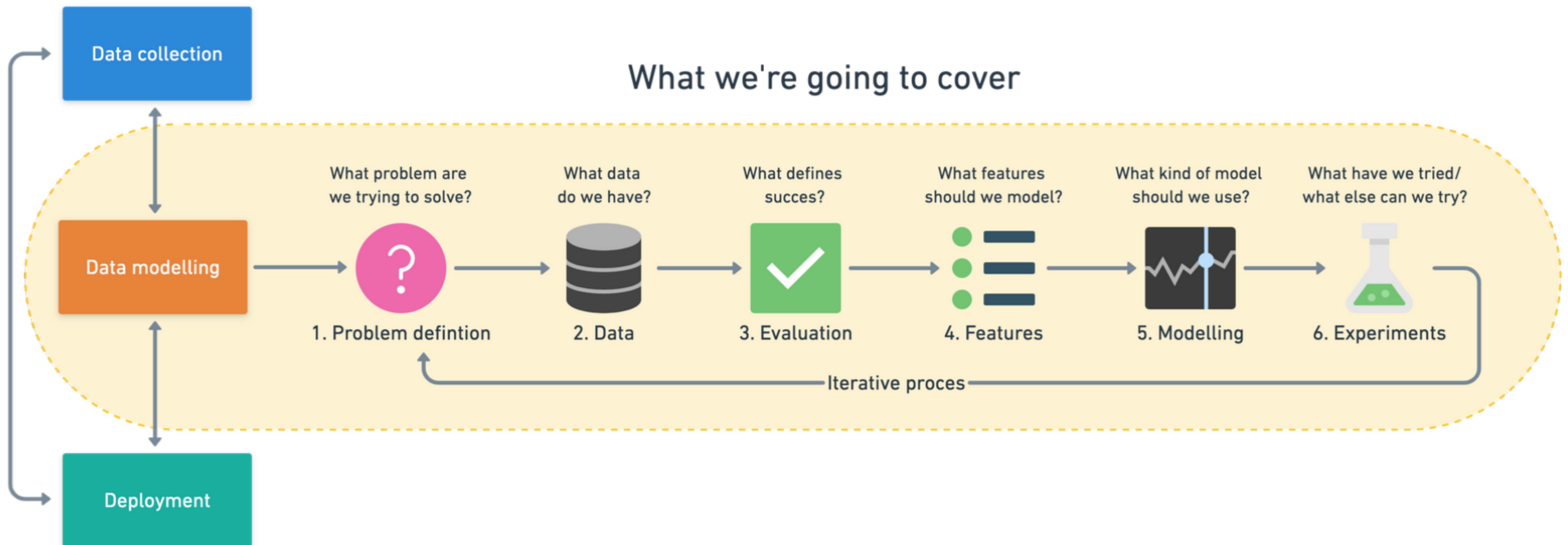
**Instructions in your daily life?**

**What we're going to cover  
(and what you'll finish with)**

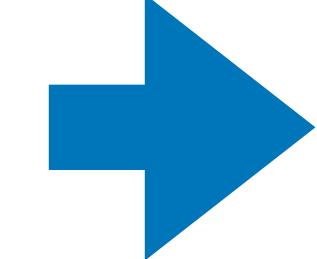
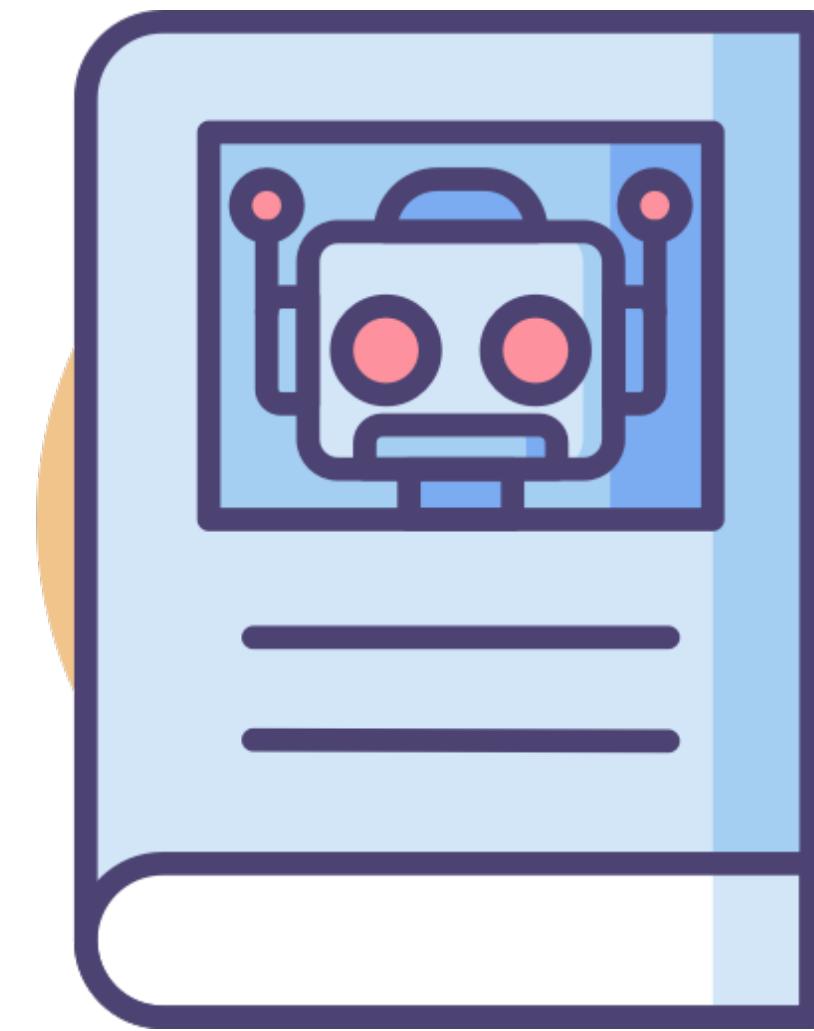
# We're focused on...

- Practical solutions
- Writing machine learning code

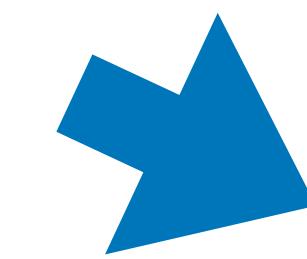
## Steps in a full machine learning project



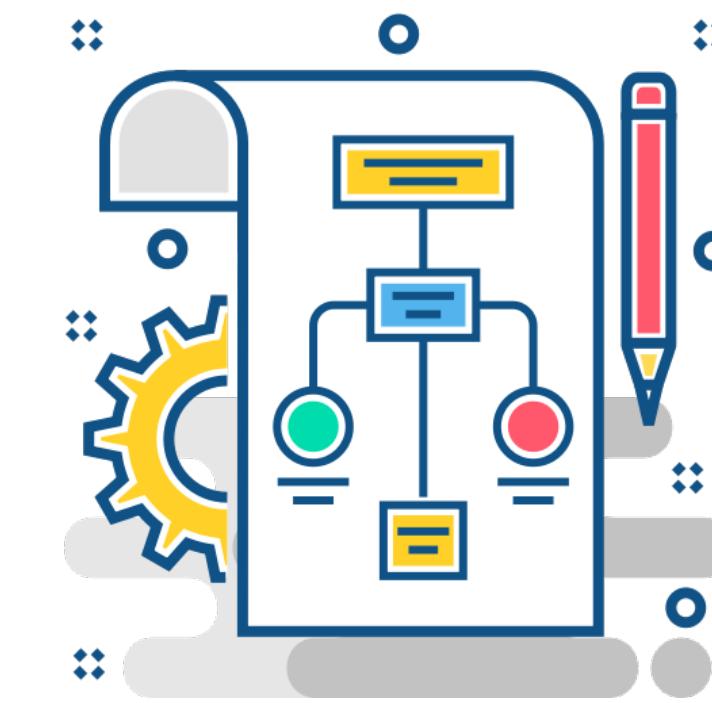
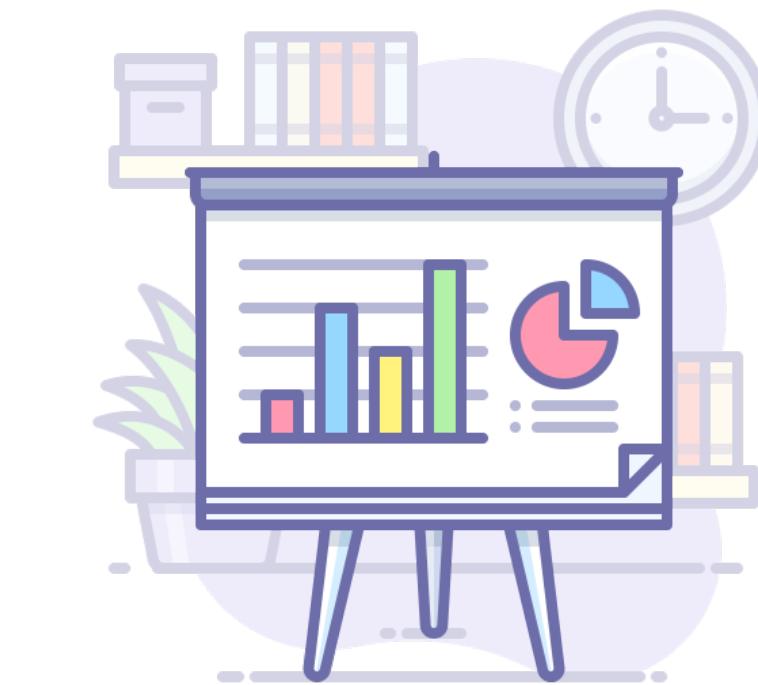
**1. Create a framework**



**2. Match to data science  
and machine learning  
tools**



**3. Learn by doing**



**Yes**

Write code

**No**

Overthink the process

Make mistakes

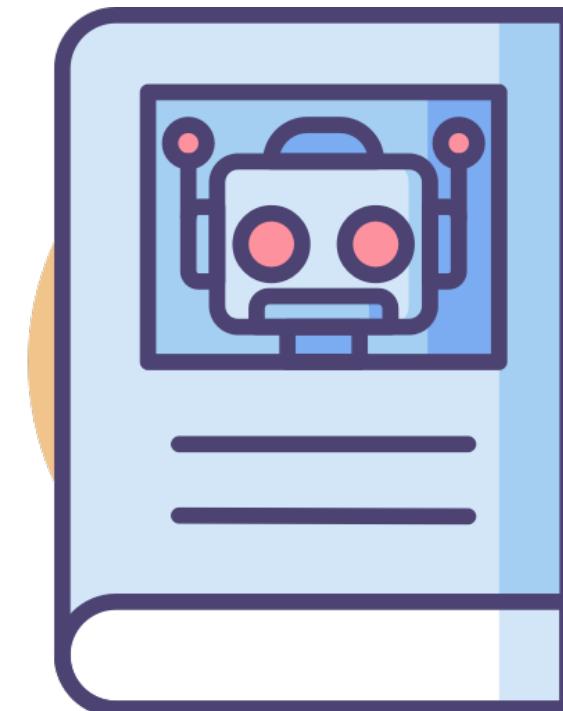
Try make things perfect

Build projects

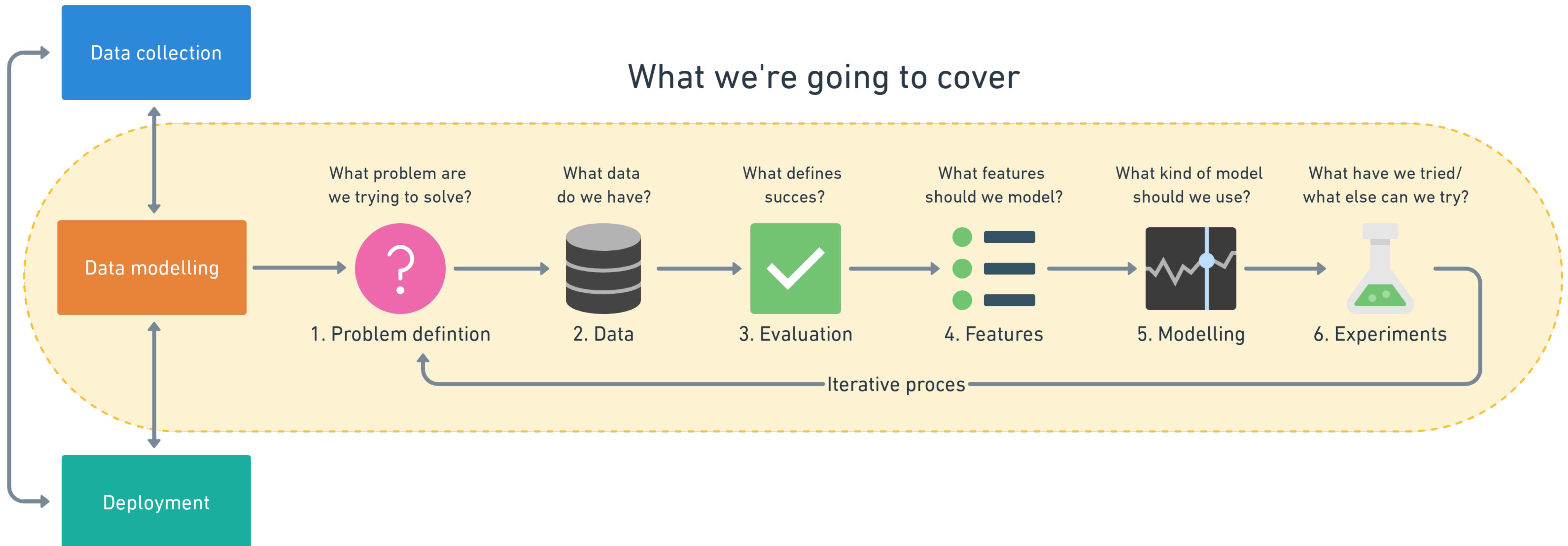
Build things from scratch

Learn what matters

# The framework we'll be using



## Steps in a full machine learning project

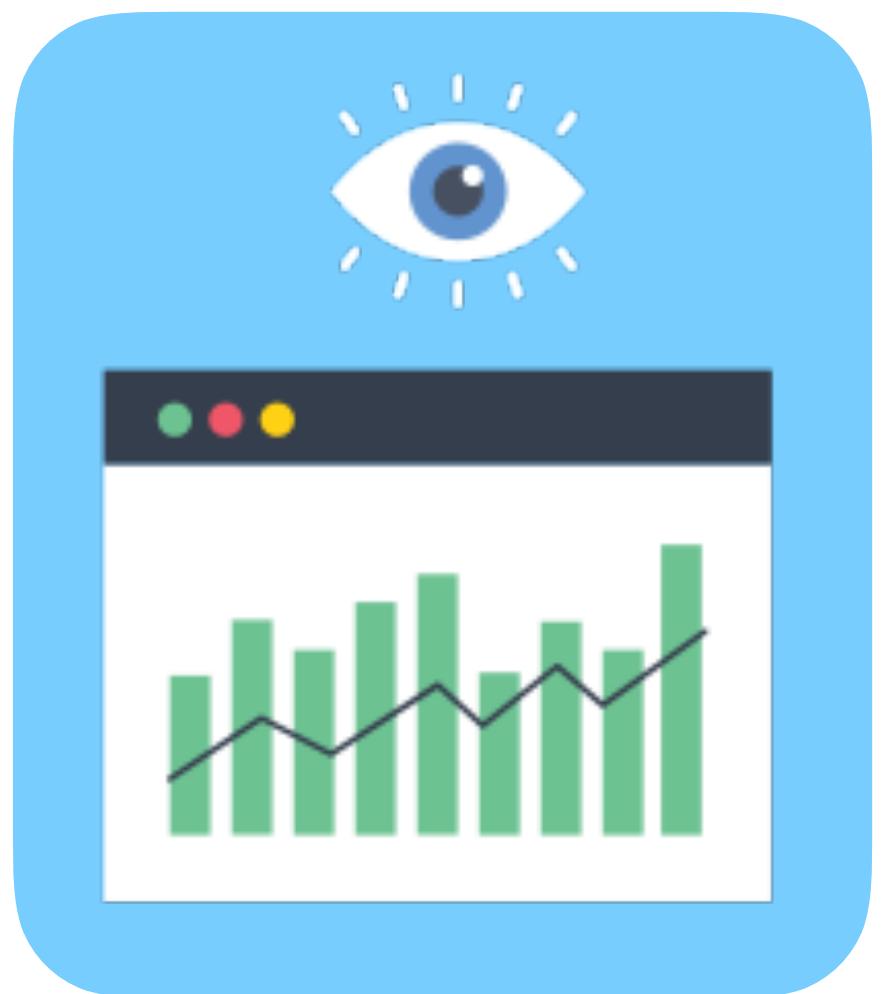


## What we're going to cover

# 1. Problem definition



**“What problem are we trying to solve?”**



**Supervised**



**Unsupervised**



**Classification**

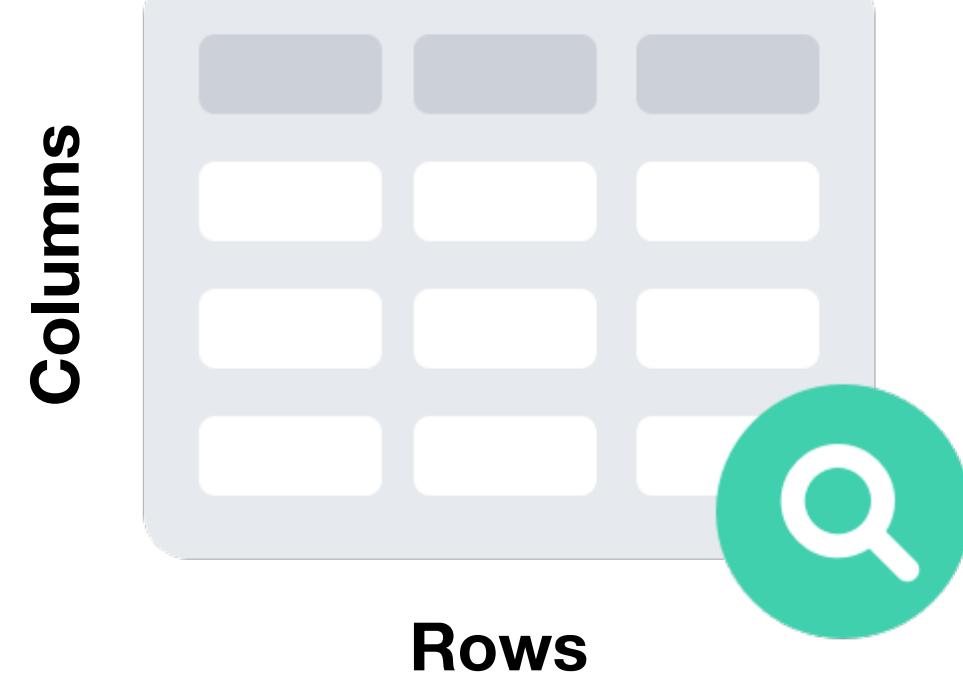


**Regression**

## 2. Data



“What kind of data do we have?”



Structured



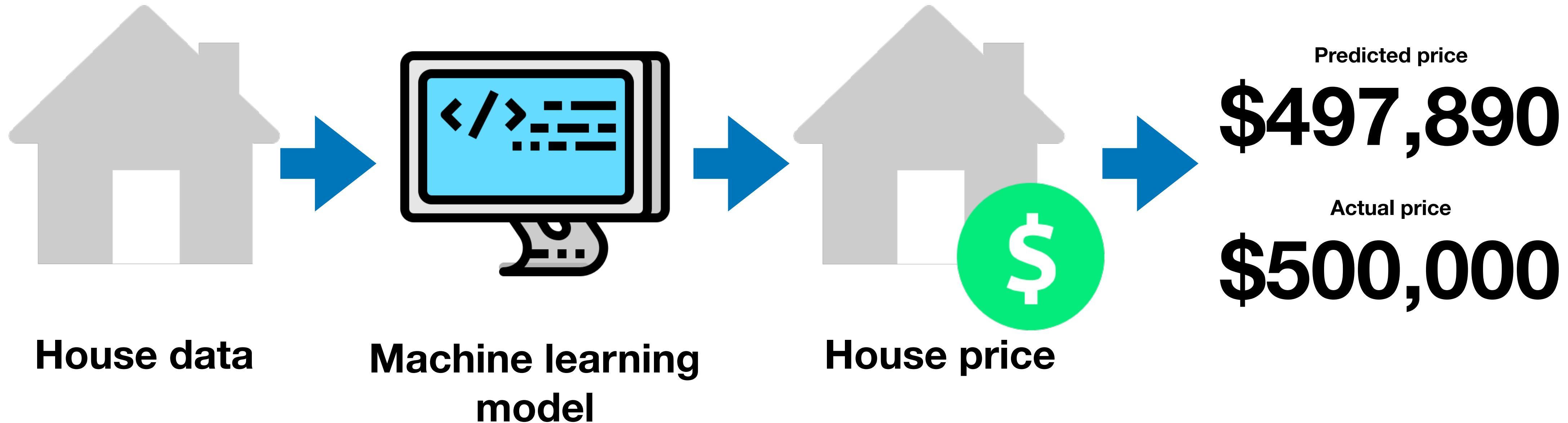
Unstructured



# 3. Evaluation



“What defines success for us?”



# 4. Features



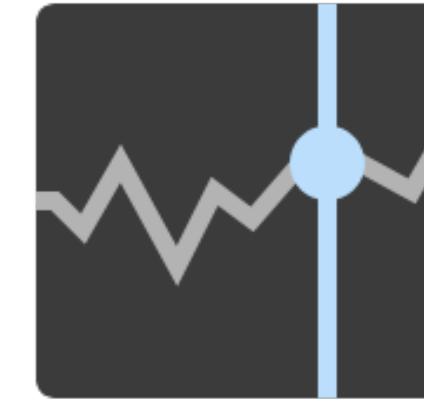
“What do we already know about the data?”



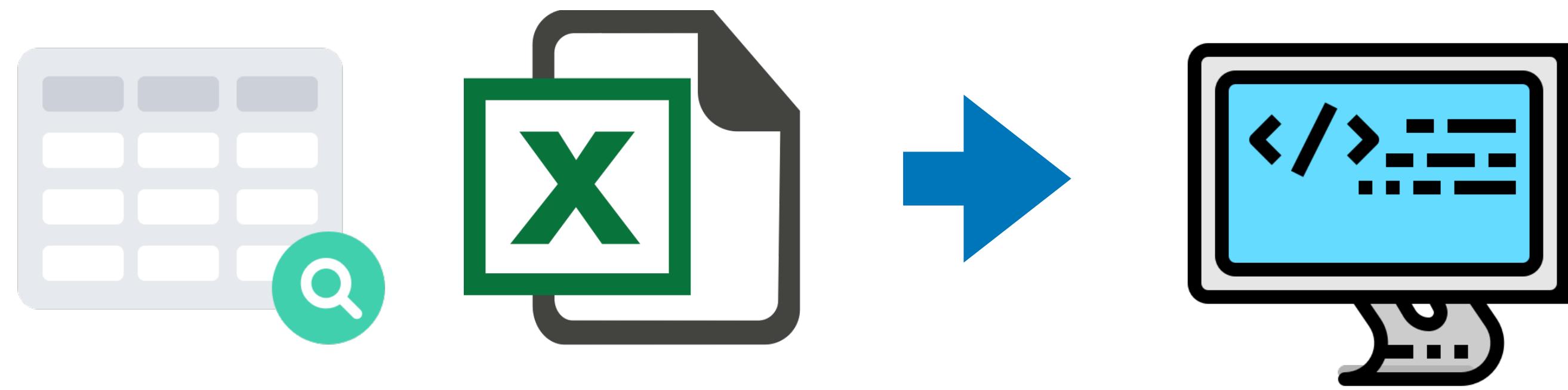
ID	Weight	Sex	Blood Pressure	Chest pain	Heart disease?
4326	110Kg	M	120 / 80	4	Yes
5681	64Kg	F	130 / 90	1	No
7911	81Kg	M	130 / 80	0	No

Table 1.0 : Patient records

# 5. Modelling



**“Based on our problem and data, what model should we use?”**



**Problem 1**

**Model 1**



**Problem 2**

**Model 2**

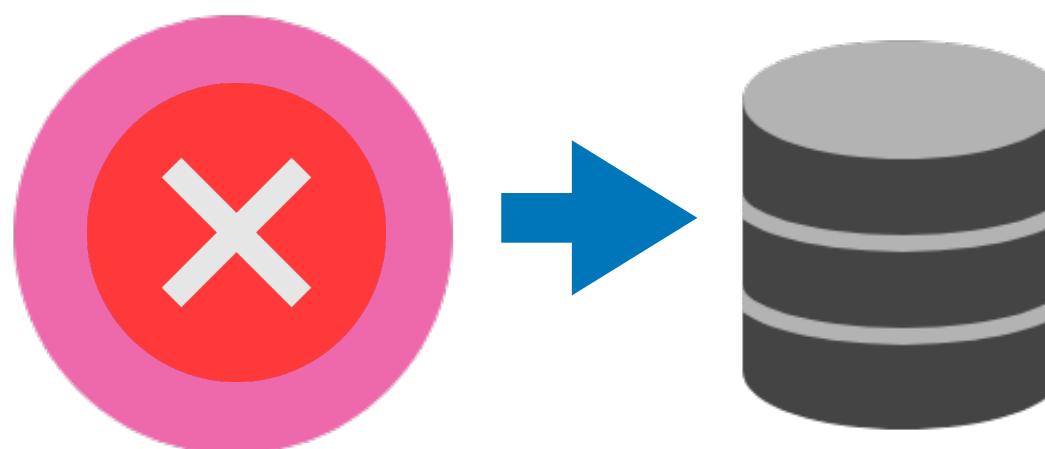
# 6. Experimentation



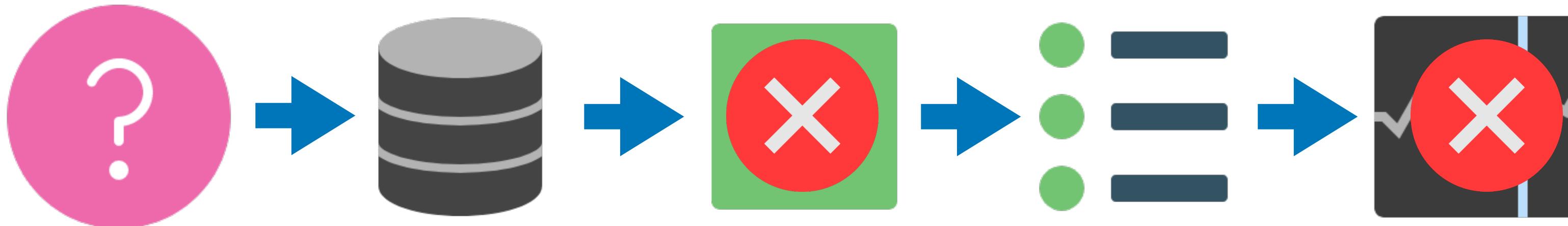
**“How could we improve/what can we try next?”**

**Attempt**

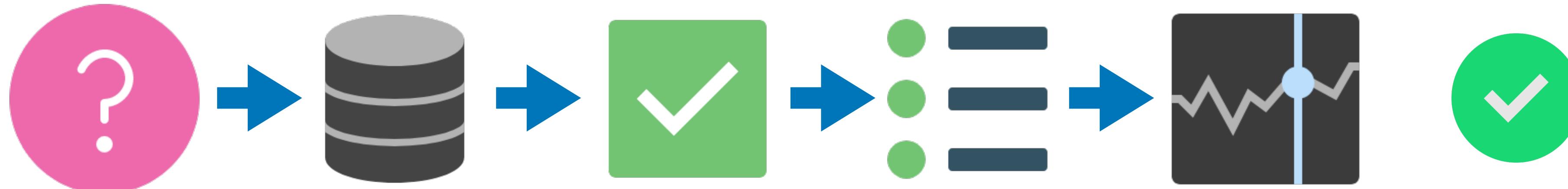
**1**



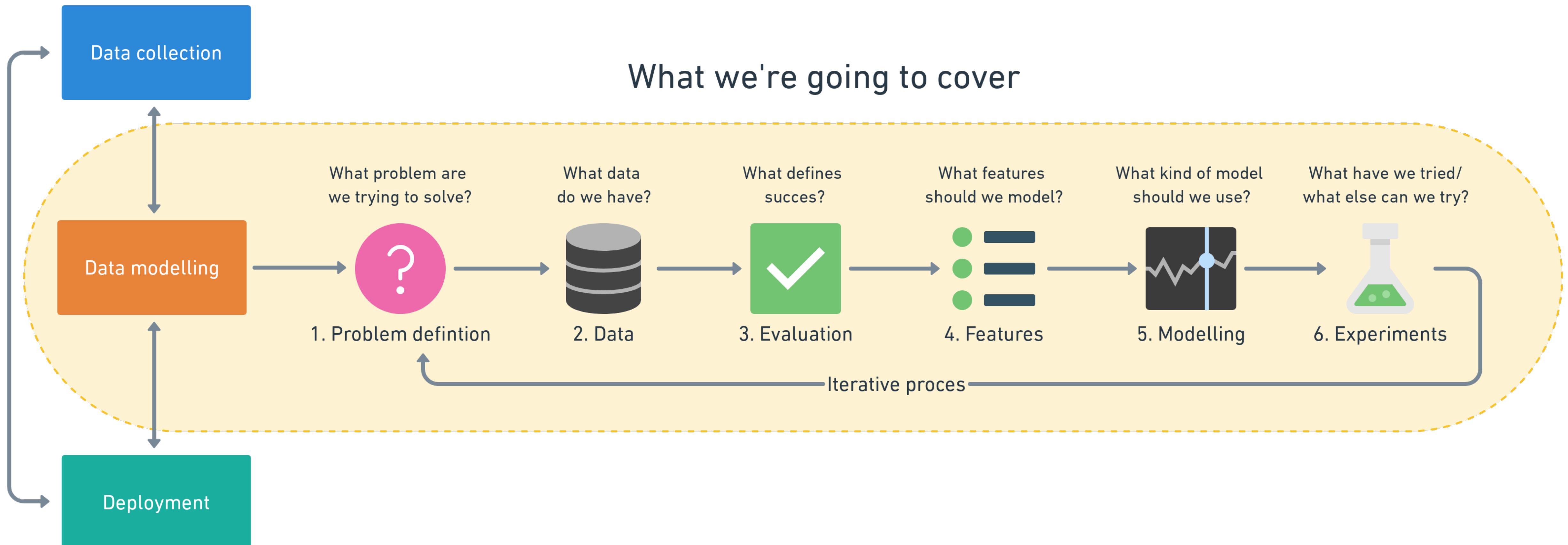
**2**



**3**



## Steps in a full machine learning project



## What we're going to cover

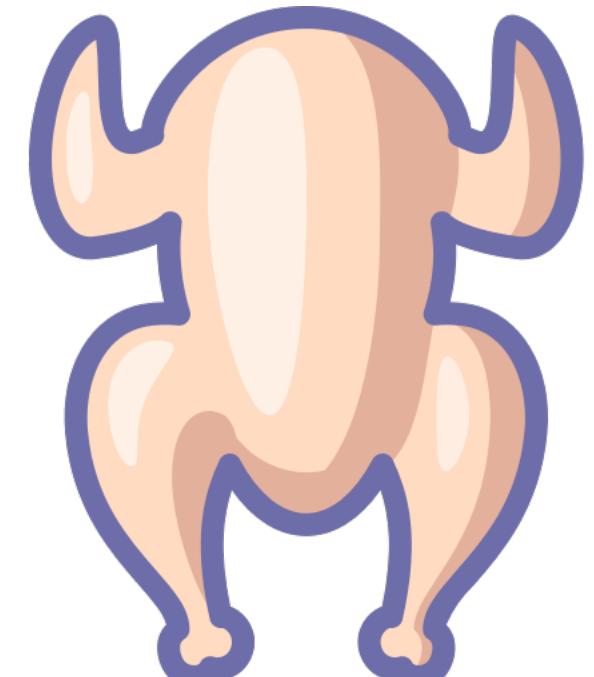
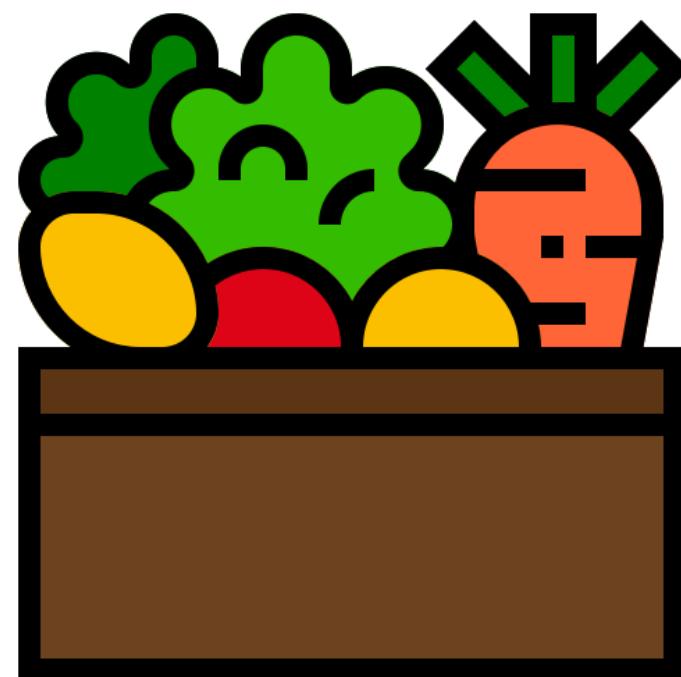
# 1. Problem definition



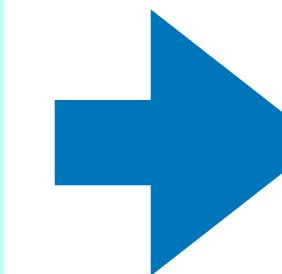
**“What problem are we trying to solve?”**

# When shouldn't you use machine learning?

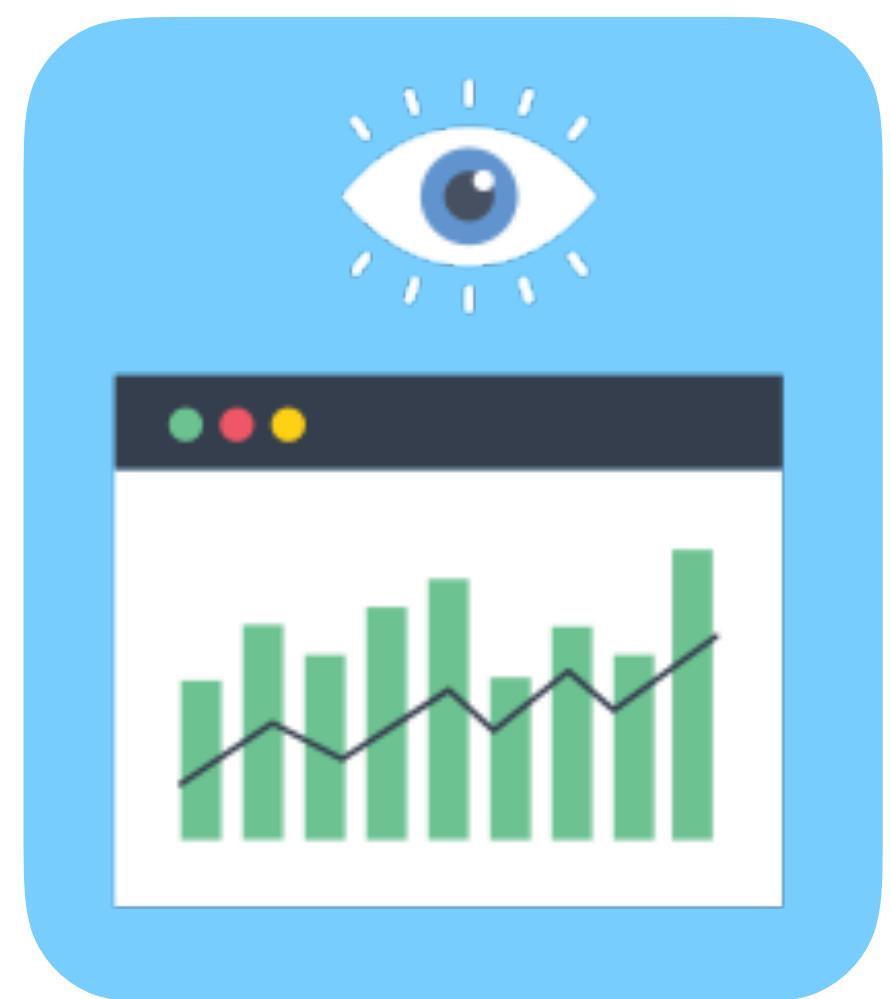
- Will a simple hand-coded instruction based system work?



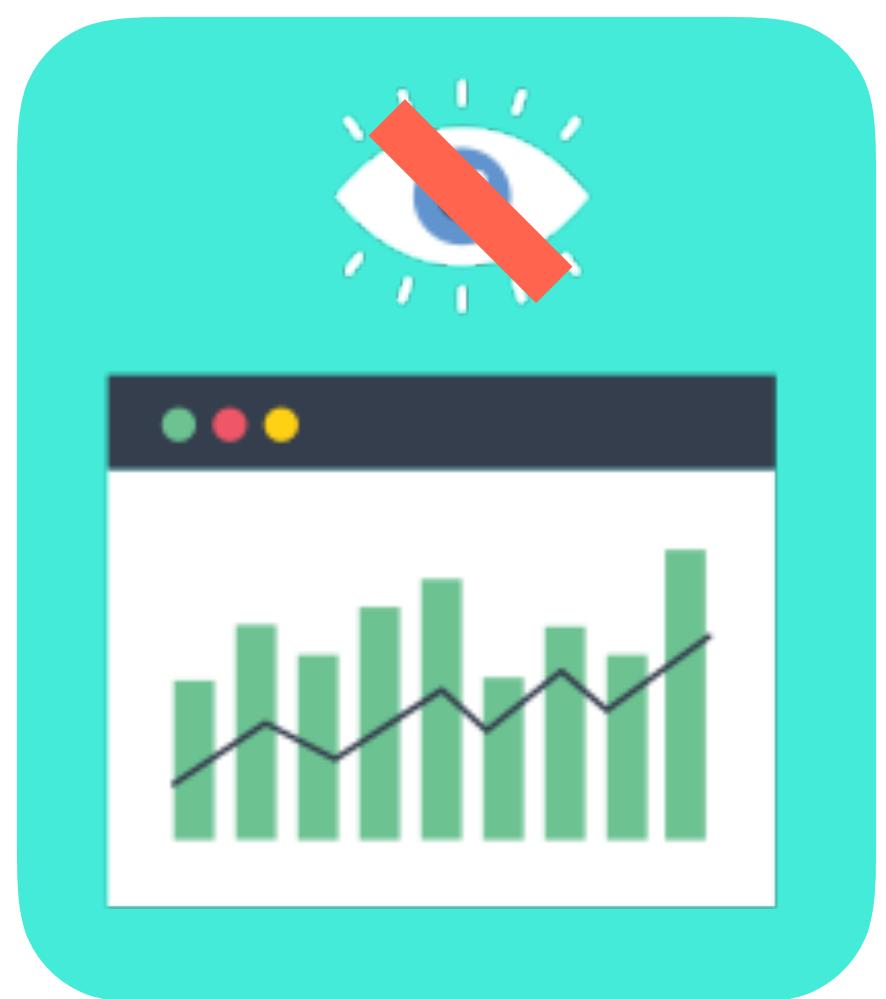
1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



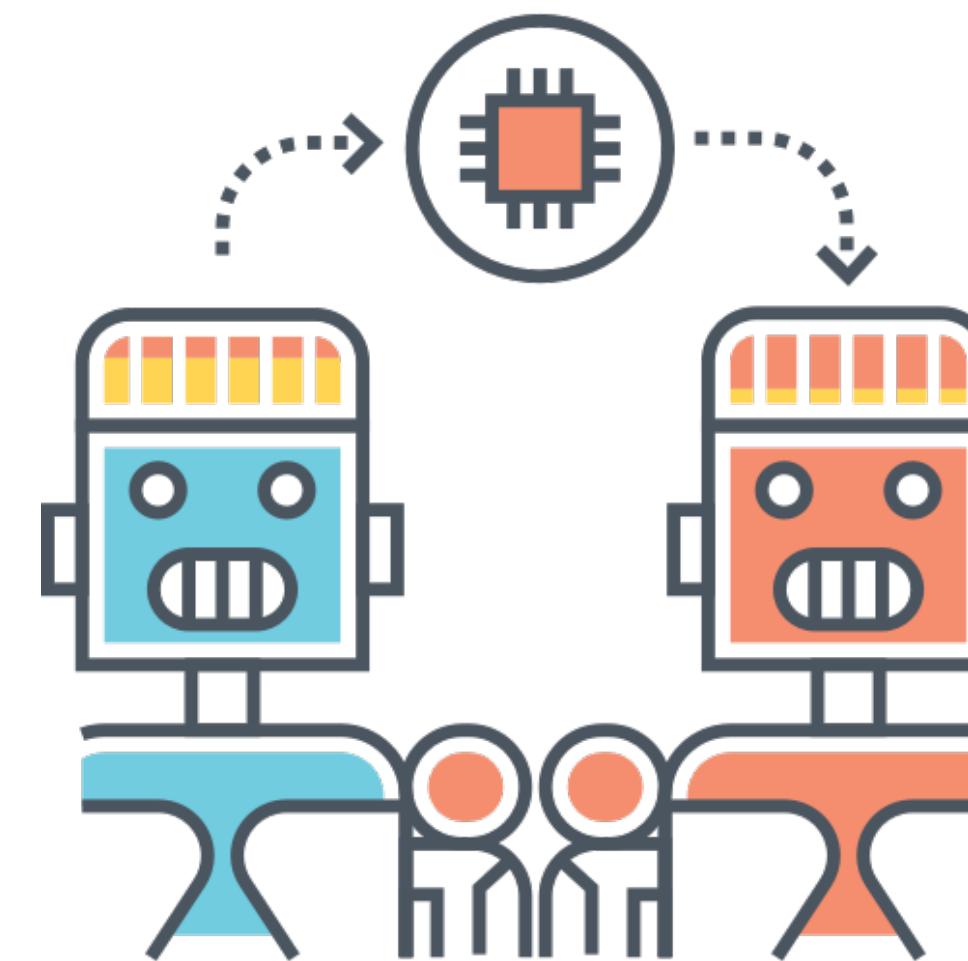
# Main types of machine learning



**Supervised  
Learning**



**Unsupervised  
Learning**

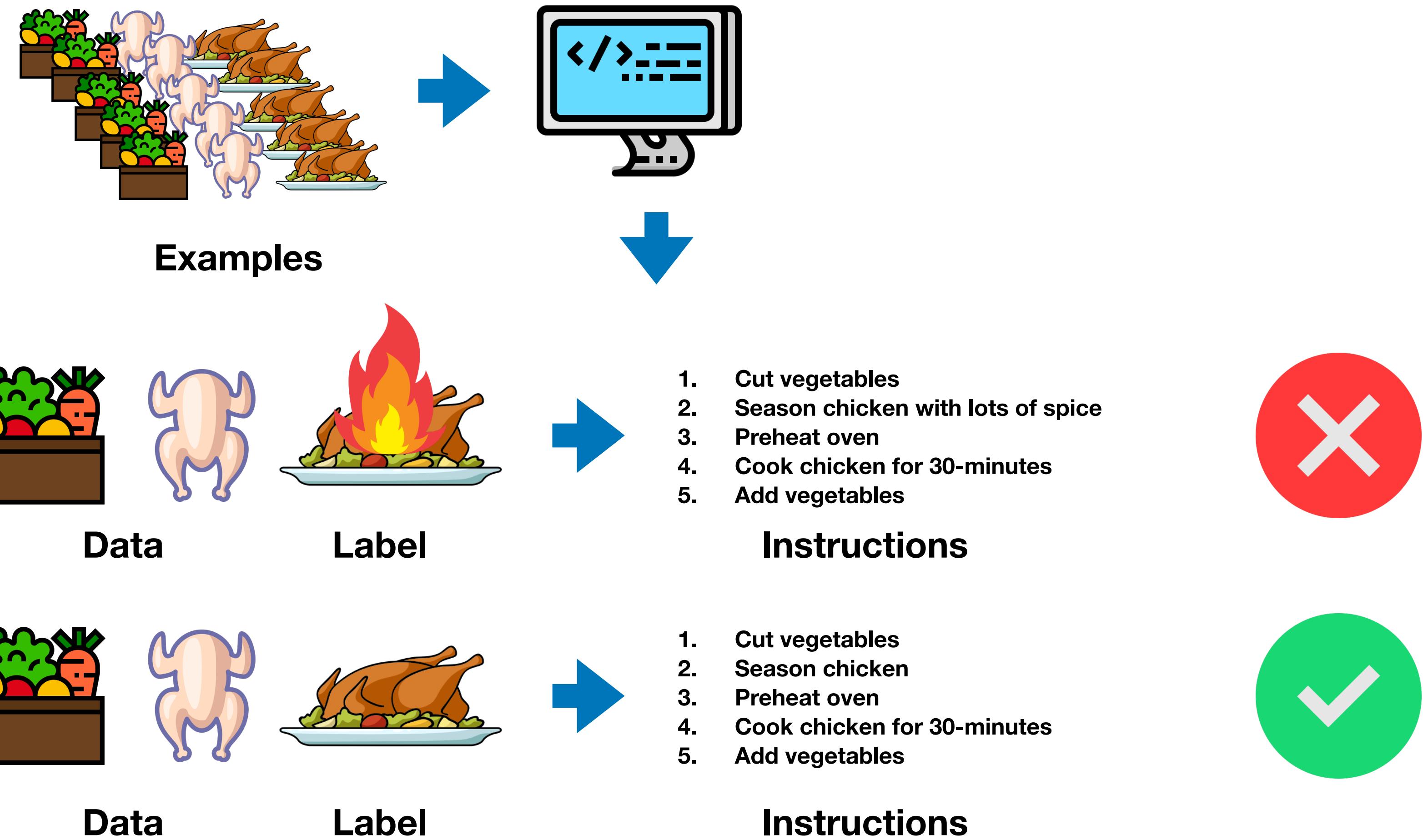
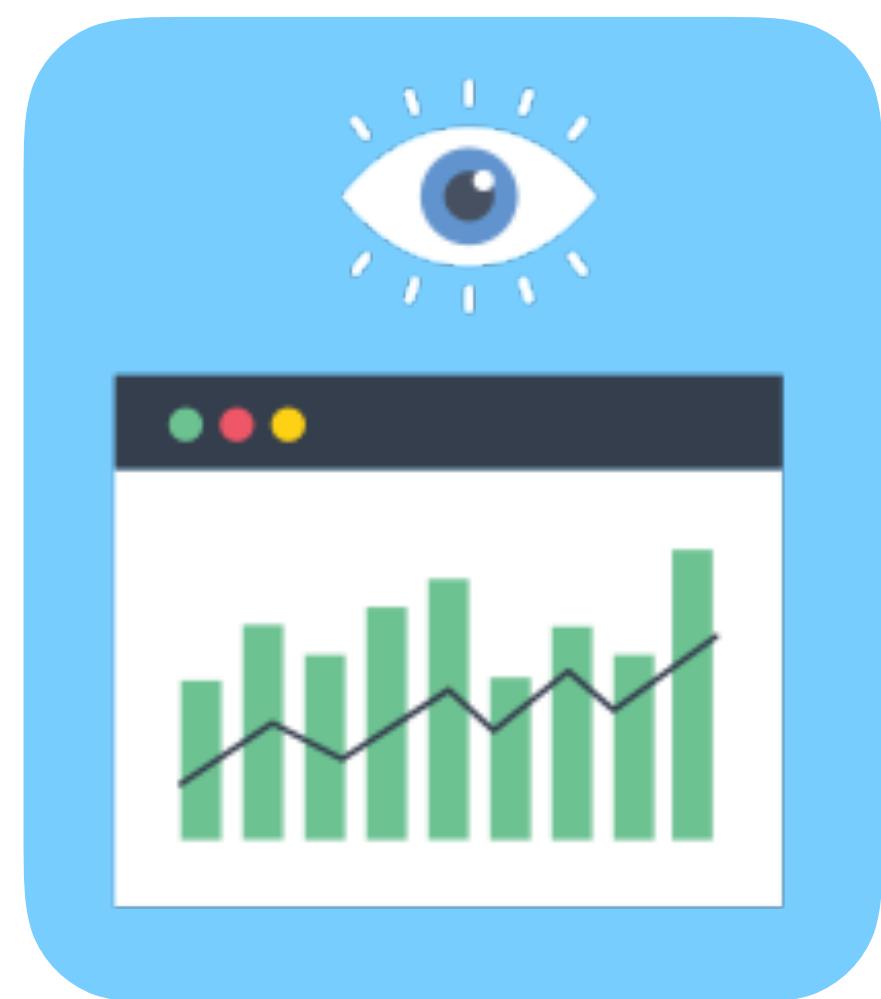


**Transfer  
Learning**



**Reinforcement  
Learning**

# Supervised learning



# Supervised learning



## Classification

- “Is this example one thing or another?”
- Binary classification = two options
- Multi-class classification = more than two options

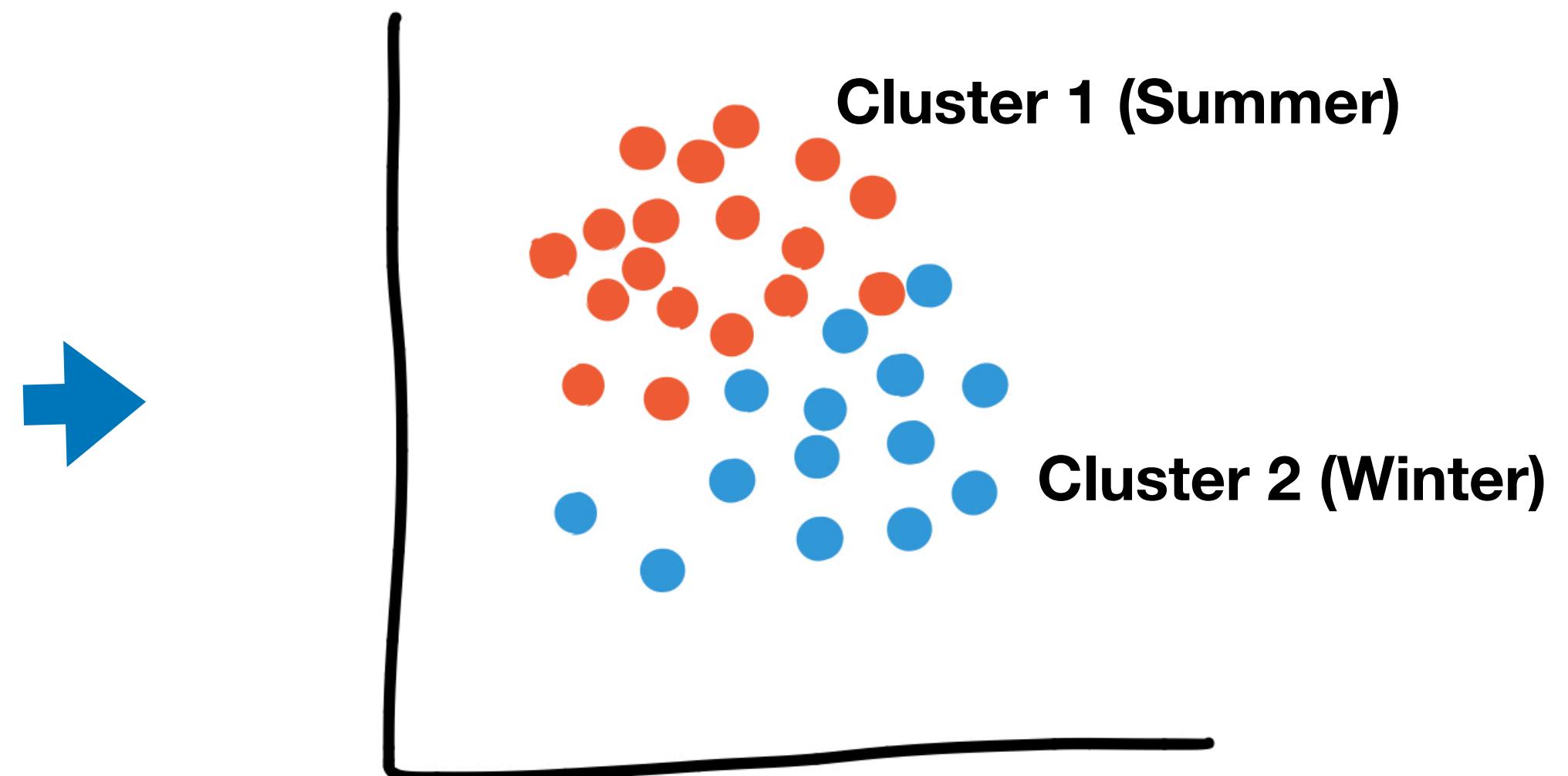


## Regression

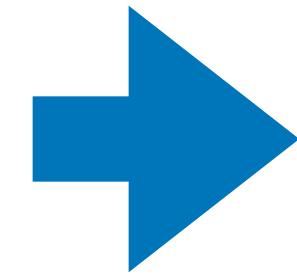
- “How much will this house sell for?”
- “How many people will buy this app?”

# Unsupervised learning

Customer ID	Purchase 1	Purchase 2
1	Sunglasses	Singlet
2	Jacket	Snow boots
3	Sunscreen	Beach towel

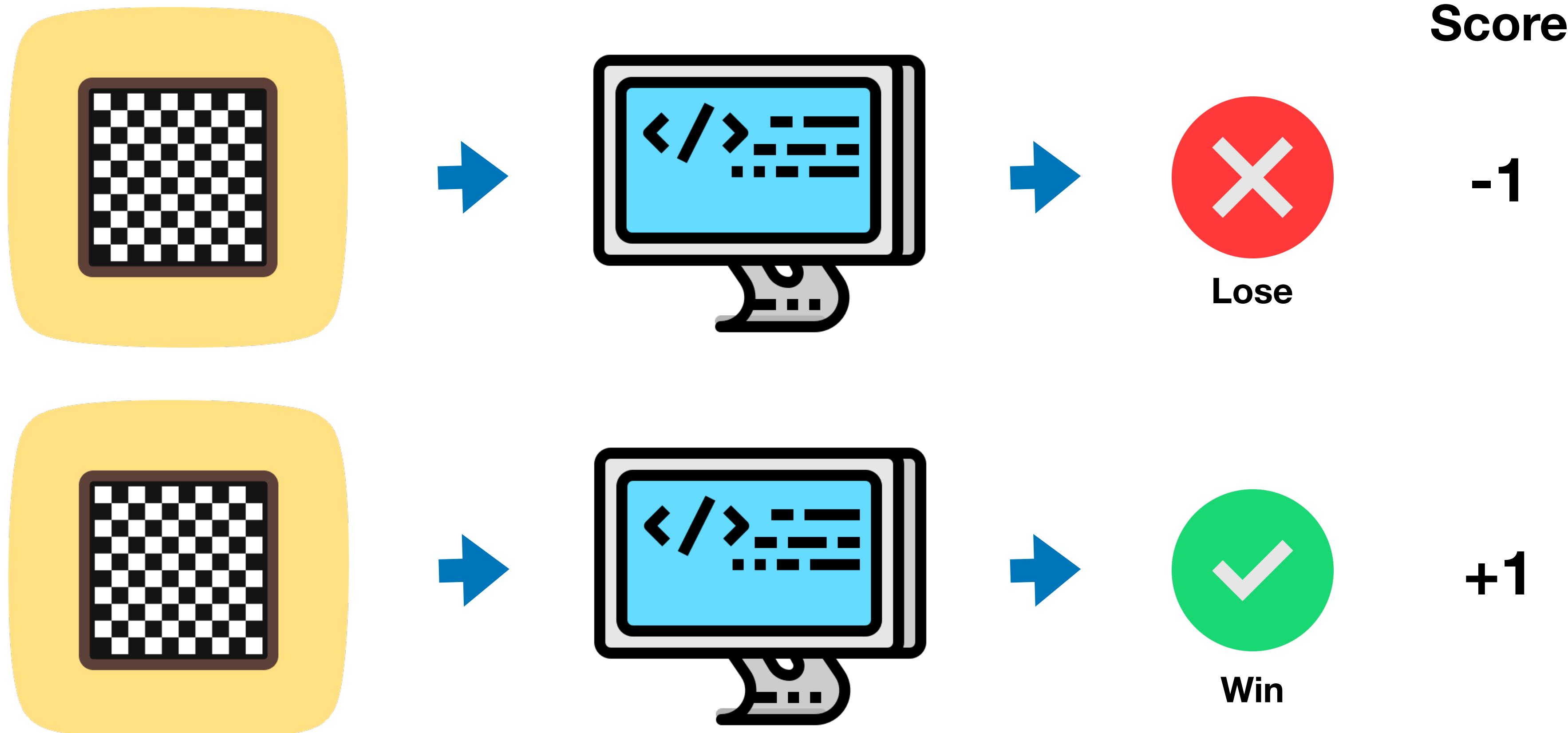


# Transfer learning

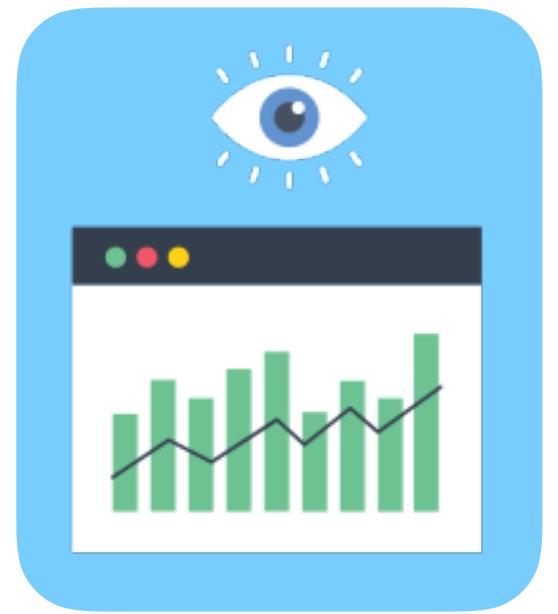


**Use what a model has learned in one domain, to help in another domain.**

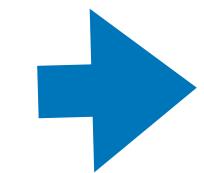
# Reinforcement learning



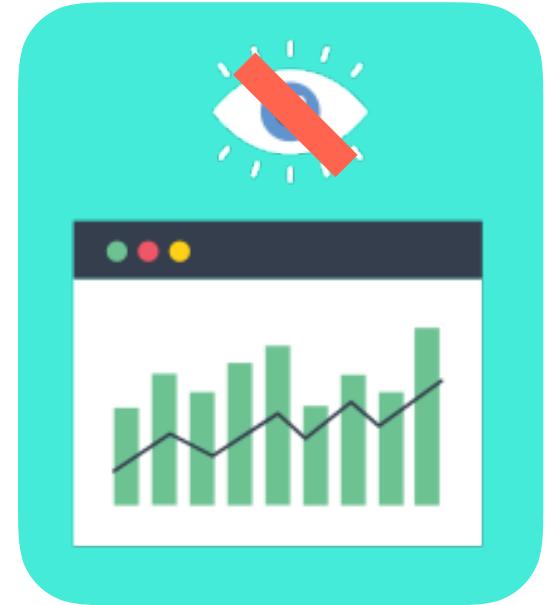
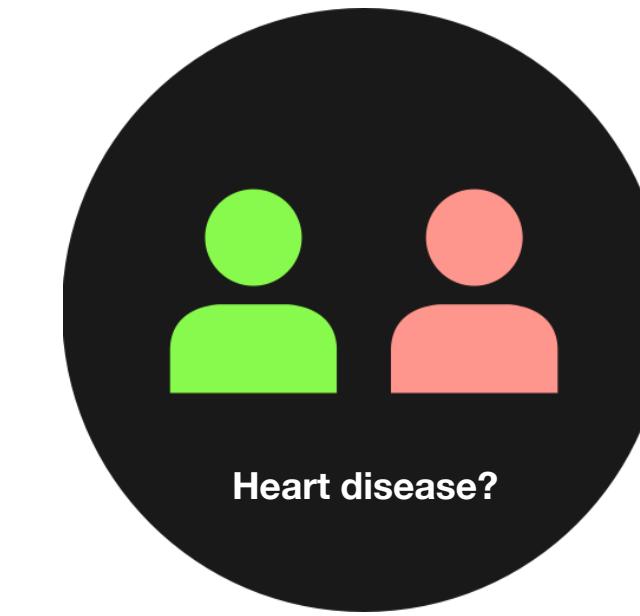
# Matching your problem



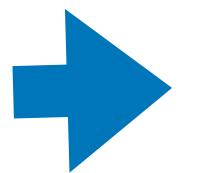
Supervised  
Learning



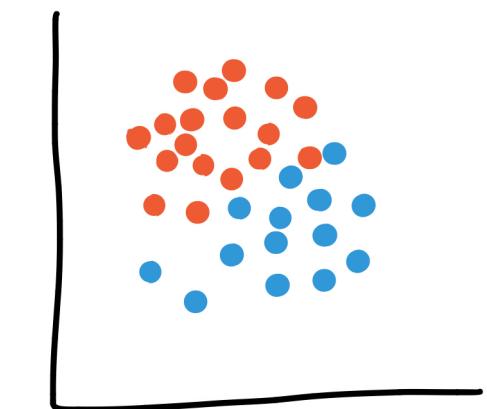
**“I know my inputs and outputs.”**



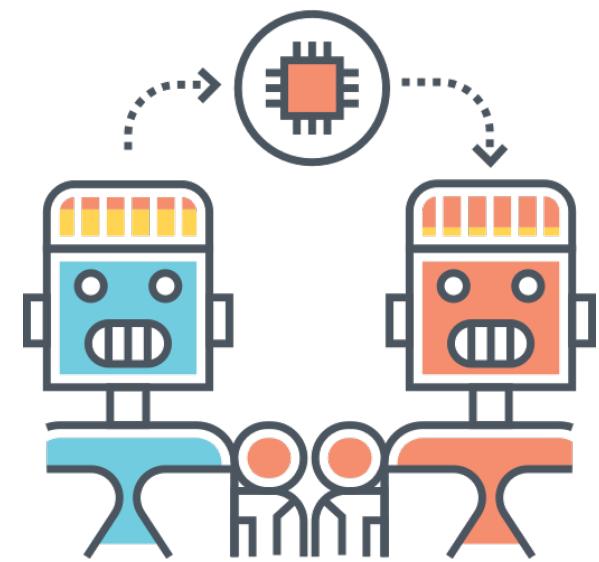
Unsupervised  
Learning



**“I’m not sure of the outputs but I have inputs.”**



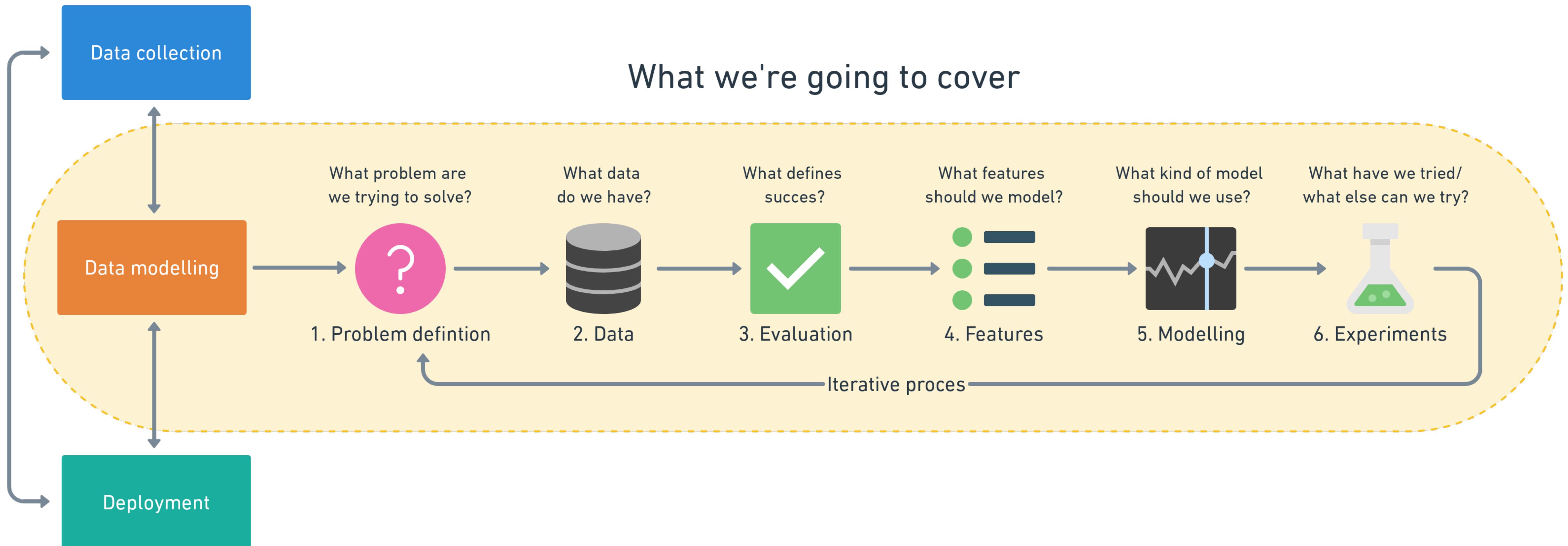
# Matching your problem



→ “I think my problem may be similar to something else.”

**Transfer  
Learning**

## Steps in a full machine learning project



## What we're going to cover

## **2.Data**



**“What kind of data do we have?”**

# Different types of data

Rows

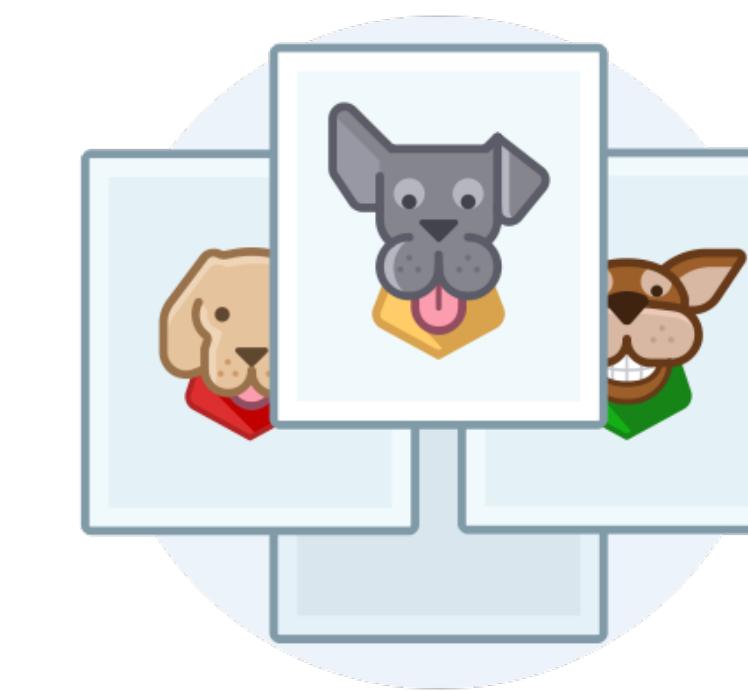
Columns

ID	Weight	Sex	Blood Pressure	Chest pain	Heart disease?
4328	110Kg	M	120 / 80	4	YES
5681	64Kg	F	130 / 90	1	NO
7911	81Kg	M	130 / 80	0	NO

Table 1.0: Patient records



Structured

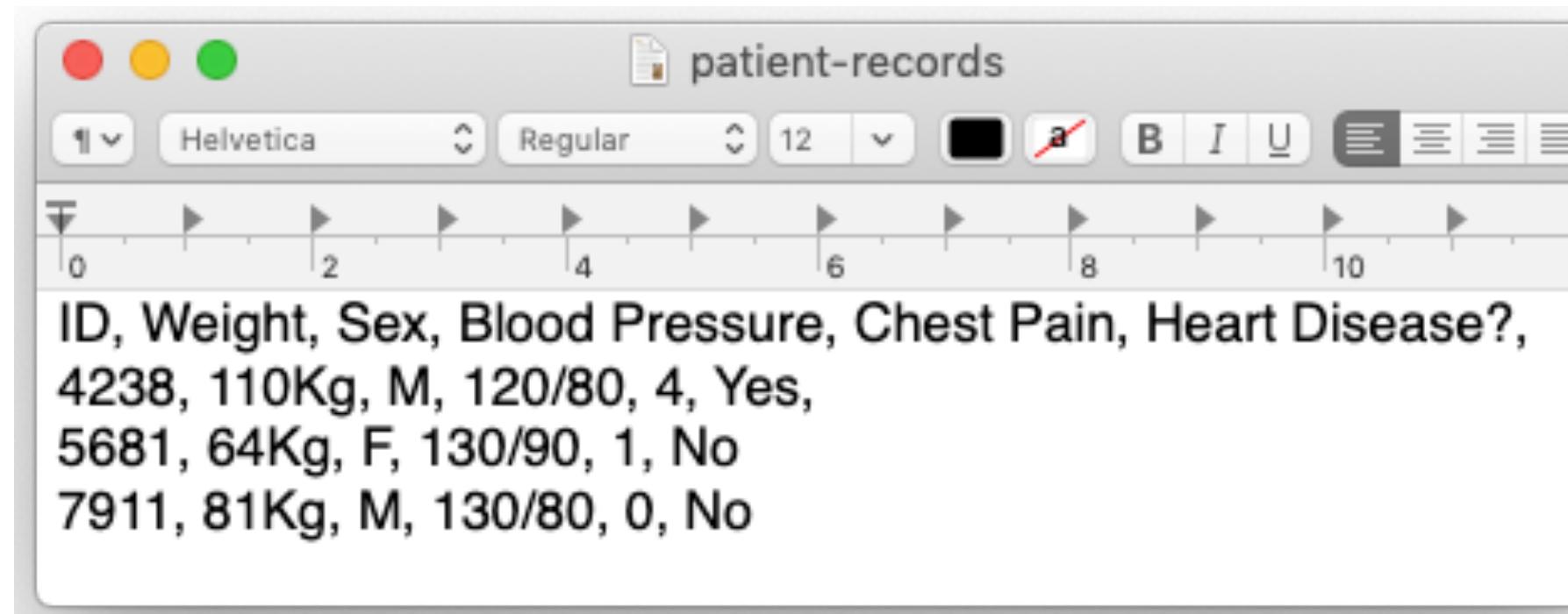


From: [daniel@mrbourke.com](mailto:daniel@mrbourke.com)  
Hey Daniel,

First of all, thank you for being so amazing.  
This machine learning course is incredible.  
Thank you for keeping it simple!

Unstructured

# Different types of data



A screenshot of a Mac OS X text editor window titled "patient-records". The content is a CSV file with the following data:

```
ID, Weight, Sex, Blood Pressure, Chest Pain, Heart Disease?,  
4238, 110Kg, M, 120/80, 4, Yes,  
5681, 64Kg, F, 130/90, 1, No  
7911, 81Kg, M, 130/80, 0, No
```



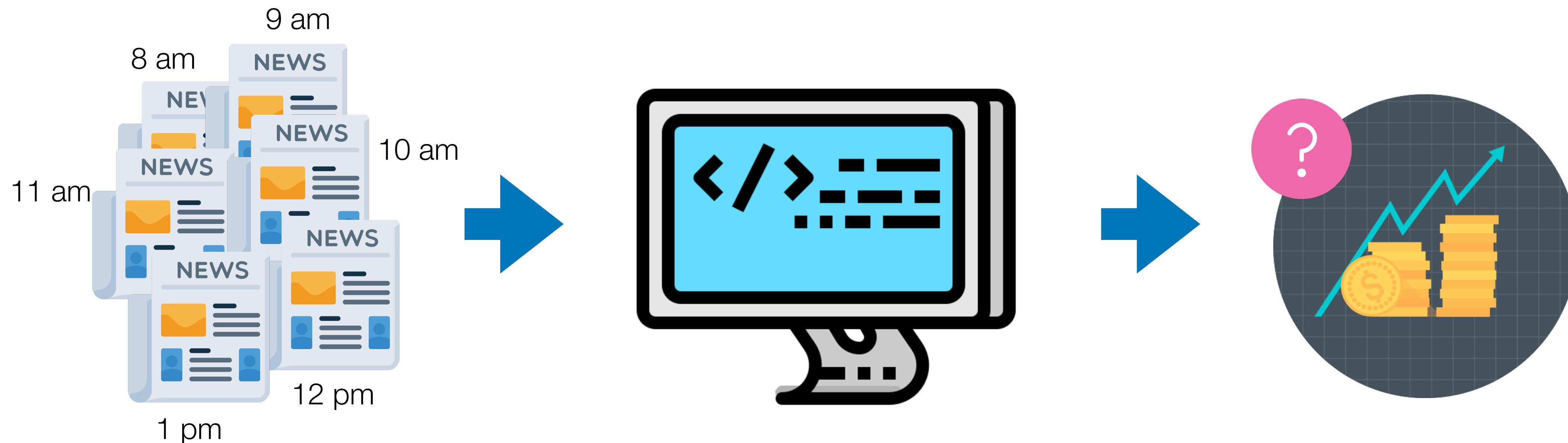
ID	Weight	Sex	Blood Pressure	Chest pain	Heart disease?
4326	110Kg	M	120 / 80	4	Yes
5681	64Kg	F	130 / 90	1	No
7911	81Kg	M	130 / 80	0	No

Table 1.0: Patient records

Static

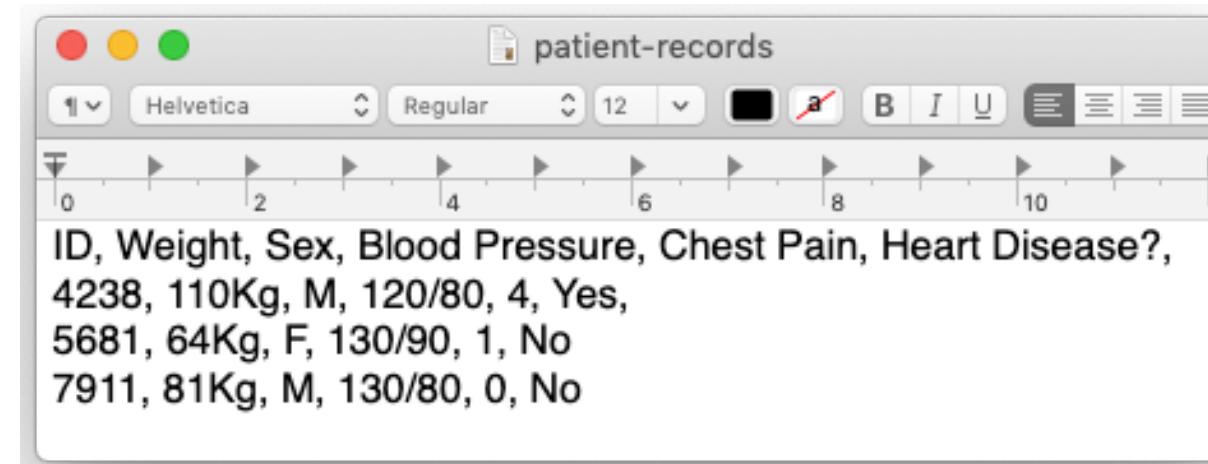
“The more data the better.”

# Different types of data



**Streaming**

# A data science workflow



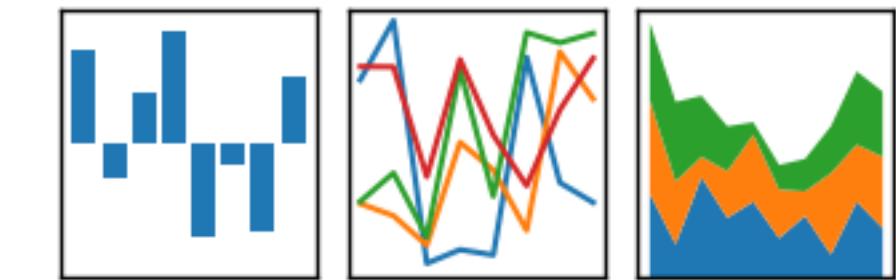
ID	Weight	Sex	Blood Pressure	Chest pain	Heart disease?
4326	110kg	M	120/80	4	Yes
5681	64kg	F	130/90	1	No
7911	81kg	M	130/80	0	No

Table 1.0: Patient records

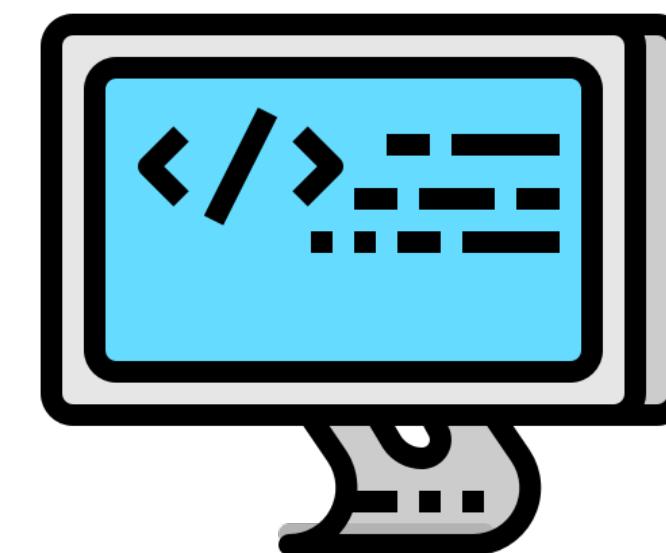
Static data

pandas

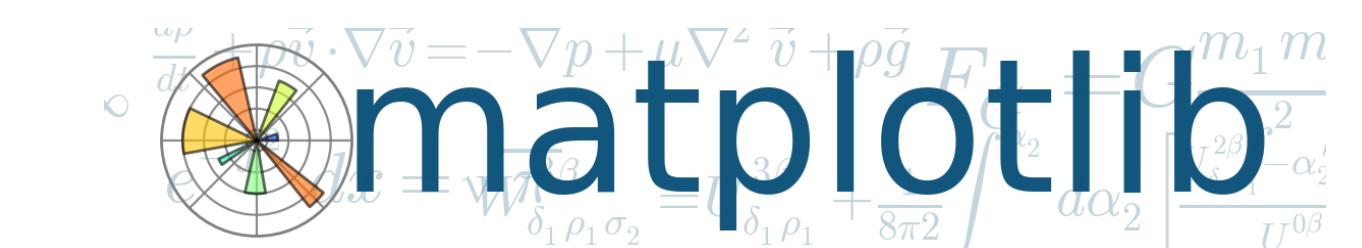
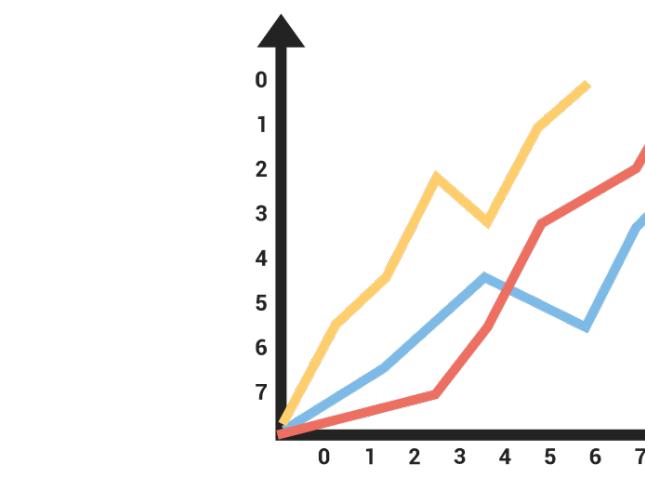
$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Data Analysis

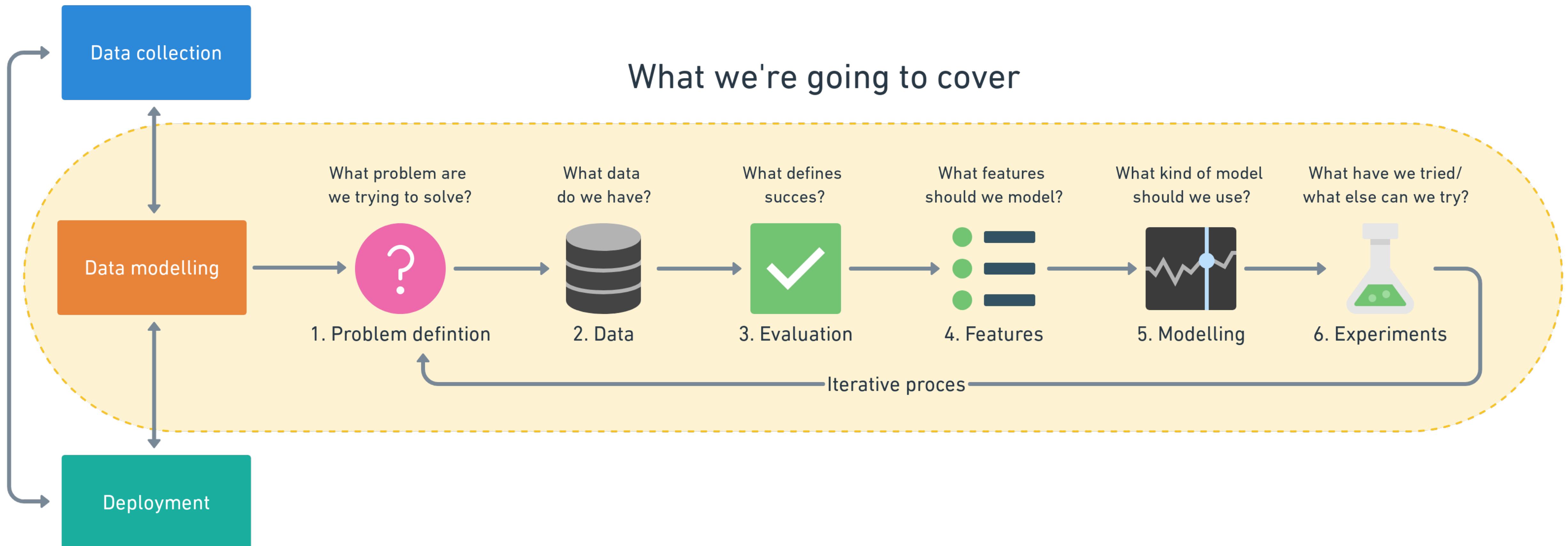


Machine learning model



**What kind of data do you use?**

## Steps in a full machine learning project



## What we're going to cover

### **3. Evaluation**

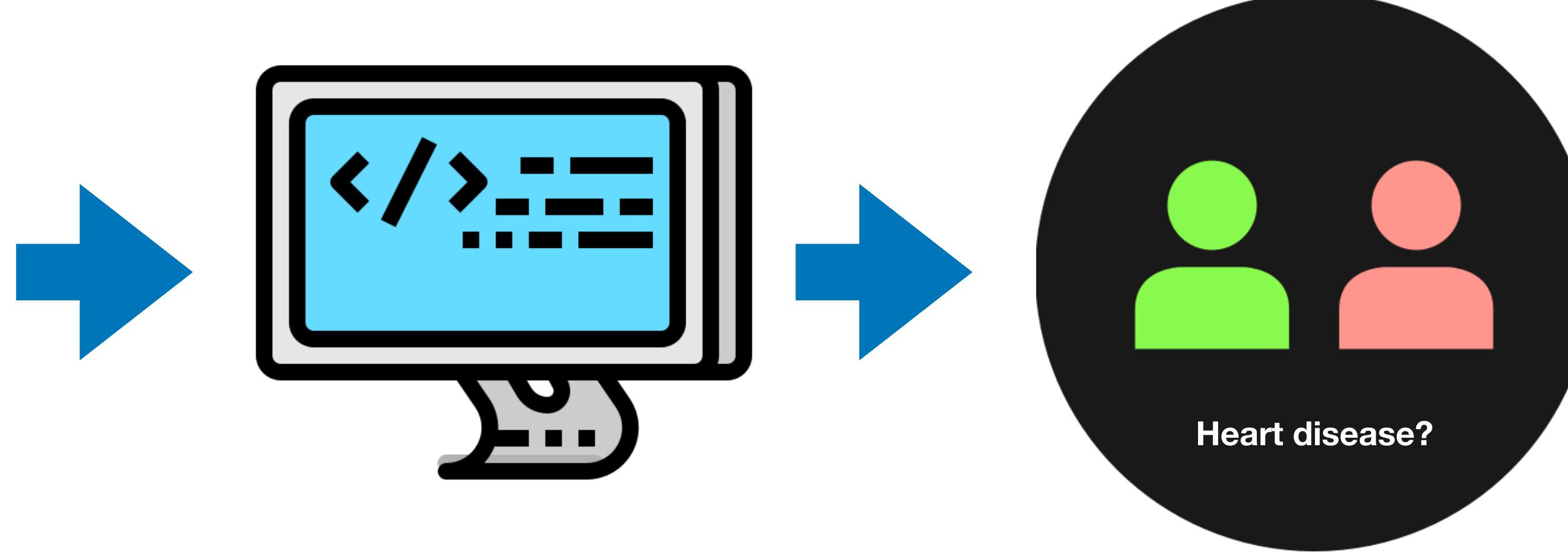


**“What defines success for us?”**

**“For this project to be worth pursuing further,  
we need a machine learning model with over 99% accuracy.”**

ID	Weight	Sex	Blood Pressure	Chest pain	Heart disease?
4326	110Kg	M	120 / 80	4	YES
5681	64Kg	F	130 / 90	1	NO
7911	81Kg	M	130 / 80	0	NO

Table 1.0: Patient records



**Machine learning  
model**

Accuracy  
**97.8%**

# Different types of metrics

## Classification

Accuracy

Precision

Recall

## Regression

Mean absolute error (MAE)

Mean squared error (MSE)

Root mean squared error  
(RMSE)

## Recommendation

Precision at K

# Classifying car insurance claims

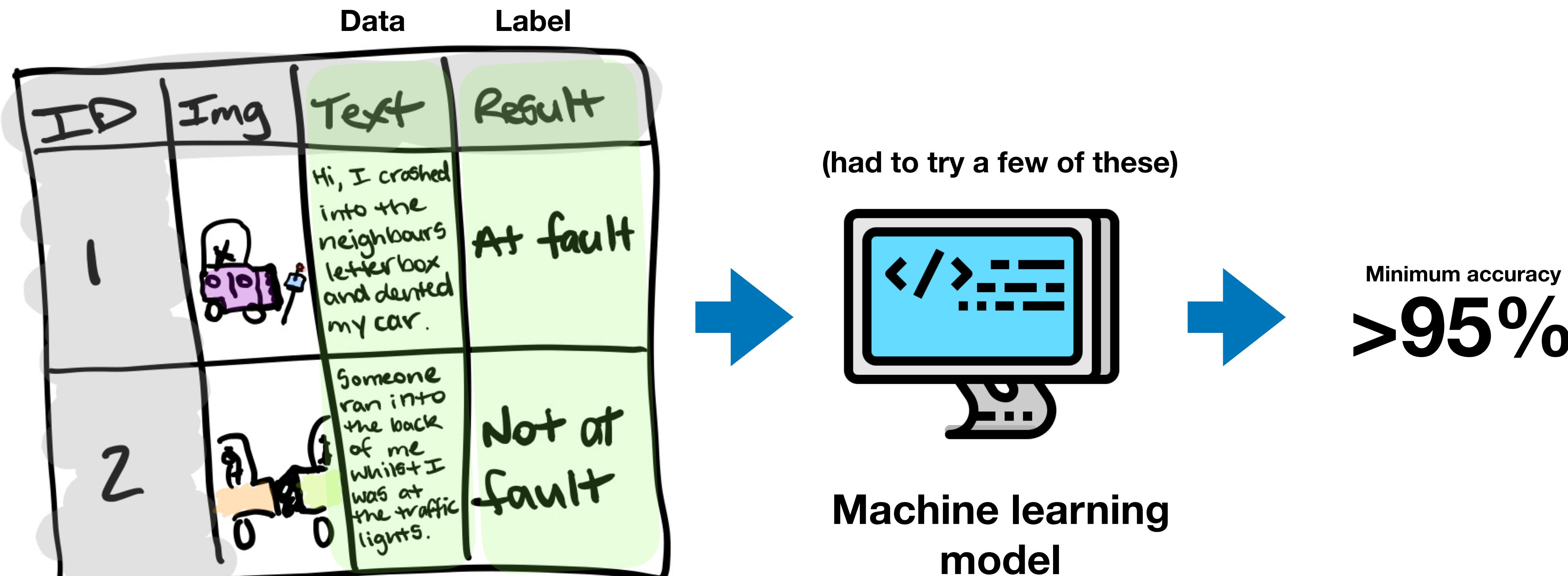
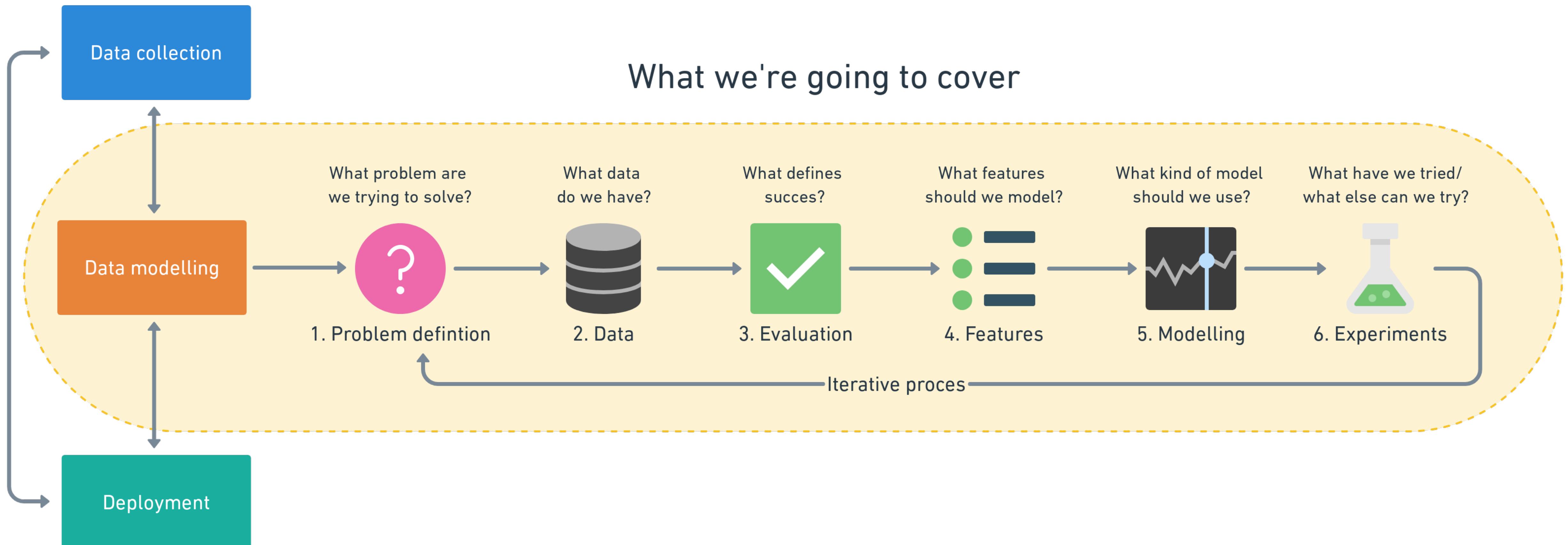


Table 2.0: Car insurance claims

**What do you measure?**

## Steps in a full machine learning project



## What we're going to cover

## 4. Features



**“What do we already know about the data?”**

# Different features of data

ID	Feature variables				Target variable
	Weight	Sex	Heart Rate	Chest pain	
4326	110Kg	M	81	4	yes
5681	64Kg	F	61	1	no
7911	81Kg	m	57	0	no

Table 1.0: Patient records

# Different features of data

ID	weight	Sex	Heart Rate	Chest pain	Heart disease?
4326	110Kg	M	81	4	Yes
5681	64Kg	F	61	1	No
7911	81Kg	m	57	0	No

Numerical features

Categorical features

Table 1.0: Patient records

# Different features of data

ID	Weight	Sex	Heart Rate	Chest pain	Heart disease?	Derived feature
4326	110Kg	M	81	4	Yes	visit in last year?
5681	64Kg	F	61	1	No	Yes
7911	81Kg	M	57	0	No	No

Table 1.0 : Patient records

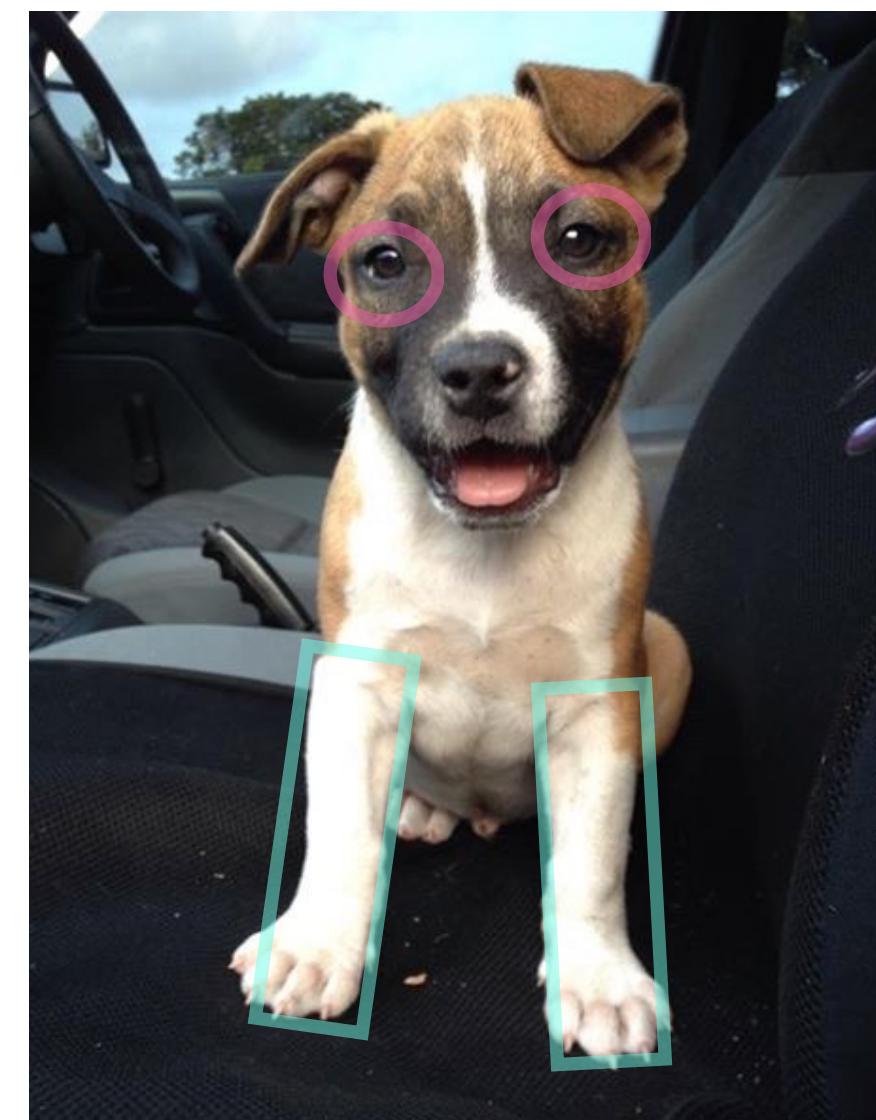
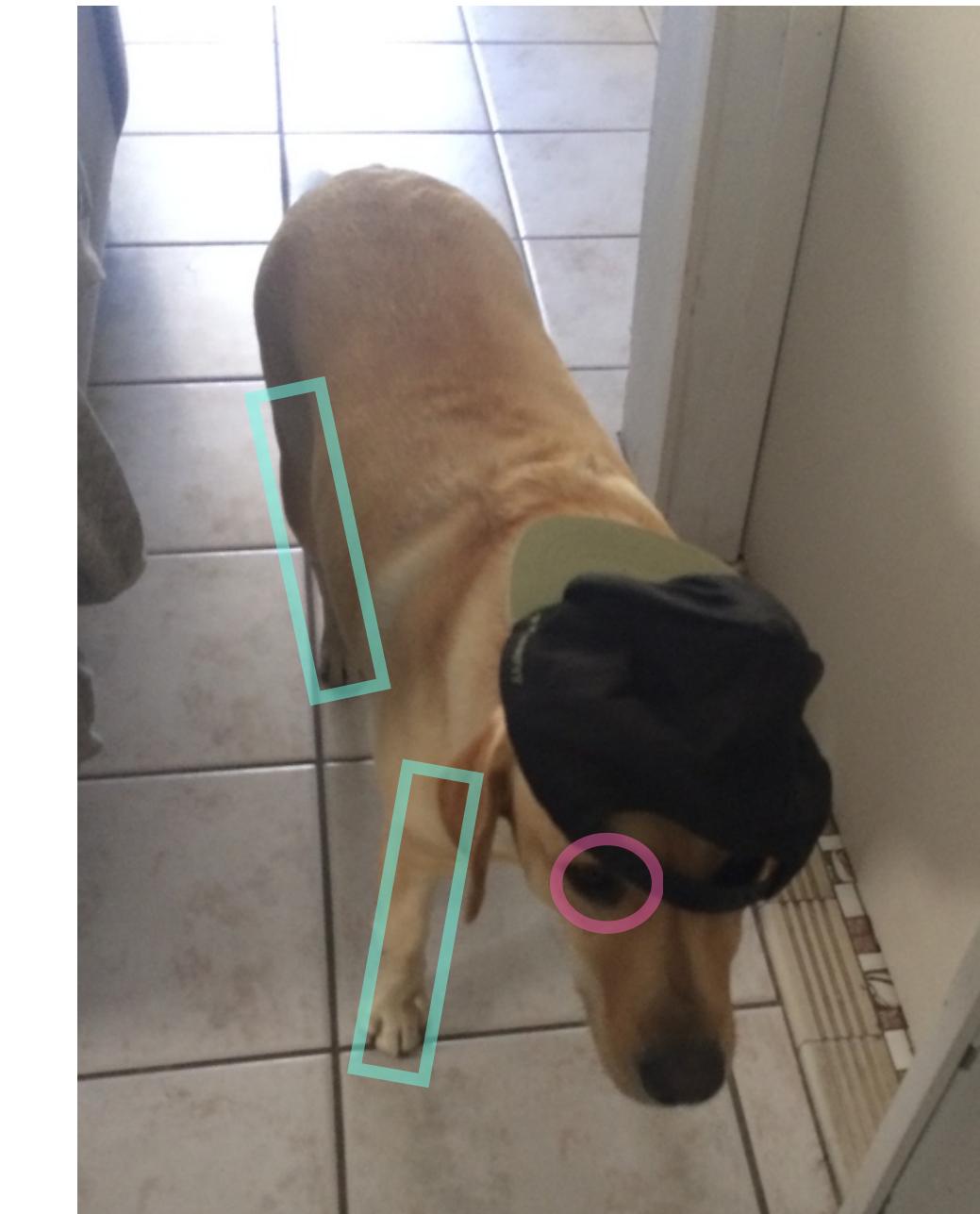
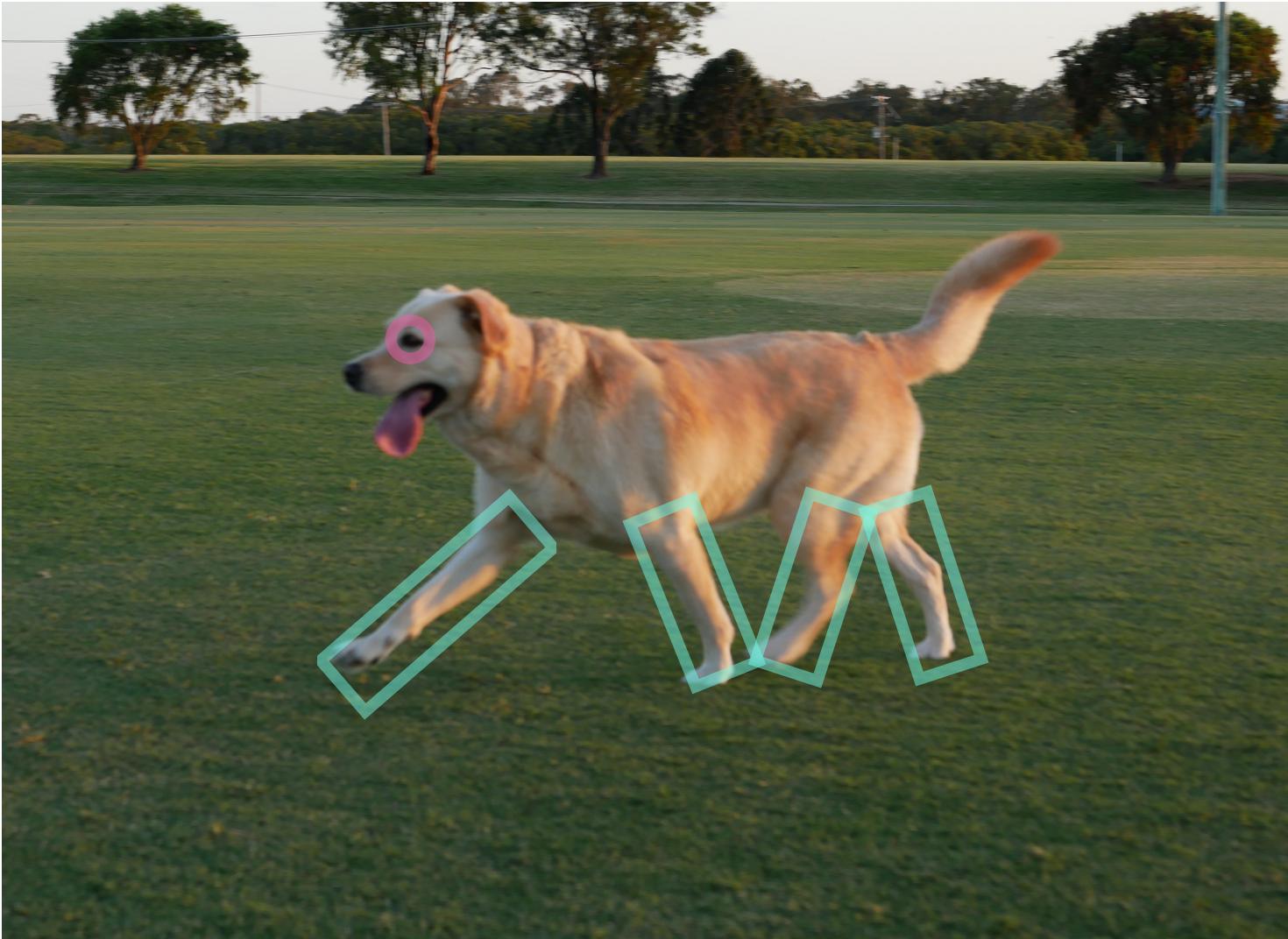
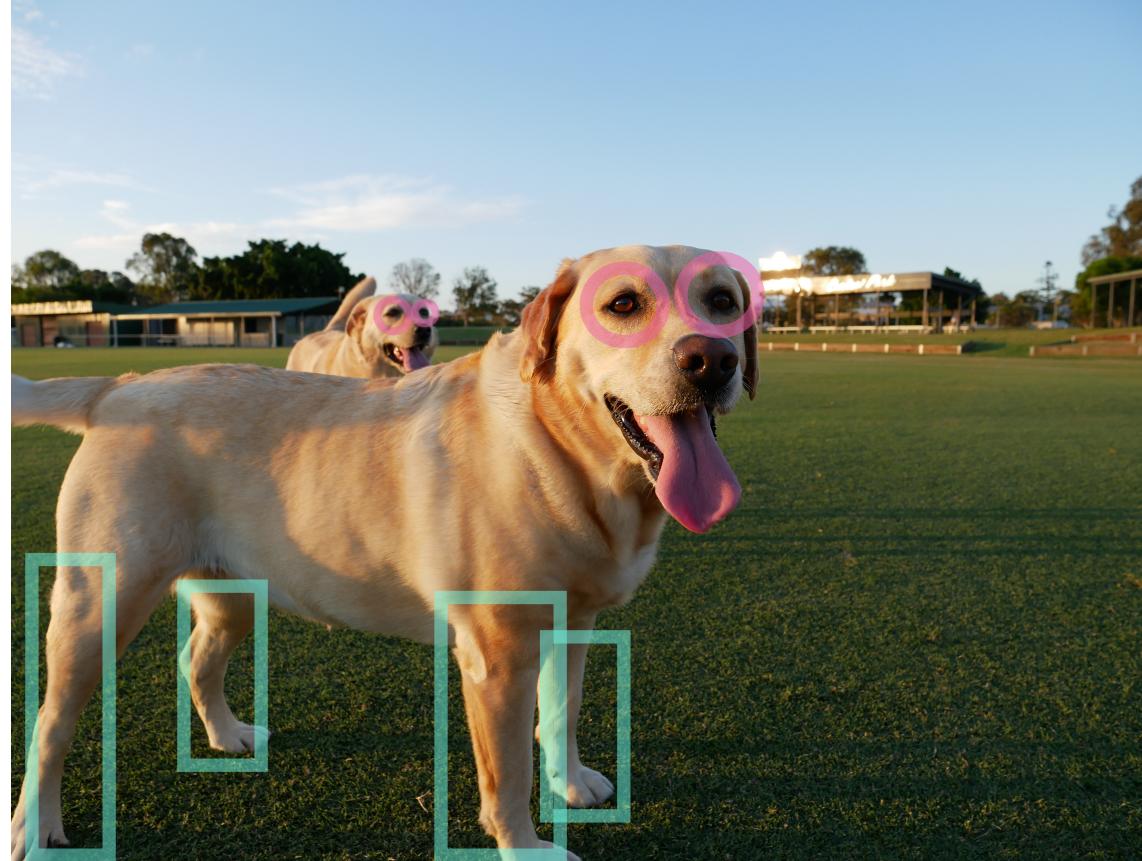
Numerical features

Categorical features

## Feature engineering

Looking at different features of data and creating new ones/altering existing ones

# Different features of data



# What features should you use?

ID	Weight	Sex	Heart Rate	Chest pain	Heart disease?	Most eaten food
4326	110Kg	M	81	4	yes	Fries
5681	64Kg	F	61	1	NO	?
7911	81Kg	m	57	0	NO	?

Table 1.0: Patient records

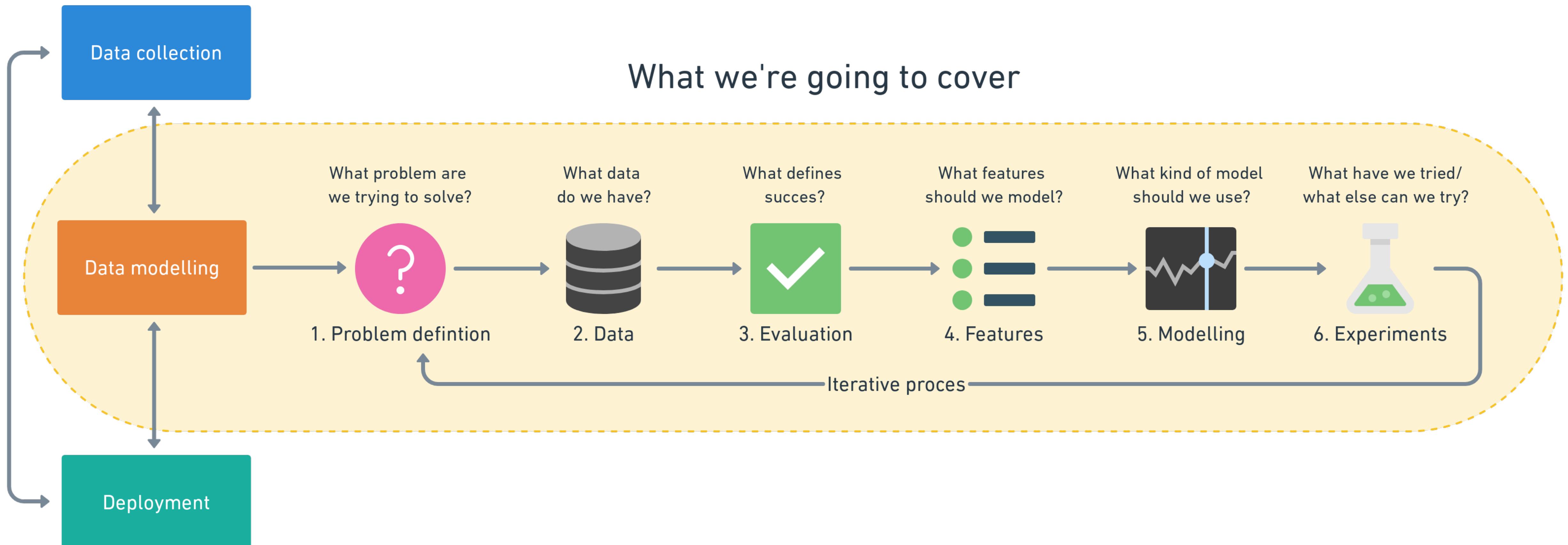
Want > 10% coverage

Feature coverage

How many samples have different features? Ideally, every sample has the same features.

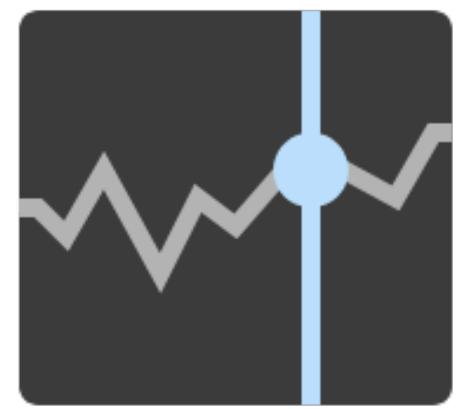
**What are features of your  
problems?**

## Steps in a full machine learning project



## What we're going to cover

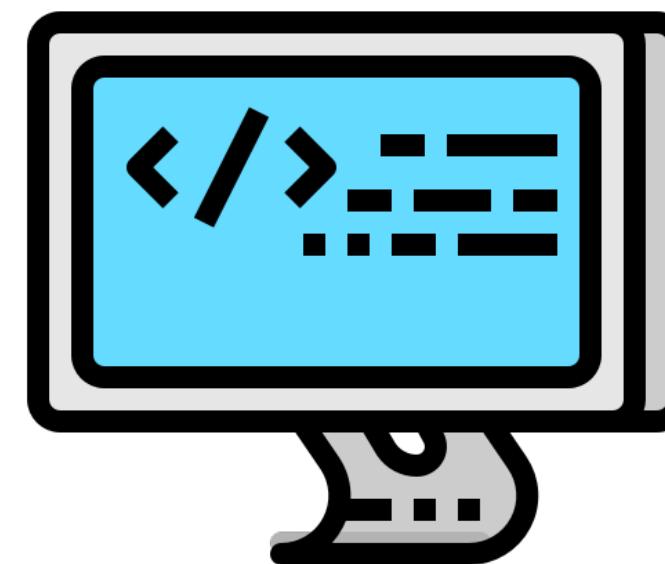
# **5. Modelling Part 1 – 3 sets**



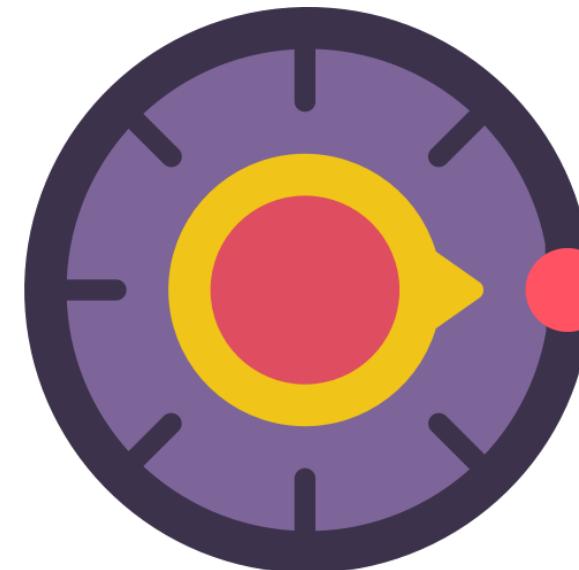
**“Based on our problem and data, what model should we use?”**

# 3 parts to modelling

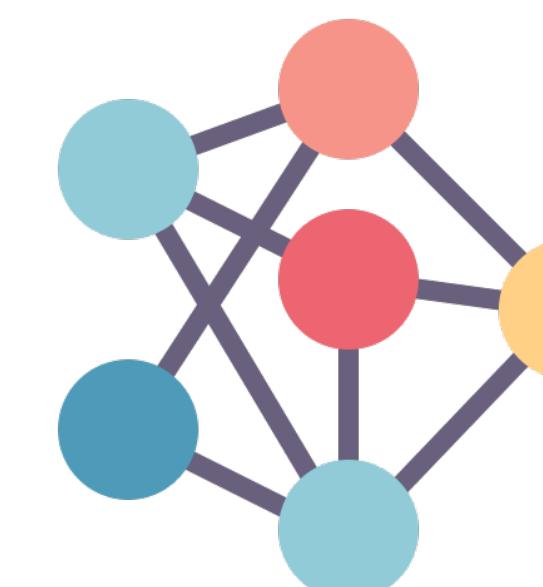
1. Choosing and training a model



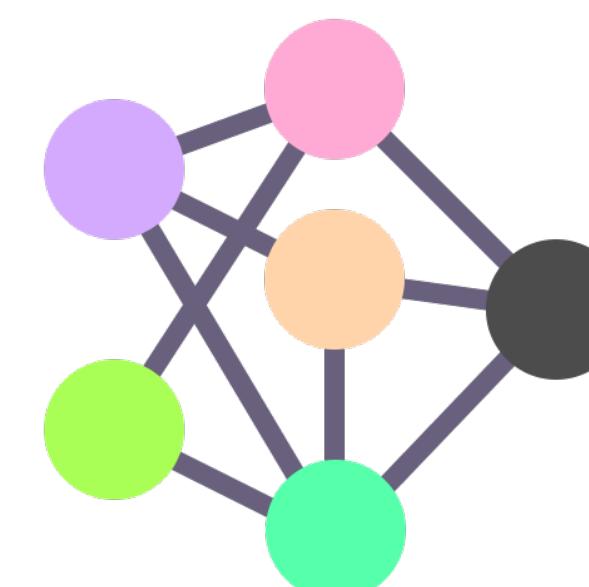
2. Tuning a model



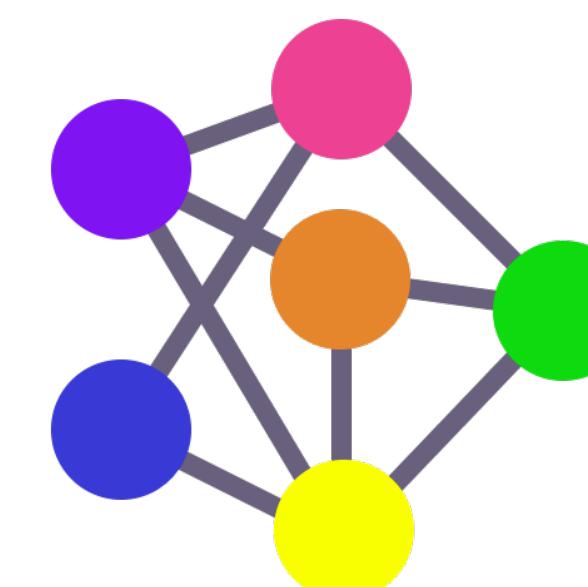
3. Model comparison



vs.

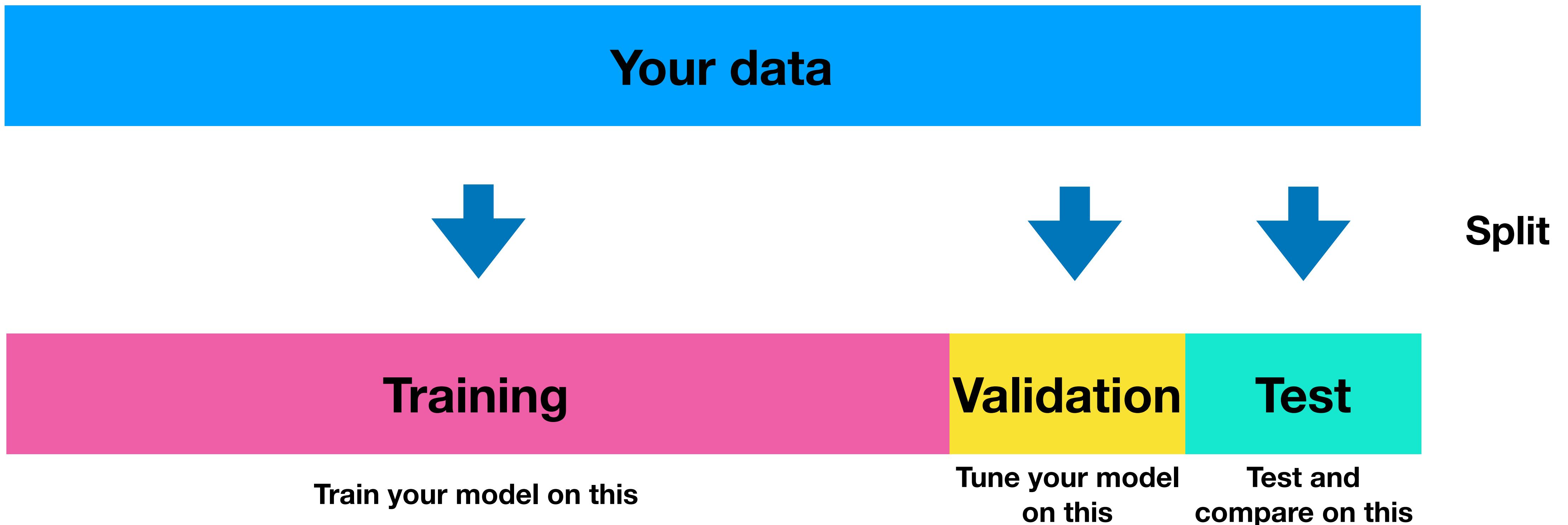


vs.



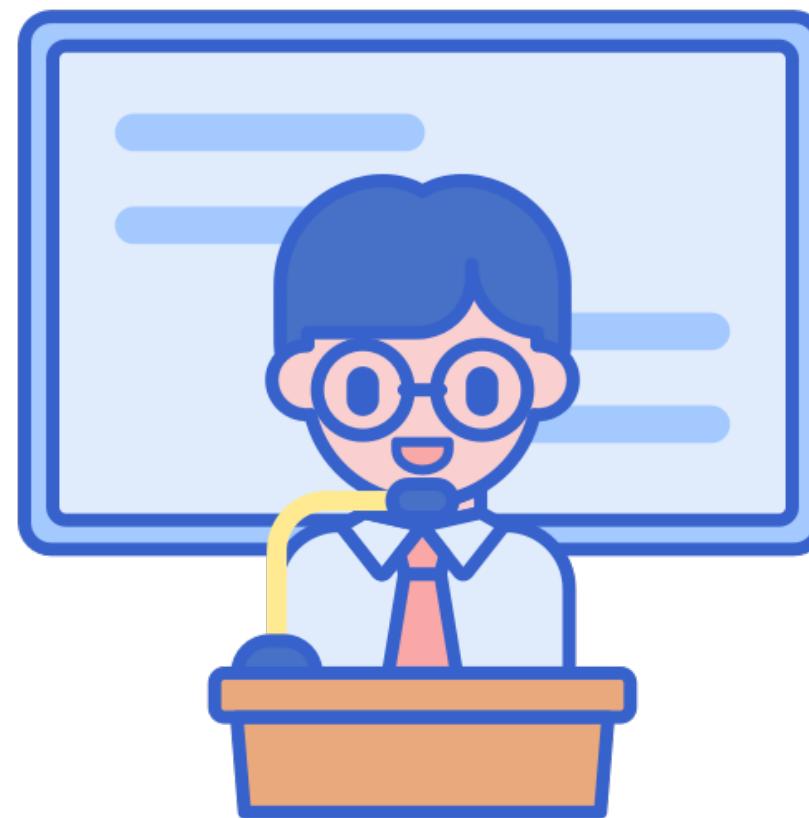
# The most important concept in machine learning

(the training, validation and test sets or 3 sets)

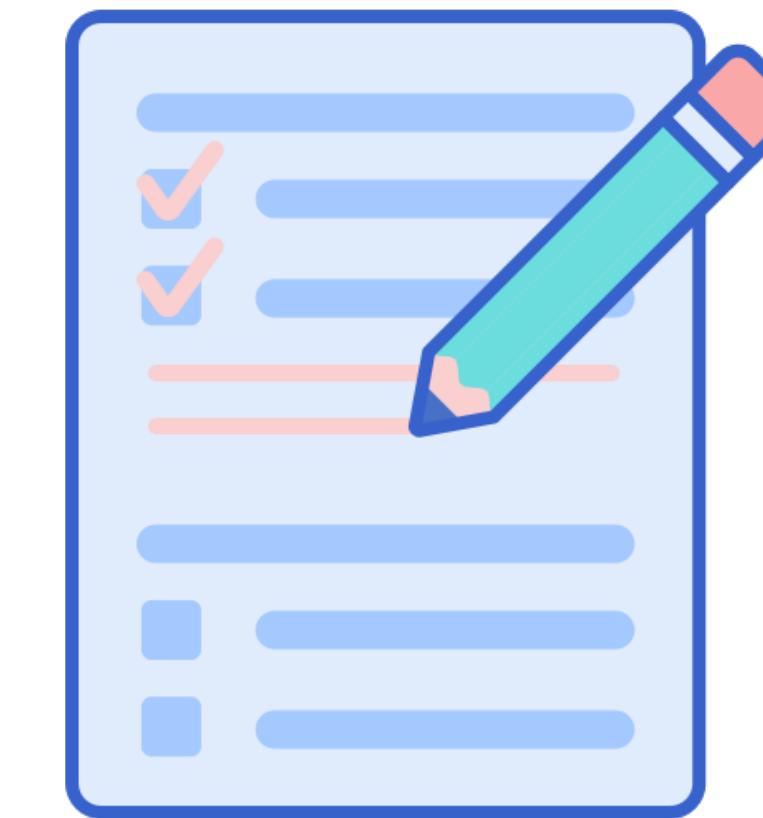
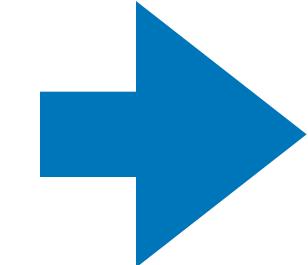
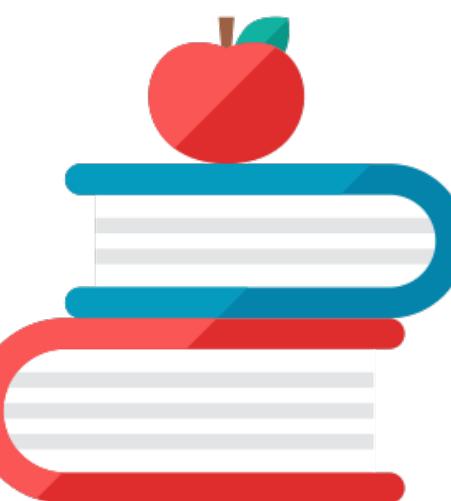


# The most important concept in machine learning

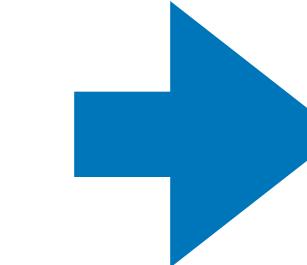
(the 3 sets)



**Course materials  
(training set)**



**Practice exam  
(validation set)**

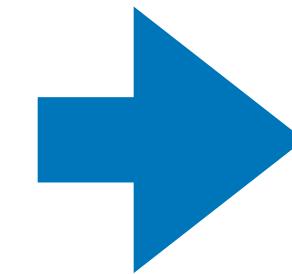
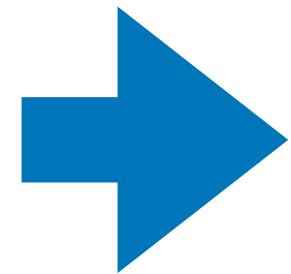
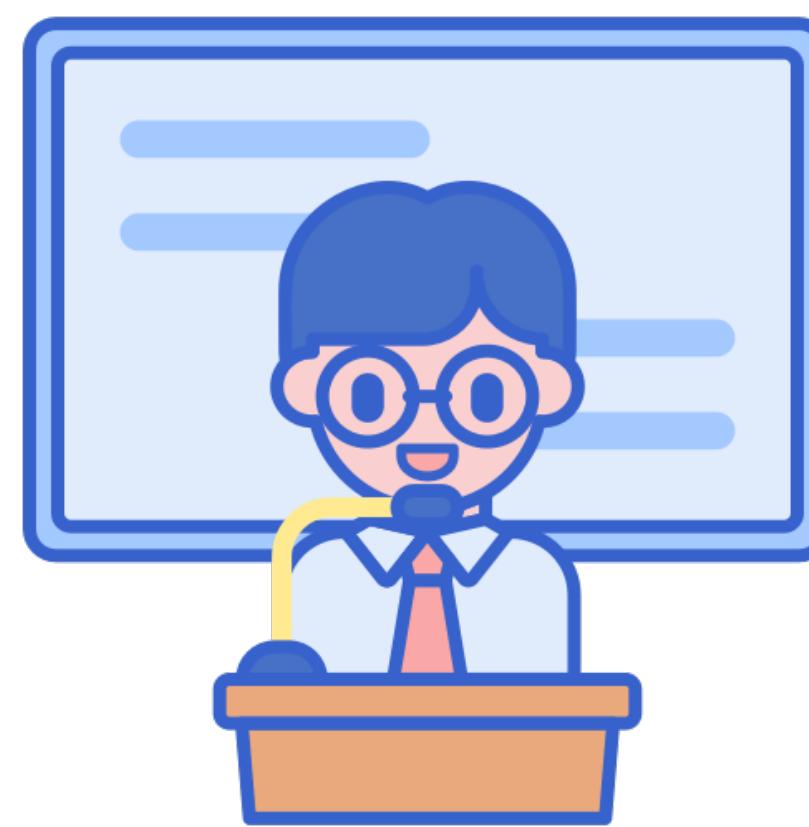


**Final exam  
(test set)**

**Generalization**

The ability for a machine learning model to perform well on data it hasn't seen before.

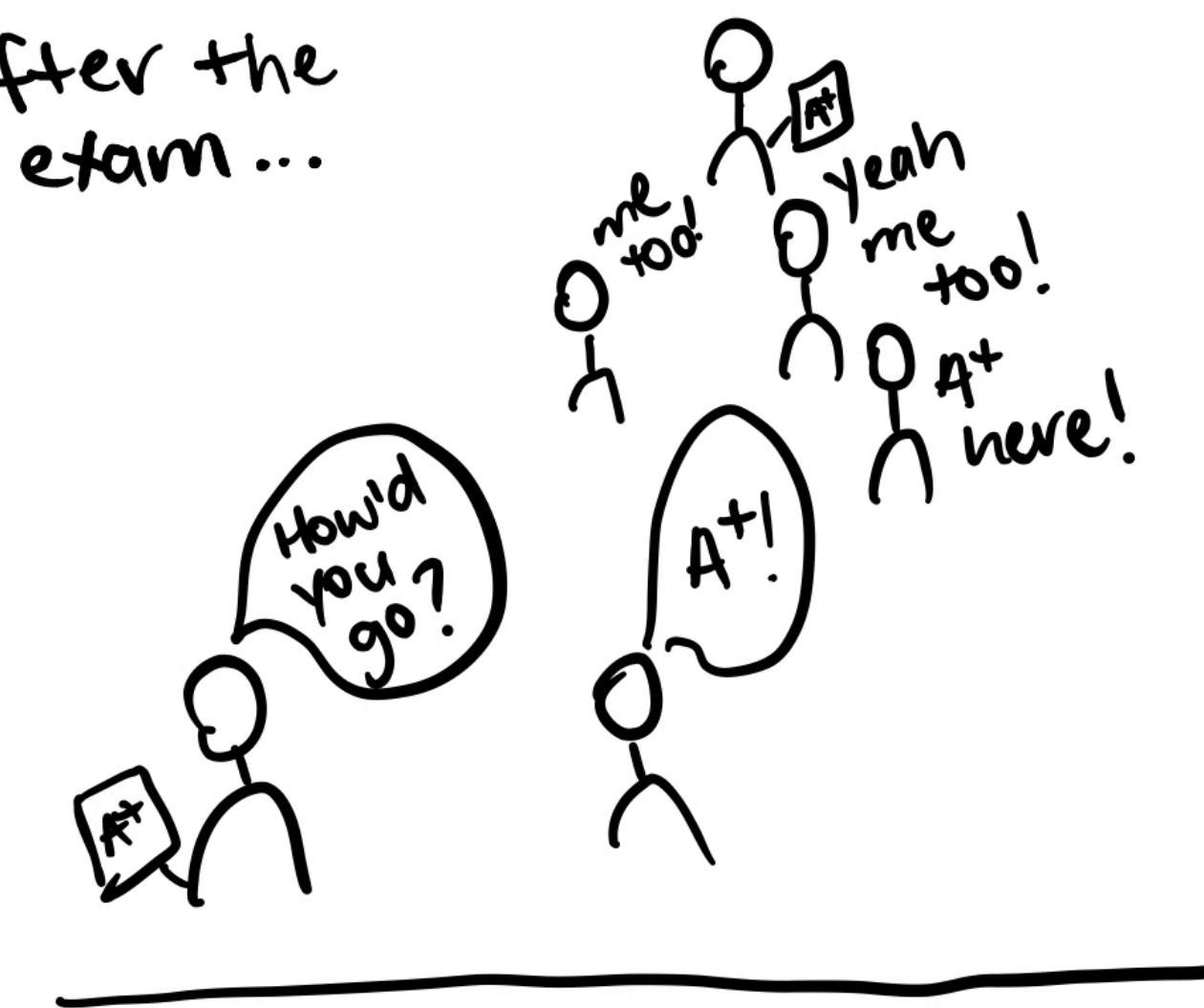
# When things go wrong



**Practice exam**  
(same as final exam set)

**Final exam**  
(already seen it)

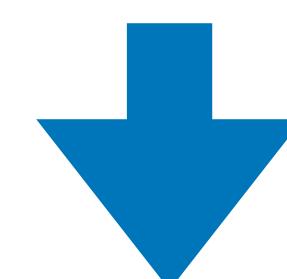
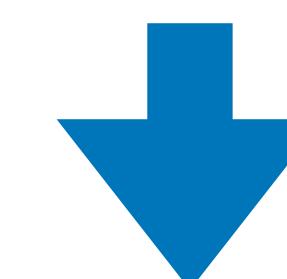
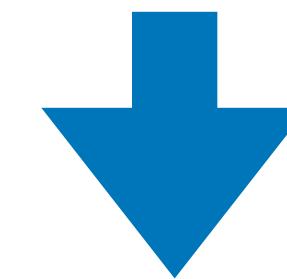
After the  
exam...



# The most important concept in machine learning

(the 3 sets)

100 patient records



Split

70 patient records

15

15

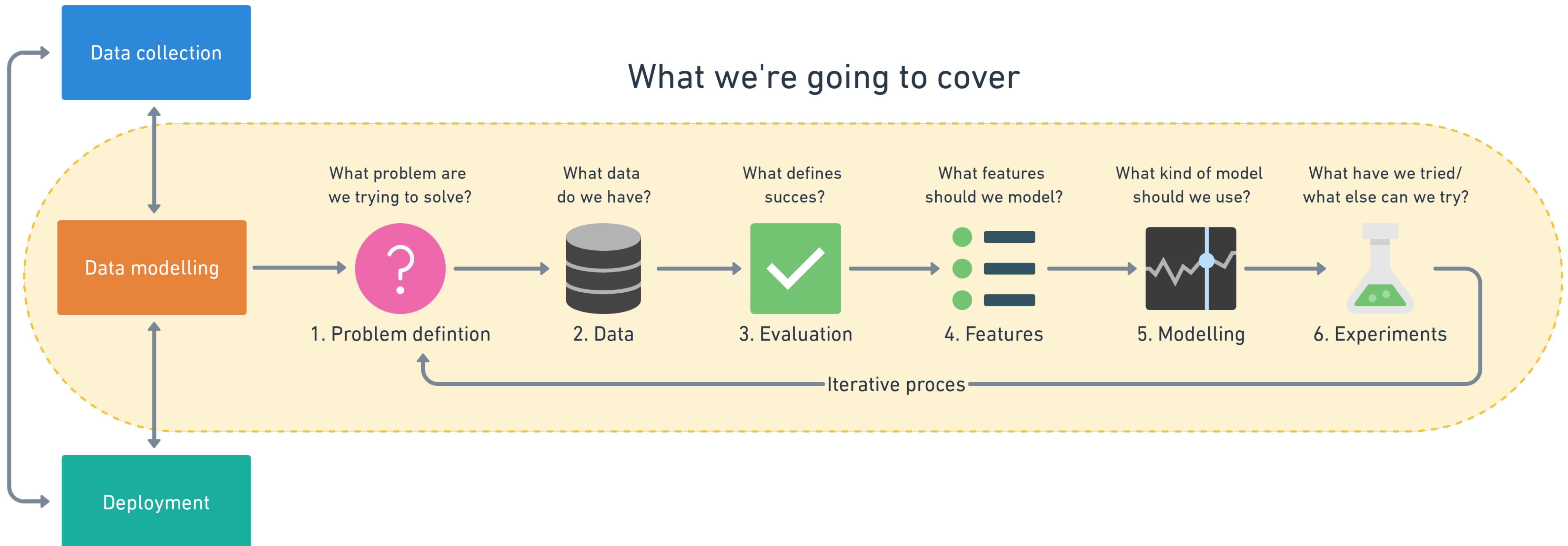
Training split (70-80%)

Validation split  
(10-15%)

Test split  
(10-15%)

**What was the last thing you testing  
your ability on?**

## Steps in a full machine learning project



## What we're going to cover

# 5. Modelling Part 2 – Choosing

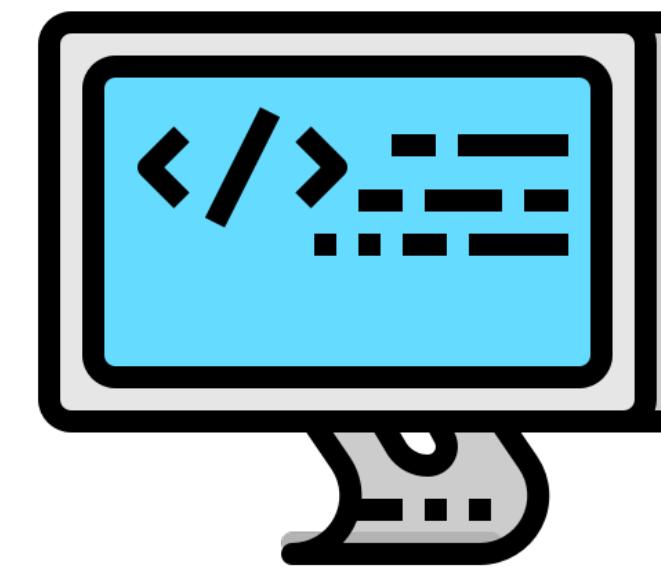


**“Based on our problem and data, what model should we use?”**

# 3 parts to modelling

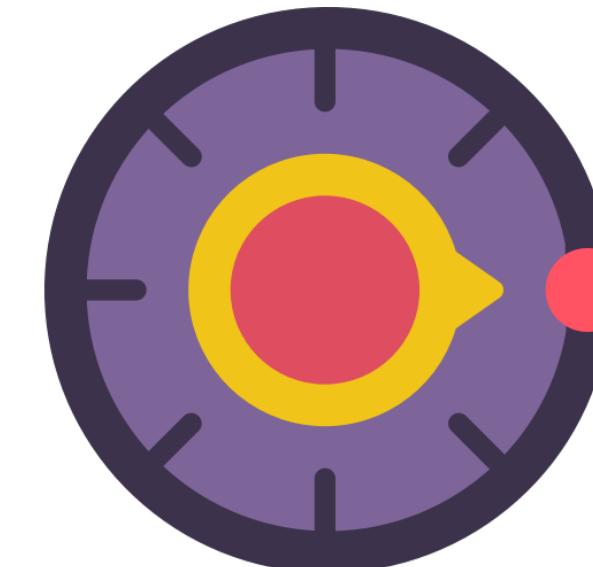
## 1. Choosing and training a model

Training Data



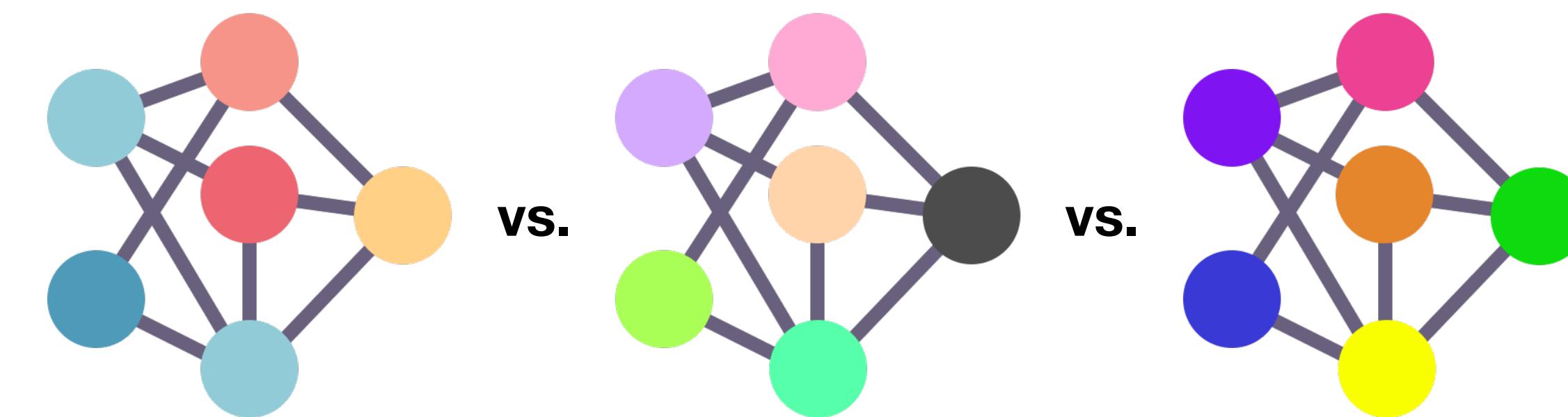
## 2. Tuning a model

Validation Data

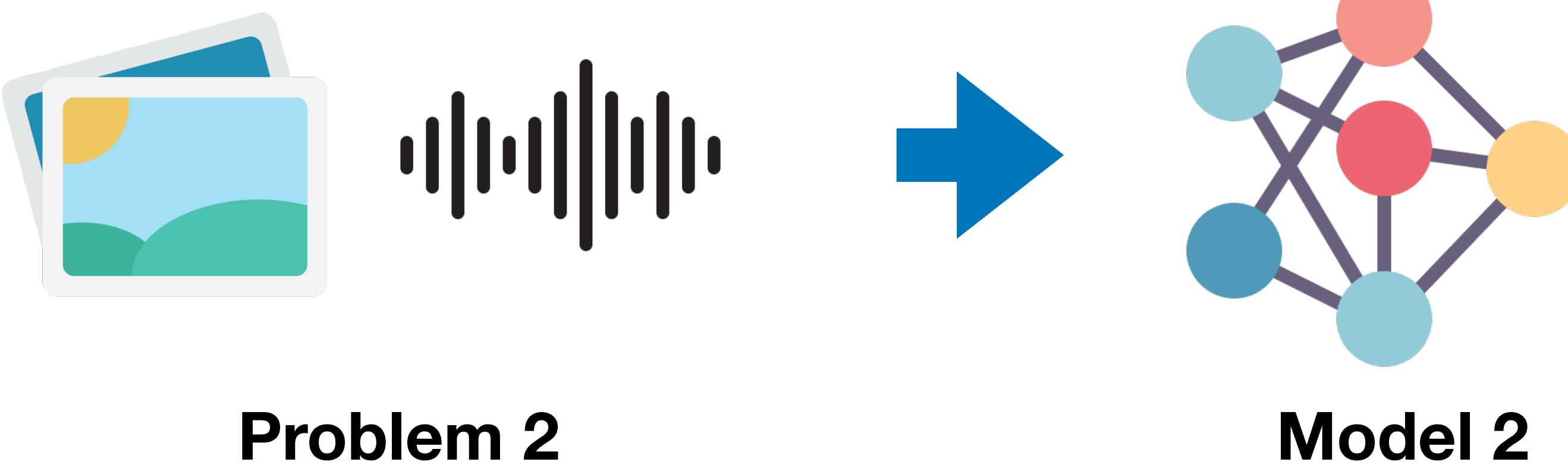


## 3. Model comparison

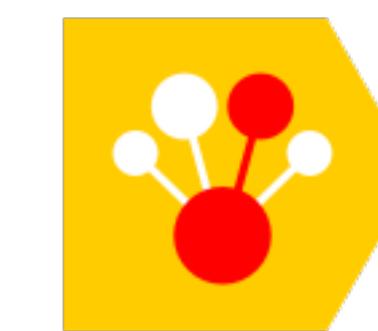
Test Data



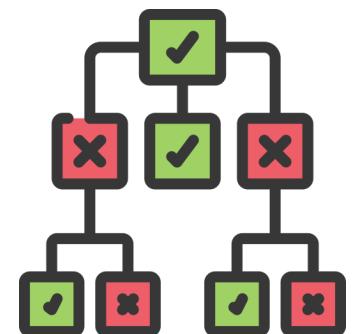
# Choosing a model



**Structured Data**



*dmlc*  
**XGBoost**

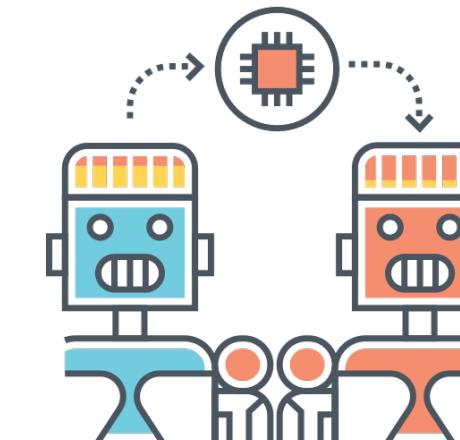


**Random Forest**

**Unstructured Data**

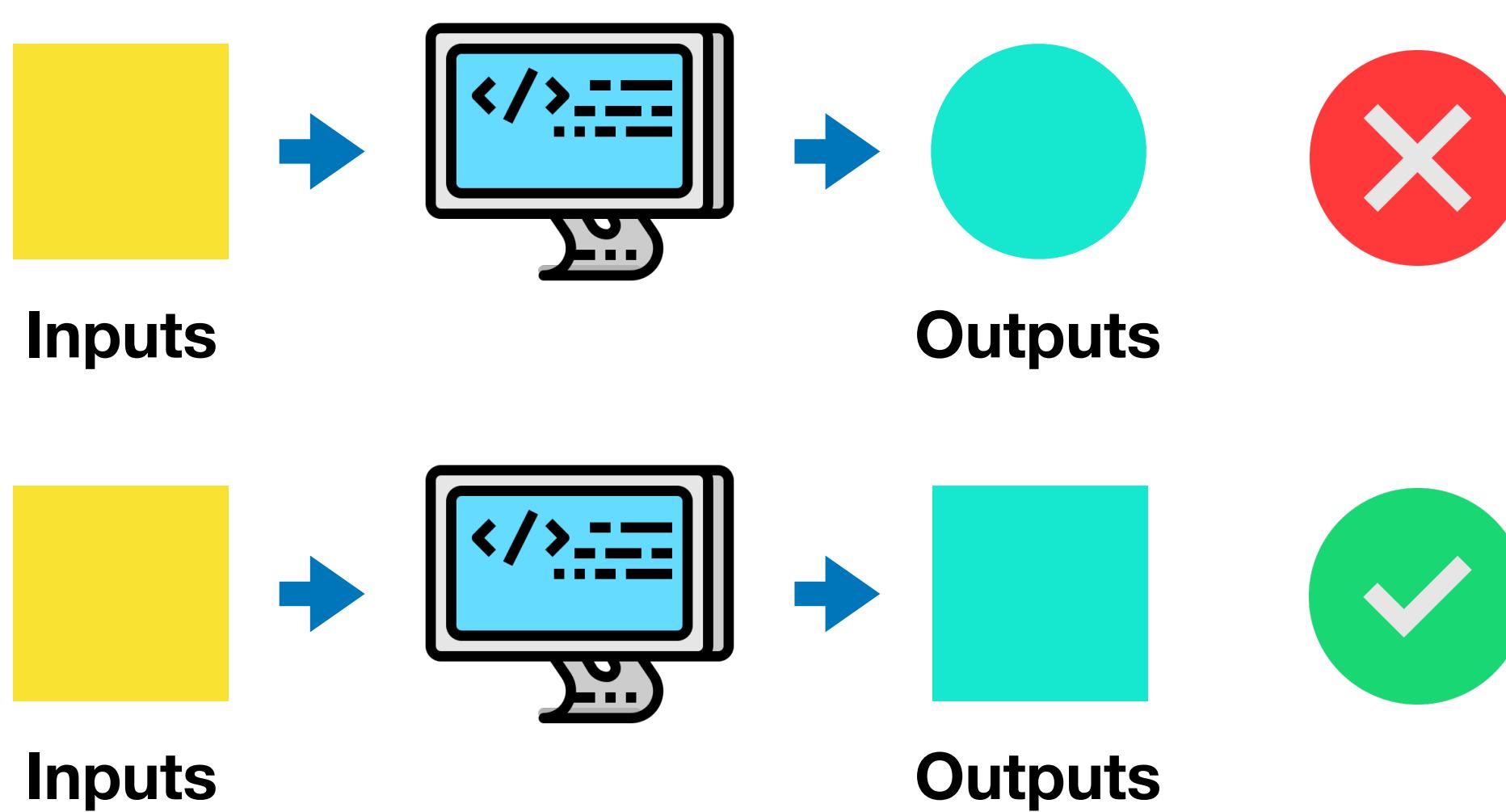


**Deep Learning**



**Transfer Learning**

# Training a model



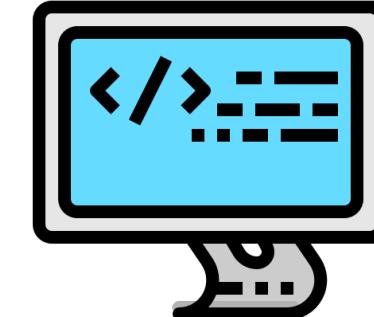
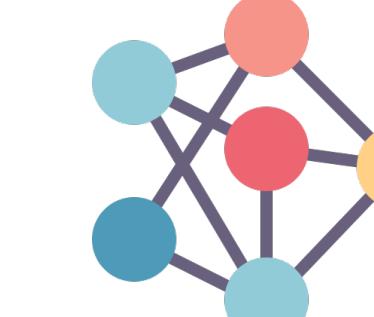
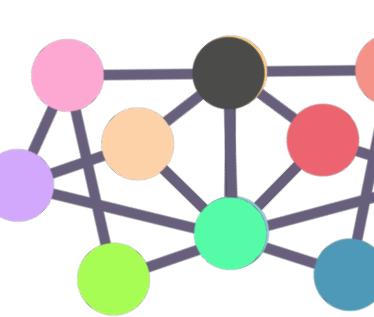
ID	weight	Sex	Heart Rate	Chest pain	y (label)
4326	110Kg	M	81	4	Yes
5681	64Kg	F	61	1	No
7911	81Kg	M	57	0	No

Table 1.0 : Patient records

Training Data

# Goal: Minimise time between experiments

## Experiment

	Inputs	Model	Outputs	Accuracy	Training time
1				87.5%	3 min
2				91.3%	92 min
3				94.7%	176 min

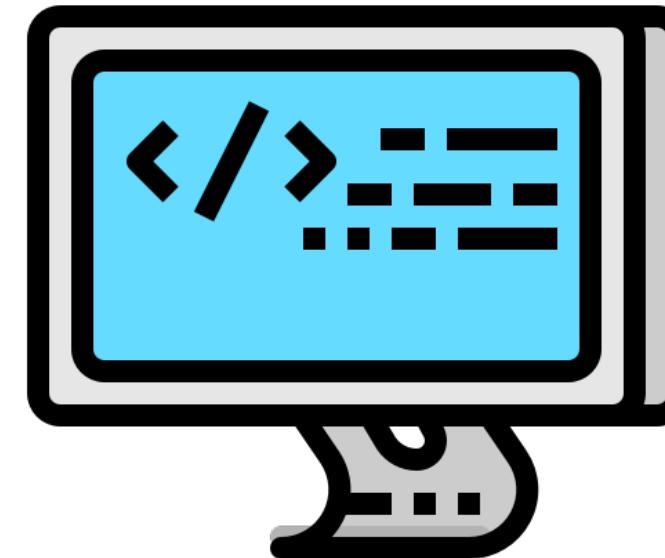
# Things to remember

- Some models work better than others on different problems
- Don't be afraid to try things
- Start small and build up (add complexity) as you need

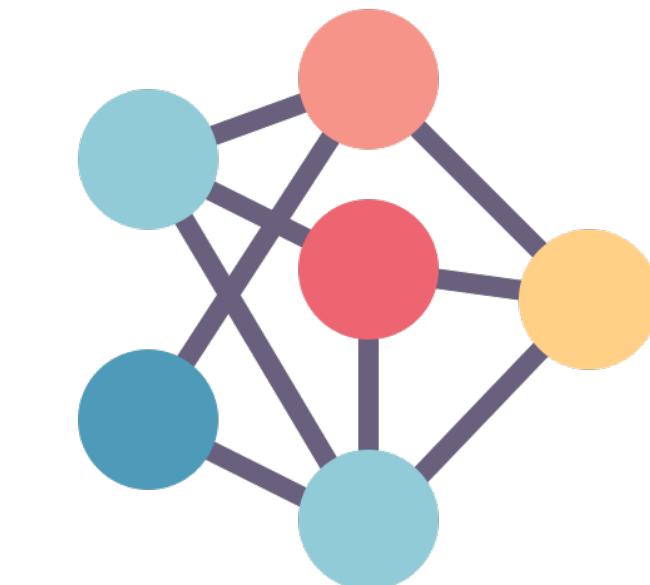
# Up next

## 1. Choosing and training a model

Training Data

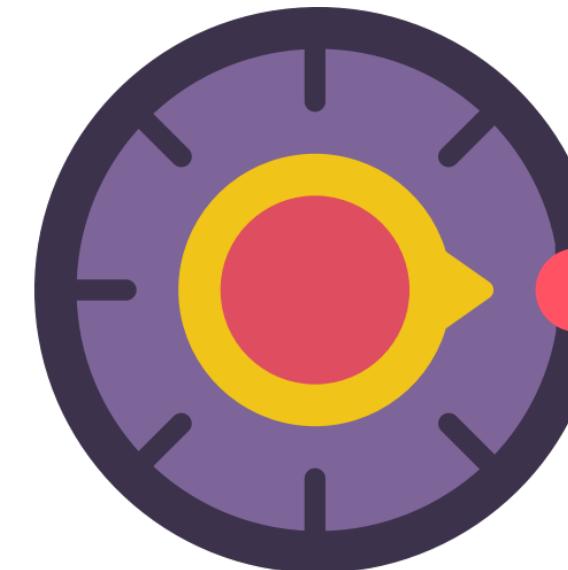


or



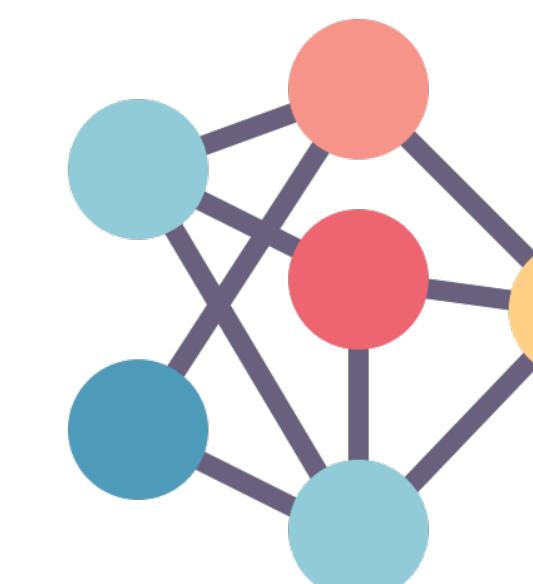
## 2. Tuning a model

Validation Data

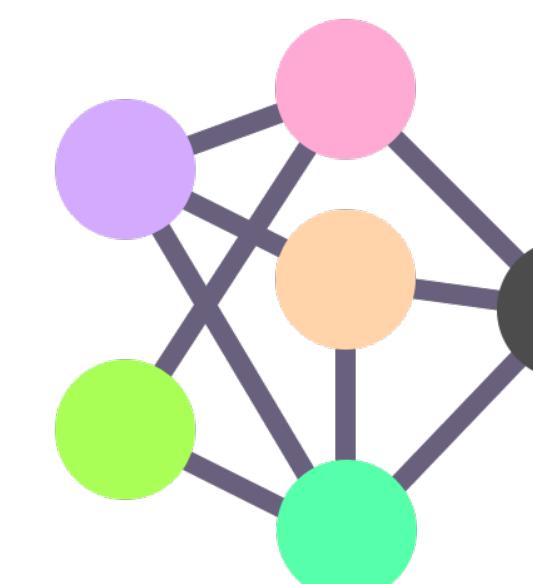


## 3. Model comparison

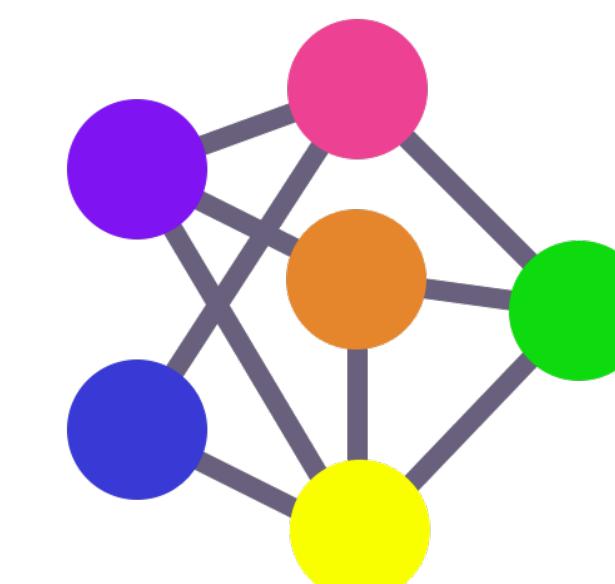
Test Data



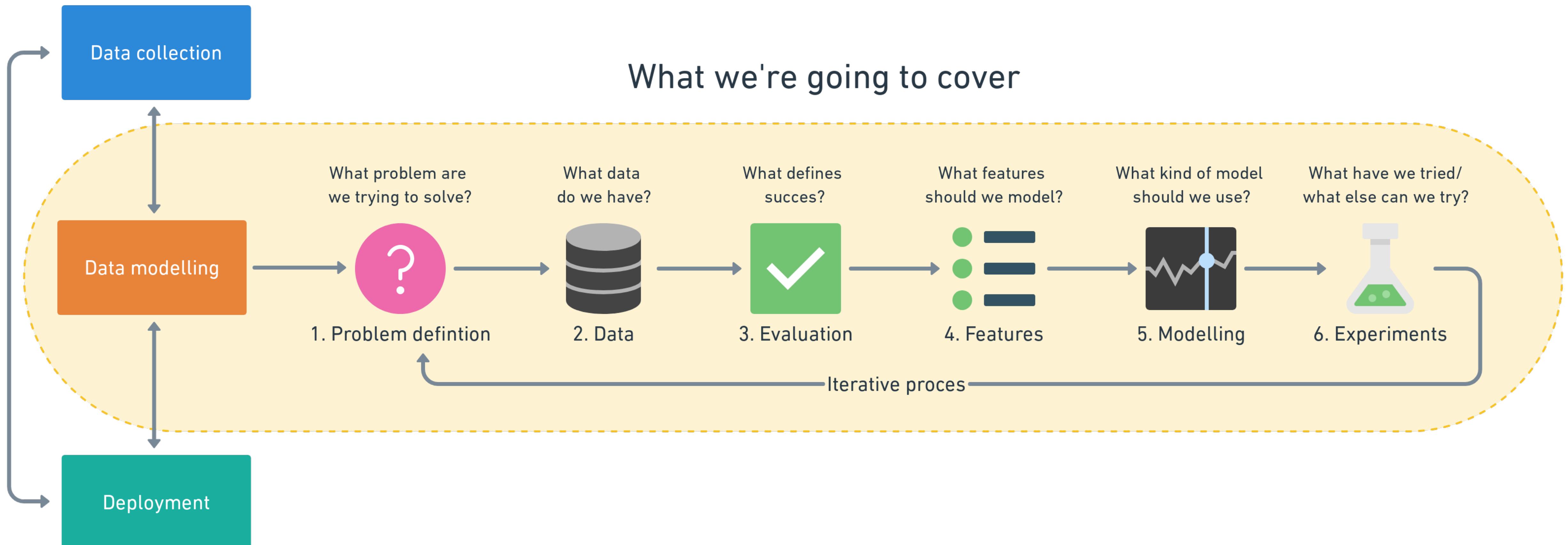
vs.



vs.

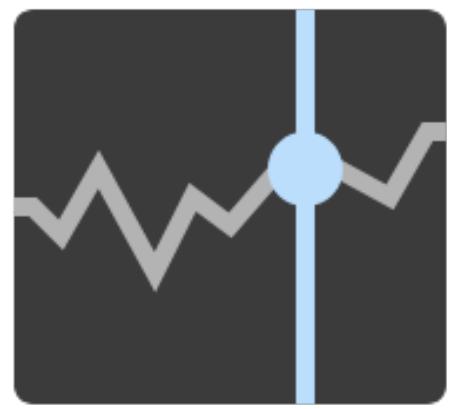


## Steps in a full machine learning project



## What we're going to cover

# 5. Modelling Part 3 – Tuning

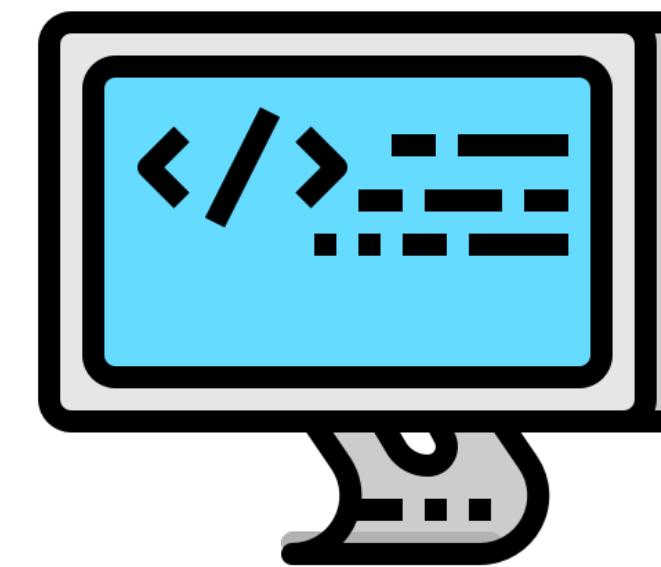


**“Based on our problem and data, what model should we use?”**

# 3 parts to modelling

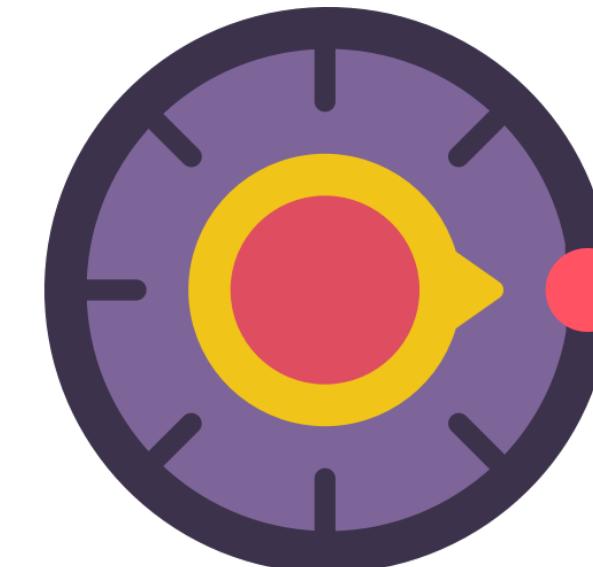
## 1. Choosing and training a model

Training Data



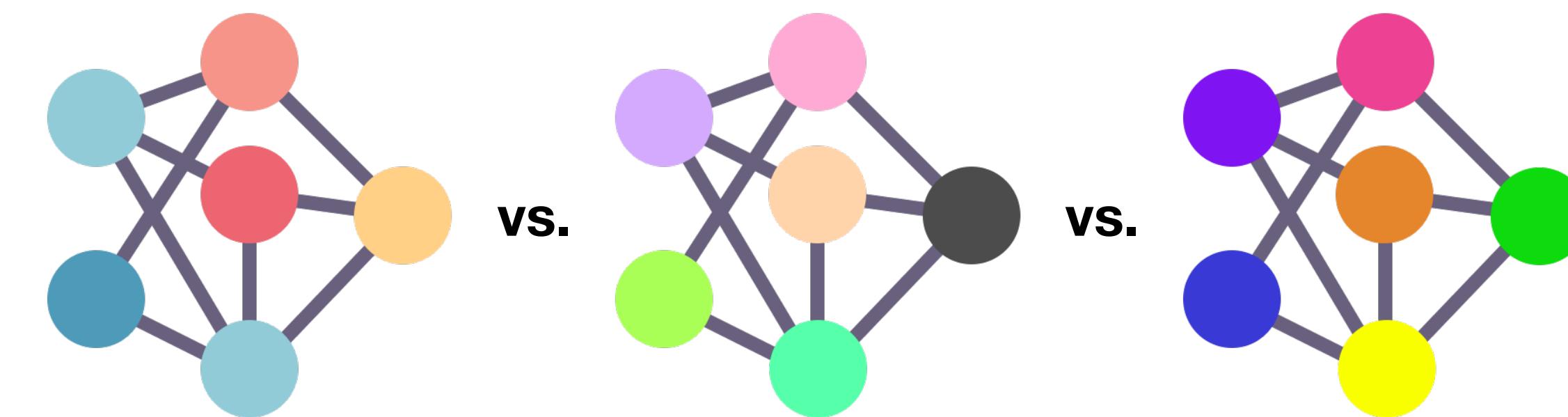
## 2. Tuning a model

Validation Data



## 3. Model comparison

Test Data



# Tuning a model



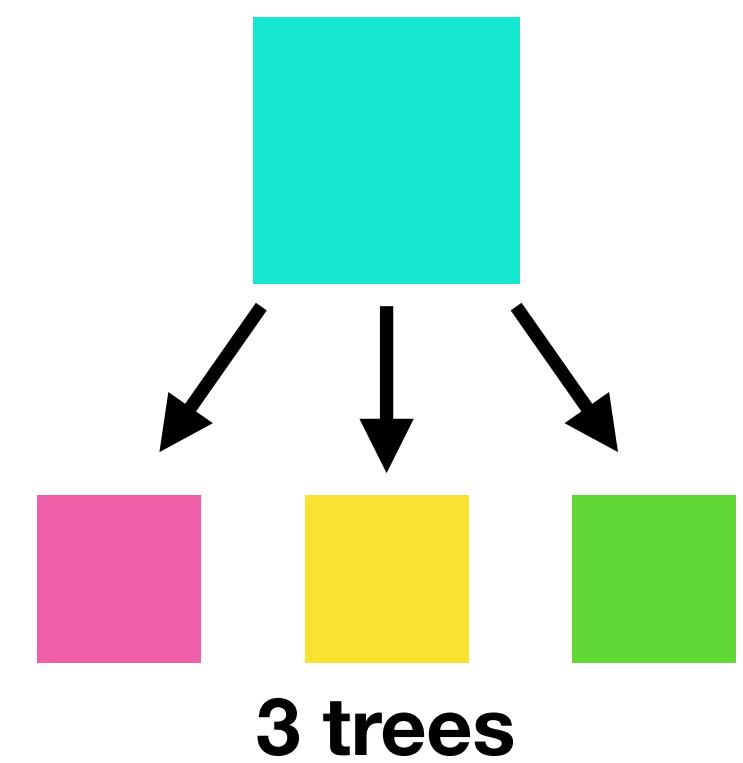
Cooking time: 1 hour  
Temperature: 180°C



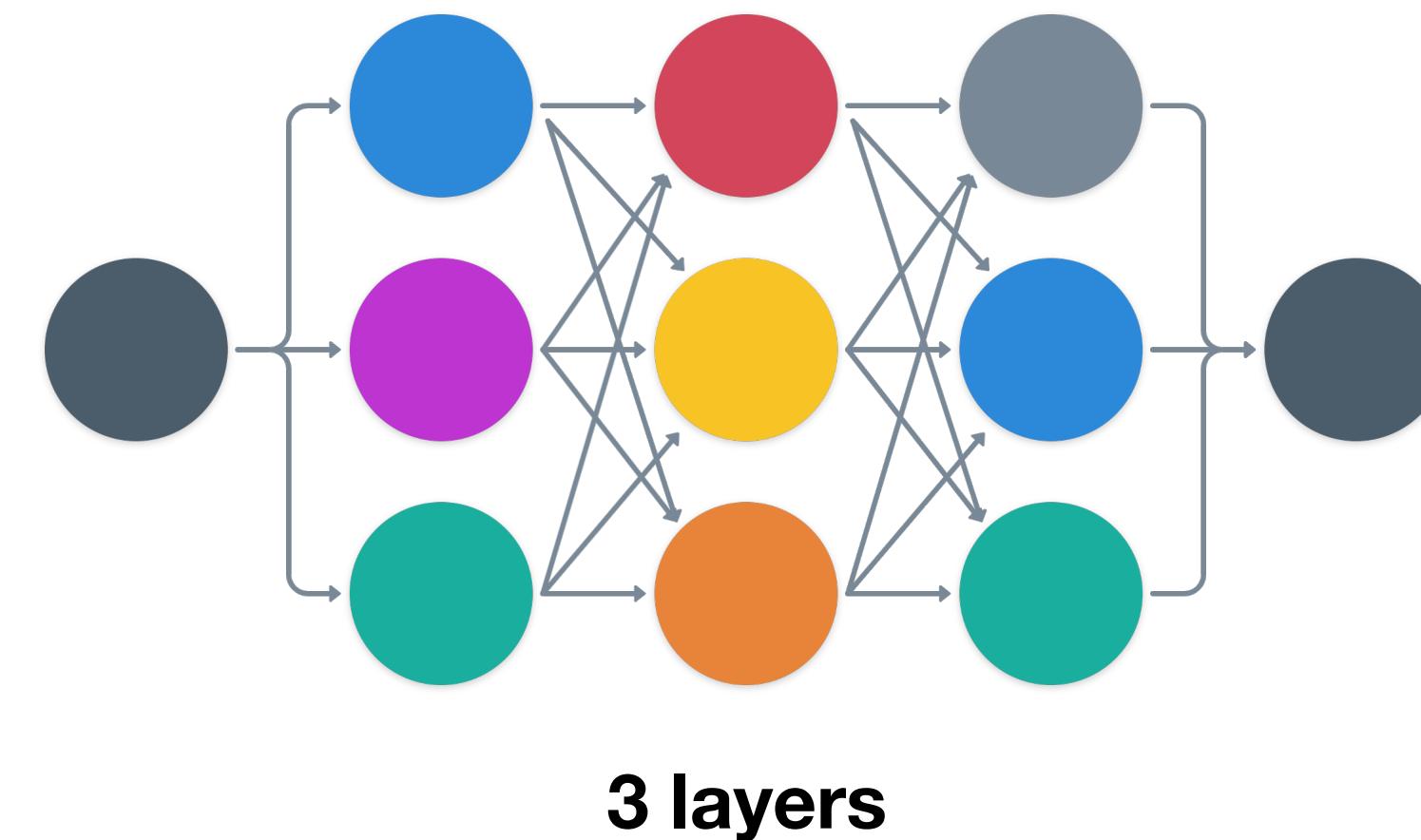
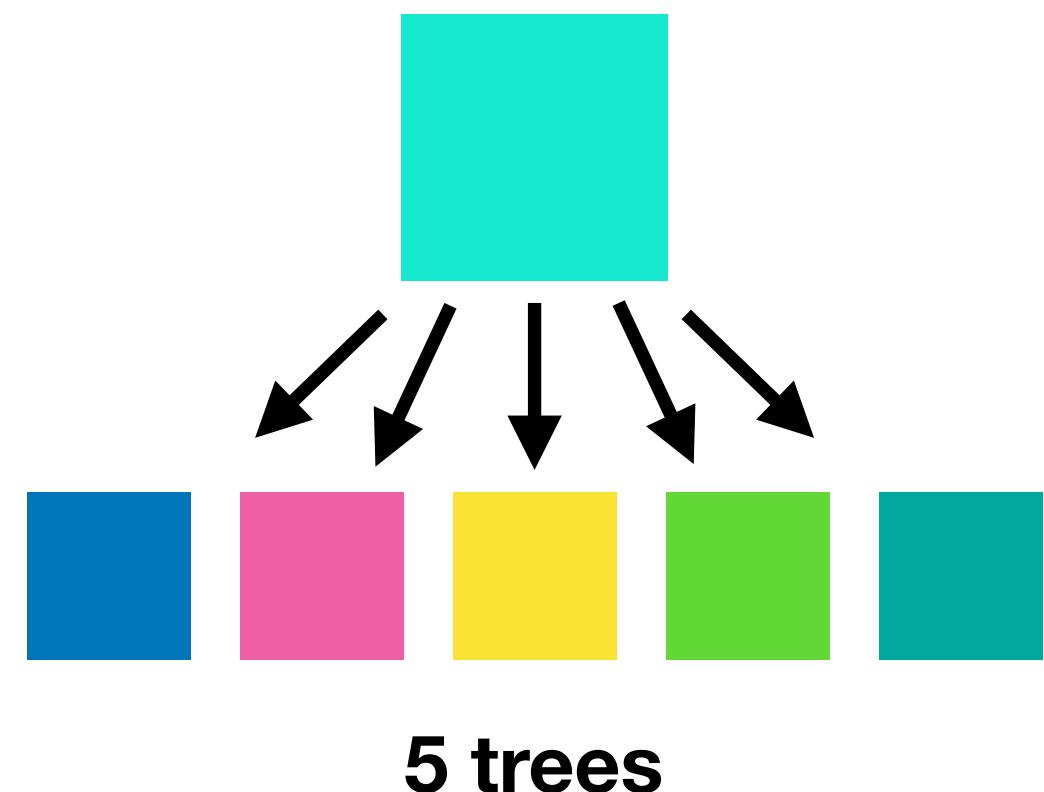
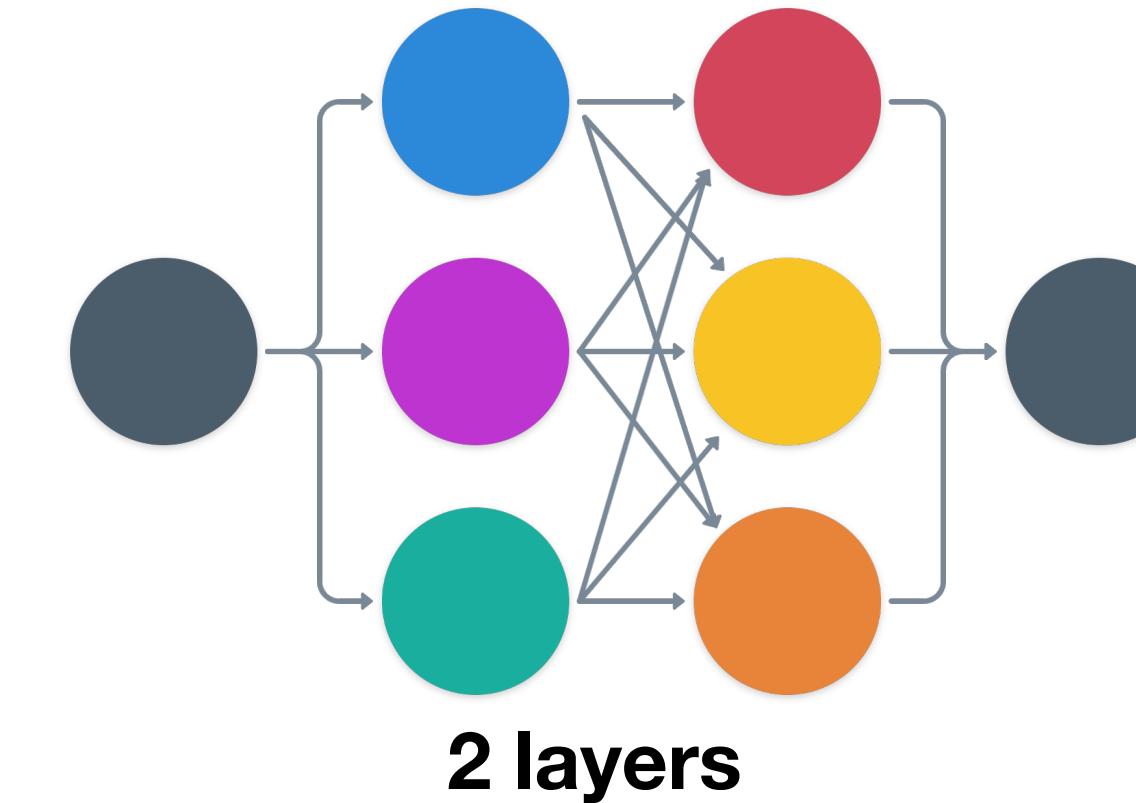
Cooking time: 1 hour  
Temperature: 200°C

# Tuning a model

**Random Forest**



**Neural Networks**



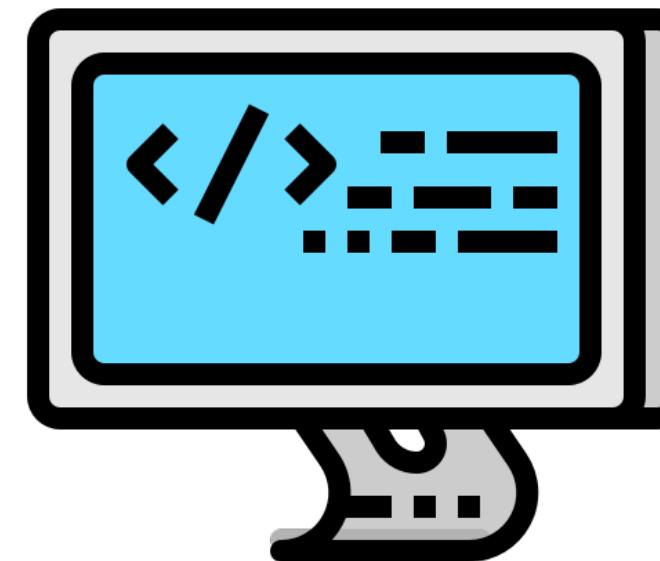
# Things to remember

- Machine learning models have hyperparameters you can adjust
- A models first results aren't its last
- Tuning can take place on training or validation data sets

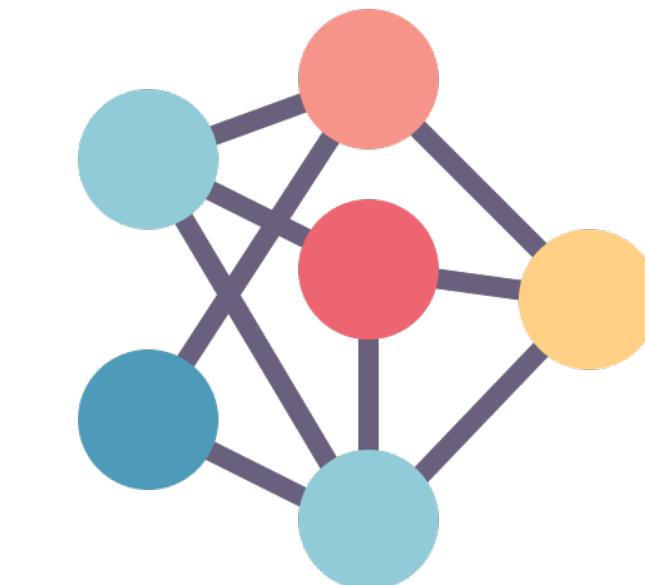
# Up next

## 1. Choosing and training a model

Training Data

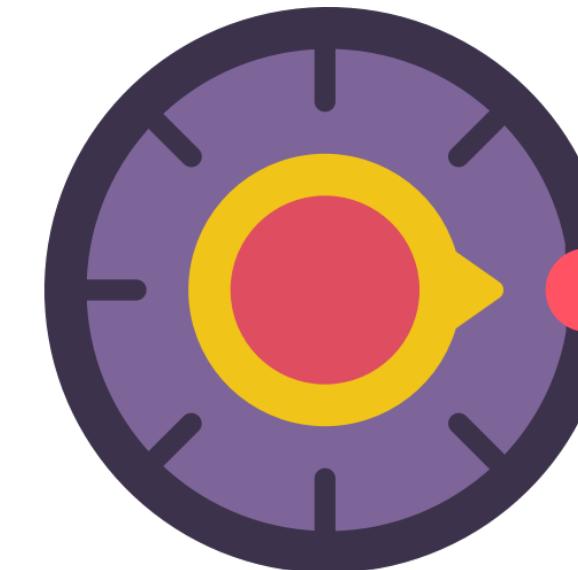


or



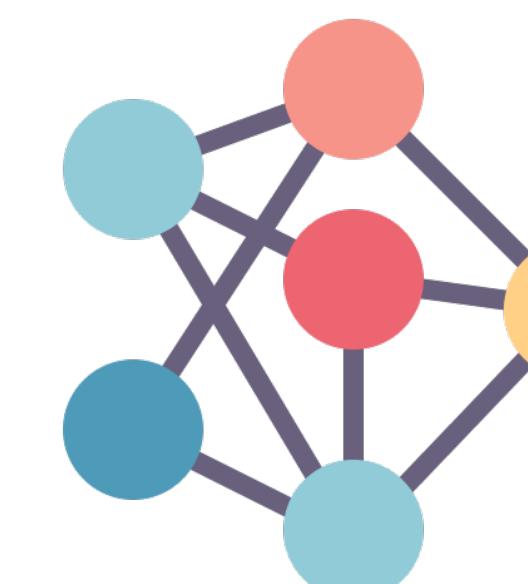
## 2. Tuning a model

Validation Data

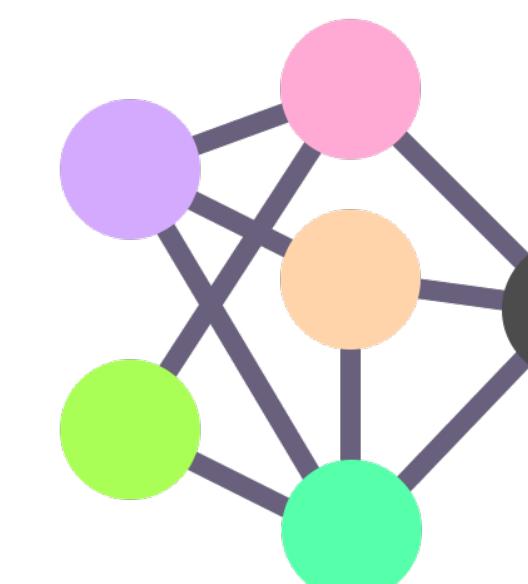


## 3. Model comparison

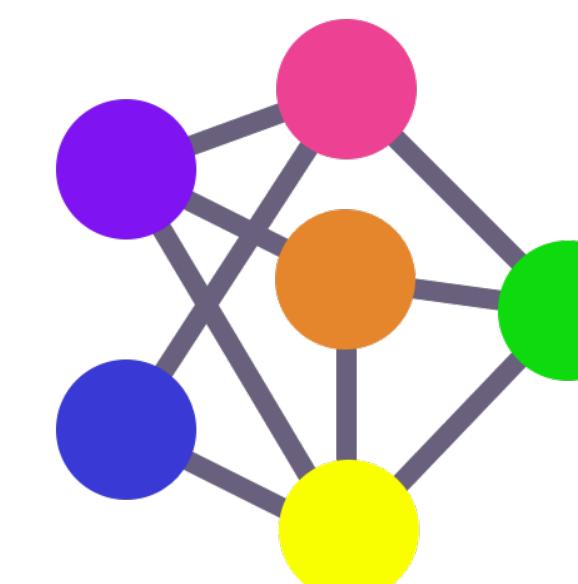
Test Data



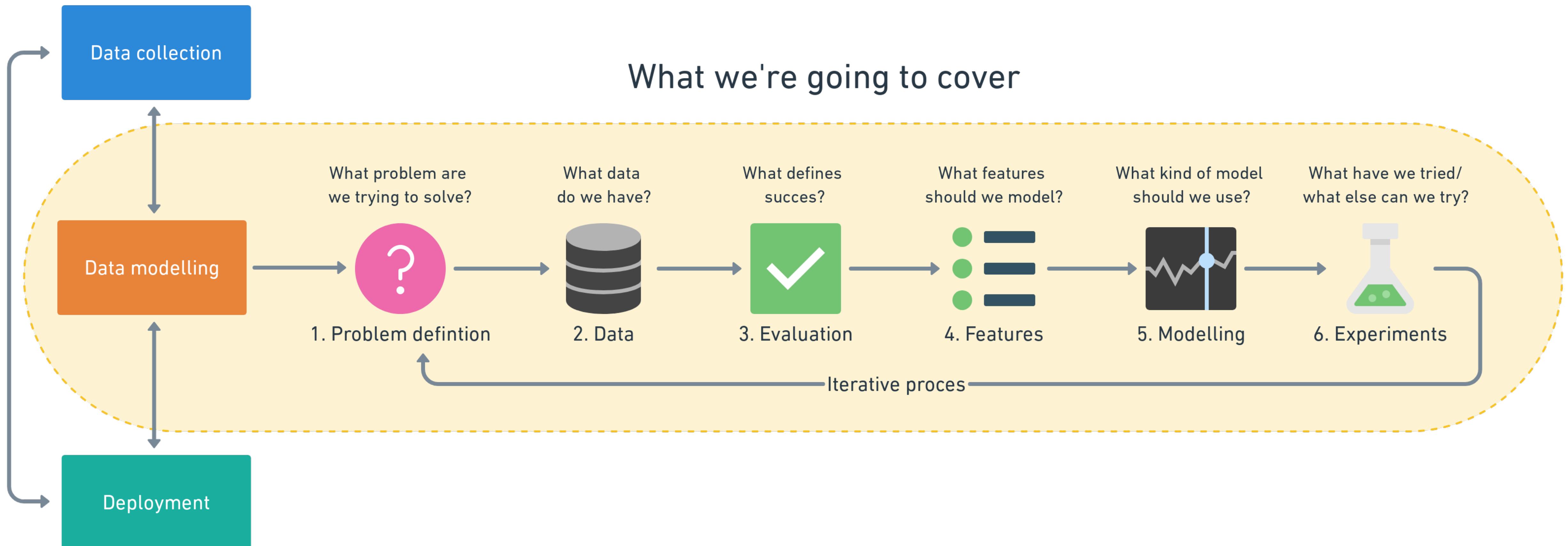
vs.



vs.



## Steps in a full machine learning project



## What we're going to cover



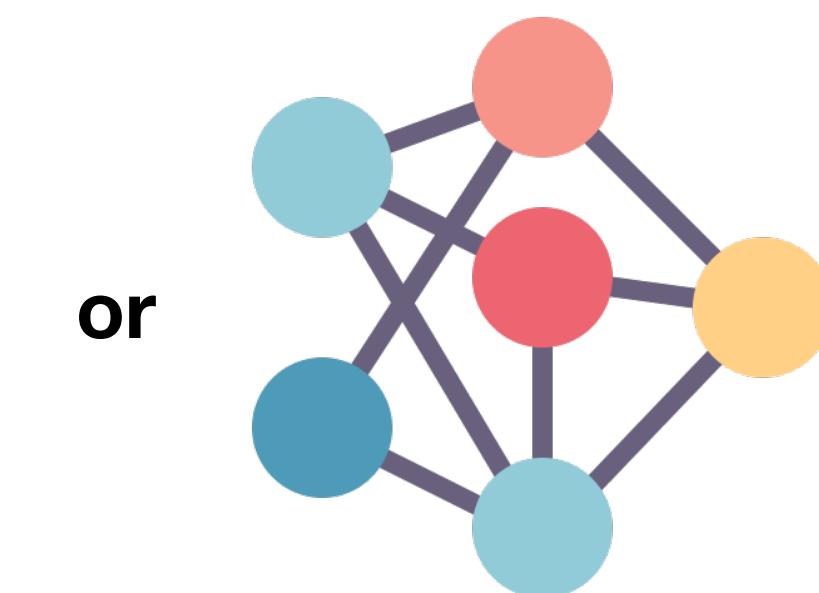
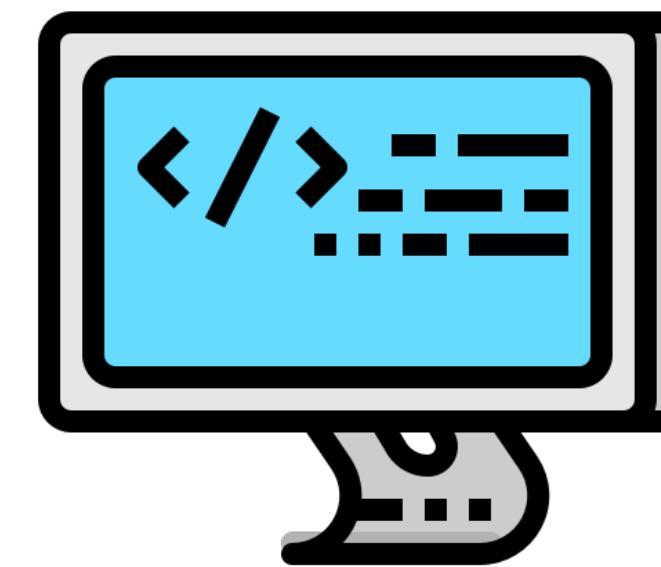
# 5. Modelling Part 4 – Comparison

**“How will our model perform in the real world?”**

# 3 parts to modelling

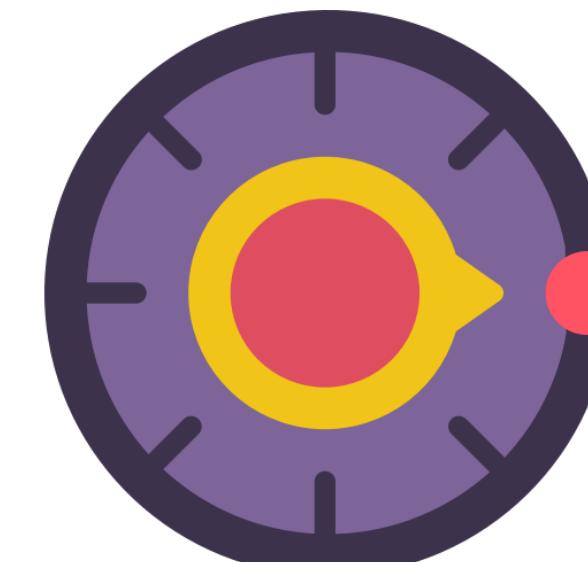
## 1. Choosing and training a model

Training Data



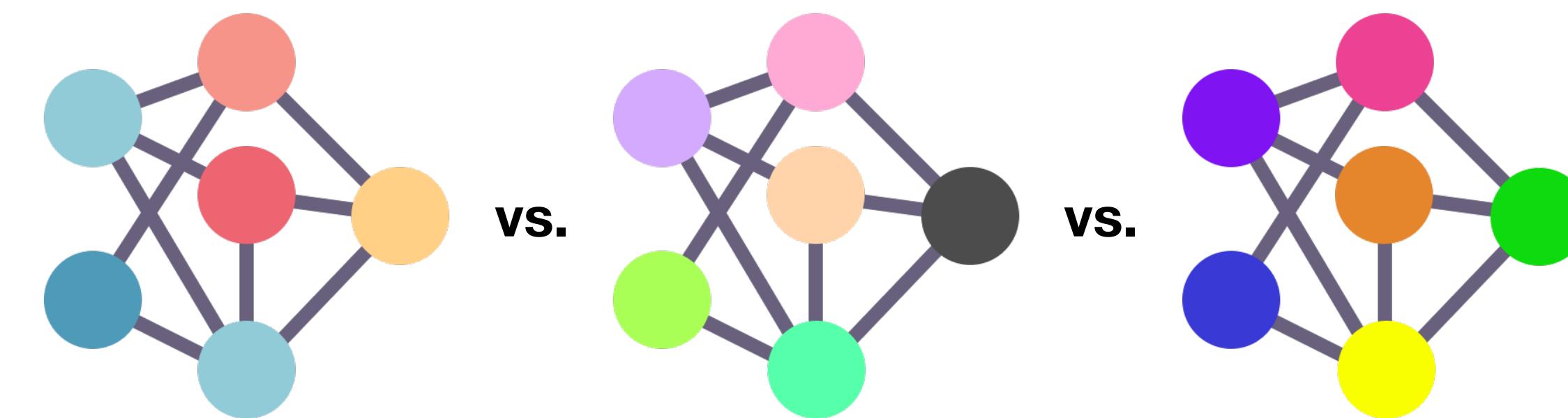
## 2. Tuning a model

Validation Data

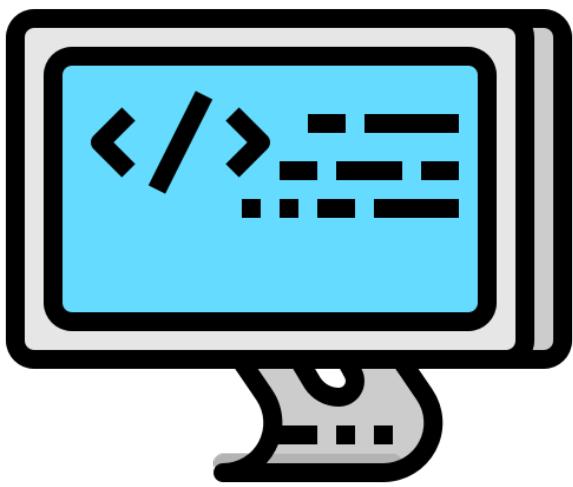


## 3. Model comparison

Test Data



# Testing a model



**Data Set**

**Performance**

Training

98%

Test

96%

Underfitting  
(potential)

**Data Set**

**Performance**

Training

64%

Test

47%

Overfitting  
(potential)

**Data Set**

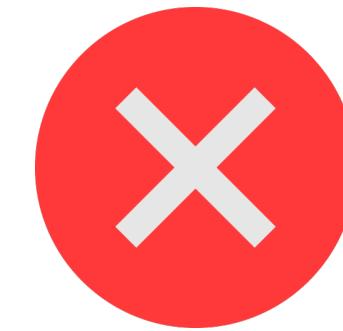
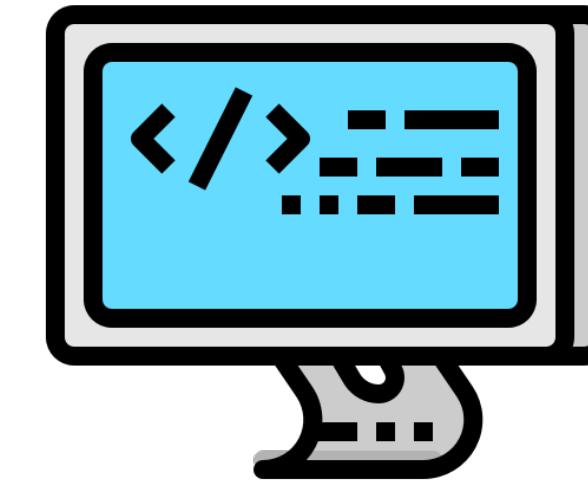
**Performance**

Training

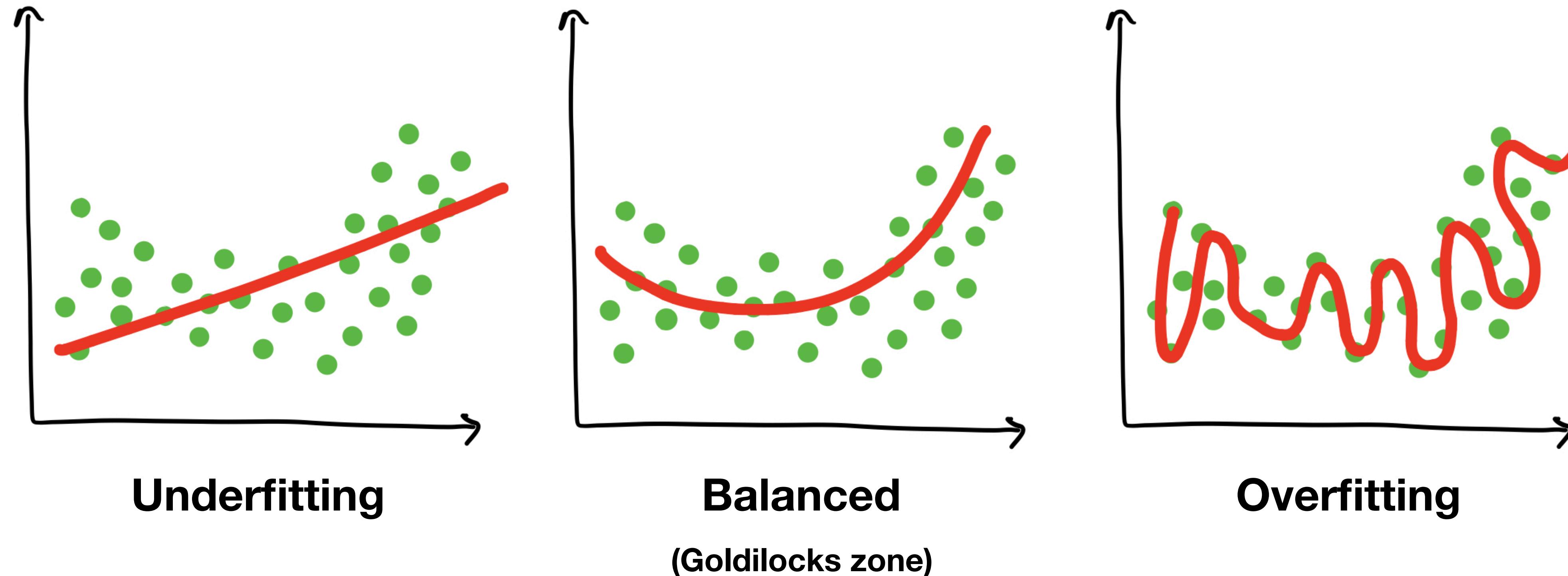
93%

Test

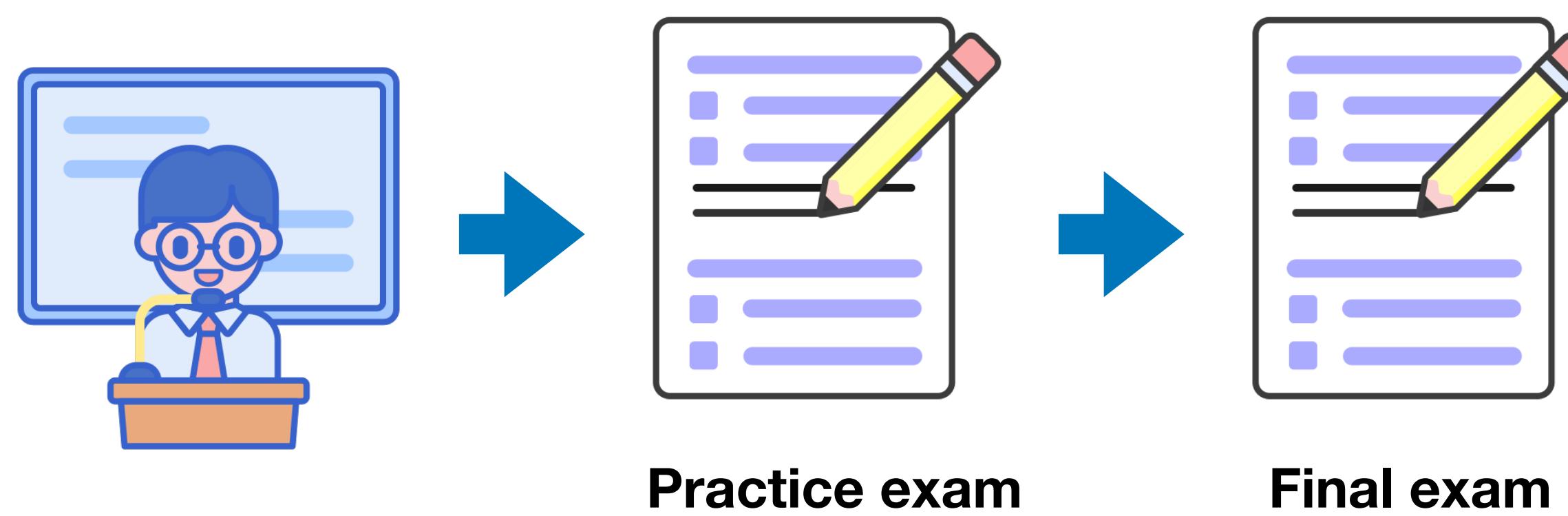
99%



# Overfitting and underfitting



# Overfitting and underfitting



Practice exam

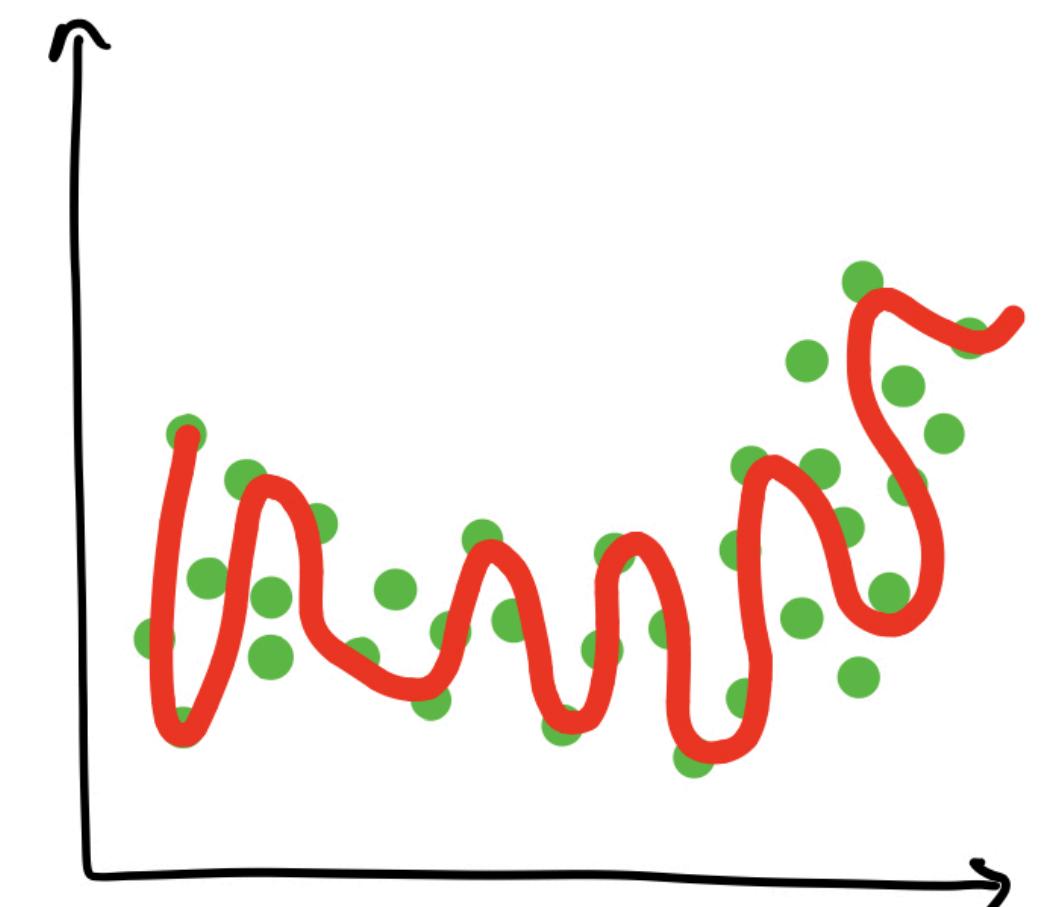
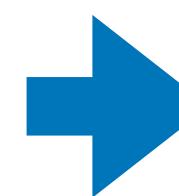
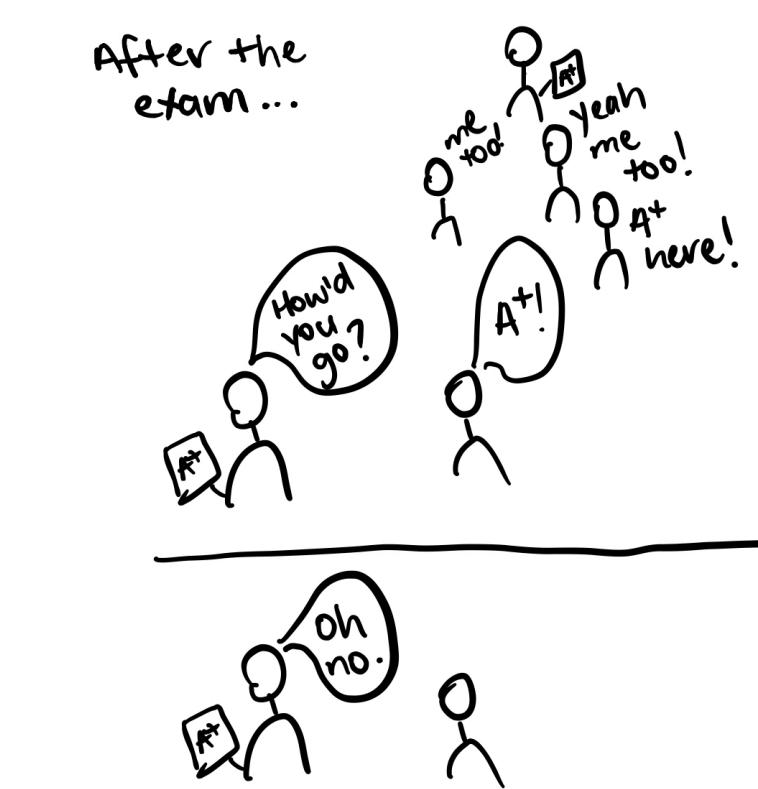
Final exam



Training Data

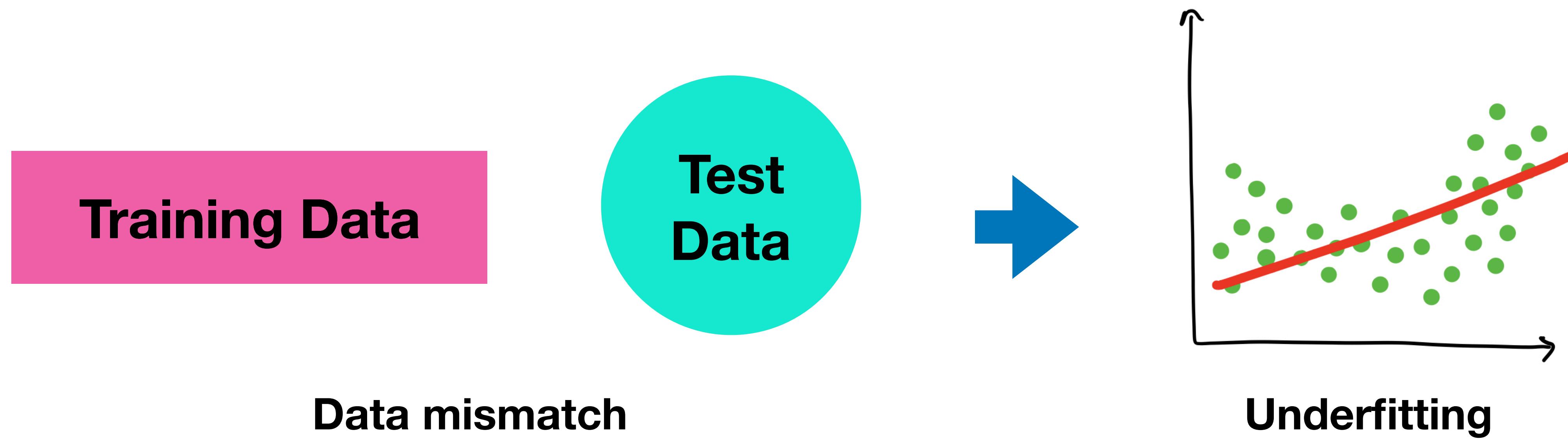
Test Data

Data leakage

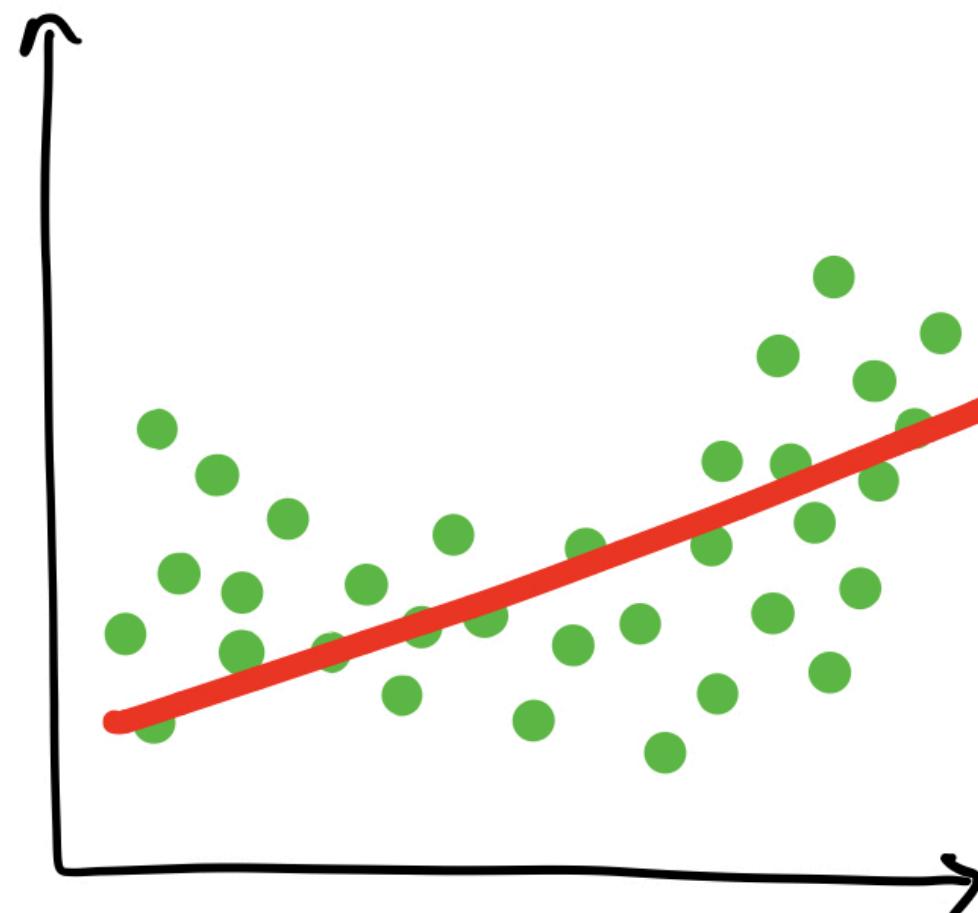


Overfitting

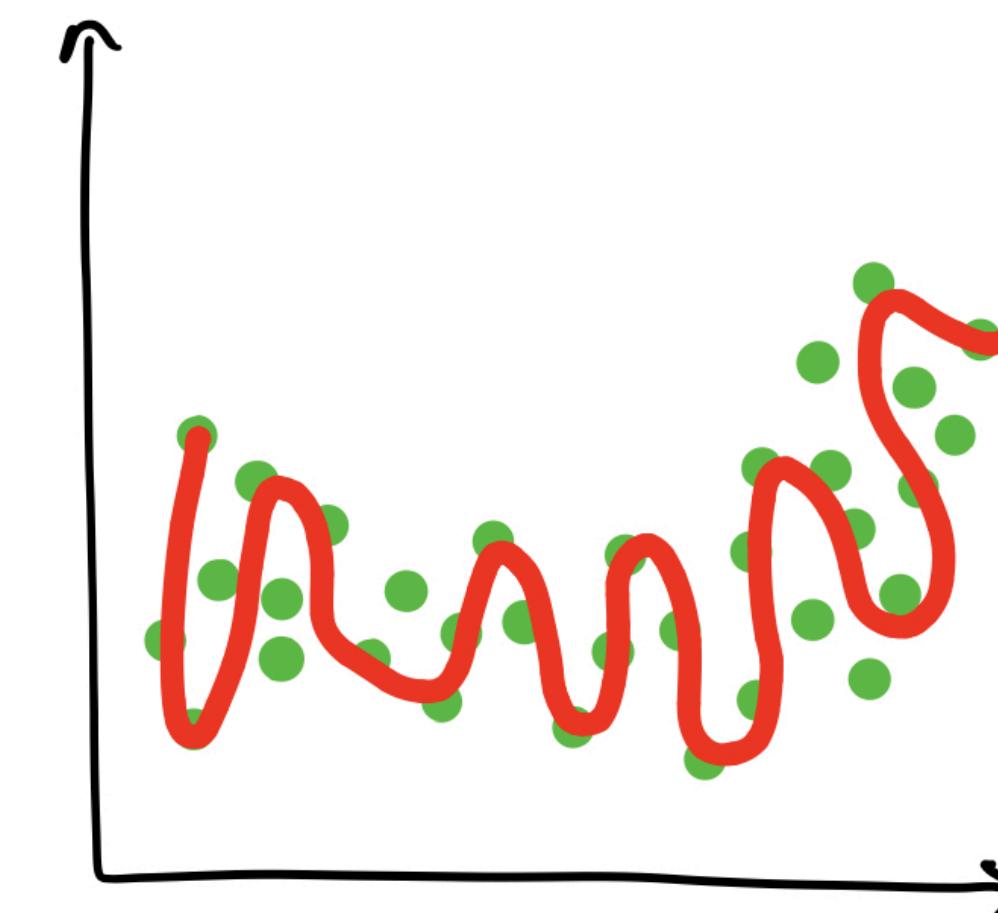
# Overfitting and underfitting



# Fixes for overfitting and underfitting



**Underfitting**



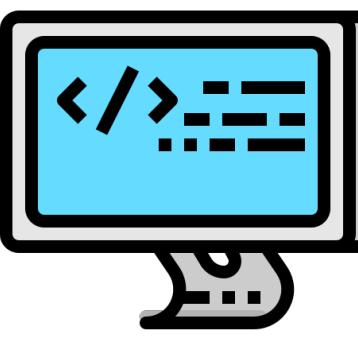
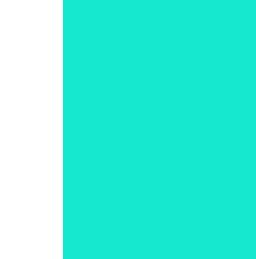
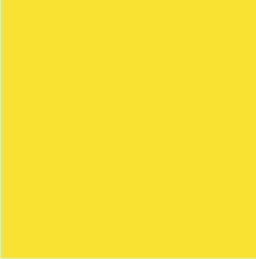
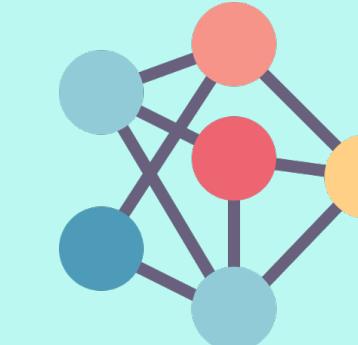
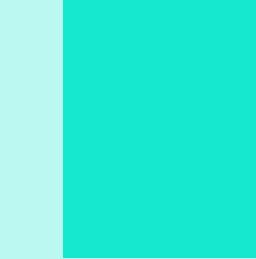
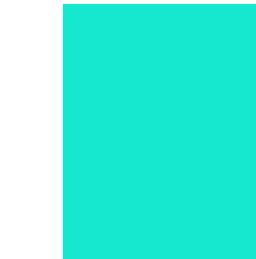
**Overfitting**

- Try a more advanced model
- Increase model hyperparameters
- Reduce amount of features
- Train longer

- Collect more data
- Try a less advanced model

# Comparing models

## Experiment

			Accuracy	Training time	Prediction time	
1	 Inputs	 Model 1	 Outputs	87.5%	3 min	0.5 sec
2	 Inputs	 Model 2	 Outputs	91.3%	92 min	1 sec
3	 Inputs	 Model 3	 Outputs	94.7%	176 min	4 sec

# Things to remember

- Want to avoid overfitting and underfitting (head towards generality)
- Keep the test set separate at all costs
- Compare apples to apples
- One best performance metric does not equal best model

Before we get into the experimentation side of things, it's worth having a little reminder of overfitting and underfitting are.

All experiments should be conducted on different portions of your data.

Training data set Use this set for model training, 7080% of your data is the standard.

Validation/development data set Use this set for model hyperparameter tuning and experimentation evaluation, 1015% of your data is the standard.

Test data set Use this set for model testing and comparison, 1015% of your data is the standard.

These amounts can fluctuate slightly, depending on your problem and the data you have.

Poor performance on training data means the model hasn't learned properly and is underfitting. Try a different model, improve the existing one through hyperparameter or collect more data.

Great performance on the training data but poor performance on test data means your model doesn't generalize well. Your model may be overfitting the training data. Try using a simpler model or making sure your the test data is of the same style your model is training on.

# Up next

Another form of overfitting can come in the form of better performance on test data than training data. This may mean your testing data is leaking into your training data (incorrect data splits) or you've spent too much time optimizing your model for the test set data. Ensure your training and test datasets are kept separate at all times and avoid optimizing a models performance on the test set (use the training and validation sets for model improvement).

Poor performance once deployed (in the real world) means theres a difference in what you trained and tested your model on and what is actually happening. Ensure the data you're using during experimentation matches up with the data you're using in production.

## 6. Experimentation



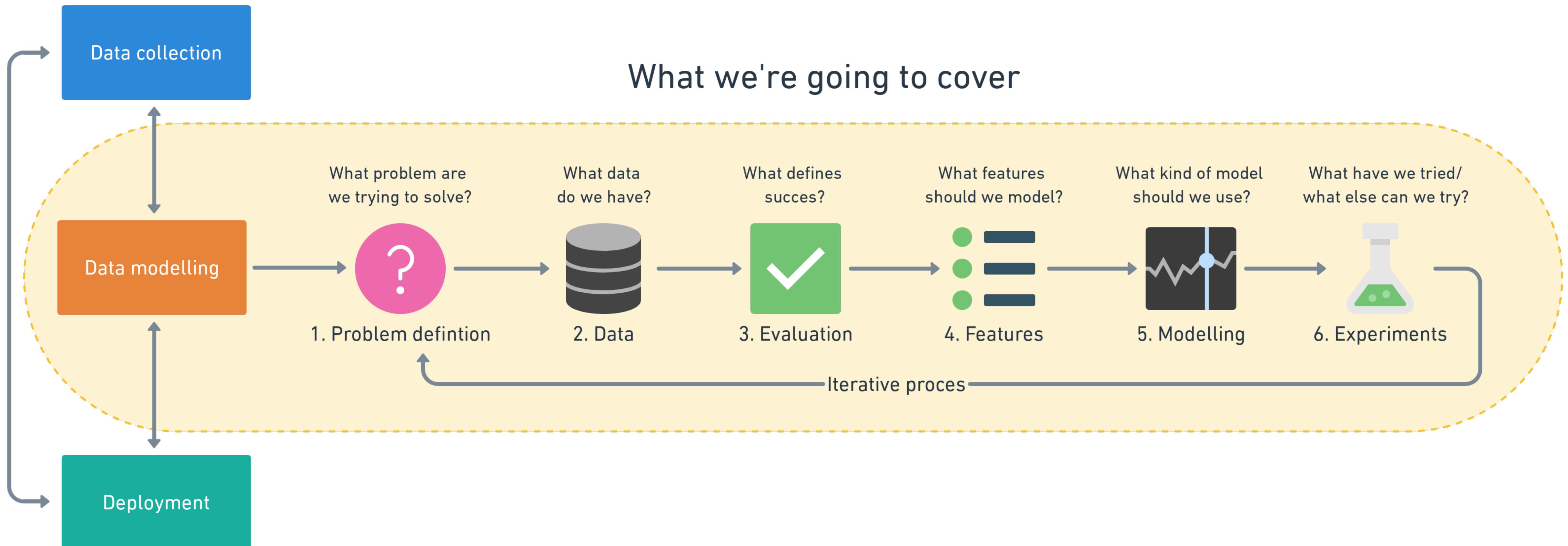
**“How could we improve/what can we try next?”**

# 6. Experimentation



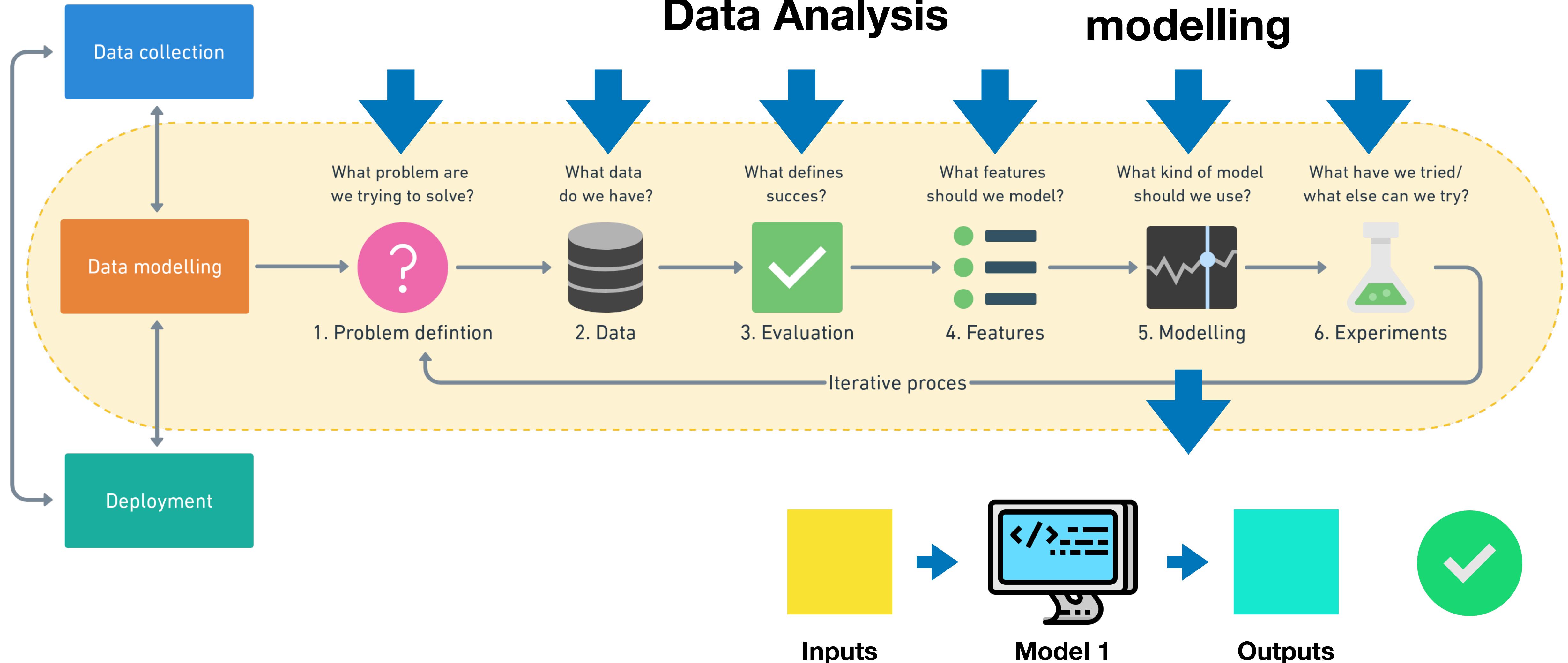
**“How could we improve/what can we try next?”**

## Steps in a full machine learning project



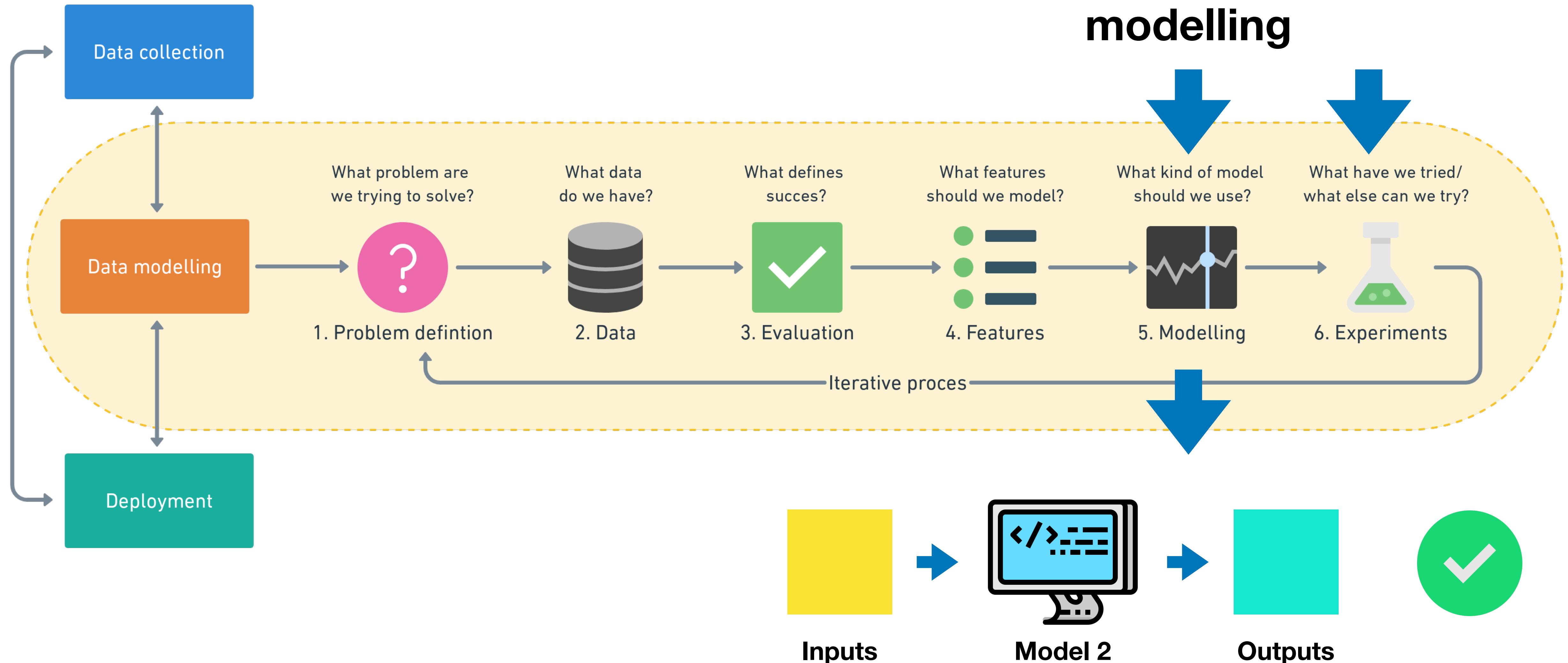
## What we're going to cover

Steps in a full machine learning project



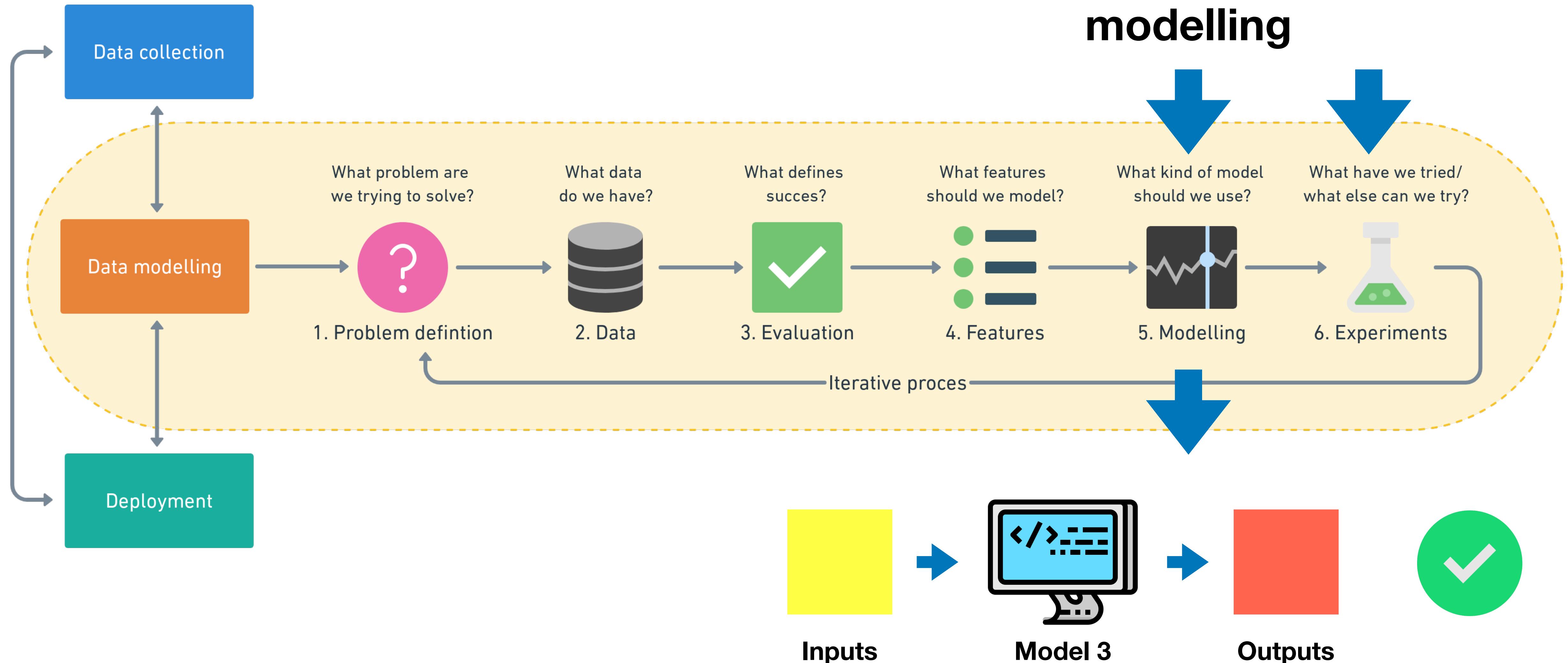
# Machine learning modelling

Steps in a full machine learning project



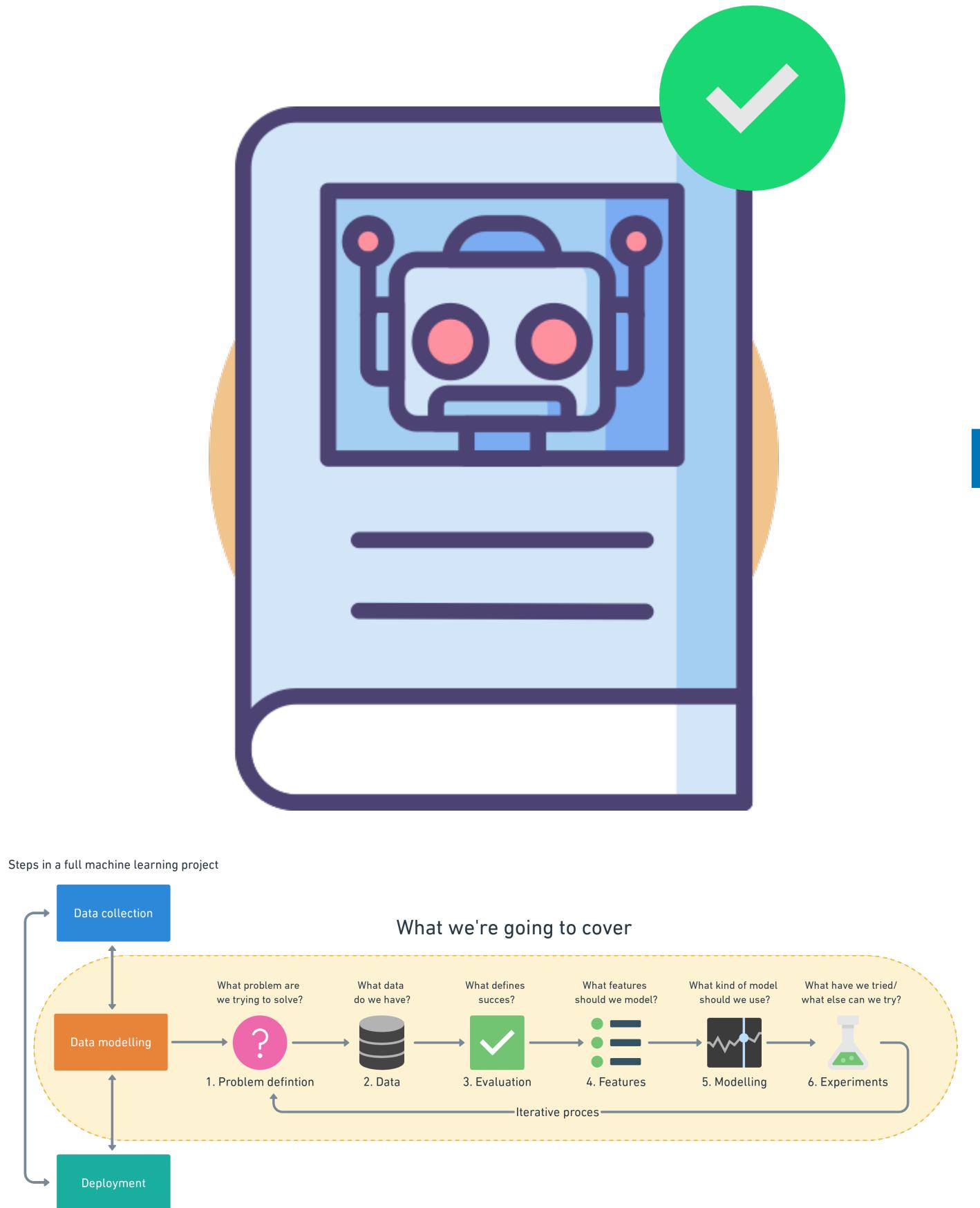
# Machine learning modelling

Steps in a full machine learning project

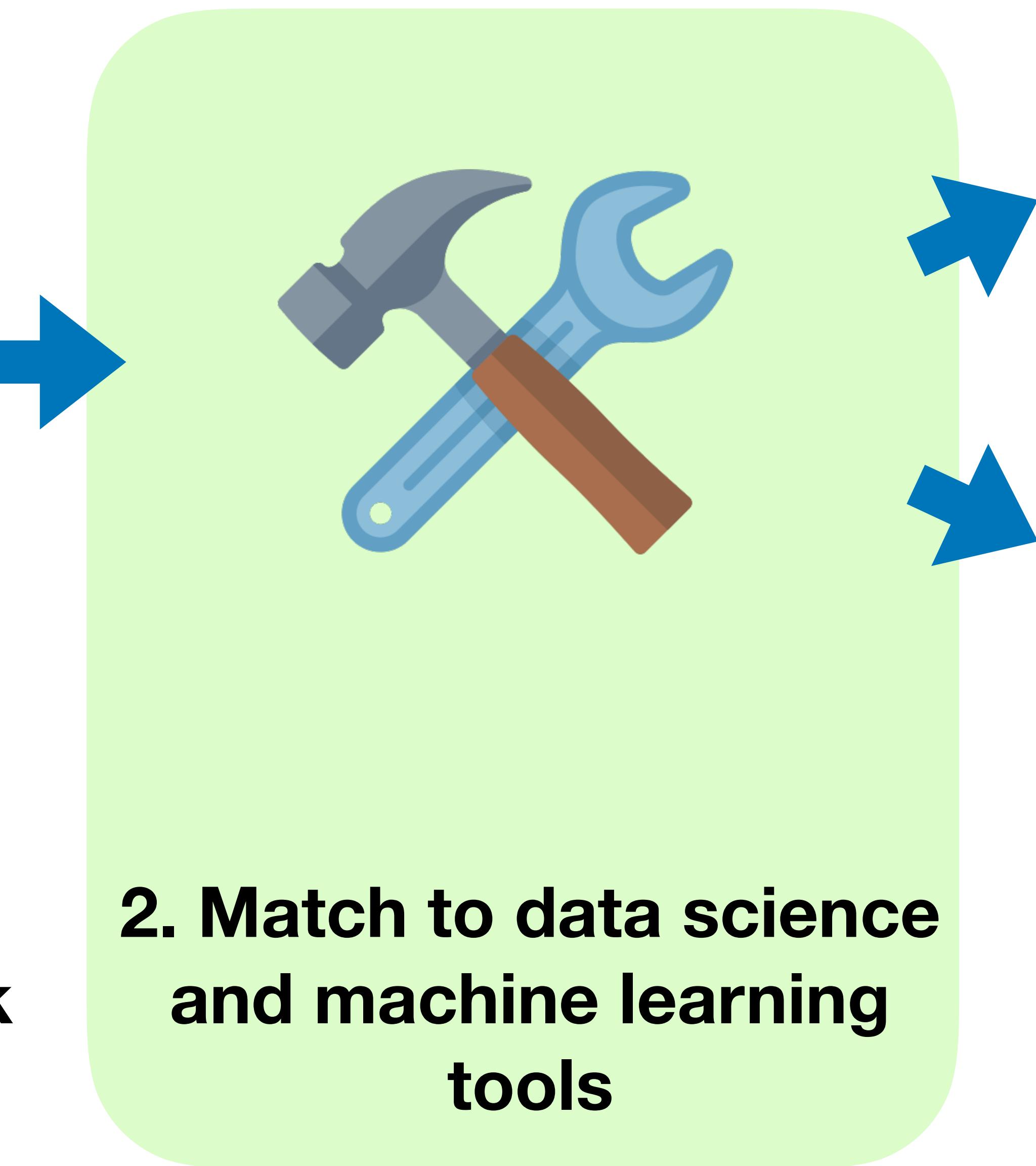


# Up next

## 1. Create a framework



## 2. Match to data science and machine learning tools



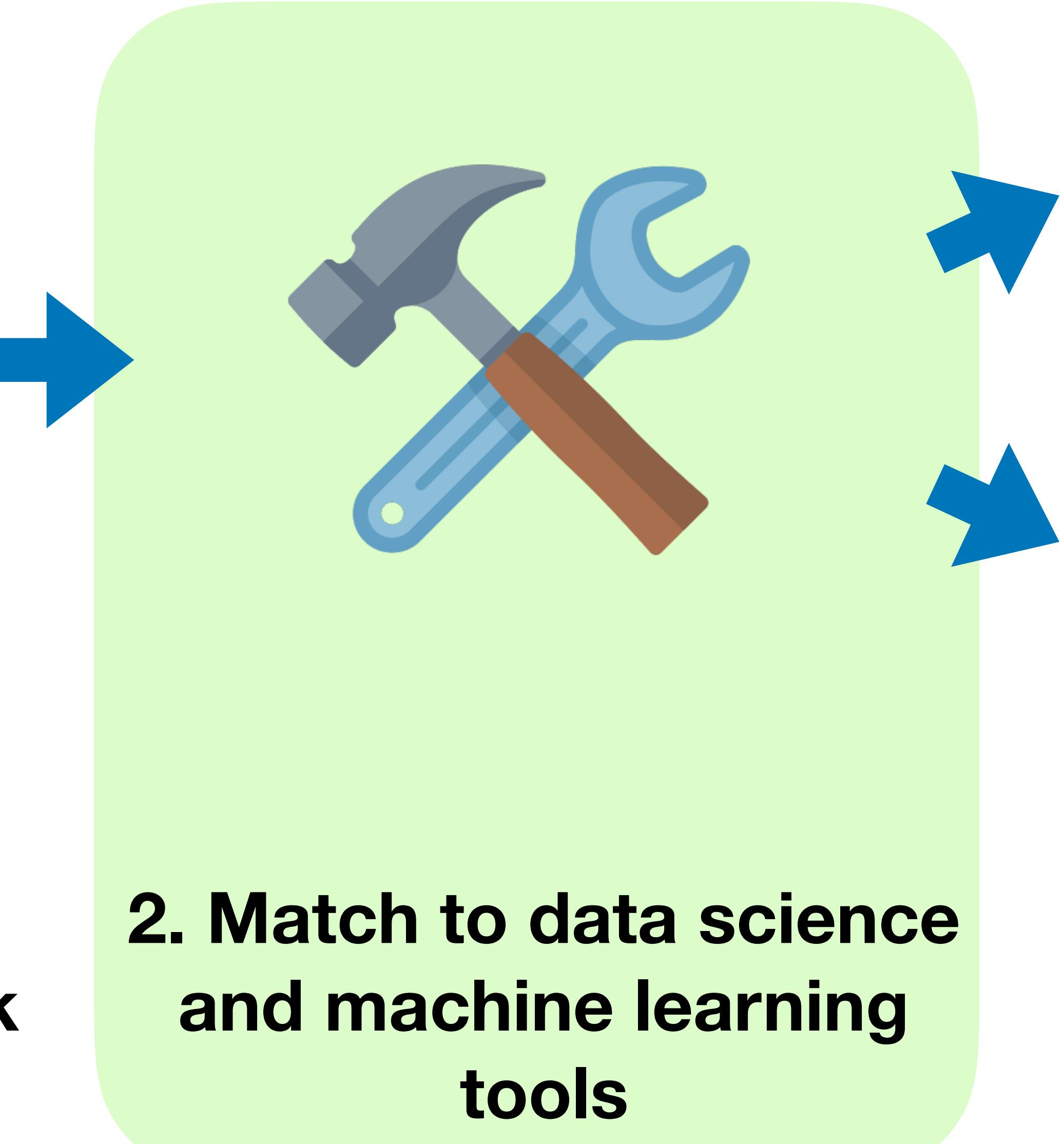
## 3. Learn by doing



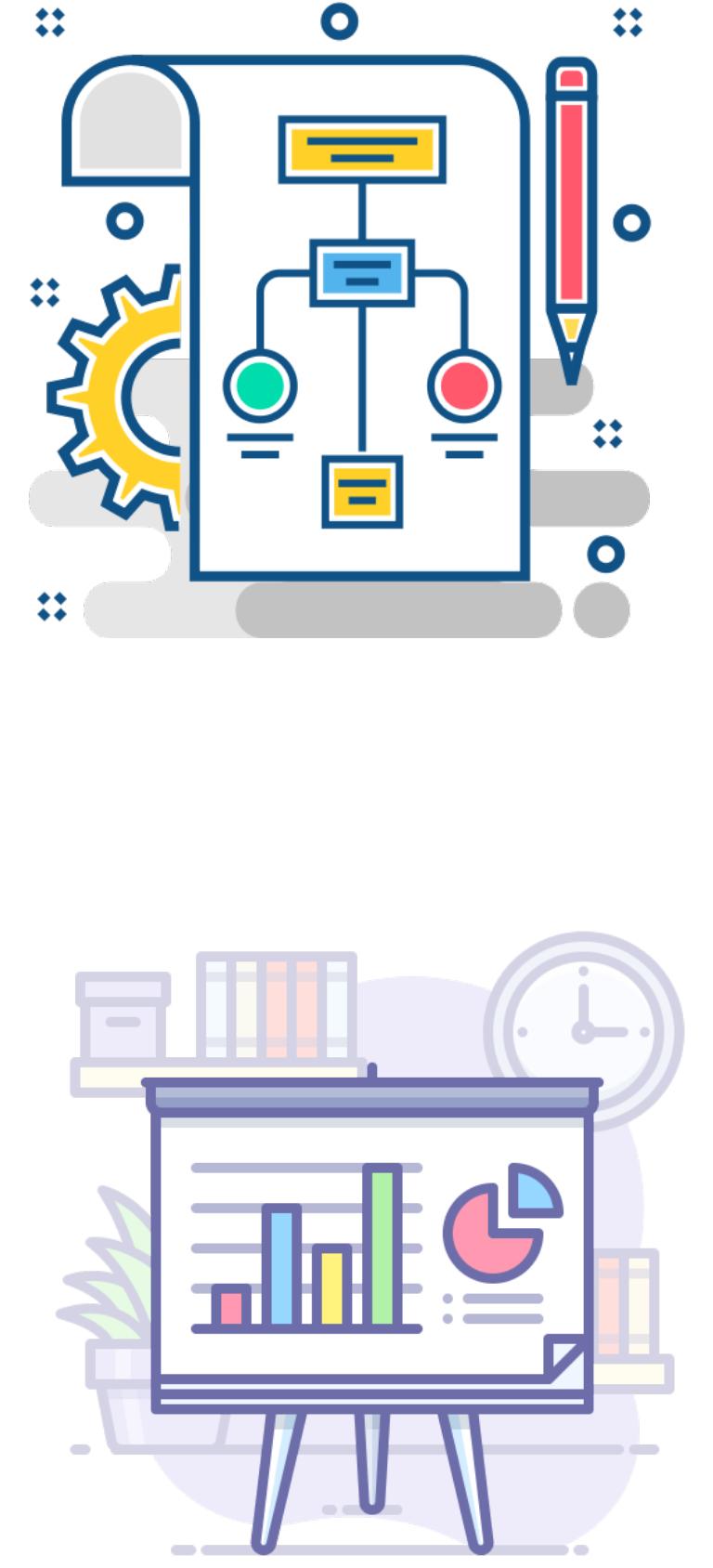
## 1. Create a framework



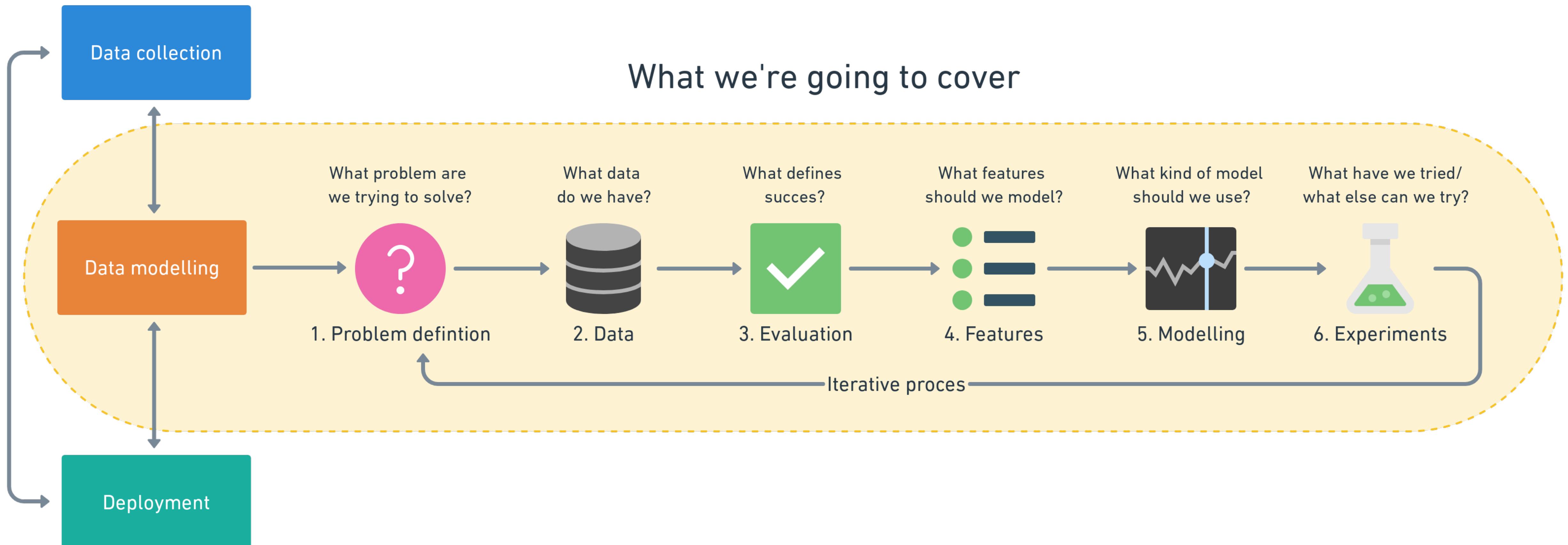
## 2. Match to data science and machine learning tools



## 3. Learn by doing



## Steps in a full machine learning project



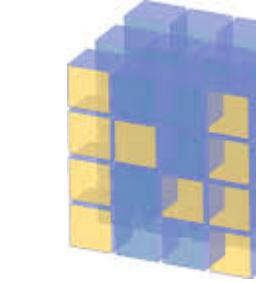
## What we're going to cover



Your  
computer



ANACONDA®

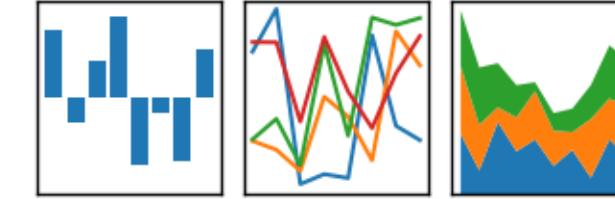


NumPy



matplotlib

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



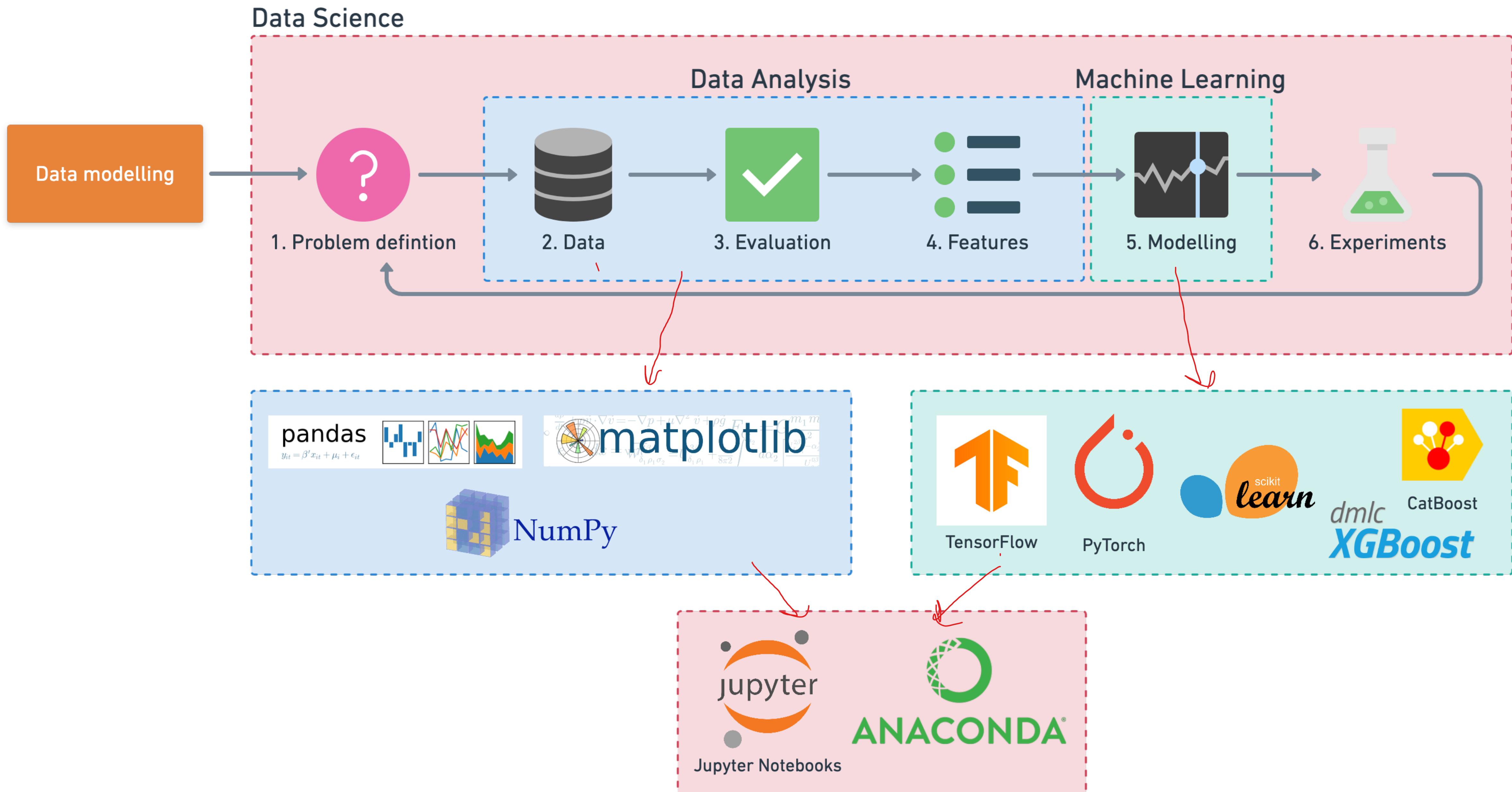
pandas



dmlc  
**XGBoost**



# Tools you can use



# Up next



# ANACONDA®

Massive effort getting through the first section! There was a lot to go through and if you're thinking, "wow this is too much...", don't worry, you're not alone. Machine learning is broad and learning a new skill takes time.

One place you might want to check out for some auxiliary learning whilst you go through the rest of this course is the Elements of AI website. It's got some great introductory explanations of many of the concepts involved in machine learning, data science and artificial intelligence. Of course, going through the Elements of AI material is completely optional. But if you need a break from coding (we'll be doing plenty of it), remember it's always there.

Now you've had an introduction to machine learning, let's get your computer set up for it!

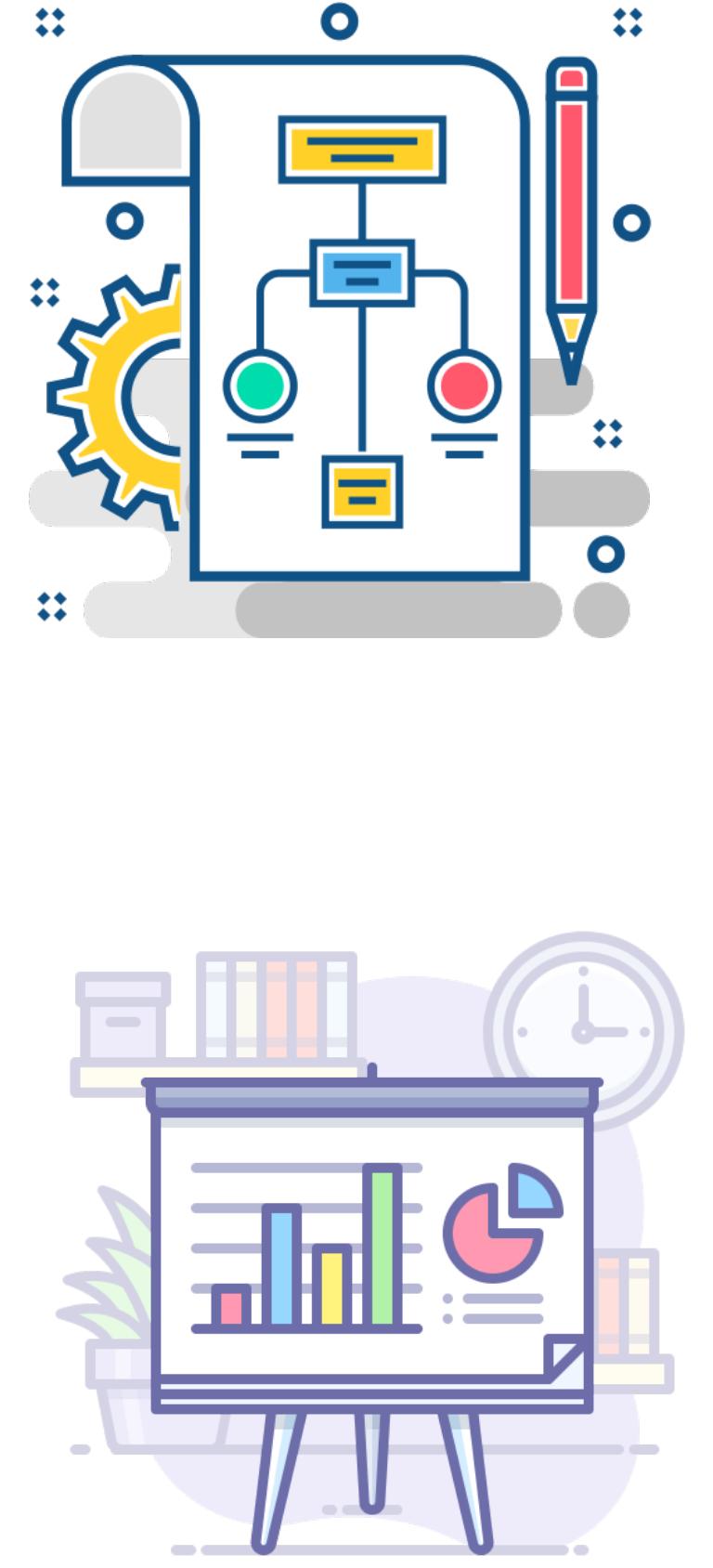
## 1. Create a framework



## 2. Match to data science and machine learning tools



## 3. Learn by doing

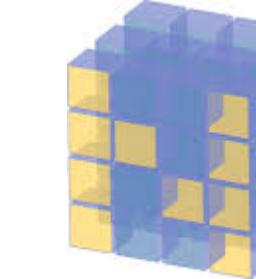




Your  
computer



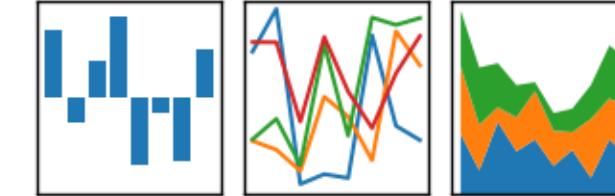
ANACONDA®  
MINICONDA®  
CONDA®



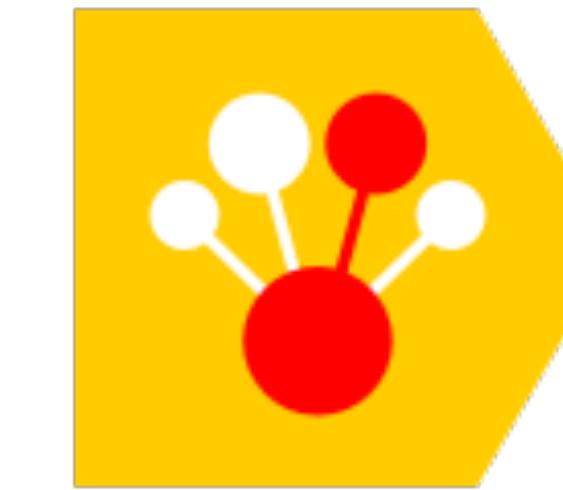
NumPy

$\frac{\partial^2 \vec{v}}{\partial t^2} + \vec{n} \cdot \nabla \vec{v} = -\nabla p + \mu \nabla^2 \vec{v} + \rho \vec{g}$  **matplotlib**  $F = \frac{G m_1 m}{r^2}$

pandas  
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



dmlc  
**XGBoost**





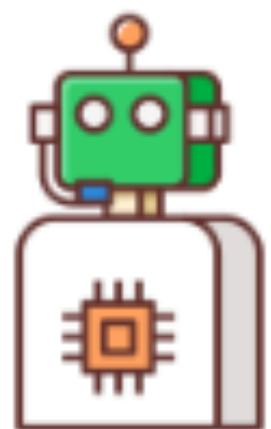
ANACONDA®



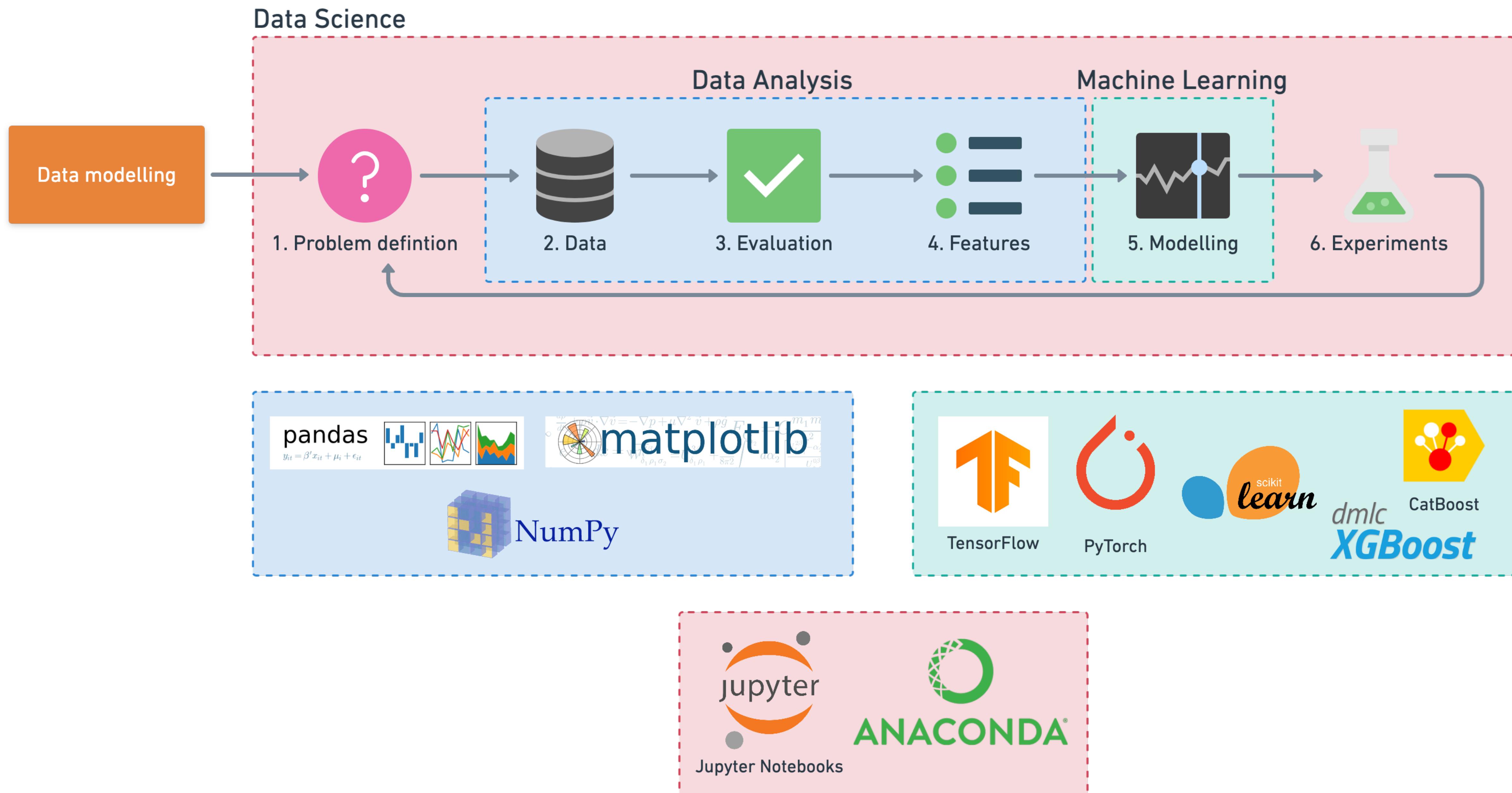
MINICONDA®



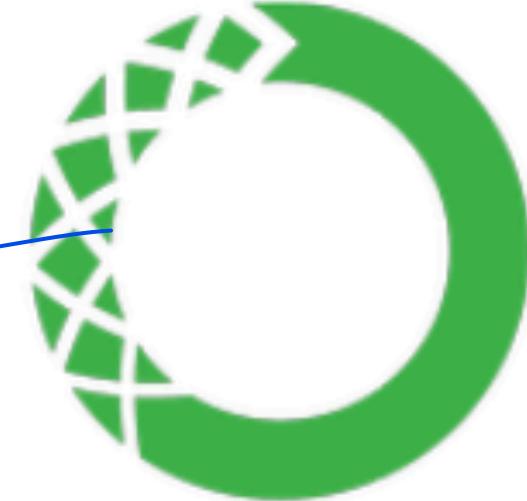
CONDA®



# Tools you can use



Thinks fix for after &  
(recreating env.)  
↓



= ~3 GB

There may come a time where you want to share the contents of your Conda environment.

This could be to share a project workflow with a colleague or with someone else who's trying to set up their system to have access to the same tools as yours.

There are a couple of ways to do this:

1. Share your entire project folder (including the environment folder containing all of your Conda packages).
2. Share a .yml (pronounced YAM-L) file of your Conda environment.

The benefit of 1 is it's a very simple setup, share the folder, activate the environment, run the code. However, an environment folder can be quite a large file to share.

That's where 2 comes in. A .yml is basically a text file with instructions to tell Conda how to set up an environment.

For example, to export the environment we created earlier at /Users/daniel/Desktop/project\_1/env as a YAML file called environment.yml we can use the command:

```
conda env export --prefix /Users/daniel/Desktop/project_1/env > environment.yml
```

After running the export command, we can see our new .yml file stored as environment.yml.

A sample .yml file might look like the following:

```
name: my_ml_env
```

```
dependencies:
```

- numpy
- pandas
- scikit-learn
- jupyter
- matplotlib

Of course, your actual file will depend on the packages you've installed in your environment.

For more on sharing an environment, check out the Conda documentation on sharing environments.

Finally, to create an environment called env\_from\_file from a .yml file called environment.yml, you can run the command:

```
conda env create --file environment.yml --name env_from_file
```

For more on creating an environment from a .yml file, check out the Conda documentation on creating an environment from a .yml file.



= ~3 GB



= ~200 MB

# Up next

CONDA®





**ANACONDA®**



**Software Distributions**

**MINICONDA®**



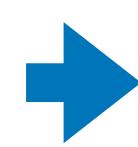
**CONDA®**



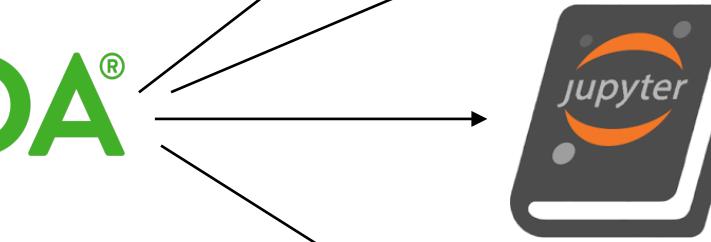
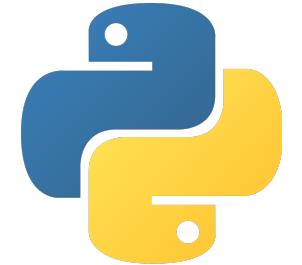
**Package Manager**



Your  
computer



**MINICONDA® + CONDA®**

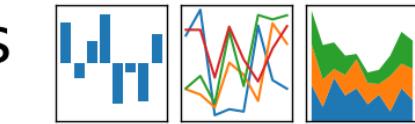


$\frac{\partial^2 \psi}{\partial r^2} - \frac{1}{r^2} \nabla^2 \psi = -\nabla p + \mu \nabla^2 \vec{v} + \rho \vec{g}$

**matplotlib**

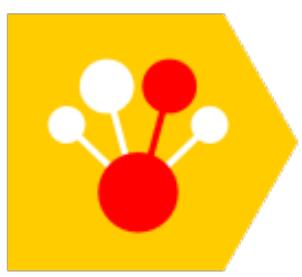
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

pandas



scikit  
*learn*

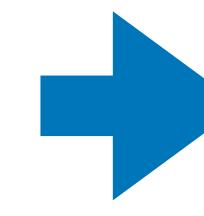
NumPy



dmlc  
**XGBoost**



# Up next



## Project 1





Your  
computer

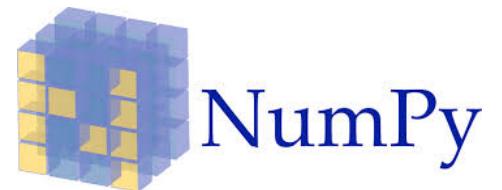


**MINICONDA® + CONDA®**



$\frac{\partial^2}{\partial r^2} \vec{v} \cdot \nabla \vec{v} = -\nabla p + \mu \nabla^2 \vec{v} + \rho \vec{g}$

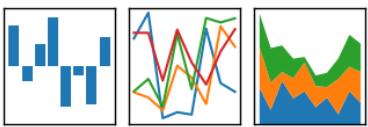
**matplotlib**



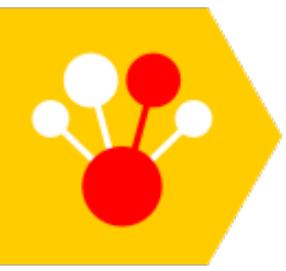
**NumPy**

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

**pandas**



**scikit-learn**



**dmlc**  
**XGBoost**



Steps to take in new project :

Start new  
Project

Create new project  
folder

Store all relevant  
files (environment,  
data, notebooks)  
in the project folder

# Anaconda :

Download Anaconda and install it on your computer

Test the installation via terminal ( conda list)

Load up a jupyter notebook

import pandas, Numpy, Matplotlib, sklearn

Heart disease?



Your computer

## Project Folder

ID	Weight	Sex	Blood Pressure	Chest pain	Heart disease?
4326	110kg	M	120 / 80	4	Yes
5681	64kg	F	130 / 90	1	No
7911	81kg	M	130 / 80	0	No

Table 1.0 : Patient records

## Data

CONDA®



matplotlib



pandas



NumPy

scikit  
learn

## Environment

conda create --prefix ./env pandas numpy matplotlib scikit-learn

using this command

Create

a custom

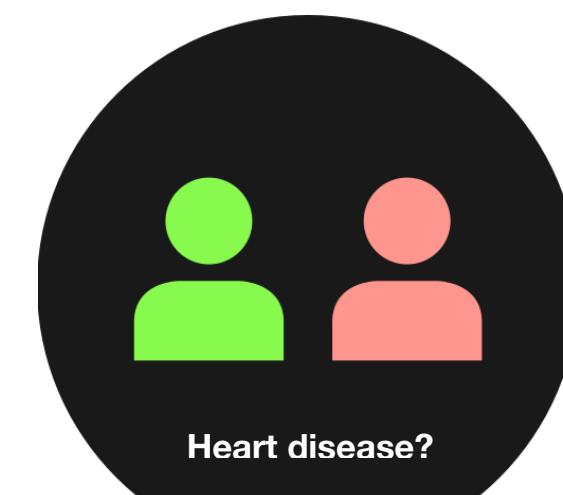
environment

- ent

using  
CONDA

within  
the project  
folder!

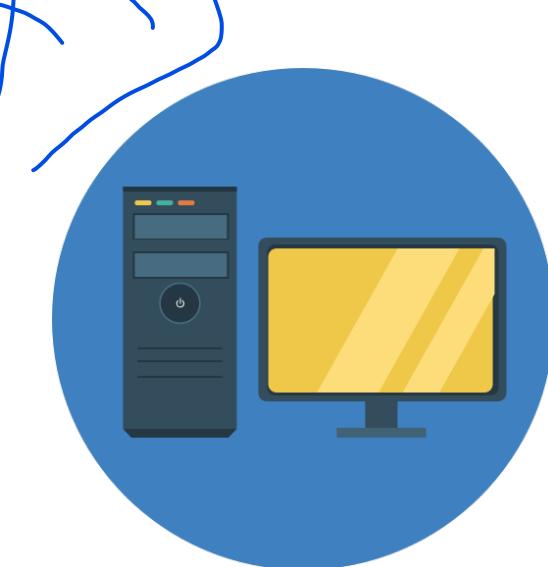
↑



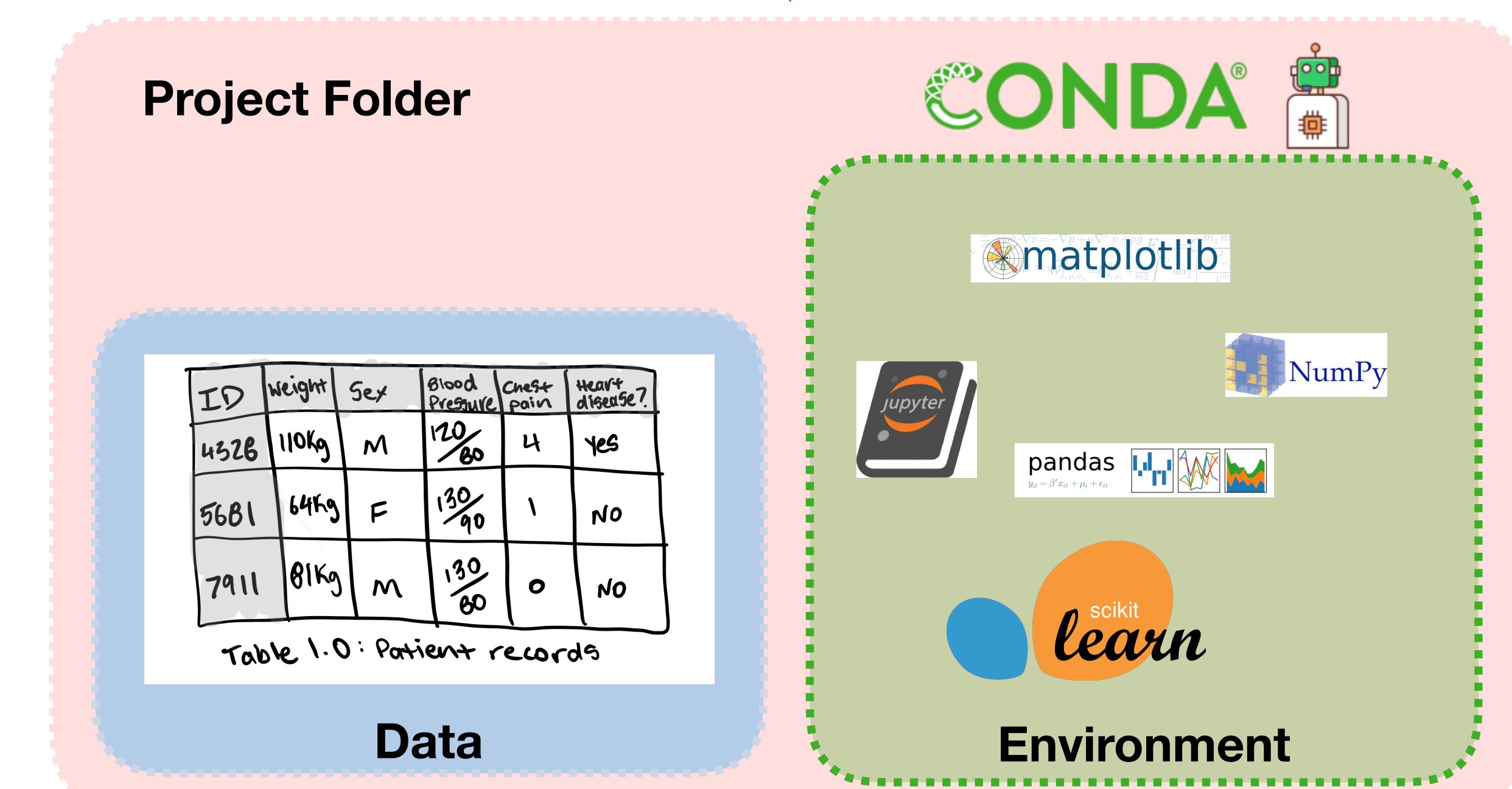
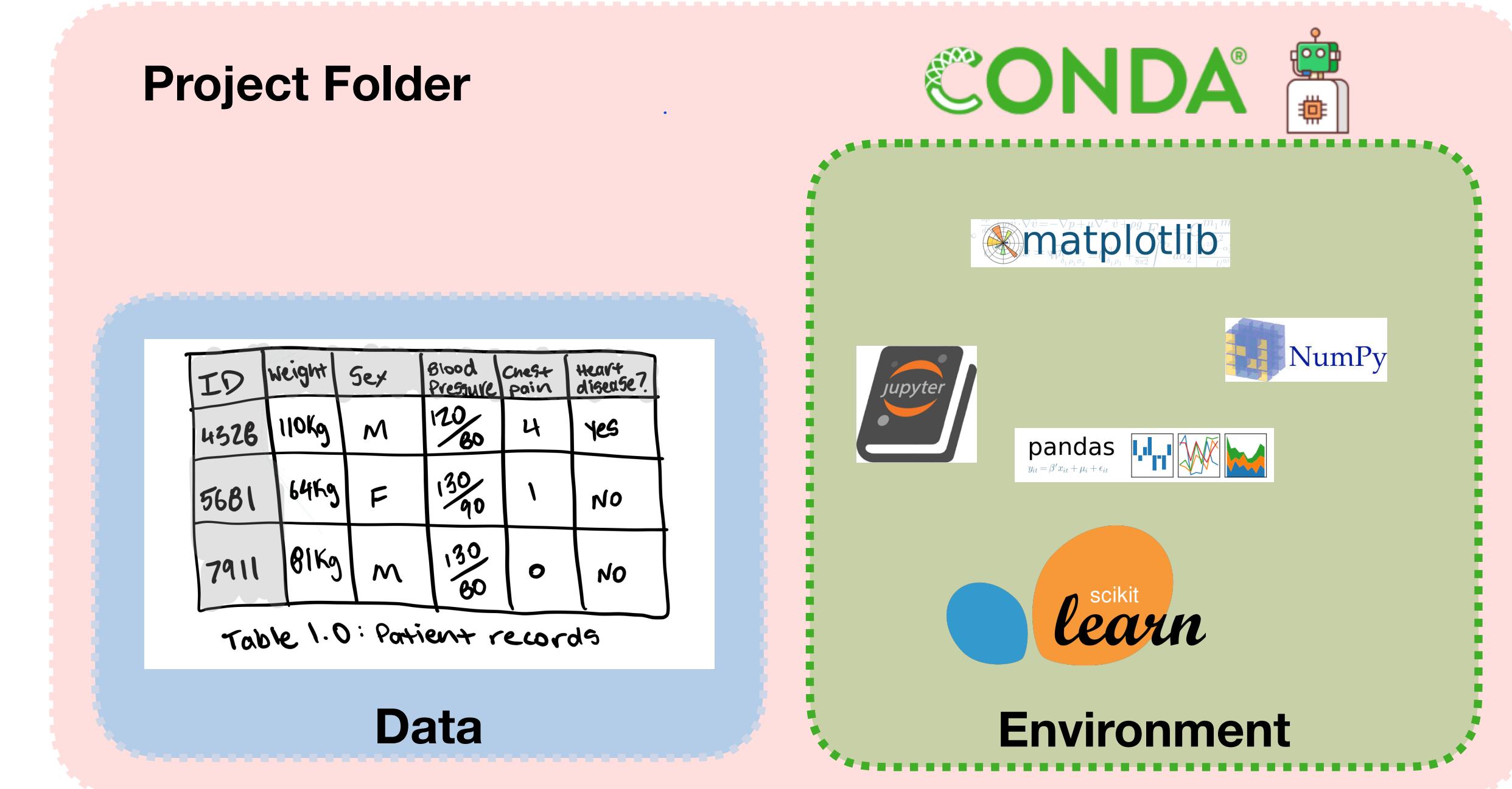
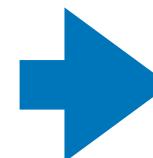
All you have to do now is share the sample\_project\_1 folder with your team since environment is already setup and can run anywhere!

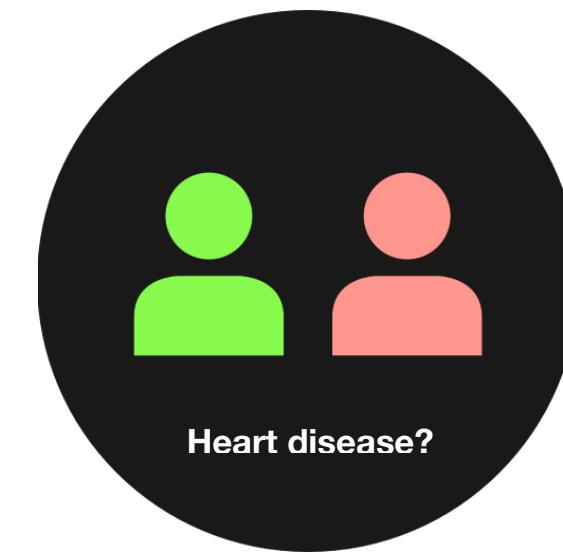


Your computer



Someone else's computer





Your  
computer

## Project Folder

ID	Weight	Sex	Blood Pressure	Chest pain	heart disease?
4326	110kg	M	120 / 80	4	yes
5681	64kg	F	130 / 90	1	no
7911	81kg	M	130 / 80	0	no

Table 1.0 : Patient records

Data

CONDA®



Environment

## Project Folder

ID	Weight	Sex	Blood Pressure	Chest pain	heart disease?
4326	110kg	M	120 / 80	4	yes
5681	64kg	F	130 / 90	1	no
7911	81kg	M	130 / 80	0	no

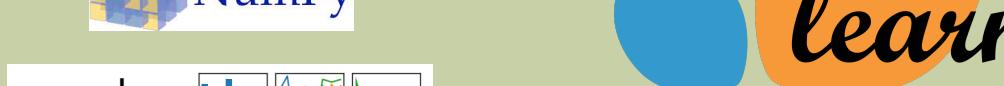
Table 1.0 : Patient records

Data

CONDA®

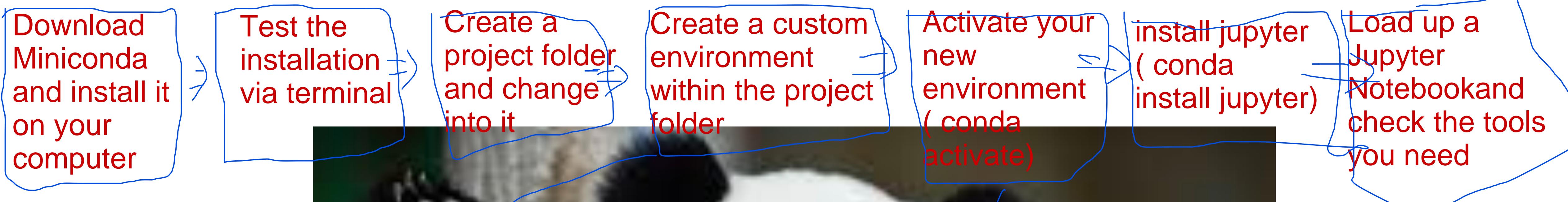


Workspace



Environment

MINI CONDA



conda create --prefix ./env pandas numpy matplotlib scikit-learn

#To activate this environment, use  
conda activate C:\Users\yaduv\OneDrive\Desktop\sample\_project\env

that is basically  
the path of  
env,

# To deactivate this environment (because of doing work on another project or bcz of some other reasons)  
conda deactivate

we can activate again using above activate command if we

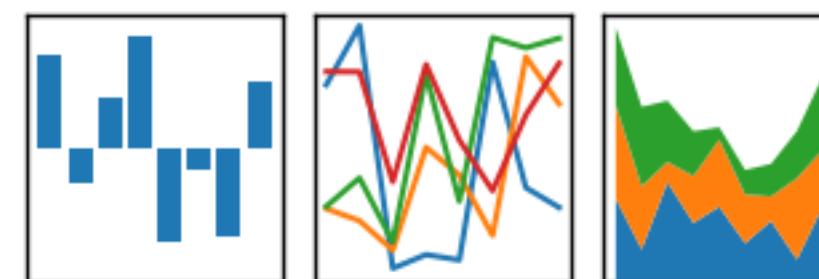
to back again to the same project,

import numpy  
import sklearn  
import pandas  
import matplotlib.pyplot

we can use  
these commands  
in jupyter  
notebook

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Data

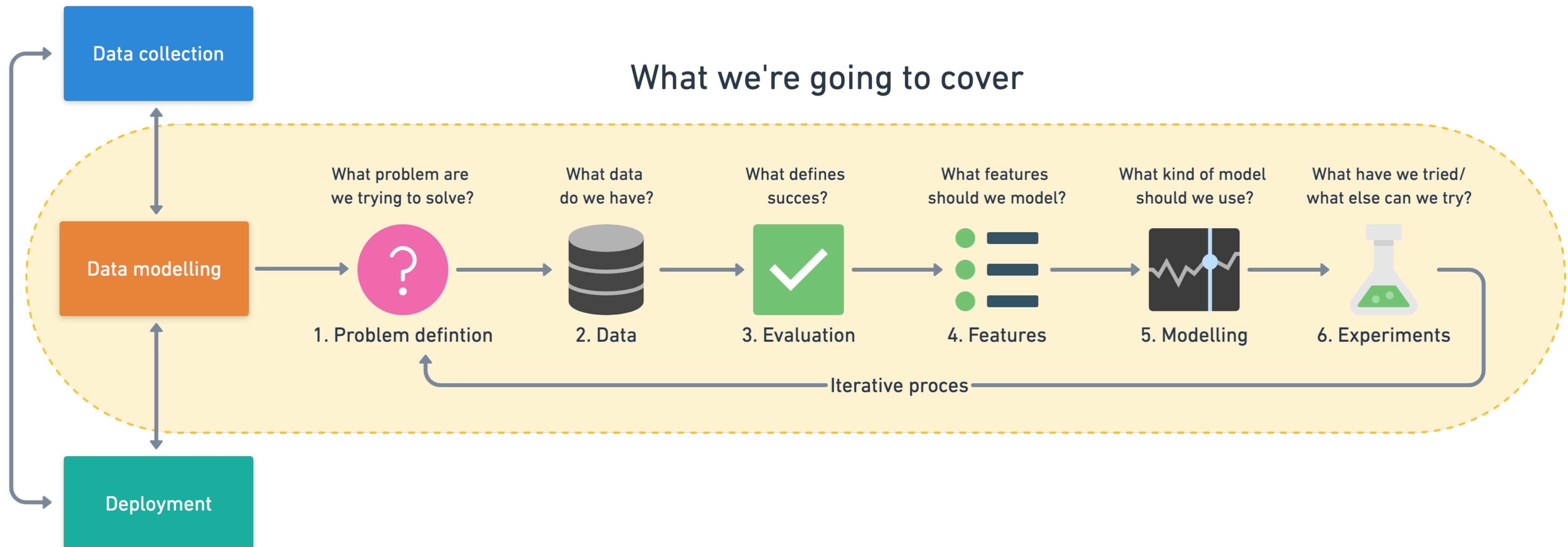


it's used to explore data, analyze data, manipulate data, get it ready for machine learning.

ctrl + c is for  
shutting down  
the jupyter  
notebook

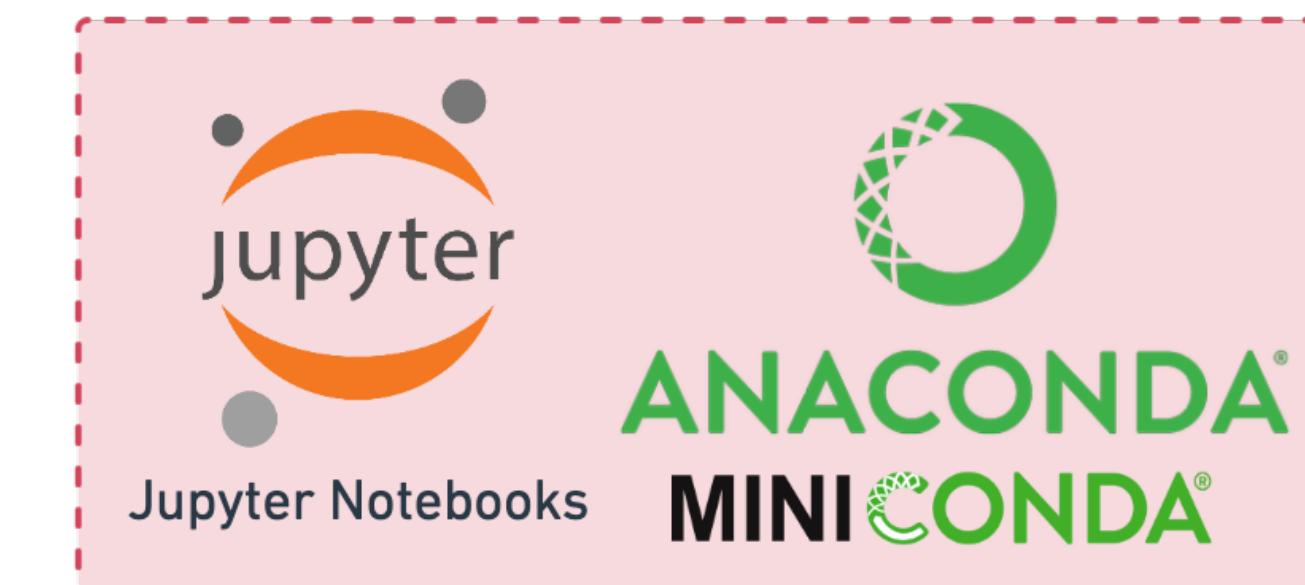
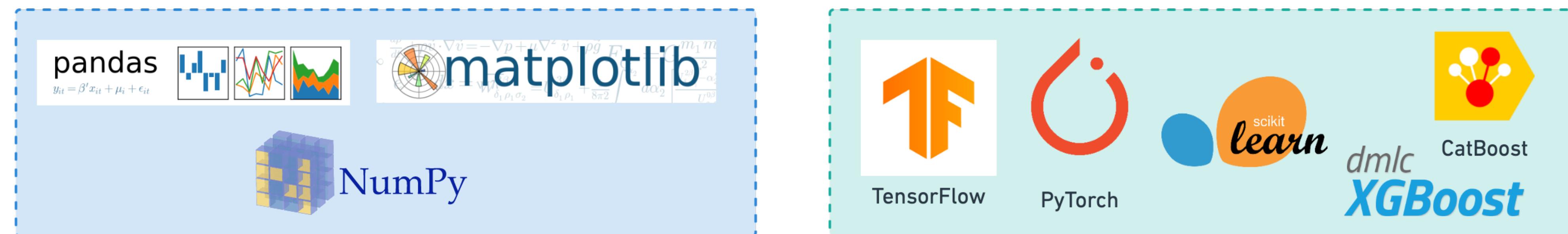
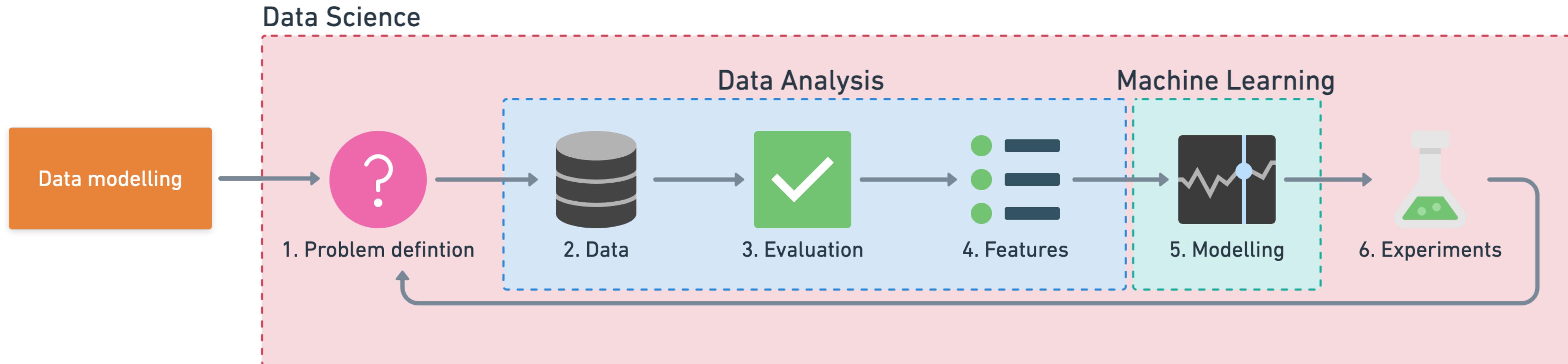
in working  
terminal

## Steps in a full machine learning project



## What we're going to cover

# Tools you can use



# Why pandas?

- Simple to use
- Integrated with many other data science & ML Python tools
- Helps you get your data ready for machine learning

# What are we going to cover?

- Most useful functions
- pandas Datatypes
- Importing & exporting data
- Describing data
- Viewing & selecting data
- Manipulating data

# Where can you get help?

- Follow along with the code

Try it for yourself

- Search for it

Try again

- Ask



A screenshot of a Jupyter Notebook interface. The title bar says "introduction-to-pandas - Jupyter". The main content area shows a section titled "1. Datatypes" with text about Series and DataFrame. Below that is a code cell with Python code to create a Series and its output. The output cell shows a Pandas Series object with three items: BMW, Toyota, and Honda.

```
In [6]: # Creating a series of car types
cars = pd.Series(["BMW", "Toyota", "Honda"])
cars
```

```
Out[6]: 0      BMW
1      Toyota
2      Honda
dtype: object
```



A screenshot of the pandas documentation website at [pandas.pydata.org/pandas-docs/stable/](https://pandas.pydata.org/pandas-docs/stable/). The page title is "pandas 0.25.2 documentation". It features a sidebar with a "Table Of Contents" and a main content area with the heading "pandas: powerful Python data analysis toolkit". The page also includes links for "What's New in 0.25.2", "Installation", "Getting started", "User Guide", "Pandas ecosystem", "API reference", "Development", and "Release Notes".

In some of the lectures, you'll notice .csv files being imported from file using something like:

```
heart_disease = pd.read_csv("data/heart-disease.csv")
```

This is helpful if you have the data downloaded to your computer or in the same directory as your notebook.

But if you don't, another great feature of pandas is being able to import .csv files directly from a URL.

For example, for the heart-disease.csv file, using the `read_csv()` function you can directly import it using the URL from the course GitHub repo:

```
heart_disease = pd.read_csv("
```

```
https://raw.githubusercontent.com/mrdbourke/zero-to-mastery-ml/master/data/heart-disease.csv")
```

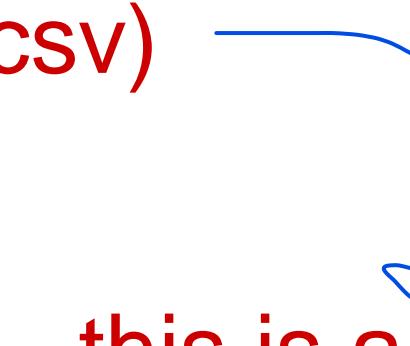
Note: If you're using a link from GitHub, make sure it's in the "raw" format, by clicking the raw button.

# Let's code!

Comma-separated value (.csv)

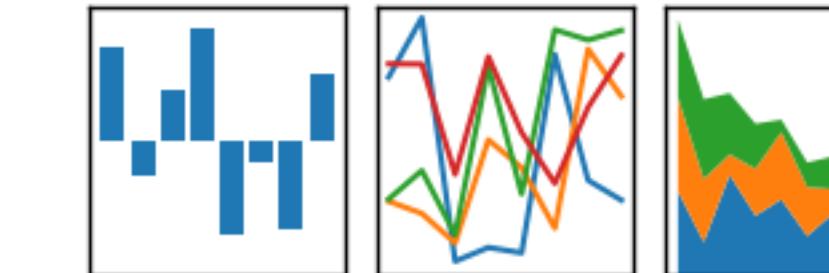
Microsoft Excel (.xlsx)

Web page(.html)



pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



this is a very common data storage file type and Pandas works beautifully with it.

Remember, the main difference between a function and an attribute is one doesn't have Brackets, attributes don't have brackets and functions, are going to perform some kind of operation.

# Anatomy of a DataFrame

for Columns

Column (axis = 1)

Index number (starts at 0 by default)

for Rows

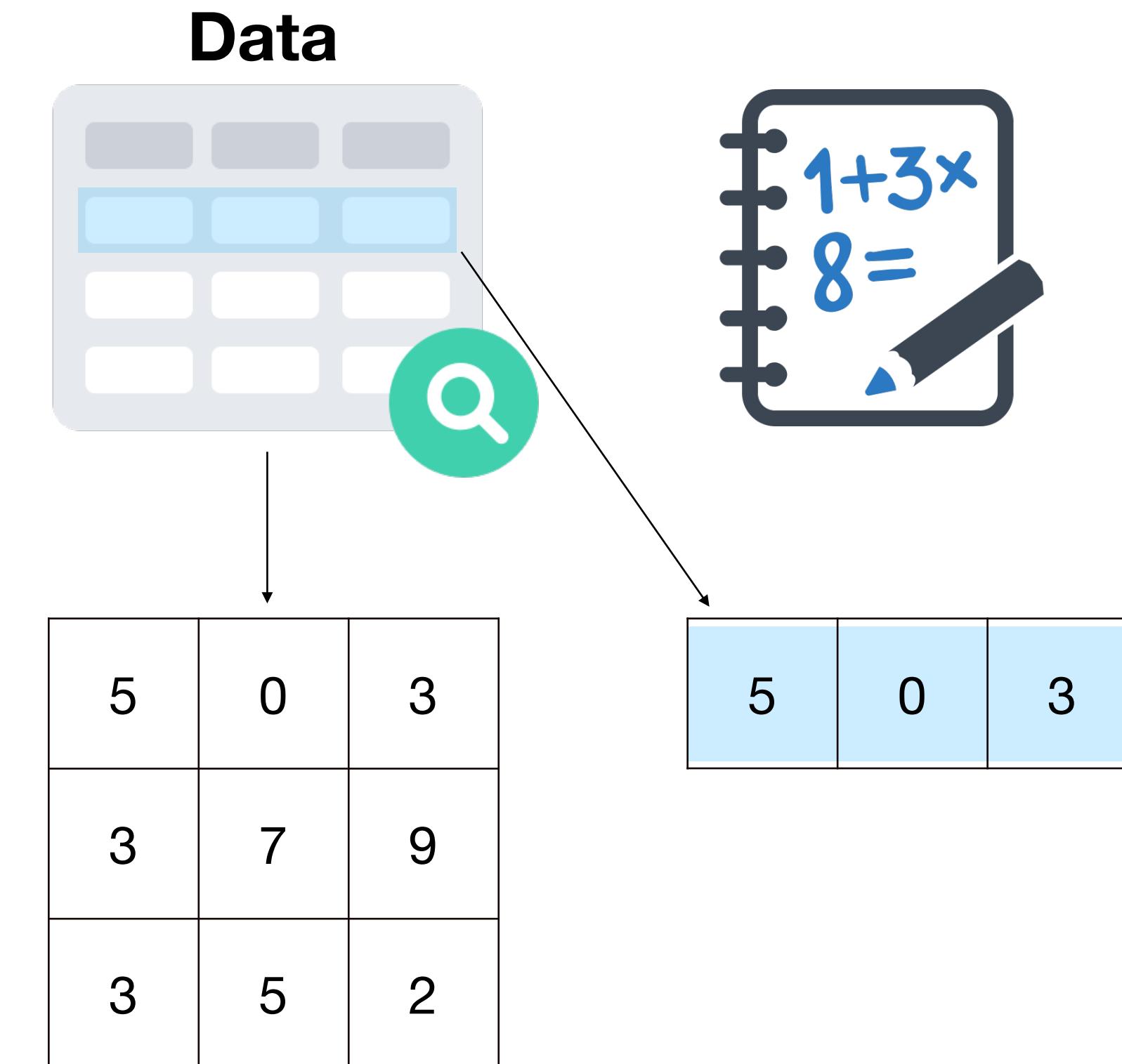
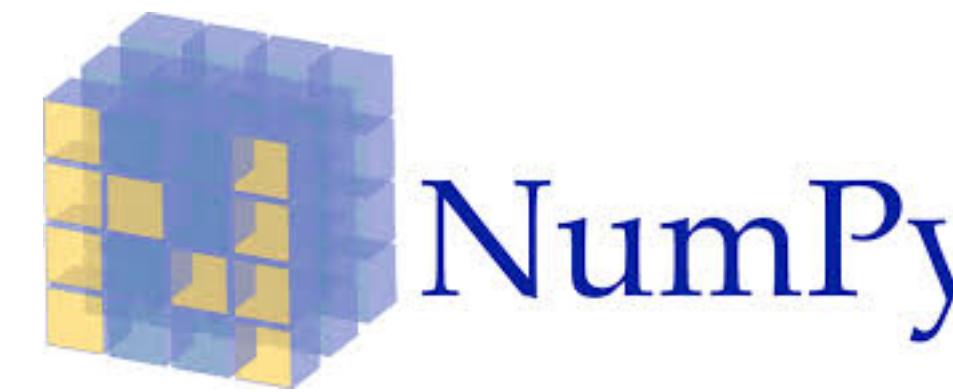
Row (axis = 0)

	Make	Colour	Odometer	Doors	Price	Column name
0	Toyota	White	150043	4	\$4,000	
1	Honda	Red	87899	4	\$5,000	
2	Toyota	Blue	32549	3	\$7,000	
3	BMW	Black	11179	5	\$22,000	
4	Nissan	White	213095	4	\$3,500	

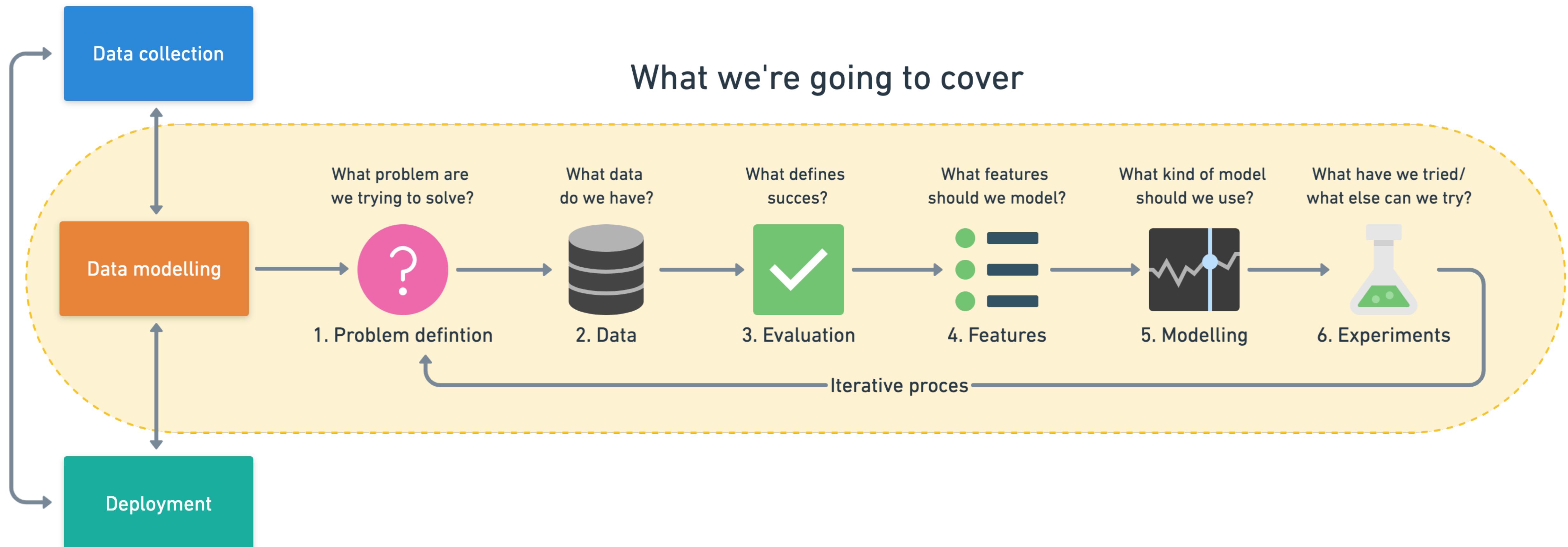
remember, in Python lists, data frames and series start from zero

many functionalities that NumPy offers us are actually written in C a programming language that is really, really fast. If we used Python lists, it would actually be a lot slower to do some of the things that we're about to do.

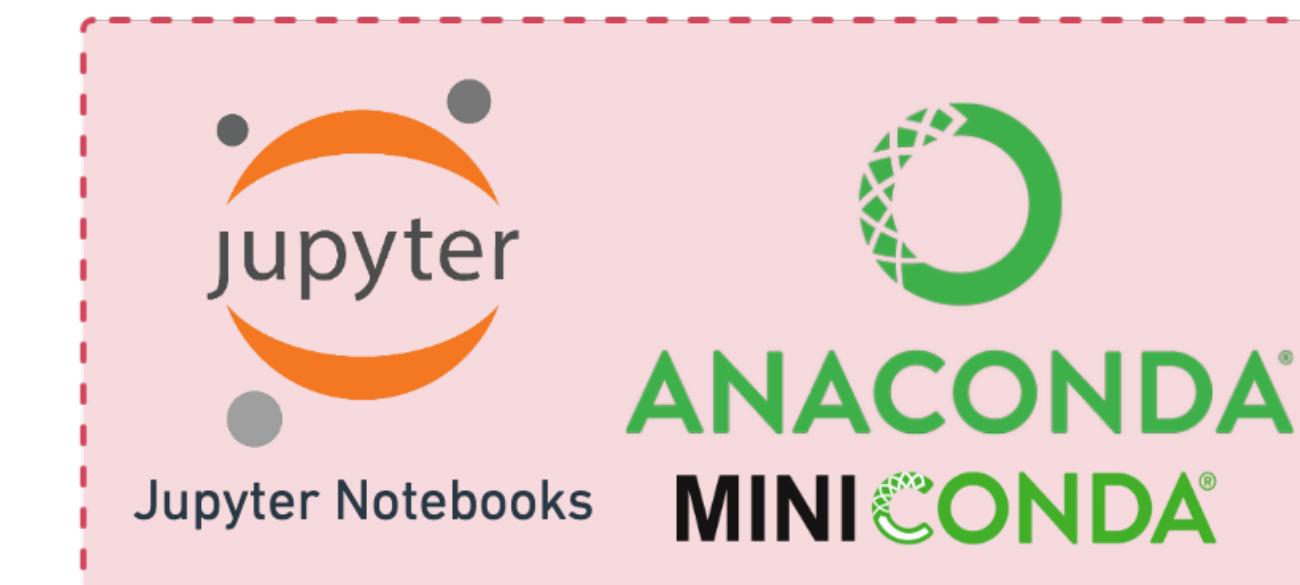
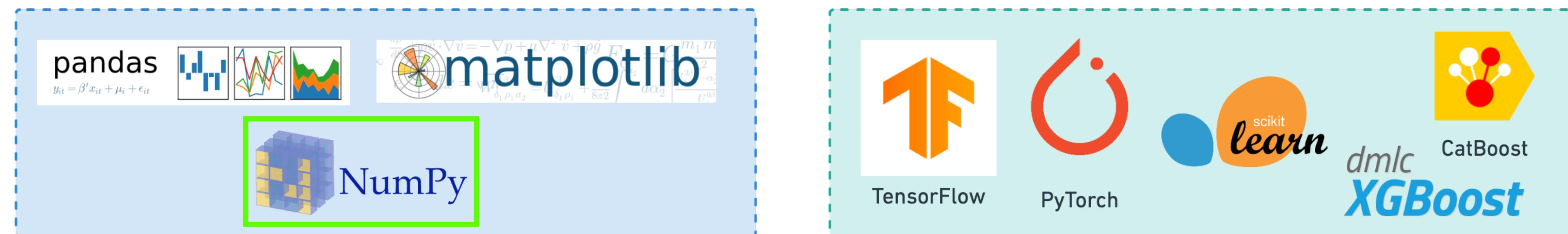
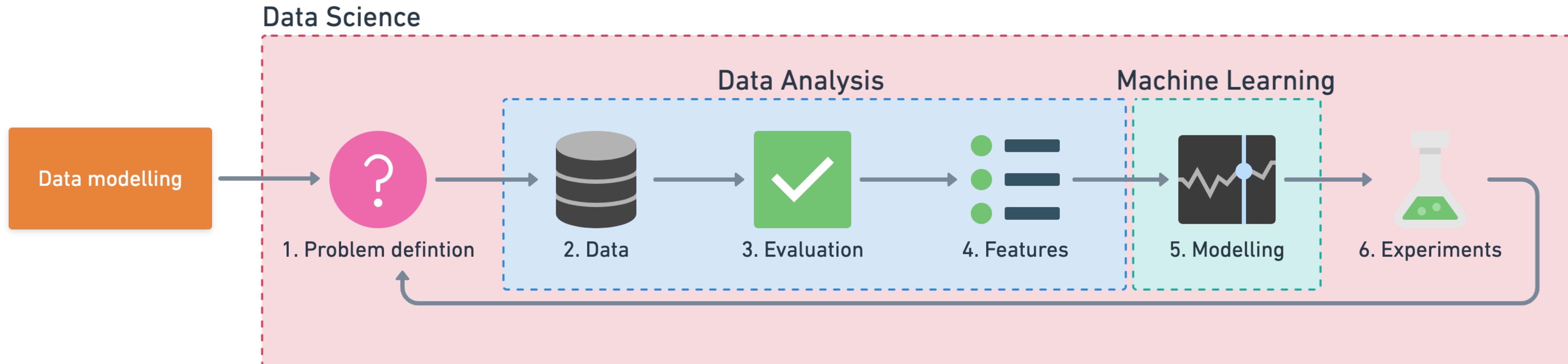
# What is NumPy?



## Steps in a full machine learning project



# Tools you can use



# Why NumPy?

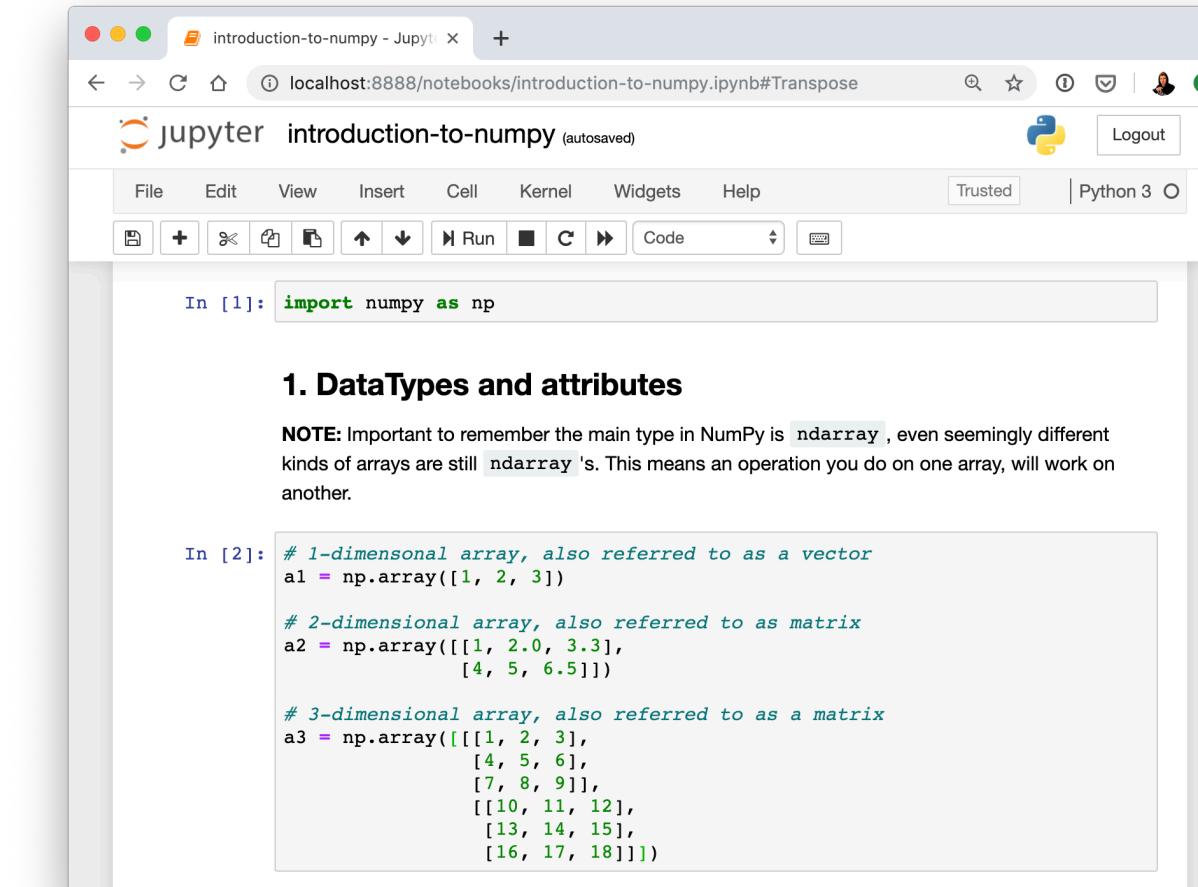
- It's fast
- Behind the scenes optimizations written in C
- Vectorization via broadcasting (avoiding loops)
- Backbone of other Python scientific packages

# What are we going to cover?

- Most useful functions
- NumPy datatypes & attributes (ndarray)
- Creating arrays
- Viewing arrays & matrices
- Manipulating & comparing arrays
- Sorting arrays
- Use cases

# Where can you get help?

- Follow along with the code



In [1]: `import numpy as np`

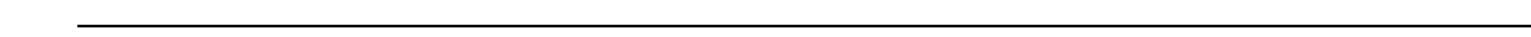
**1. DataTypes and attributes**

NOTE: Important to remember the main type in NumPy is `ndarray`, even seemingly different kinds of arrays are still `ndarray`'s. This means an operation you do on one array, will work on another.

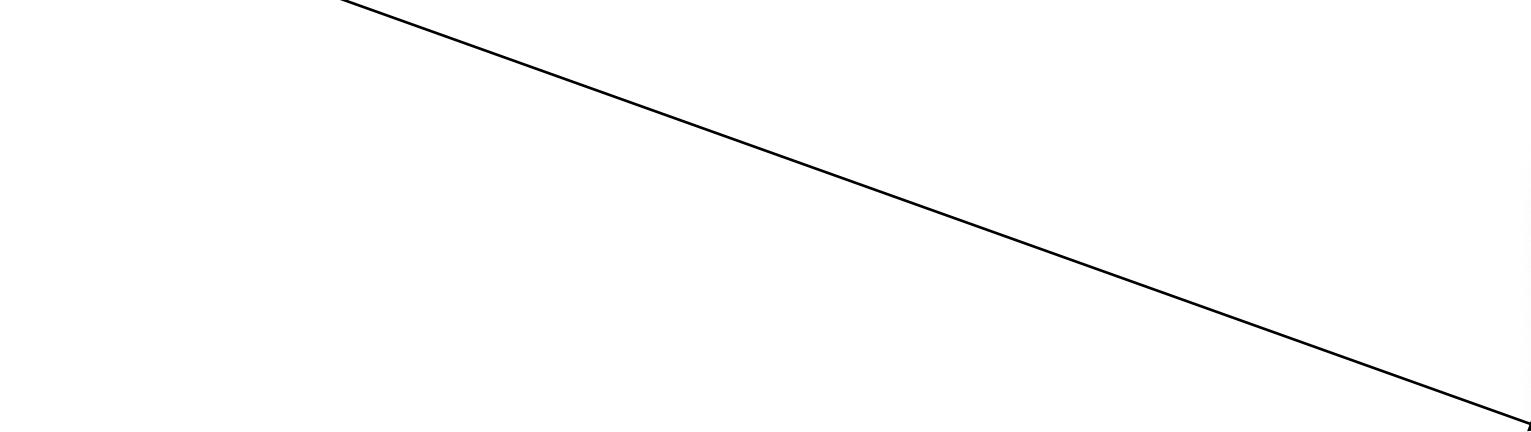
In [2]: `# 1-dimensional array, also referred to as a vector  
a1 = np.array([1, 2, 3])  
  
# 2-dimensional array, also referred to as matrix  
a2 = np.array([[1, 2.0, 3.3],  
 [4, 5, 6.5]])  
  
# 3-dimensional array, also referred to as a matrix  
a3 = np.array([[[1, 2, 3],  
 [4, 5, 6],  
 [7, 8, 9]],  
 [[10, 11, 12],  
 [13, 14, 15],  
 [16, 17, 18]]])`

- Try it for yourself

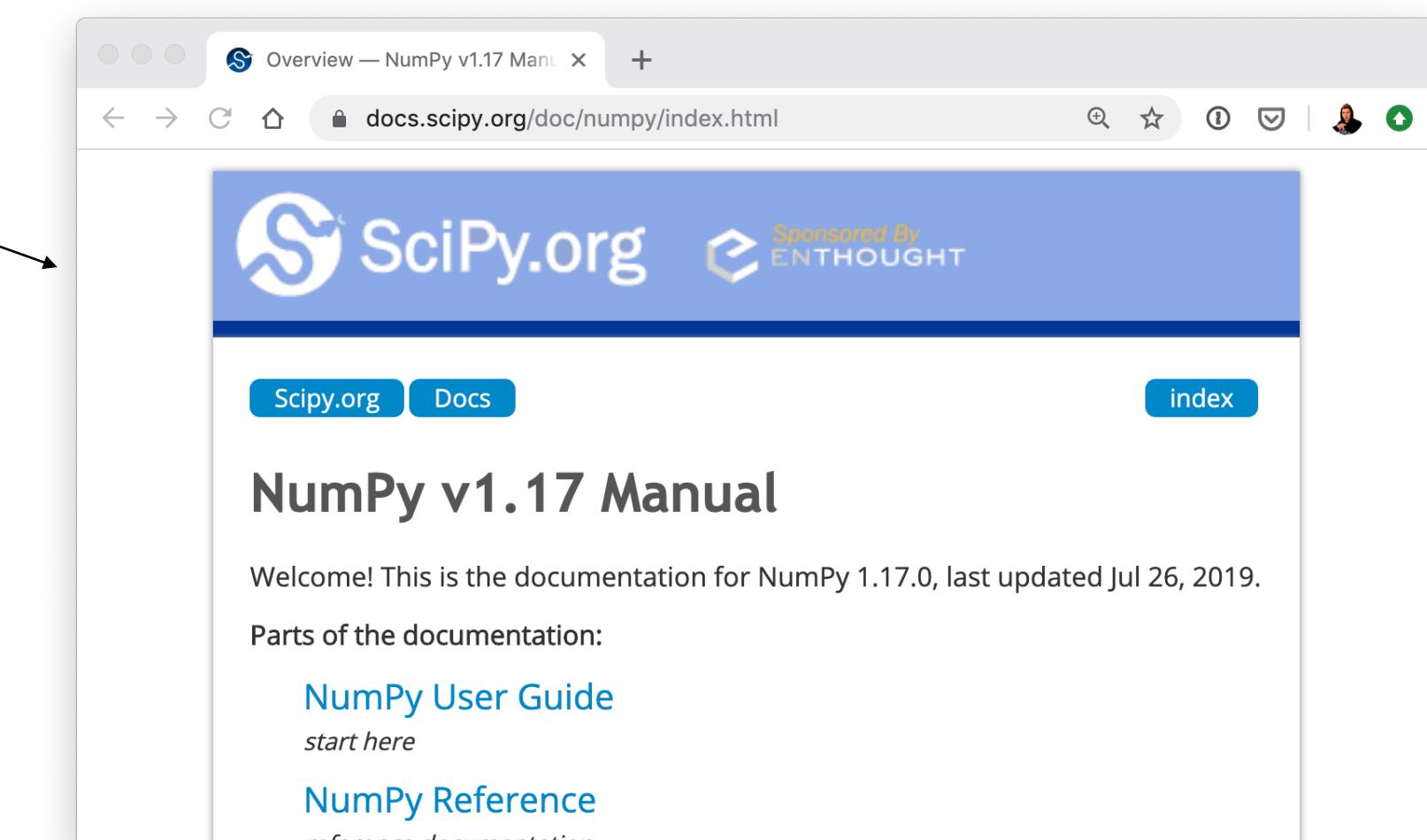
- Search for it



- Try again



- Ask

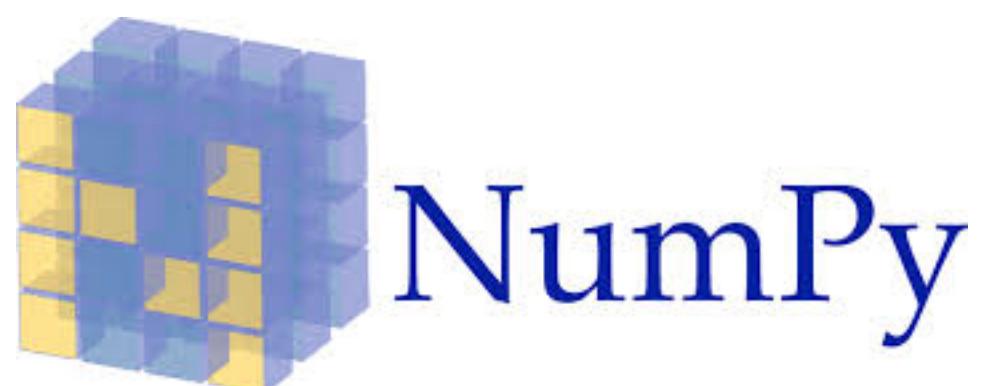


The main difference is the bottom left array. In the videos it's displayed incorrectly.

If you'd like to get the same array as the "Anatomy of a NumPy Array" slide shows in the following videos, run the following code:

```
# Same array as slide in video
a3 = np.array([[[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9],
                [10, 11, 12]],
               [[13, 14, 15],
                [16, 17, 18]]])
```

## Let's code!



And just another reminder, figuring out the different shapes of arrays can be a bit of a challenge! It takes a fair bit of practice to get used to them.

The code and concepts in the lectures are all correct, however, the slide displays the wrong array for the 3-dimensional array (you'll see these in the upcoming videos).

The correct slide should be:

# Anatomy of a NumPy array

## Data

1	1	2	3
3			

## NumPy

```
array([1, 2, 3])
```

## Details

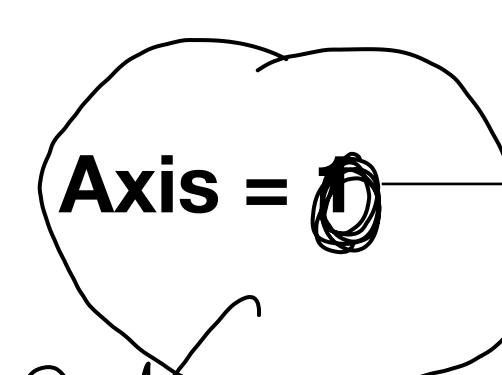
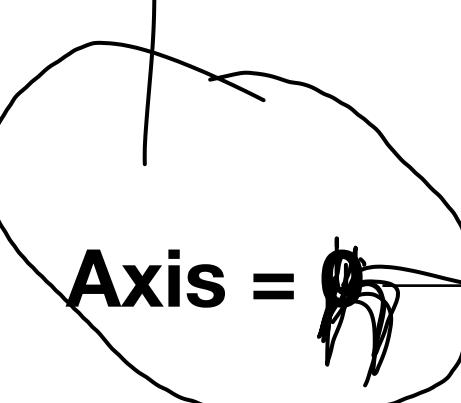
- Names: Array, vector
- 1-dimensional
- Shape = (1, 3)

for depth

1	2	3.3
4	5	6.5
3		

```
array([[1., 2., 3.3],  
       [4., 5., 6.5]])
```

- Names: Array, matrix
- More than 1-dimension
- Shape = (2, 3)



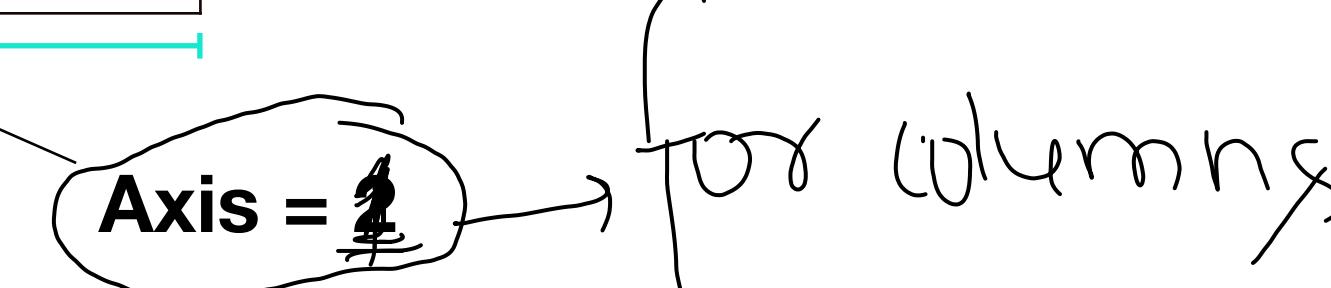
for rows

10	11	12
1	2	3
4	5	6
7	8	9

```
array([[[ 1,  2,  3],  
        [ 4,  5,  6],  
        [ 7,  8,  9]],
```

```
      [[10, 11, 12],  
       [13, 14, 15],  
       [16, 17, 18]]])
```

- Names: Array, matrix
- More than 1-dimension
- Shape = (2, 3, 3)



for columns

# Dot product vs. element-wise

**Element-wise**

$$\begin{array}{|c|c|} \hline A & B \\ \hline C & D \\ \hline \end{array} * \begin{array}{|c|c|} \hline E & F \\ \hline G & H \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline A^*E & B^*F \\ \hline C^*G & D^*H \\ \hline \end{array}$$

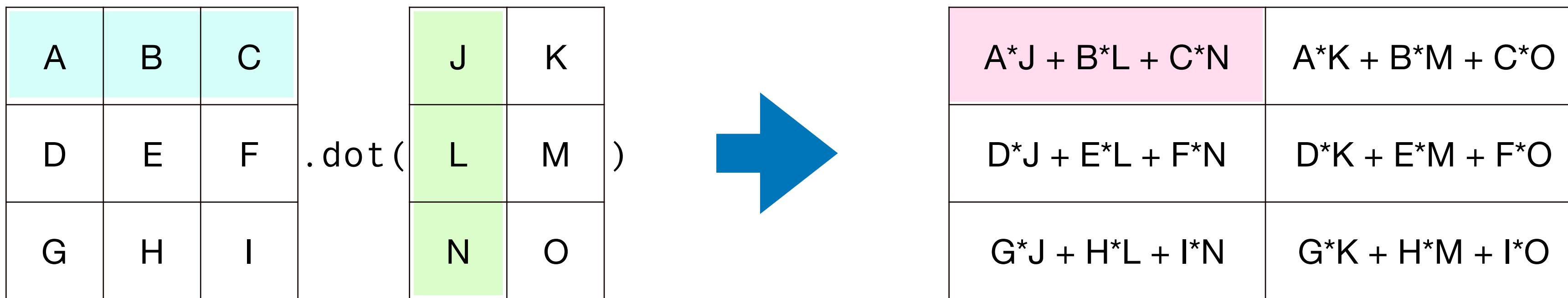
**2x2**                    **2x2**                    **2x2**

**Dot product**

$$\begin{array}{|c|c|c|} \hline A & B & C \\ \hline D & E & F \\ \hline G & H & I \\ \hline \end{array} \cdot \text{dot} \left( \begin{array}{|c|c|} \hline J & K \\ \hline L & M \\ \hline N & O \\ \hline \end{array} \right) \rightarrow \begin{array}{|c|c|} \hline A^*J + B^*L + C^*N & A^*K + B^*M + C^*O \\ \hline D^*J + E^*L + F^*N & D^*K + E^*M + F^*O \\ \hline G^*J + H^*L + I^*N & G^*K + H^*M + I^*O \\ \hline \end{array}$$

**3x3**                    **3x2**                    **3x2**

# Dot product

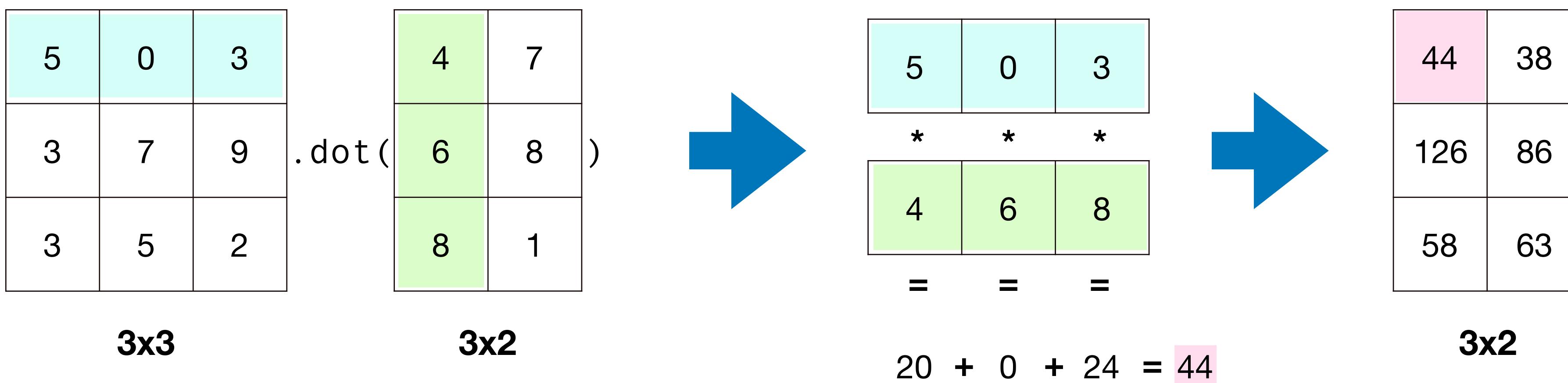


**3x3** ← → **3x2**

Numbers on the inside must match

**3x2**

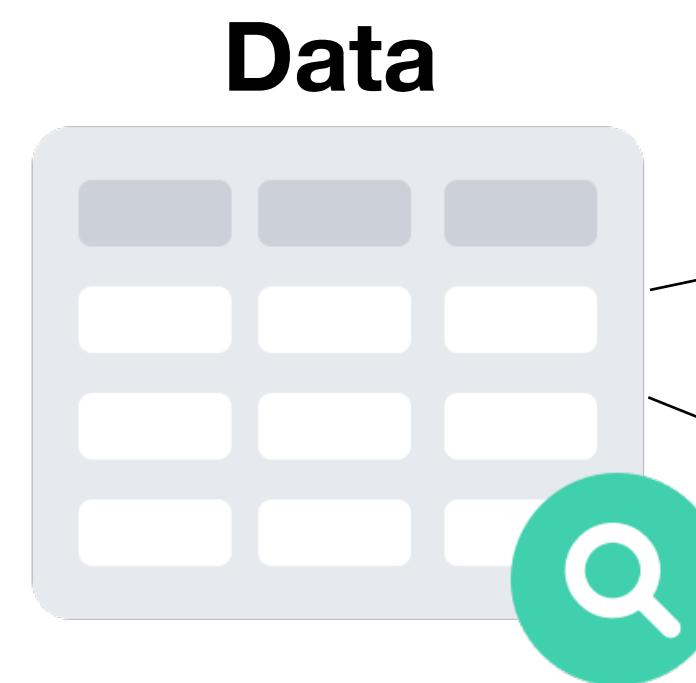
New size is same as outside numbers



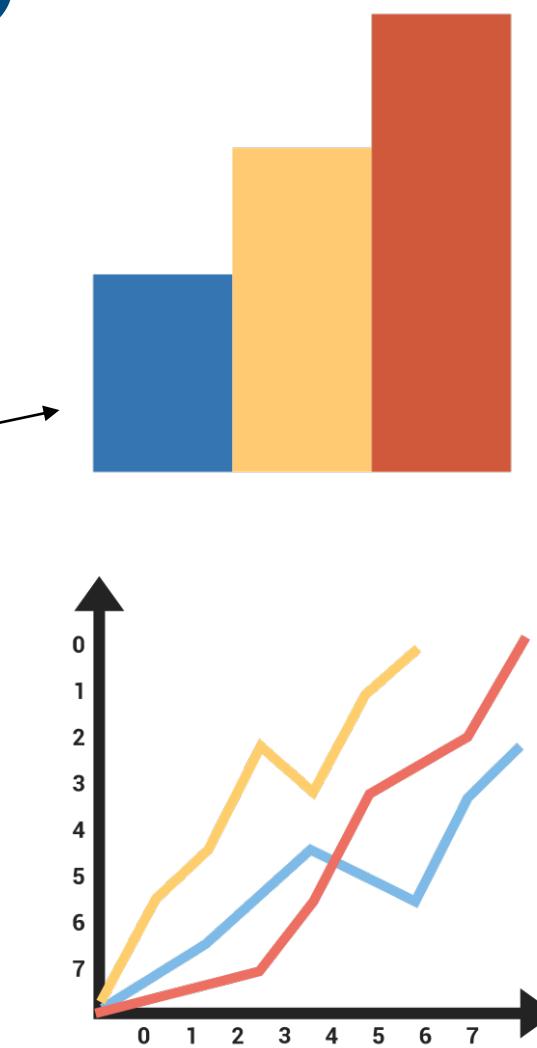
For a live demo, checkout [www.matrixmultiplication.xyz](http://www.matrixmultiplication.xyz)

# What is Matplotlib?

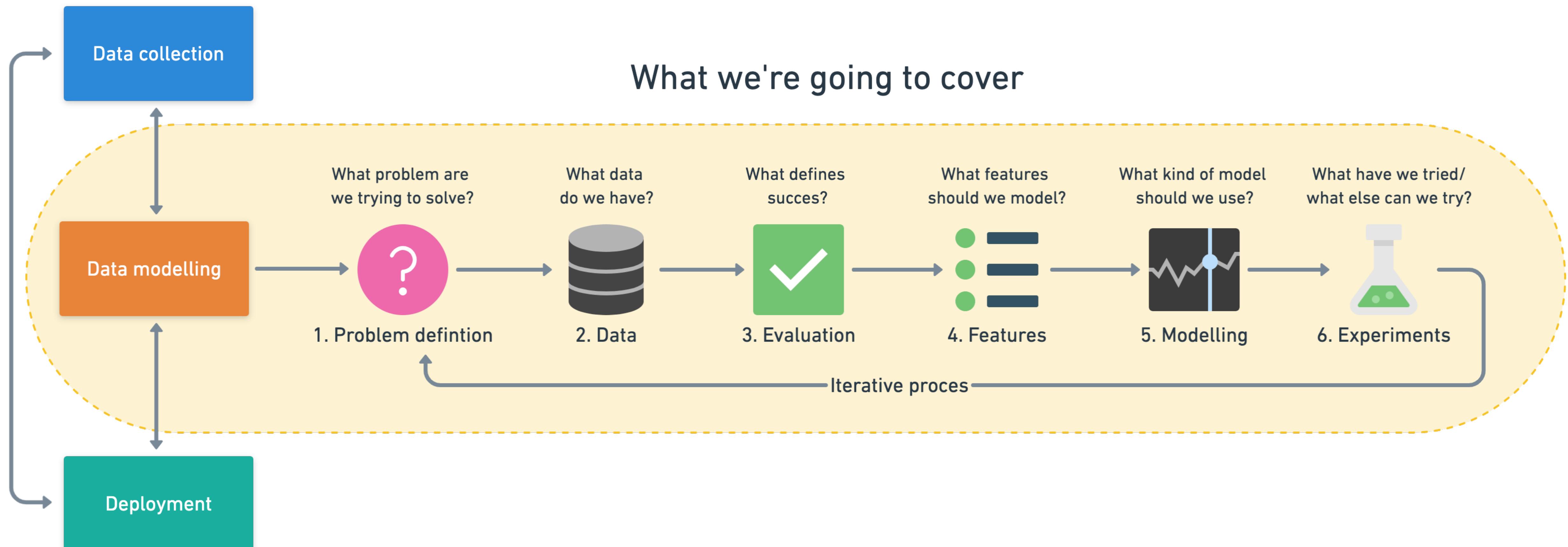
matplotlib



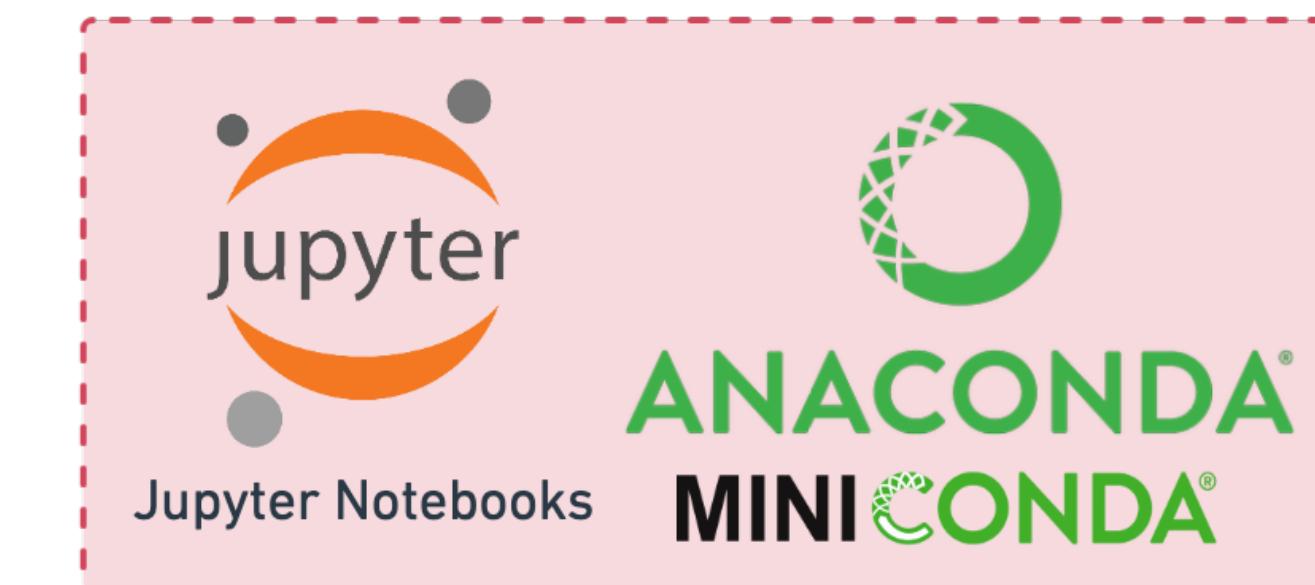
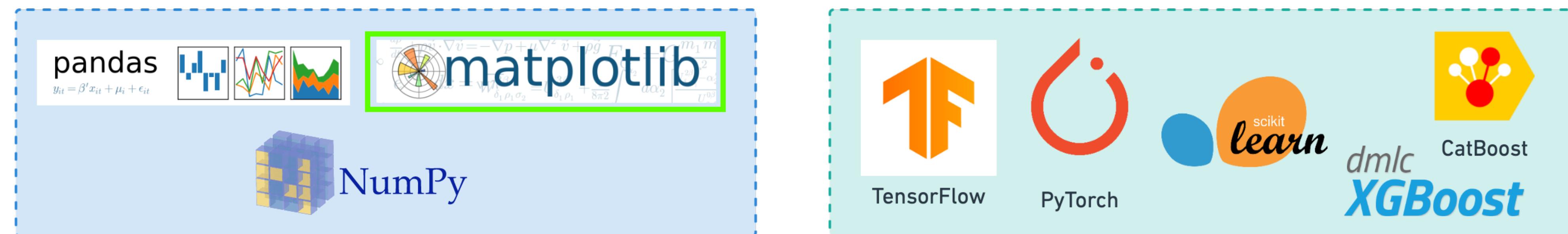
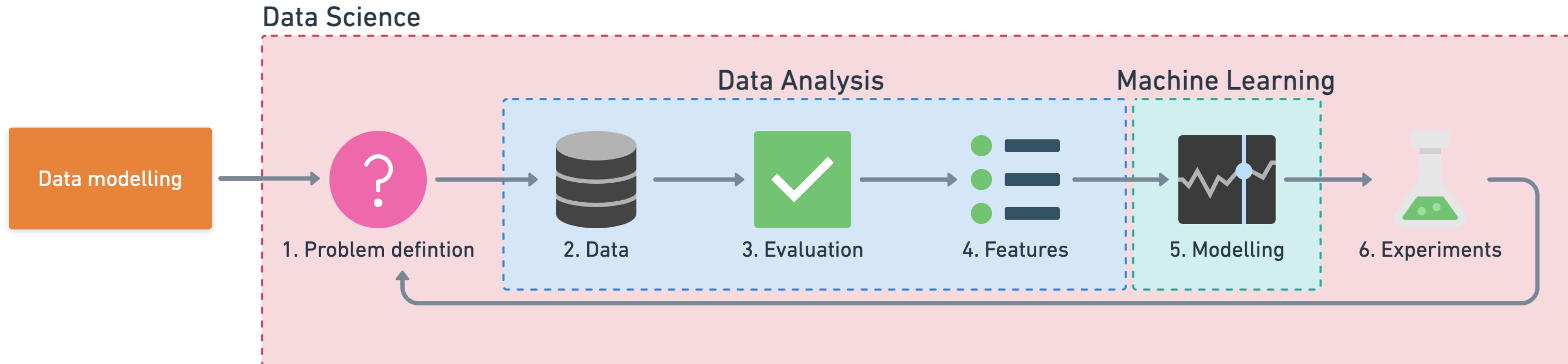
Data



## Steps in a full machine learning project



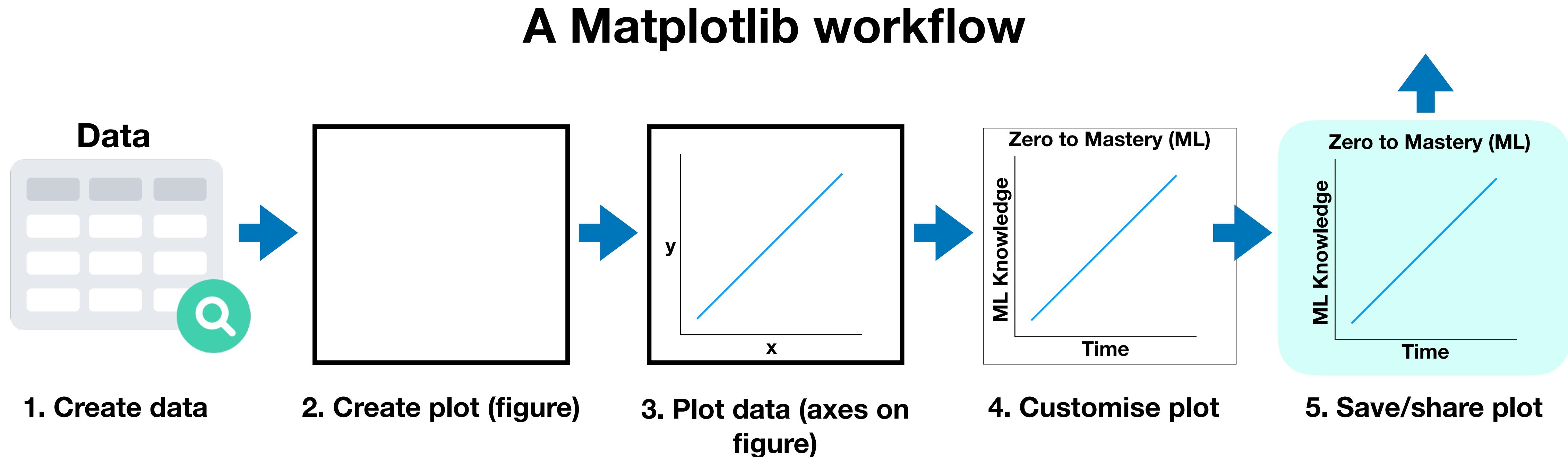
# Tools you can use



# Why Matplotlib?

- Built on NumPy arrays (and Python)
- Integrates directly with pandas
- Can create basic or advanced plots
- Simple to use interface (once you get the foundations)

# What are we going to cover?

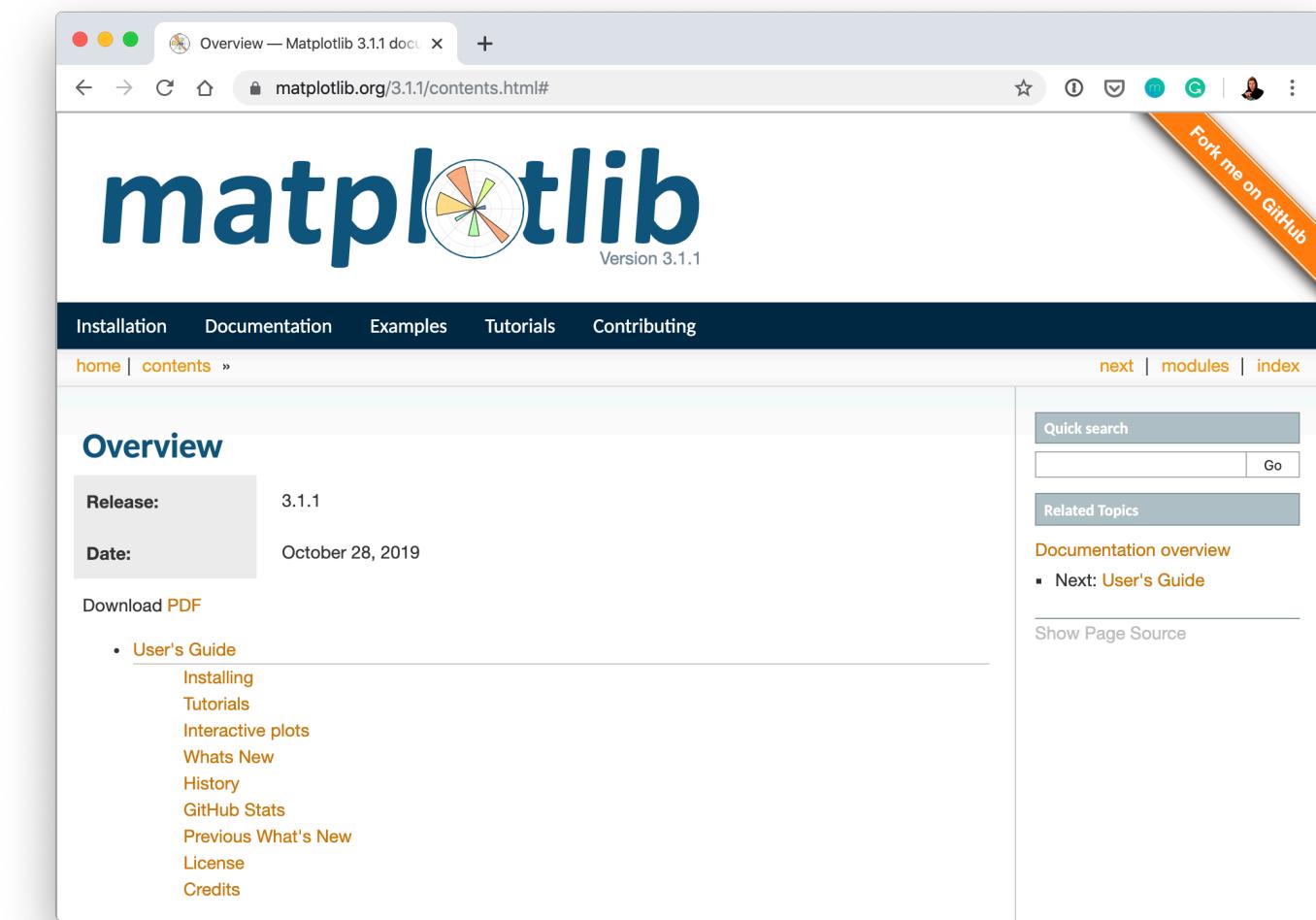
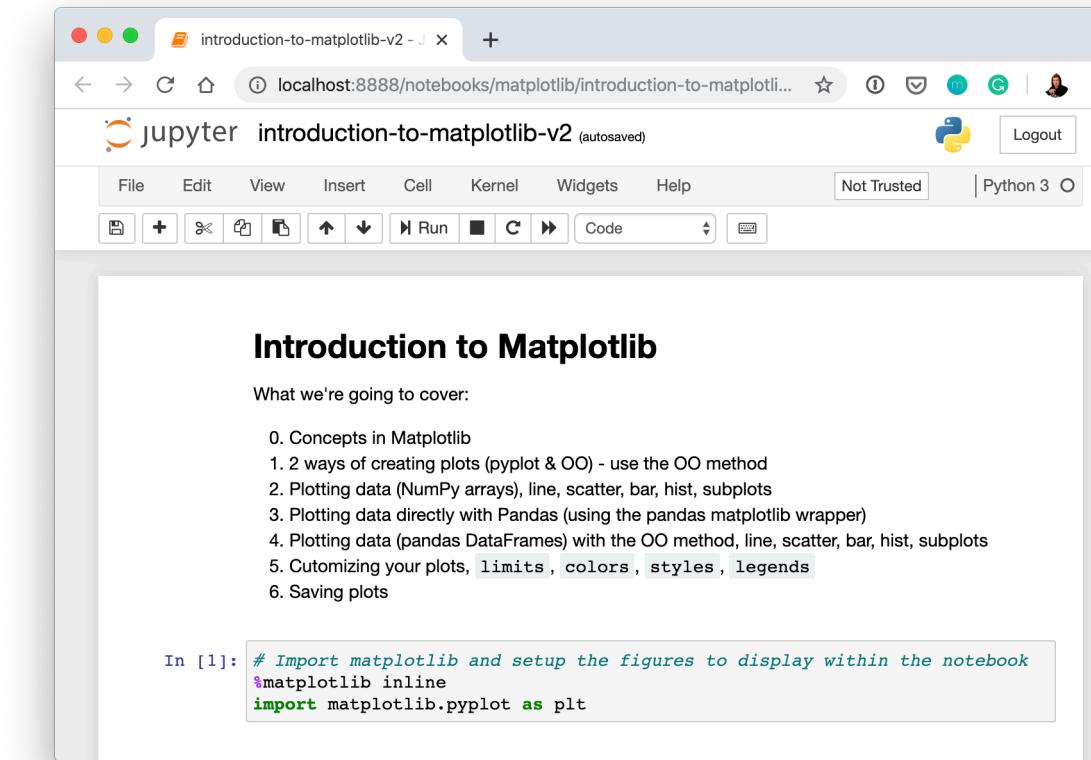


# What are we going to cover?

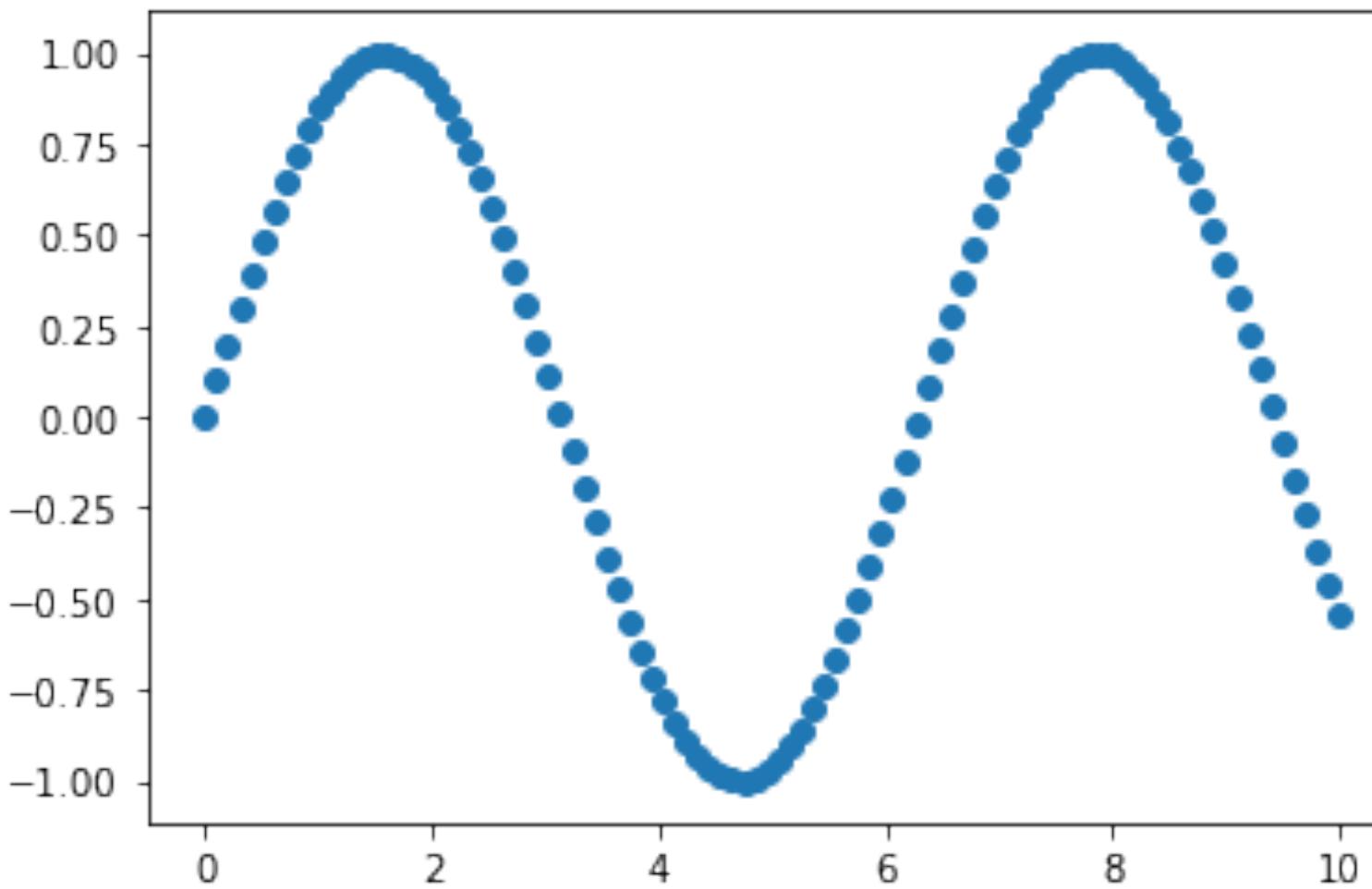
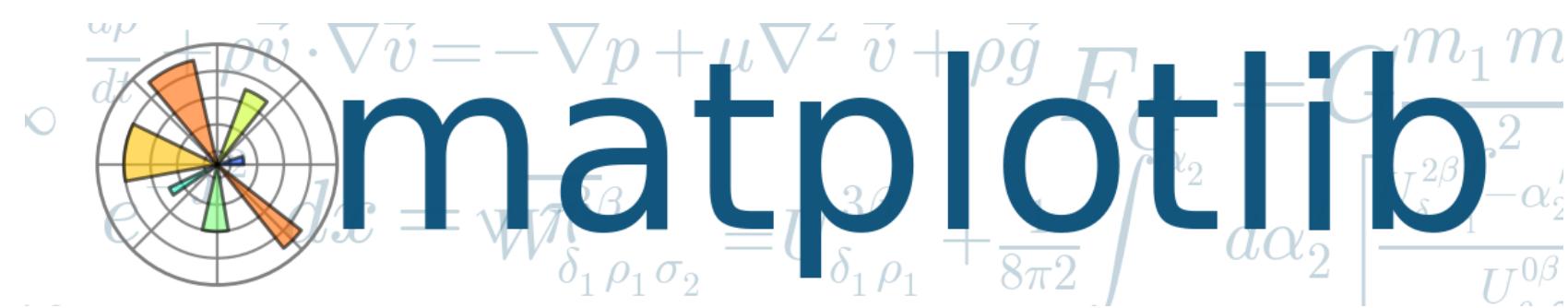
- Matplotlib workflow
- Importing Matplotlib and the 2 ways of plotting
- Plotting data from NumPy arrays
- Plotting data from pandas DataFrames
- Customizing plots
- Saving and sharing plots

# Where can you get help?

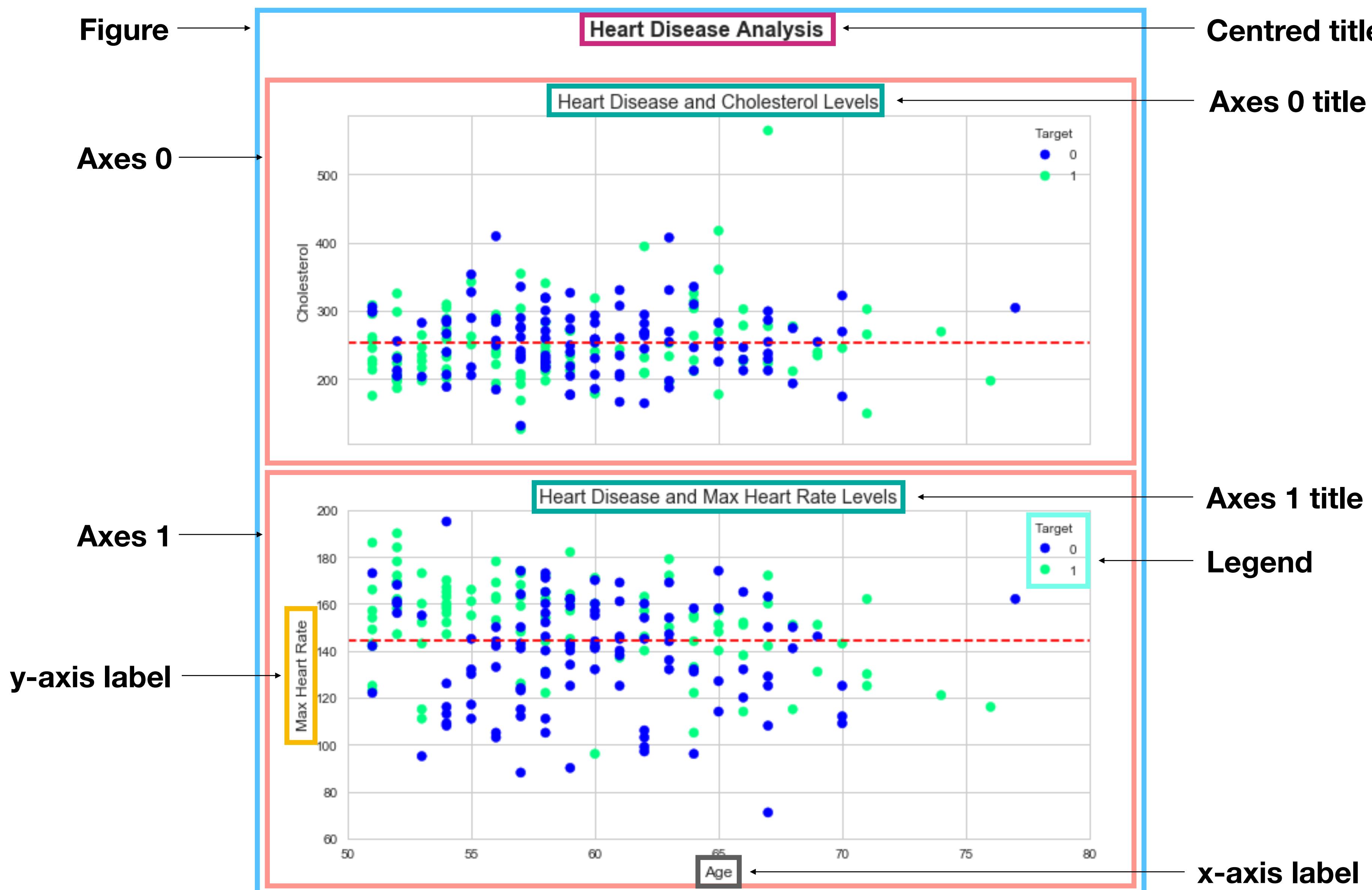
- Follow along with the code
- Try it for yourself
- Search for it
- Try again
- Ask



# Let's plot!



# Anatomy of a Matplotlib plot



# Anatomy of a Matplotlib plot

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid') # set plot style

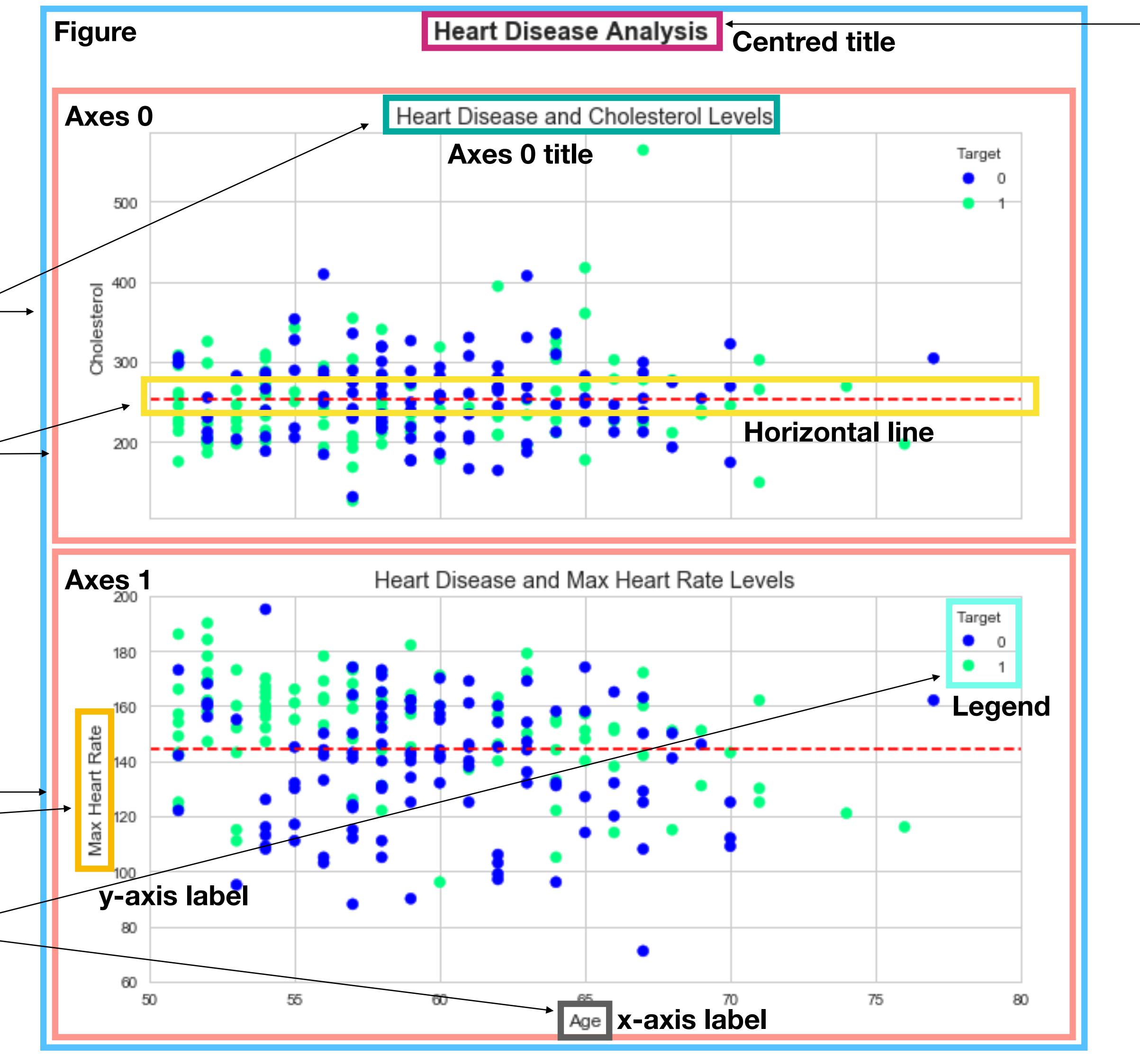
# Read in and manipulate data
heart_disease = pd.read_csv('../data/heart-disease.csv')
over_50 = heart_disease[heart_disease['age'] > 50]

# Create figure (plot) with 2 axes
fig, (ax0, ax1) = plt.subplots(nrows=2,
                               ncols=1,
                               sharex=True,
                               figsize=(10, 10))

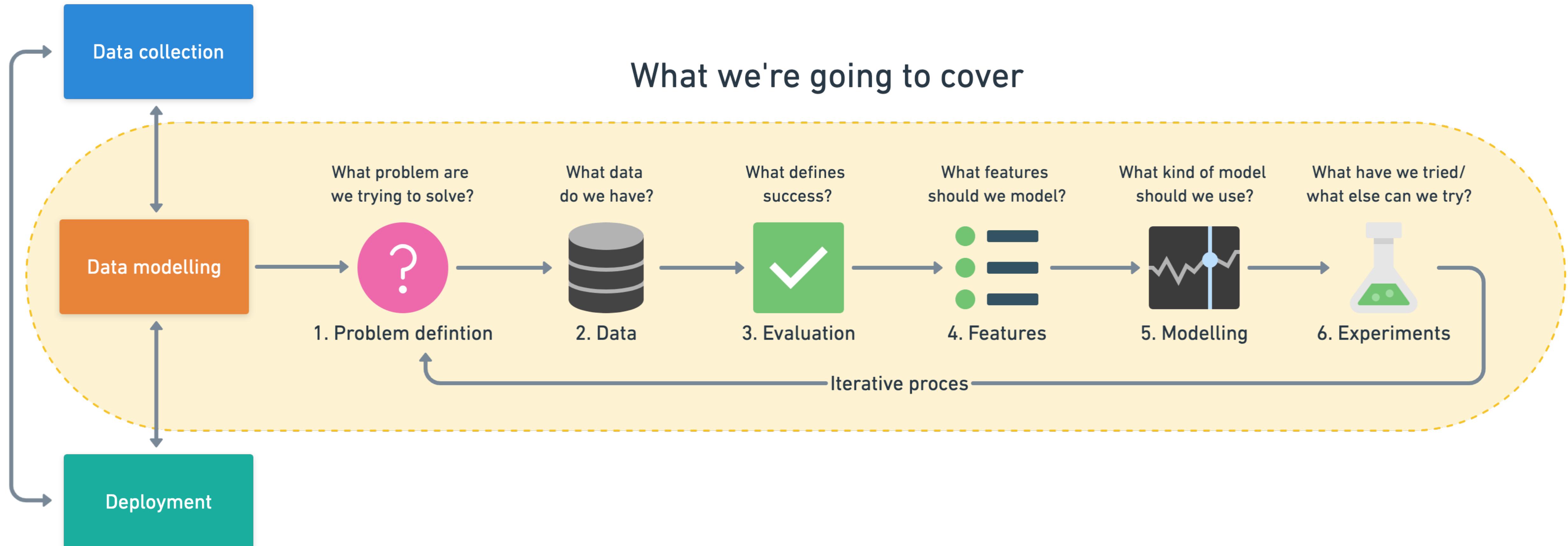
# Add data, titles, meanline (axhline) and legend to axes 0
scatter = ax0.scatter(over_50["age"],
                      over_50["chol"],
                      c=over_50["target"],
                      cmap='winter')
ax0.set(title="Heart Disease and Cholesterol Levels",
        ylabel="Cholesterol",
        xlim=[50, 80])
ax0.axhline(y=over_50["chol"].mean(),
             color='r',
             linestyle='--',
             label="Average");
ax0.legend(*scatter.legend_elements(), title="Target")

# Add data, titles, meanline (axhline) and legend to axes 1
scatter = ax1.scatter(over_50["age"],
                      over_50["thalach"],
                      c=over_50["target"],
                      cmap='winter')
ax1.set(title="Heart Disease and Max Heart Rate Levels",
        xlabel="Age",
        ylabel="Max Heart Rate",
        ylim=[60, 200])
ax1.axhline(y=over_50["thalach"].mean(),
             color='r',
             linestyle='--',
             label="Average");
ax1.legend(*scatter.legend_elements(), title="Target")

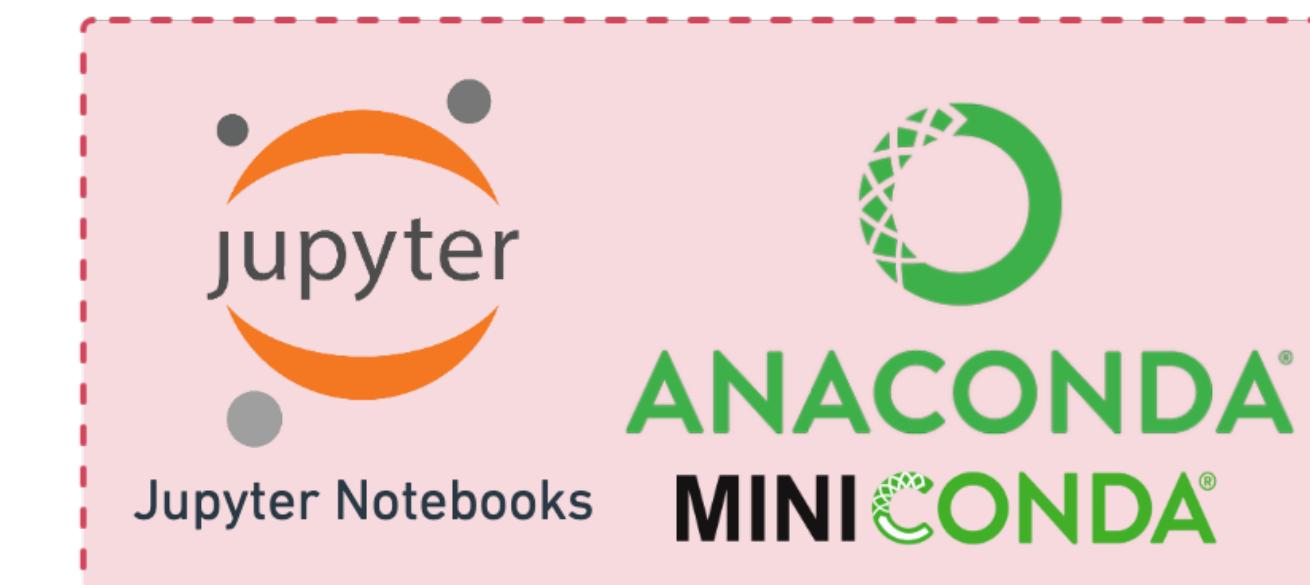
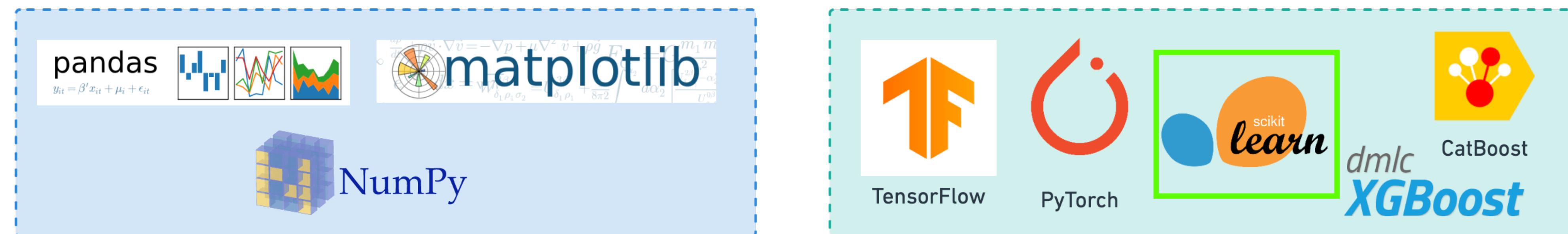
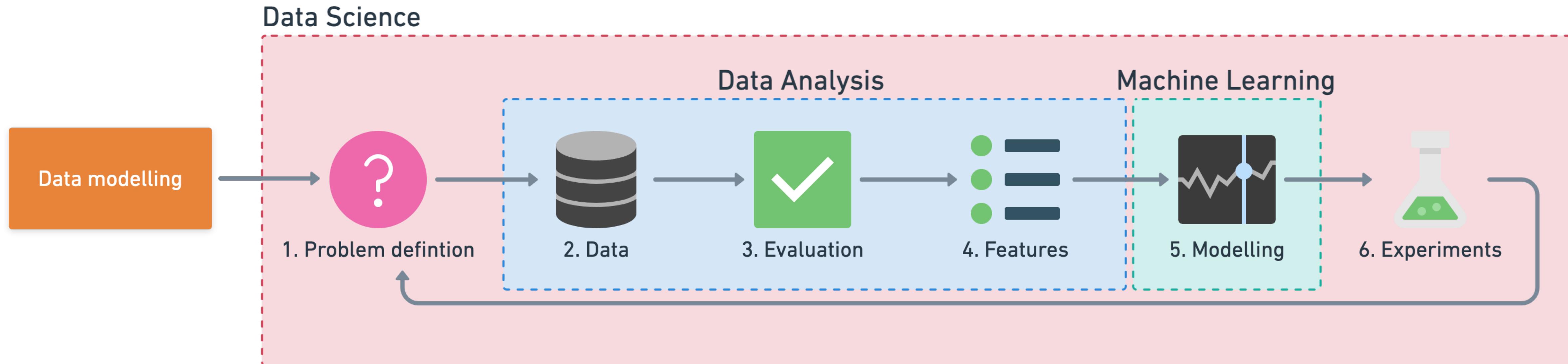
# Title the figure
fig.suptitle('Heart Disease Analysis', fontsize=16, fontweight='bold');
```



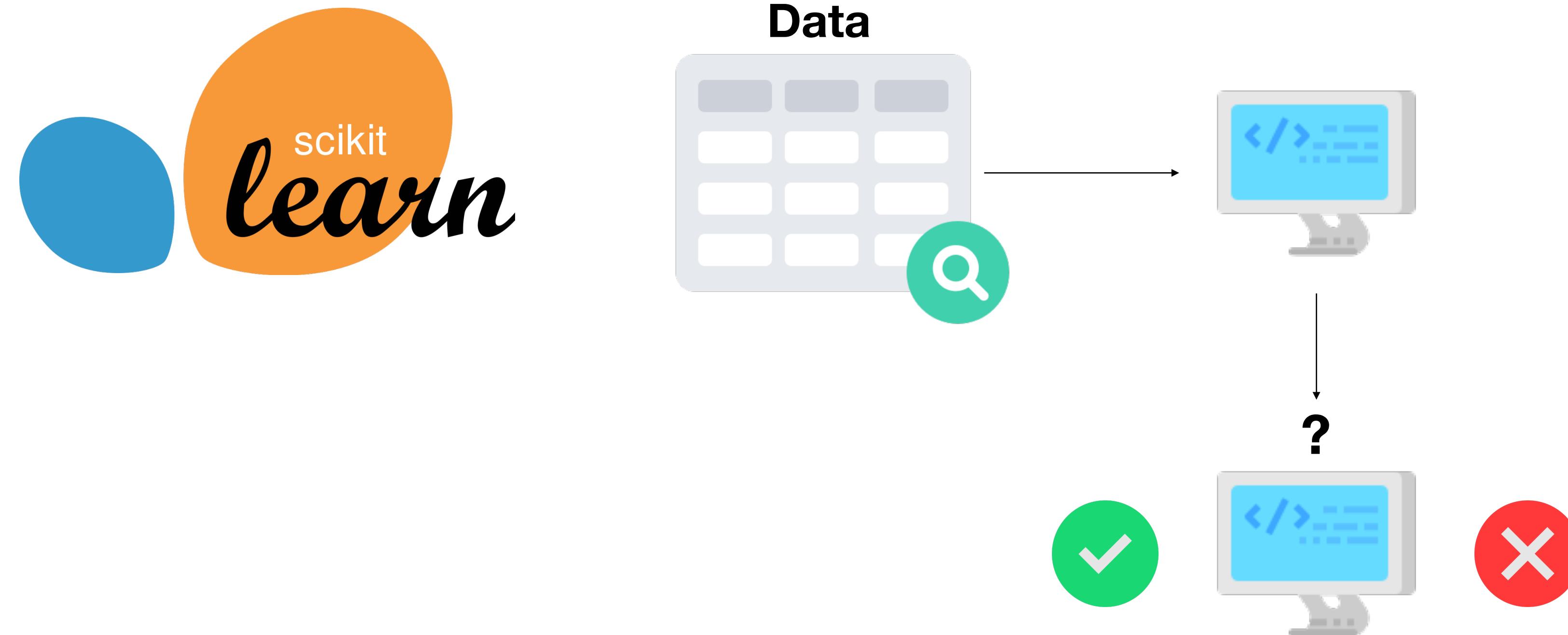
## Steps in a full machine learning project



# Tools you can use



# What is Scikit-Learn (sklearn)?

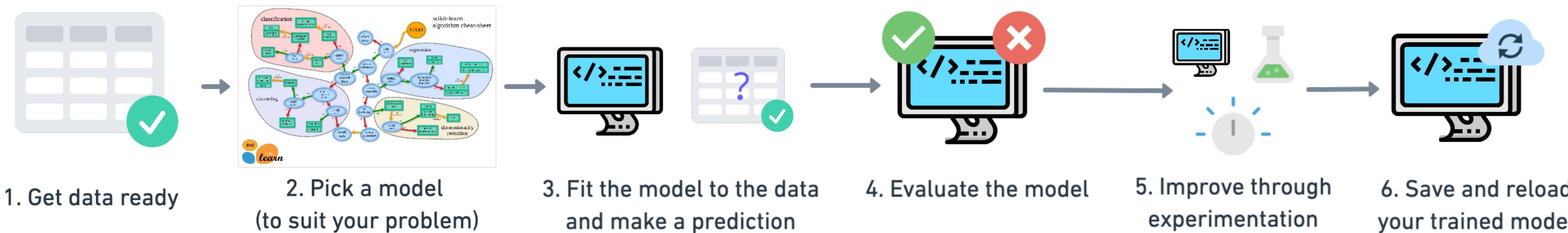


# Why Scikit-Learn?

- Built on NumPy and Matplotlib (and Python)
- Has many in-built machine learning models
- Methods to evaluate your machine learning models
- Very well-designed API

# What are we going to cover?

## A Scikit-Learn workflow

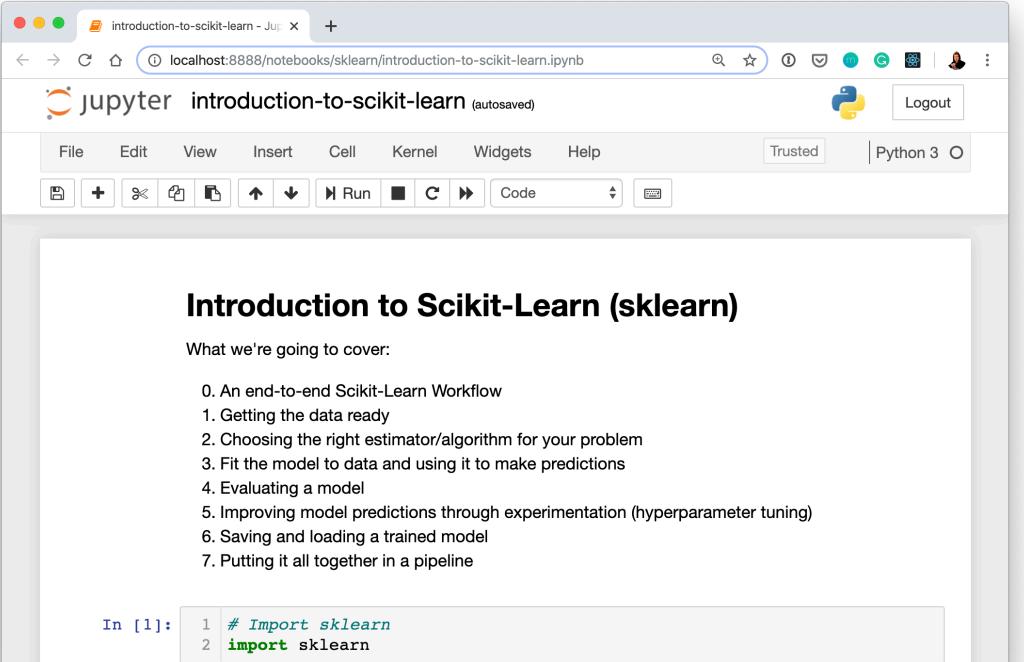


# What are we going to cover?

- An end-to-end Scikit-Learn workflow
- Getting data ready (to be used with machine learning models)
- Choosing a machine learning model
- Fitting a model to the data (learning patterns)
- Making predictions with a model (using patterns)
- Evaluating model predictions
- Improving model predictions
- Saving and loading models

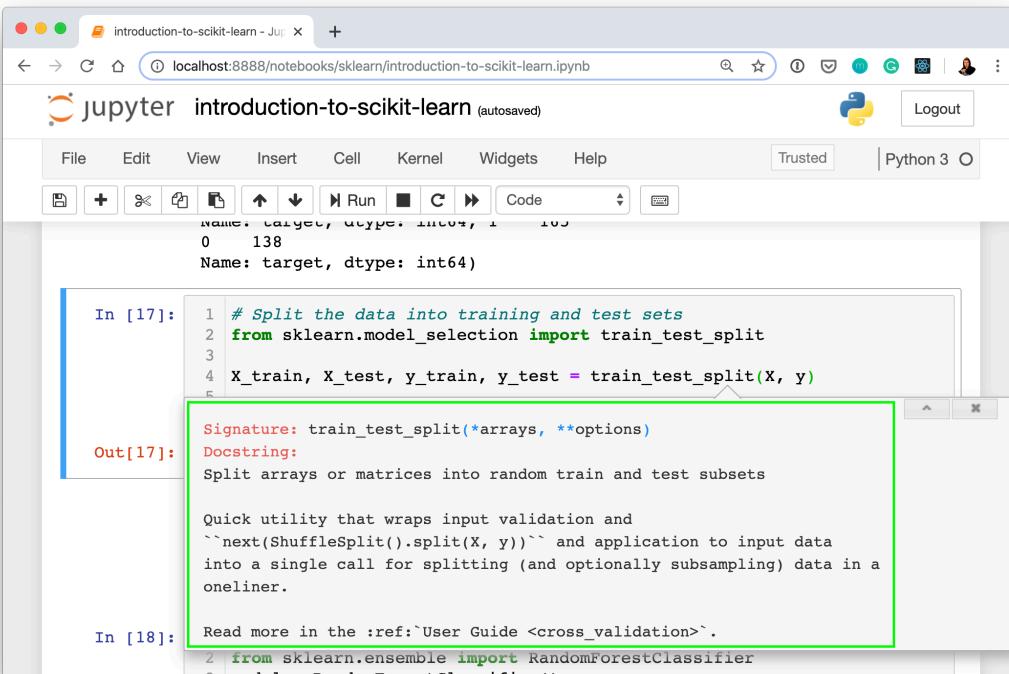
# Where can you get help?

- Follow along with the code



```
In [1]: 1 # Import sklearn  
2 import sklearn
```

- Try it for yourself



```
In [17]: 1 # Split the data into training and test sets  
2 from sklearn.model_selection import train_test_split  
3  
4 X_train, X_test, y_train, y_test = train_test_split(X, y)  
  
Out[17]:  
Signature: train_test_split(*arrays, **options)  
Docstring:  
Split arrays or matrices into random train and test subsets  
  
Quick utility that wraps input validation and  
``next(ShuffleSplit().split(X, y))`` and application to input data  
into a single call for splitting (and optionally subsampling) data in a  
oneliner.  
  
In [18]: Read more in the ref:`User Guide <cross_validation>`.  
1 from sklearn.ensemble import RandomForestClassifier  
2 model = RandomForestClassifier()
```

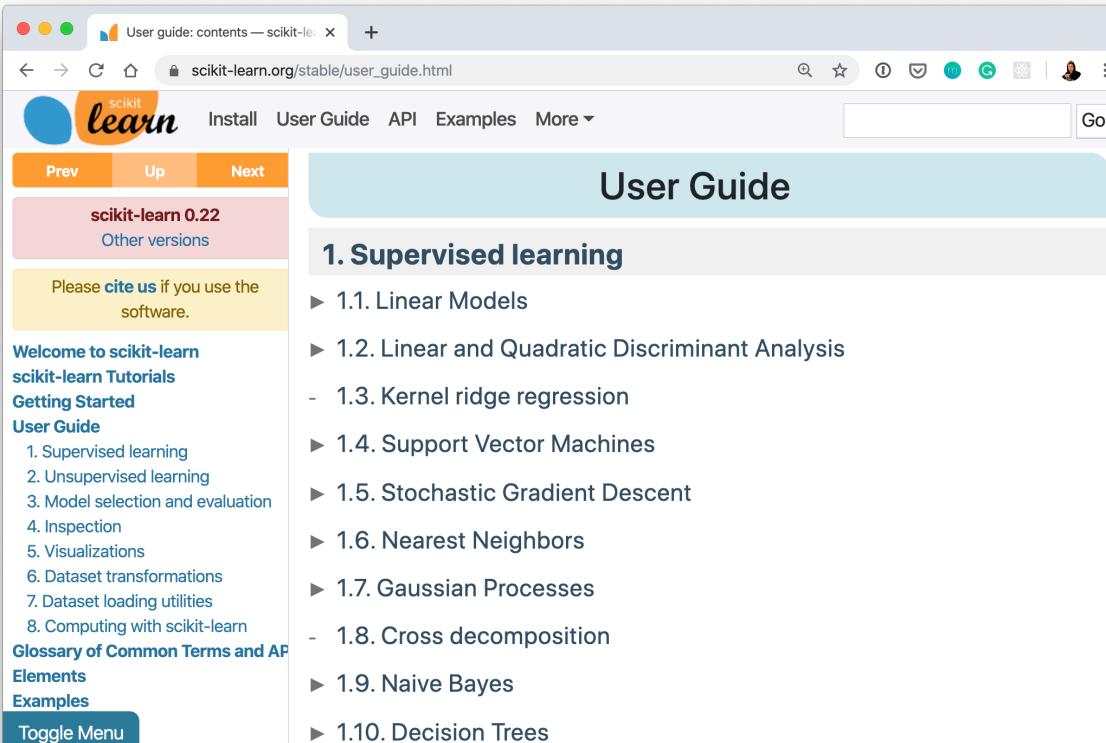
- Press SHIFT + TAB to read the docstring

- Search for it



- Try again

- Ask



scikit-learn 0.22  
Other versions

Please cite us if you use the software.

Welcome to scikit-learn  
scikit-learn Tutorials  
Getting Started  
User Guide

- 1. Supervised learning
- 2. Unsupervised learning
- 3. Model selection and evaluation
- 4. Inspection
- 5. Visualizations
- 6. Dataset transformations
- 7. Dataset loading utilities
- 8. Computing with scikit-learn

Glossary of Common Terms and API  
Elements  
Examples

## User Guide

### 1. Supervised learning

- ▶ 1.1. Linear Models
- ▶ 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- ▶ 1.4. Support Vector Machines
- ▶ 1.5. Stochastic Gradient Descent
- ▶ 1.6. Nearest Neighbors
- ▶ 1.7. Gaussian Processes
- 1.8. Cross decomposition
- ▶ 1.9. Naive Bayes
- ▶ 1.10. Decision Trees

# Let's model!



# Supervised learning



## Classification

- “Is this example one thing or another?”
- Binary classification = two options
- Multi-class classification = more than two options



## Regression

- “How much will this house sell for?”
- “How many people will buy this app?”

# One Hot Encoding

A process used to turn categories into numbers.

Car	Colour	Car	Red	Green	Blue
0	Red	0	1	0	0
1	Green	1	0	1	0
2	Blue	2	0	0	1
3	Red	3	1	0	0

# Classification and Regression metrics

## Classification

**Accuracy**

Precision

Recall

F1

## Regression

**R<sup>2</sup> (r-squared)**

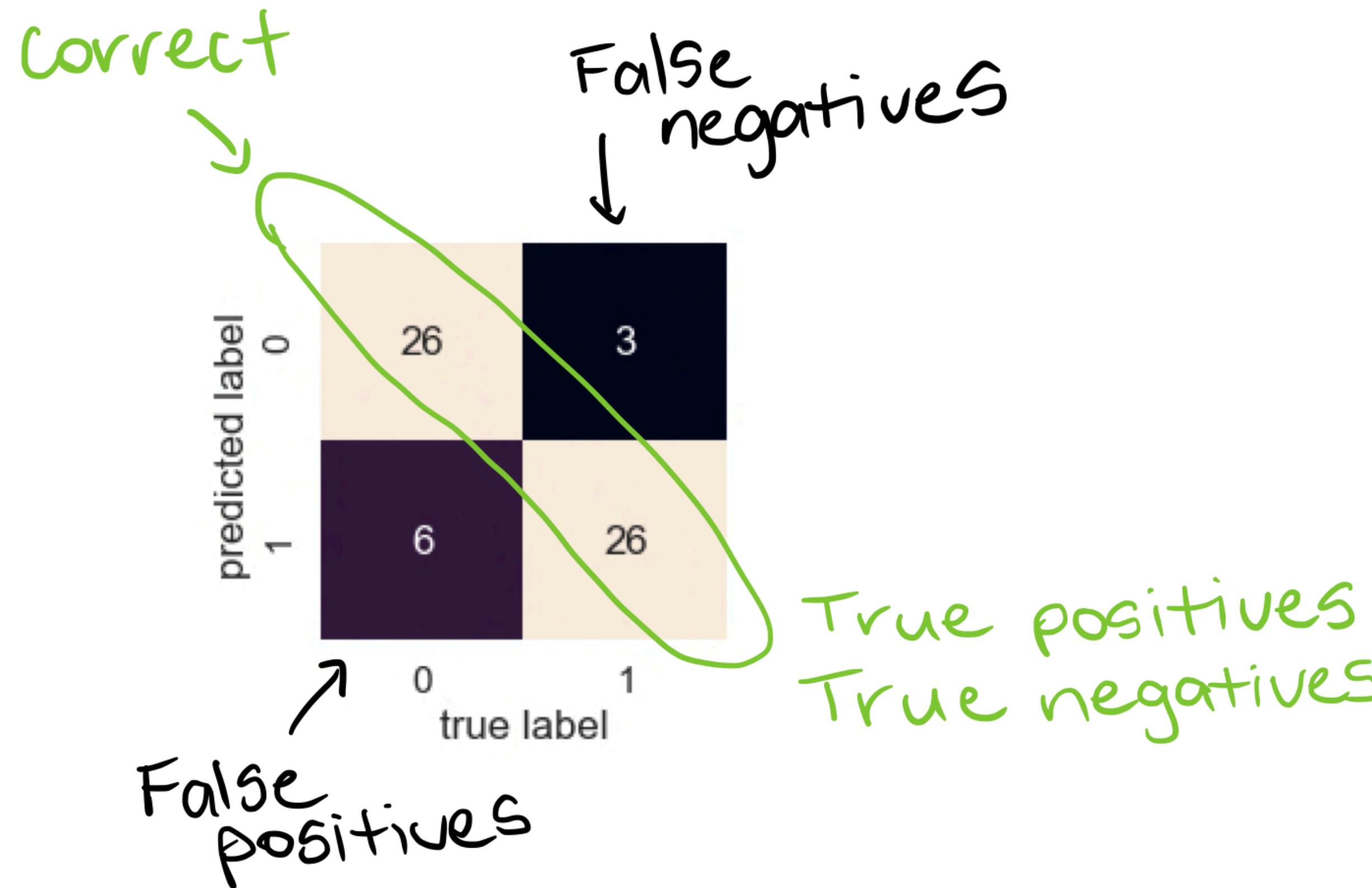
Mean absolute error (MAE)

Mean squared error (MSE)

Root mean squared error (RMSE)

**Bold** = default evaluation in Scikit-Learn

# Confusion matrix anatomy



- **True positive** = model predicts 1 when truth is 1
- **False positive** = model predicts 1 when truth is 0
- **True negative** = model predicts 0 when truth is 0
- **False negative** = model predicts 0 when truth is 1

# Classification report anatomy

```
1 from sklearn.metrics import classification_report  
2  
3 print(classification_report(y_test, y_preds))
```

	precision	recall	f1-score	support
0	0.81	0.90	0.85	29
1	0.90	0.81	0.85	32
accuracy			0.85	61
macro avg	0.85	0.85	0.85	61
weighted avg	0.86	0.85	0.85	61

- **Precision** - Indicates the proportion of positive identifications (model predicted class 1) which were actually correct. A model which produces no false positives has a precision of 1.0.
- **Recall** - Indicates the proportion of actual positives which were correctly classified. A model which produces no false negatives has a recall of 1.0.
- **F1 score** - A combination of precision and recall. A perfect model achieves an F1 score of 1.0.
- **Support** - The number of samples each metric was calculated on.
- **Accuracy** - The accuracy of the model in decimal form. Perfect accuracy is equal to 1.0.
- **Macro avg** - Short for macro average, the average precision, recall and F1 score between classes. Macro avg doesn't class imbalance into effort, so if you do have class imbalances, pay attention to this metric.
- **Weighted avg** - Short for weighted average, the weighted average precision, recall and F1 score between classes. Weighted means each metric is calculated with respect to how many samples there are in each class. This metric will favour the majority class (e.g. will give a high value when one class out performs another due to having more samples).

# Which classification metric should you use?

- **Accuracy** is a good measure to start with if all classes are balanced (e.g. same amount of samples which are labelled with 0 or 1).
- **Precision** and **recall** become more important when classes are imbalanced.
- If false positive predictions are worse than false negatives, aim for higher precision.
- If false negative predictions are worse than false positives, aim for higher recall.
- **F1-score** is a combination of precision and recall.

# Which regression metric should you use?

- **R<sup>2</sup>** is similar to accuracy. It gives you a quick indication of how well your model might be doing. Generally, the closer your **R<sup>2</sup>** value is to 1.0, the better the model. But it doesn't really tell exactly how wrong your model is in terms of how far off each prediction is.
- **MAE** gives a better indication of how far off each of your model's predictions are on average.
- As for **MAE** or **MSE**, because of the way MSE is calculated, squaring the differences between predicted values and actual values, it amplifies larger differences. Let's say we're predicting the value of houses (which we are).
  - Pay more attention to MAE: When being \$10,000 off is **twice** as bad as being \$5,000 off.
  - Pay more attention to MSE: When being \$10,000 off is **more than twice** as bad as being \$5,000 off.

# Improving a model (via hyperparameter tuning)

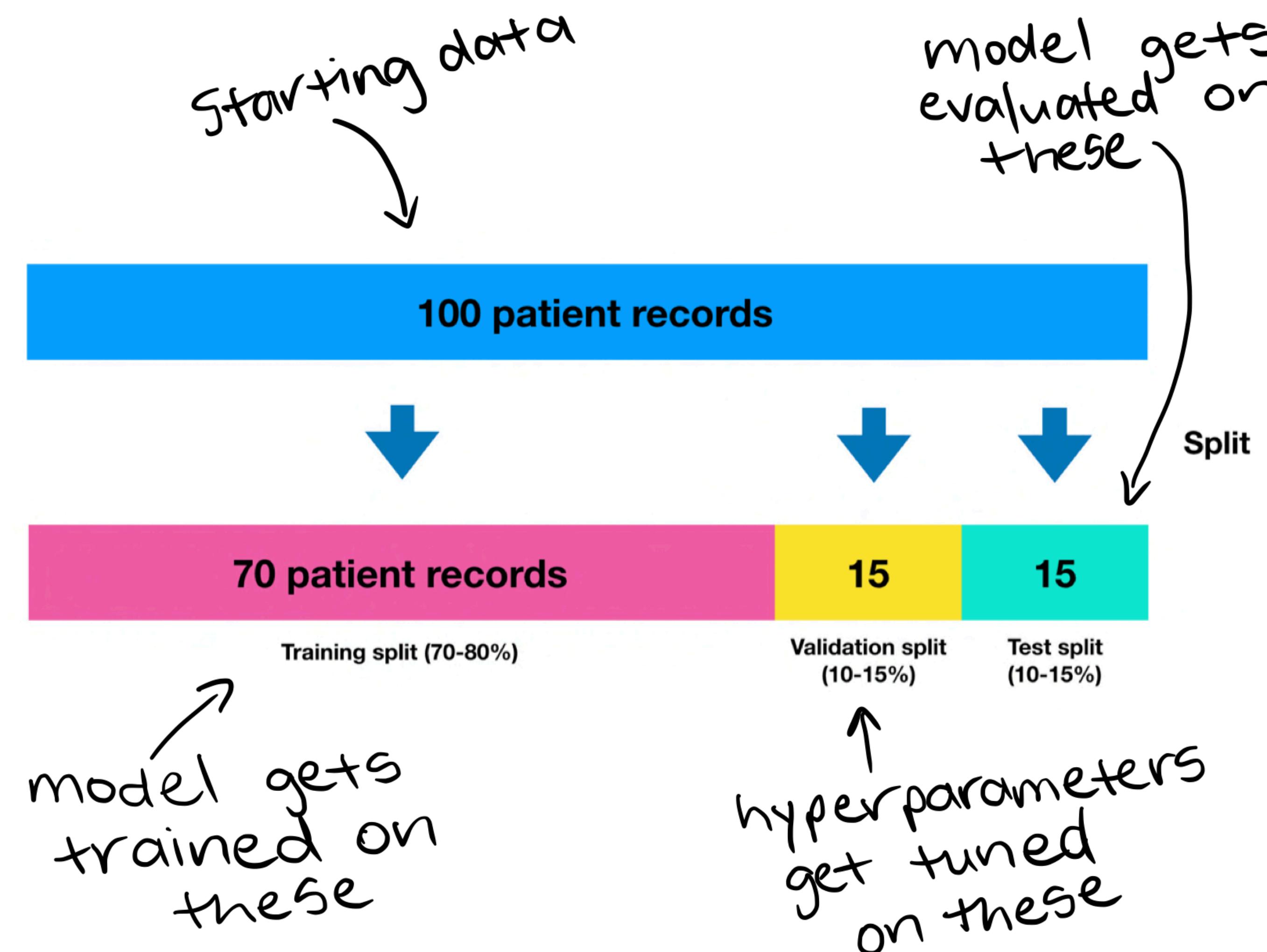


Cooking time: 1 hour  
Temperature: 180°C



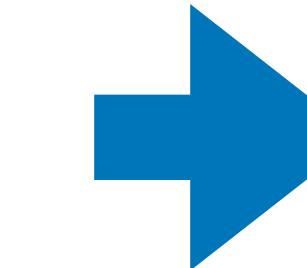
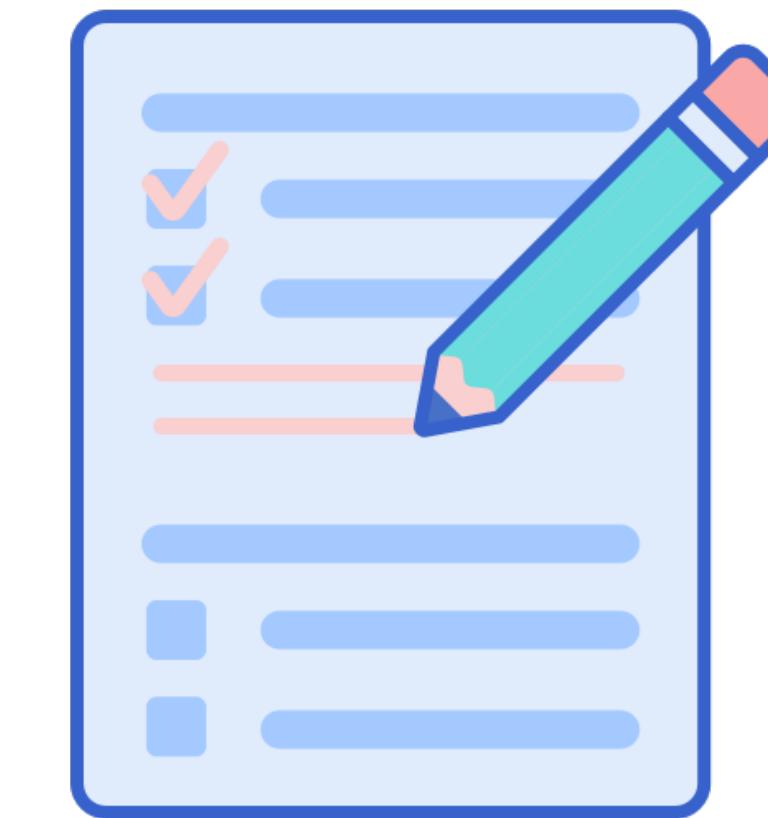
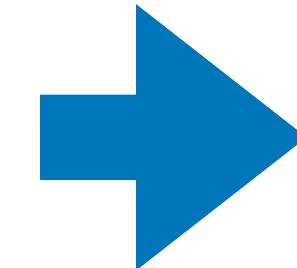
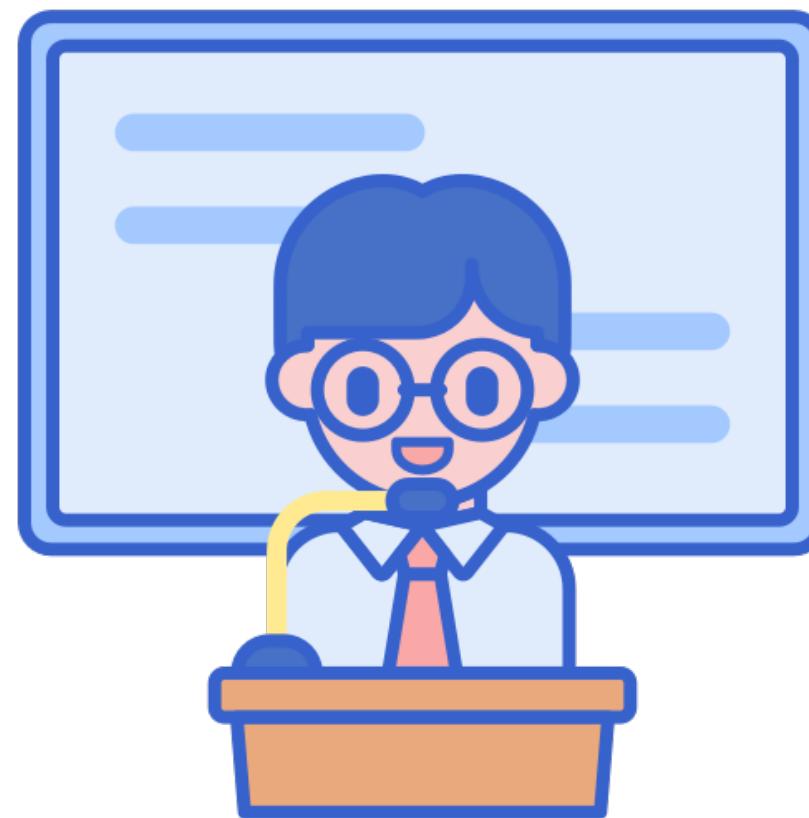
Cooking time: 1 hour  
Temperature: 200°C

# Tuning Hyperparameters by Hand



# The most important concept in machine learning

(the 3 sets)



**Course materials  
(training set)**

**Practice exam  
(validation set)**

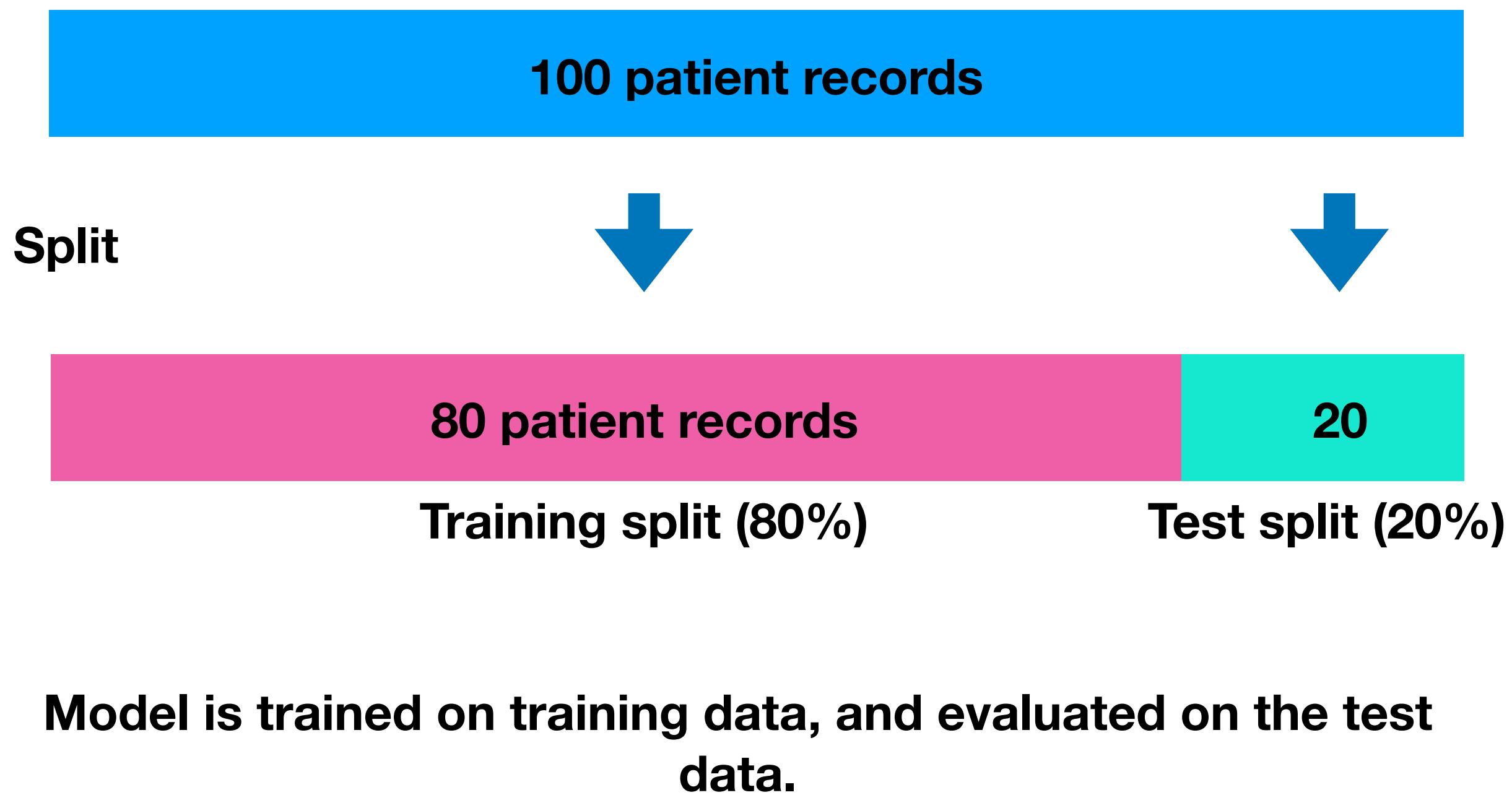
**Final exam  
(test set)**

**Generalization**

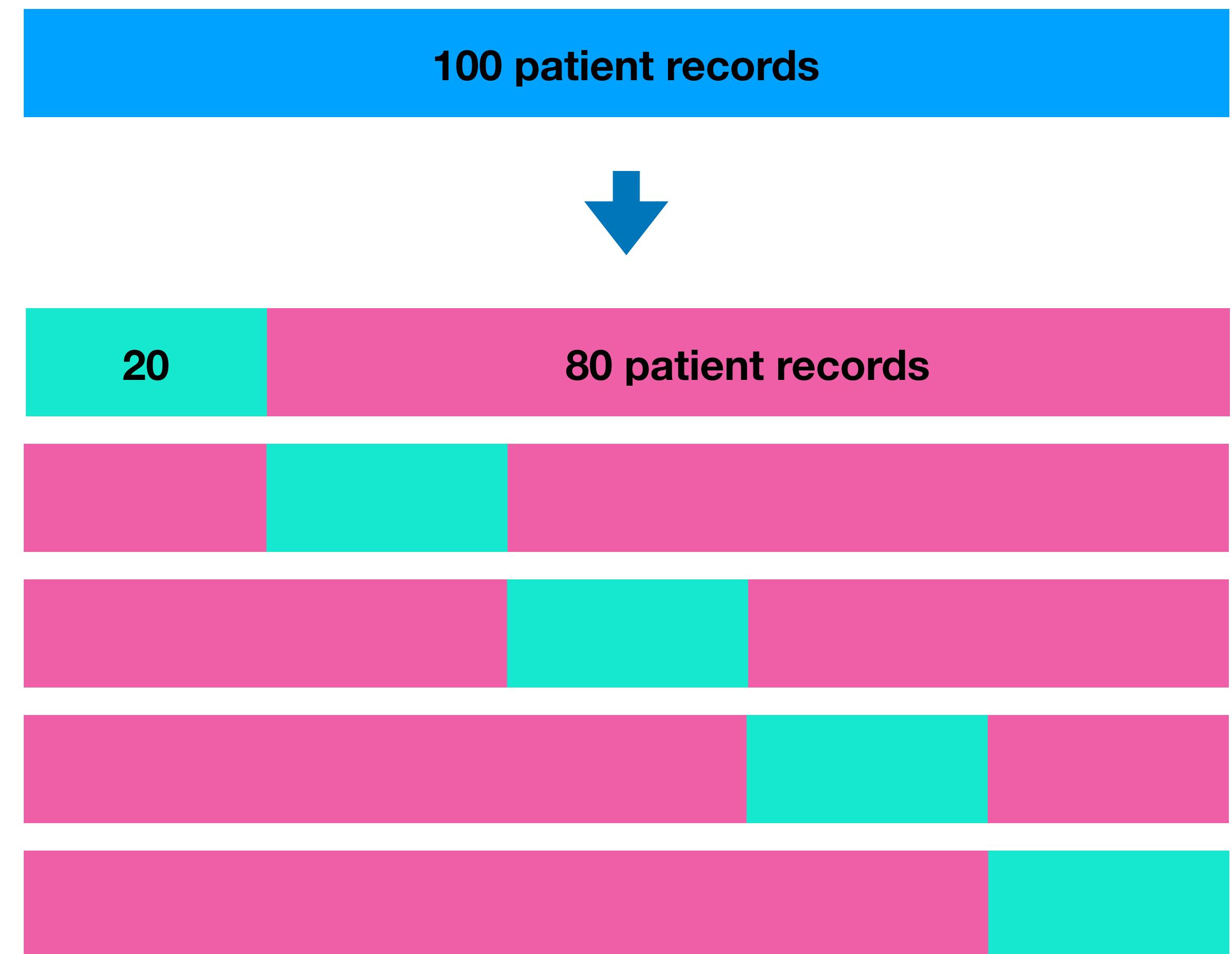
The ability for a machine learning model to perform well on data it hasn't seen before.

# Cross-validation

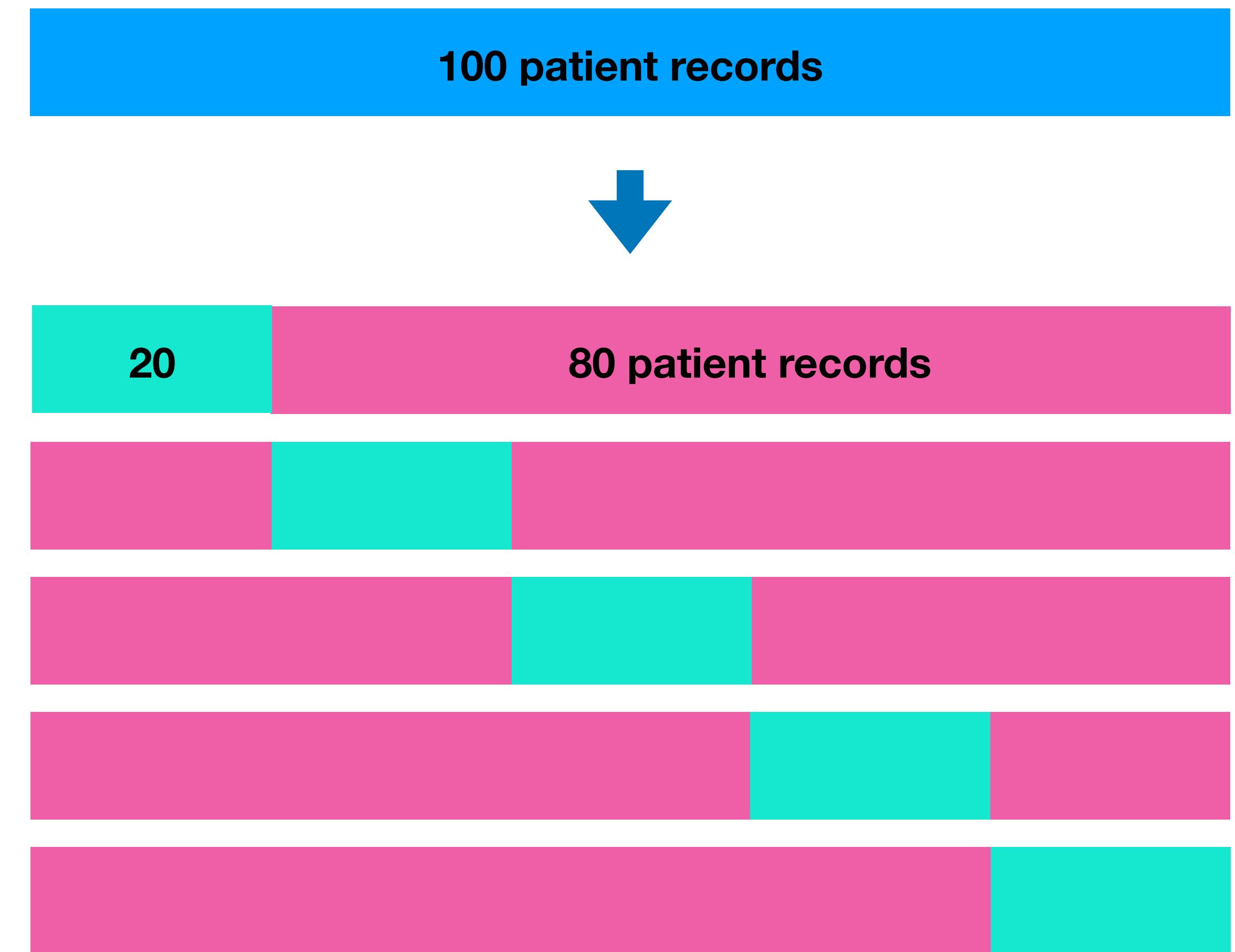
## Normal Train & Test Split



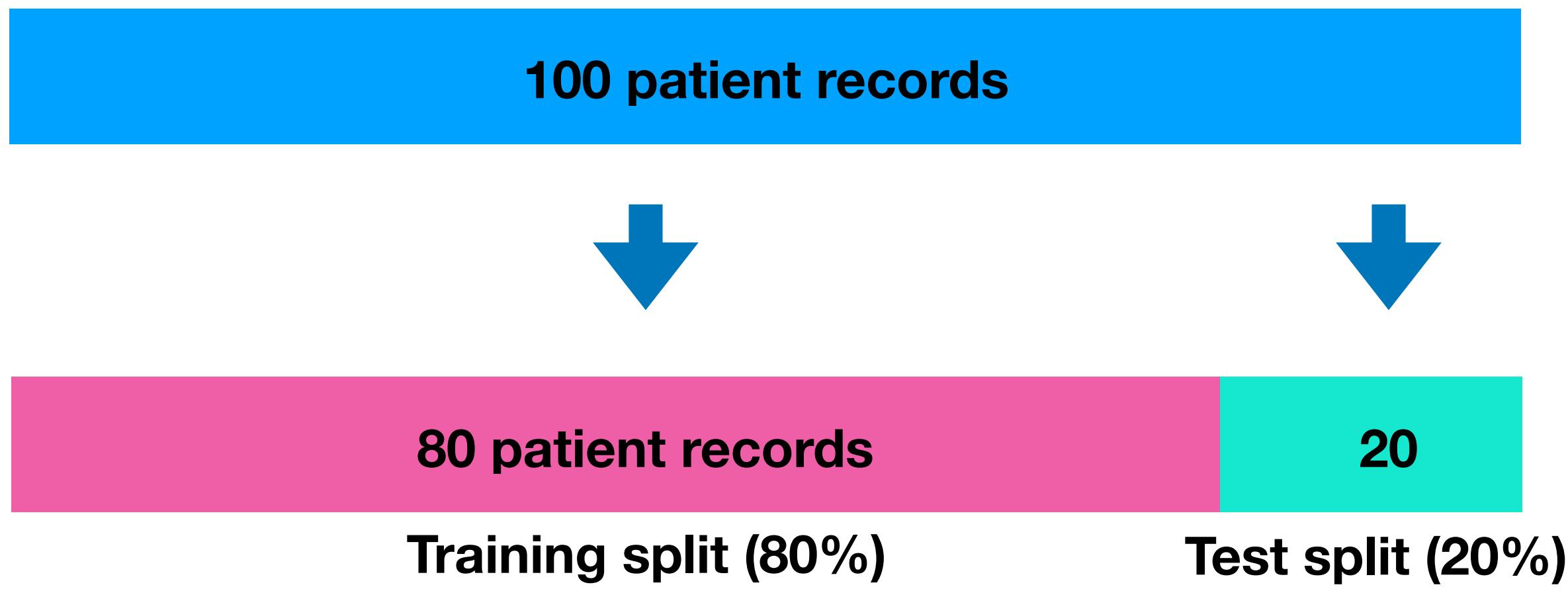
## 5-fold Cross-validation



## 5-fold Cross-validation



## Normal Train & Test Split



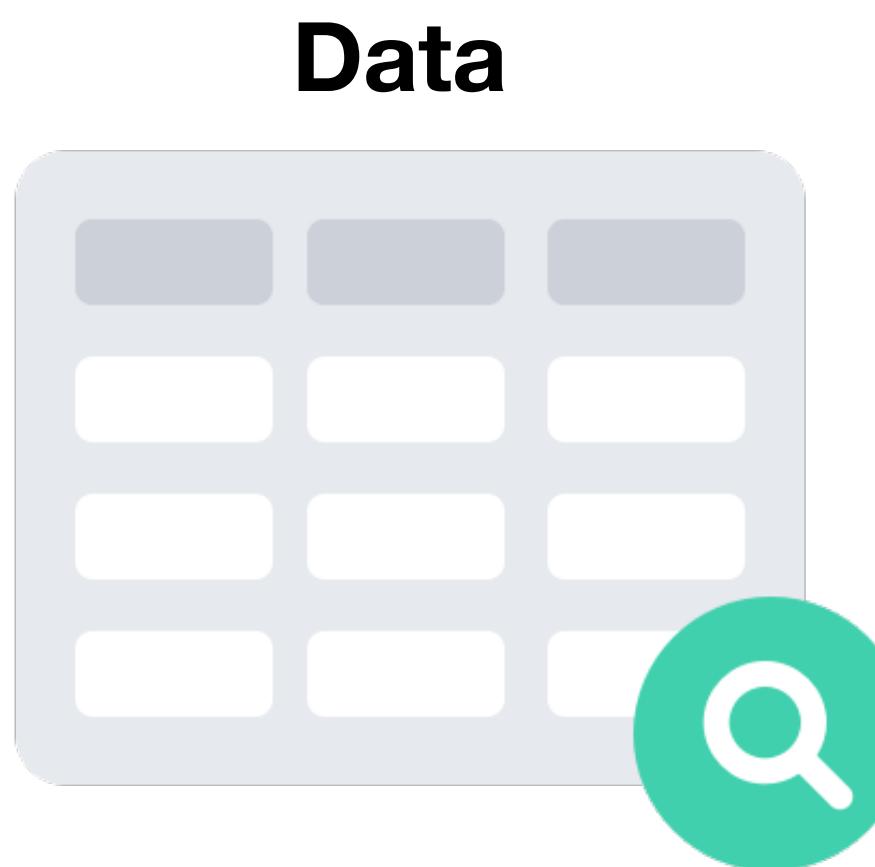
**Figure 1.0:** Model is trained on training data, and evaluated on the test data.

**Figure 2.0:** Model is trained on training data, and evaluated on the test data.

# Things to remember

- All data should be numerical
- There should be no missing values
- Manipulate the test set the same as the training set
- Never test on data you've trained on
- Tune hyperparameters on validation set OR use cross-validation
- One best performance metric doesn't mean the best model

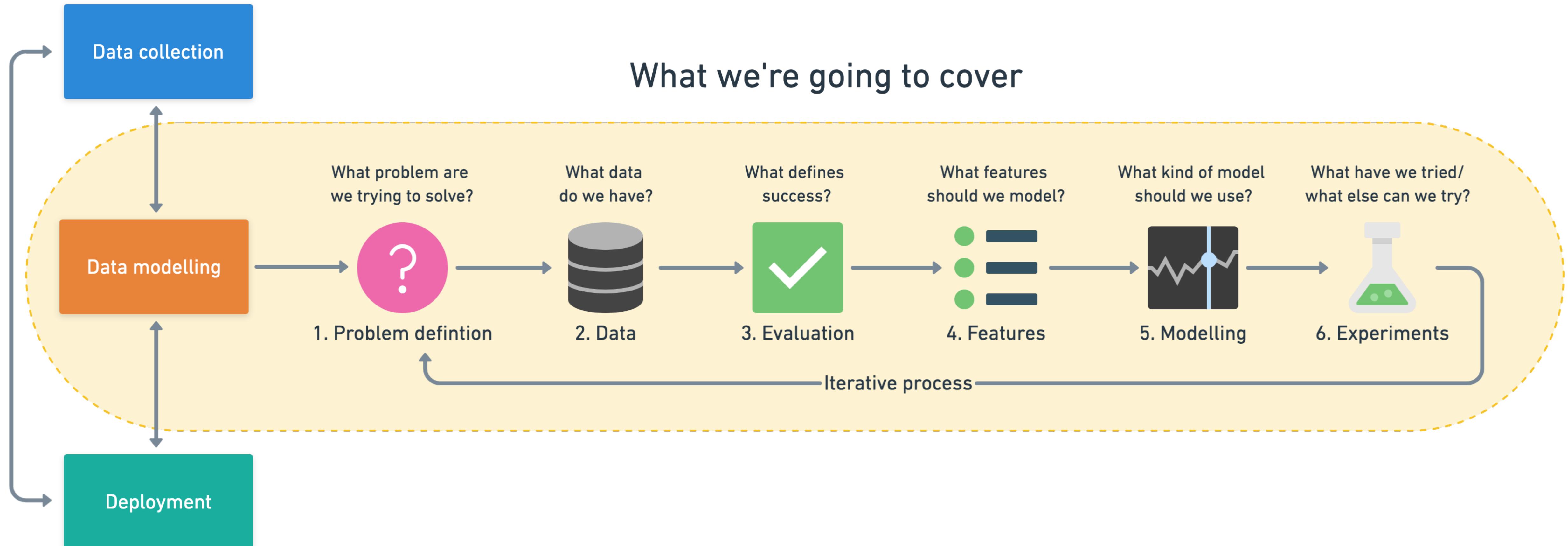
# What is structured data?



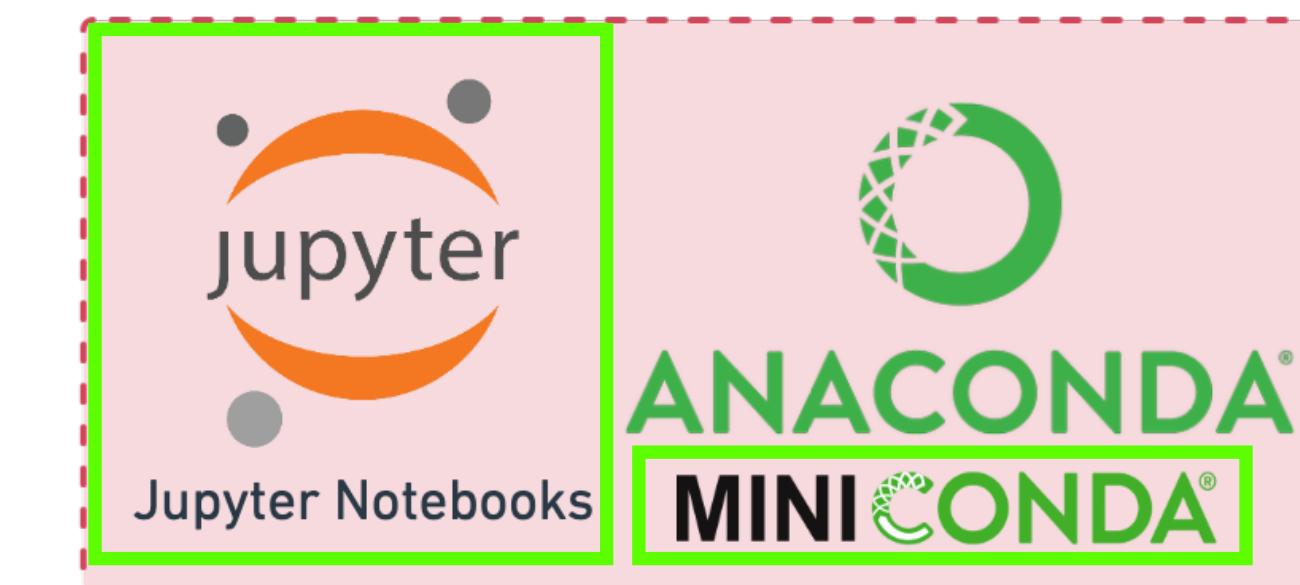
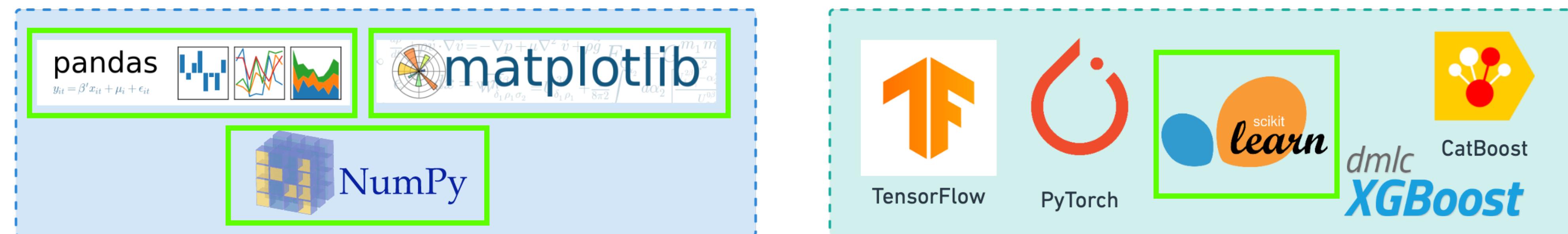
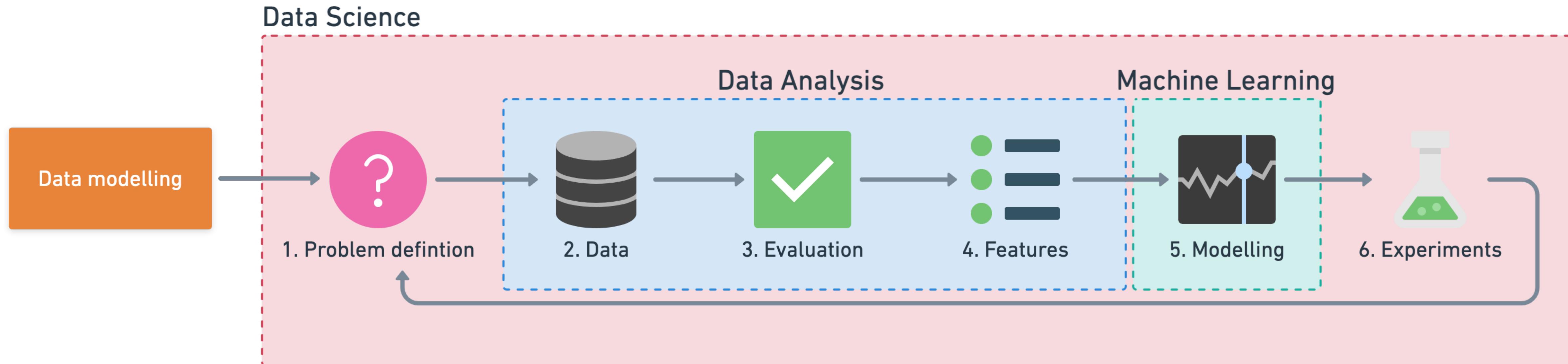
ID	Weight	Sex	Heart Rate	Chest pain	Heart disease?
4326	110Kg	M	81	4	Yes
5681	64Kg	F	61	1	No
7911	81Kg	M	57	0	No

Table 1.0 : Patient records

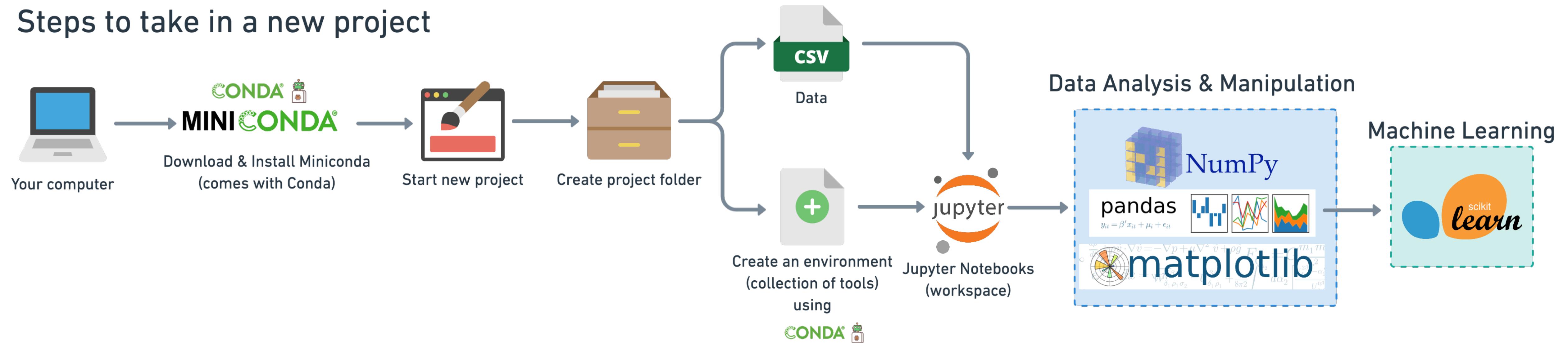
## Steps in a full machine learning project



# Tools you can use

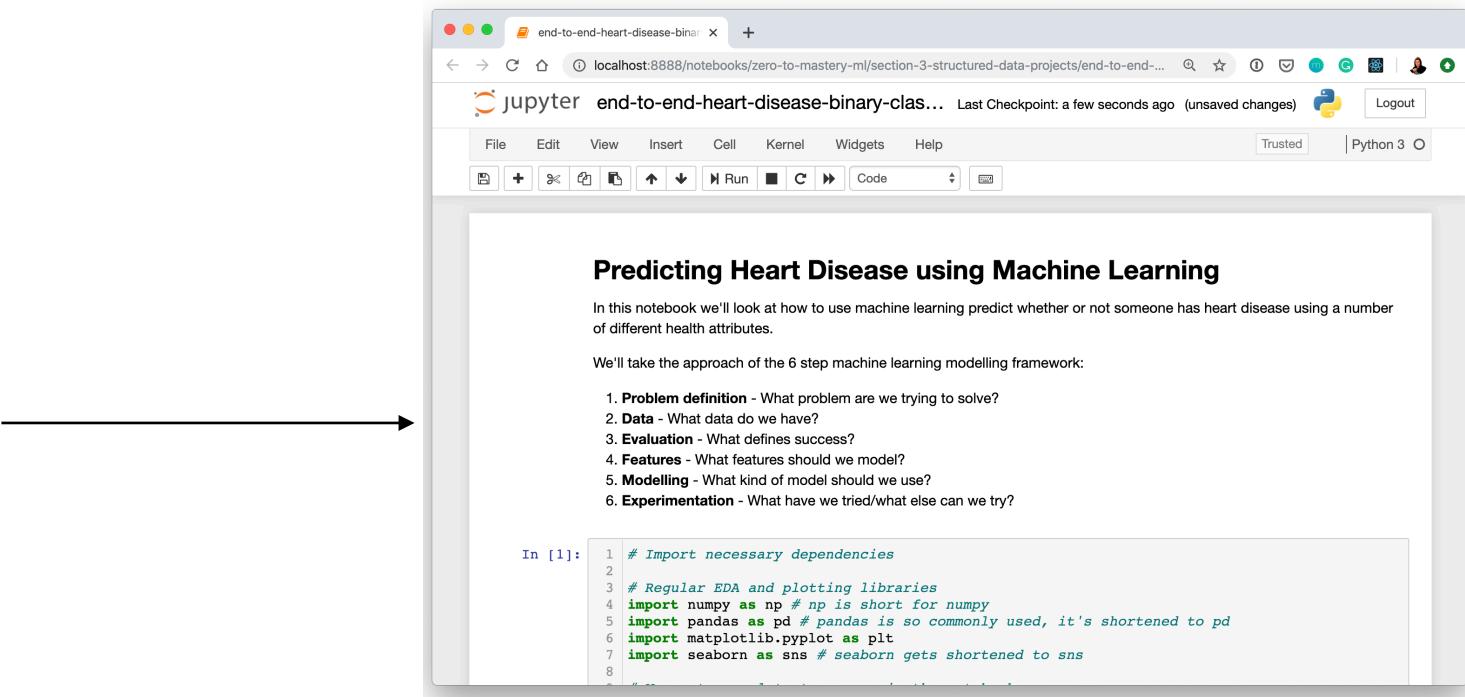


## Steps to take in a new project



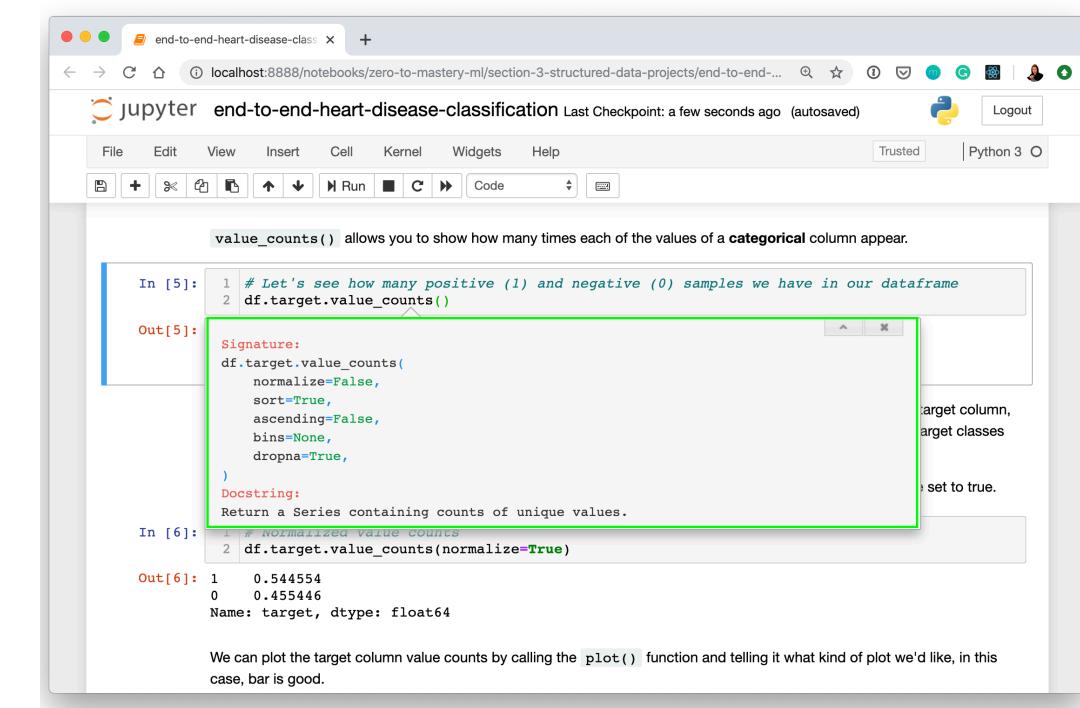
# Where can you get help?

- Follow along with the code



```
In [1]: 1 # Import necessary dependencies
2
3 # Regular EDA and plotting libraries
4 import numpy as np # np is short for numpy
5 import pandas as pd # pandas is so commonly used, it's shortened to pd
6 import matplotlib.pyplot as plt
7 import seaborn as sns # seaborn gets shortened to sns
```

- Try it for yourself



```
In [5]: 1 # Let's see how many positive (1) and negative (0) samples we have in our dataframe
2 df.target.value_counts()

Out[5]:
Signature:
df.target.value_counts(
    normalize=False,
    sort=True,
    ascending=False,
    bins=None,
    dropna=True,
)
Docstring:
Return a Series containing counts of unique values.
In [6]: 1 df.target.value_counts(normalize=True)

Out[6]: 1 0.544554
0 0.455446
Name: target, dtype: float64
```

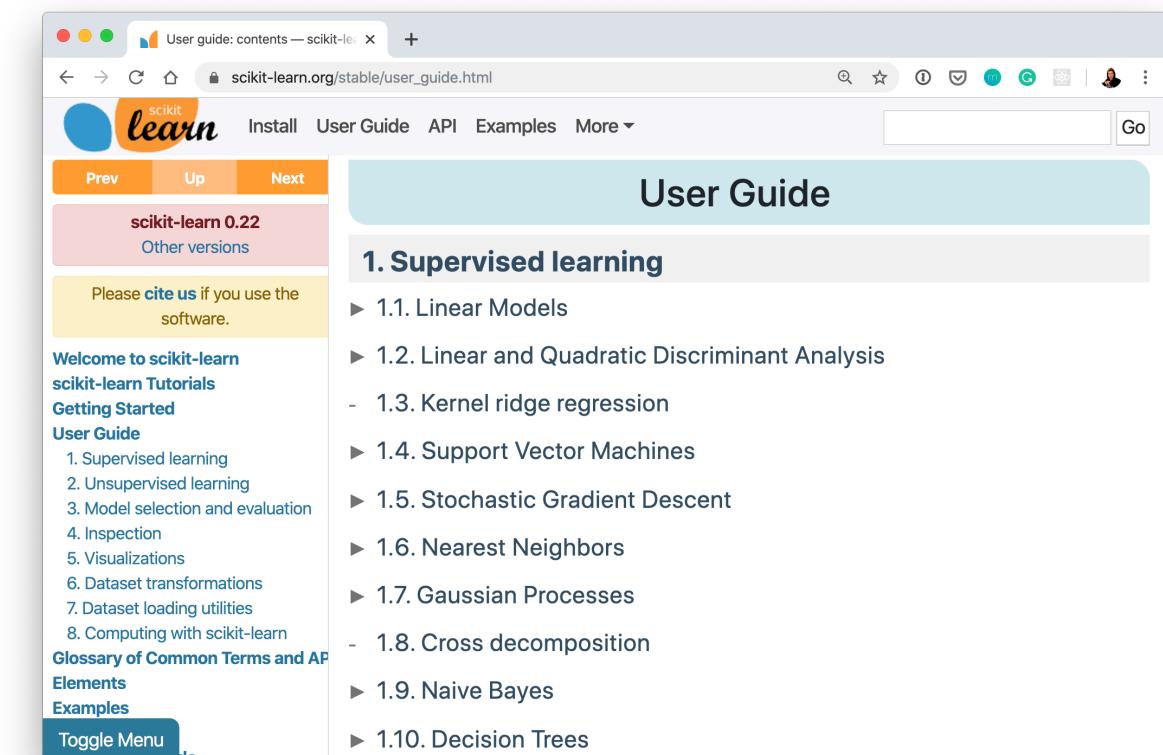
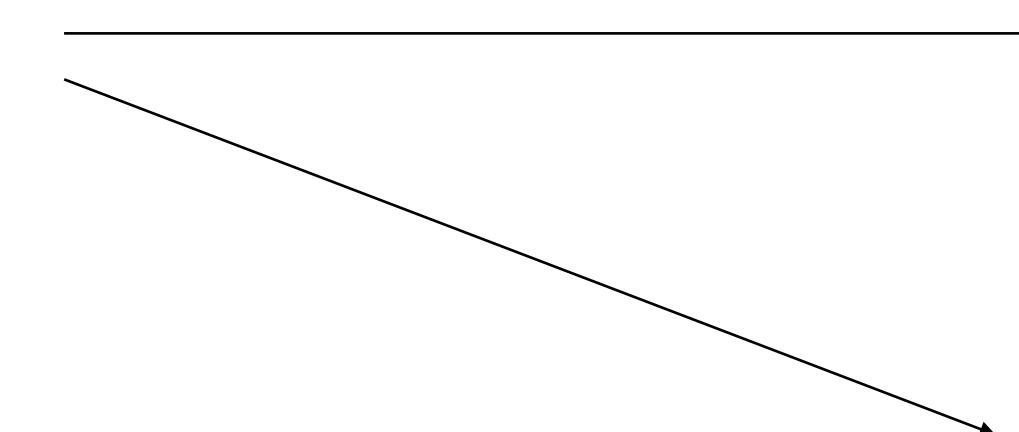
- Press SHIFT + TAB to read the docstring



- Search for it

- Try again

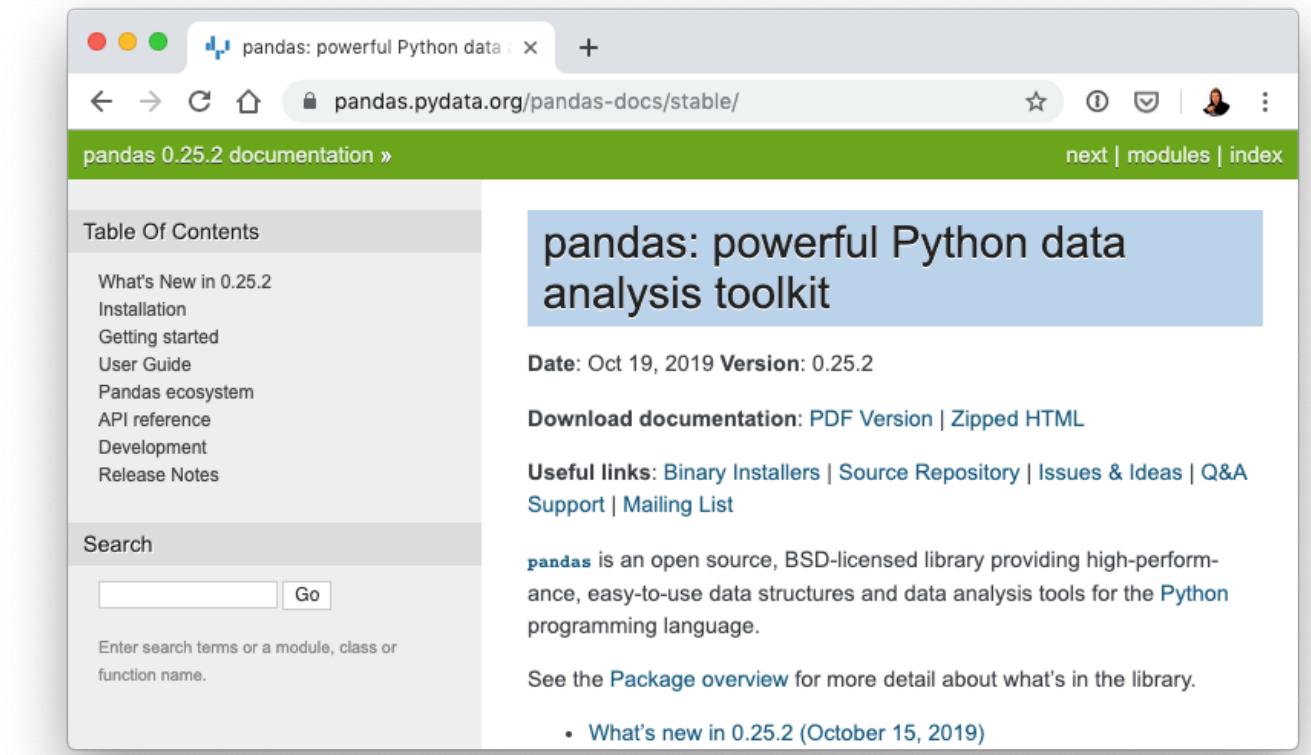
- Ask



## User Guide

### 1. Supervised learning

- ▶ 1.1. Linear Models
- ▶ 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- ▶ 1.4. Support Vector Machines
- ▶ 1.5. Stochastic Gradient Descent
- ▶ 1.6. Nearest Neighbors
- ▶ 1.7. Gaussian Processes
- 1.8. Cross decomposition
- ▶ 1.9. Naive Bayes
- ▶ 1.10. Decision Trees



## pandas: powerful Python data analysis toolkit

Date: Oct 19, 2019 Version: 0.25.2

Download documentation: PDF Version | Zipped HTML

Useful links: Binary Installers | Source Repository | Issues & Ideas | Q&A Support | Mailing List

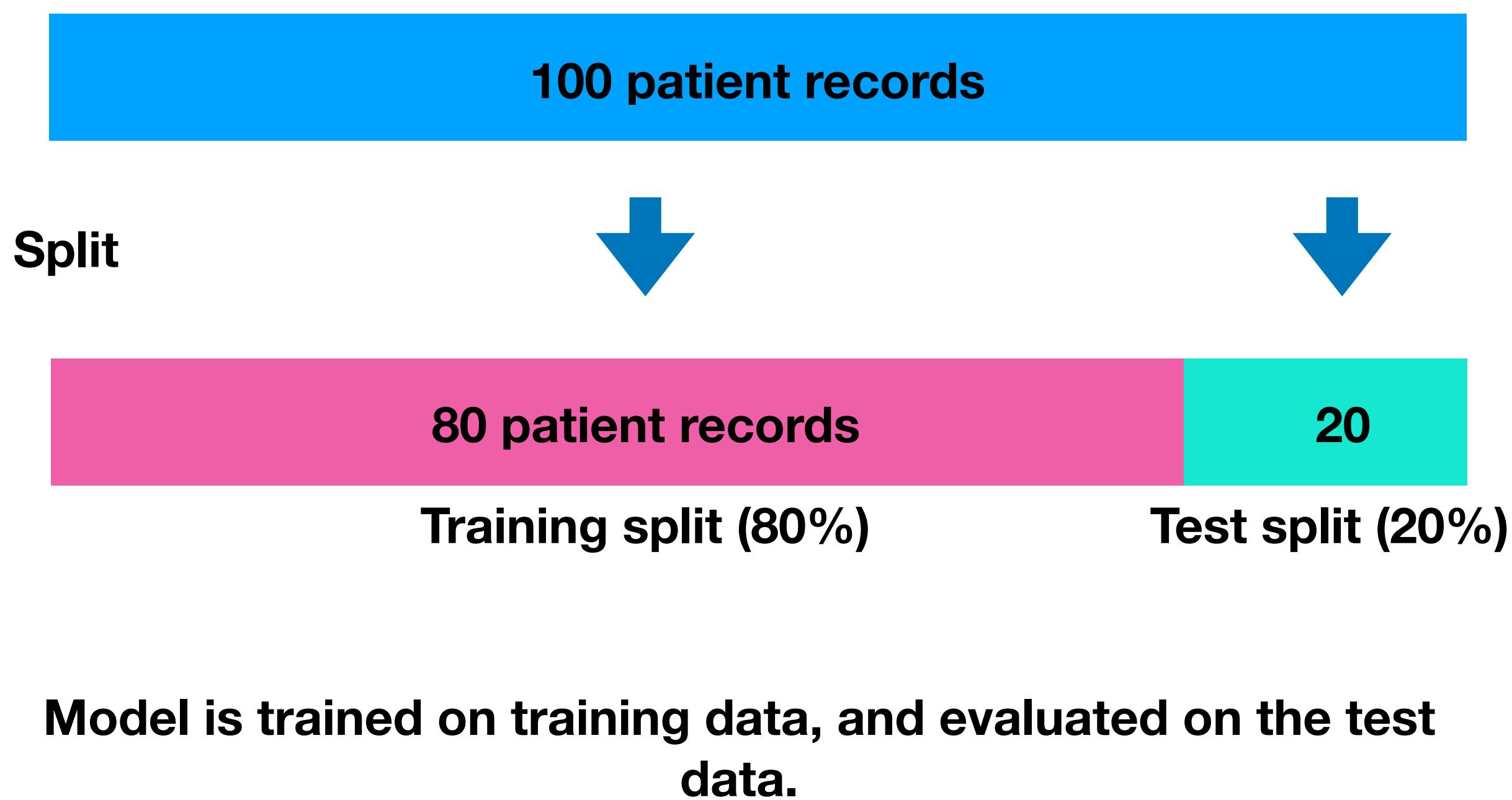
pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

See the Package overview for more detail about what's in the library.

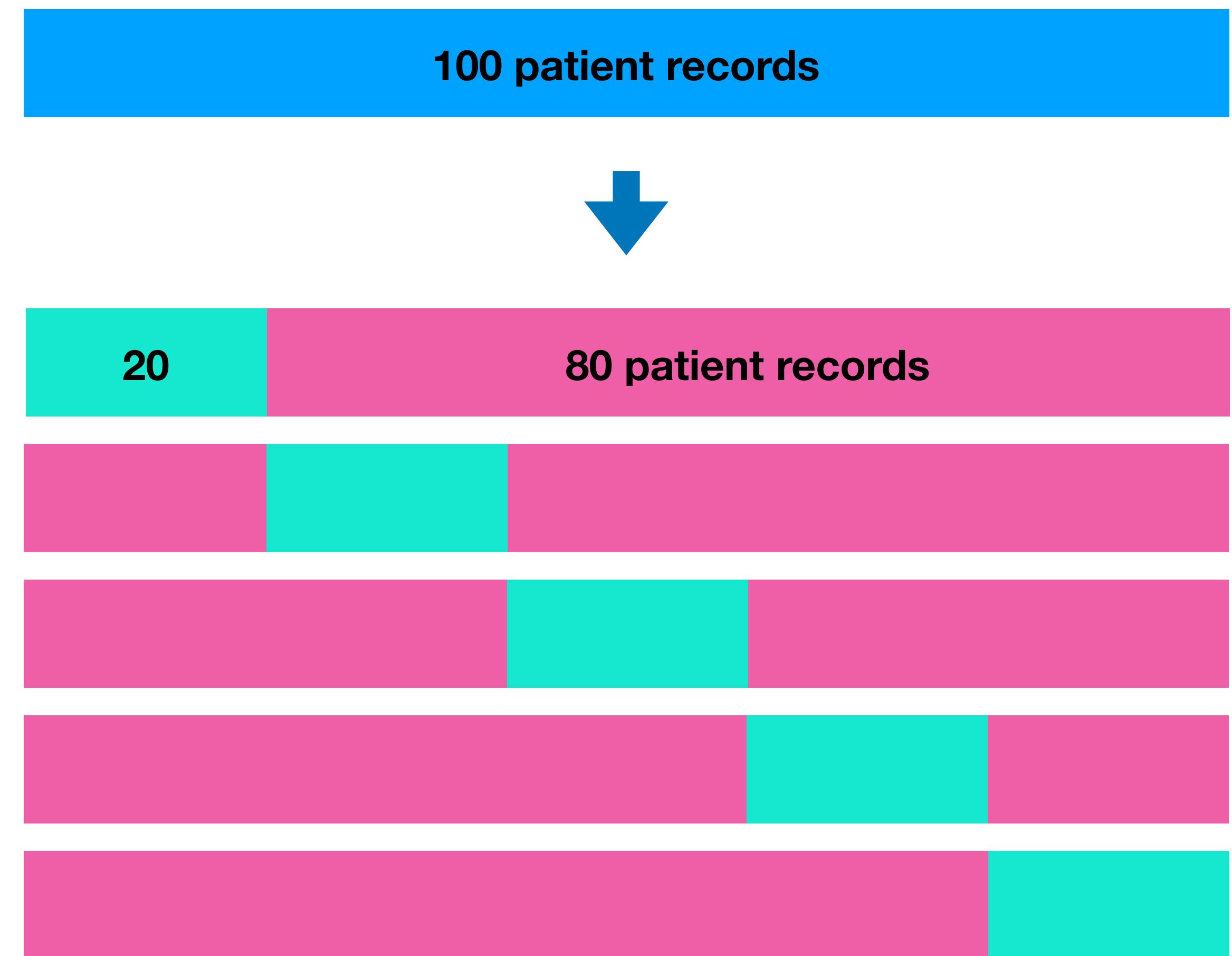
- What's new in 0.25.2 (October 15, 2019)

# Cross-validation

## Normal Train & Test Split



## 5-fold Cross-validation



Model is trained on 5 different versions of training data, and evaluated on 5 different versions of the test data.

# Classification and Regression metrics

## Classification

**Accuracy**

Precision

Recall

F1

## Regression

**R<sup>2</sup> (r-squared)**

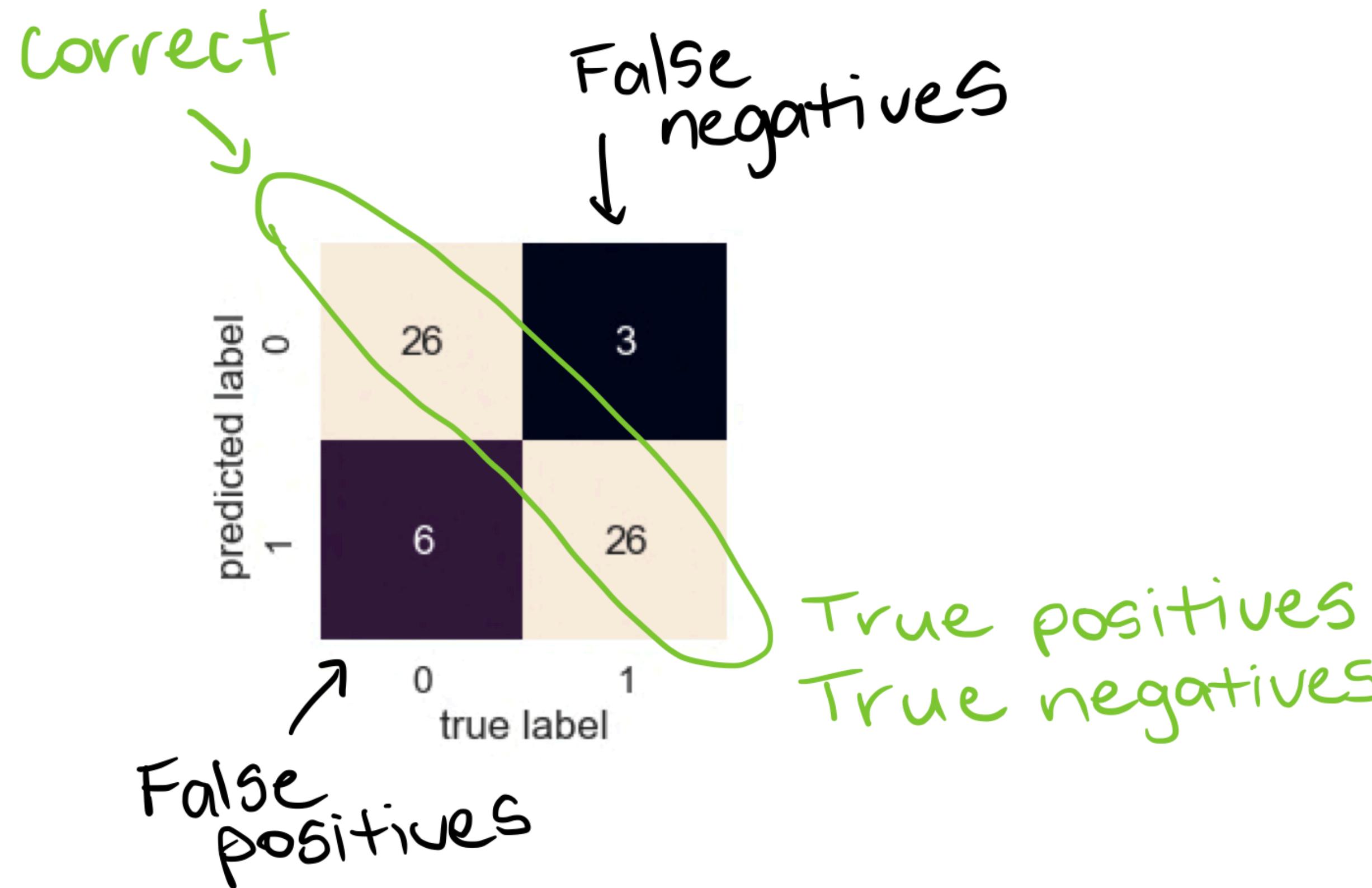
Mean absolute error (MAE)

Mean squared error (MSE)

Root mean squared error (RMSE)

**Bold** = default evaluation in Scikit-Learn

# Confusion matrix anatomy



- **True positive** = model predicts 1 when truth is 1
- **False positive** = model predicts 1 when truth is 0
- **True negative** = model predicts 0 when truth is 0
- **False negative** = model predicts 0 when truth is 1

# Classification report anatomy

```
1 from sklearn.metrics import classification_report  
2  
3 print(classification_report(y_test, y_preds))
```

	precision	recall	f1-score	support
0	0.81	0.90	0.85	29
1	0.90	0.81	0.85	32
accuracy			0.85	61
macro avg	0.85	0.85	0.85	61
weighted avg	0.86	0.85	0.85	61

- **Precision** - Indicates the proportion of positive identifications (model predicted class 1) which were actually correct. A model which produces no false positives has a precision of 1.0.
- **Recall** - Indicates the proportion of actual positives which were correctly classified. A model which produces no false negatives has a recall of 1.0.
- **F1 score** - A combination of precision and recall. A perfect model achieves an F1 score of 1.0.
- **Support** - The number of samples each metric was calculated on.
- **Accuracy** - The accuracy of the model in decimal form. Perfect accuracy is equal to 1.0.
- **Macro avg** - Short for macro average, the average precision, recall and F1 score between classes. Macro avg doesn't class imbalance into effort, so if you do have class imbalances, pay attention to this metric.
- **Weighted avg** - Short for weighted average, the weighted average precision, recall and F1 score between classes. Weighted means each metric is calculated with respect to how many samples there are in each class. This metric will favour the majority class (e.g. will give a high value when one class out performs another due to having more samples).

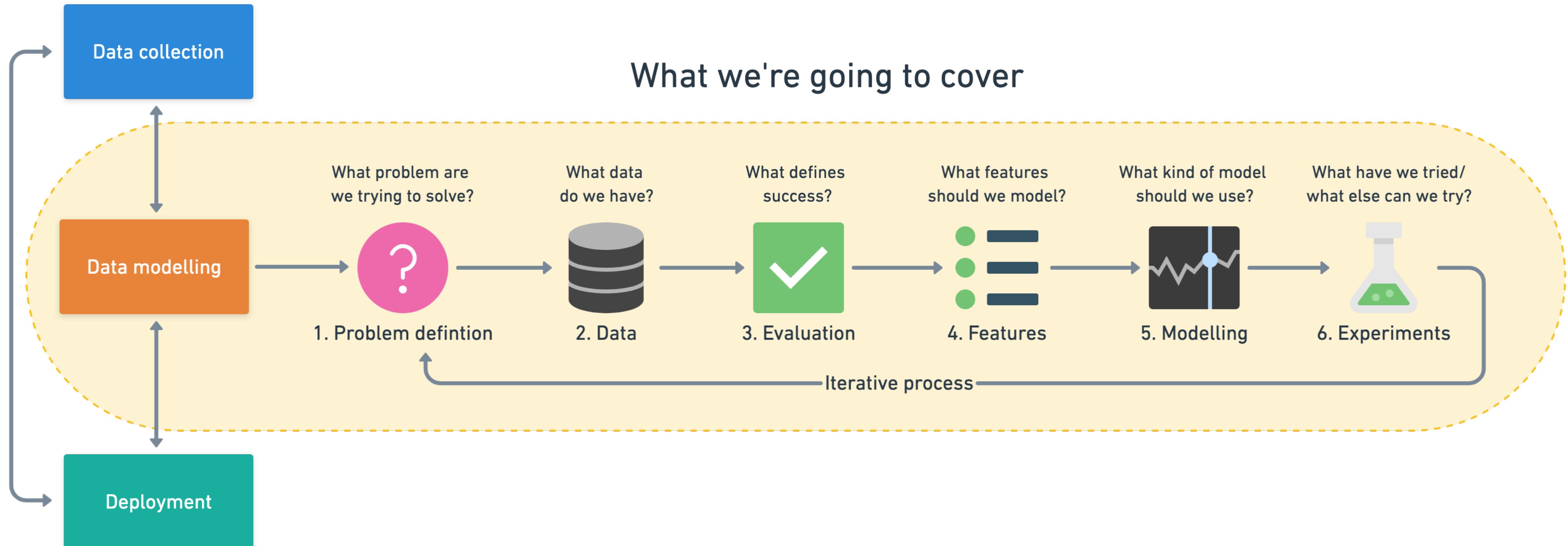
# Which classification metric should you use?

- **Accuracy** is a good measure to start with if all classes are balanced (e.g. same amount of samples which are labelled with 0 or 1).
- **Precision** and **recall** become more important when classes are imbalanced.
- If false positive predictions are worse than false negatives, aim for higher precision.
- If false negative predictions are worse than false positives, aim for higher recall.
- **F1-score** is a combination of precision and recall.

# **Structured Data Project 2: Predicting the sale price of Bulldozers (regression)**

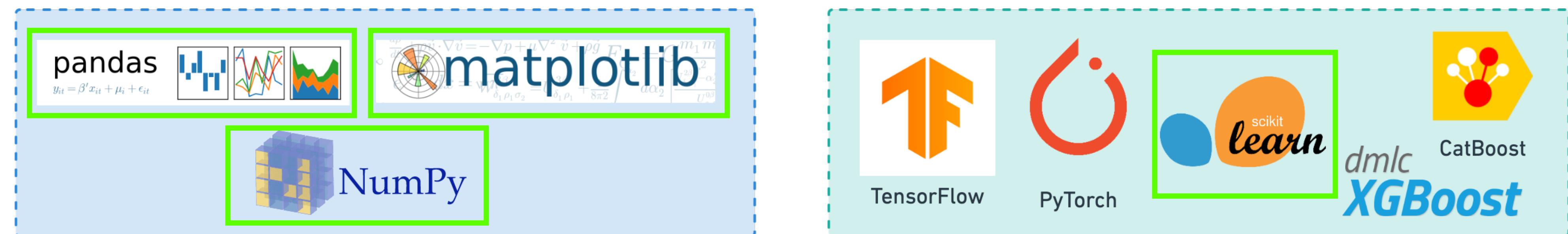
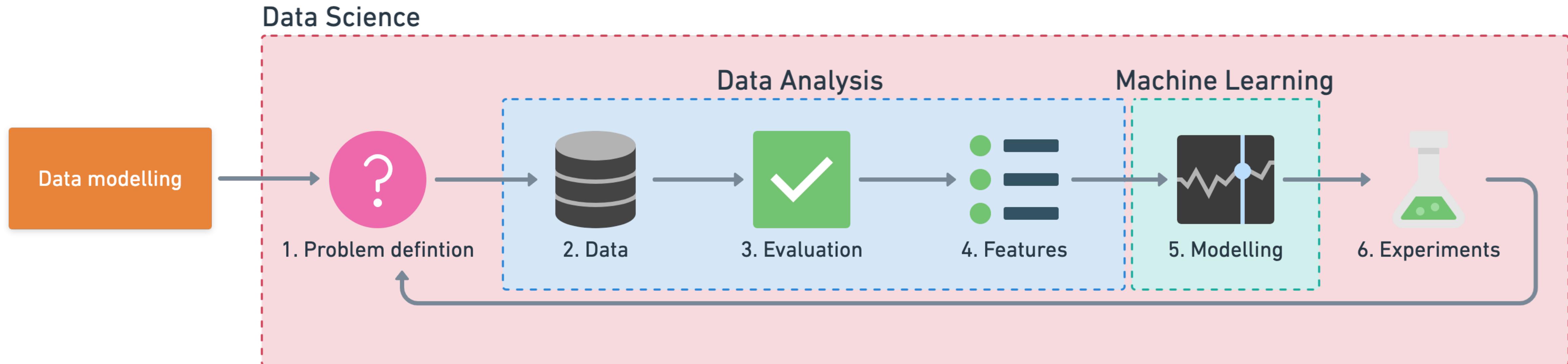


## Steps in a full machine learning project

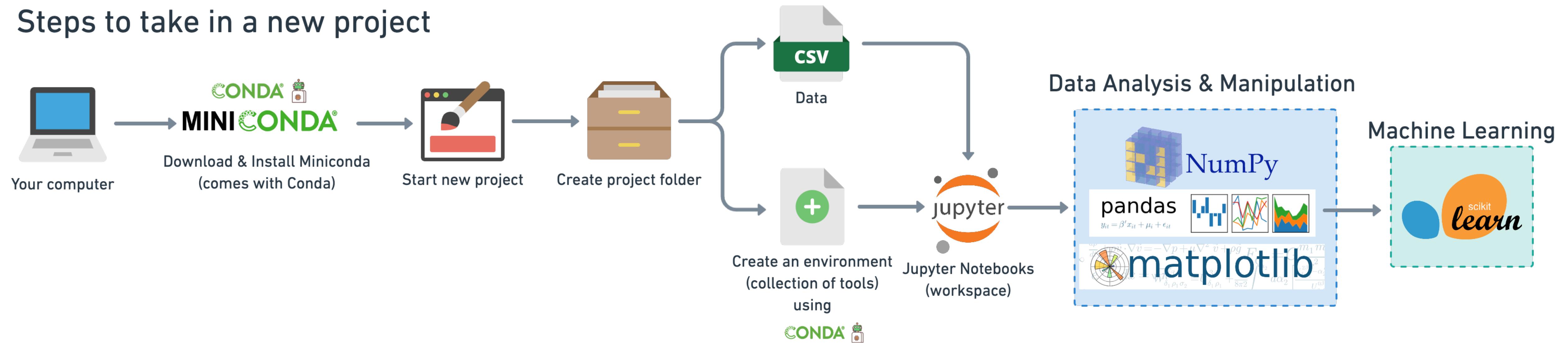


## What we're going to cover

# Tools you can use

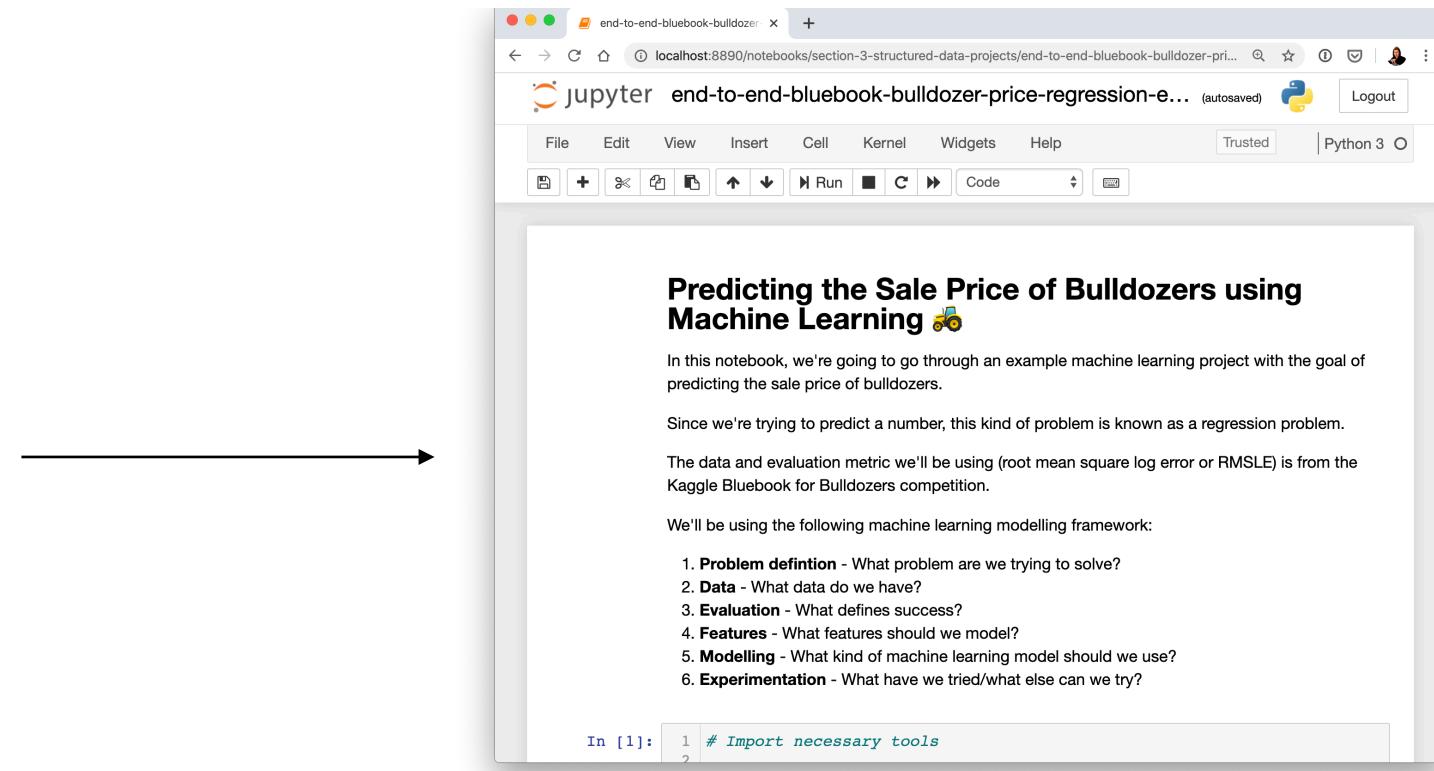


## Steps to take in a new project



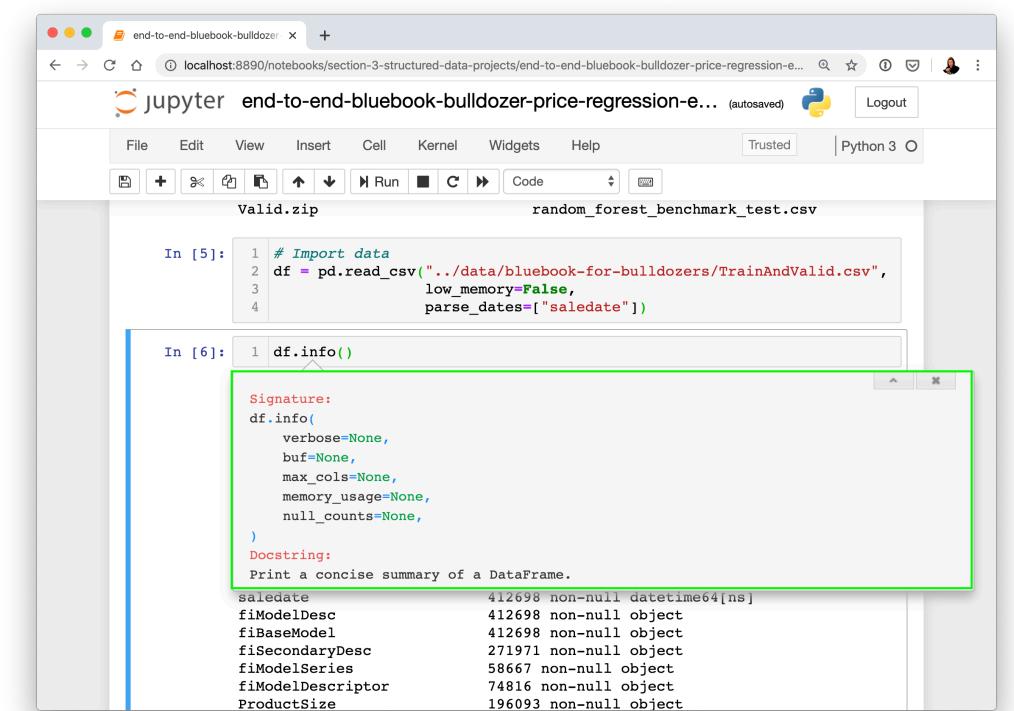
# Where can you get help?

- Follow along with the code



```
In [1]: 1 # Import necessary tools
```

- Try it for yourself



```
In [5]: 1 # Import data
2 df = pd.read_csv("../data/bluebook-for-bulldozers/TrainAndValid.csv",
3                  low_memory=False,
4                  parse_dates=['saledate'])

In [6]: 1 df.info()
```

```
Signature: df.info()
    verbose=None,
    buffer=None,
    max_col=None,
    memory_usage=None,
    null_counts=None,
)
Docstring:
Print a concise summary of a DataFrame.

saledate        412698 non-null datetime64[ns]
fimodeldesc    412698 non-null object
fimodel       412698 non-null object
fissecondarydesc  271971 non-null object
fimodelseries   58667 non-null object
fimodeldescriptor  74816 non-null object
ProductSize      196093 non-null object
```

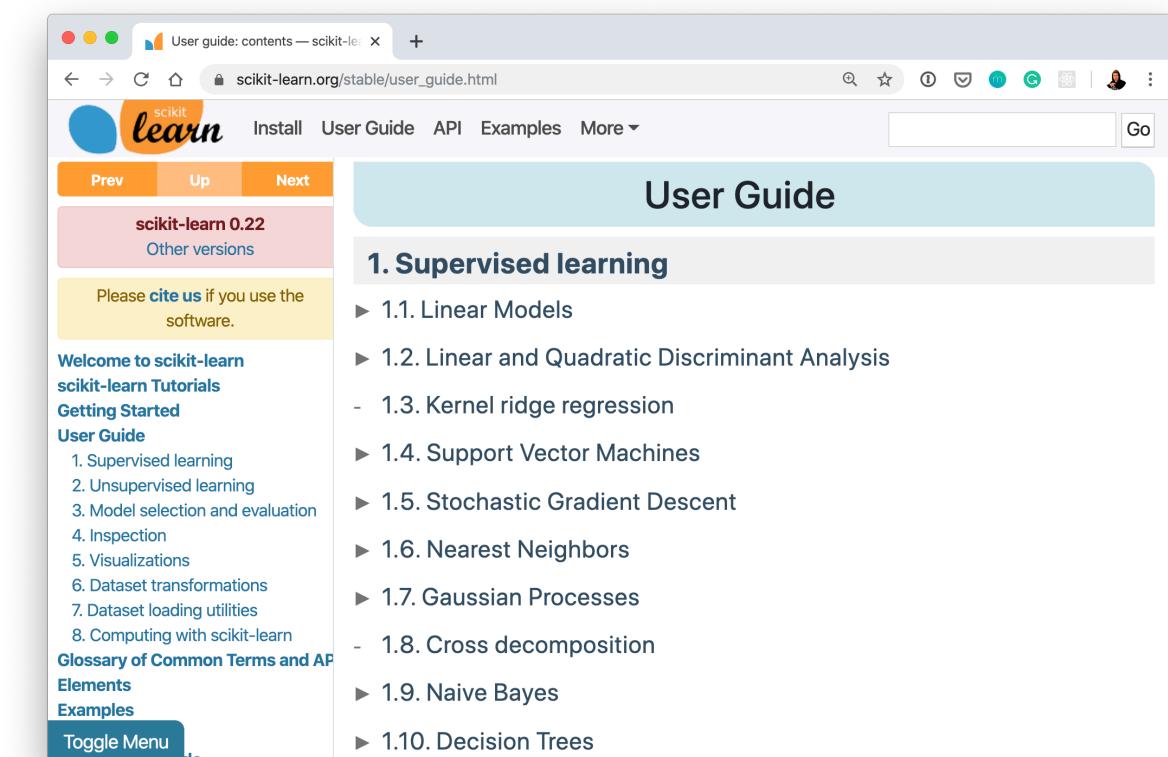
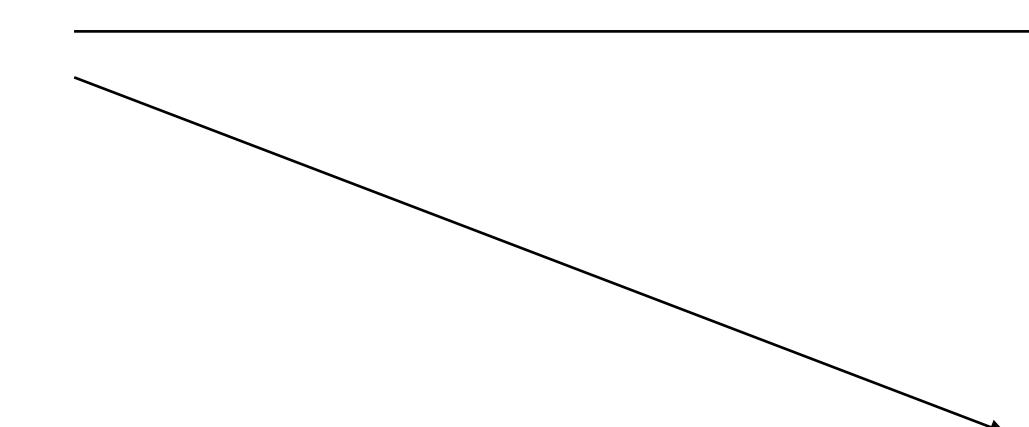
- Press SHIFT + TAB to read the docstring

- Search for it



- Try again

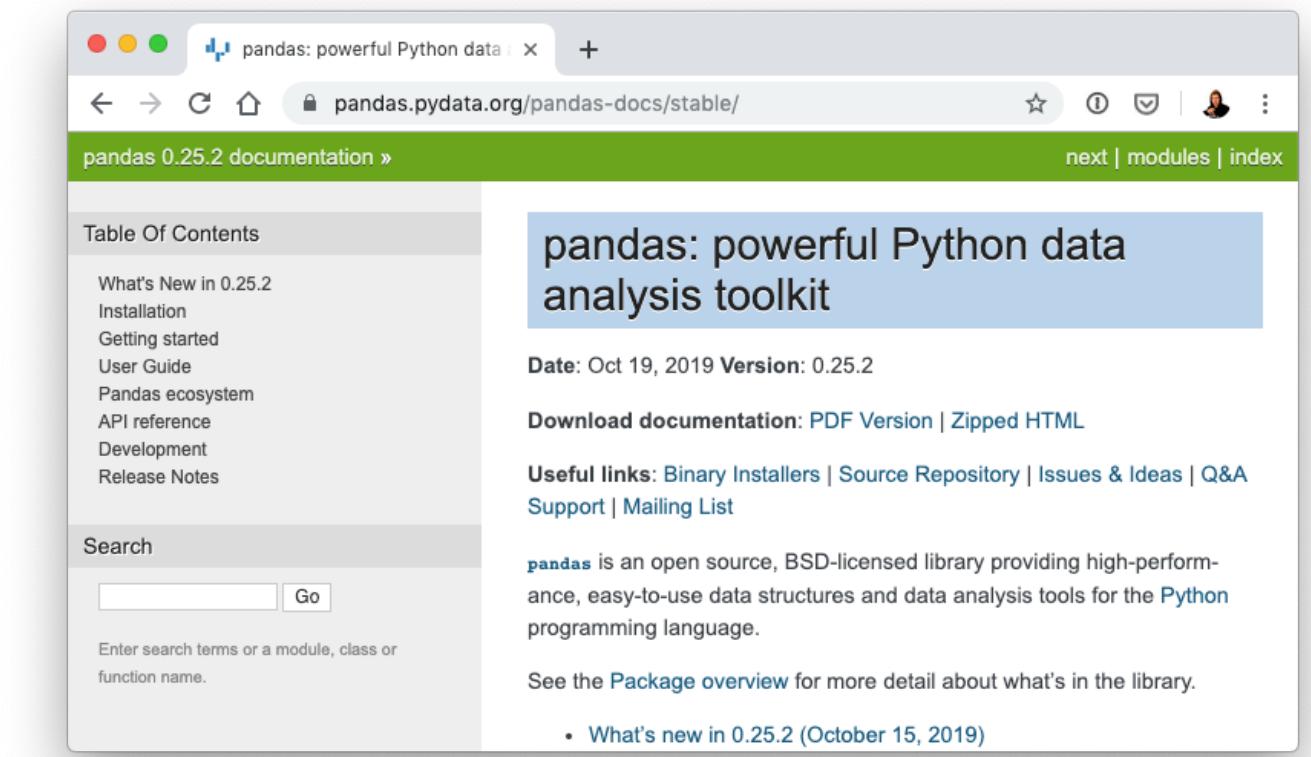
- Ask



## User Guide

### 1. Supervised learning

- ▶ 1.1. Linear Models
- ▶ 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- ▶ 1.4. Support Vector Machines
- ▶ 1.5. Stochastic Gradient Descent
- ▶ 1.6. Nearest Neighbors
- ▶ 1.7. Gaussian Processes
- 1.8. Cross decomposition
- ▶ 1.9. Naive Bayes
- ▶ 1.10. Decision Trees



## pandas: powerful Python data analysis toolkit

Date: Oct 19, 2019 Version: 0.25.2

Download documentation: PDF Version | Zipped HTML

Useful links: Binary Installers | Source Repository | Issues & Ideas | Q&A Support | Mailing List

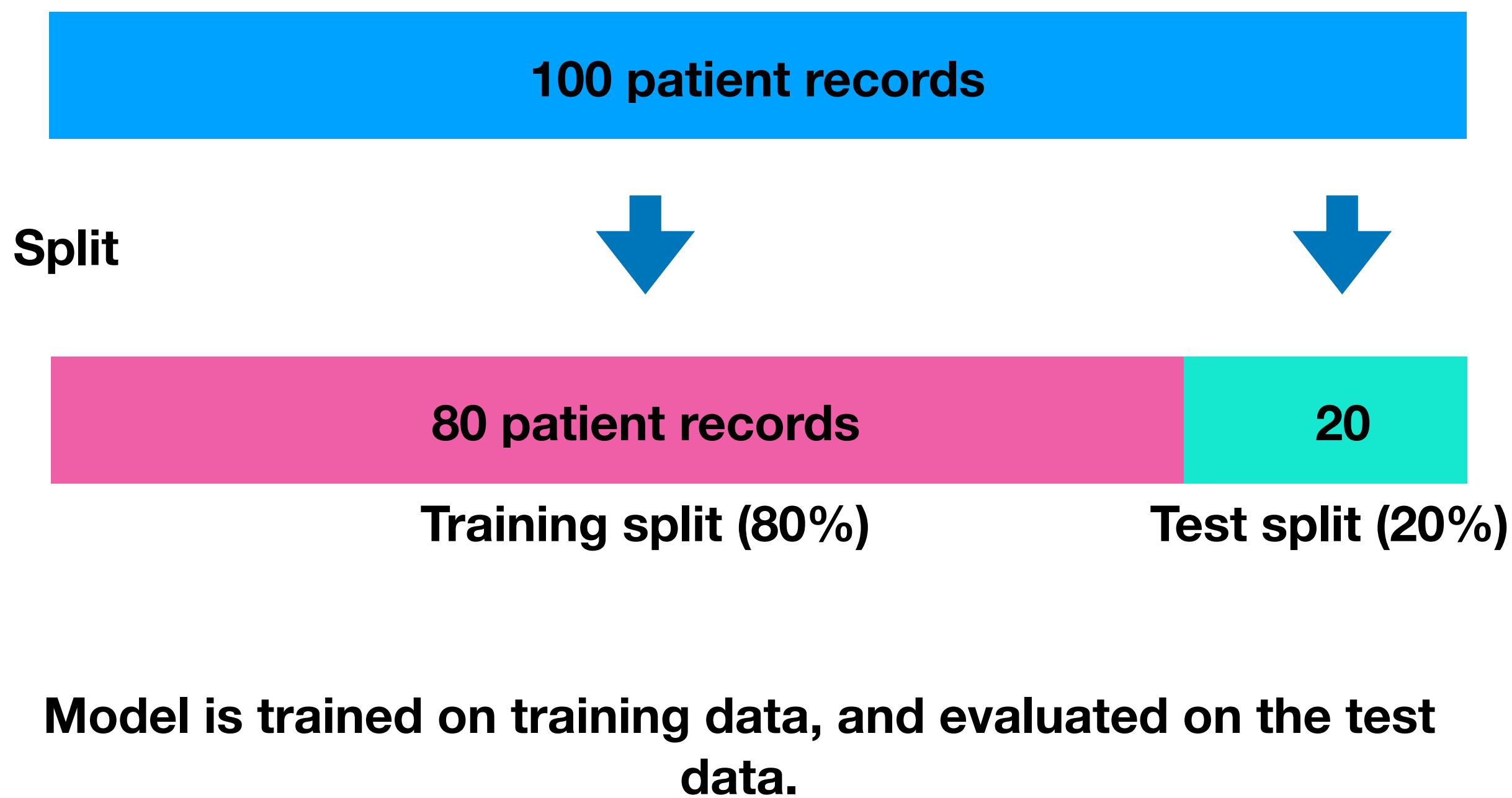
pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

See the Package overview for more detail about what's in the library.

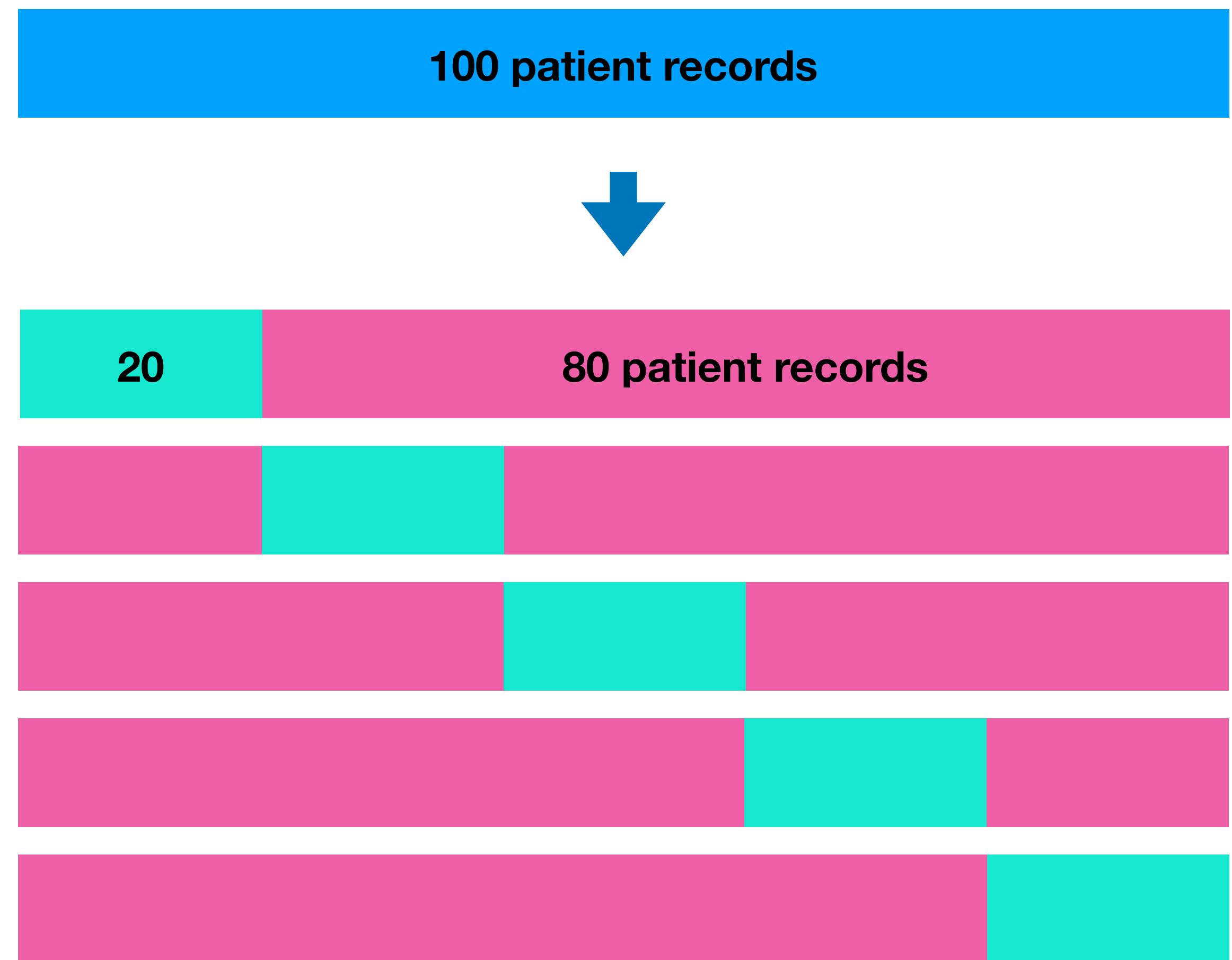
- What's new in 0.25.2 (October 15, 2019)

# Cross-validation

## Normal Train & Test Split



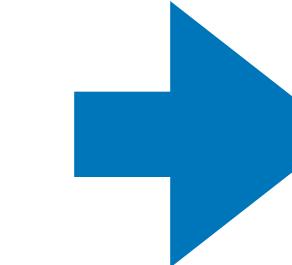
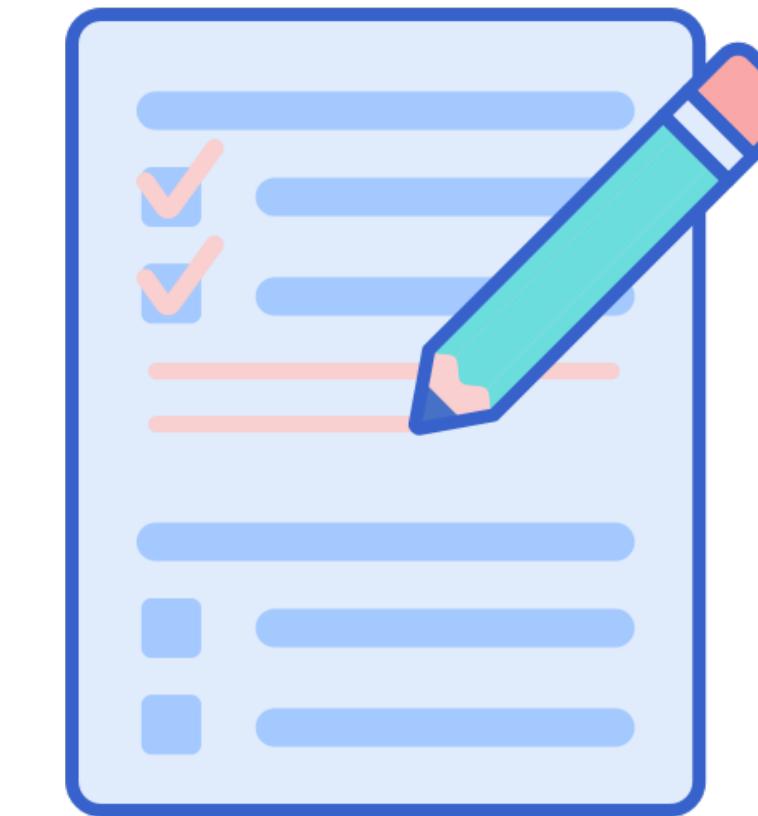
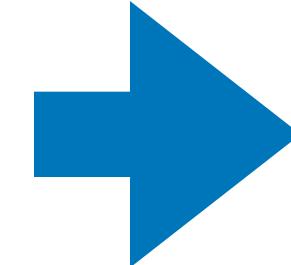
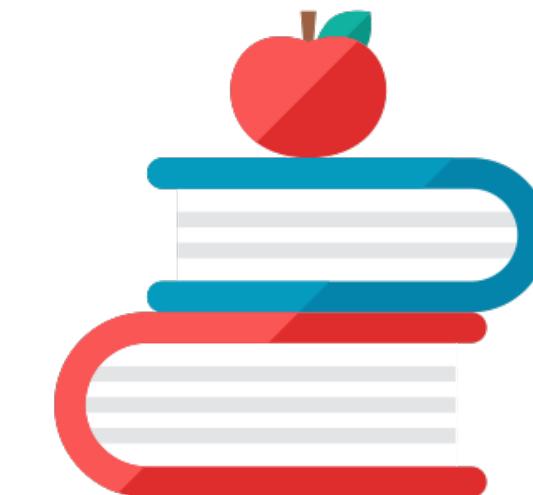
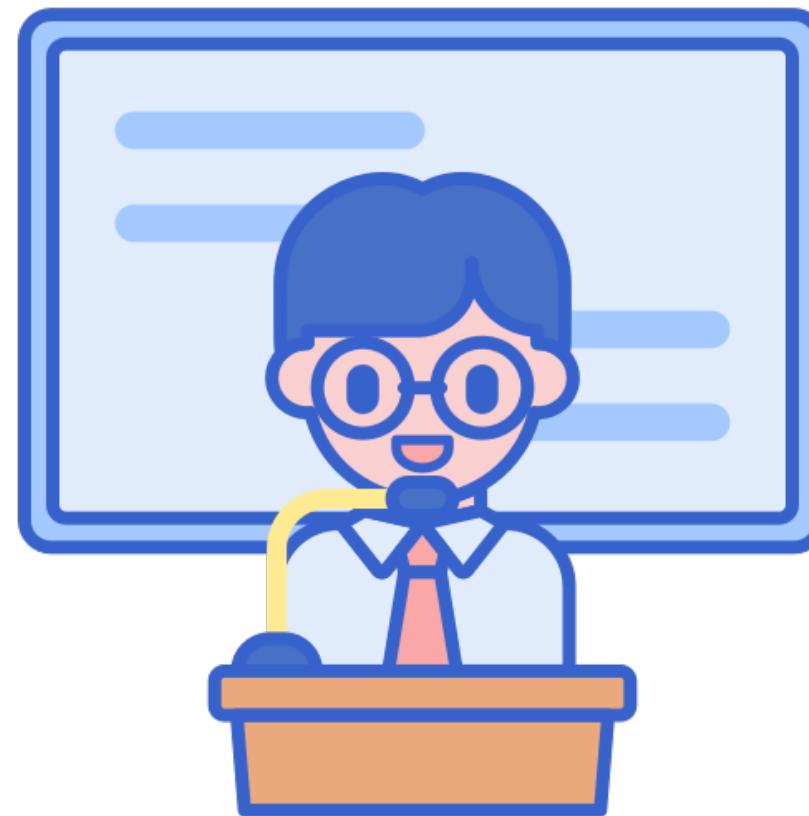
## 5-fold Cross-validation



Model is trained on 5 different versions of training data, and evaluated on 5 different versions of the test data.

# The most important concept in machine learning

(the 3 sets)



**Course materials  
(training set)**

**Practice exam  
(validation set)**

**Final exam  
(test set)**

**Generalization**

The ability for a machine learning model to perform well on data it hasn't seen before.

# Classification and Regression metrics

## Classification

**Accuracy**

Precision

Recall

F1

## Regression

**R<sup>2</sup> (r-squared)**

Mean absolute error (MAE)

Mean squared error (MSE)

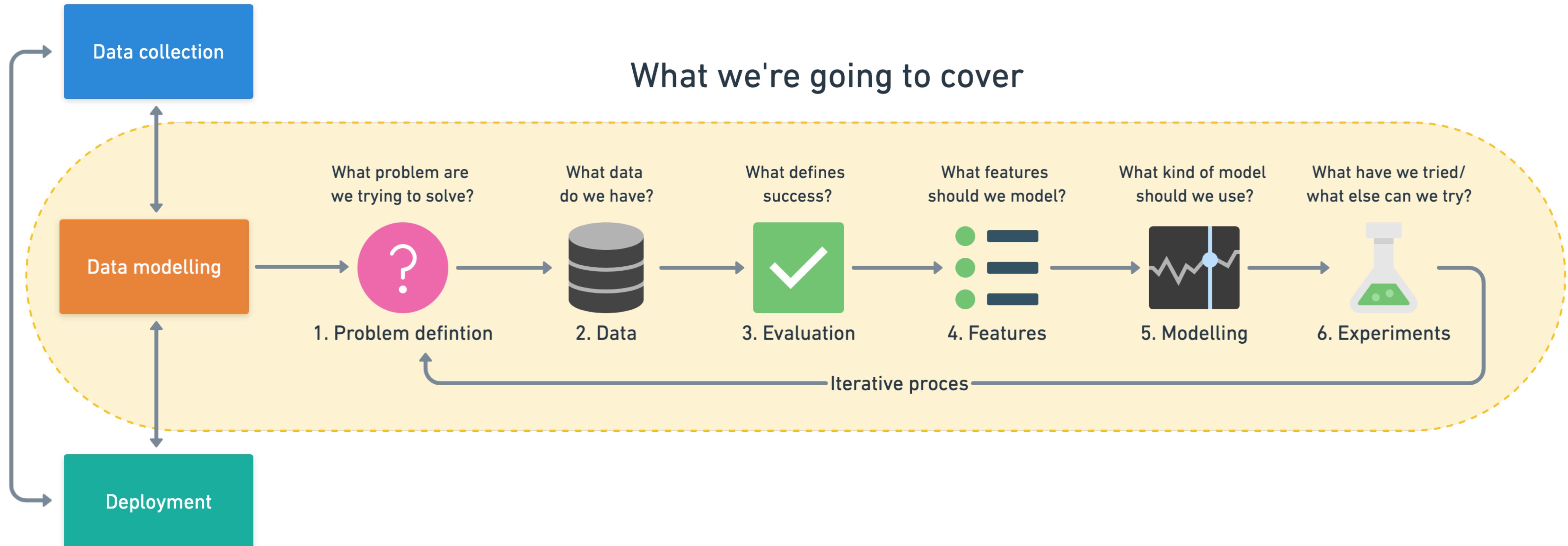
Root mean squared error (RMSE)

**Bold** = default evaluation in Scikit-Learn

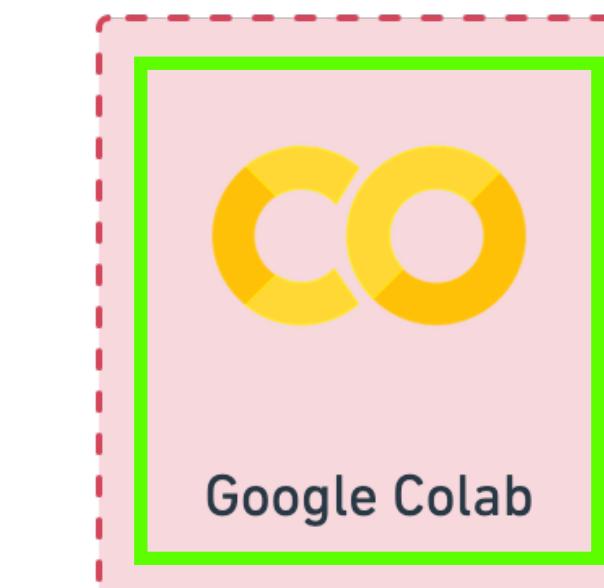
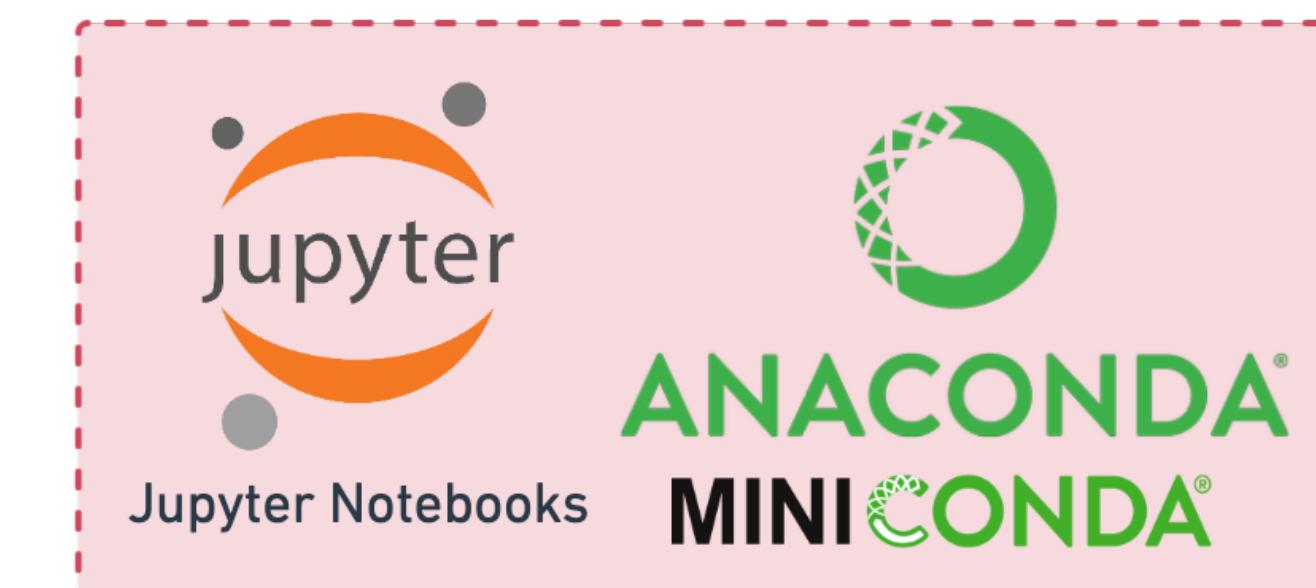
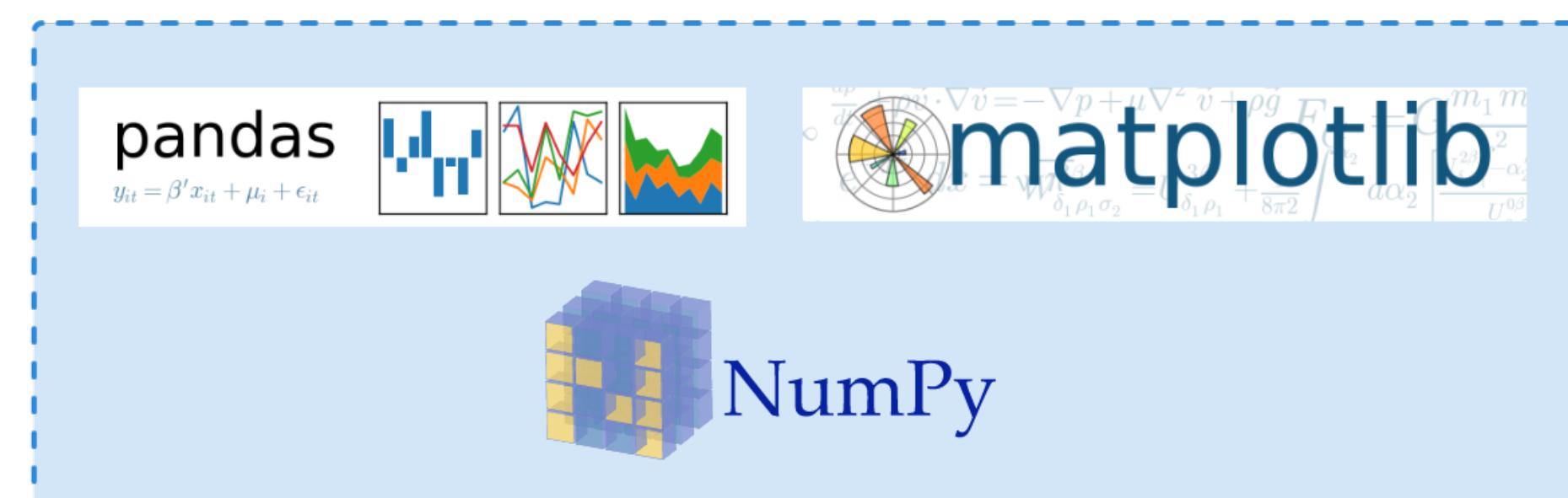
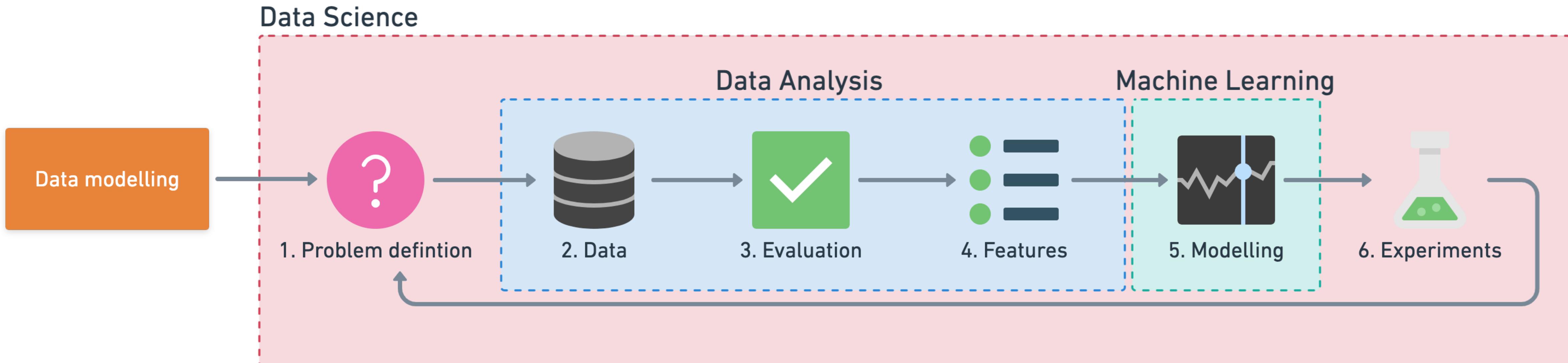
# Which regression metric should you use?

- **R<sup>2</sup>** is similar to accuracy. It gives you a quick indication of how well your model might be doing. Generally, the closer your **R<sup>2</sup>** value is to 1.0, the better the model. But it doesn't really tell exactly how wrong your model is in terms of how far off each prediction is.
- **MAE** gives a better indication of how far off each of your model's predictions are on average.
- As for **MAE** or **MSE**, because of the way MSE is calculated, squaring the differences between predicted values and actual values, it amplifies larger differences. Let's say we're predicting the value of houses (which we are).
  - Pay more attention to MAE: When being \$10,000 off is **twice** as bad as being \$5,000 off.
  - Pay more attention to MSE: When being \$10,000 off is **more than twice** as bad as being \$5,000 off.

## Steps in a full machine learning project

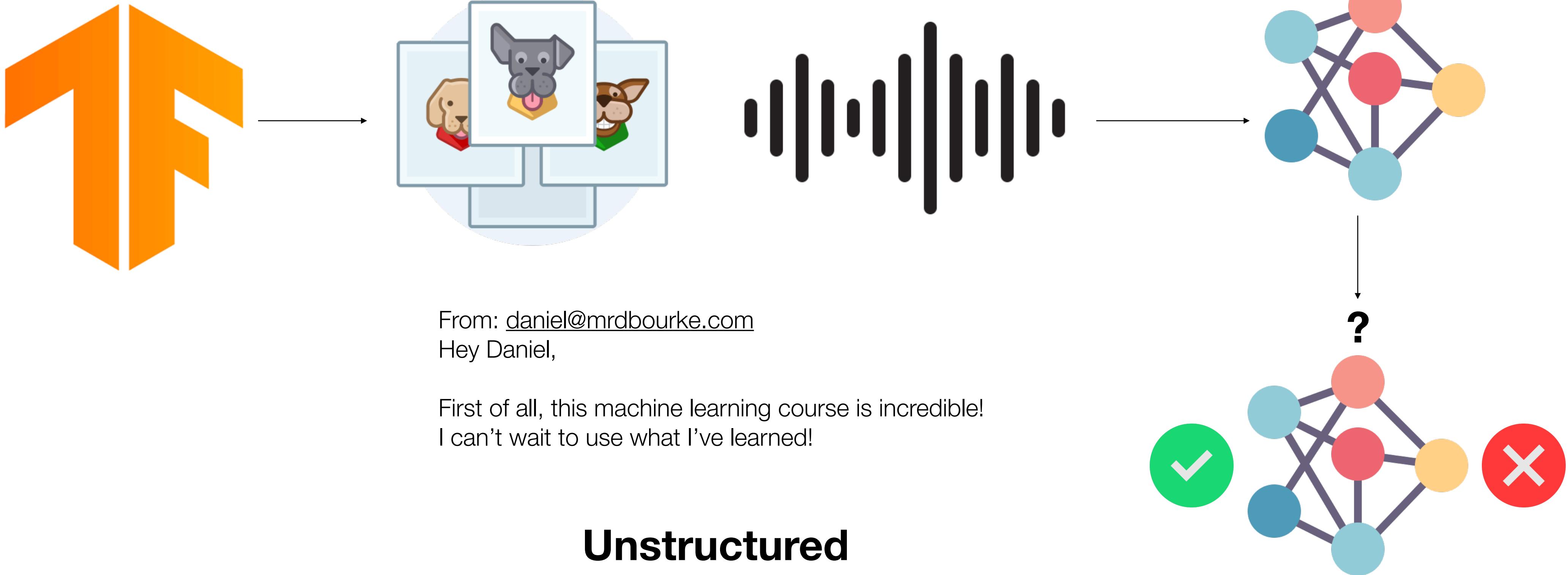


# Tools you can use



A new contender!

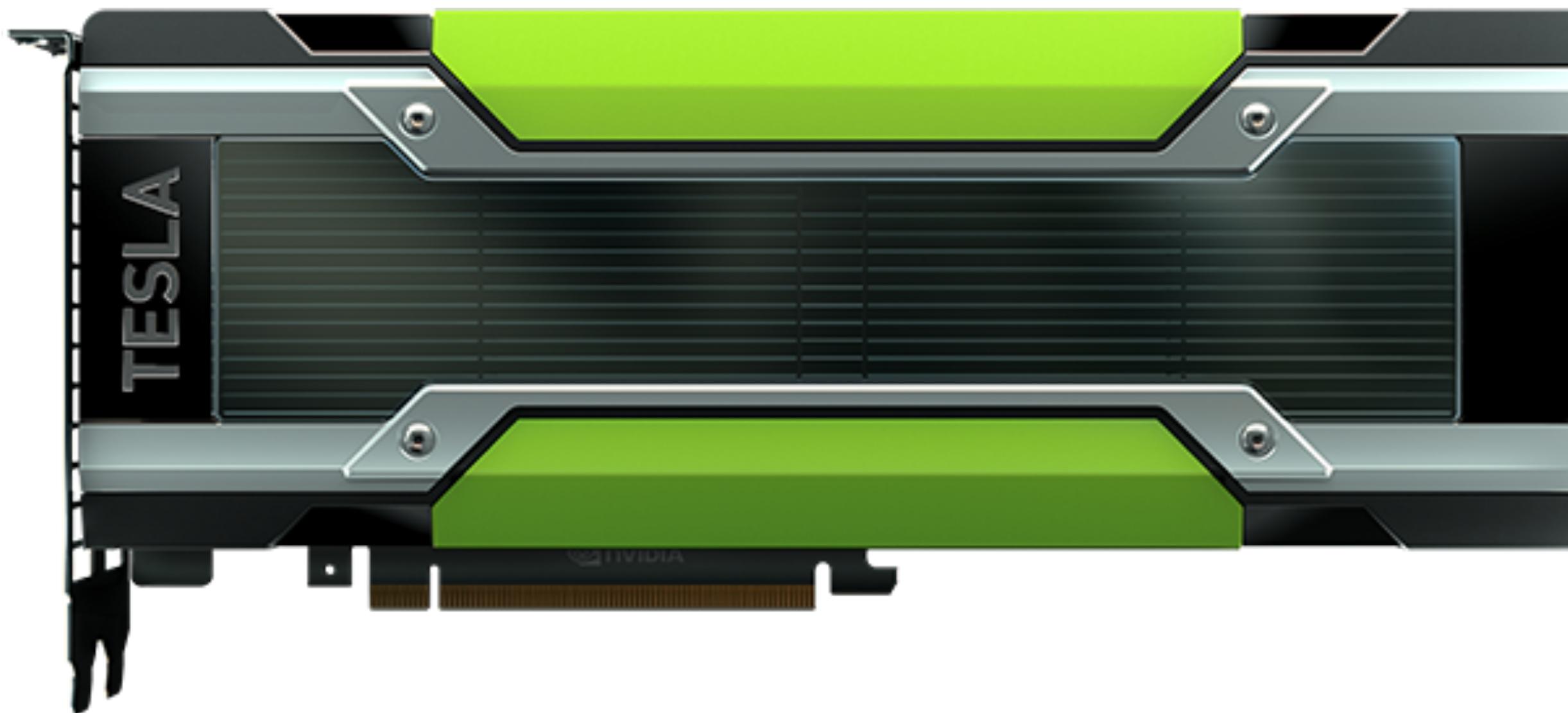
# What is TensorFlow?



# Why TensorFlow?

- Write fast deep learning code in Python (able to run on a GPU)
- Able to access many pre-built deep learning models
- Whole stack: preprocess, model, deploy
- Originally designed and used in-house by Google (now open-source)

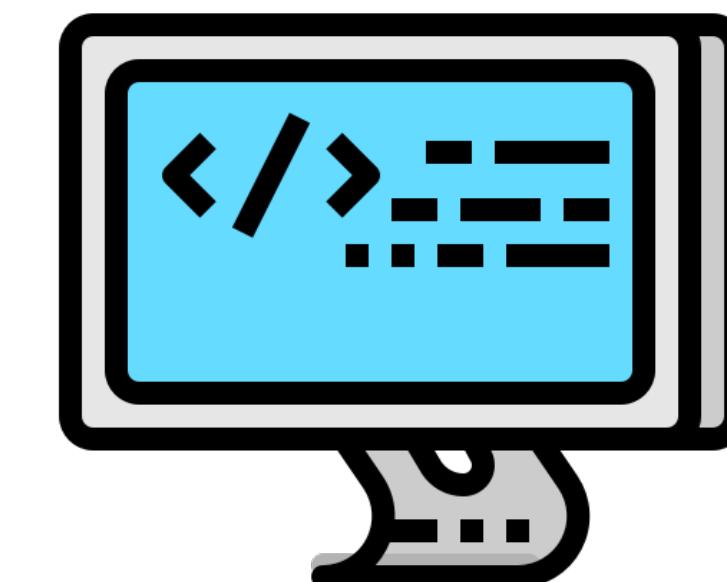
# What is a GPU?



# Choosing a model (throwback)



Problem 1 (structured)



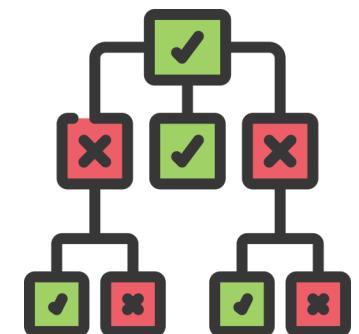
Model 1

Structured Data



CatBoost

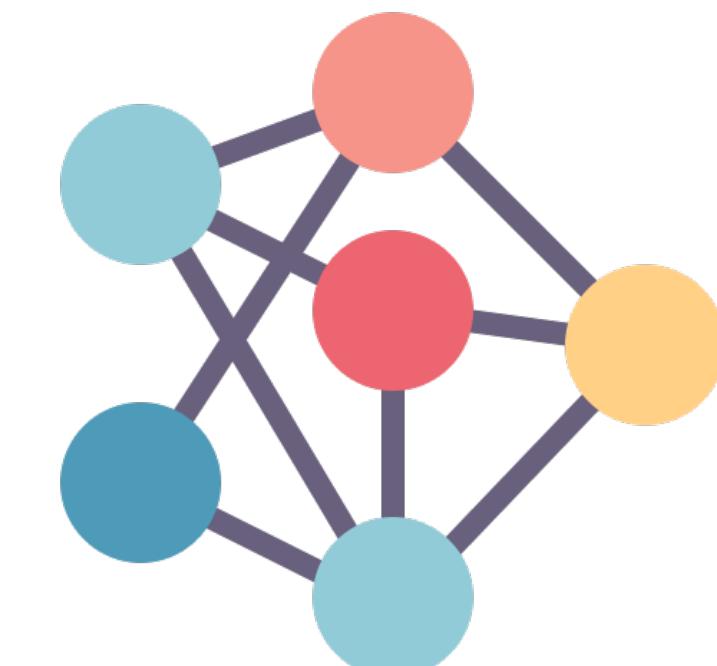
*dmlc*  
**XGBoost**



Random Forest



Problem 2 (unstructured)



Model 2

Unstructured Data

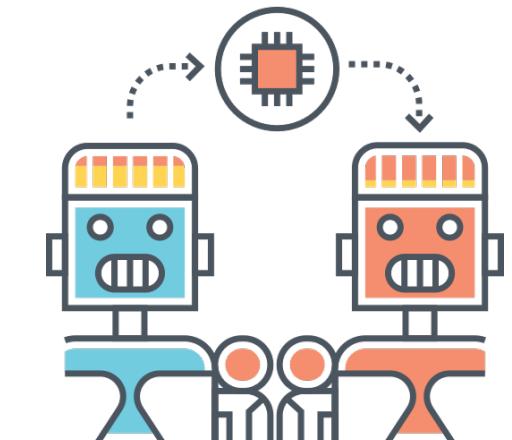


Deep Learning

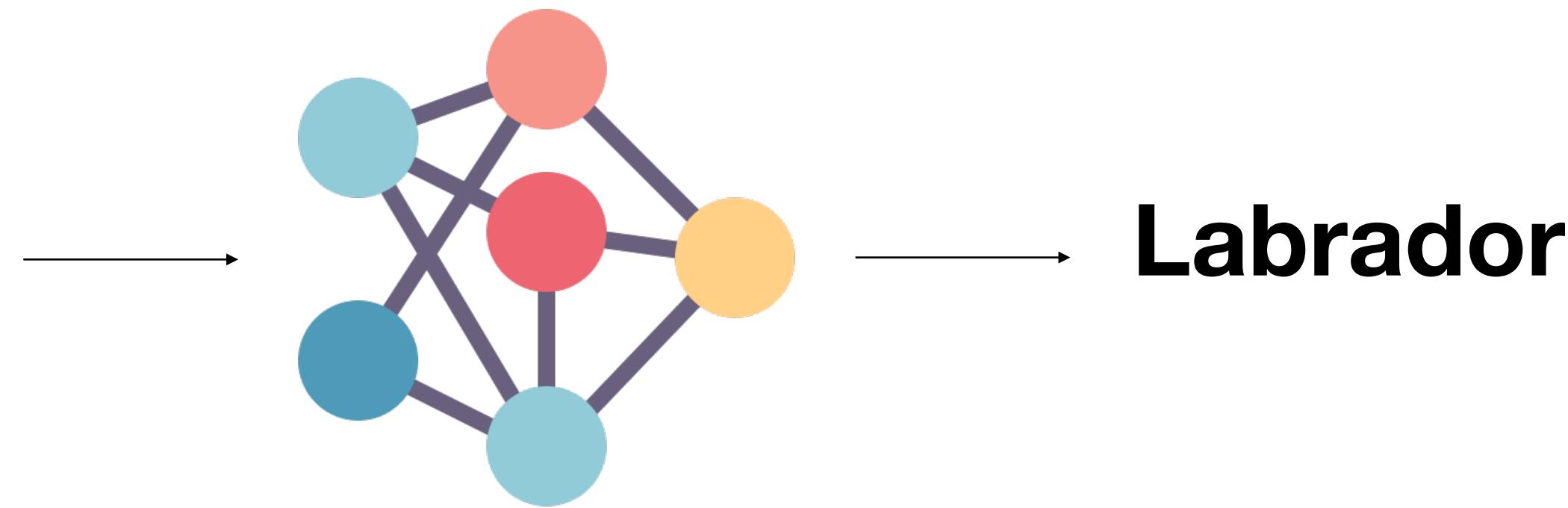
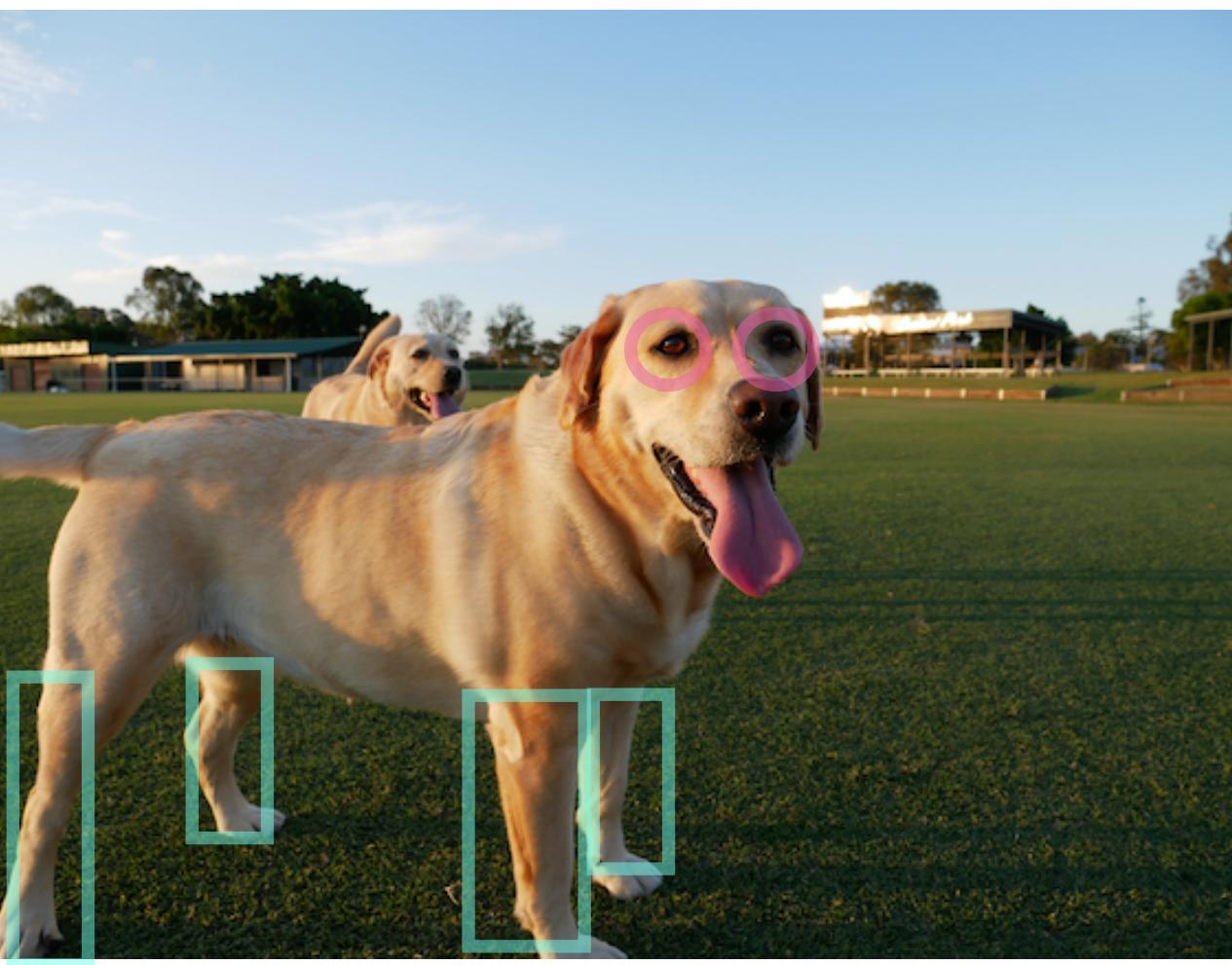


Transfer Learning

**TensorFlow**  
Hub

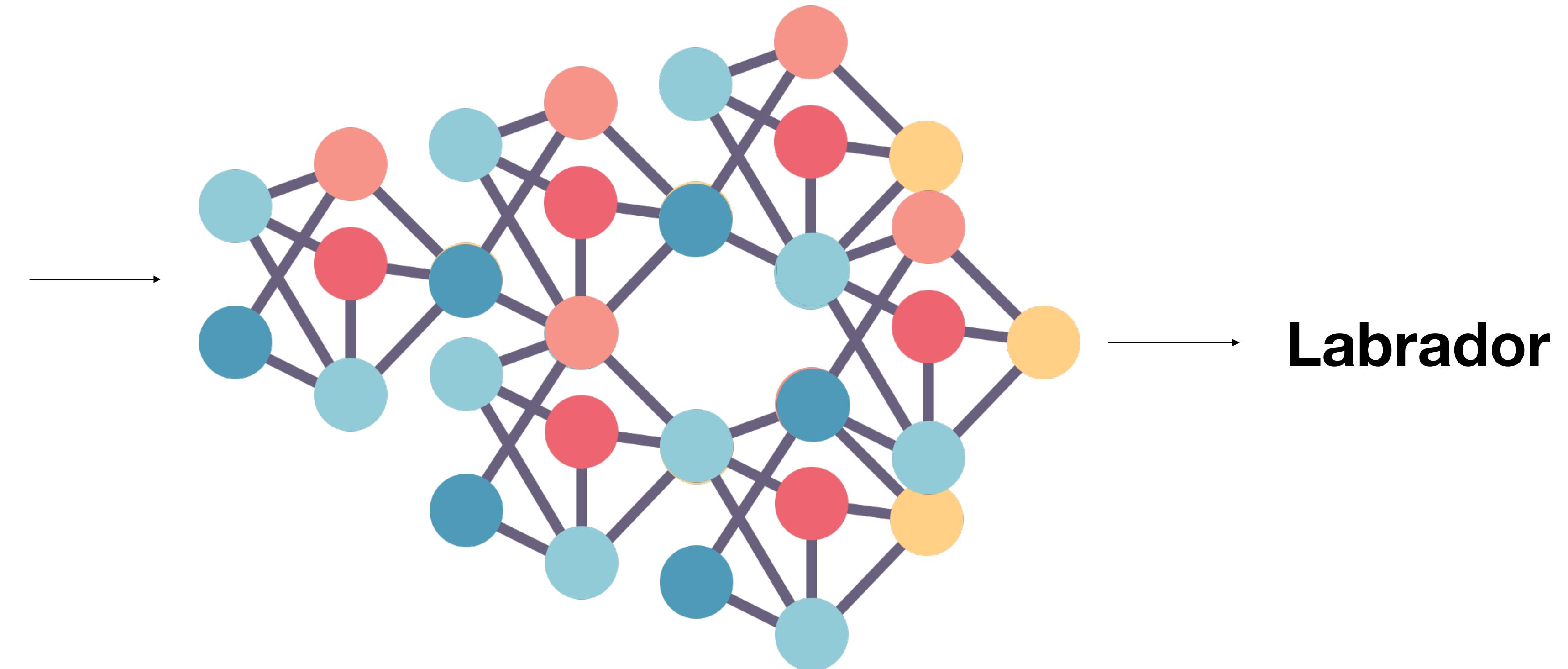
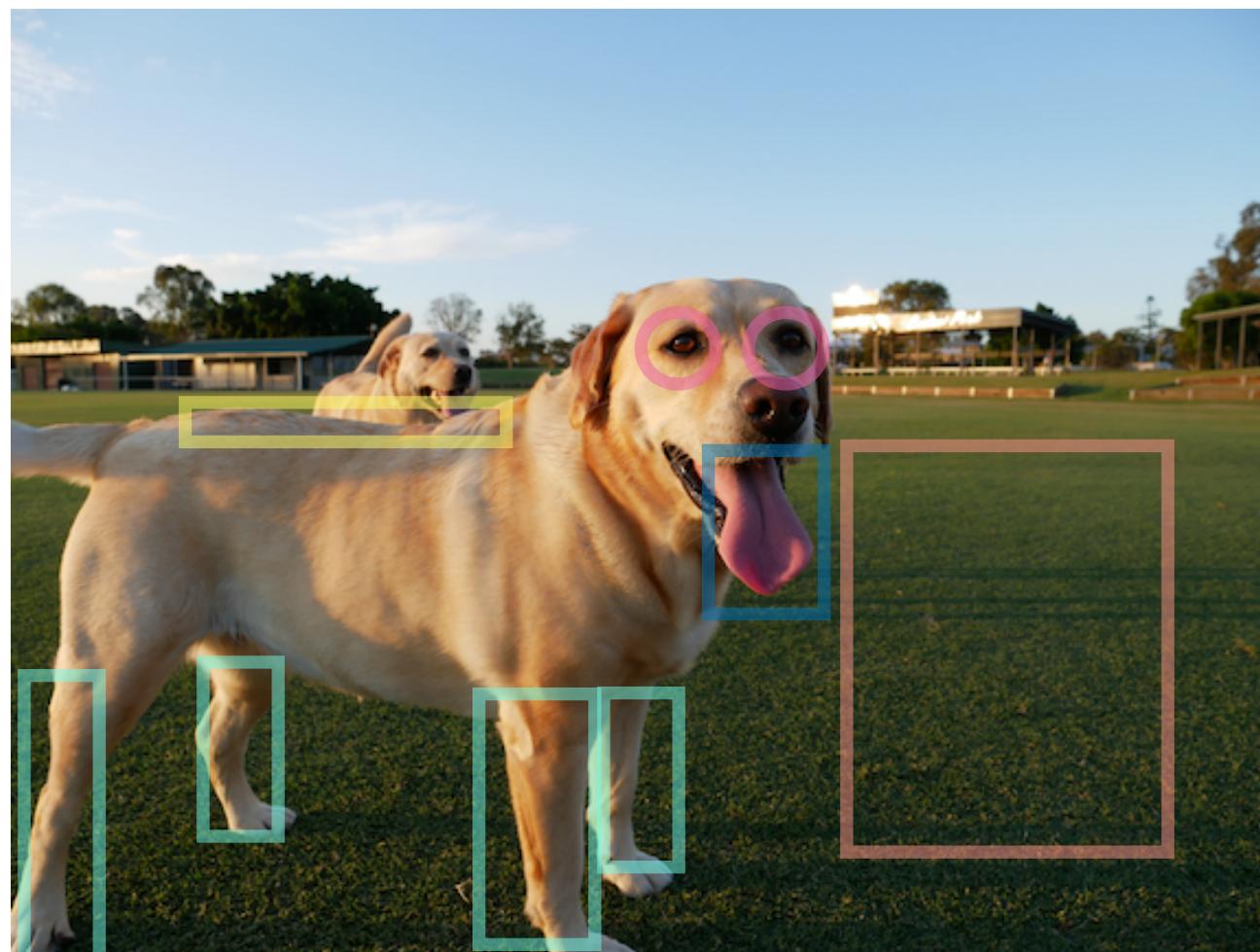


# What is deep learning? What are neural networks?

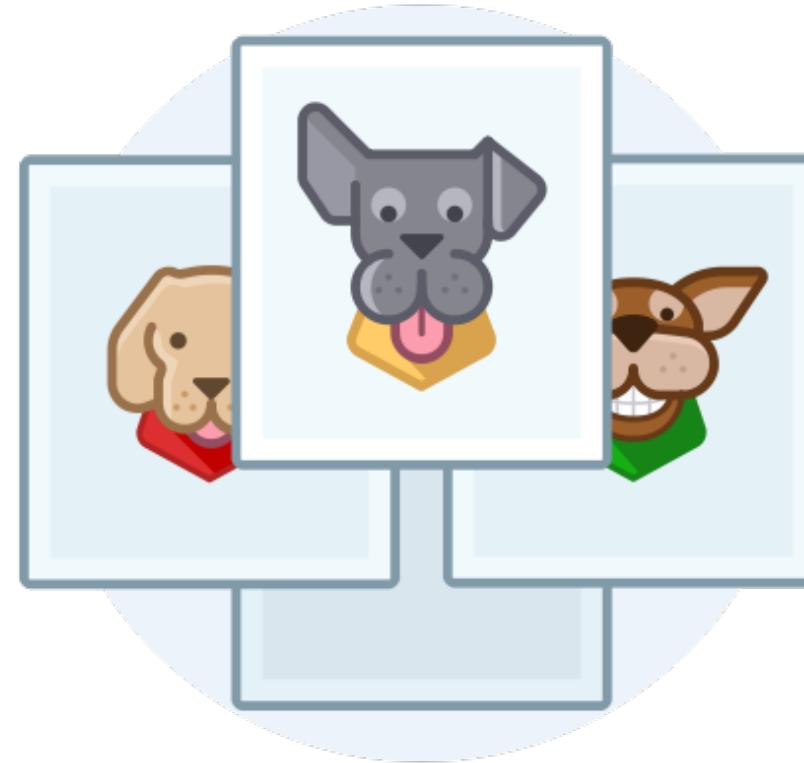


**Labrador**

# What is deep learning? What are neural networks?



# What kind of deep learning problems are there?



From: [daniel@mrbourke.com](mailto:daniel@mrbourke.com)

Hey Daniel,

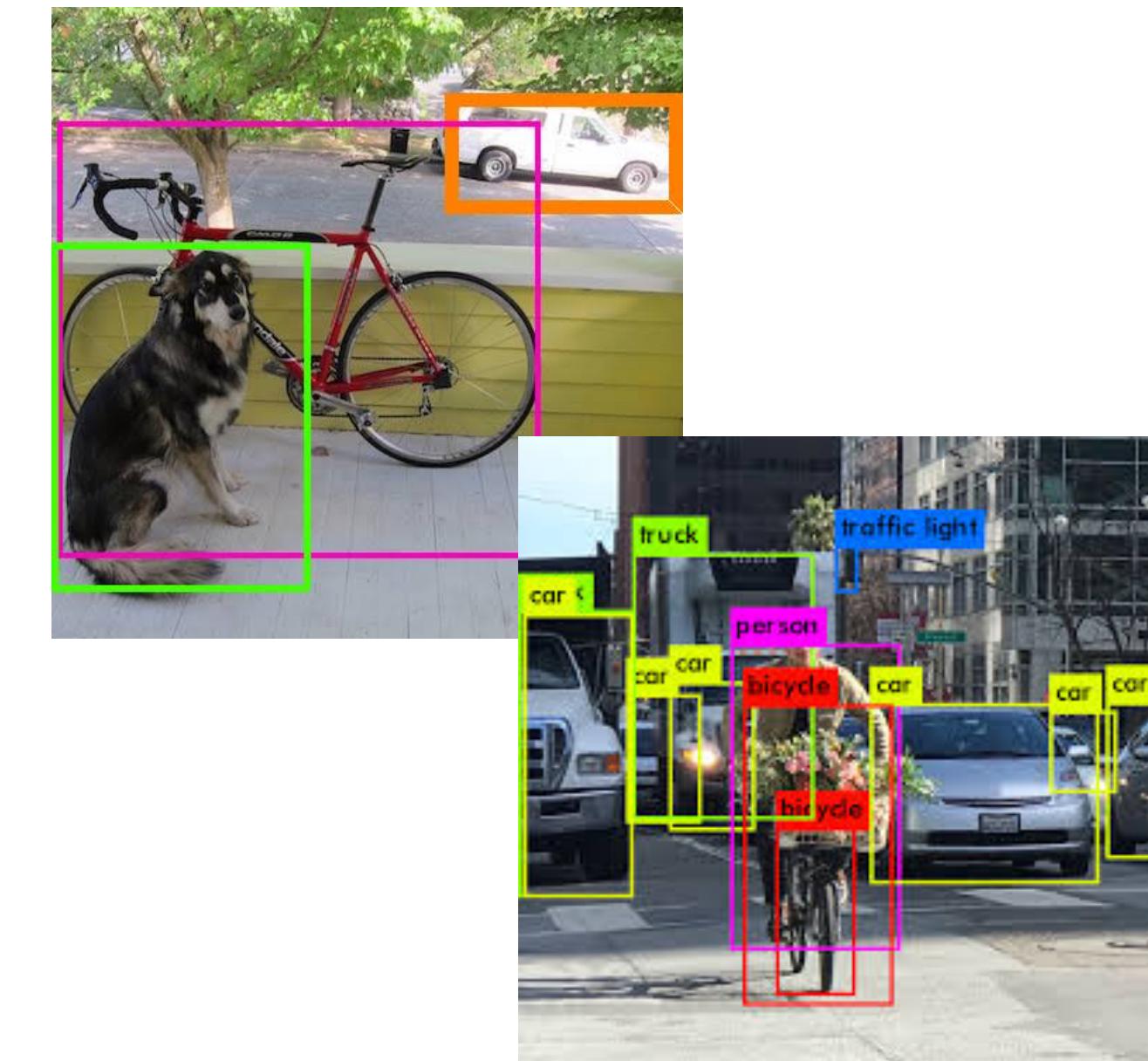
First of all, this machine learning course is incredible!  
I can't wait to use what I've learned!

**Classification**



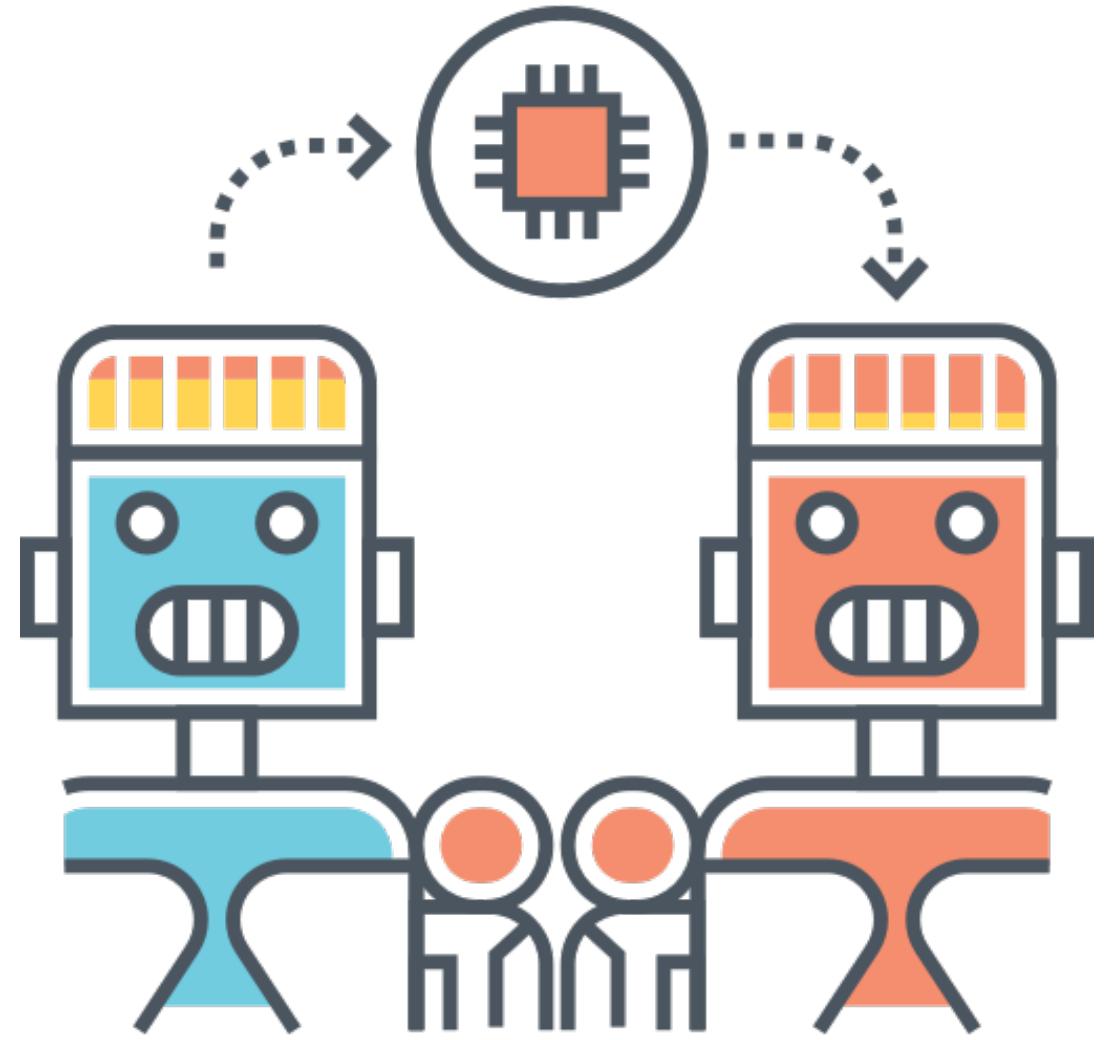
Hey Siri, where's the nearest cafe?

**Sequence to sequence  
(seq2seq)**



**Object detection**

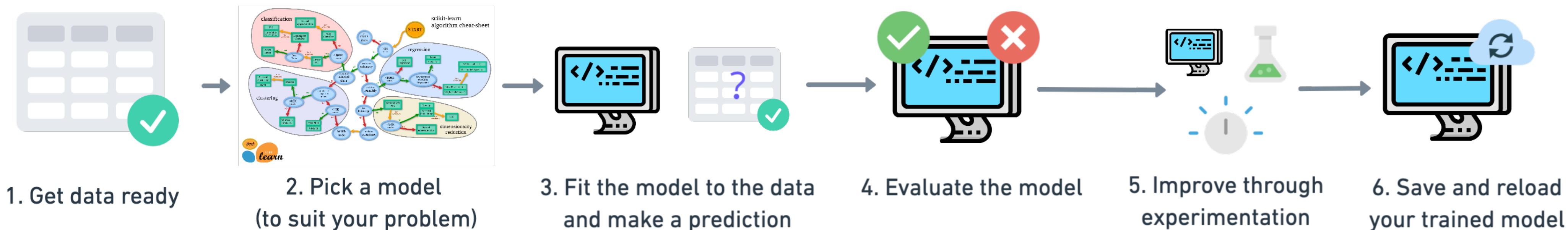
# What is transfer learning? Why use transfer learning?



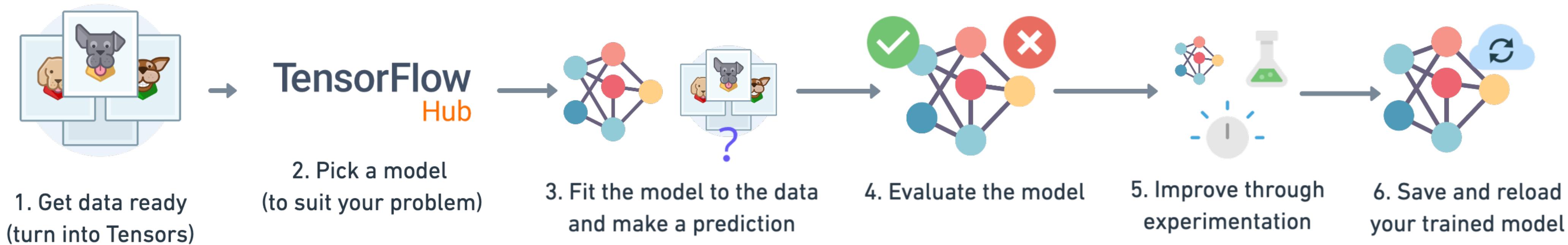
- Take what you know in one domain and apply it to another.
- Starting from scratch can be expensive and time consuming.
- Why not take advantage of what's already out there?

# What are we going to cover?

## A Scikit-Learn workflow



## A TensorFlow workflow



# What are we going to cover?

- An end-to-end multi-class classification workflow with TensorFlow
- Preprocessing image data (getting it into Tensors)
- Choosing a deep learning model
- Fitting a model to the data (learning patterns)
- Making predictions with a model (using patterns)
- Evaluating model predictions
- Saving and loading models
- Using a trained model to make predictions on custom data

# The problem we're going to work on



## Classification

- “Is this example one thing or another?”
- **Binary classification = two options**
- **Multi-class classification = more than two options**

A screenshot of a web browser displaying the Kaggle Dog Breed Identification competition page. The page has a dark header with the Kaggle logo. Below the header, there's a banner featuring a close-up photo of a dog's face. The main title is "Dog Breed Identification" with the subtitle "Determine the breed of a dog in an image". It shows "1,282 teams" participated 2 years ago. A navigation bar below the banner includes "Overview" (which is underlined), Data, Notebooks, Discussion, Leaderboard, Rules, Team, My Submissions, and Late Submission. The "Late Submission" button is highlighted in blue. On the right, there's a panel with sections for "Description" and "Evaluation". The "Description" section contains text about identifying dog breeds using deep neural networks. The "Evaluation" section contains text about the ImageNet dataset used in the competition.

# Where can you get help?

- Follow along with the code



```
Using Transfer Learning and TensorFlow 2.0 to Classify Different Dog Breeds

Who's that doggy in the window?
Dogs are incredible. But have you ever been walking down the street, seen a dog and not known what breed it is? I have. And then someone says, "it's an English Terrier" and you think, how did they know that?

In this project we're going to be using machine learning to help us identify different breeds of dogs.

To do this, we'll be using data from the Kaggle dog breed identification competition. It consists of a collection of 10,000+ labelled images of 120 different dog breeds.

This kind of problem is called multi-class image classification. It's multi-class because we're trying to classify multiple different breeds of dog. If we were only trying to classify dogs versus cats, it would be called binary classification.

Multi-class image classification is an important problem because it's the same kind of technology Tesla uses in their self-driving cars or Airbus uses in automatically adding information to their listings.

Since the most important step in a deep learning problem is getting the data ready (turning it into numbers), that's what we're going to start with.

We're going to go through the following TensorFlow/Deep Learning workflow:
```

- Try it for yourself

- Press SHIFT + CMD + SPACE to read the docstring

- Search for it
- Try again

- Ask (don't forget the Discord chat!)



```
# Create a function which builds a Keras model
def create_model(input_shape=INPUT_SHAPE, output_shape=OUTPUT_SHAPE, model_url=MODEL_URL):
    print("Building model with:", MODEL_URL)

    # Setup the model layers
    model = tf.keras.Sequential([
        hub.KerasLayer(MODEL_URL),
        tf.keras.layers.Dense(1000, activation='relu'),
        tf.keras.layers.Dense(1000, activation='relu'),
        tf.keras.layers.Dense(output_shape)
    ])

    # Compile the model
    model.compile(
        loss=tf.keras.losses.CategoricalCrossentropy(),
        optimizer=tf.keras.optimizers.Adam(),
        metrics=['accuracy']
    )

    # Build the model
    model.build(INPUT_SHAPE)

    return model
```

What's happening here?  
Setting up the model layers

TensorFlow Core

TensorFlow guide

TensorFlow 2

Effective TensorFlow

Migrate from TF1 to TF2

Convert with the upgrade script

Performance with `tf.function`

Community testing FAQ

Keras

Keras overview

Keras functional API

Train and evaluate

Write custom layers and models

Save and serialize models

Keras Recurrent Neural Networks

Masking and padding

Write custom callbacks

Mixed precision

TensorFlow 2 focuses on simplicity and ease of use, with updates like eager execution, intuitive higher-level APIs, and flexible model building on any platform.

Many guides are written as Jupyter notebooks and run directly in Google Colab—a hosted notebook environment that requires no setup. Click the [Run in Google Colab](#) button.

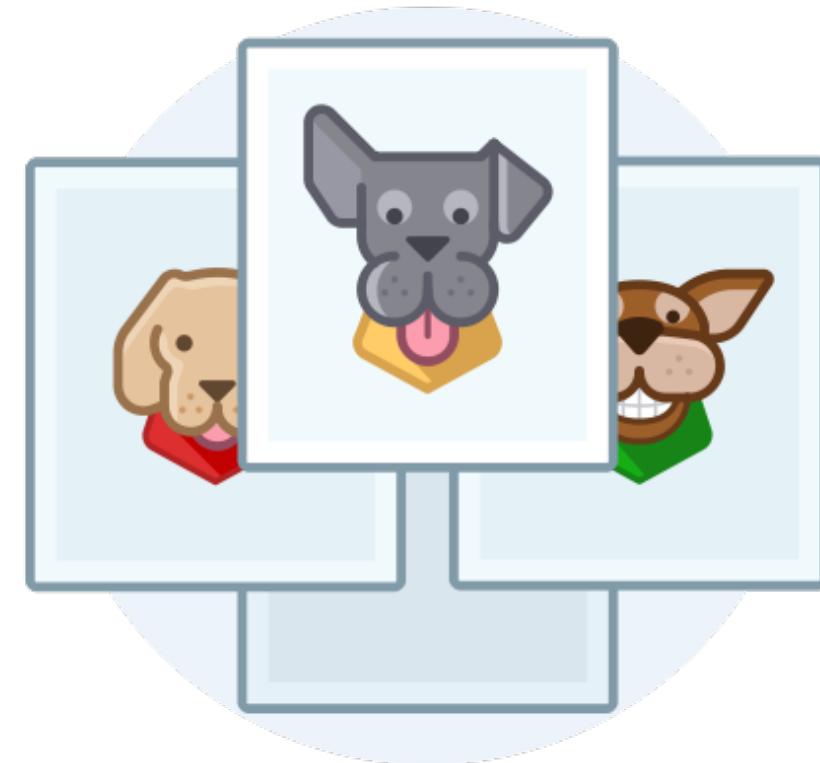
Essential documentation

Install TensorFlow

TensorFlow 2

Keras

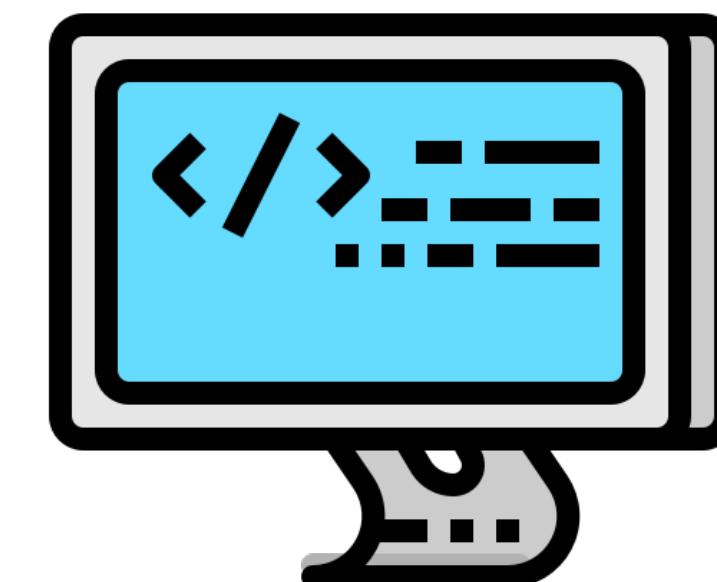
# Let's find those doggos!



# Choosing a model (throwback)

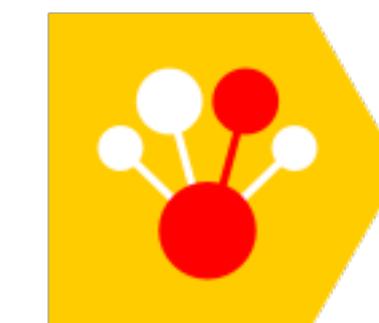


Problem 1 (structured)



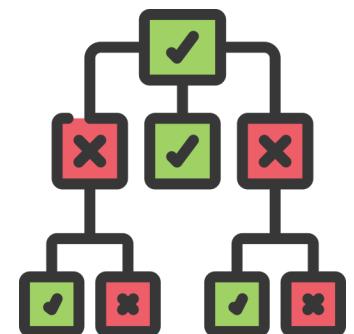
Model 1

Structured Data



CatBoost

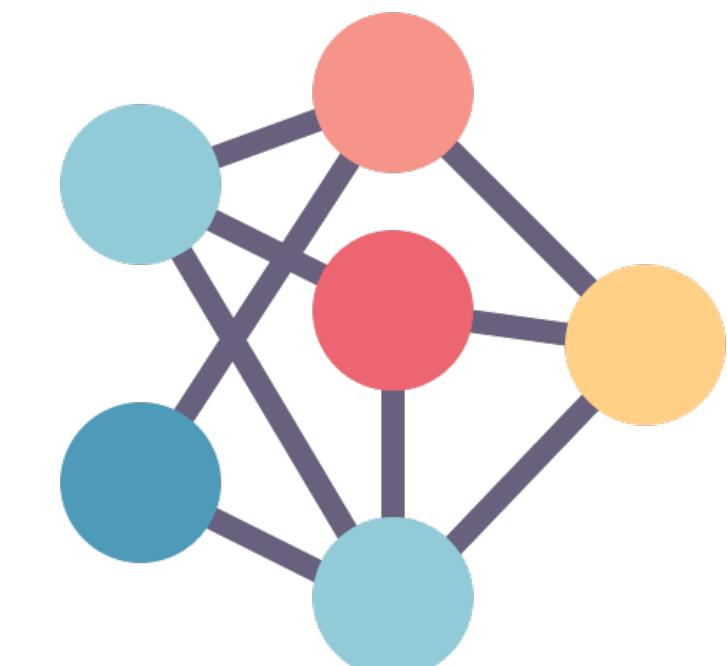
*dmlc*  
**XGBoost**



Random Forest



Problem 2 (unstructured)



Model 2

Unstructured Data

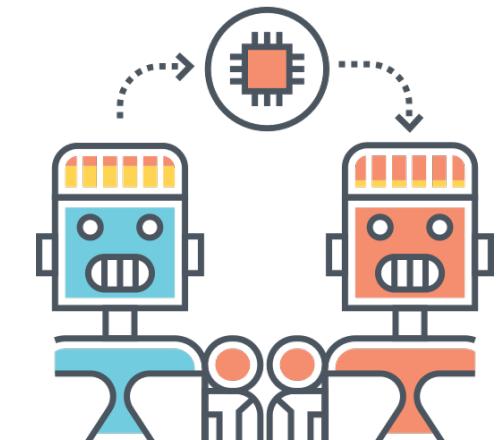


Deep Learning

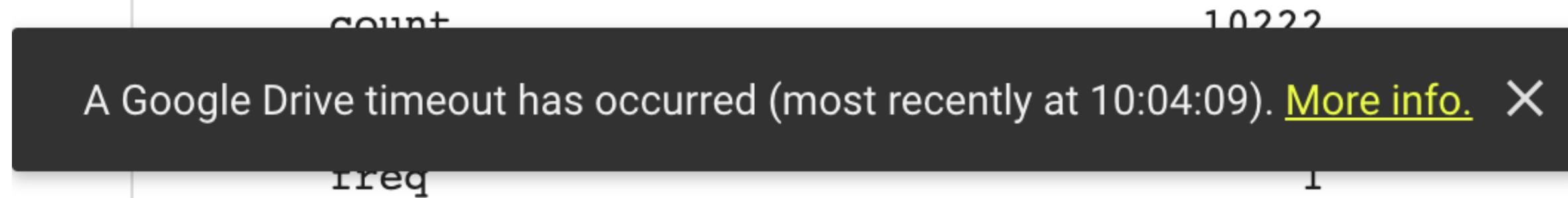


Transfer Learning

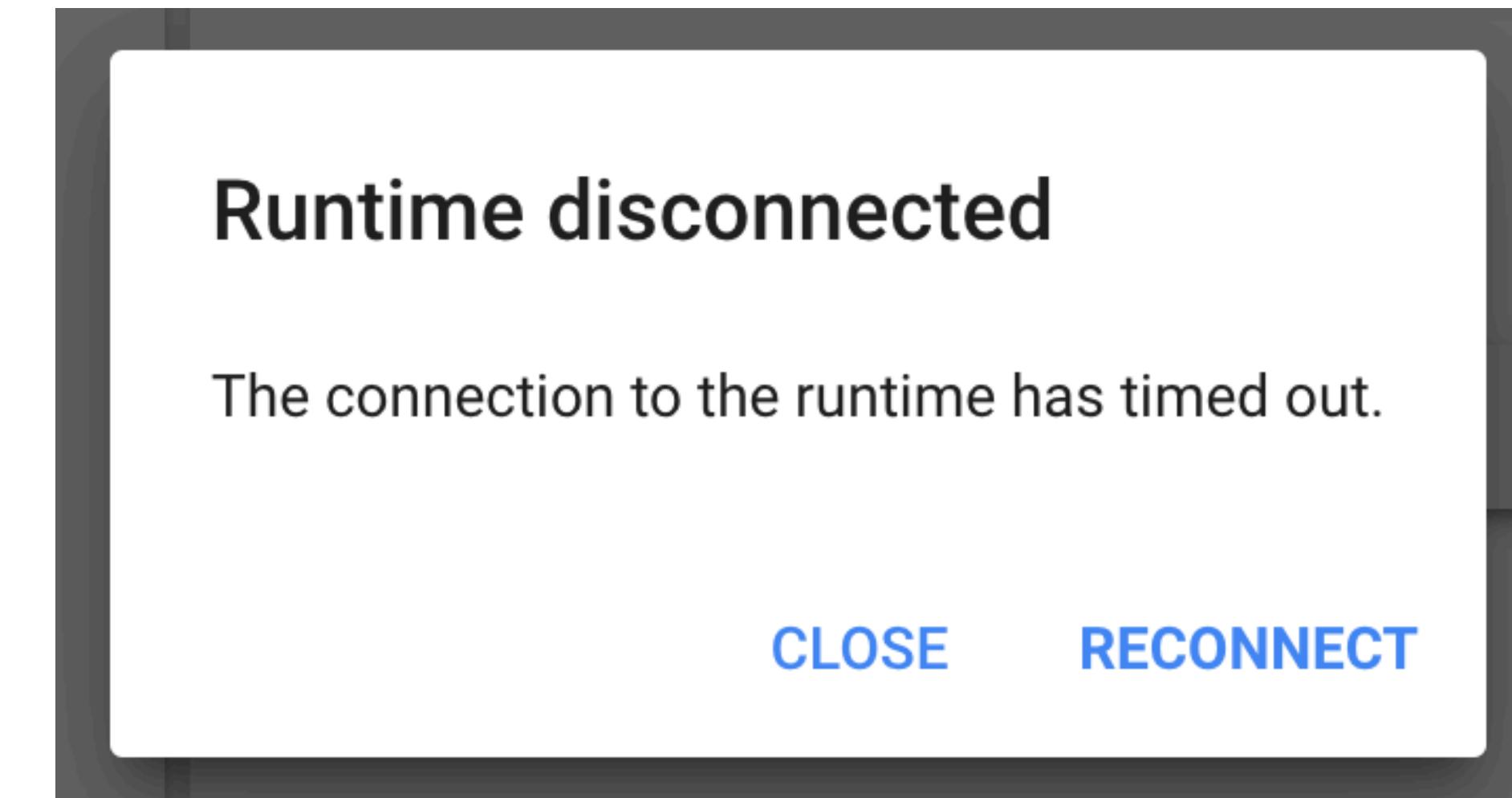
**TensorFlow**  
Hub



# Things you might see in Colab



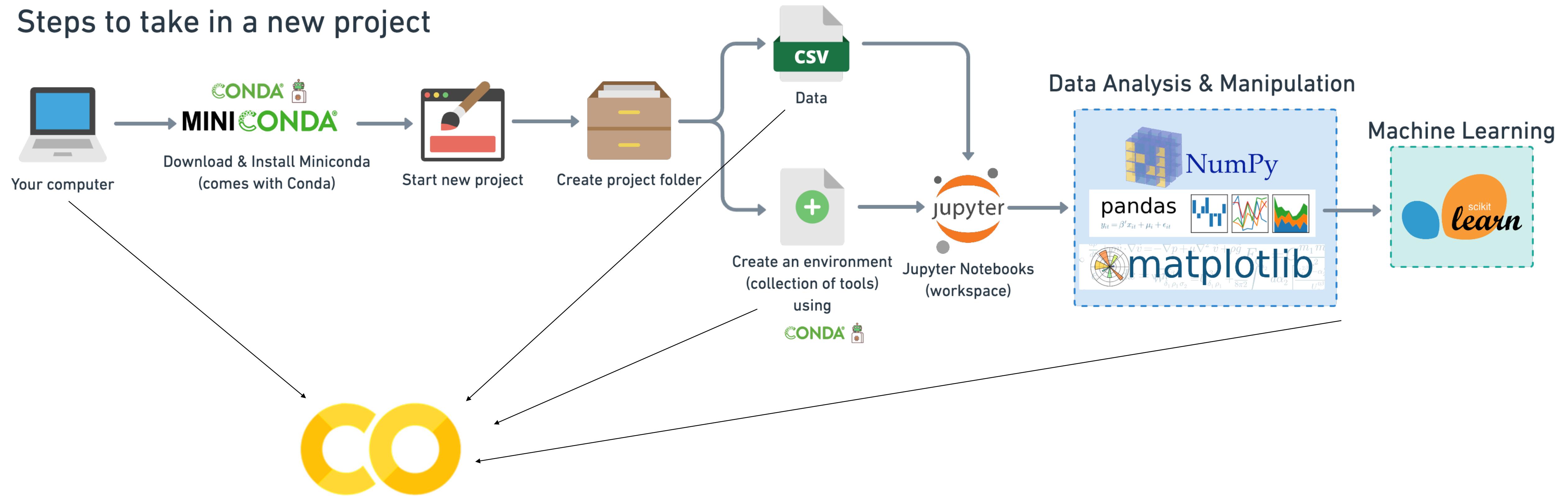
**Usually fixes itself, may need to reconnect Google Drive to Colab.**



**A Colab connection will sometimes drop out (it's hard to tell when). If it does, you'll need to reconnect and potentially rerun all of your cells (disconnecting = variables lost).**

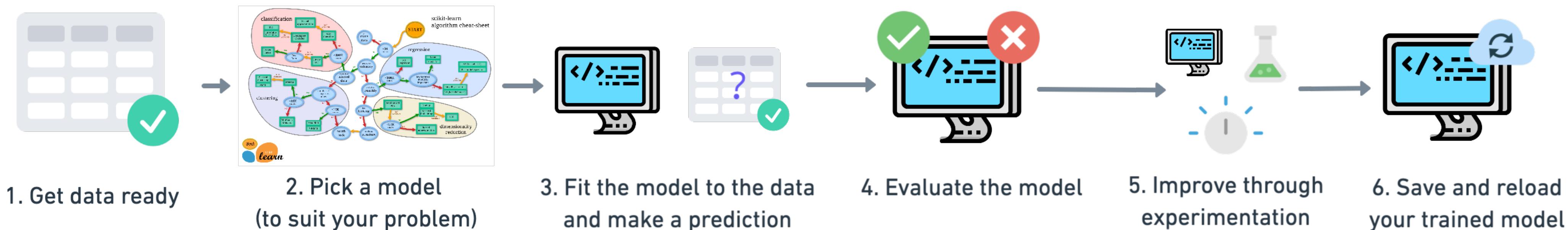
**For more information and fixes, refer to the Google Colab FAQ: <https://research.google.com/colaboratory/faq.html>**

## Steps to take in a new project

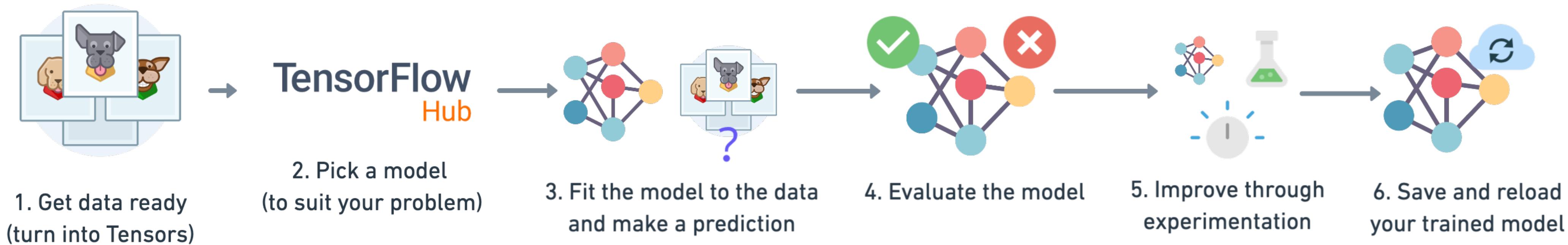


# What are we going to cover?

## A Scikit-Learn workflow

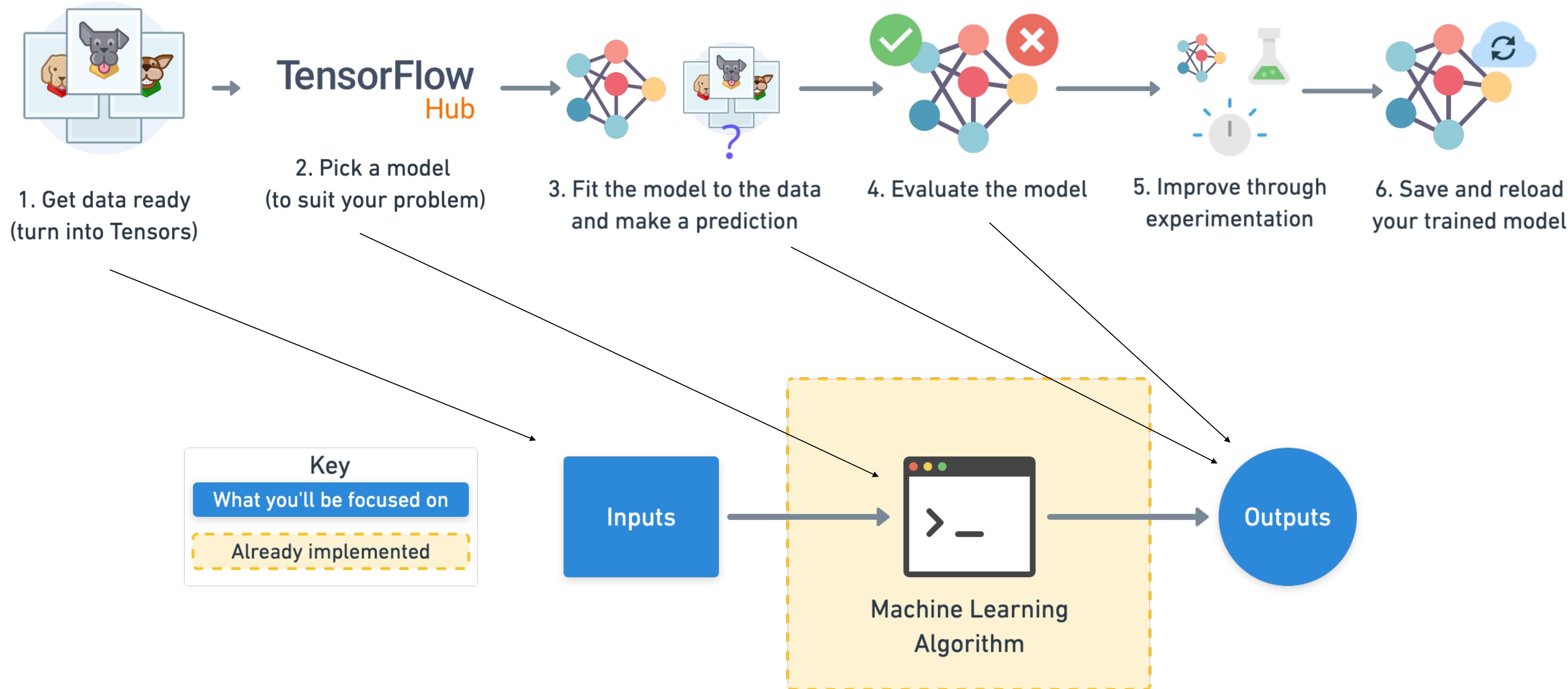


## A TensorFlow workflow



# What we're focused on

## A TensorFlow workflow



# Which activation? Which loss?

**Binary classification**

Activation: Sigmoid

**Multi-class classification**

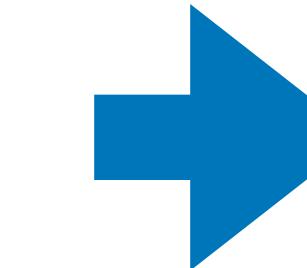
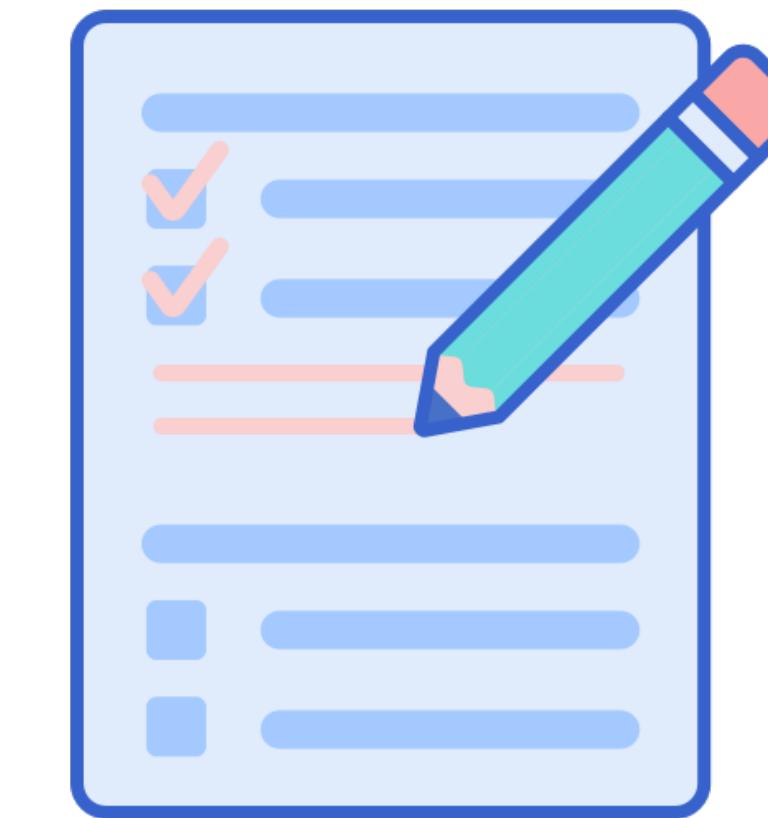
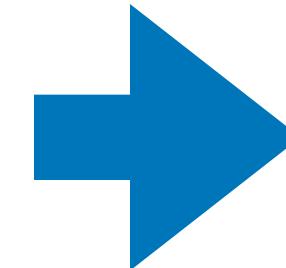
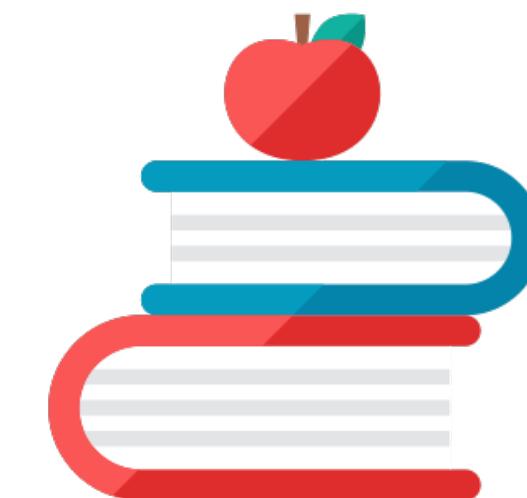
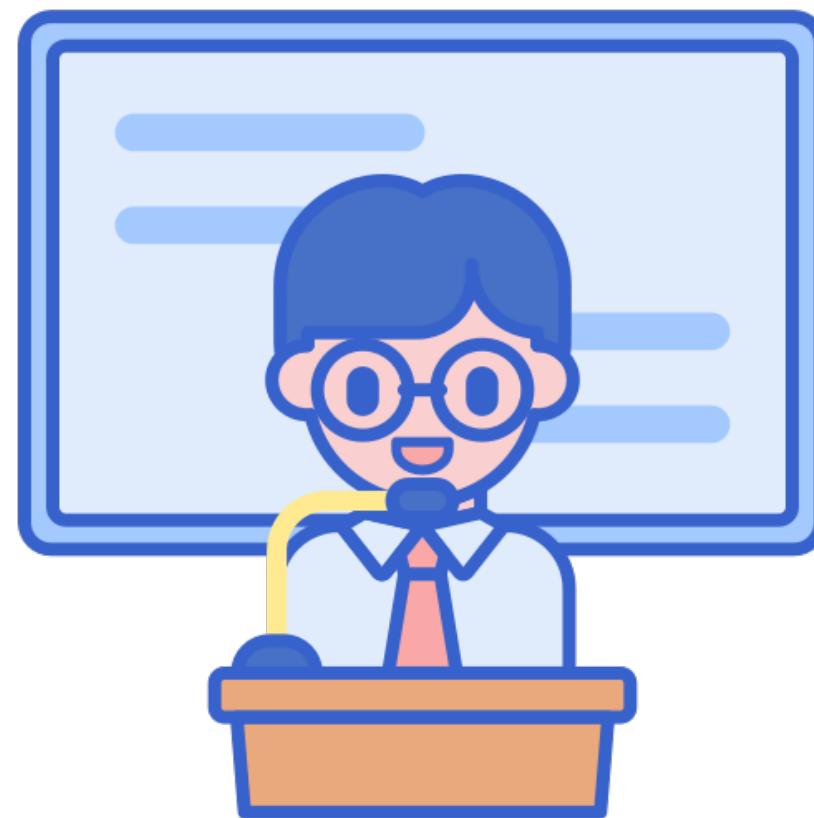
Activation: Softmax

Loss: Binary Crossentropy

Loss: Categorical Crossentropy

# The most important concept in machine learning

(the 3 sets)



**Course materials  
(training set)**

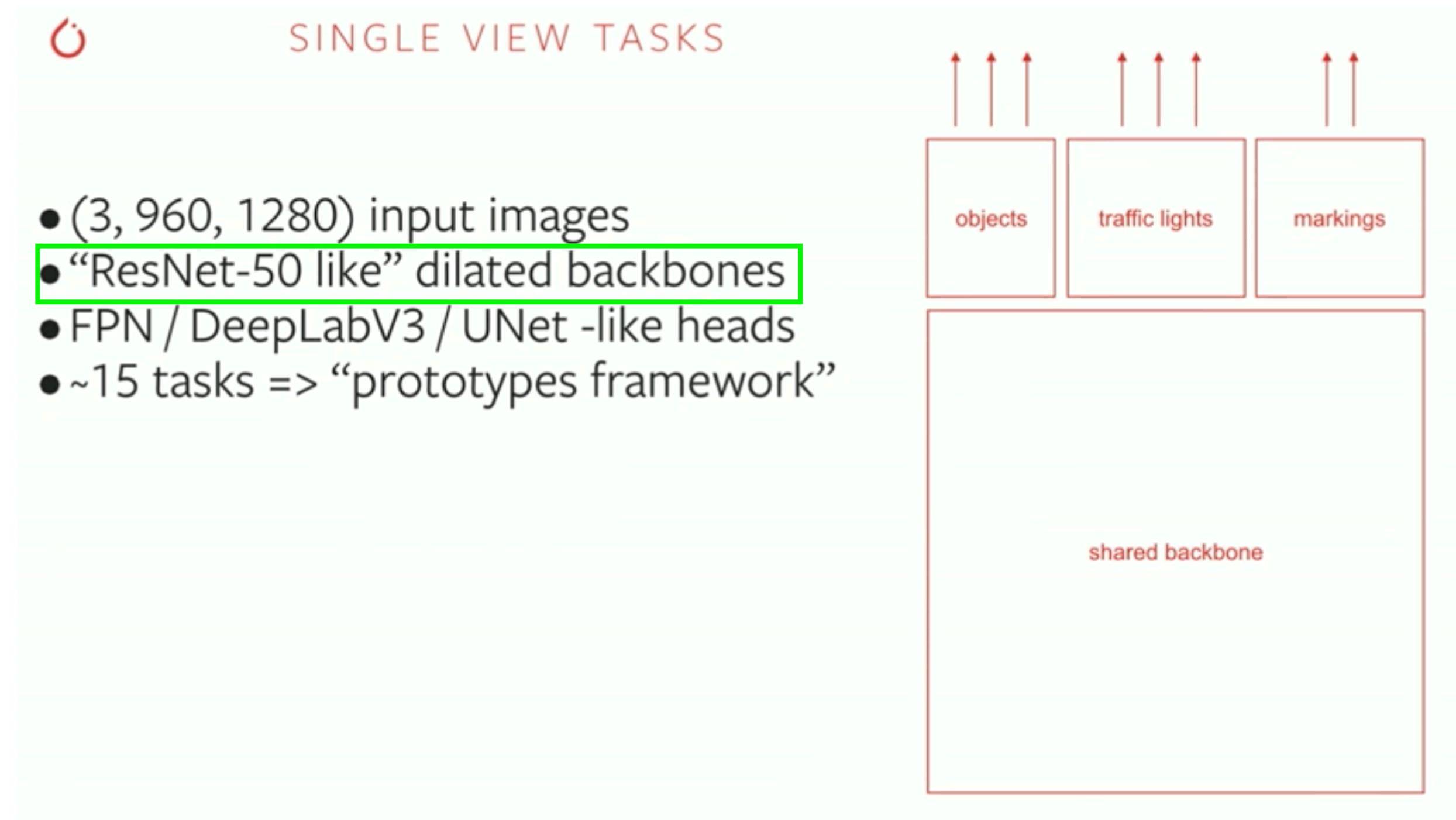
**Practice exam  
(validation set)**

**Final exam  
(test set)**

**Generalization**

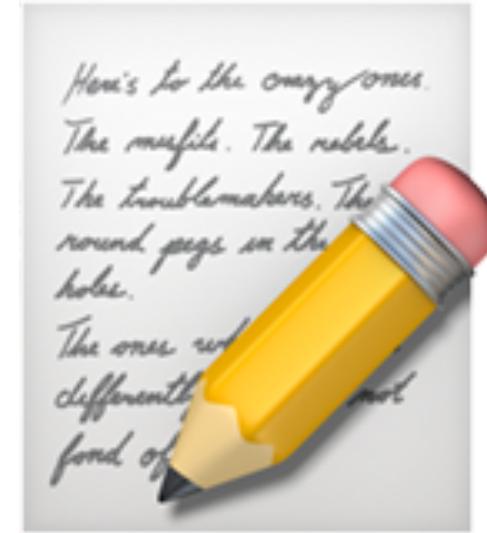
The ability for a machine learning model to perform well on data it hasn't seen before.

# Tesla using ResNet50 backbones



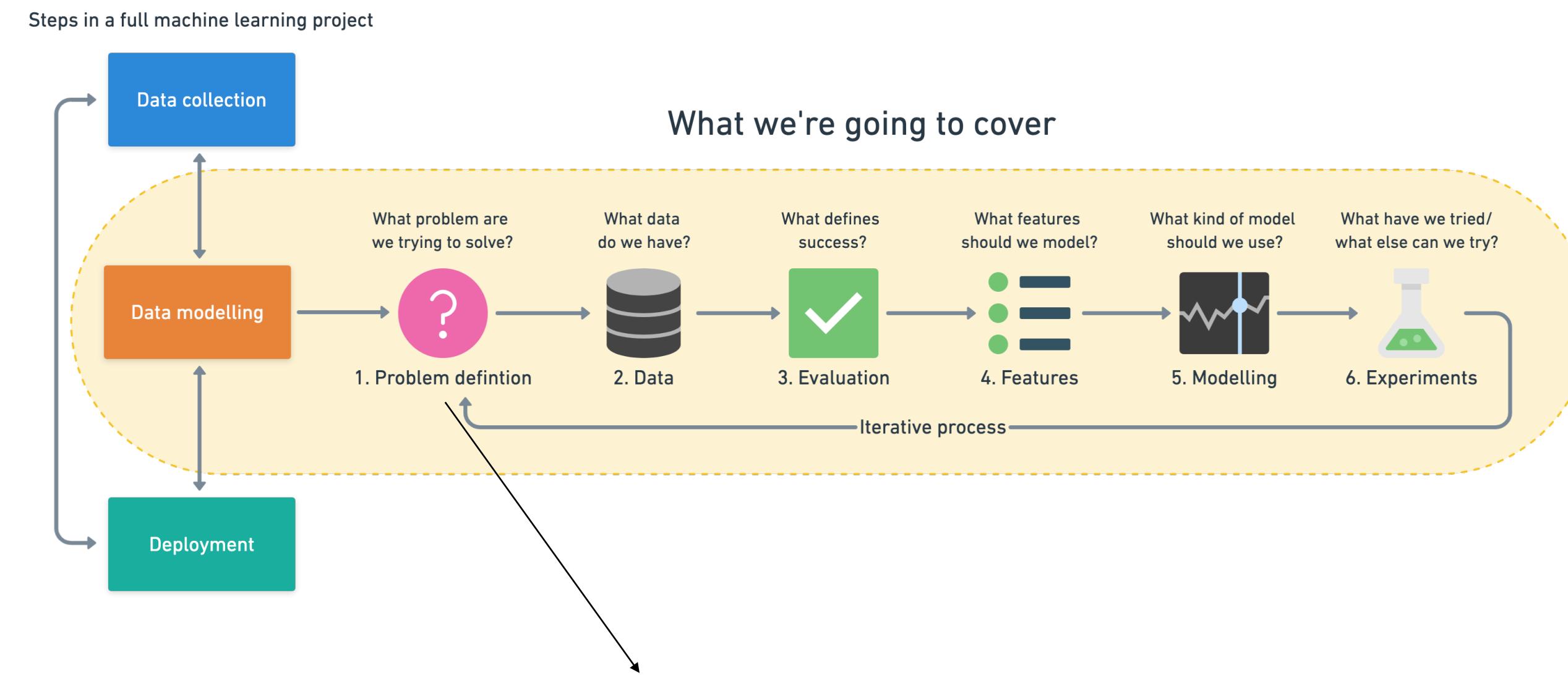
Source: “PyTorch at Tesla by Andrei Karpathy”  
<https://youtu.be/oBklltKXtDE?t=173>

# How to think about communicating and sharing your work

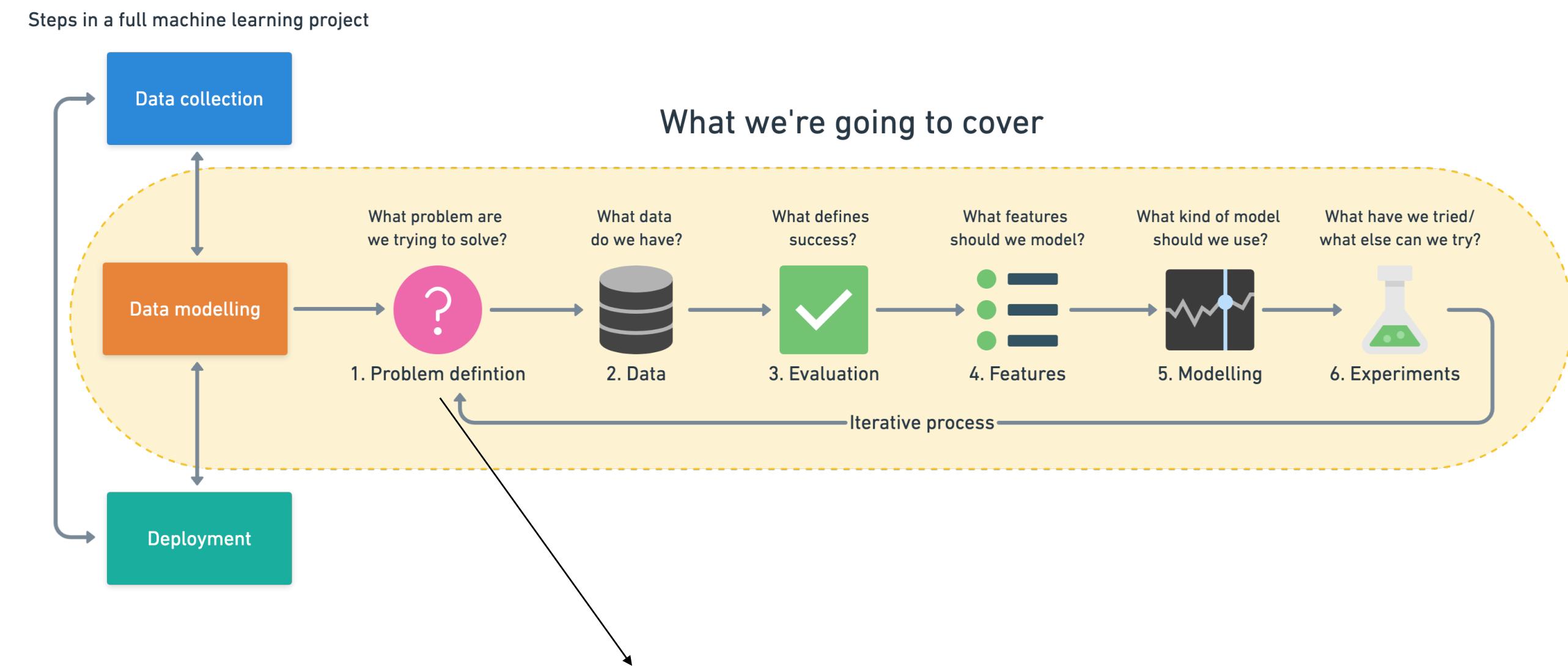


**Now you've got skills, what do you do next?**

# The most important question you can ask



# The most important question you can ask



# “Who’s it for?”

# What questions will they have?

# → What needs do they have?

# What concerns can you address before they arise?



**Heard but not understood**



**Heard and (potentially) understood**

# Who's it for?

**People on your team**

Boss

Project manager

Teammates

**People outside your team**

Clients

Customers

Fans

# Communicating with people on your team



How the project is going

What's in the way

What you've done

What you're doing next

Why you're doing something  
next

Who else could help

What's not needed

**“Who’s it for?”** → **“What do they need to know?”** → **Where you’re stuck**

What's not clear

What questions do have

Are you still working  
towards the right thing

Is there any feedback or  
advice



# The Project Manager, Boss, Senior, Lead

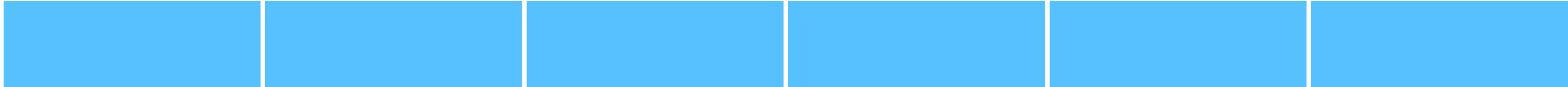


# The People You're Working With, Sitting Next to, in the Group Chat

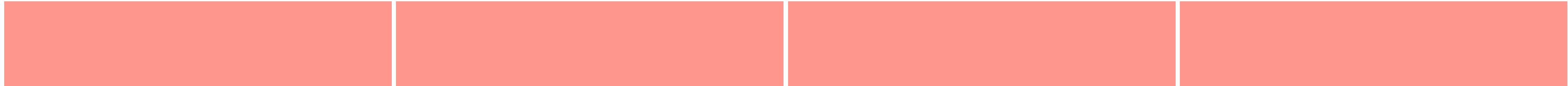


# Break it down

6-month project



4-week month



5-day week



# What did you work on today?

**What I worked on today (1-3 points on what you did):**

- What's working?
- What's not working?
- What could be improved?

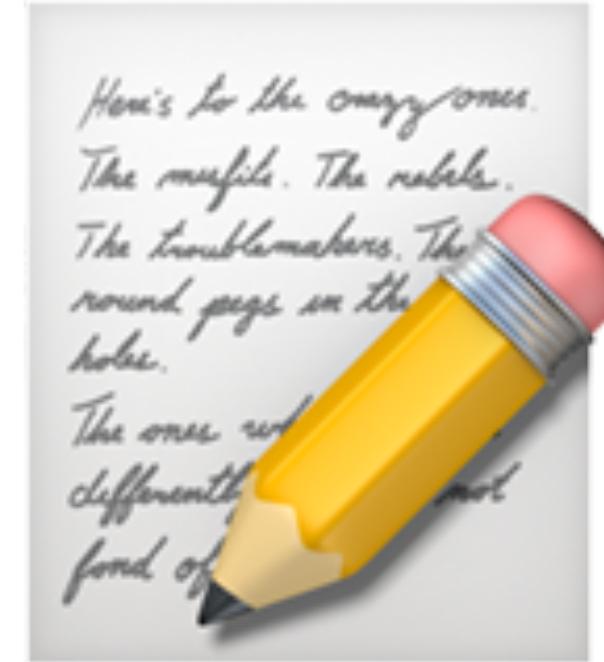
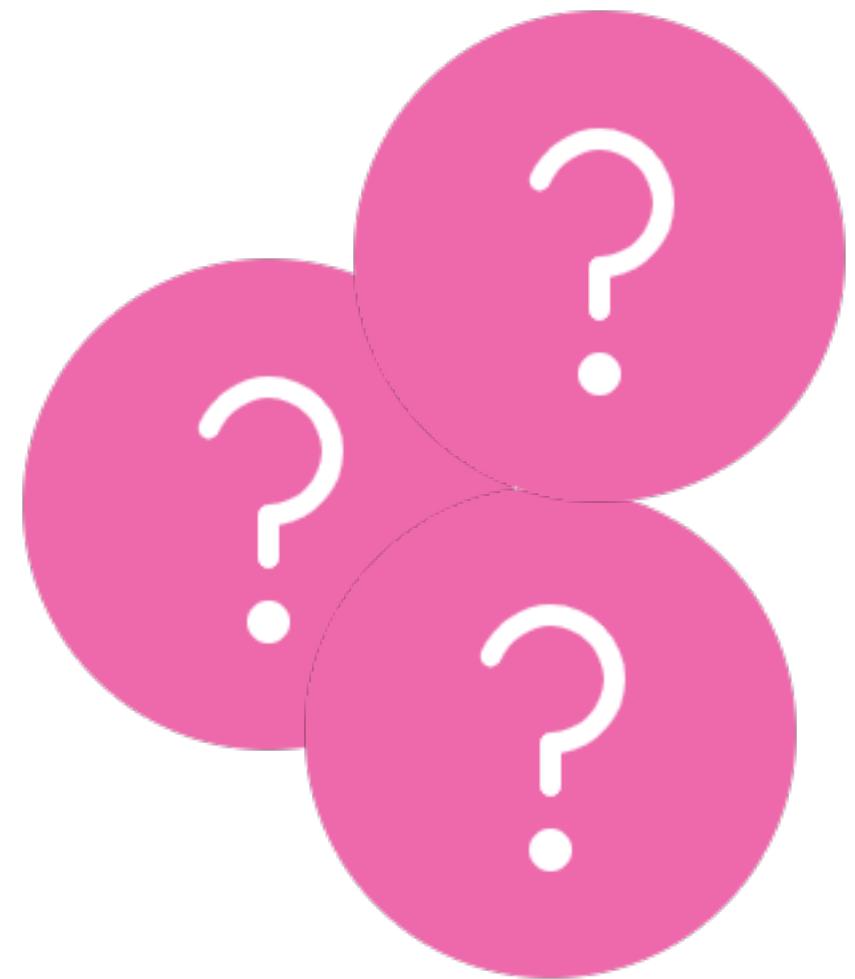
**What I'm working on next:**

- What's your next course of action? (based on the above)
- Why?
- What's holding you back?



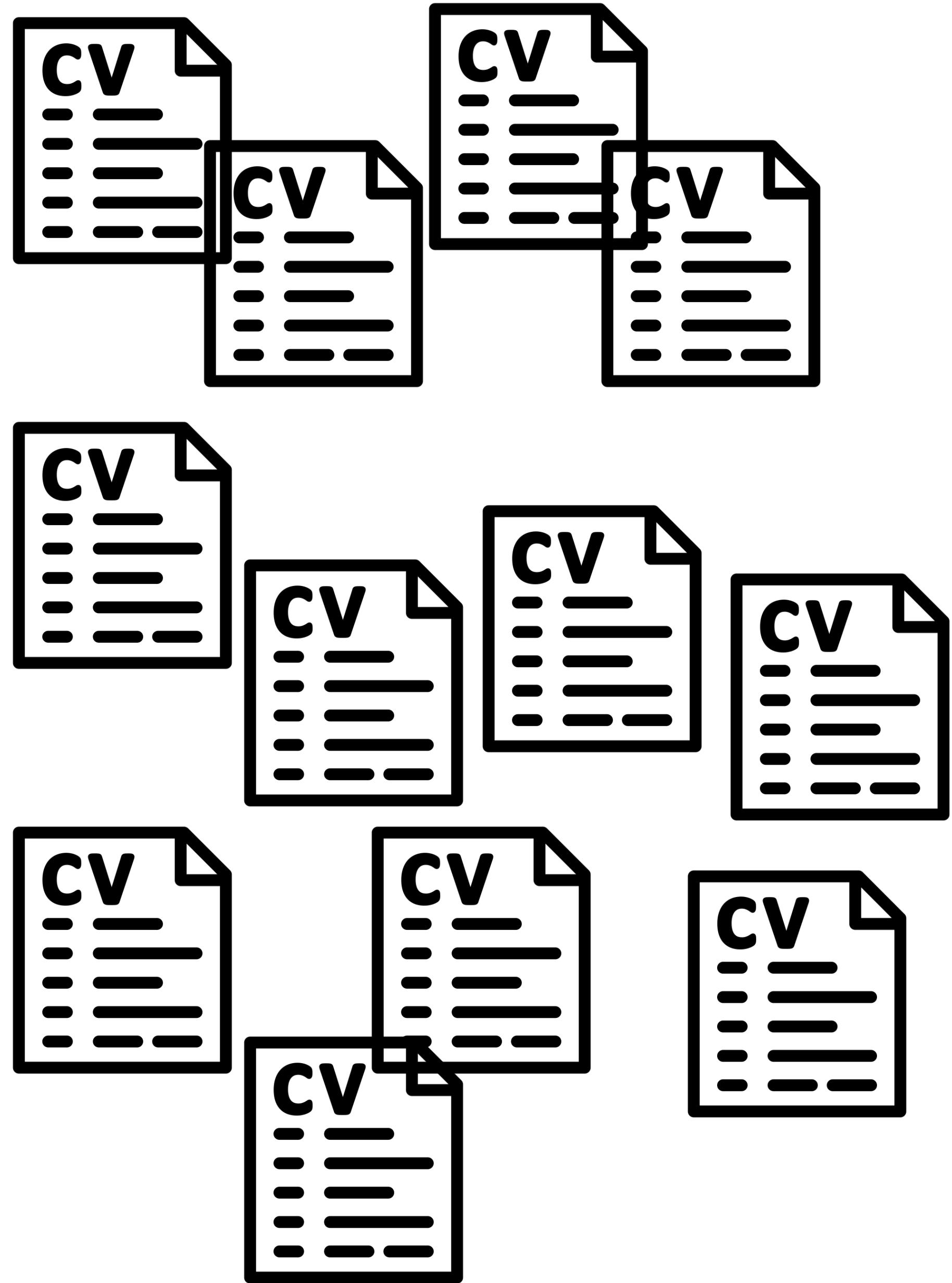
**Relate back to overall project goal**

# Take note of overlaps

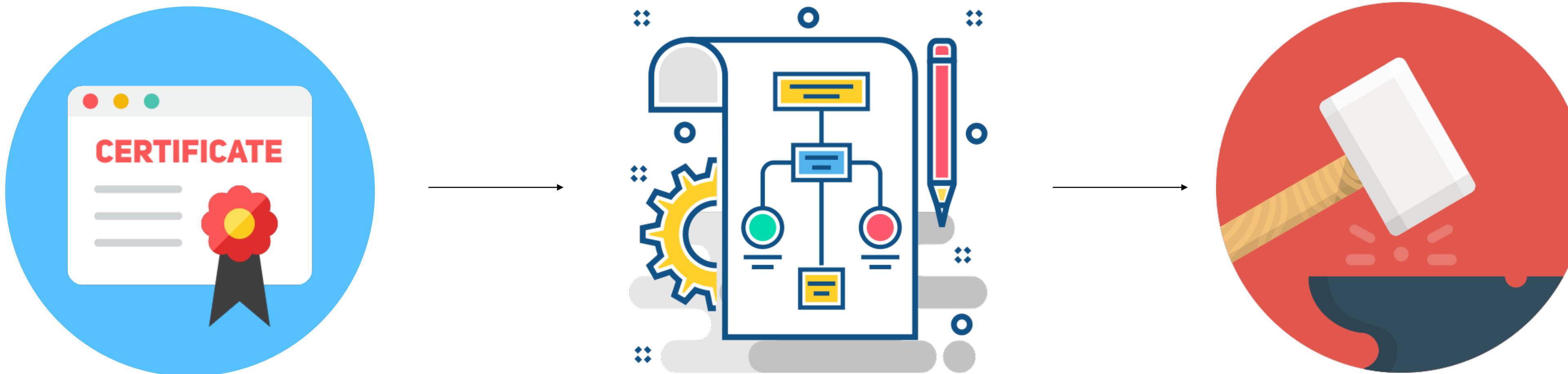


# Start the job before you have it

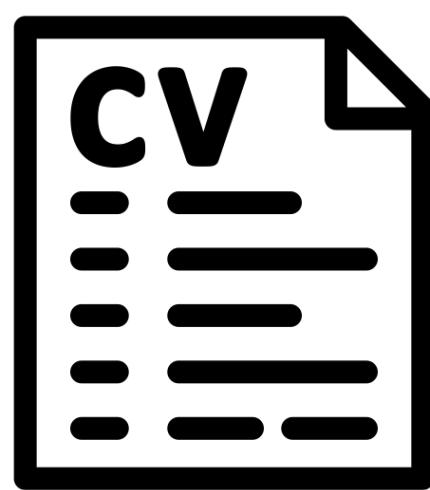
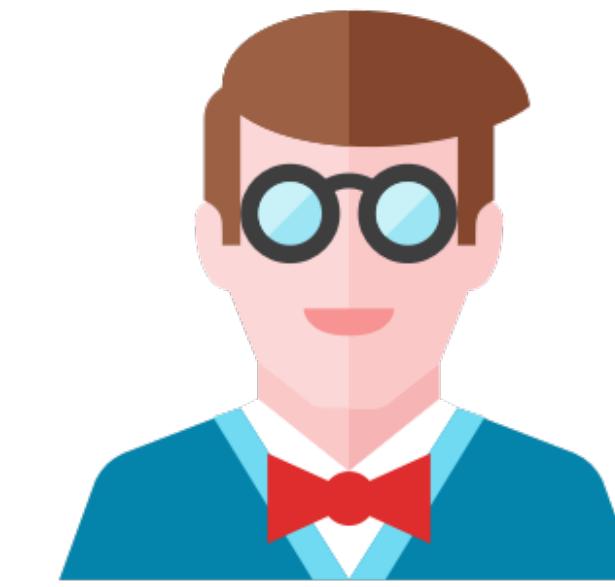
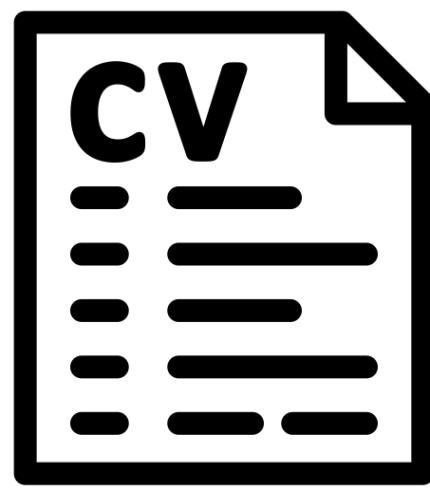




# The weekend project principle



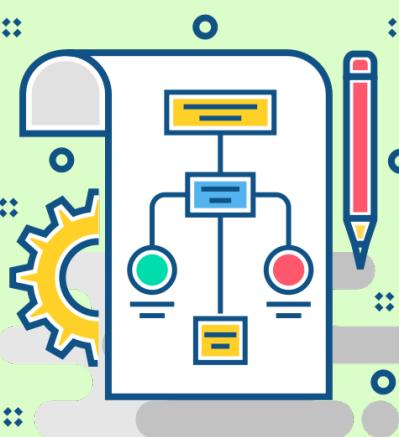
(what you work on in your own time)



# Acme Co.

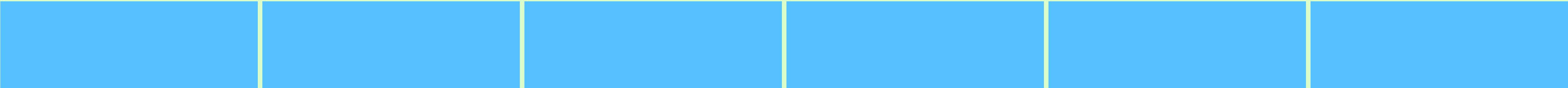


Documented on your blog



6-week project

---



# Communicating with people outside your team

**People on your team**

Boss

Project manager

Teammates

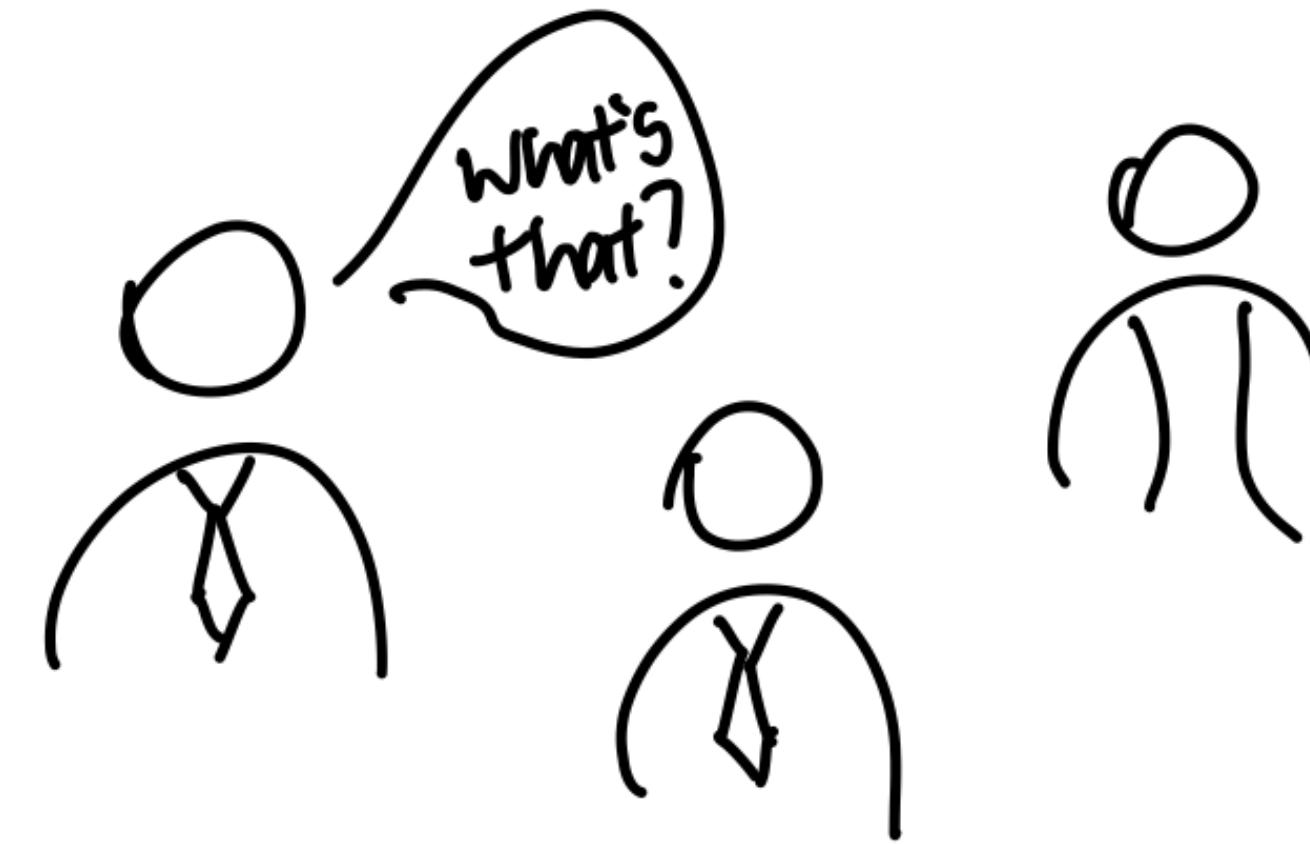
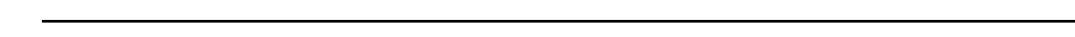
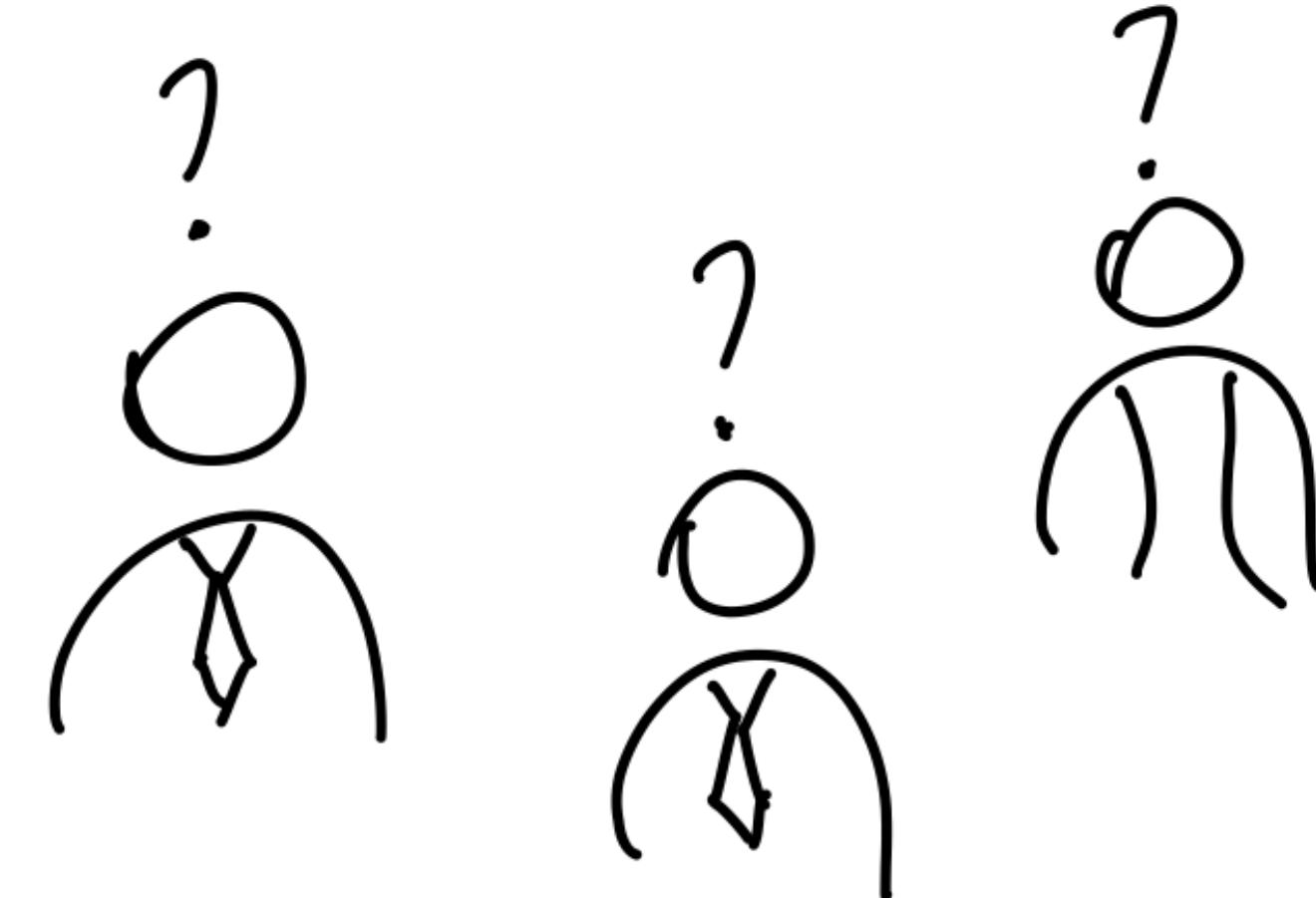
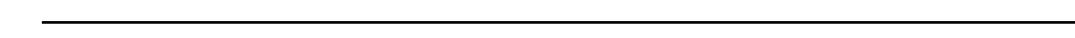
**People outside your team**

Clients

Customers

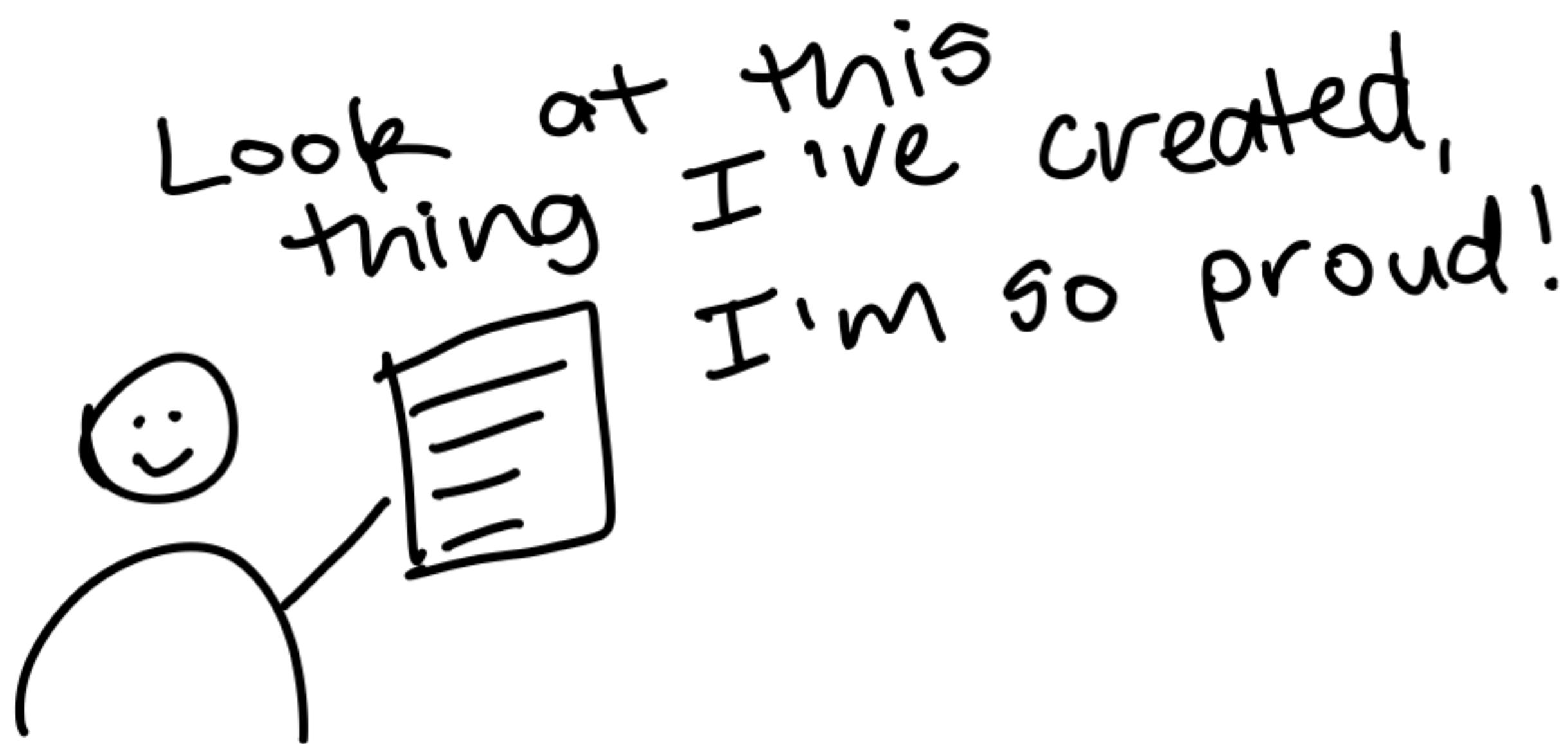
Fans

# Obvious to you, amazing to others

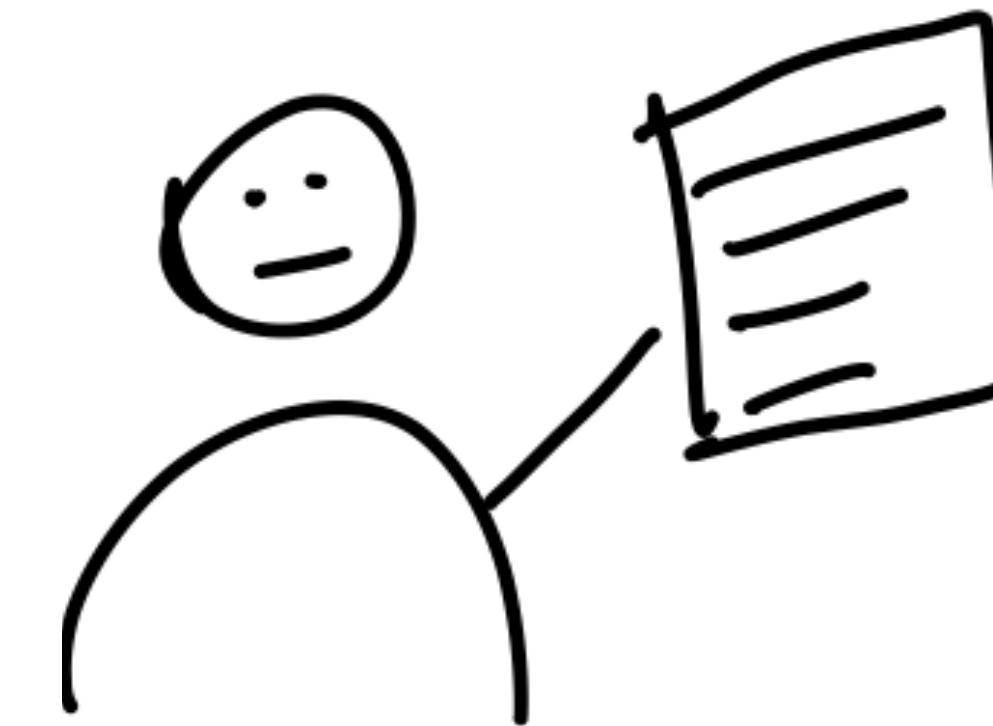


# What story are you trying to tell?

You:



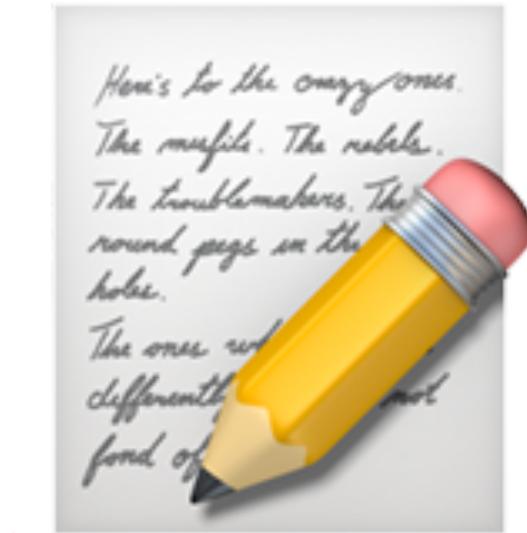
Also you:



# **Always ask, Who's it for?**

**“Who’s it for?” → “What do they need to know?” = Specific = Courage**

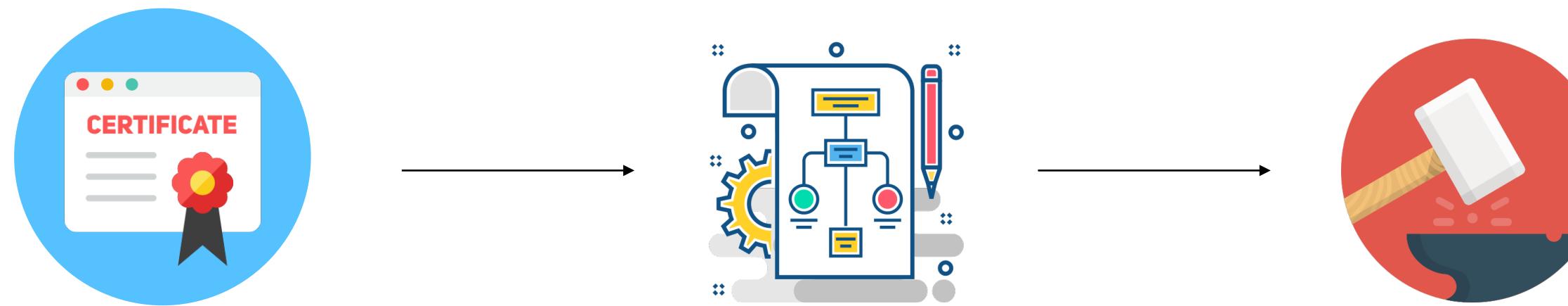
# Write it down



**What did you work on today?**

**What are you working this week?**

# Progress, not perfection



**“Here’s what I’ve done.”**