

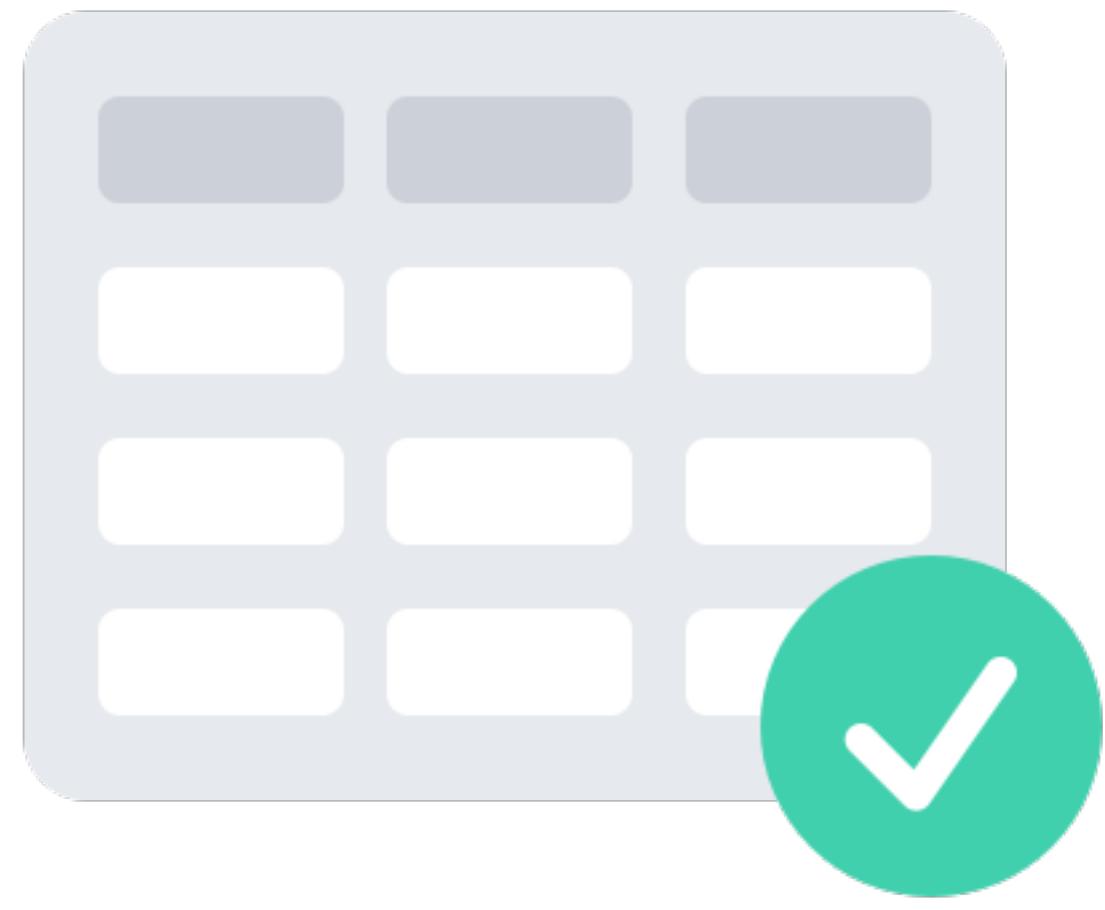
What is machine learning?



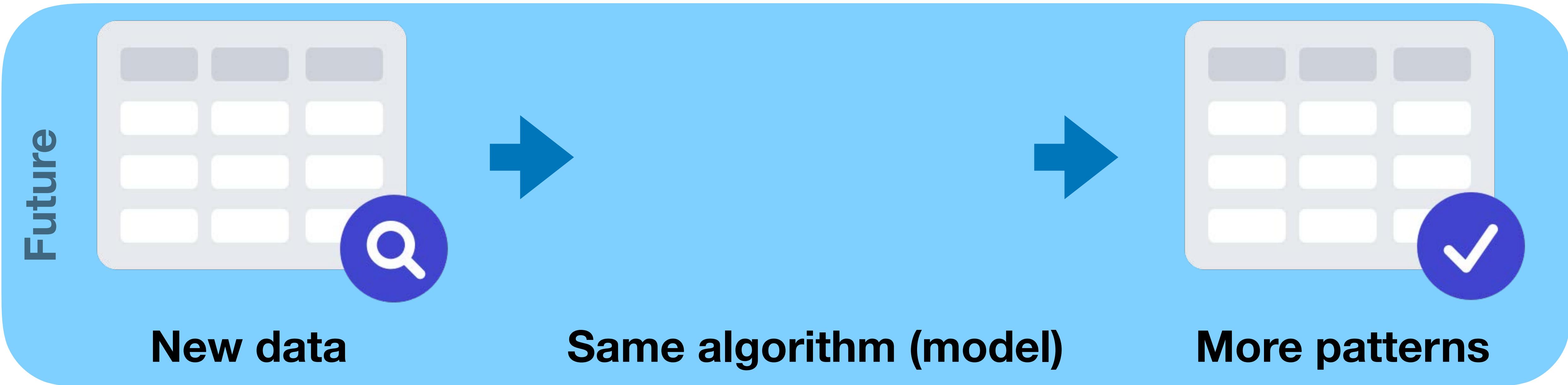
Data



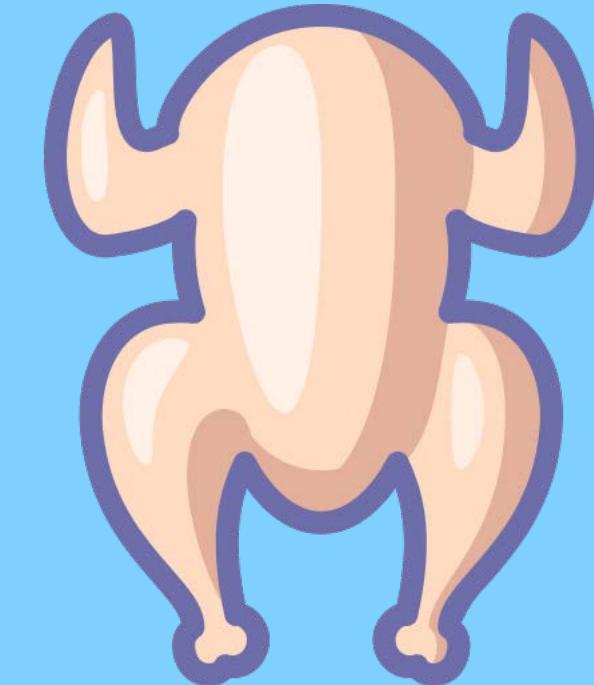
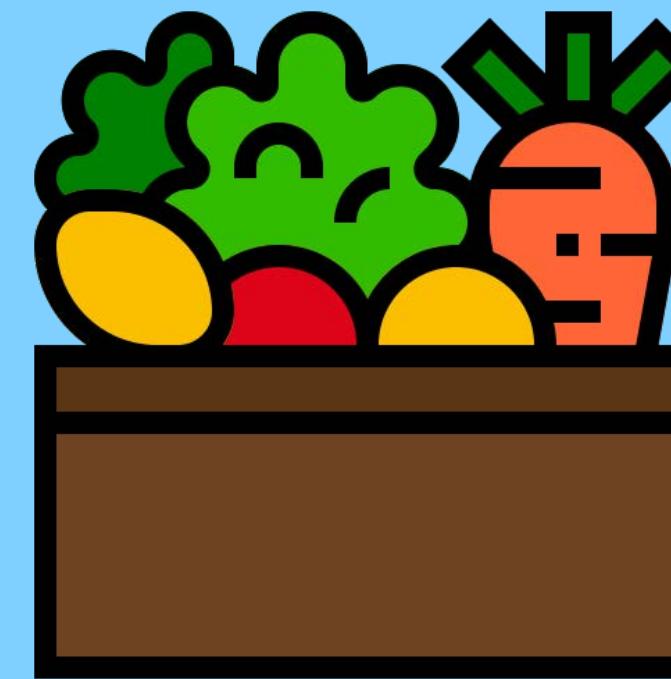
**Machine learning
algorithm**



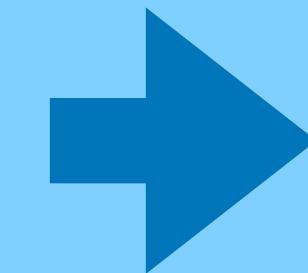
Patterns



Normal algorithm



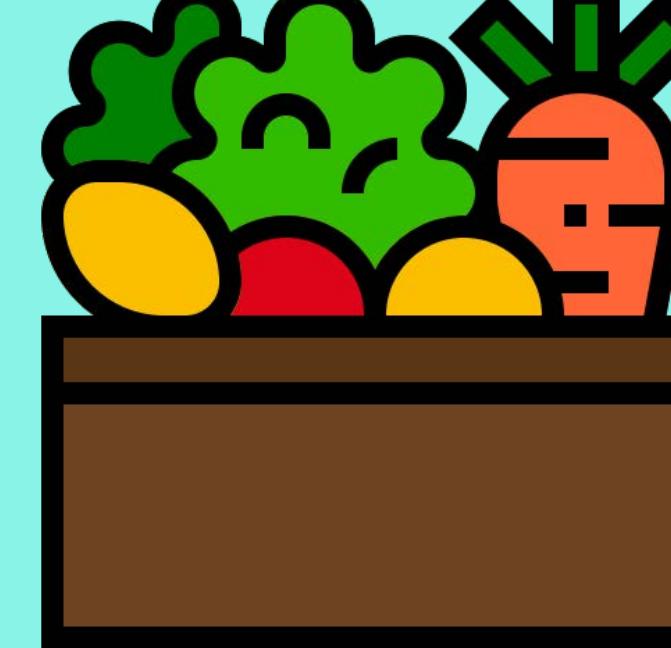
1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



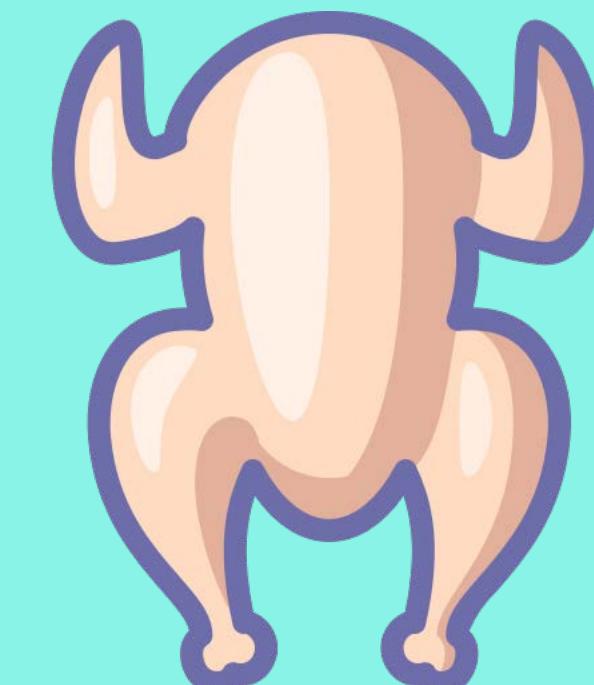
Starts with

Makes

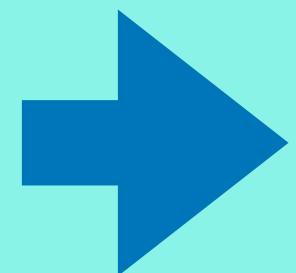
Machine learning algorithm



Inputs



Output



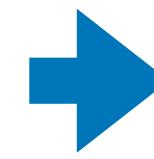
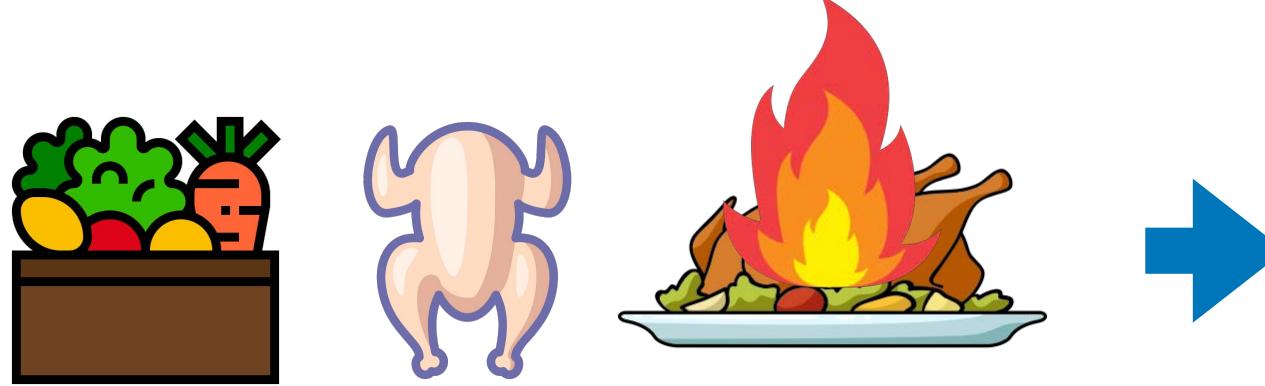
1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables

Starts with

Figures out

Attempt

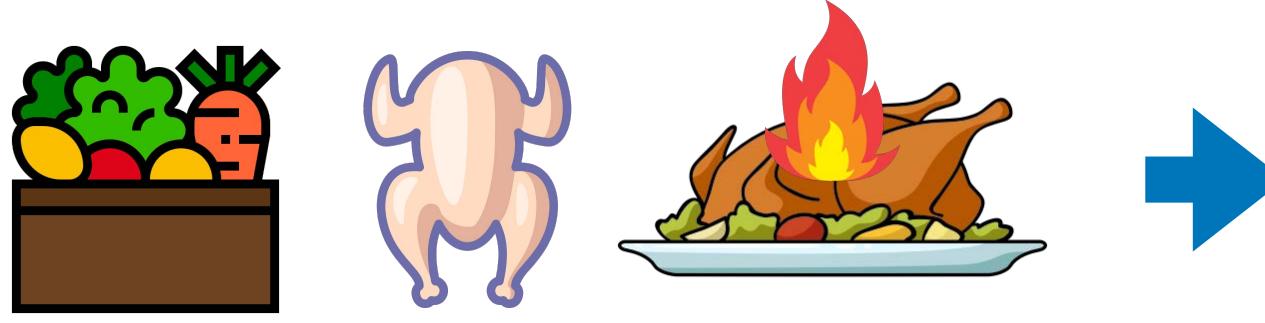
1



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



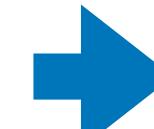
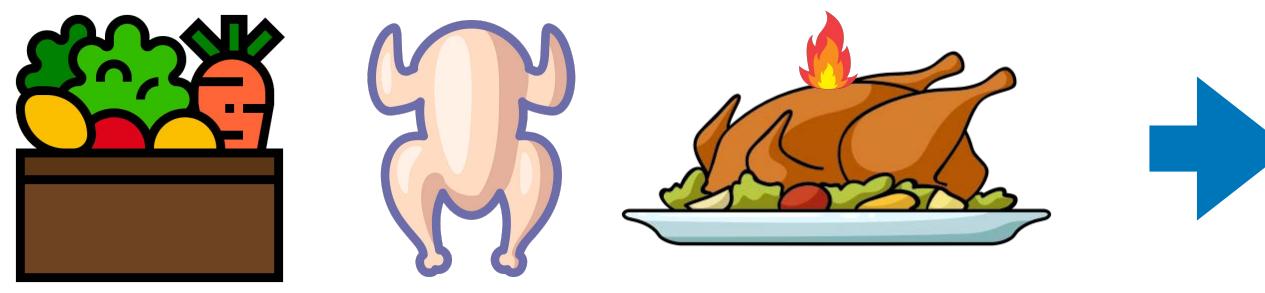
2



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



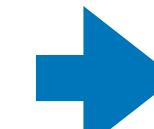
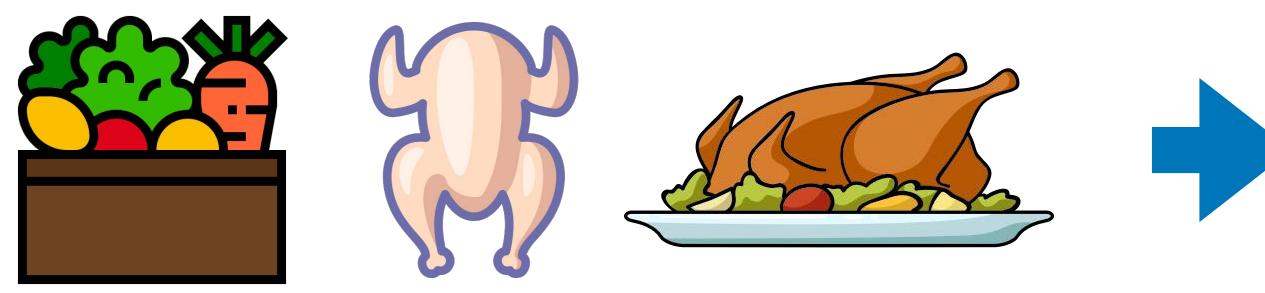
3



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



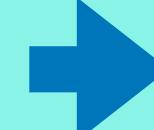
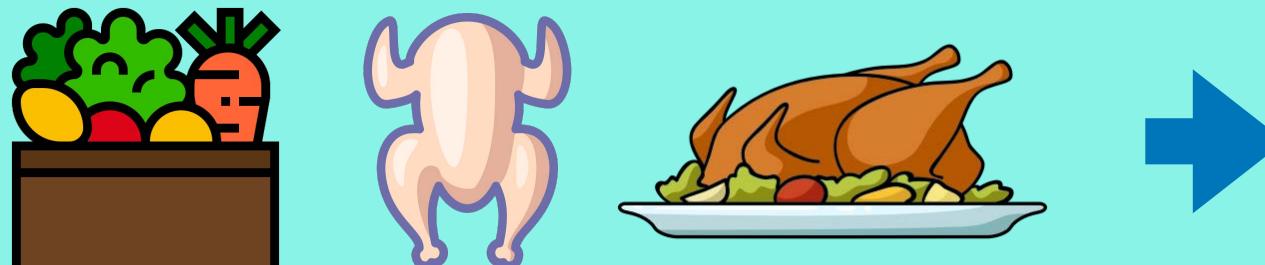
4



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



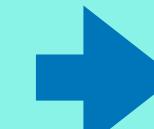
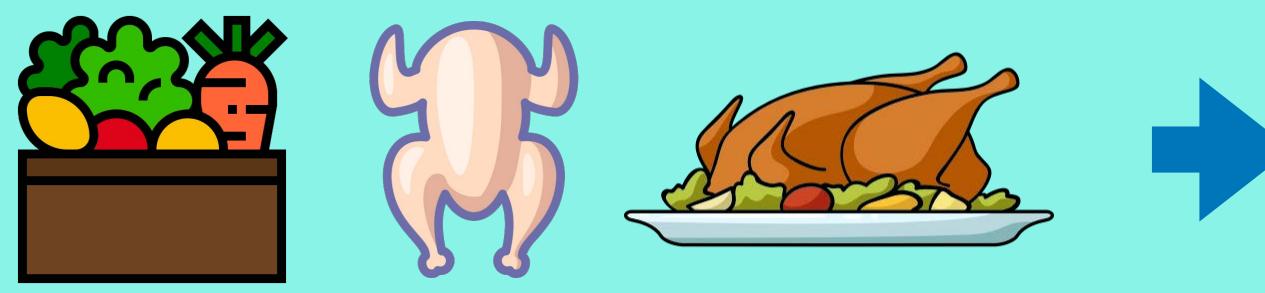
⋮



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

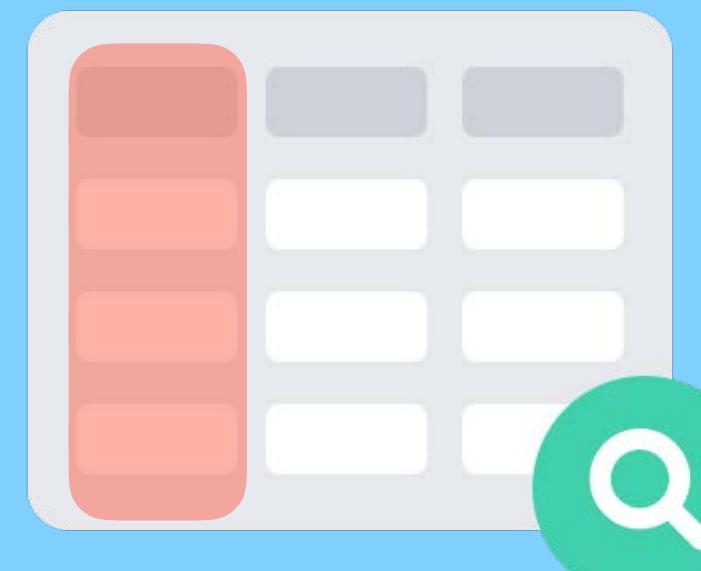


Machine learning algorithms
may try 1000's of times to find
the right instructions.

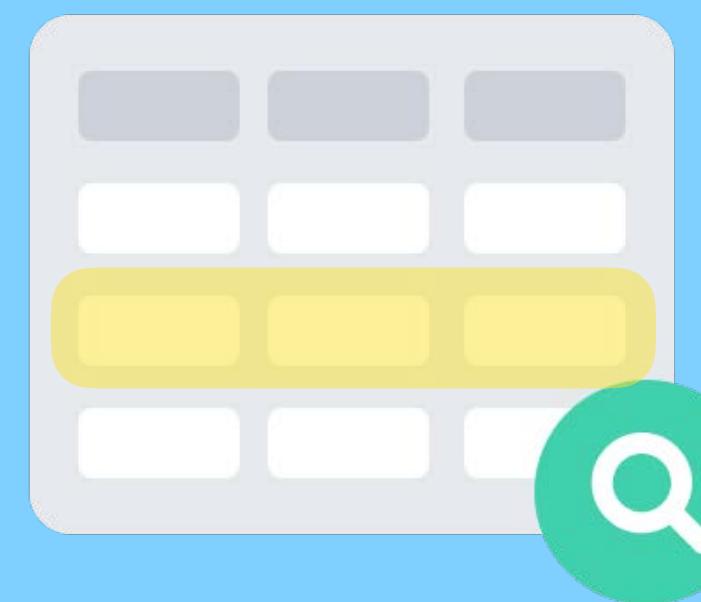
Data science

Data analysis

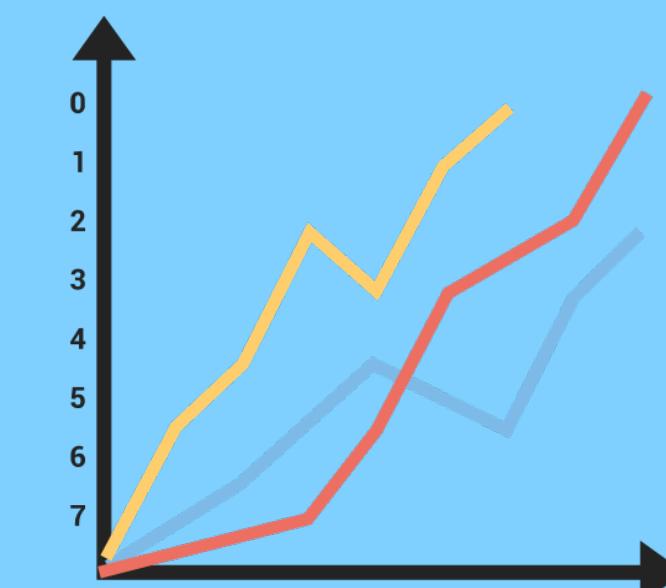
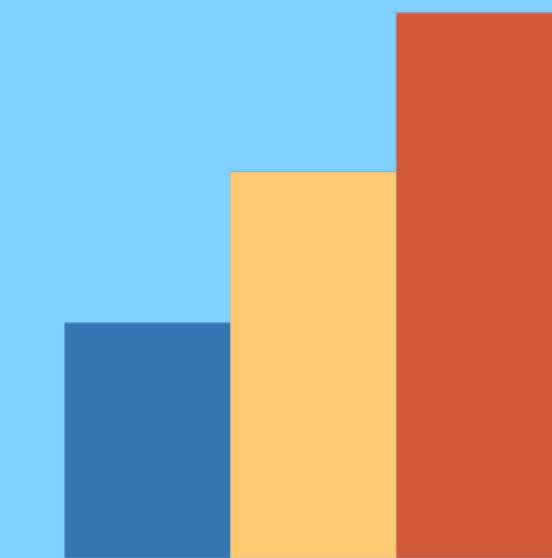
Data



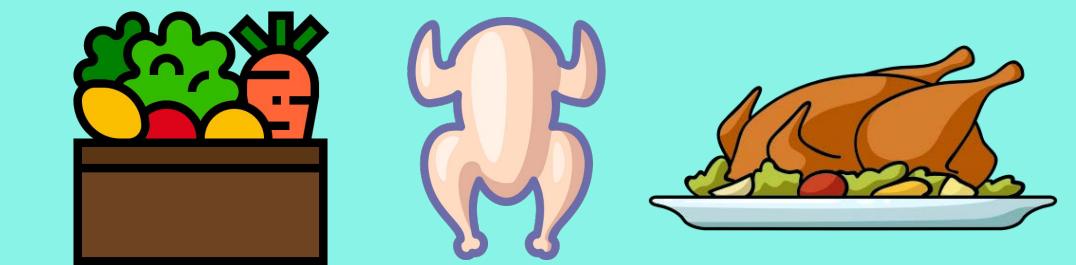
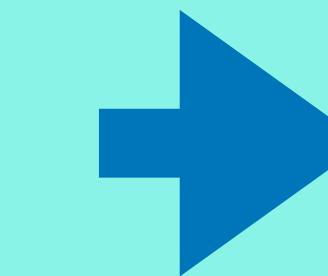
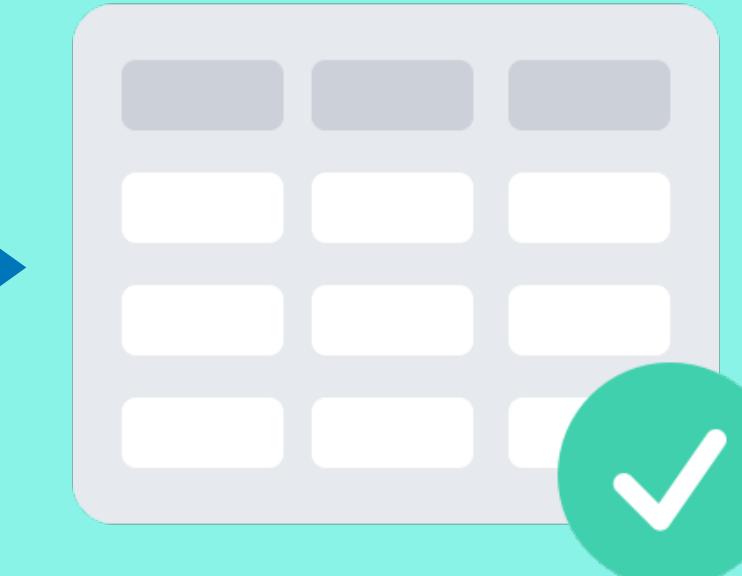
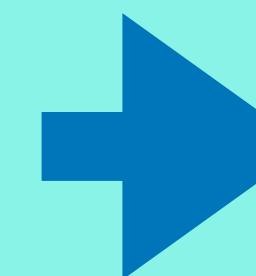
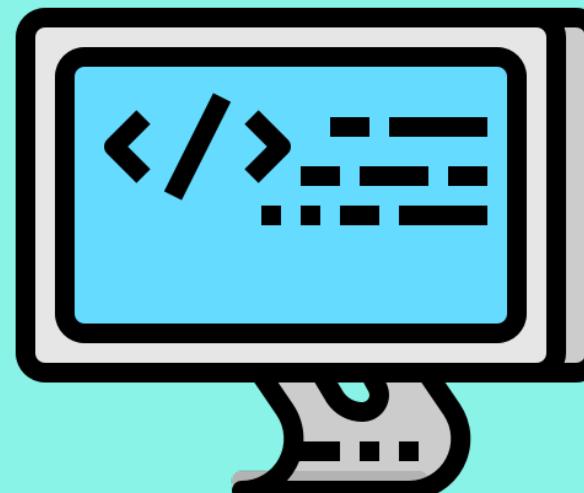
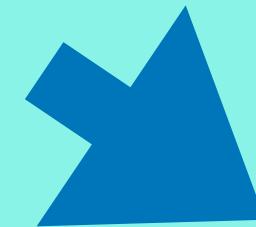
vs.



vs.



Machine learning



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

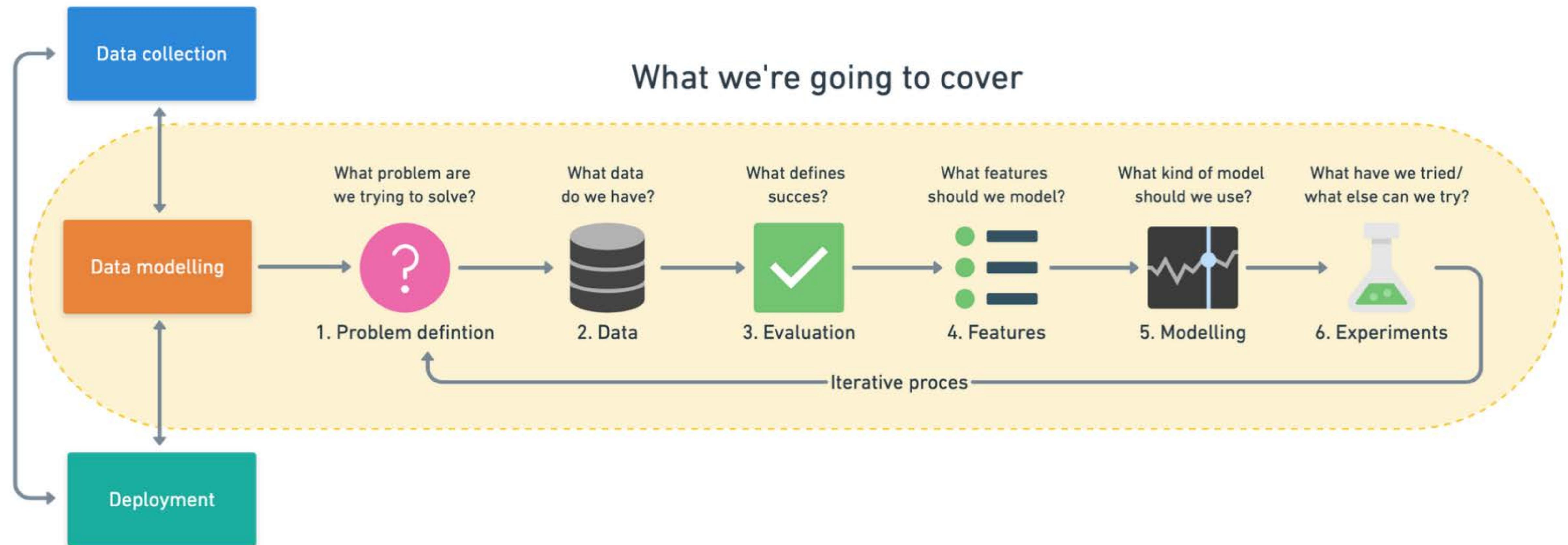
Instructions in your daily life?

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

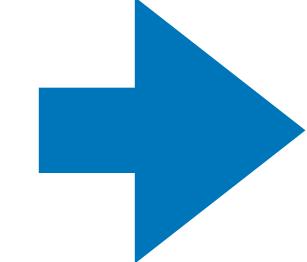
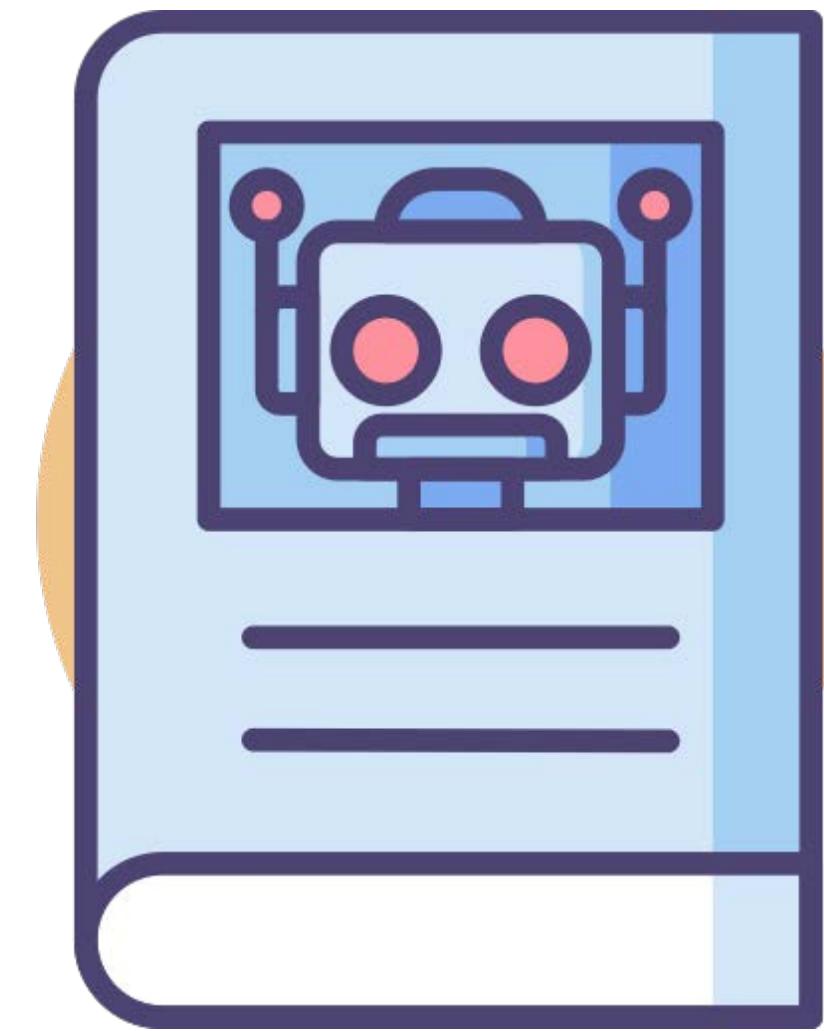
We're focused on...

- Practical solutions
- Writing machine learning code

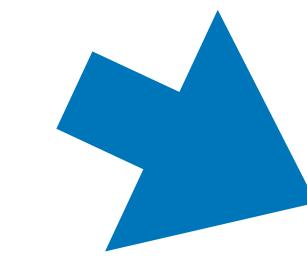
Steps in a full machine learning project



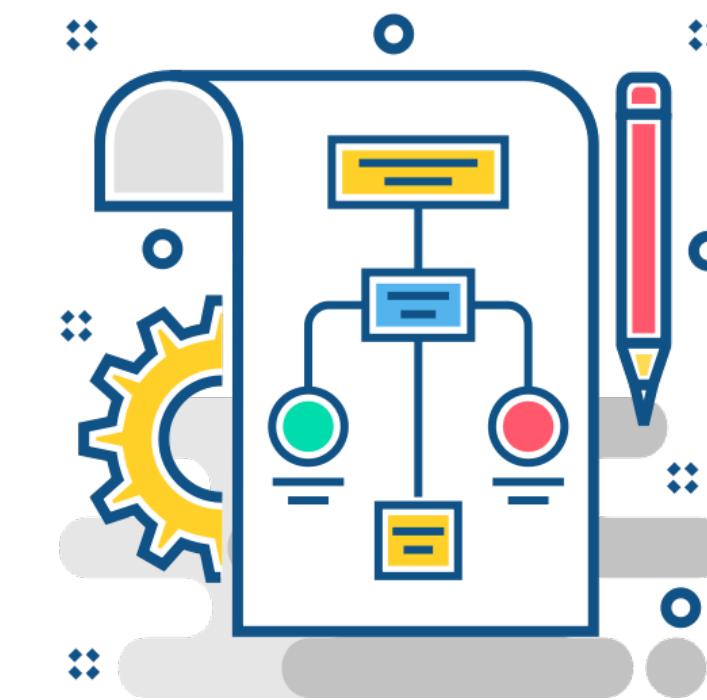
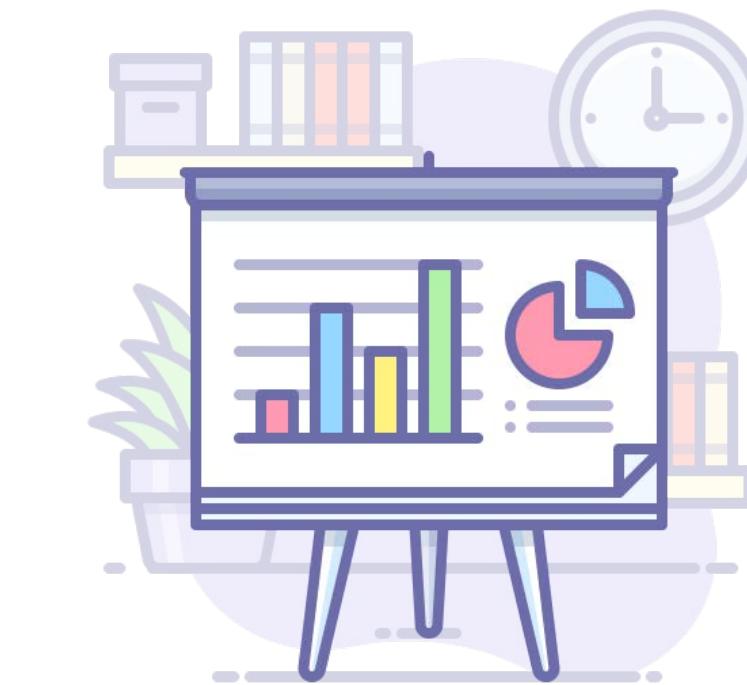
1. Create a framework



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



3. Learn by doing



Yes

Write code

No

Overthink the process

Make mistakes

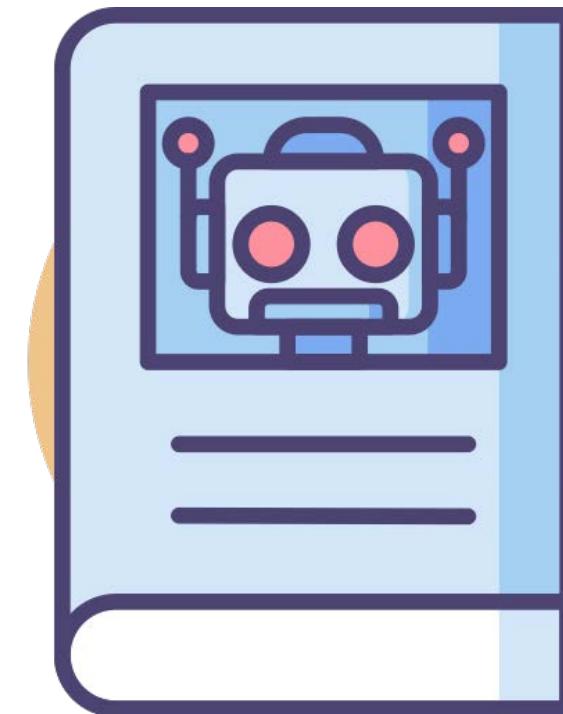
Try make things perfect

Build projects

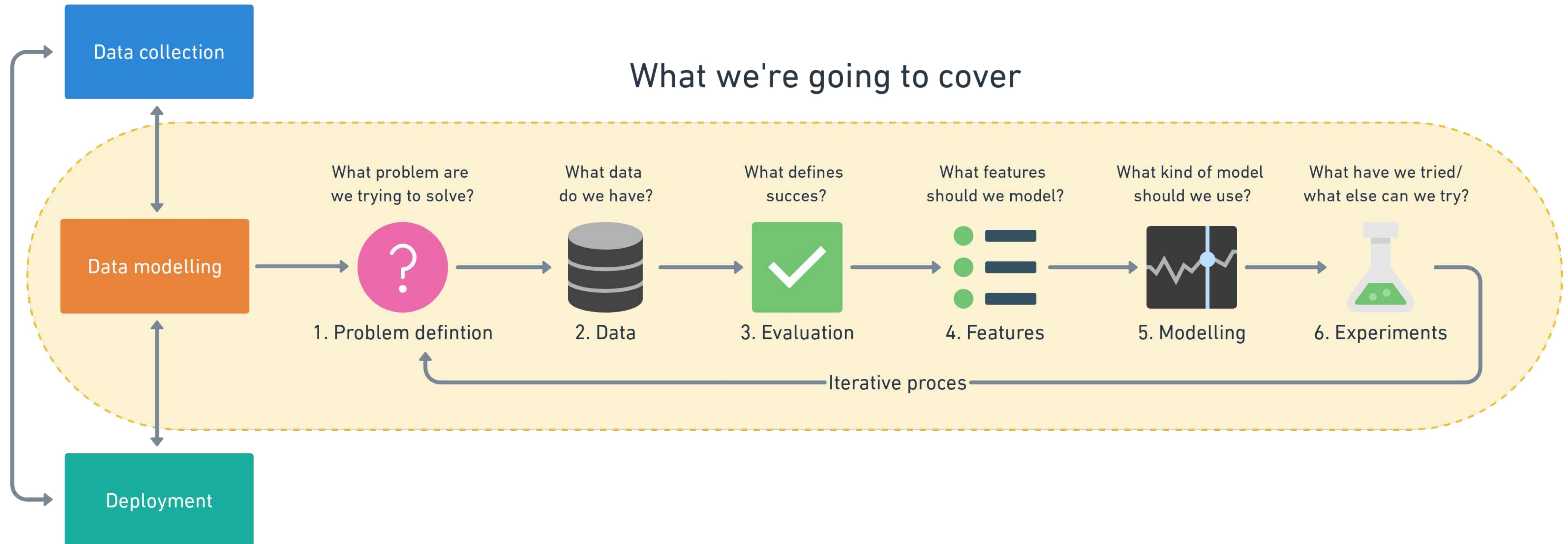
Build things from scratch

Learn what matters

The framework we'll be using



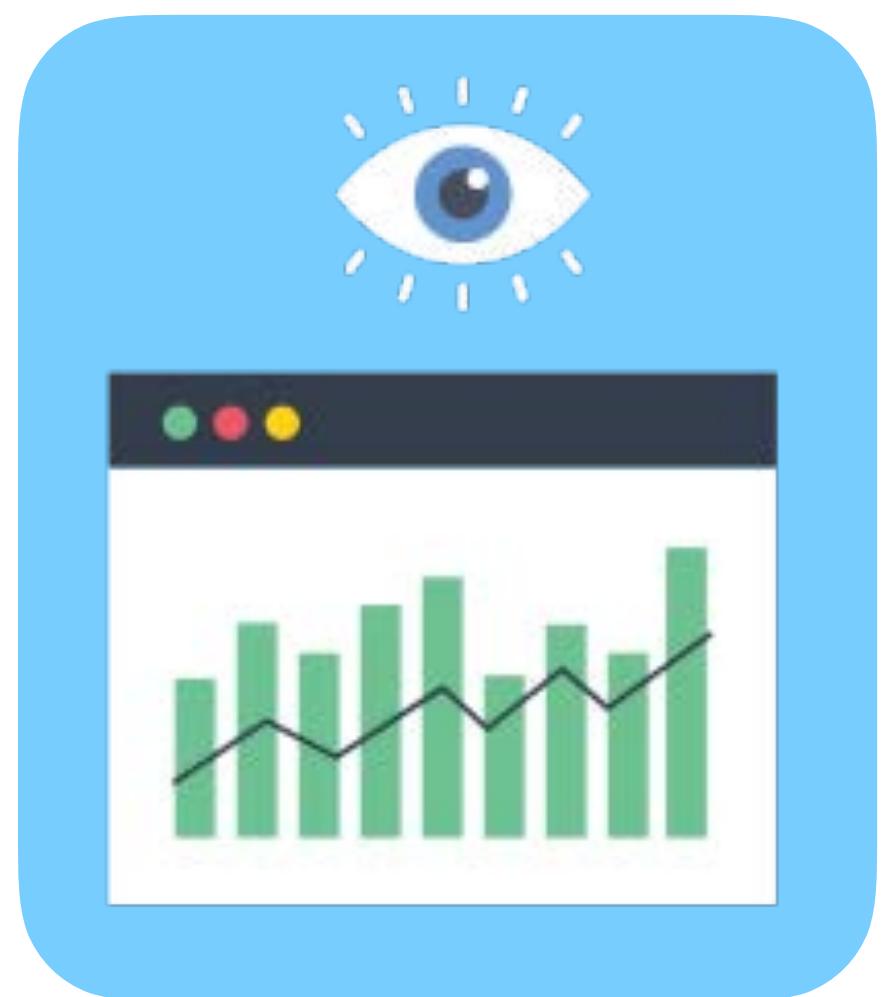
Steps in a full machine learning project



1. Problem definition



“What problem are we trying to solve?”



Supervised



Unsupervised



Classification



Regression

2. Data



“What kind of data do we have?”



Structured

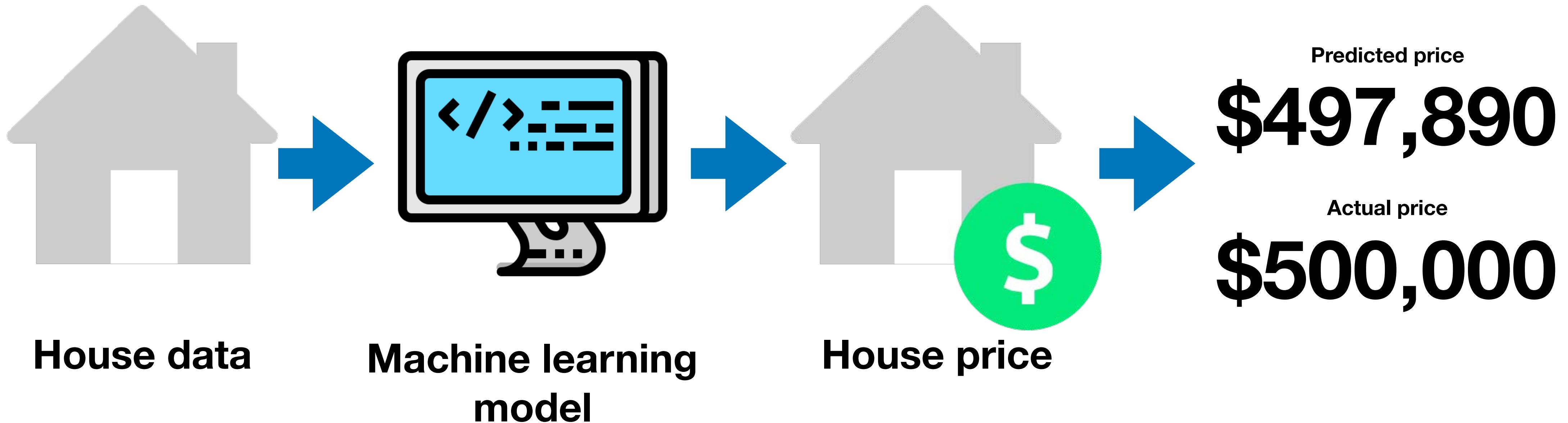


Unstructured

3. Evaluation



“What defines success for us?”



4. Features



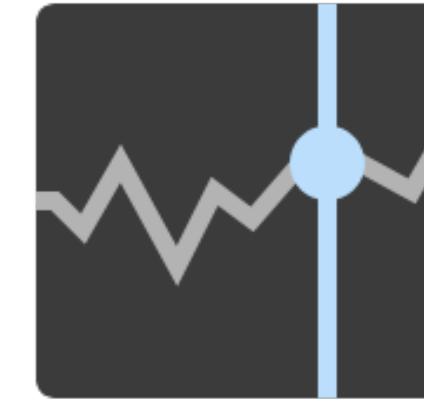
“What do we already know about the data?”



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

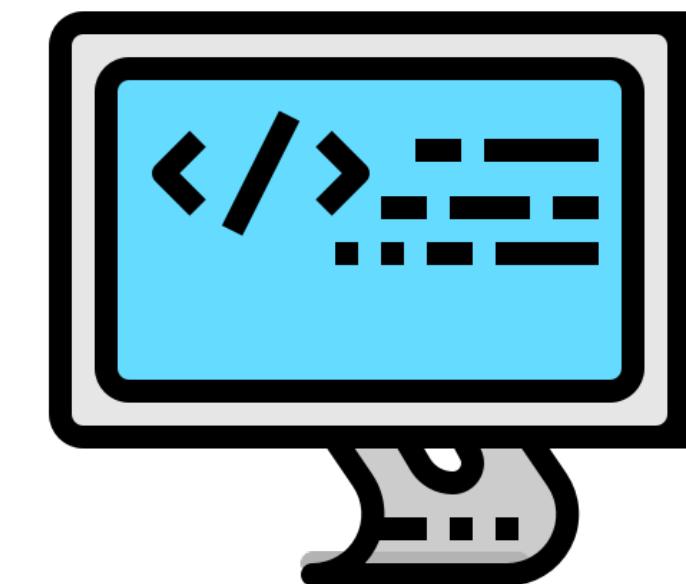
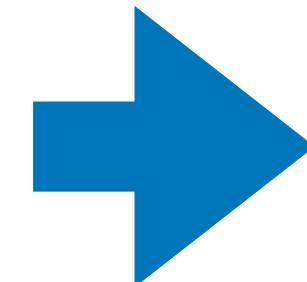
5. Modelling



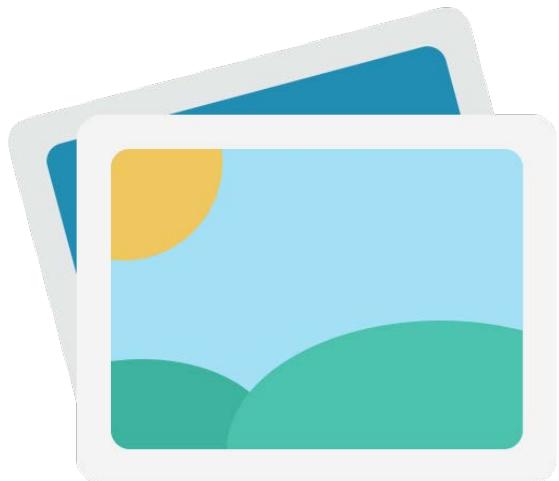
“Based on our problem and data, what model should we use?”



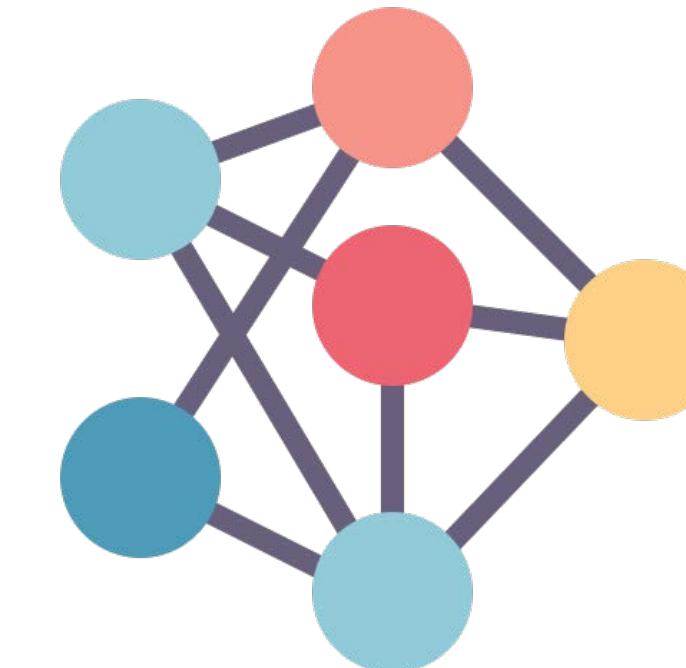
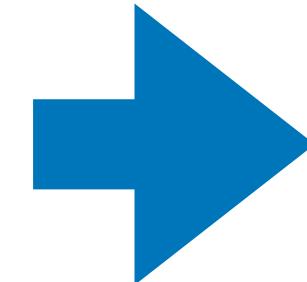
Problem 1



Model 1



Problem 2



Model 2

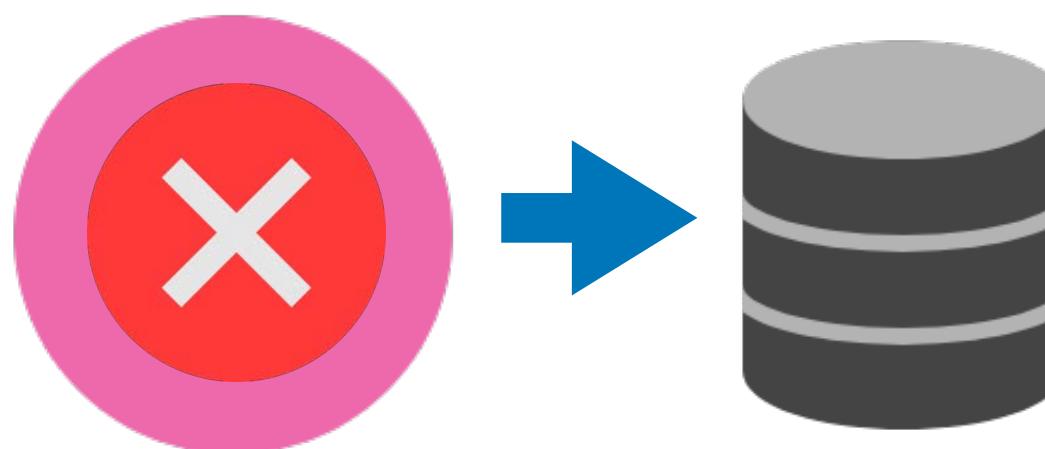
6. Experimentation



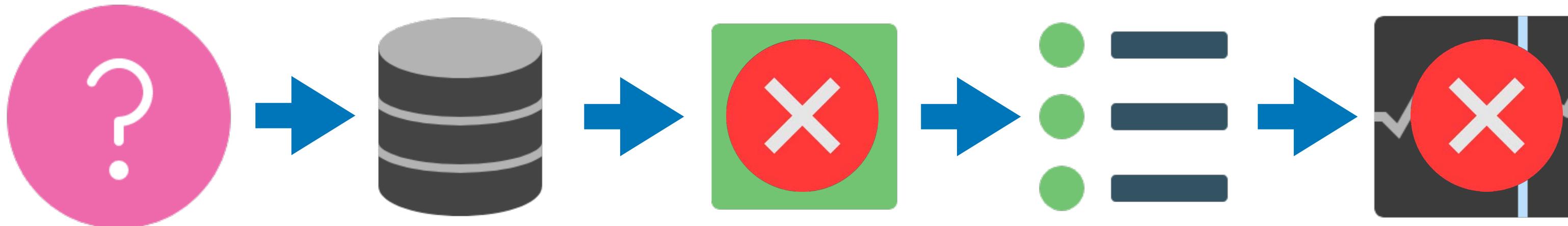
“How could we improve/what can we try next?”

Attempt

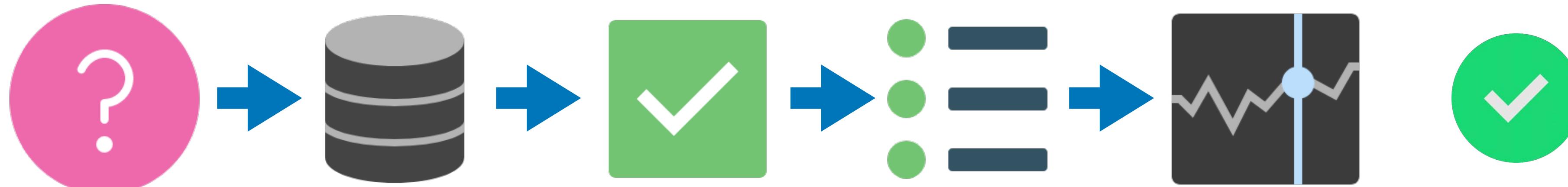
1



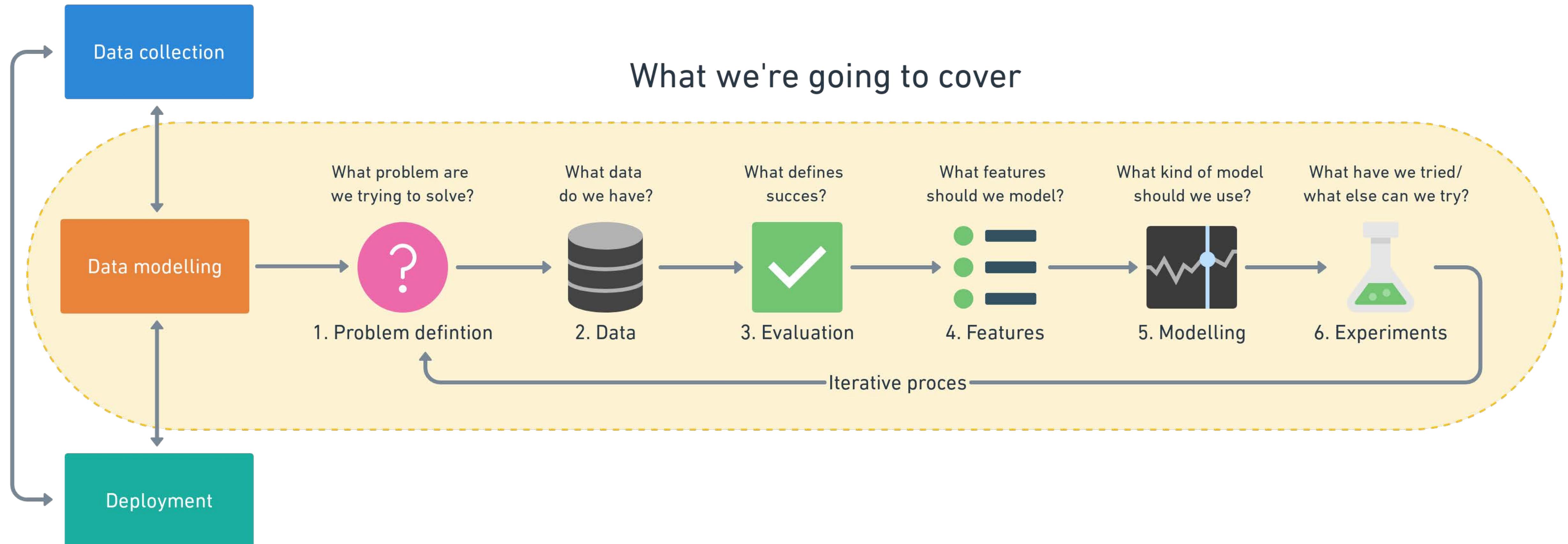
2



3



Steps in a full machine learning project



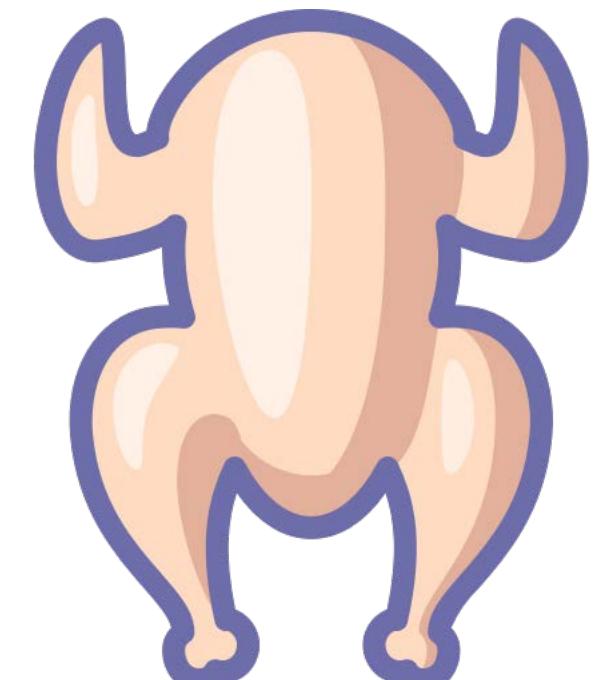
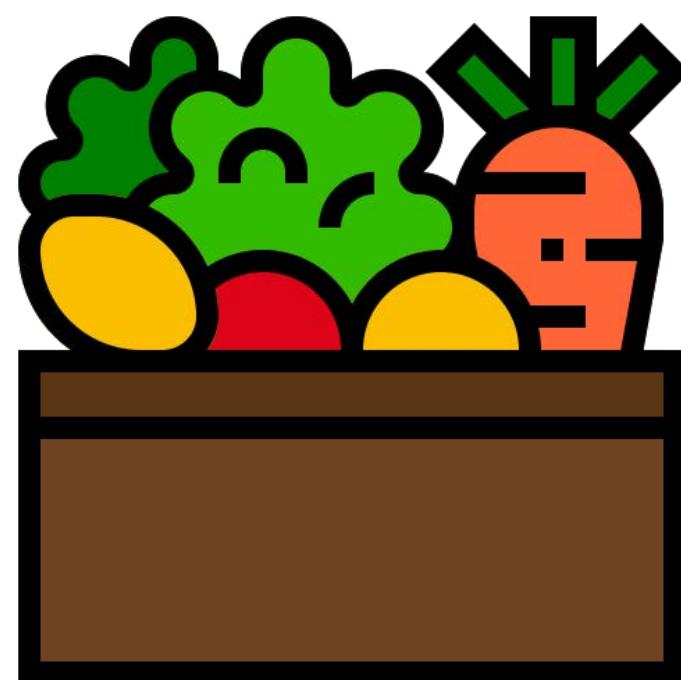
1. Problem definition



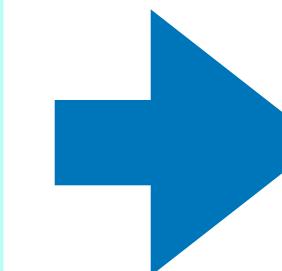
“What problem are we trying to solve?”

When shouldn't you use machine learning?

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



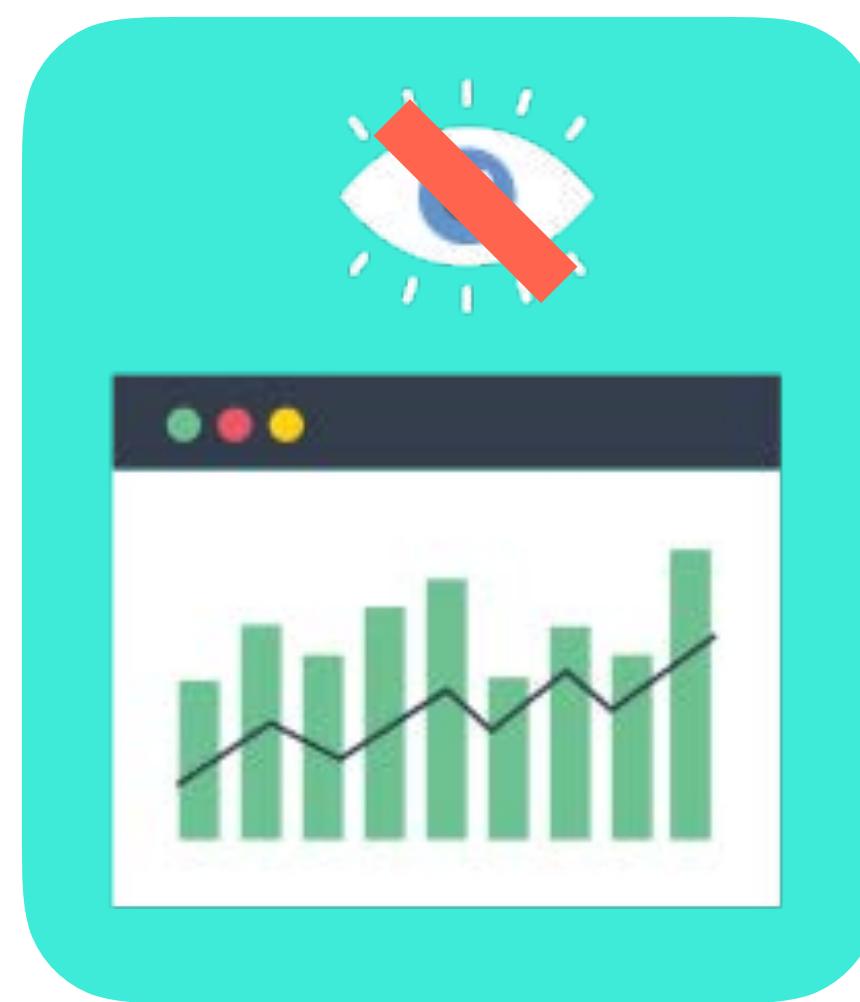
1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



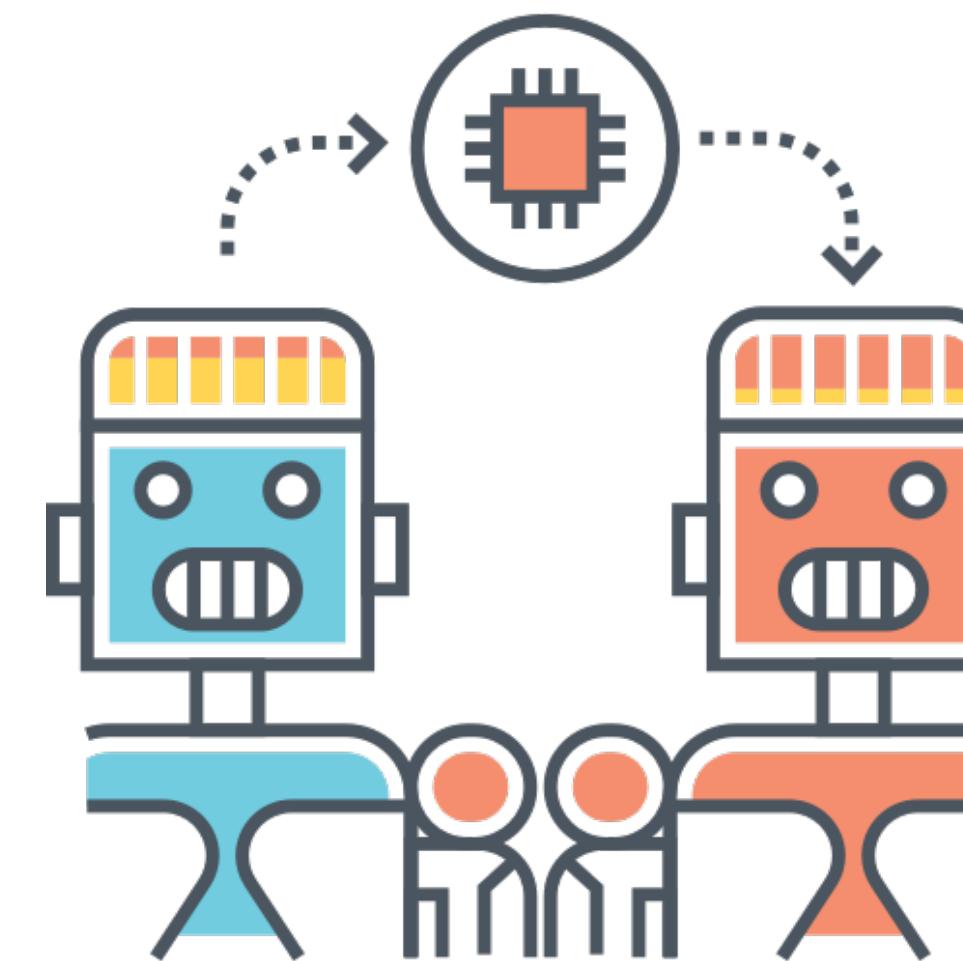
Main types of machine learning



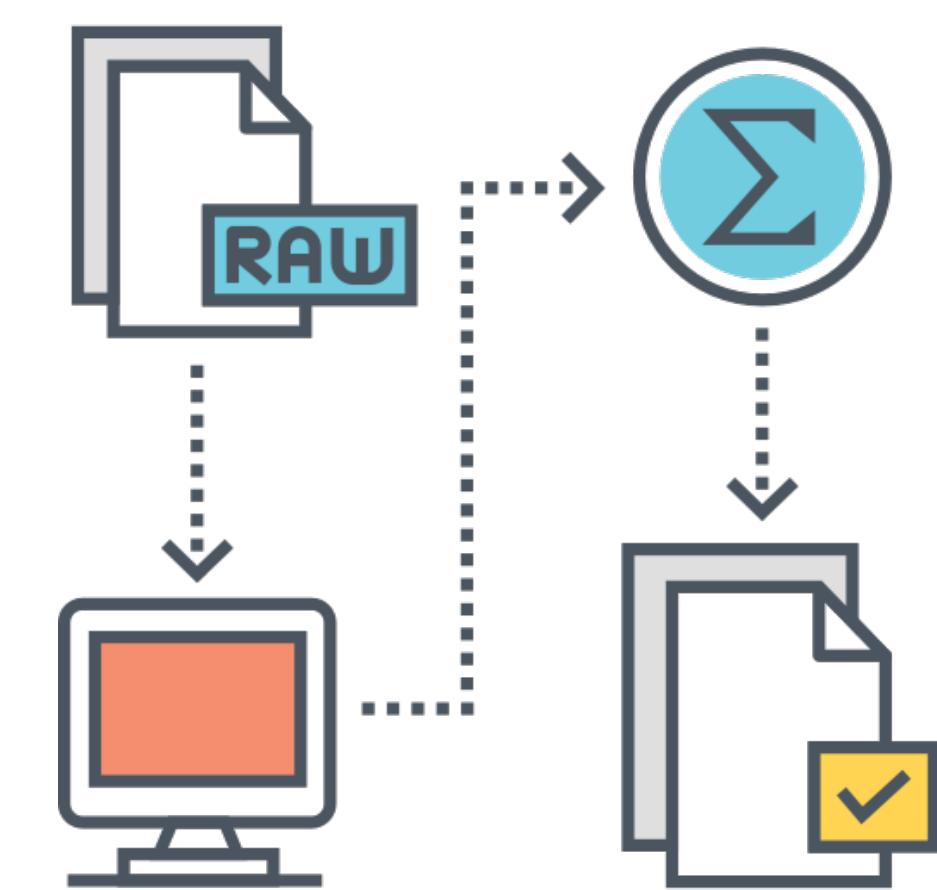
**Supervised
Learning**



**Unsupervised
Learning**

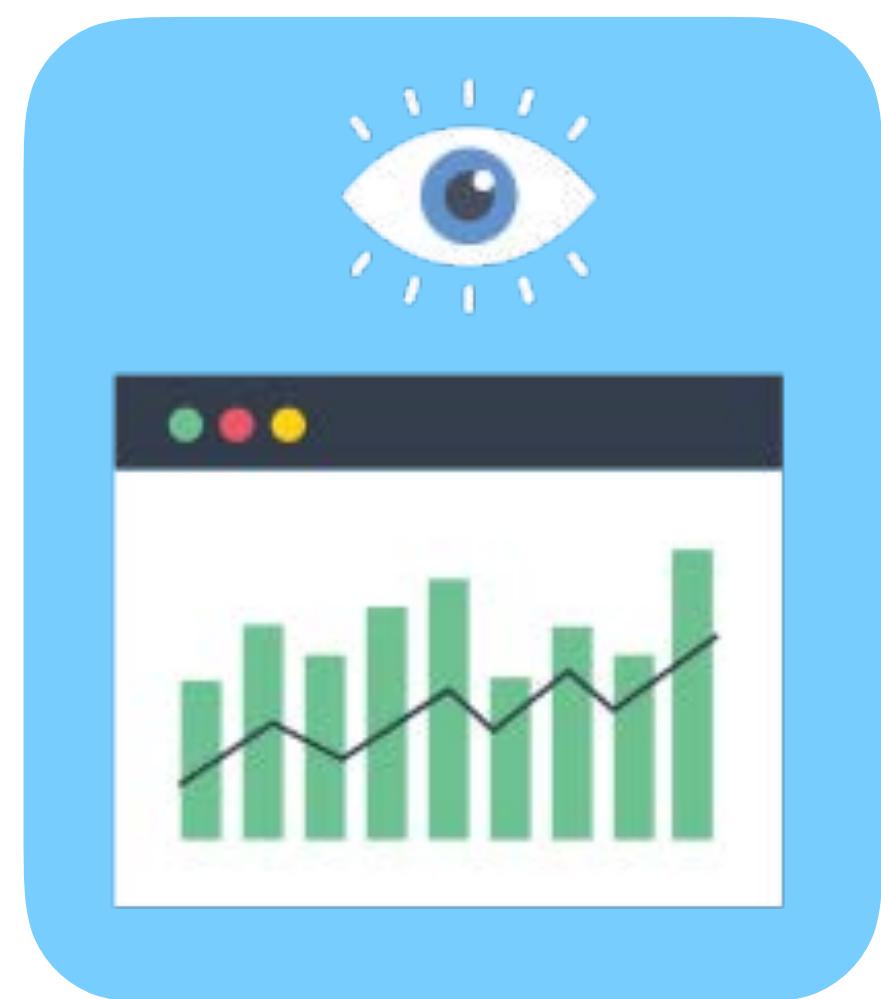


**Transfer
Learning**

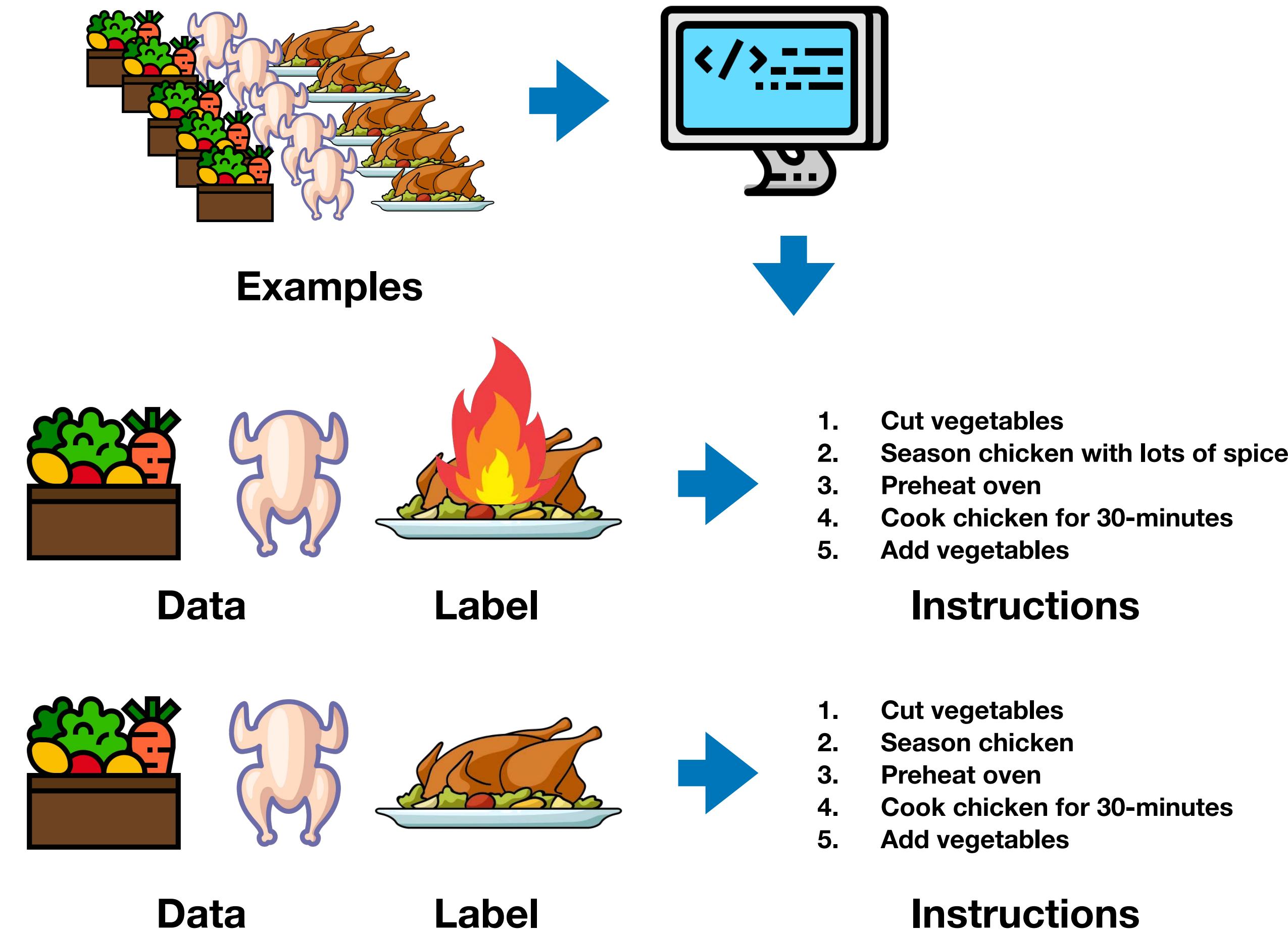


**Reinforcement
Learning**

Supervised learning



**Supervised
Learning**



Supervised learning



Classification

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

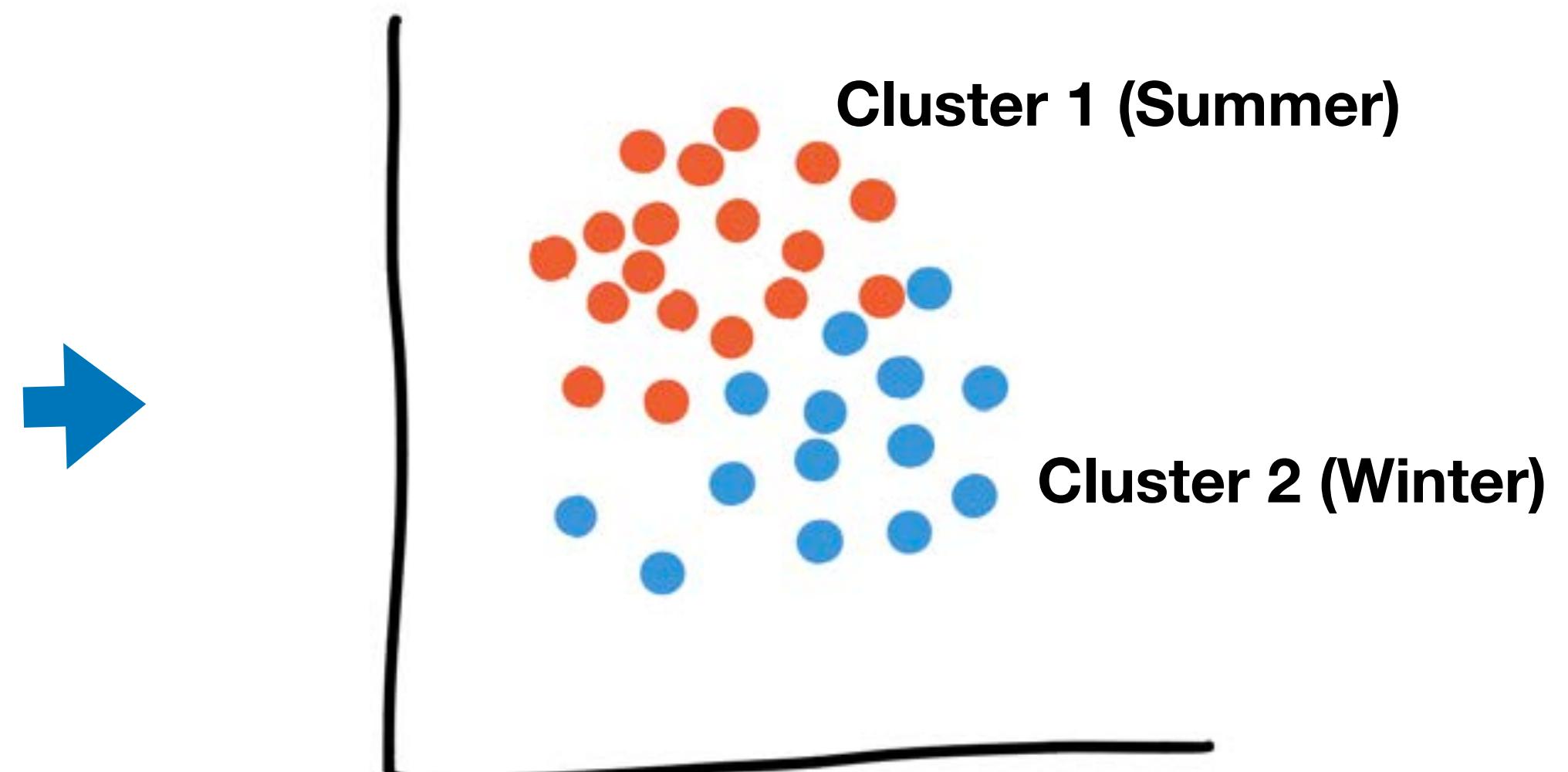


Regression

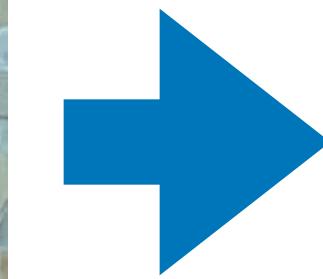
- “How much will this house sell for?”
- “How many people will buy this app?”

Unsupervised learning

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

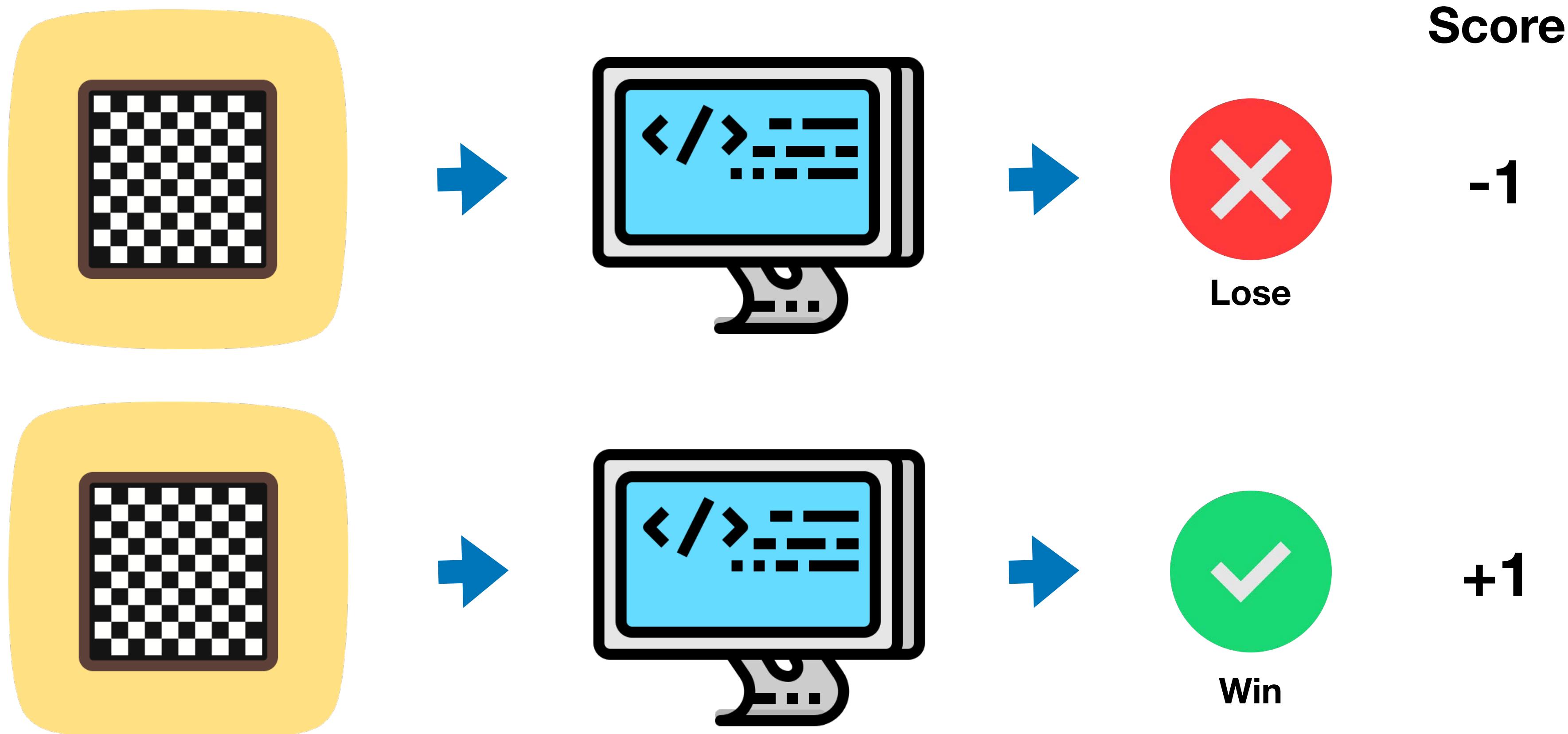


Transfer learning

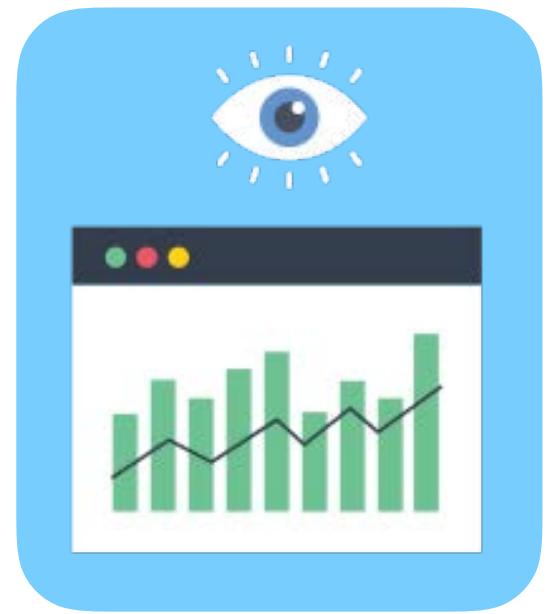


Use what a model has learned in one domain, to help in another domain.

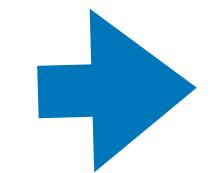
Reinforcement learning



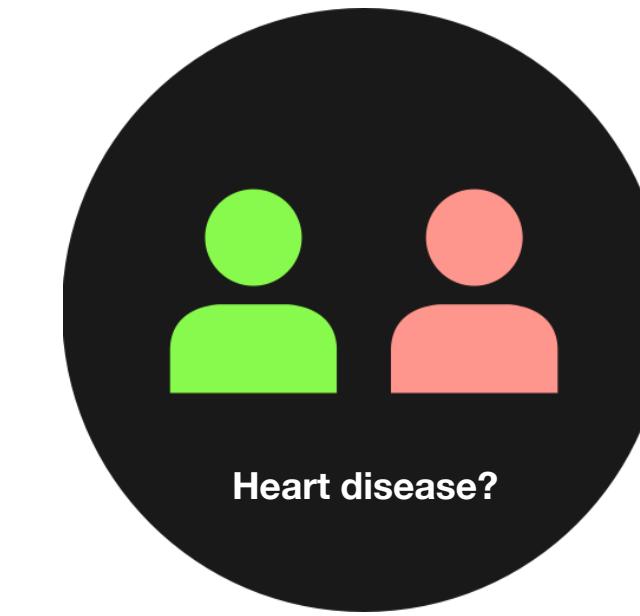
Matching your problem



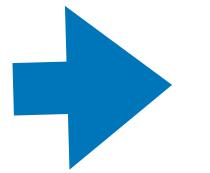
Supervised
Learning



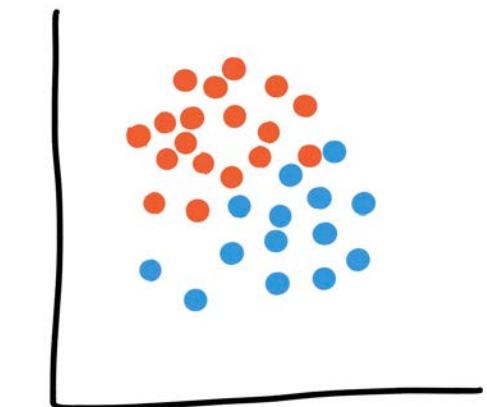
“I know my inputs and outputs.”



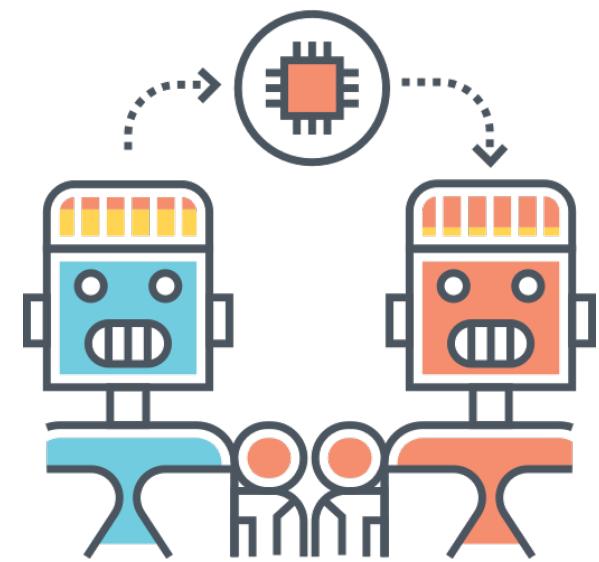
Unsupervised
Learning



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



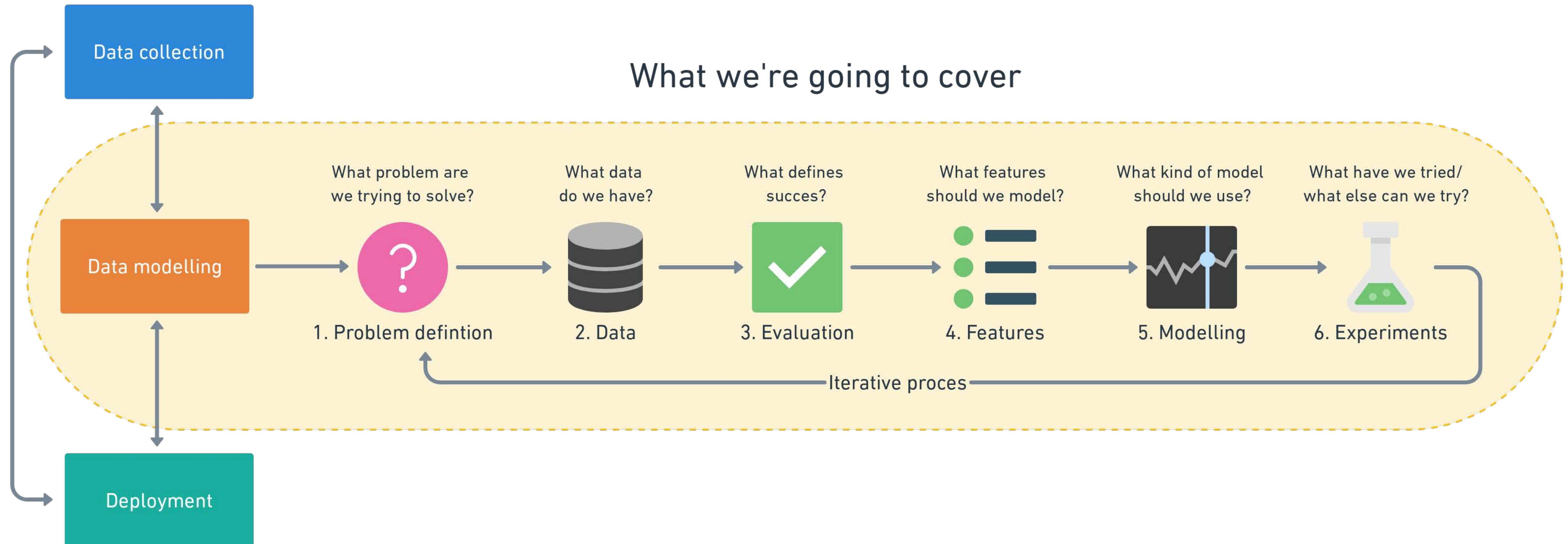
Matching your problem



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

**Transfer
Learning**

Steps in a full machine learning project



2.Data



“What kind of data do we have?”

Different types of data

Rows

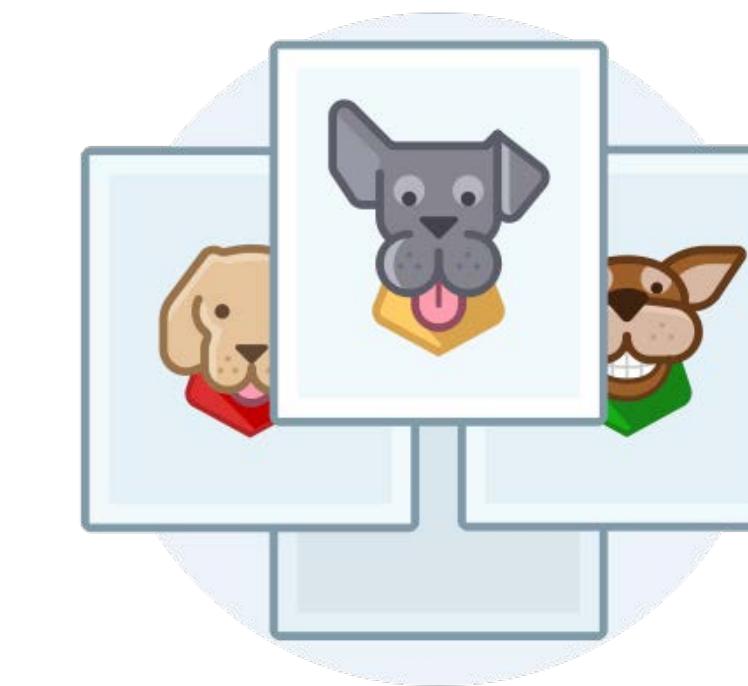
Columns

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
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

patient-records

Helvetica Regular 12 B I U

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



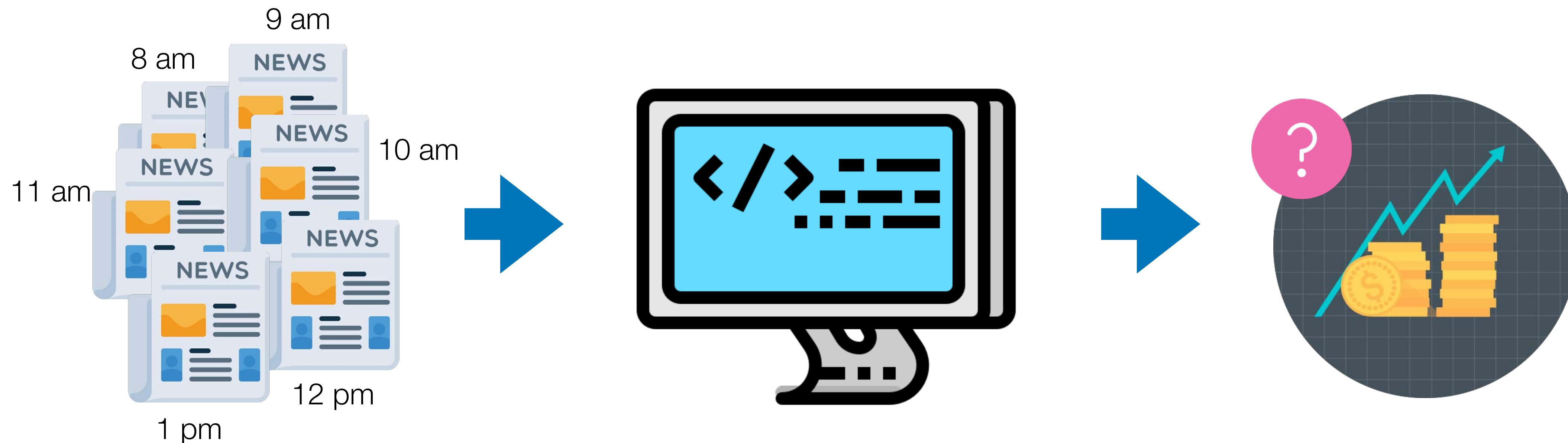
ID	Weight	Sex	Blood Pressure	Chest pain	Heart disease?
4326	110Kg	M	120/80	4	Yes
5681	64Kg	F	130/90	1	No
7911	81Kg	M	130/80	0	No

Table 1.0: Patient records

Static

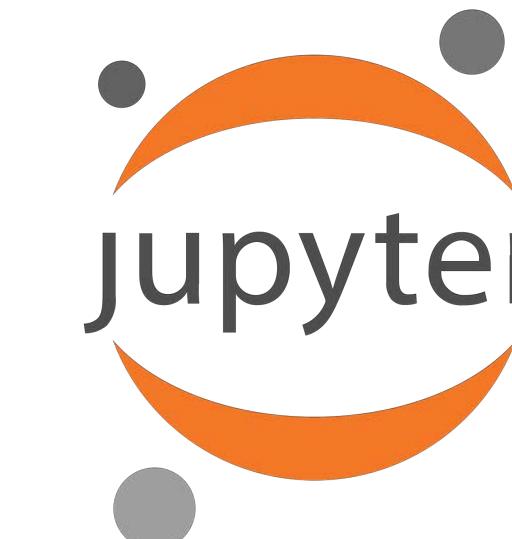
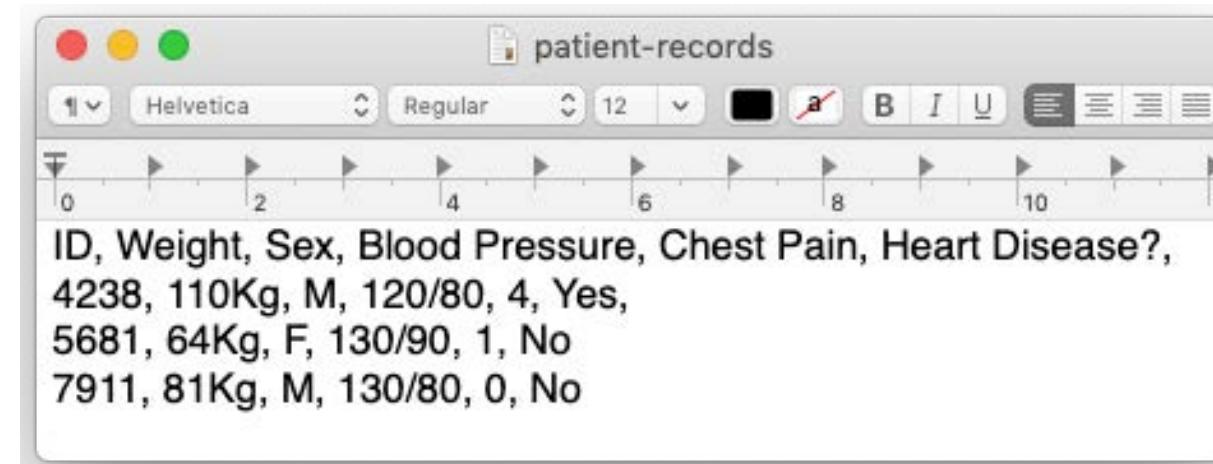
“The more data the better.”

Different types of data



Streaming

A data science workflow



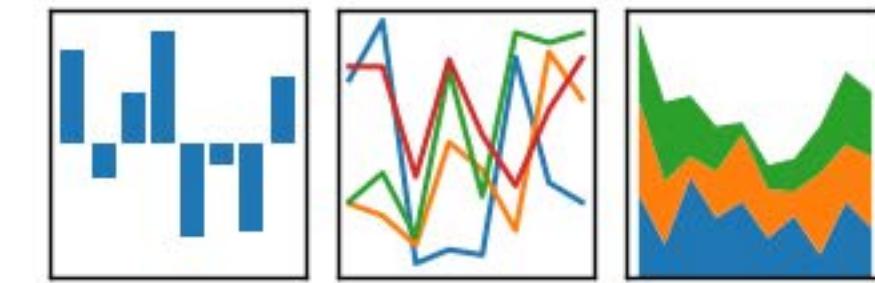
ID	Weight	Sex	Blood Pressure	Chest pain	Heart disease?
4326	110Kg	M	120/80	4	Yes
5681	64Kg	F	130/90	1	No
7911	81Kg	M	130/80	0	No

Table 1.0: Patient records

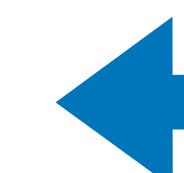
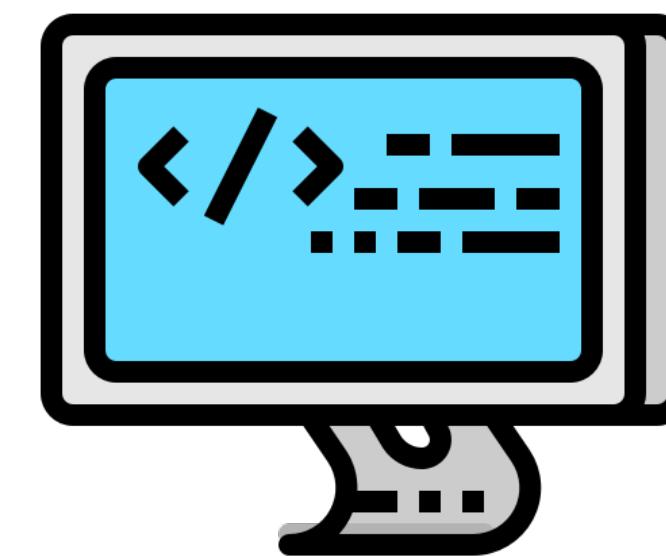
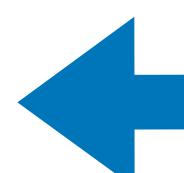
Static data

pandas

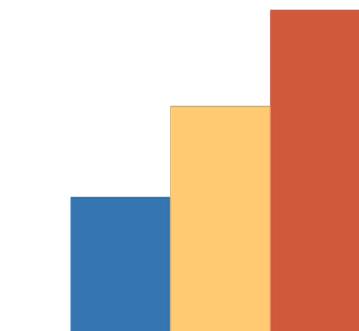
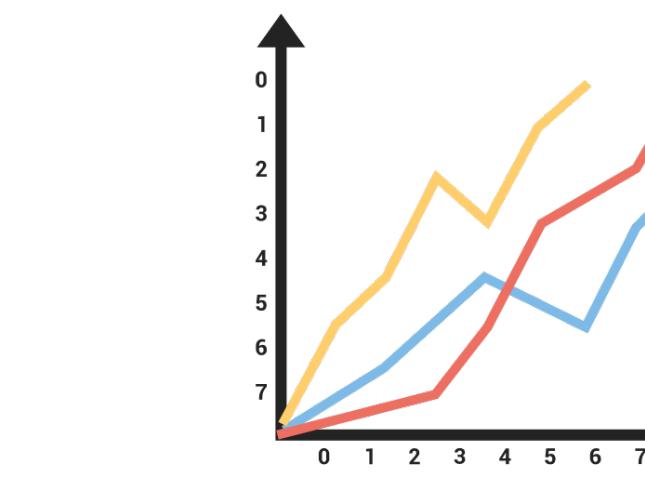
$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Data Analysis



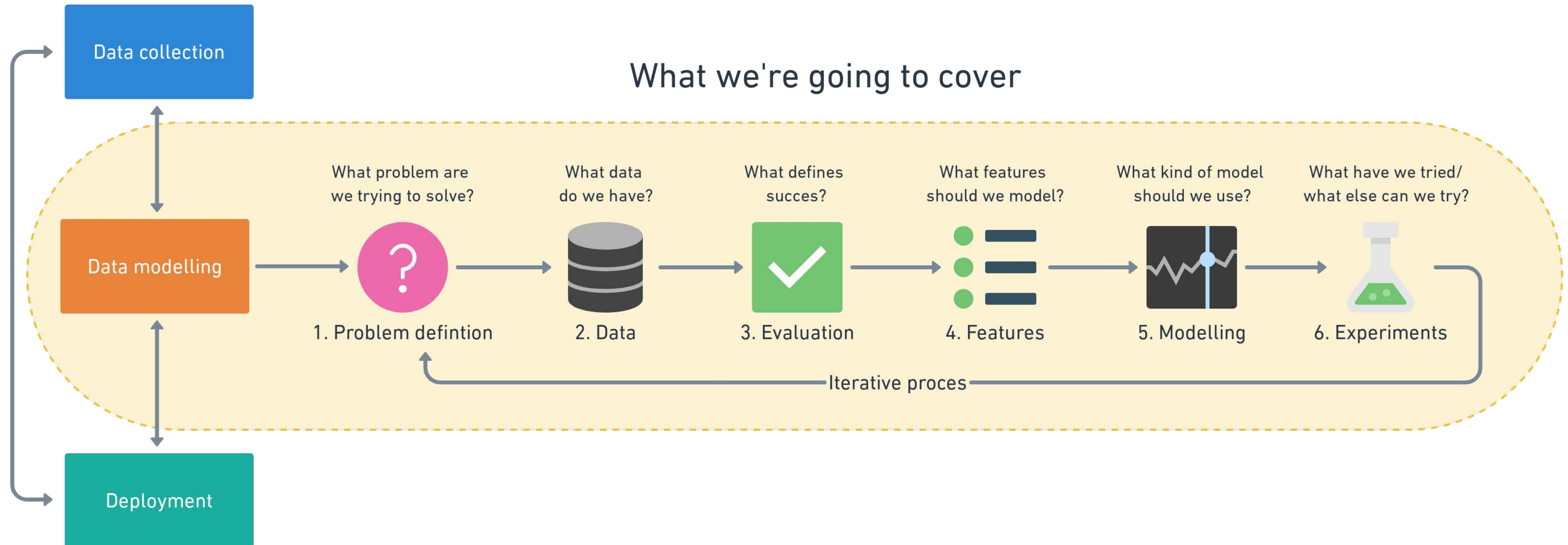
Machine learning model



matplotlib

What kind of data do you use?

Steps in a full machine learning project



3. Evaluation

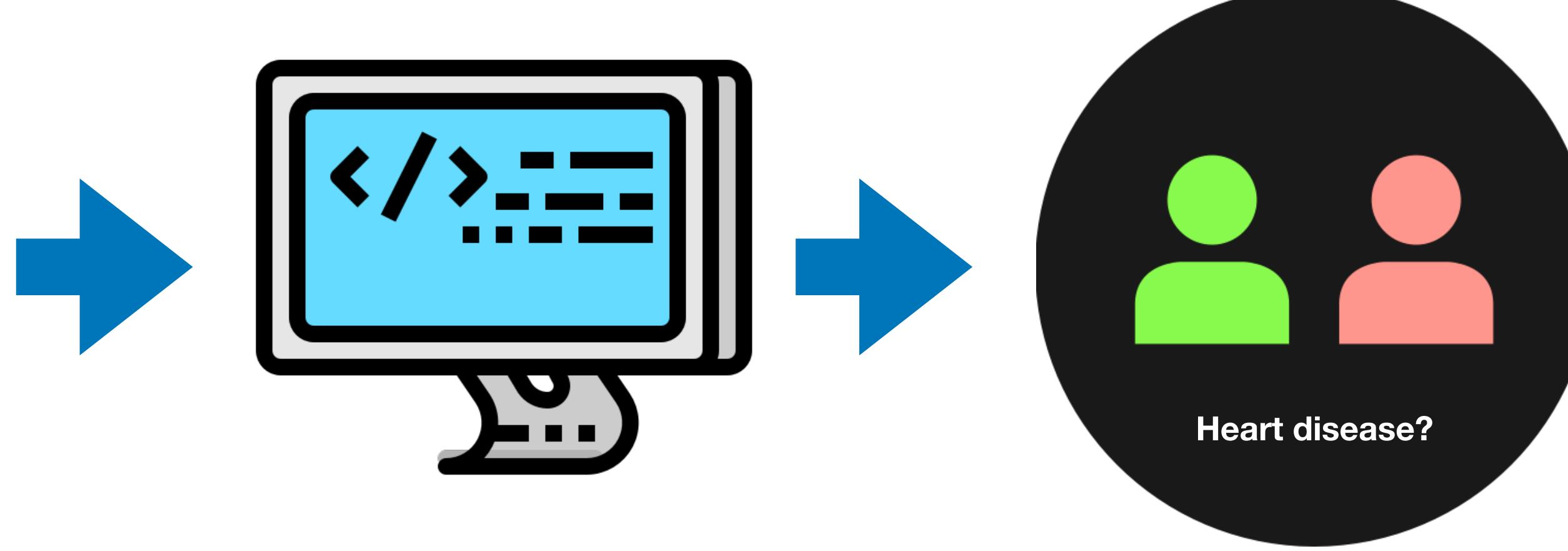


“What defines success for us?”

**“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

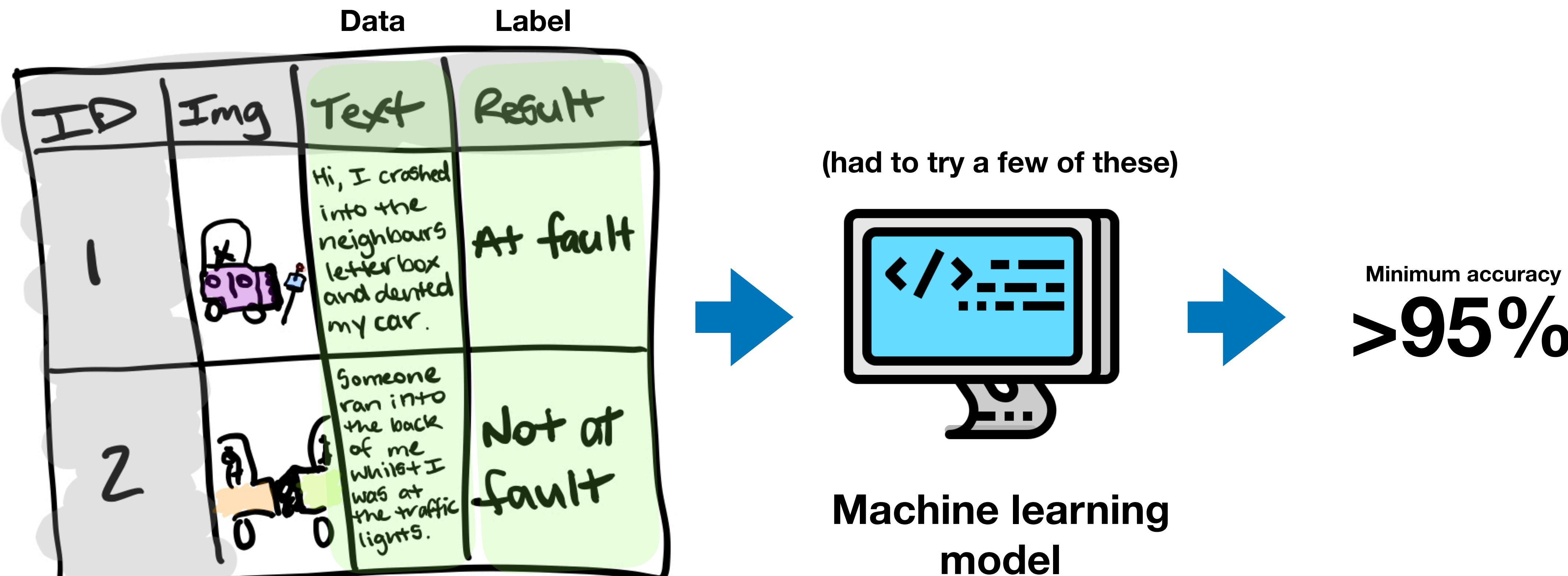
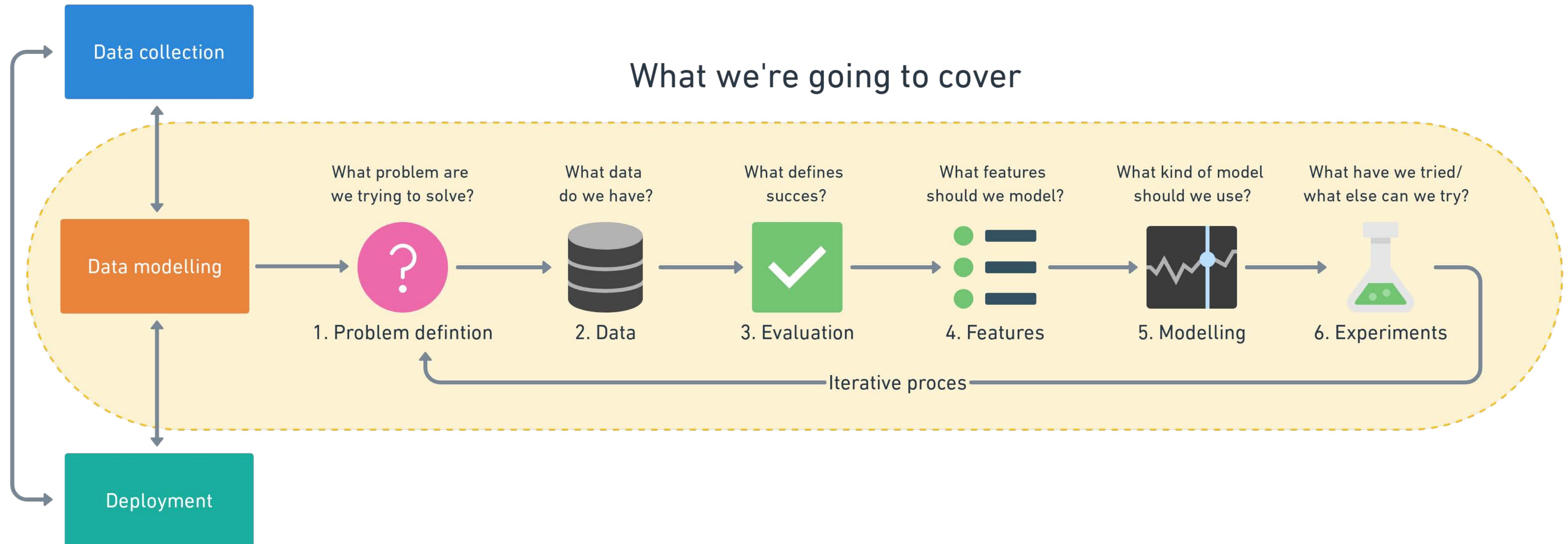


Table 2.0: Car insurance claims

What do you measure?

Steps in a full machine learning project



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

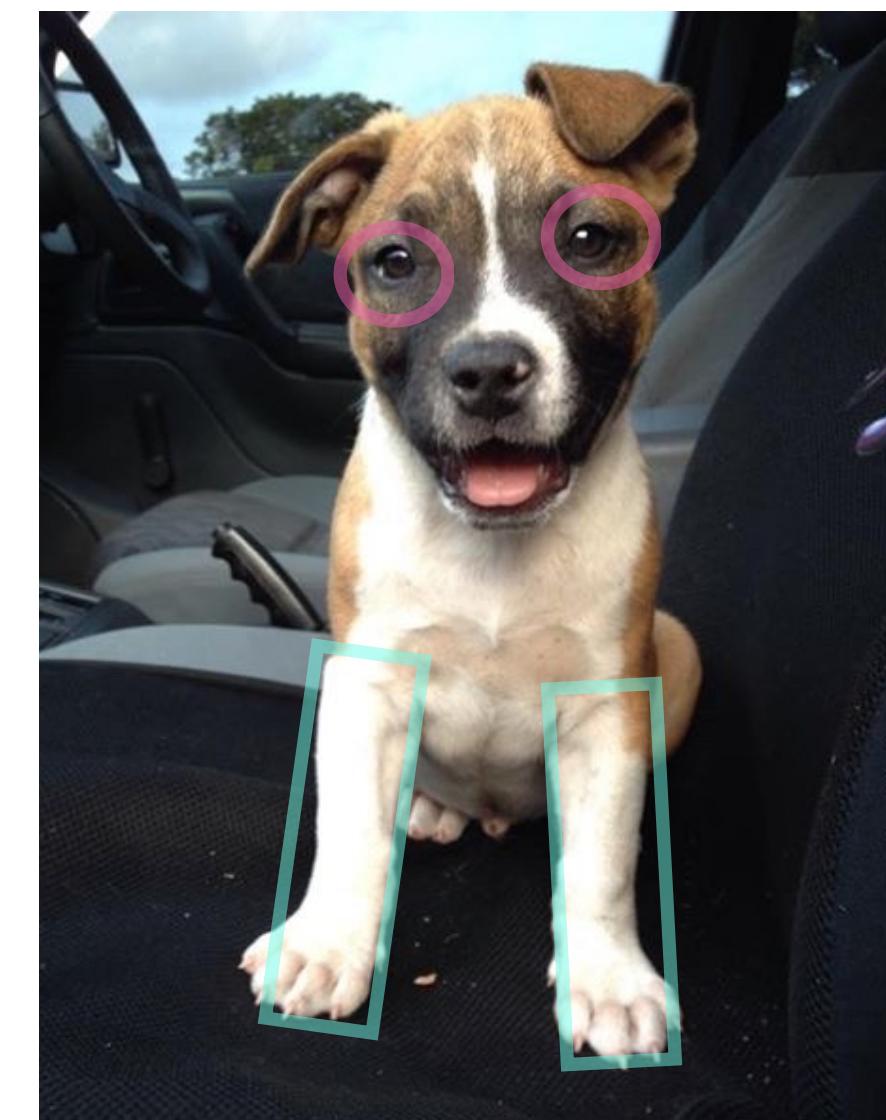
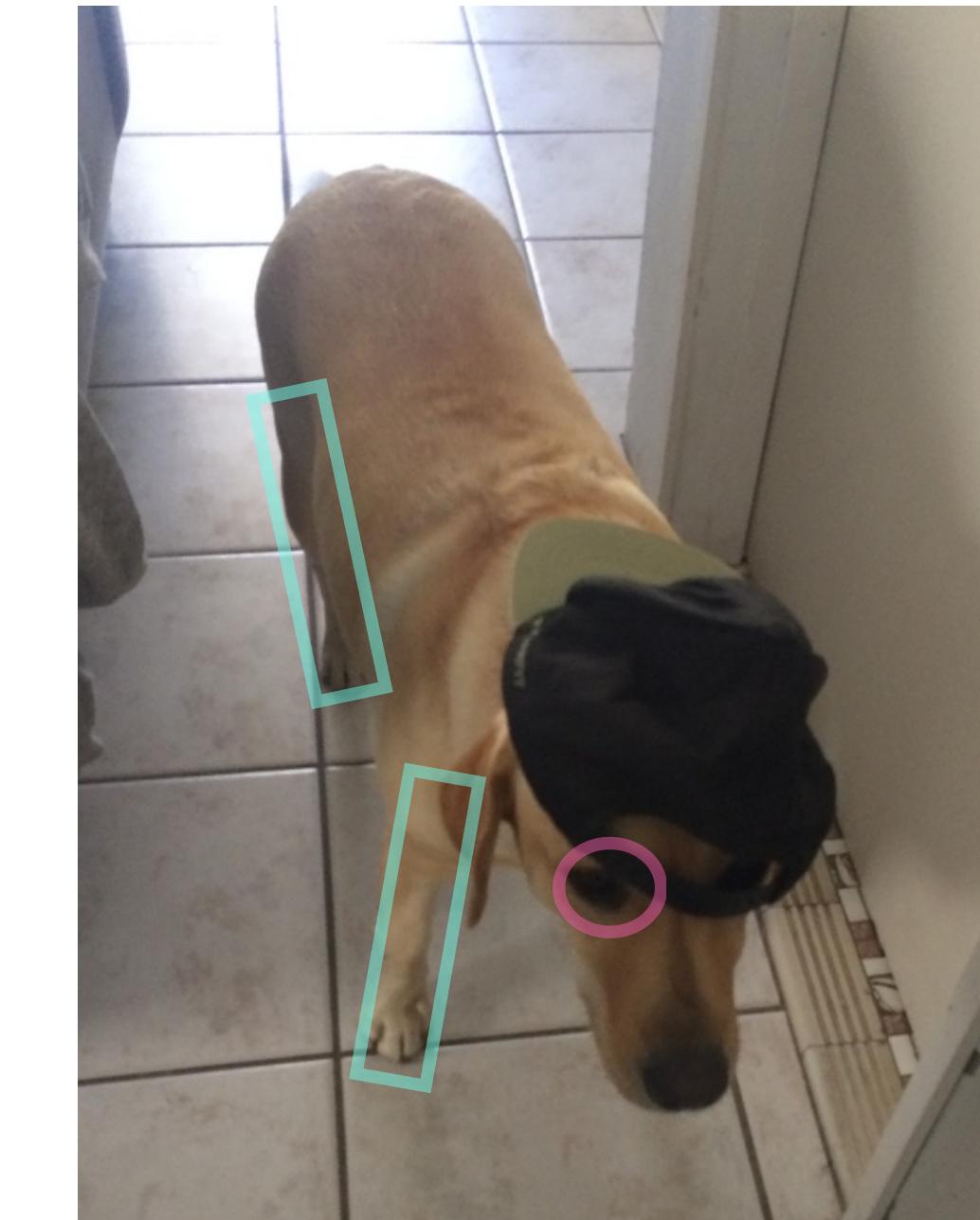
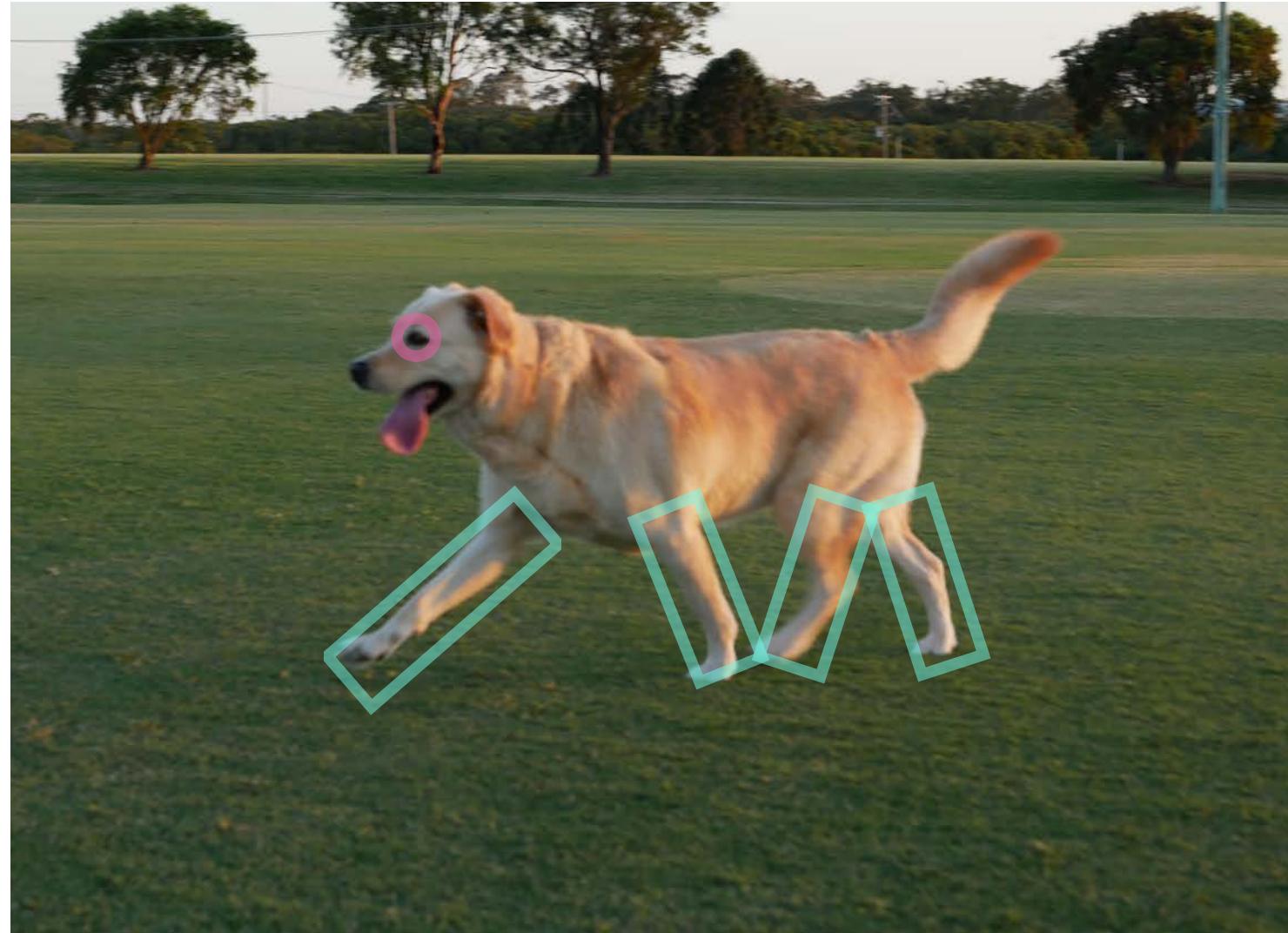
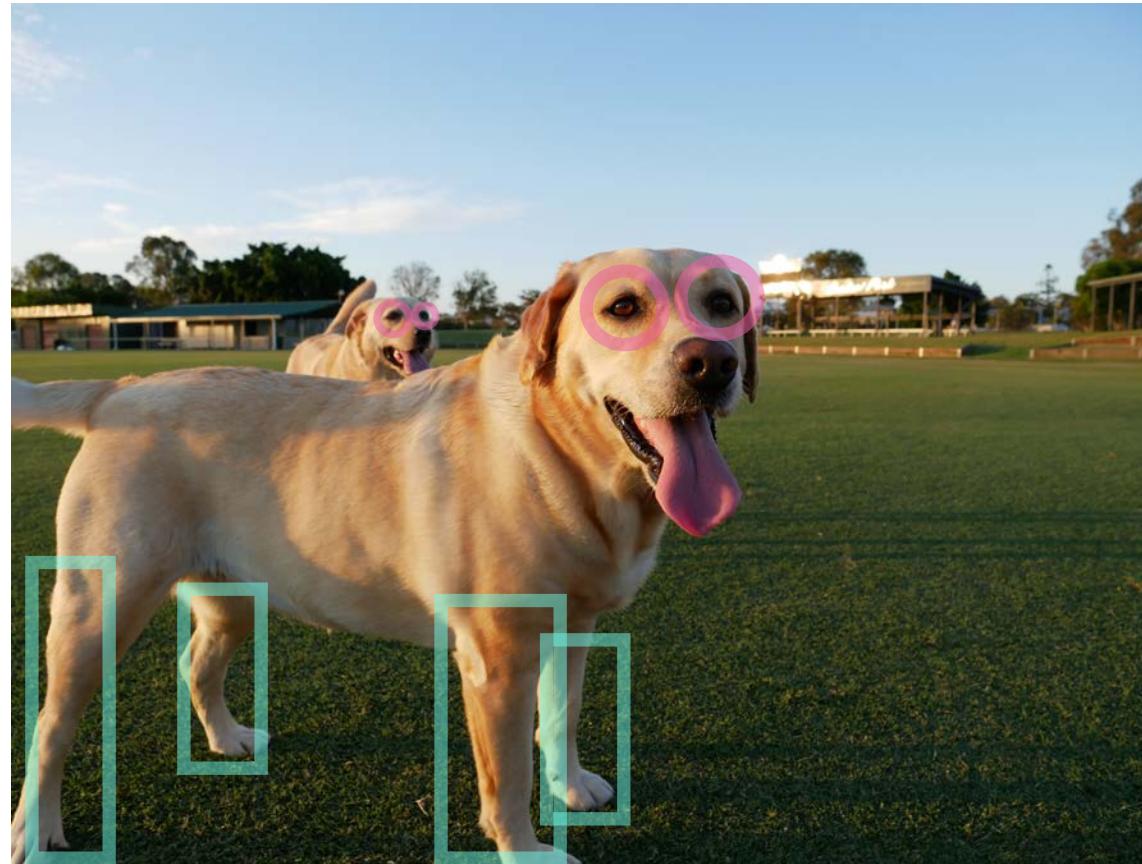
Numerical features

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

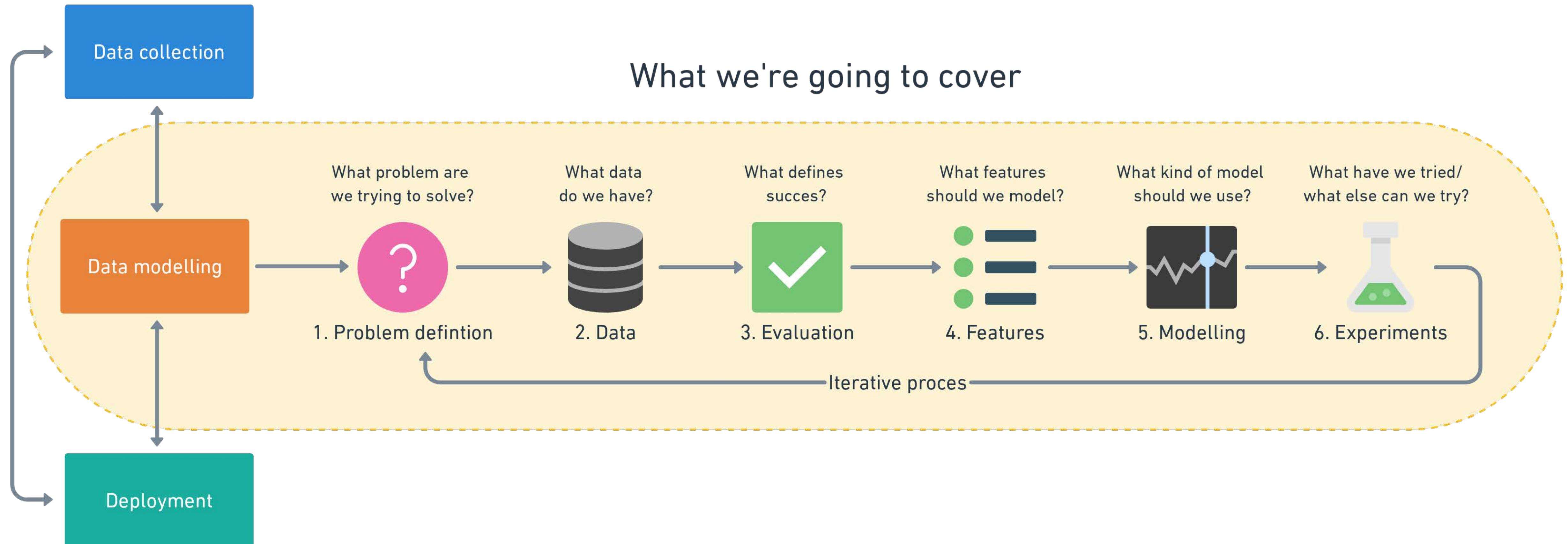
Want > 10% coverage

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



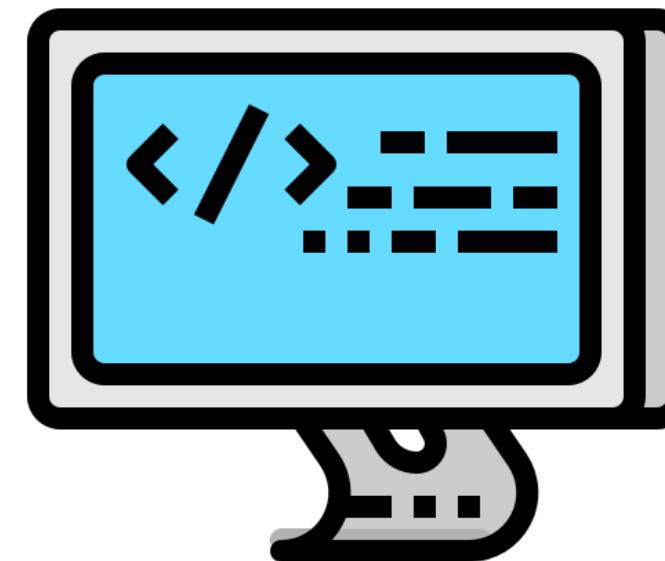
5. Modelling Part 1 – 3 sets



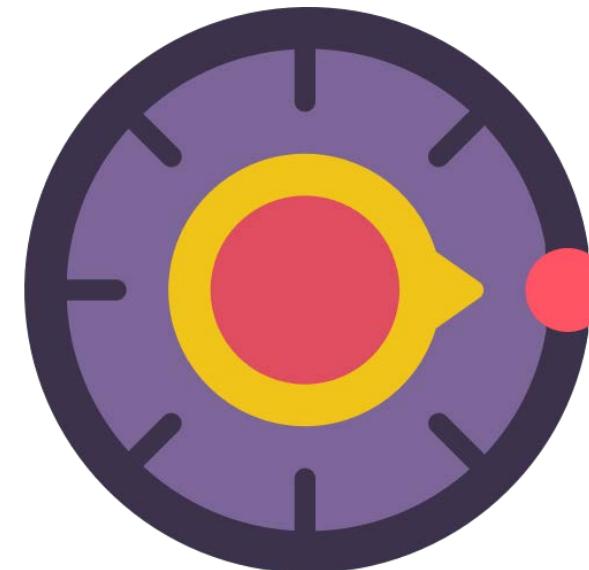
“Based on our problem and data, what model should we use?”

3 parts to modelling

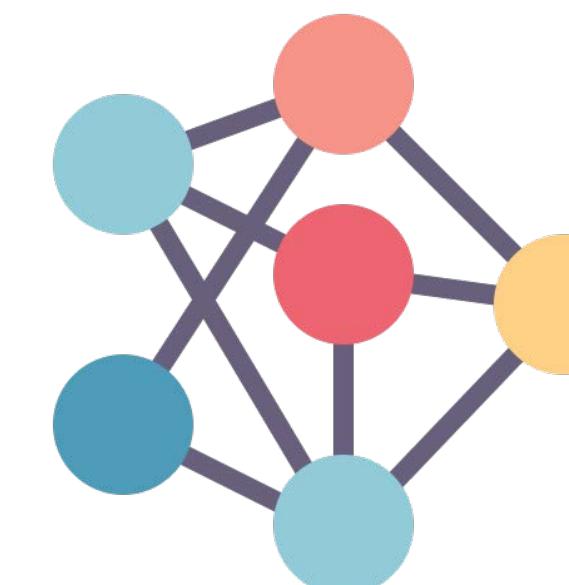
1. Choosing and training a model



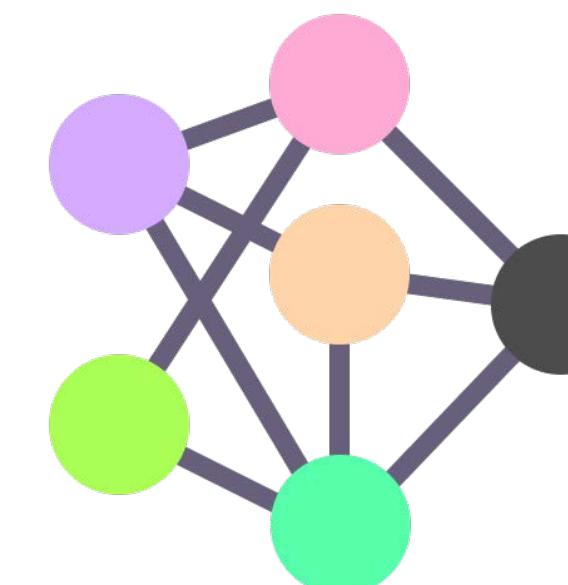
2. Tuning a model



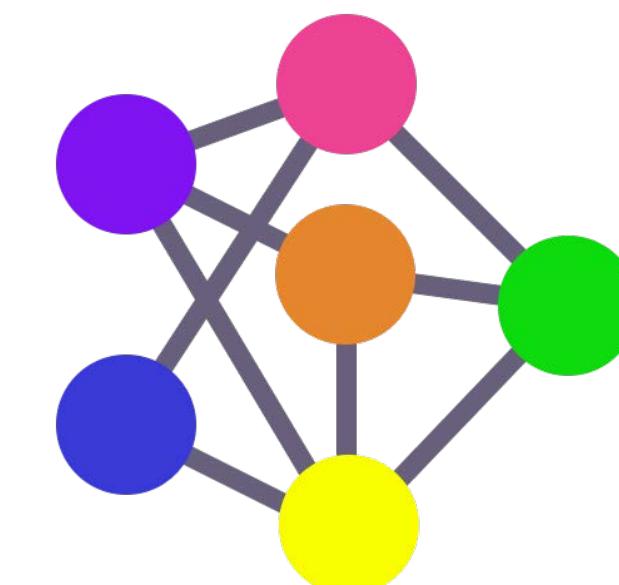
3. Model comparison



vs.

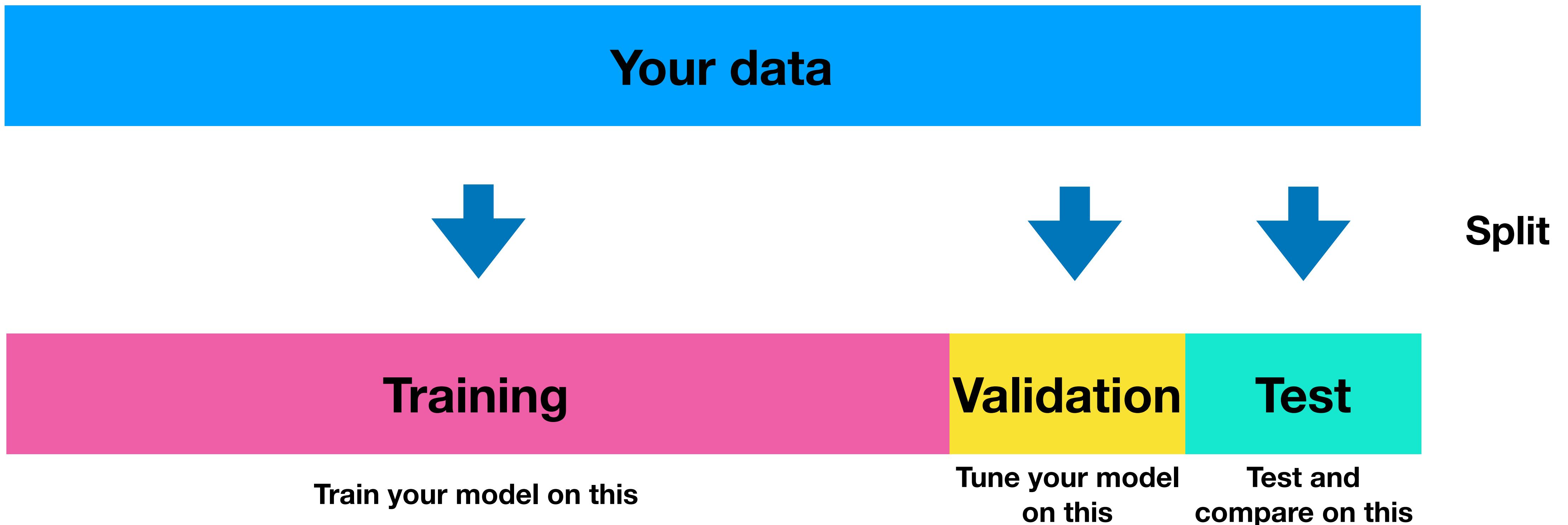


vs.



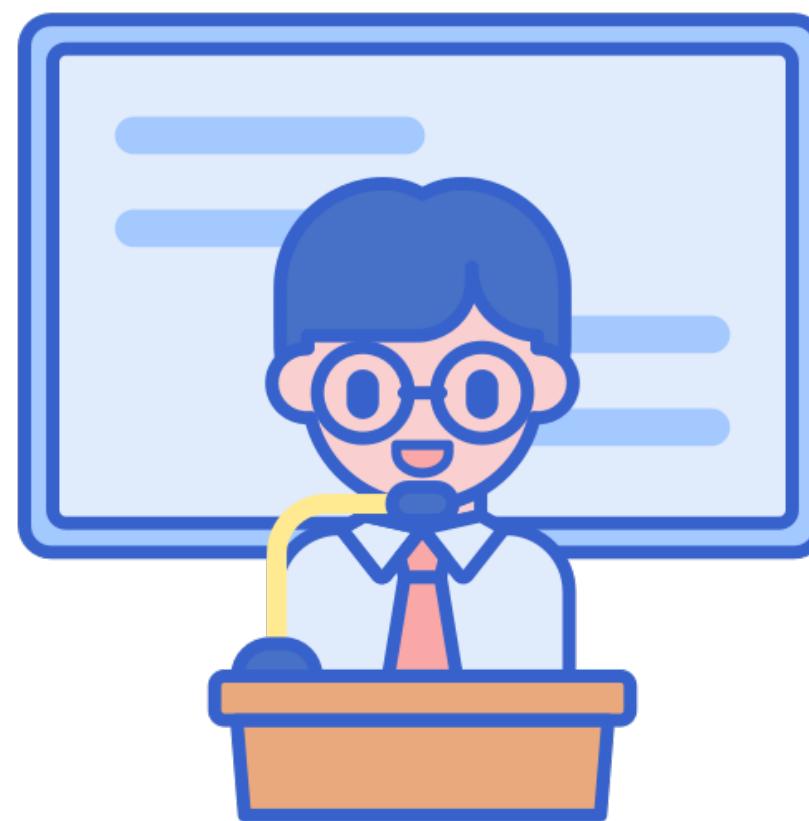
The most important concept in machine learning

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

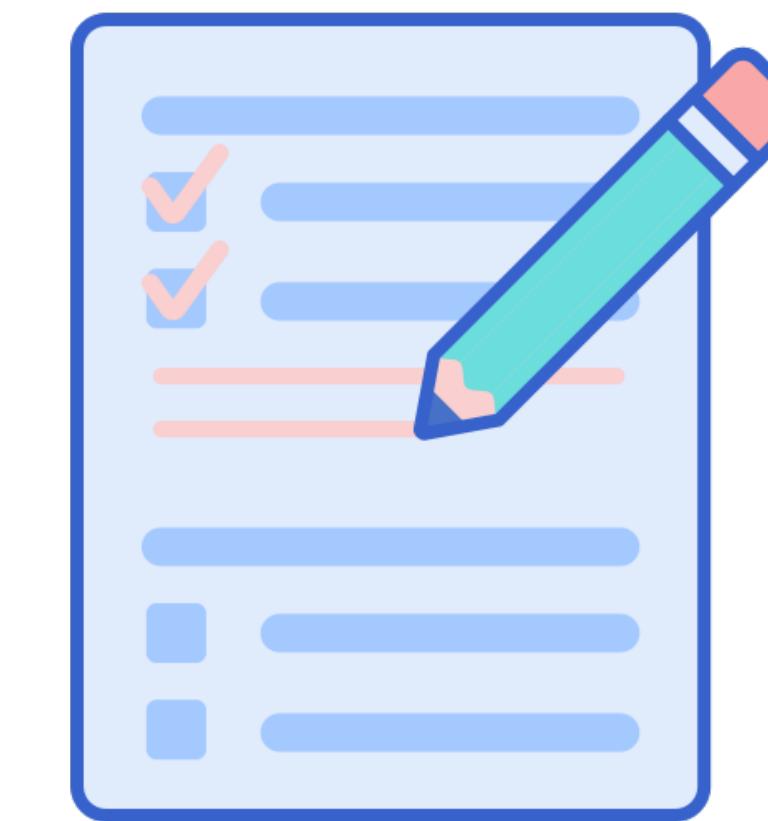
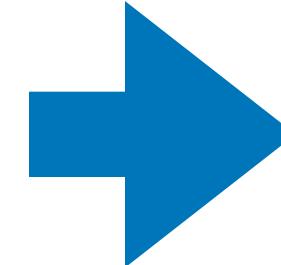


The most important concept in machine learning

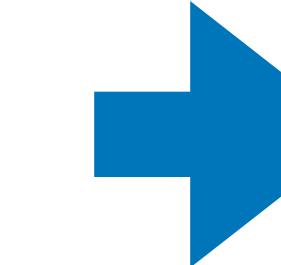
(the 3 sets)



Course materials
(training set)



Practice exam
(validation set)

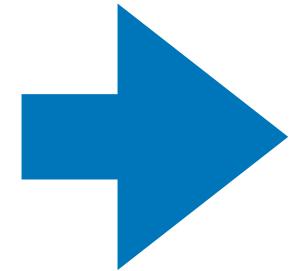
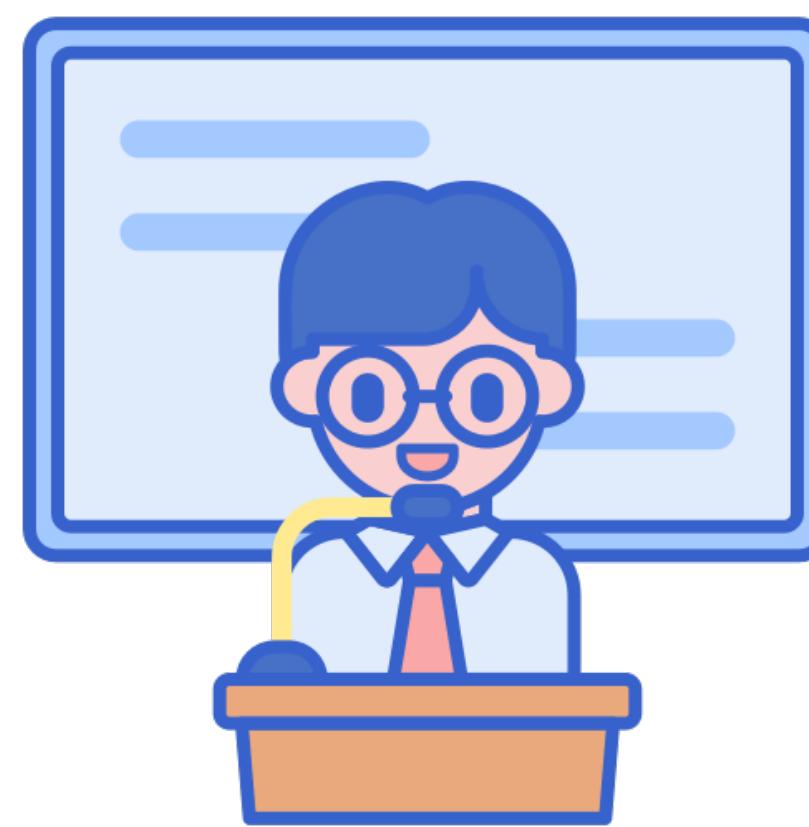


Final exam
(test set)

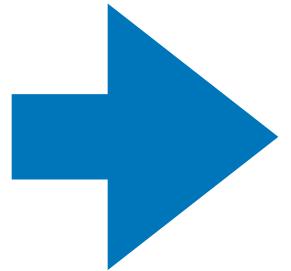
Generalization

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

When things go wrong

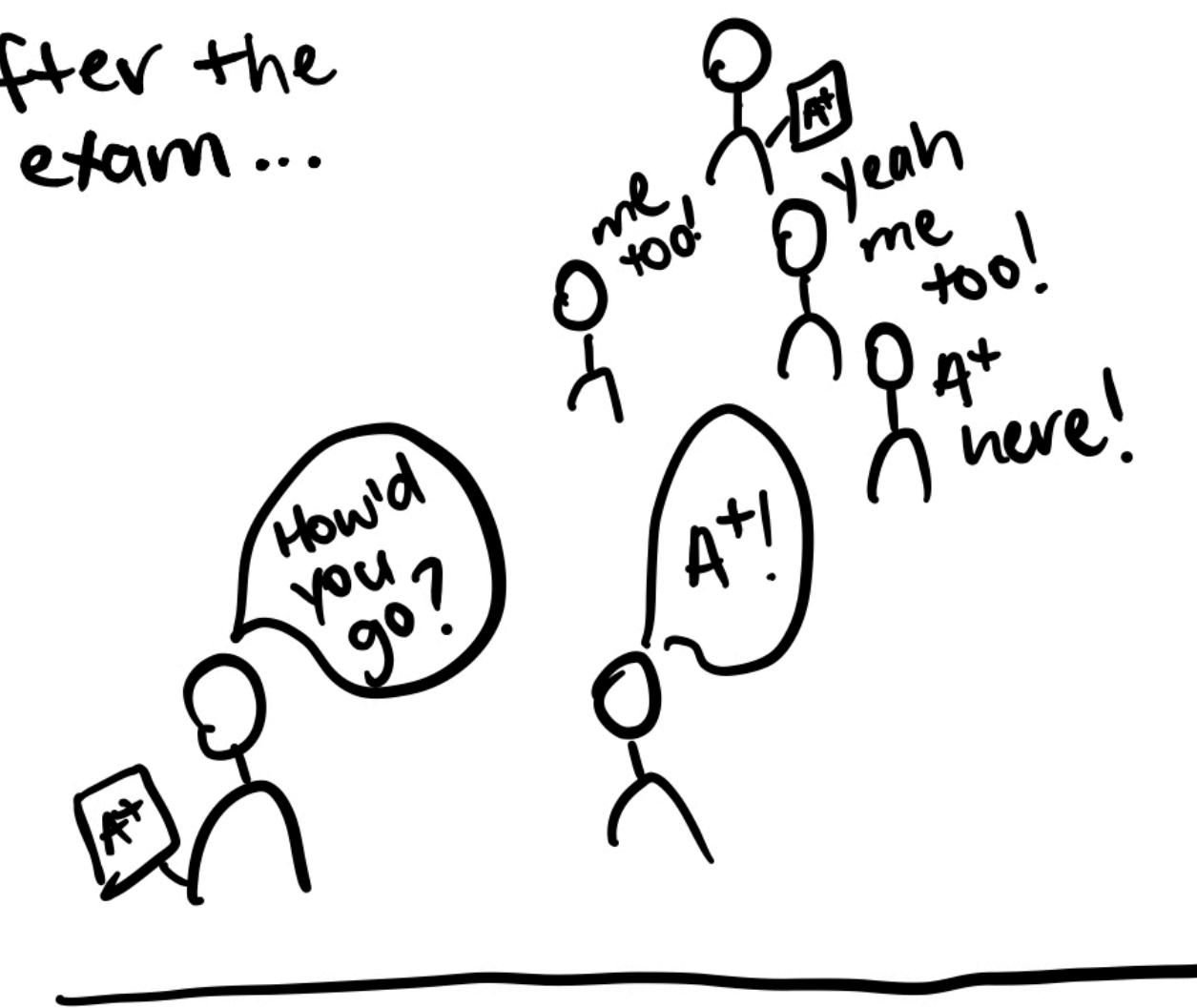


Practice exam
(same as final exam set)



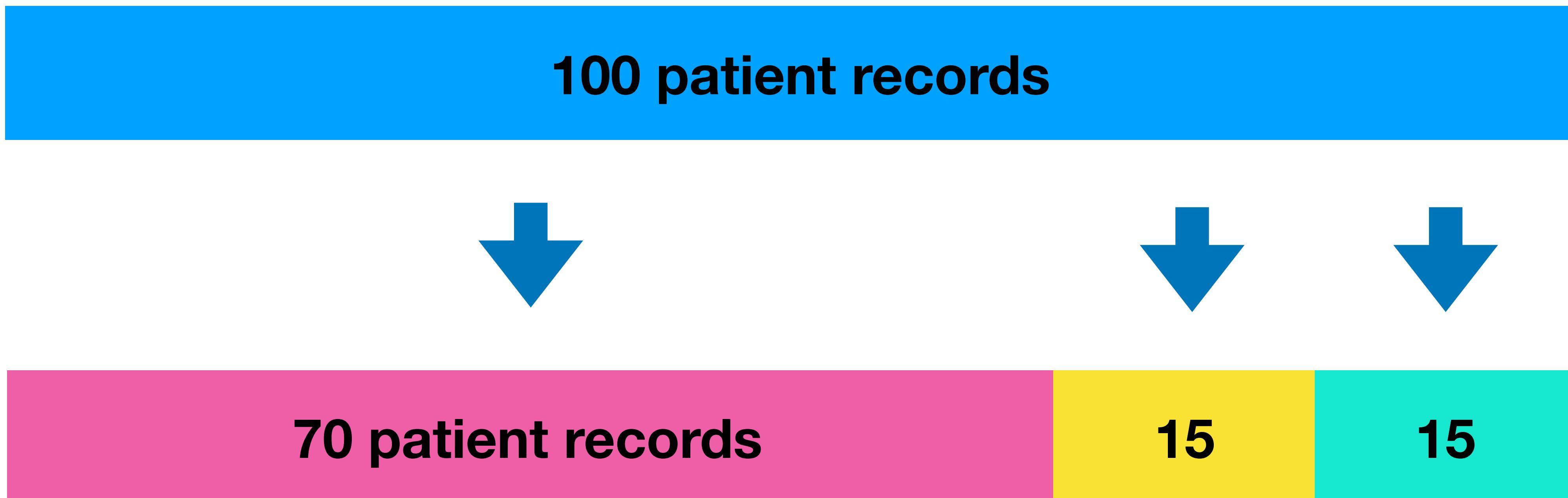
Final exam
(already seen it)

After the
exam...



The most important concept in machine learning

(the 3 sets)



Training split (70-80%)

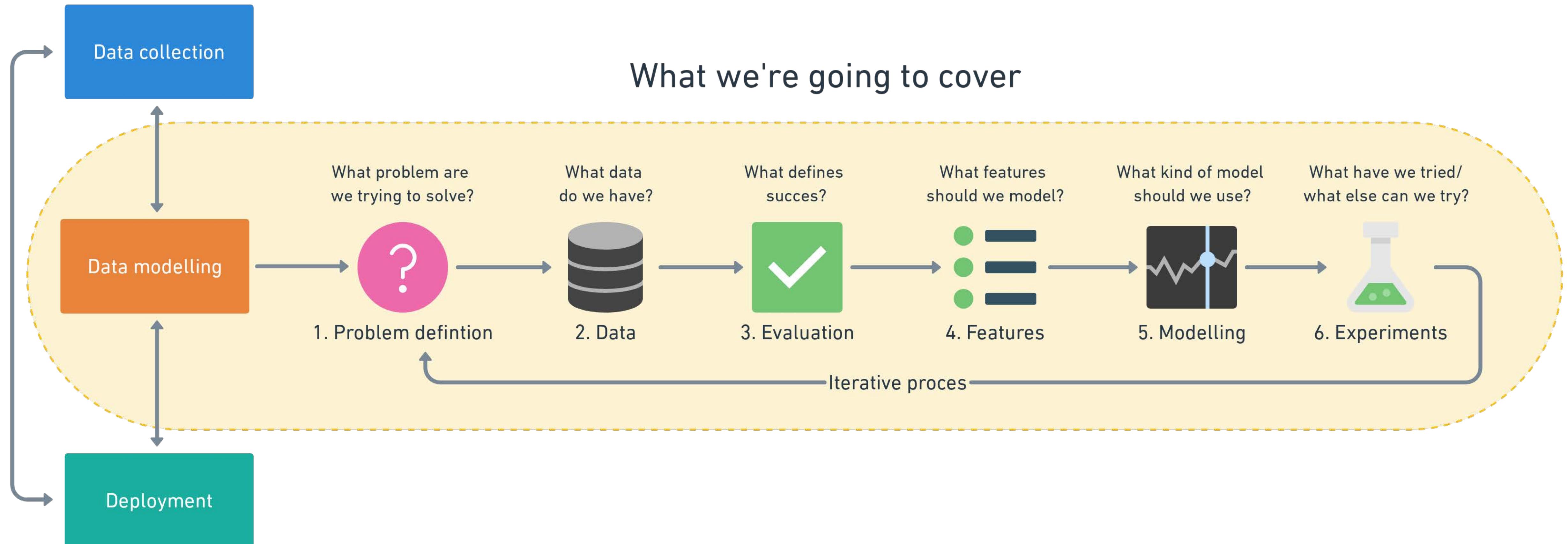
Validation split
(10-15%)

Test split
(10-15%)

Split

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

Steps in a full machine learning project



5. Modelling Part 2 – Choosing

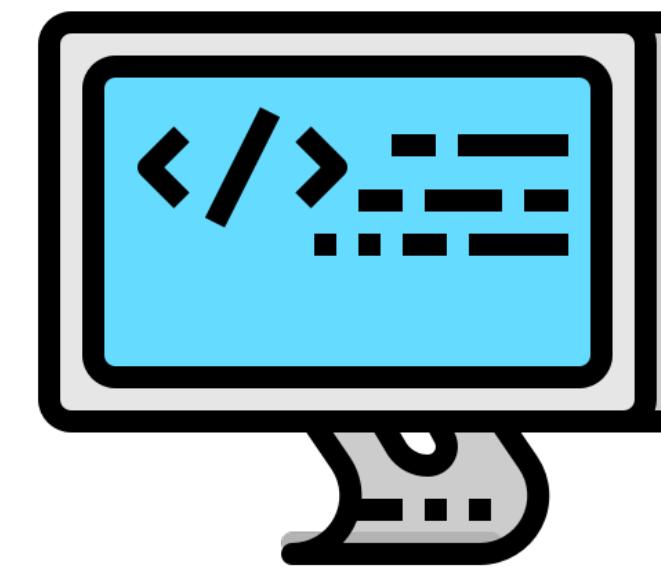


“Based on our problem and data, what model should we use?”

3 parts to modelling

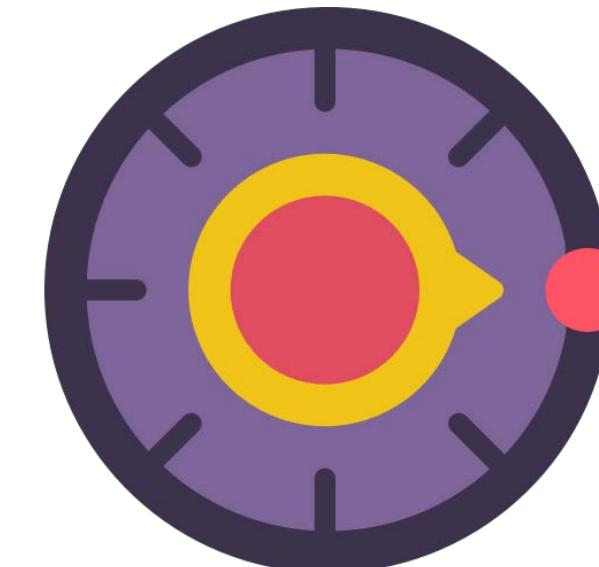
1. Choosing and training a model

Training Data



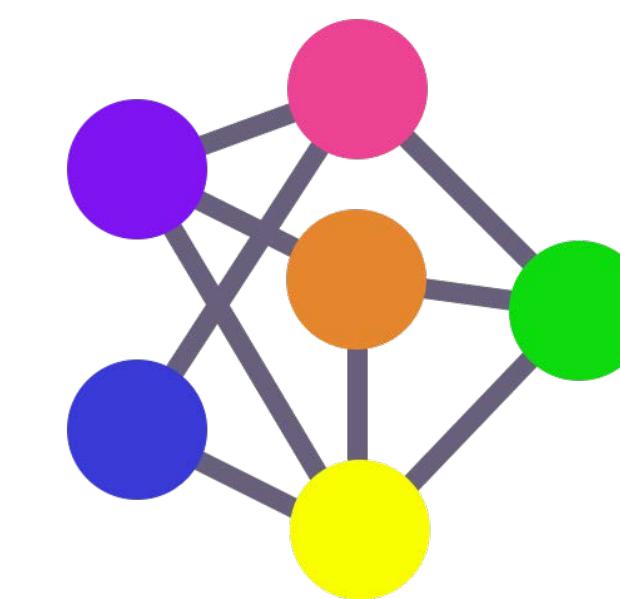
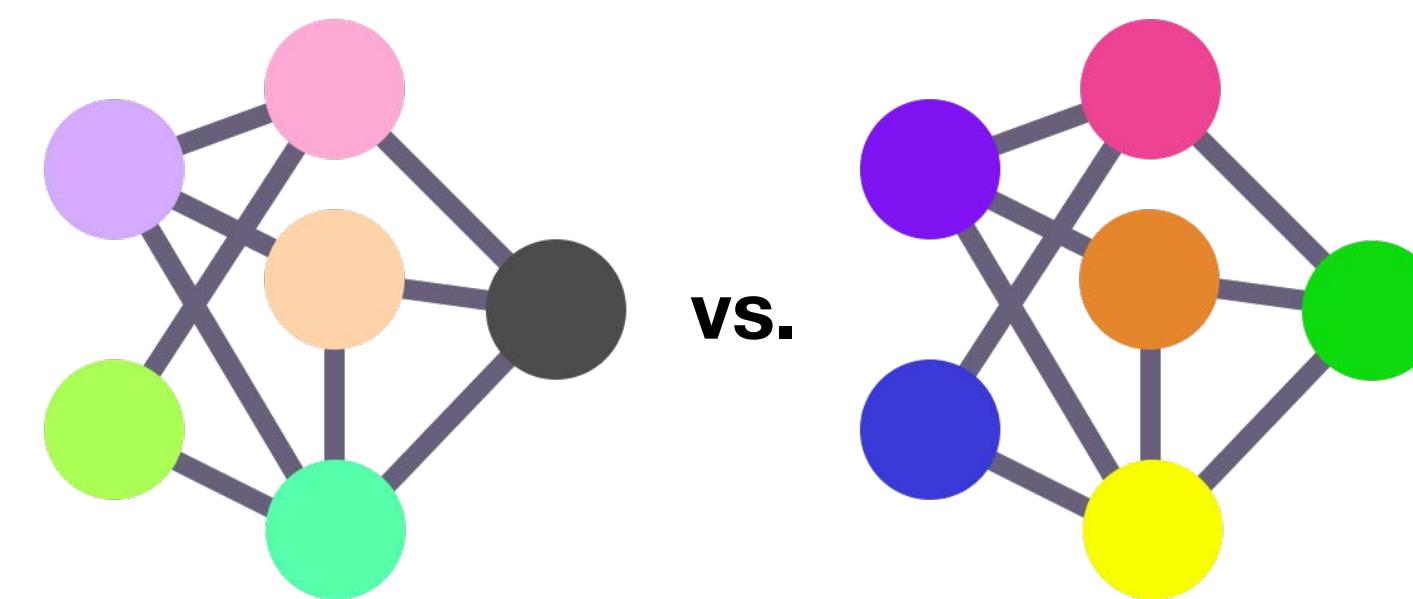
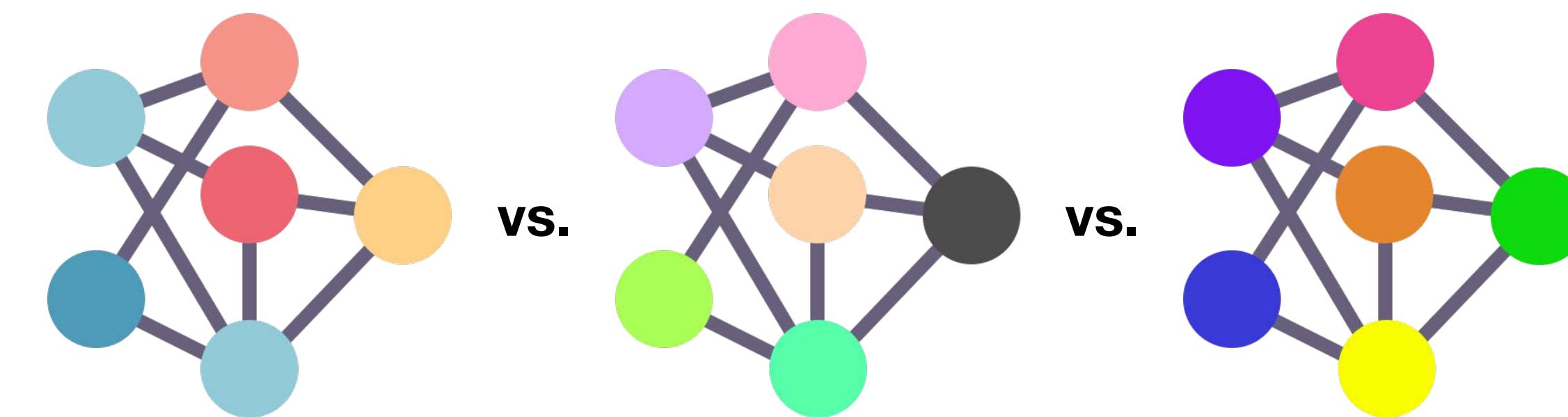
2. Tuning a model

Validation Data

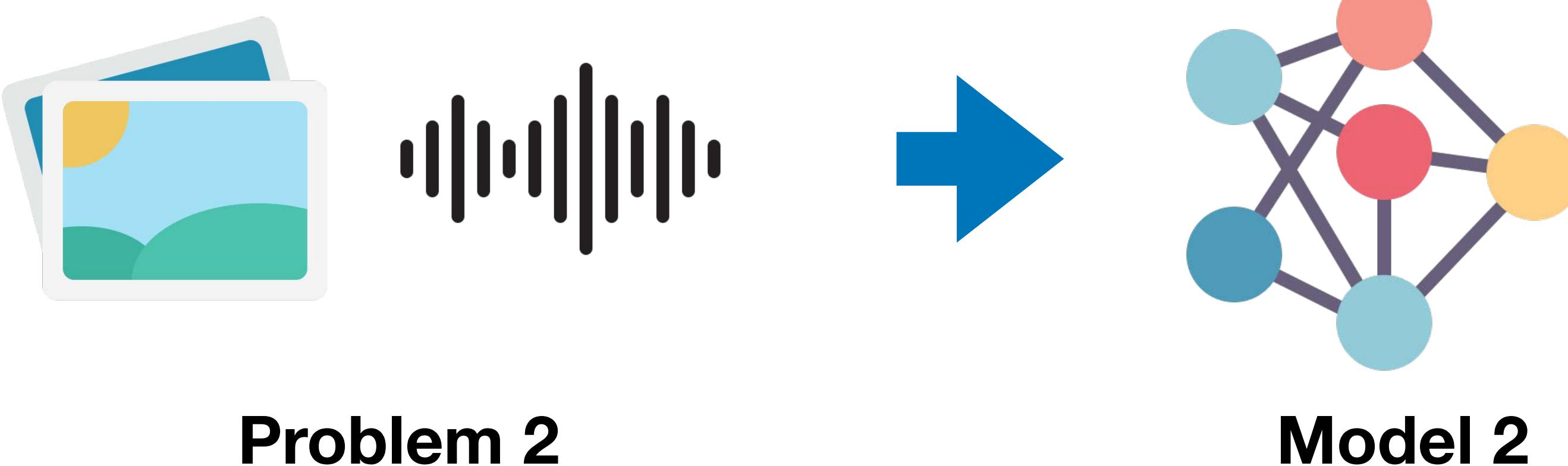


3. Model comparison

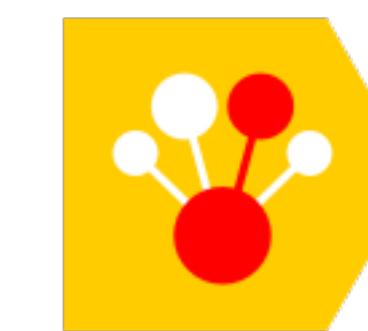
Test Data



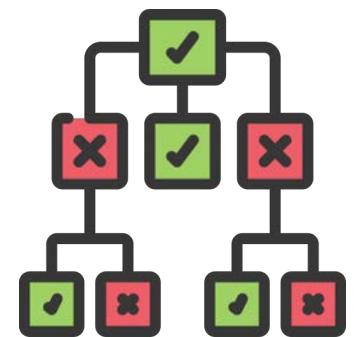
Choosing a model



Structured Data



dmlc
XGBoost

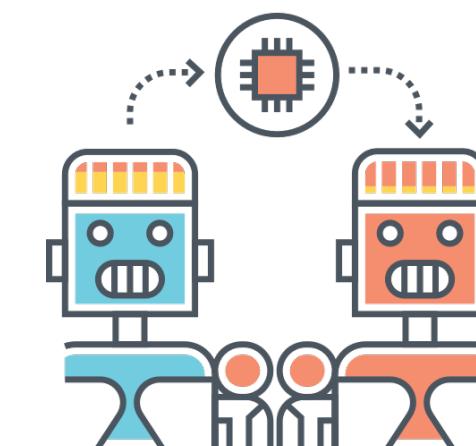


Random Forest

Unstructured Data

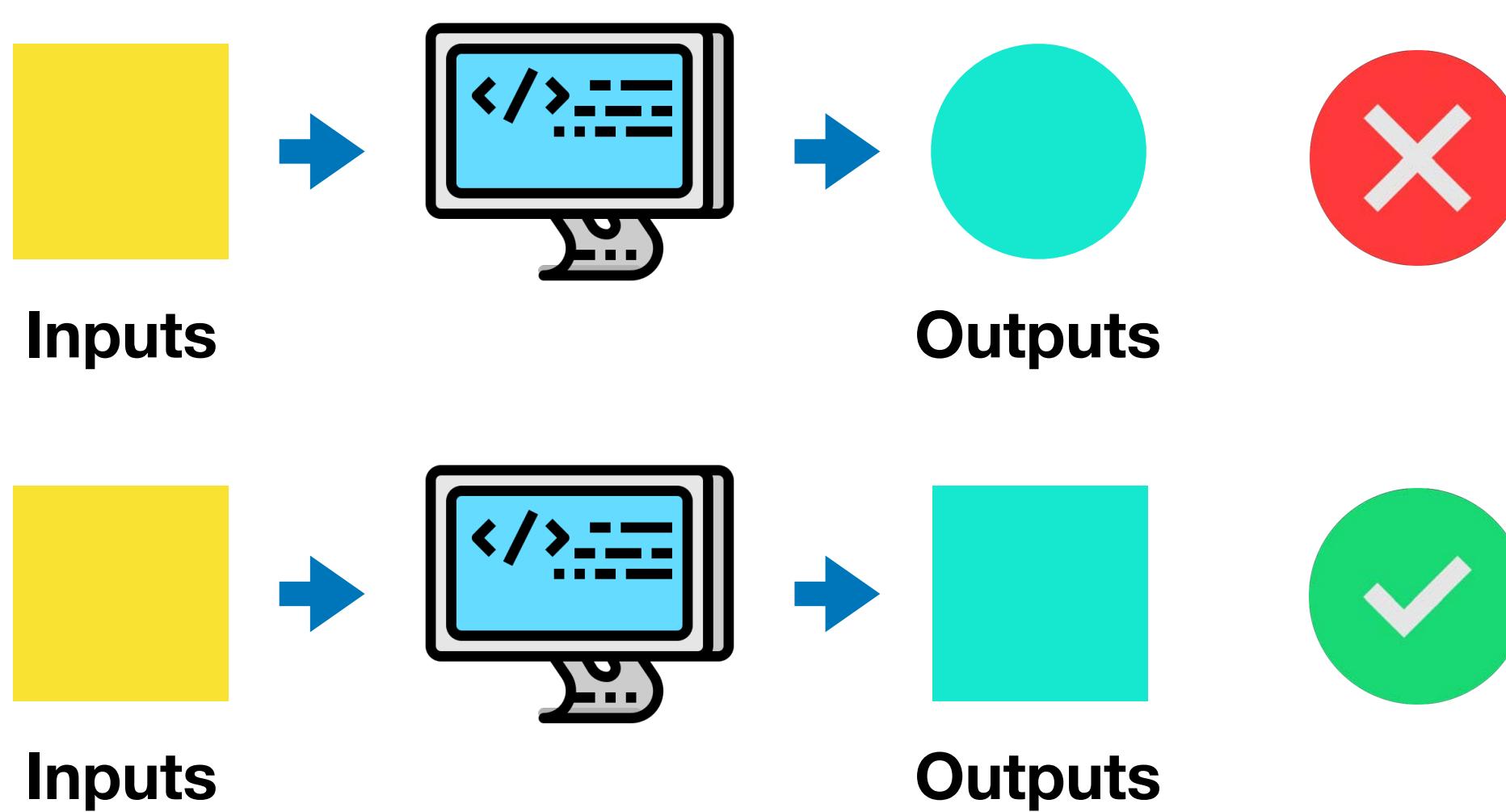


Deep Learning



Transfer Learning

Training a model



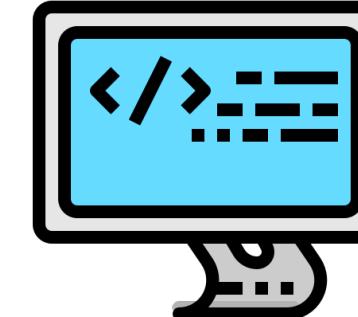
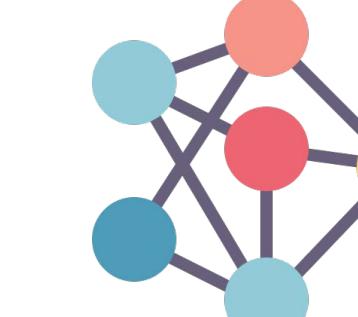
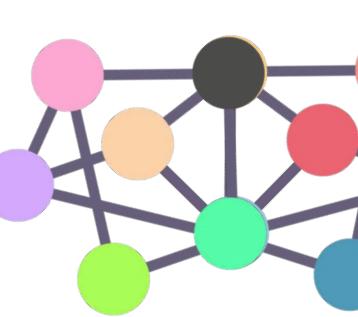
ID	X (data)	y (label)			
	weight	Sex	Heart Rate	Chest pain	Heart disease?
4326	110Kg	M	81	4	Yes
5681	64Kg	F	61	1	No
7911	81Kg	M	57	0	No

Table 1.0 : Patient records

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

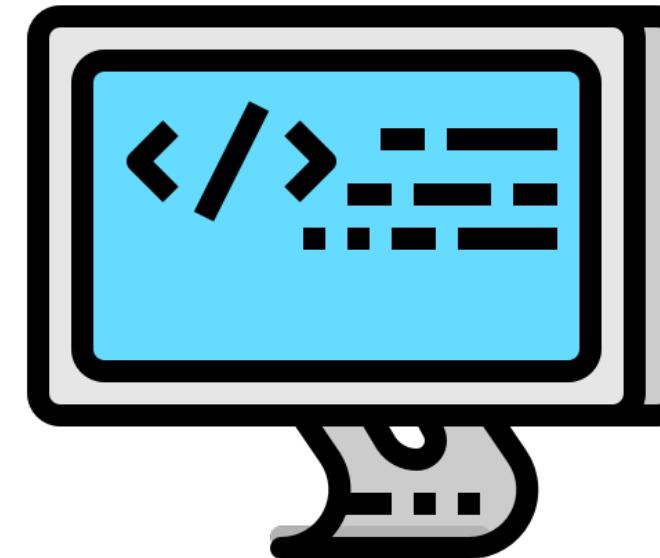
Things to remember

- 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

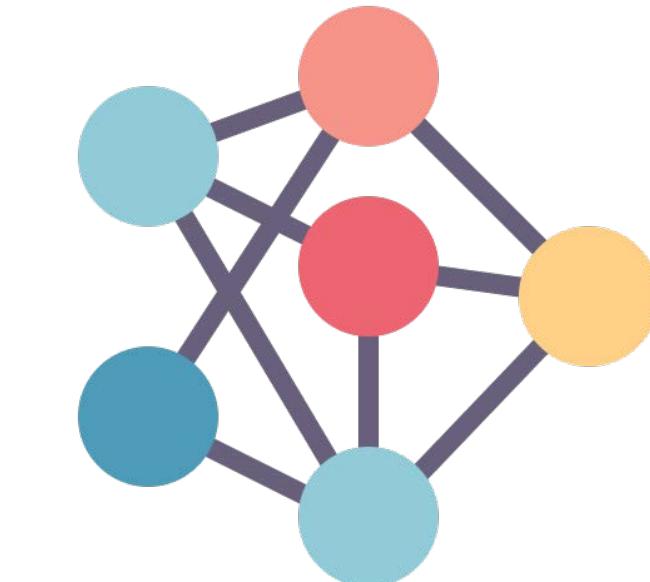
Up next

1. Choosing and training a model

Training Data



or



2. Tuning a model

Validation Data

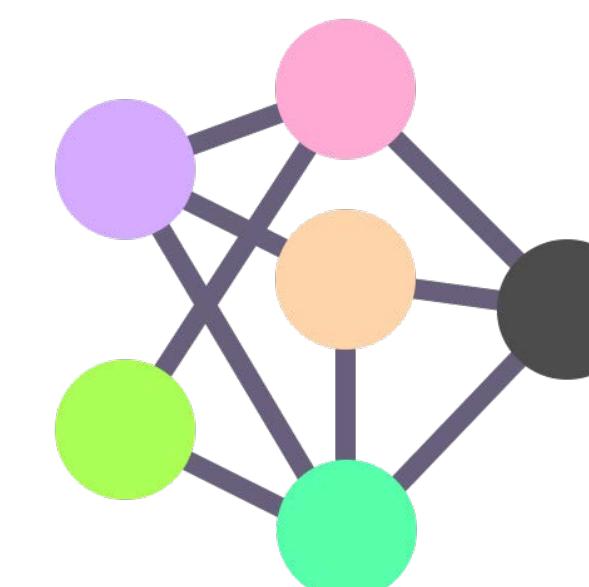


3. Model comparison

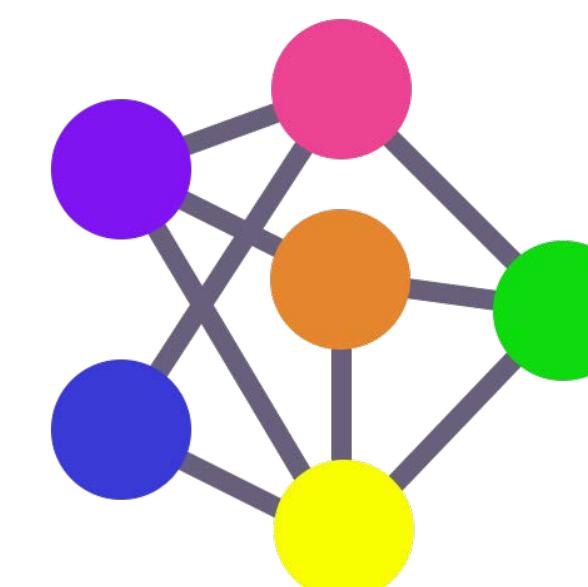
Test Data



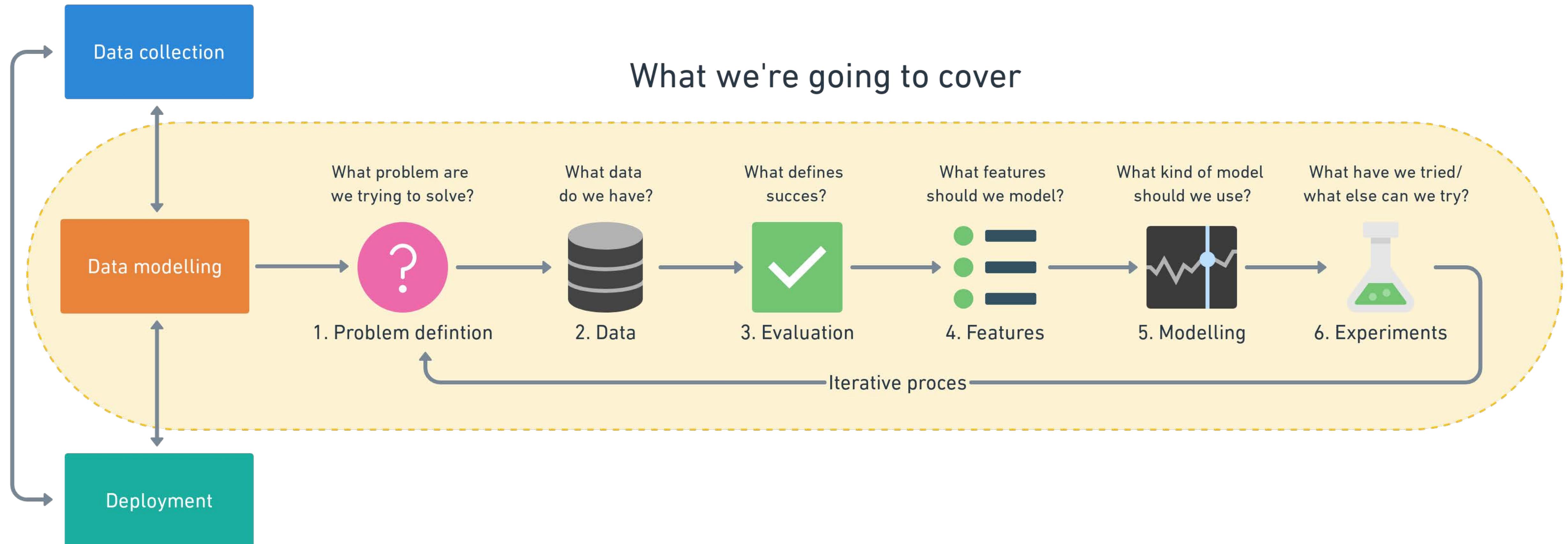
vs.



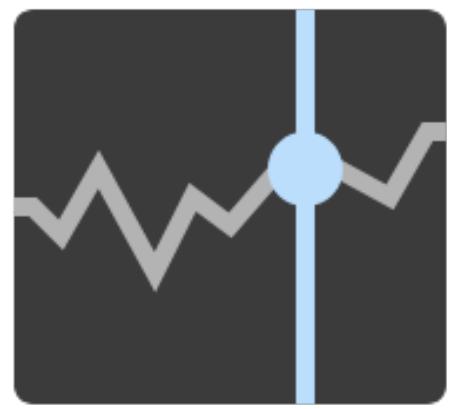
vs.



Steps in a full machine learning project



5. Modelling Part 3 – Tuning

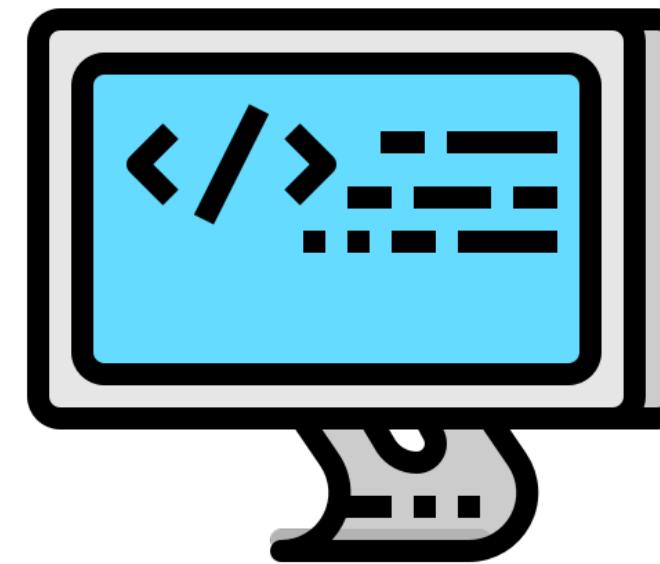


“Based on our problem and data, what model should we use?”

3 parts to modelling

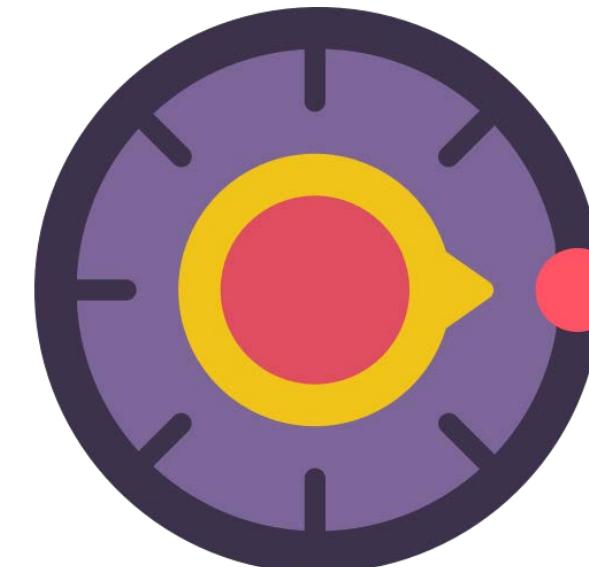
1. Choosing and training a model

Training Data



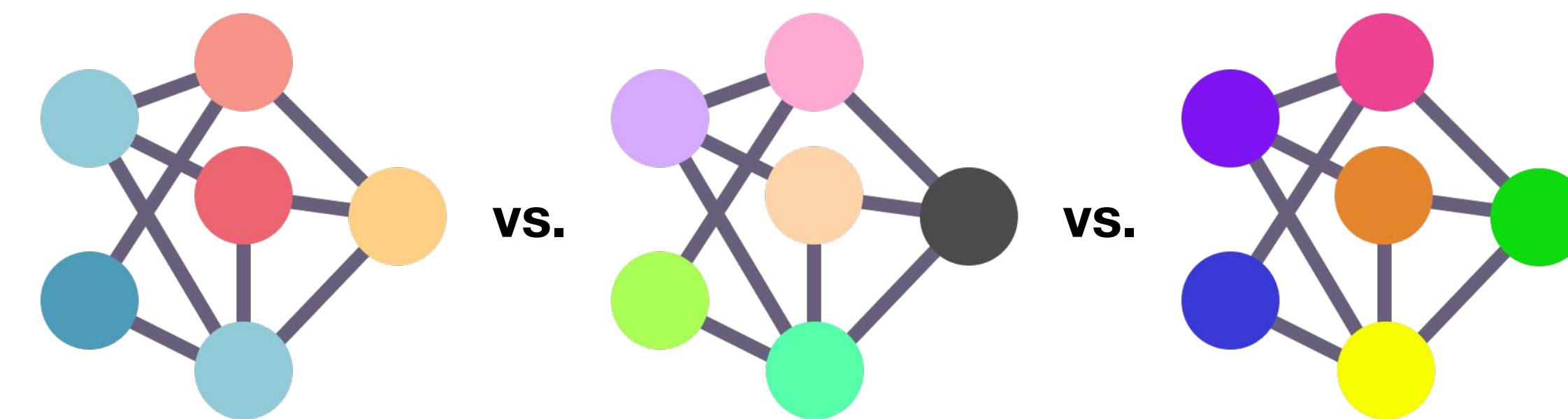
2. Tuning a model

Validation Data



3. Model comparison

Test Data



Tuning a model



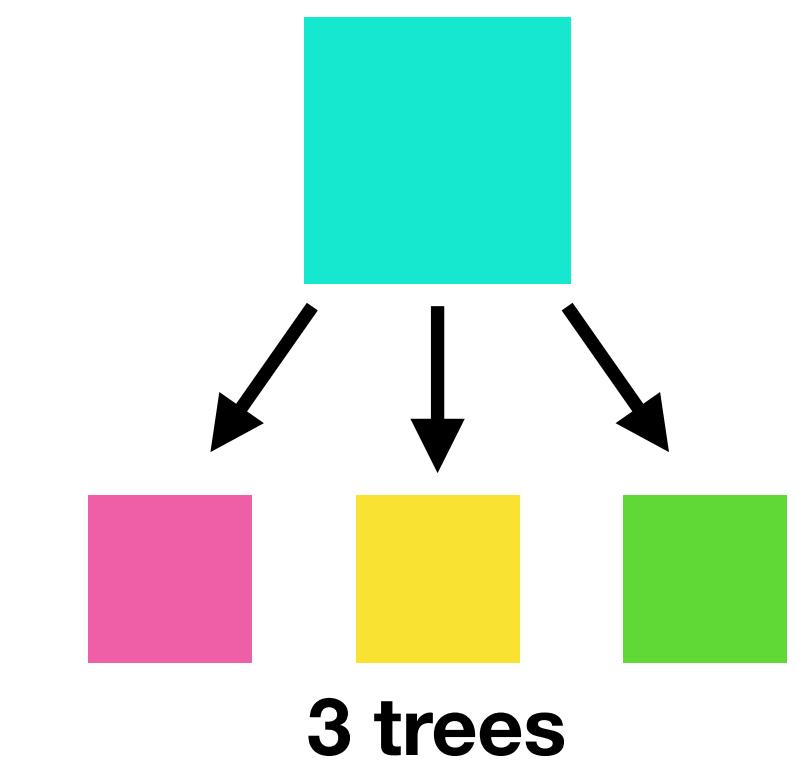
Cooking time: 1 hour
Temperature: 180°C



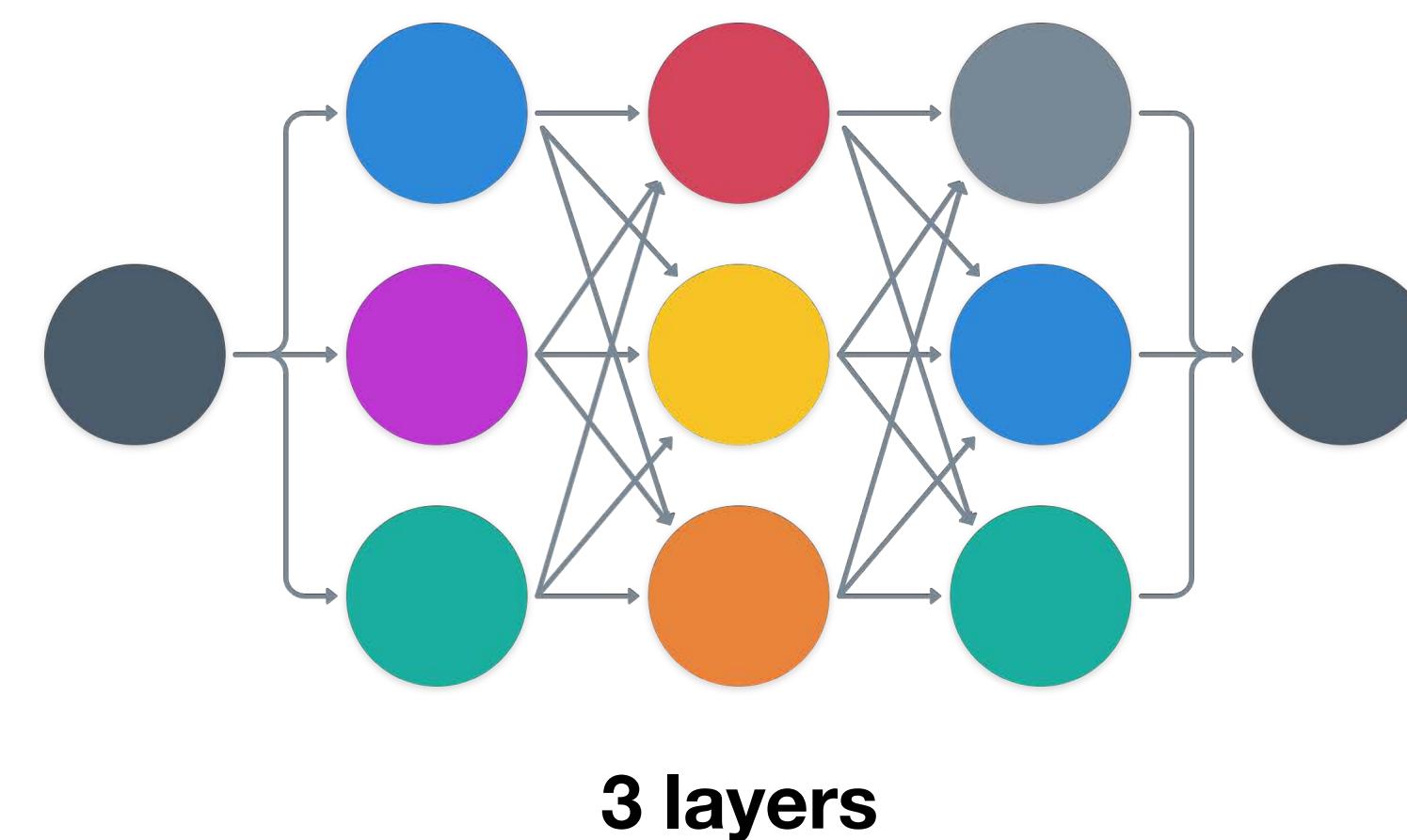
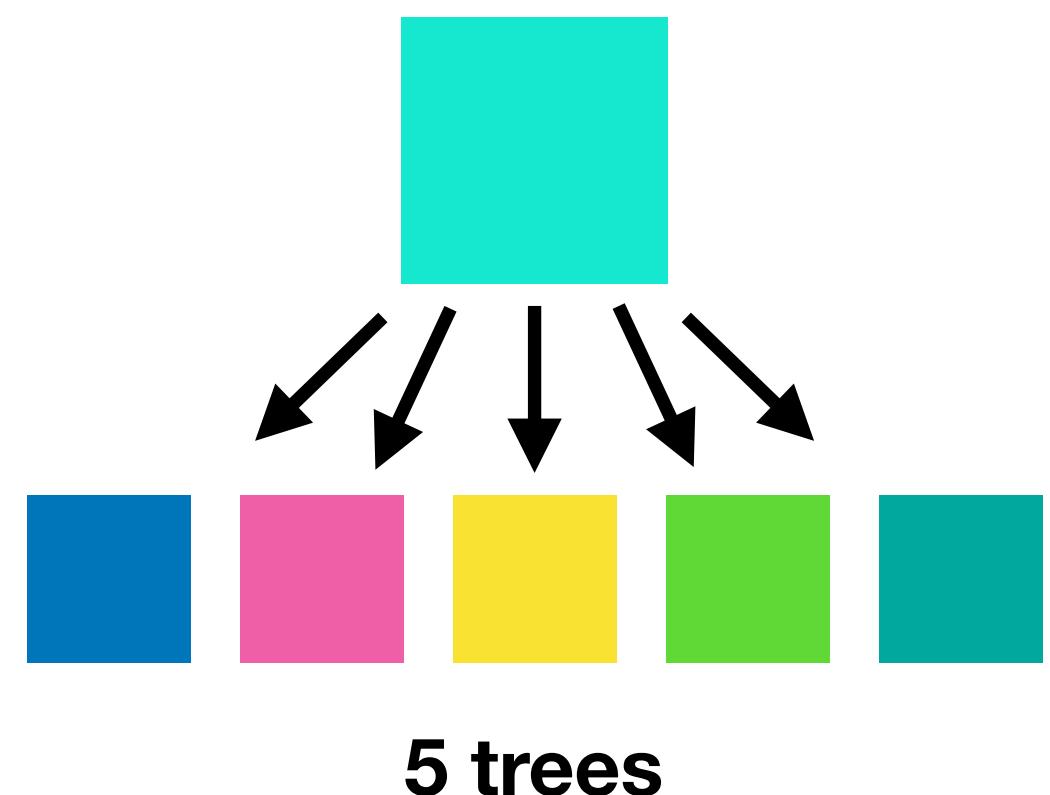
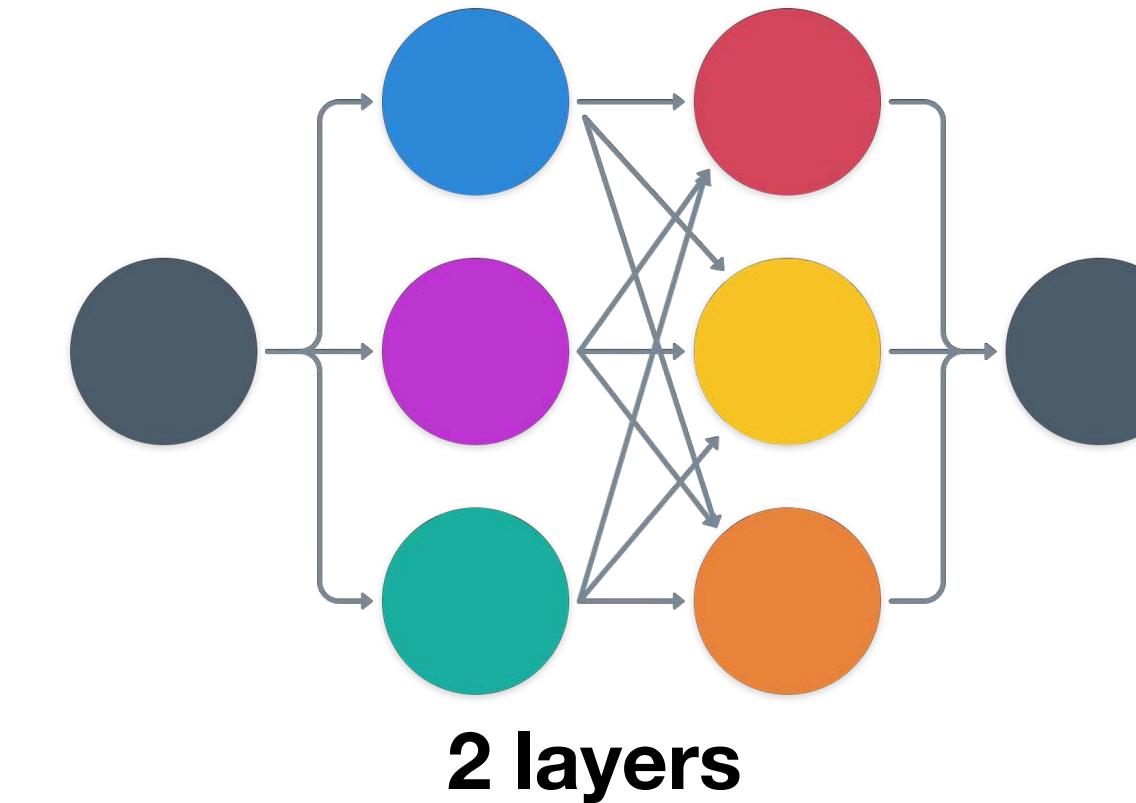
Cooking time: 1 hour
Temperature: 200°C

Tuning a model

Random Forest



Neural Networks



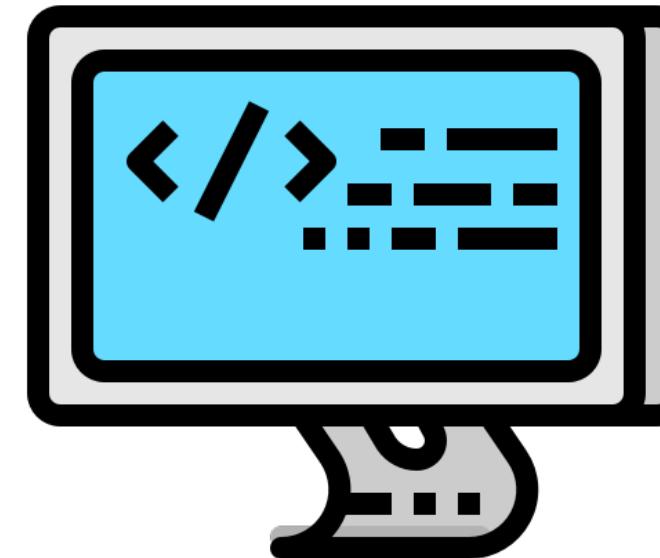
Things to remember

- 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

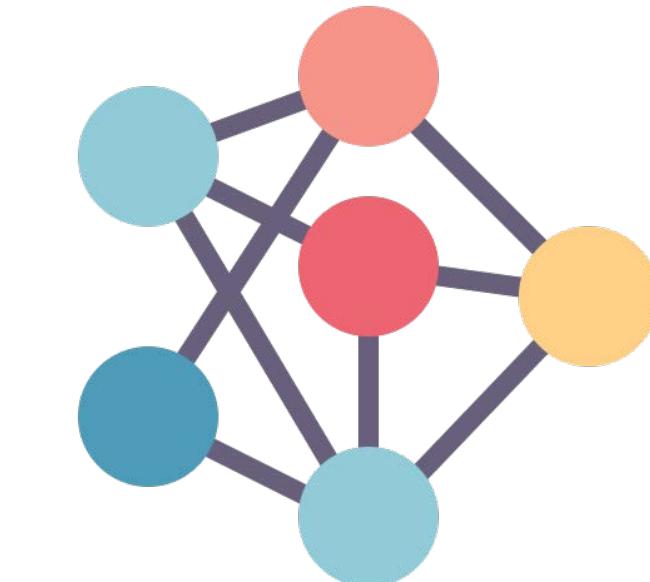
Up next

1. Choosing and training a model

Training Data

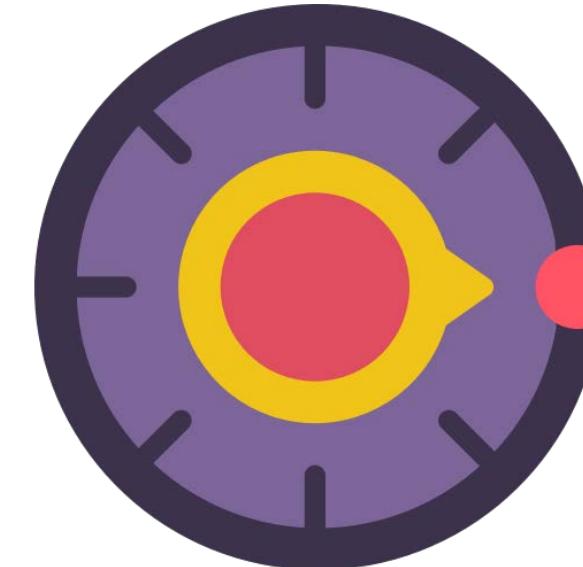


or



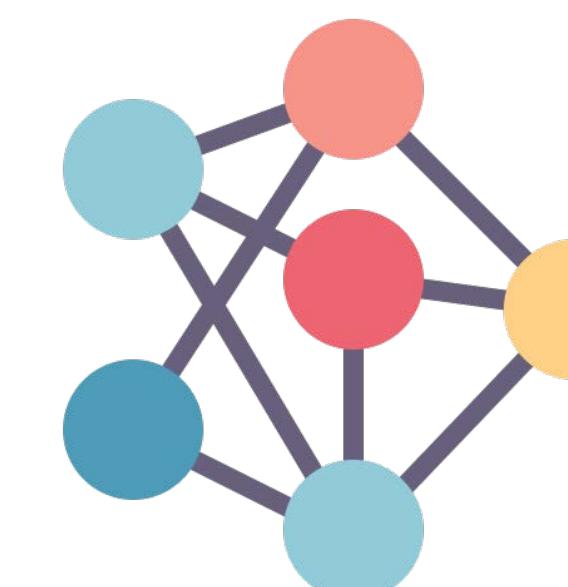
2. Tuning a model

Validation Data

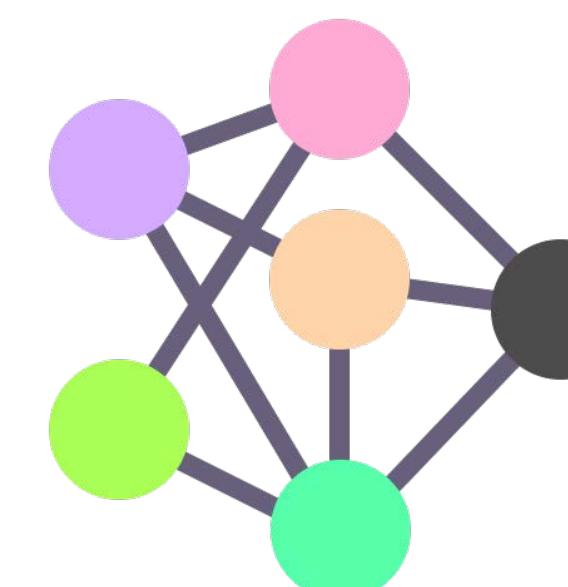


3. Model comparison

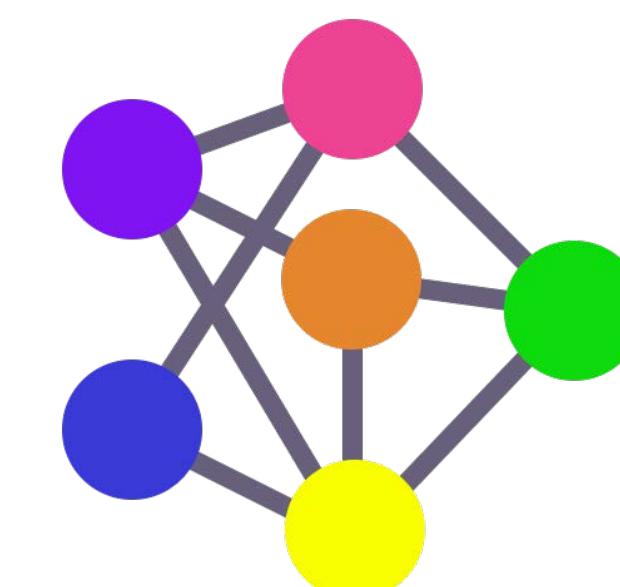
Test Data



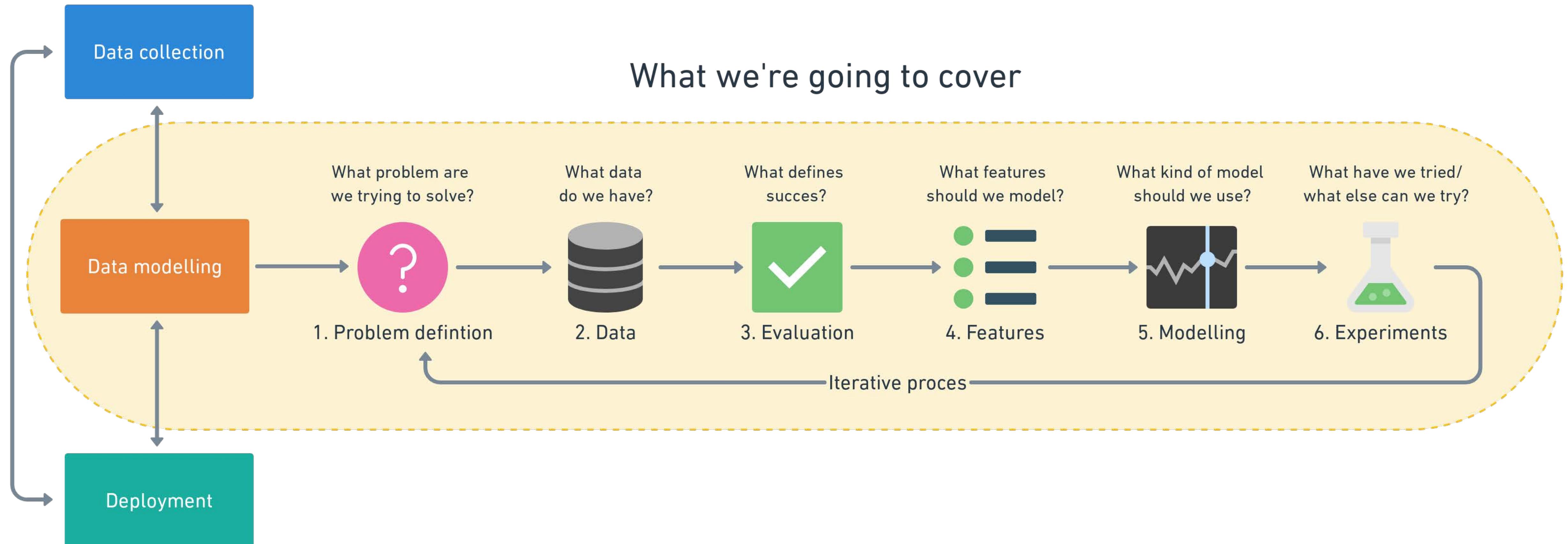
vs.



vs.



Steps in a full machine learning project





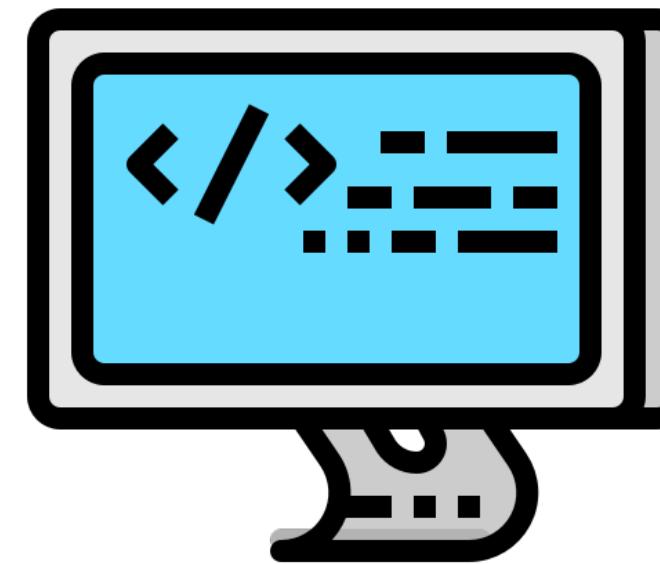
5. Modelling Part 4 – Comparison

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

3 parts to modelling

1. Choosing and training a model

Training Data



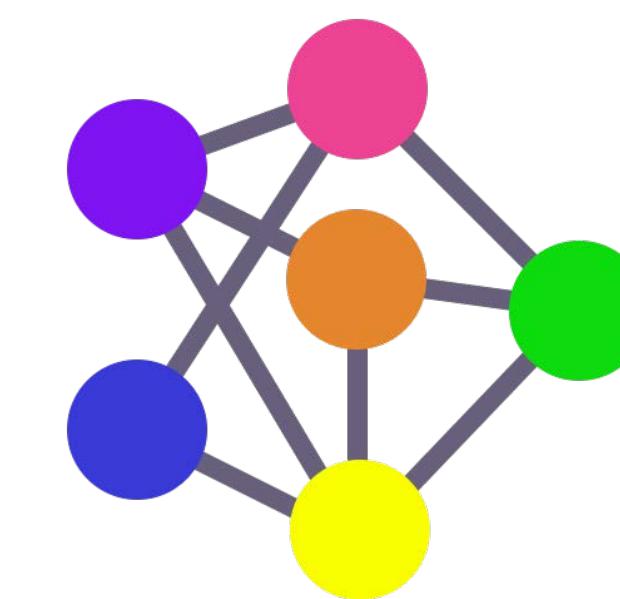
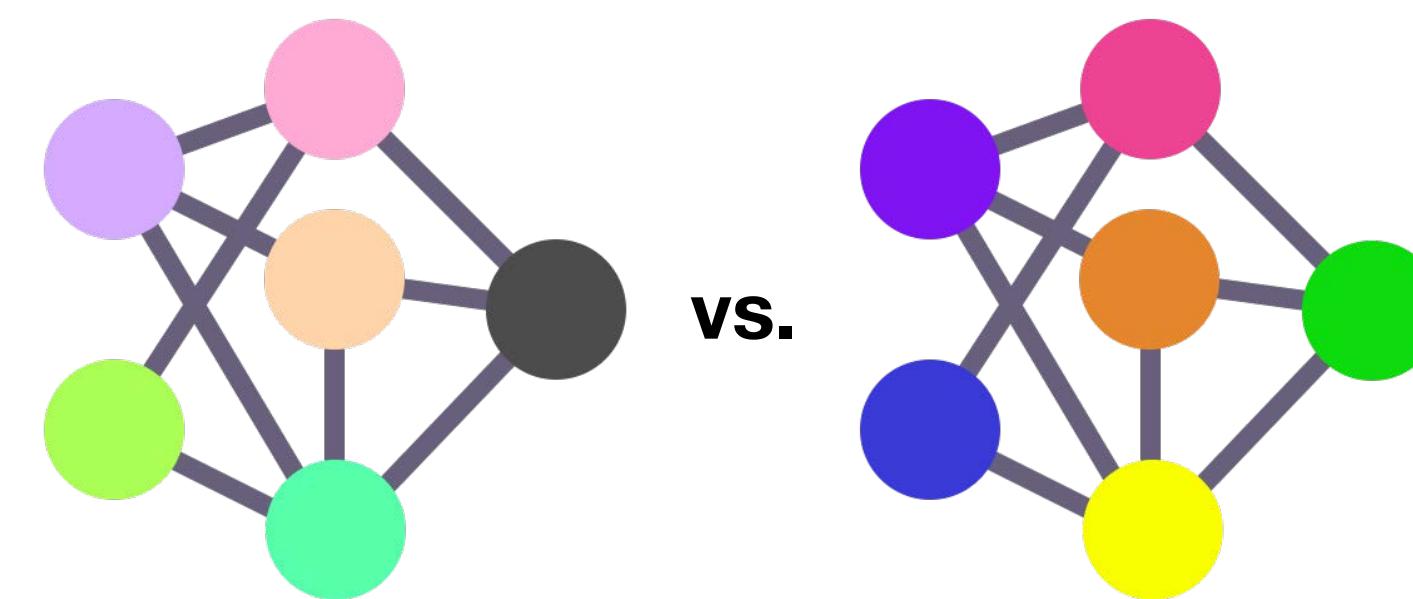
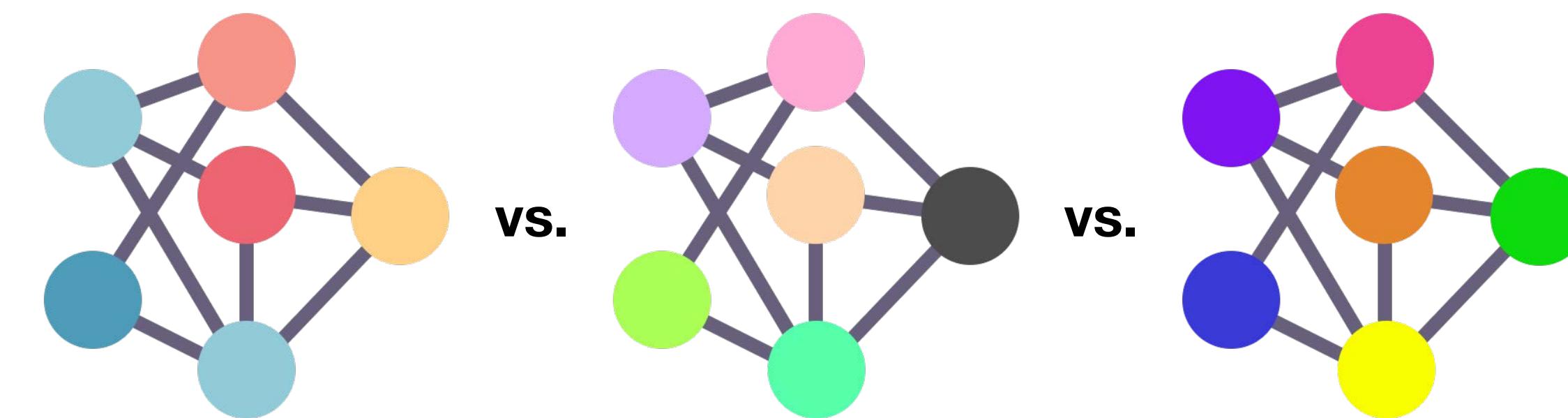
2. Tuning a model

Validation Data

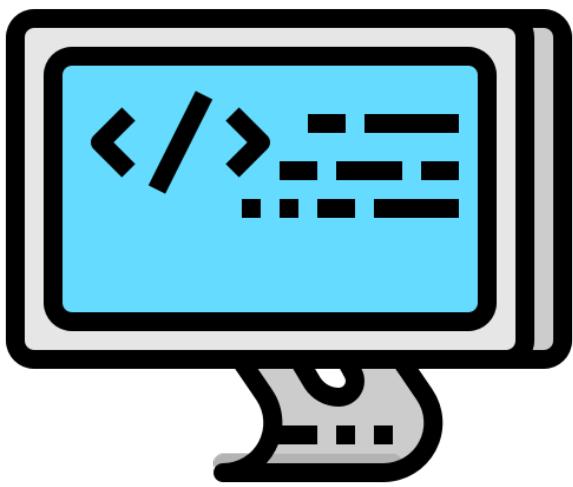


3. Model comparison

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

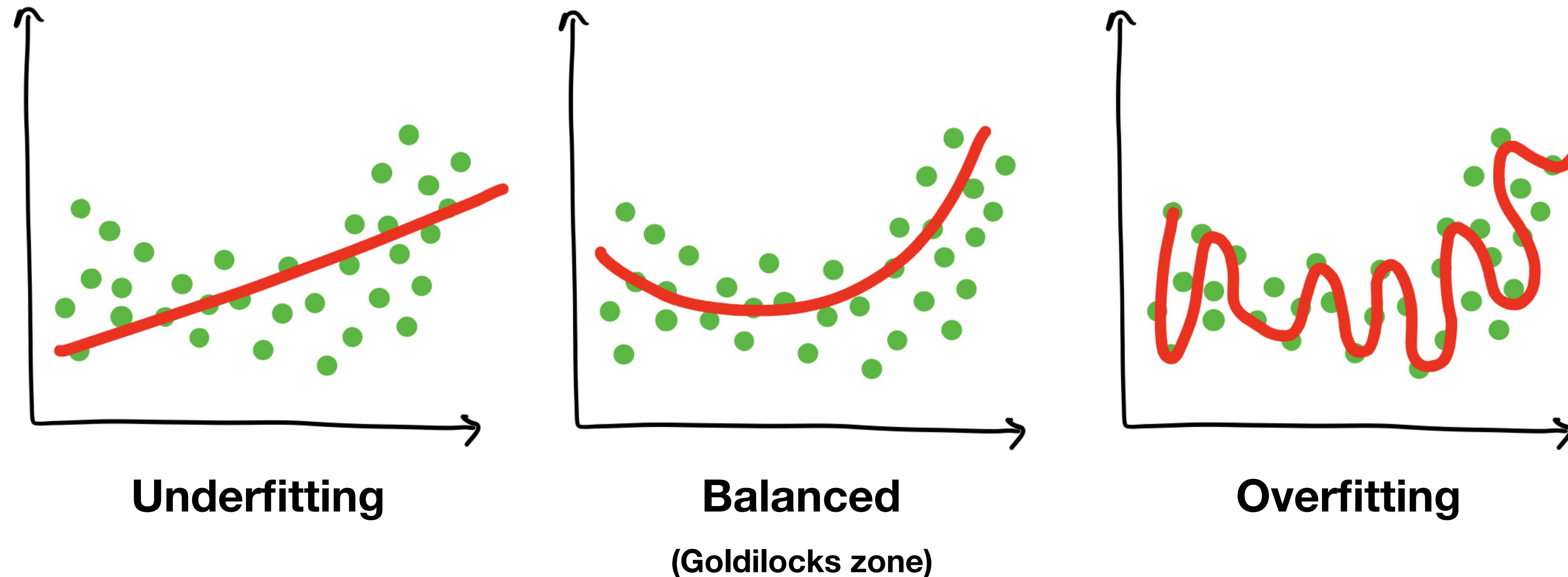
Training

93%

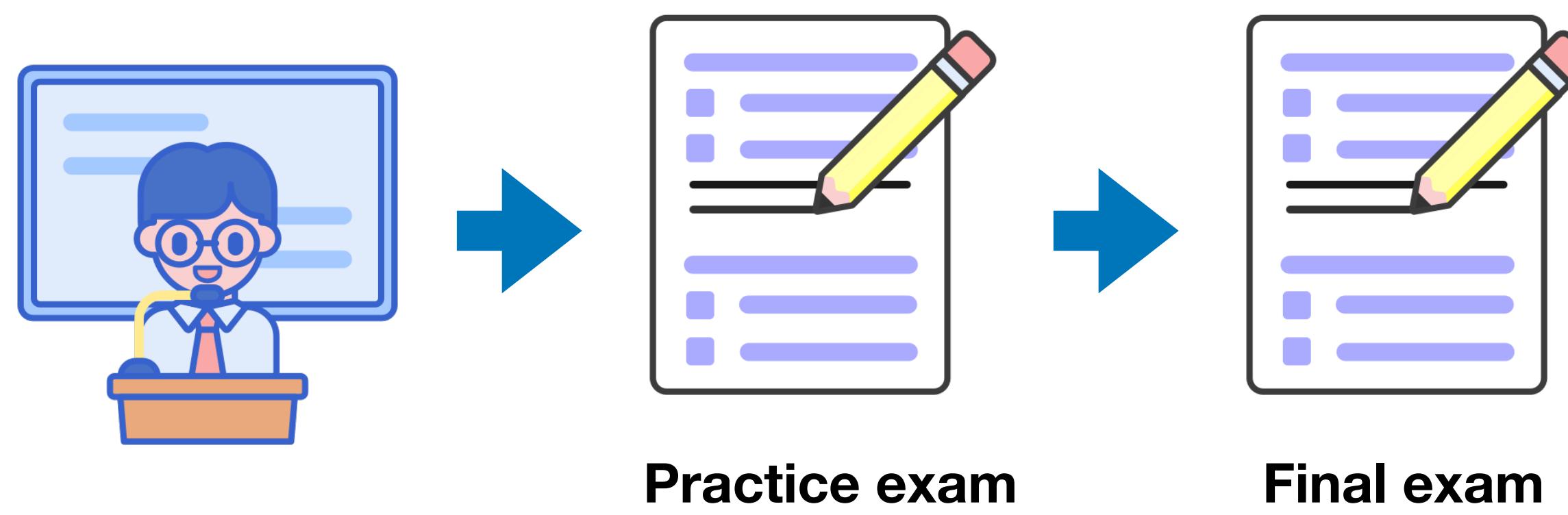
Test

99%

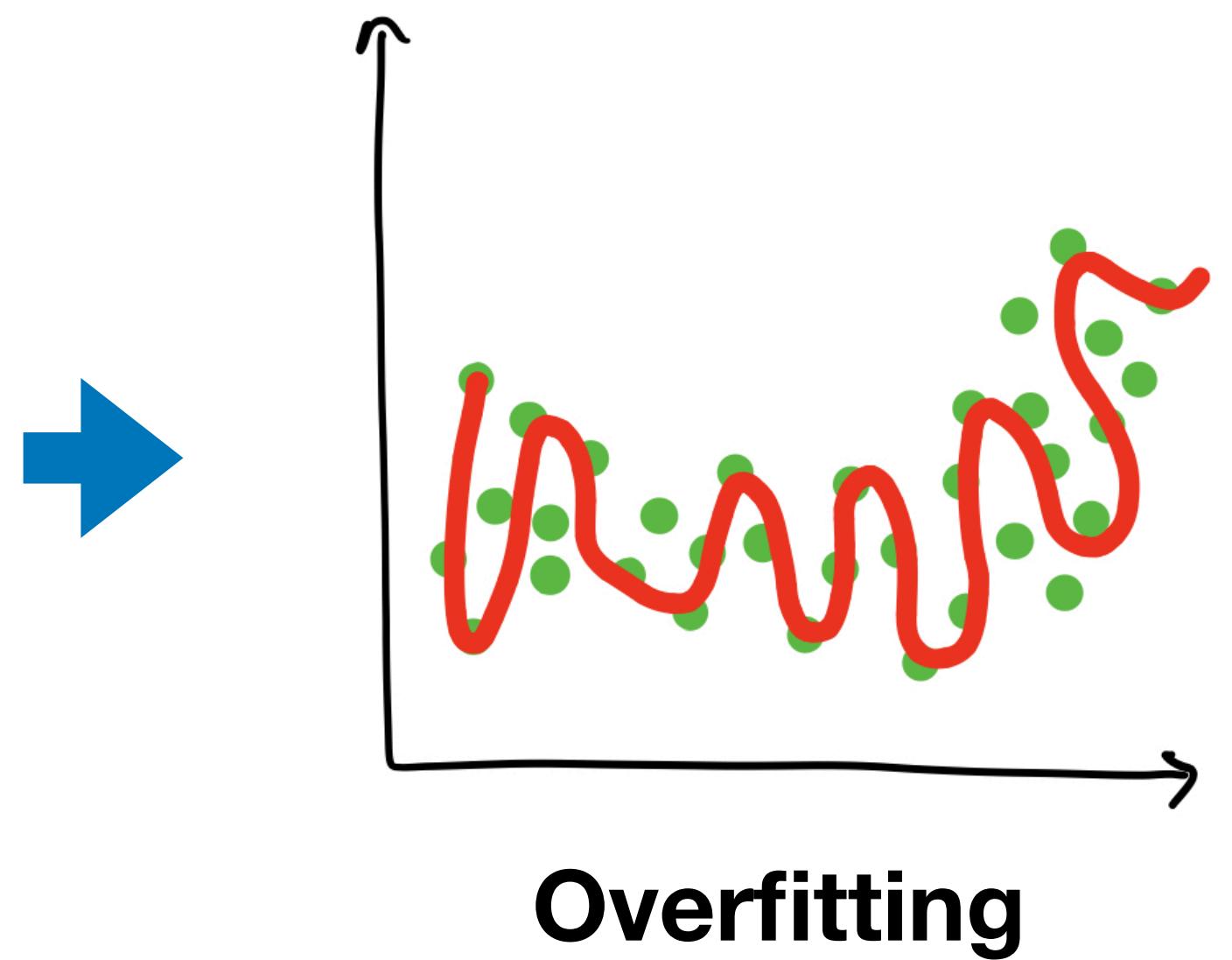
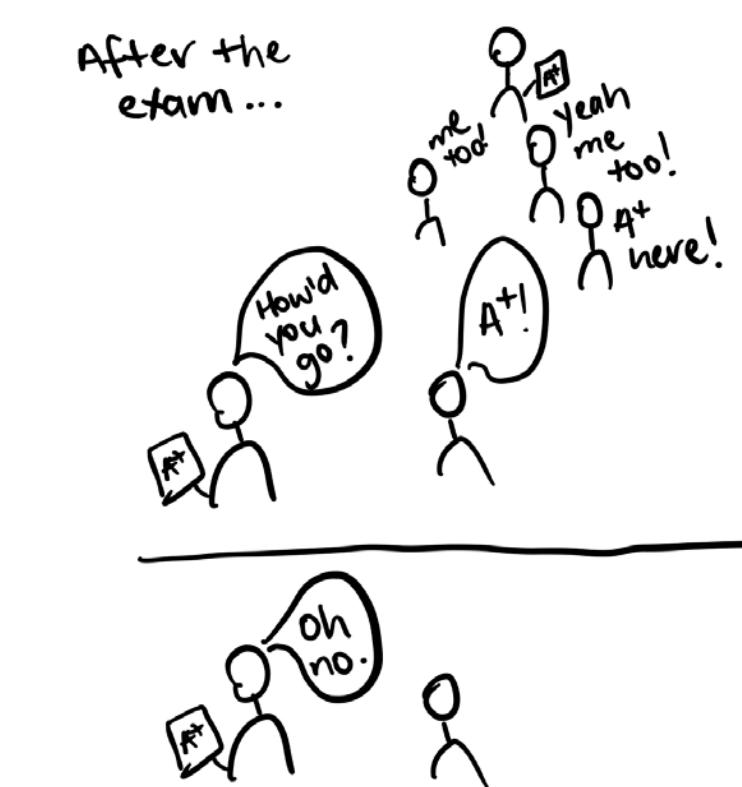
Overfitting and underfitting



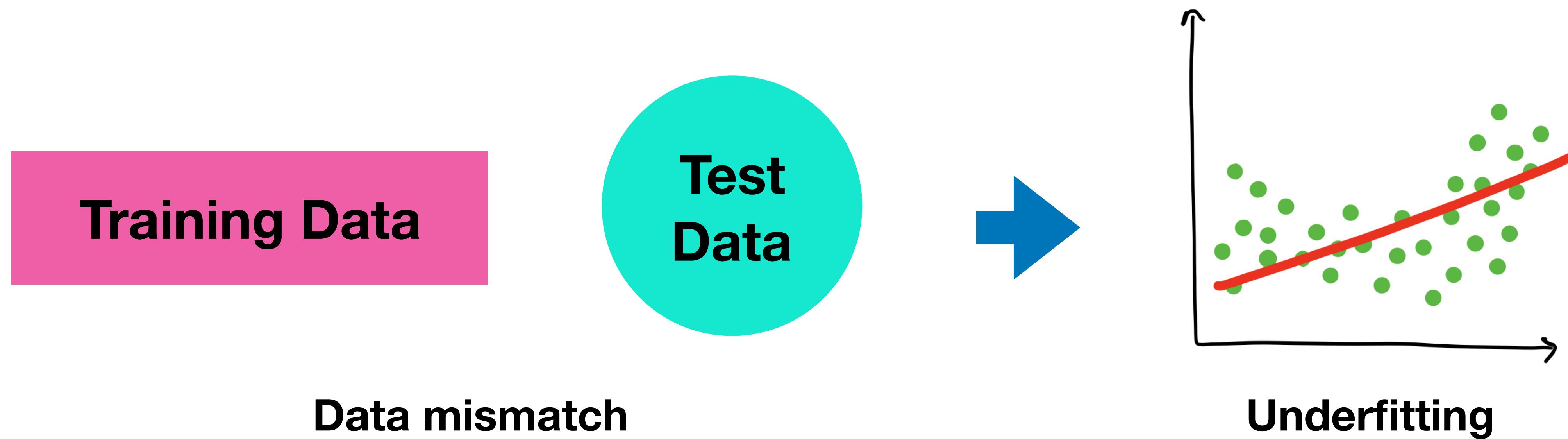
Overfitting and underfitting



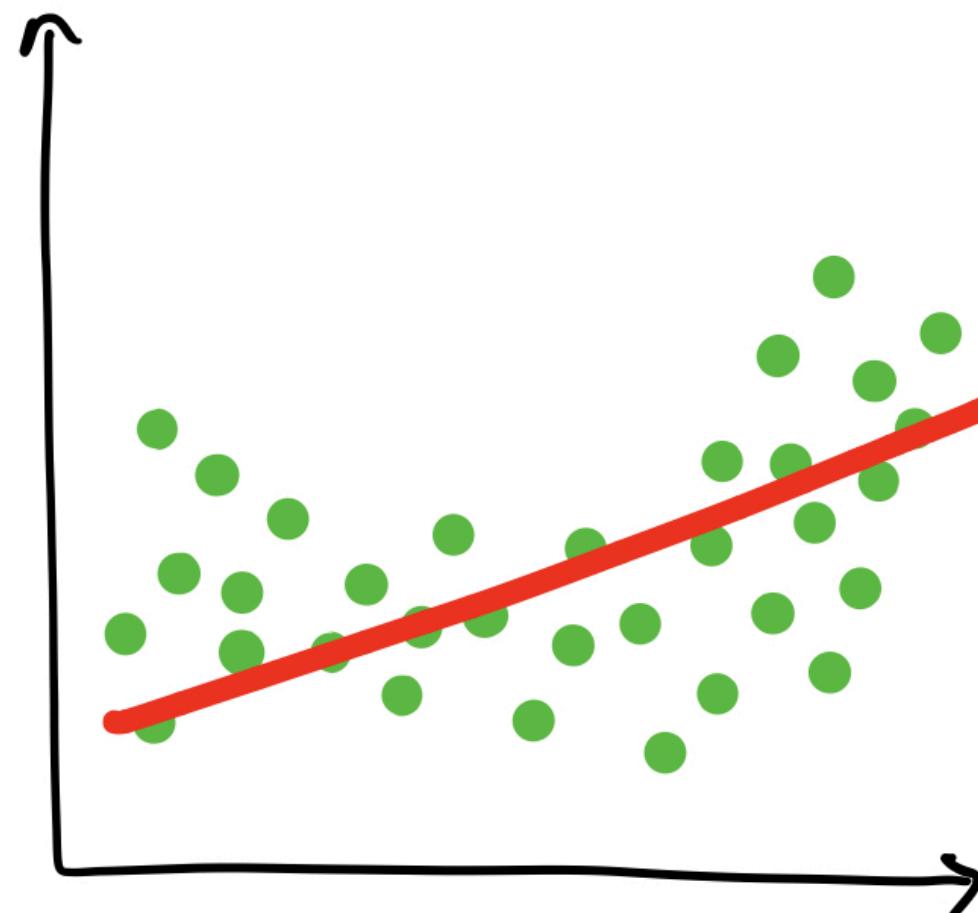
Data leakage



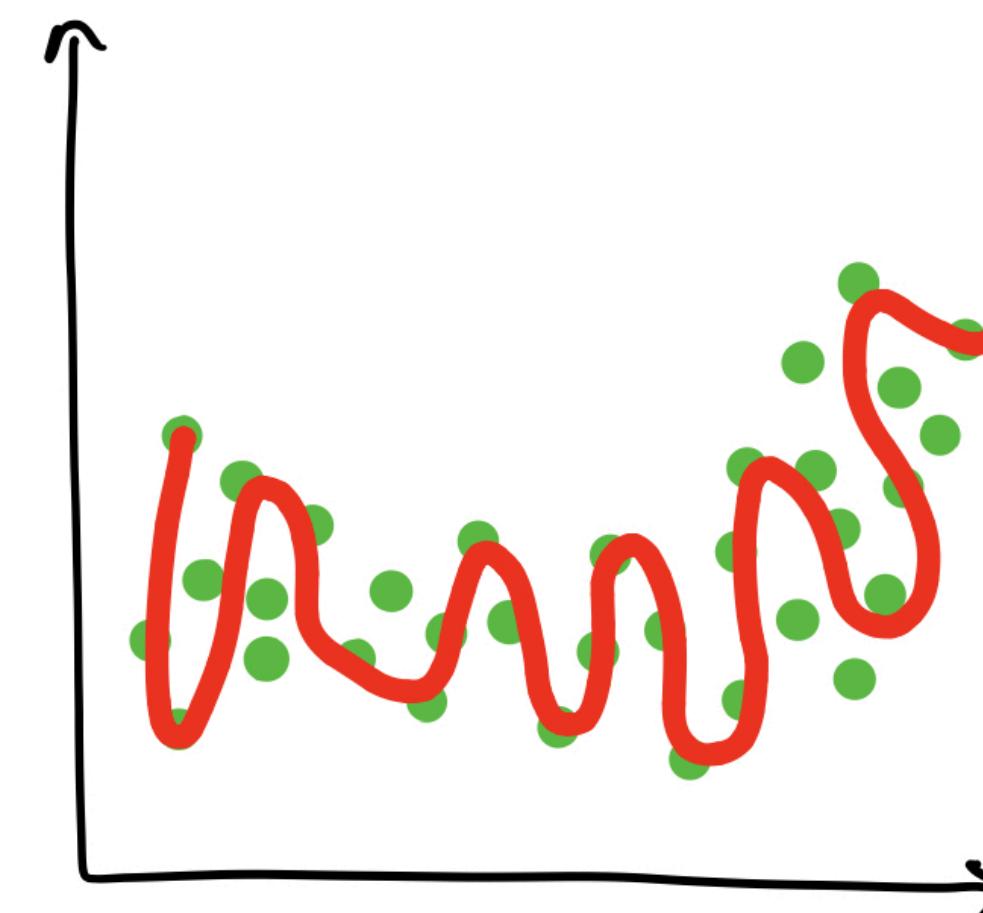
Overfitting and underfitting



Fixes for overfitting and underfitting



Underfitting



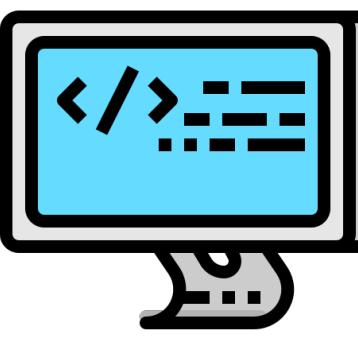
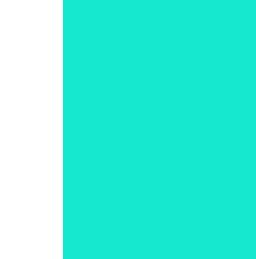
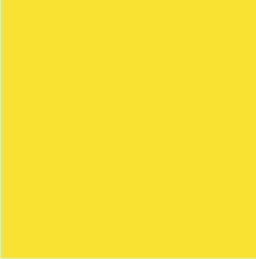
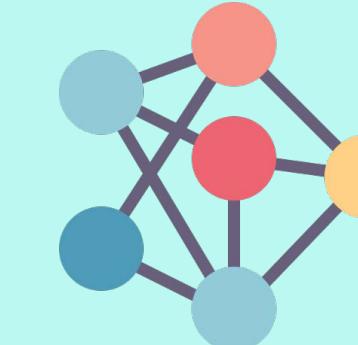
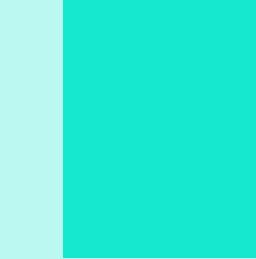
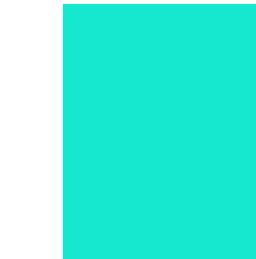
Overfitting

- 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

Up next

6. Experimentation



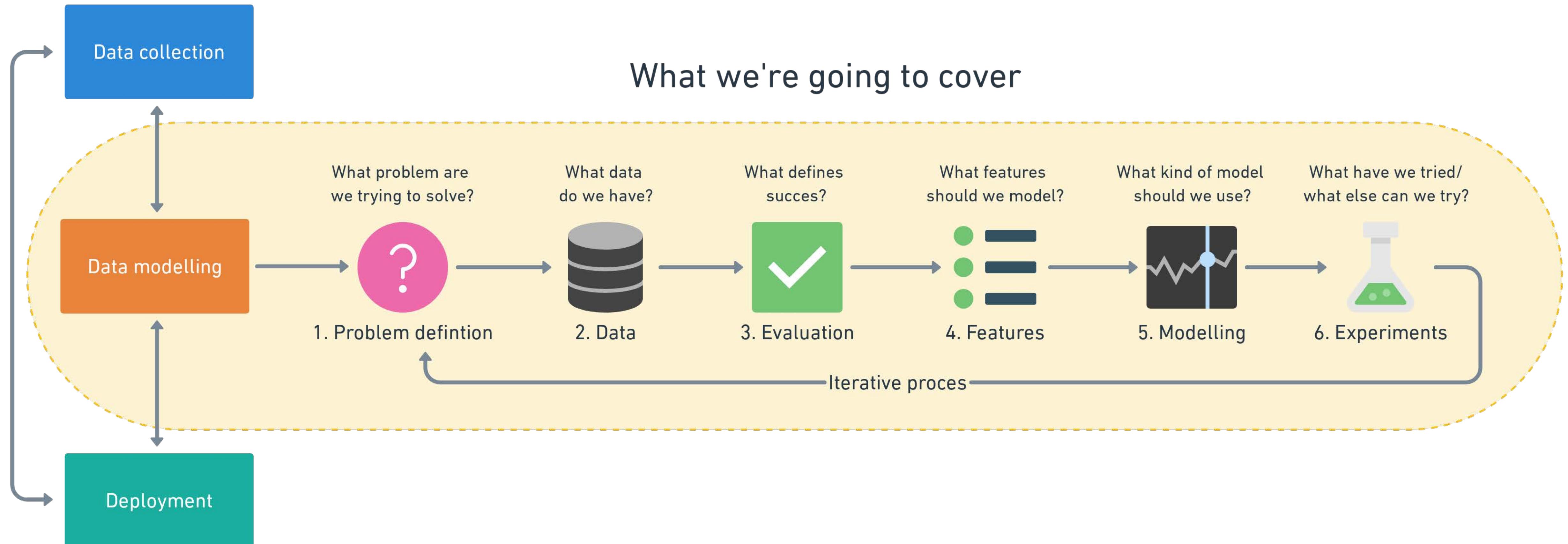
“How could we improve/what can we try next?”

6. Experimentation

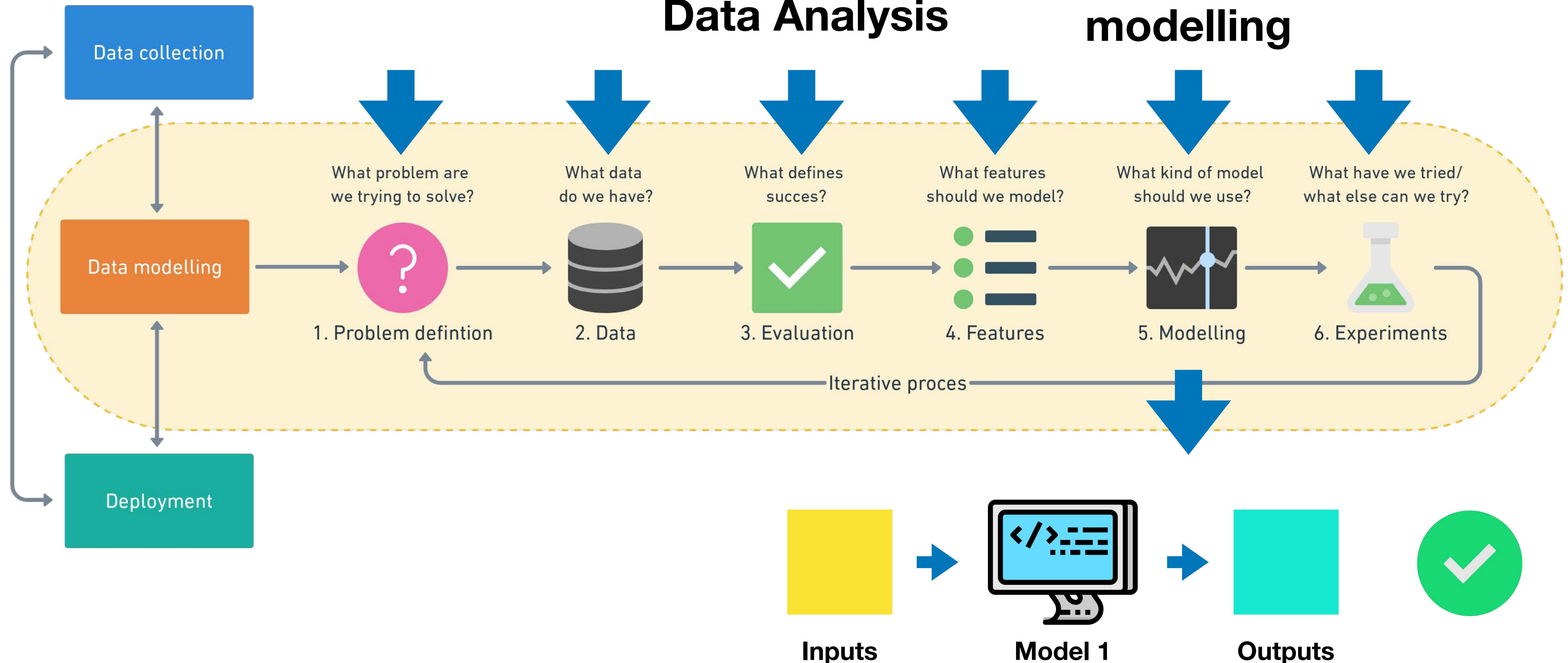


“How could we improve/what can we try next?”

Steps in a full machine learning project

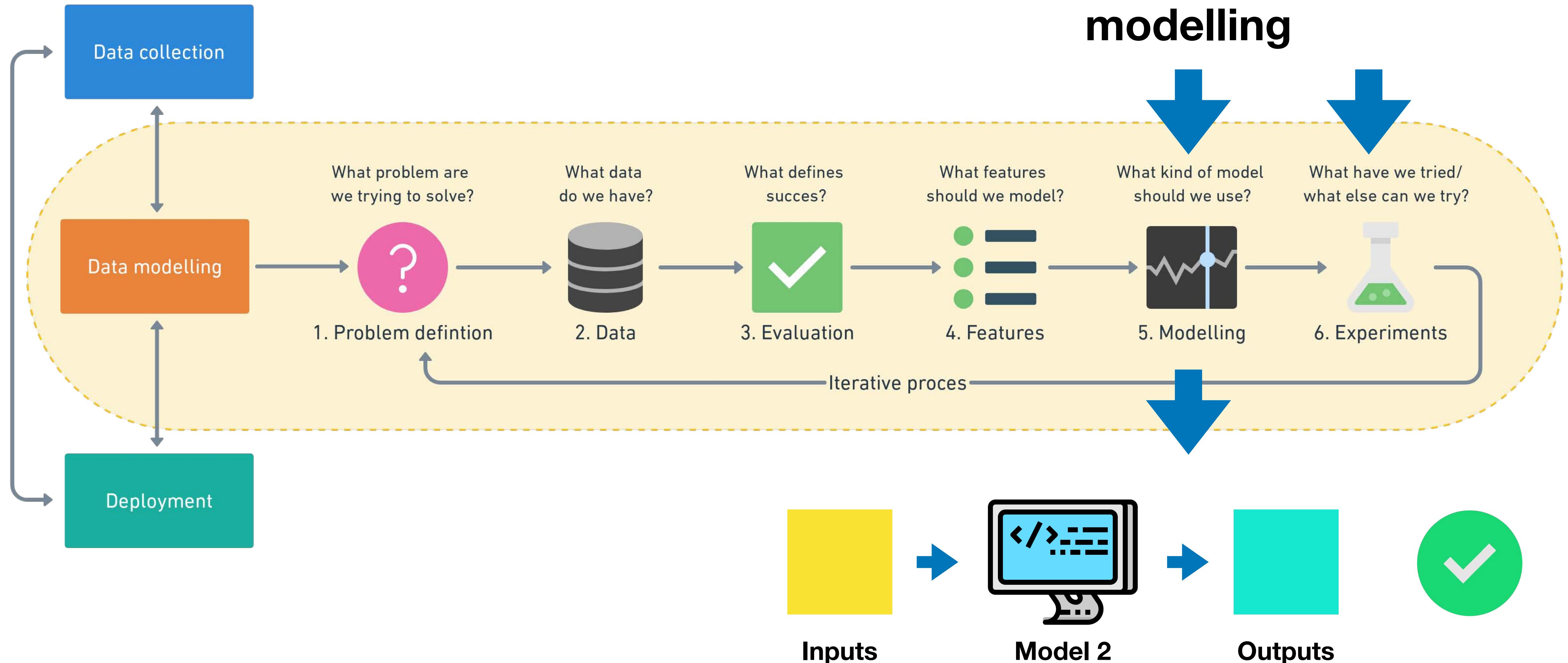


Steps in a full machine learning project



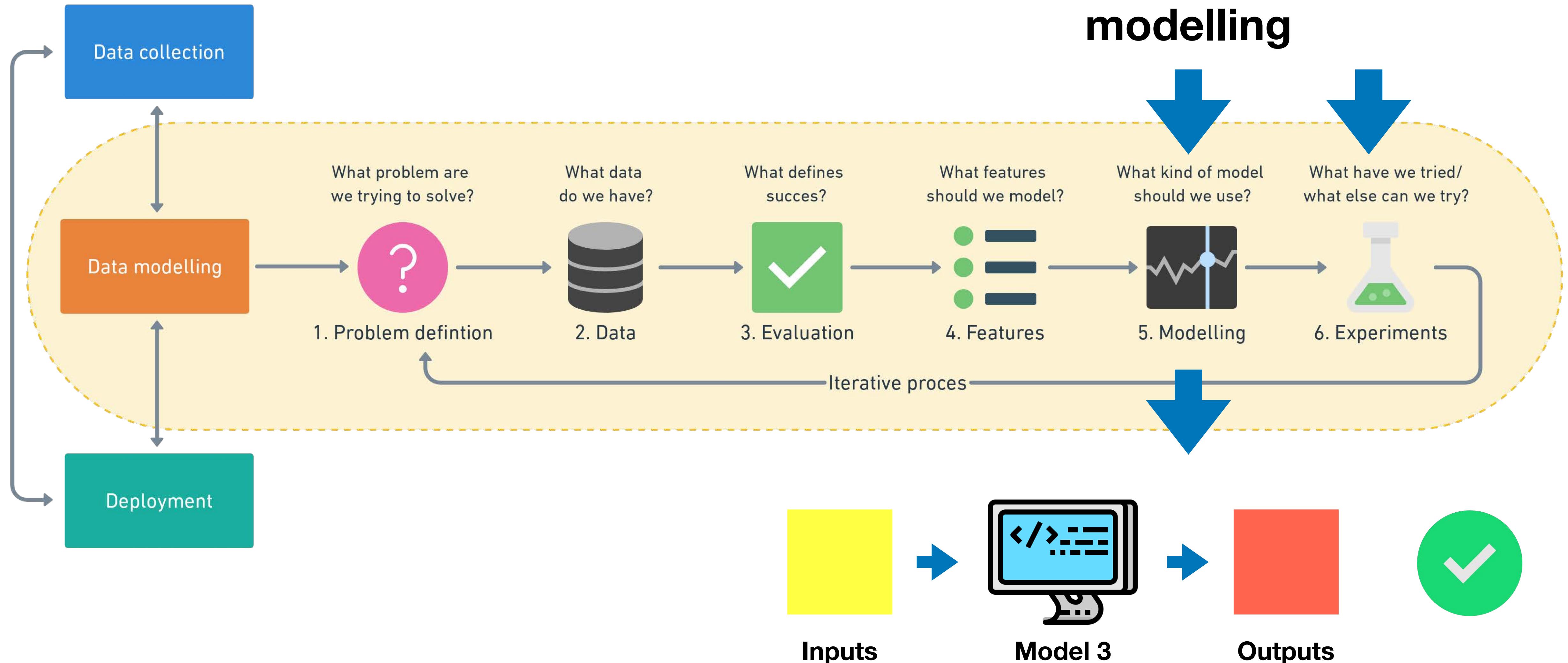
Machine learning modelling

Steps in a full machine learning project



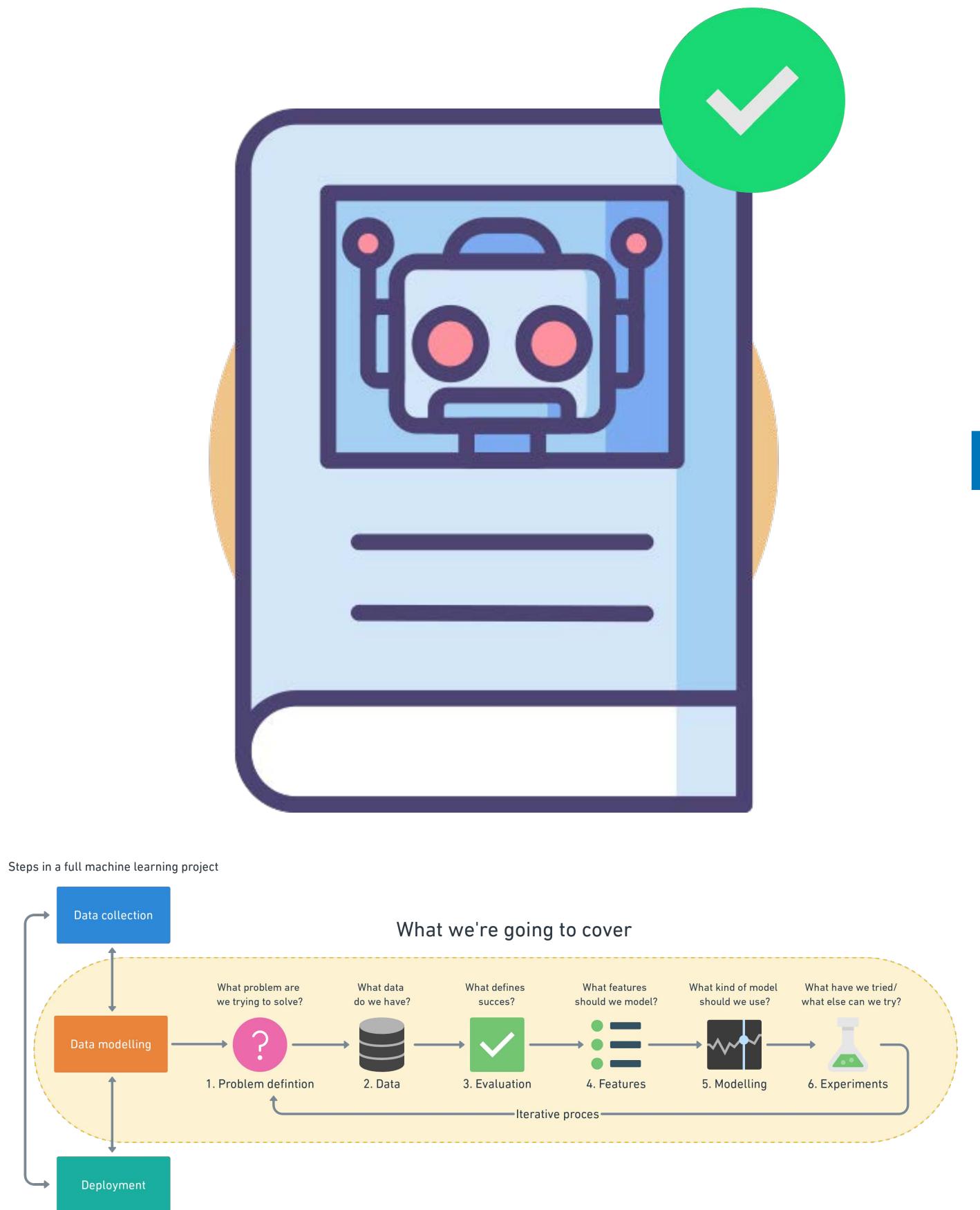
Machine learning modelling

Steps in a full machine learning project

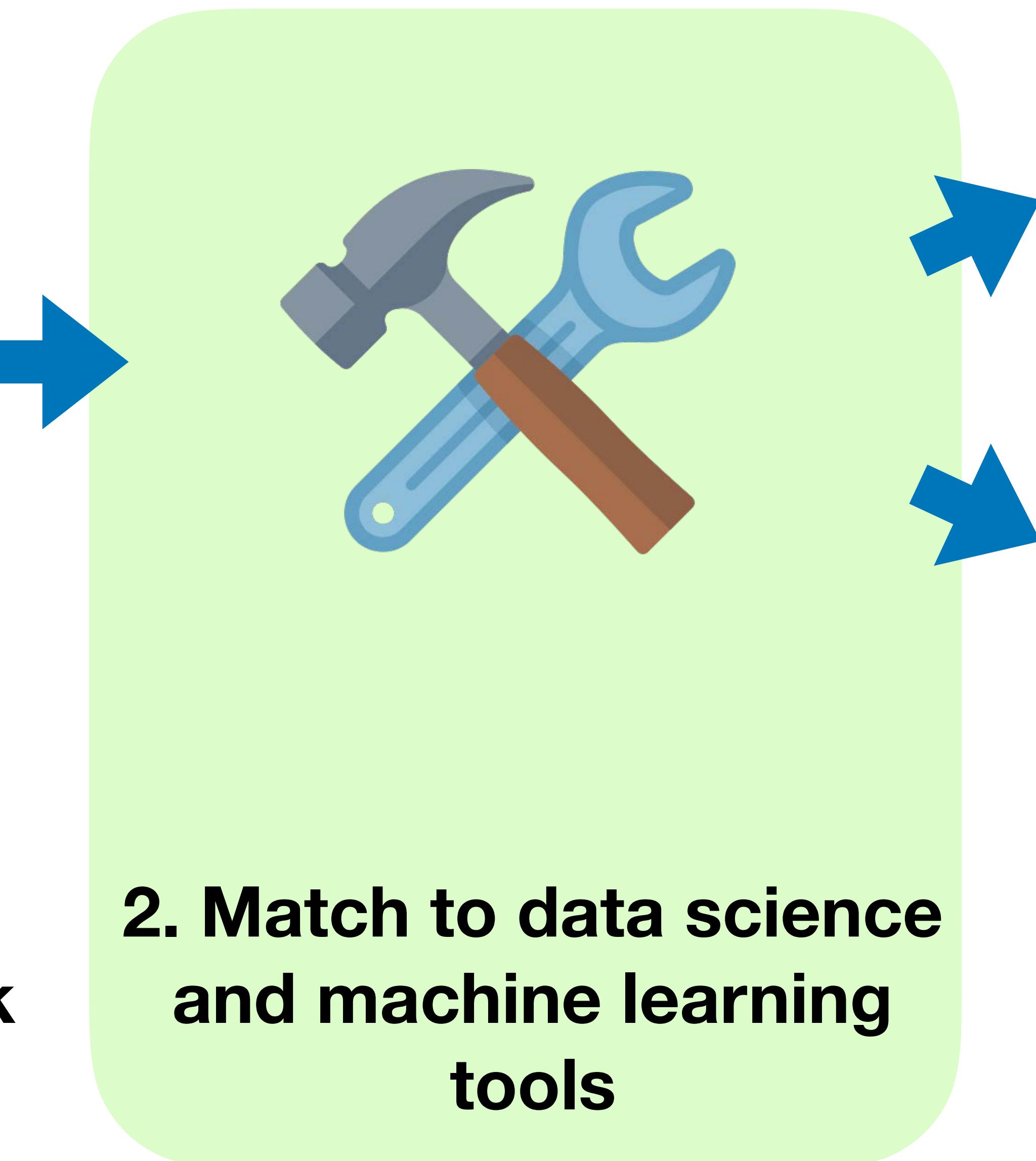


Up next

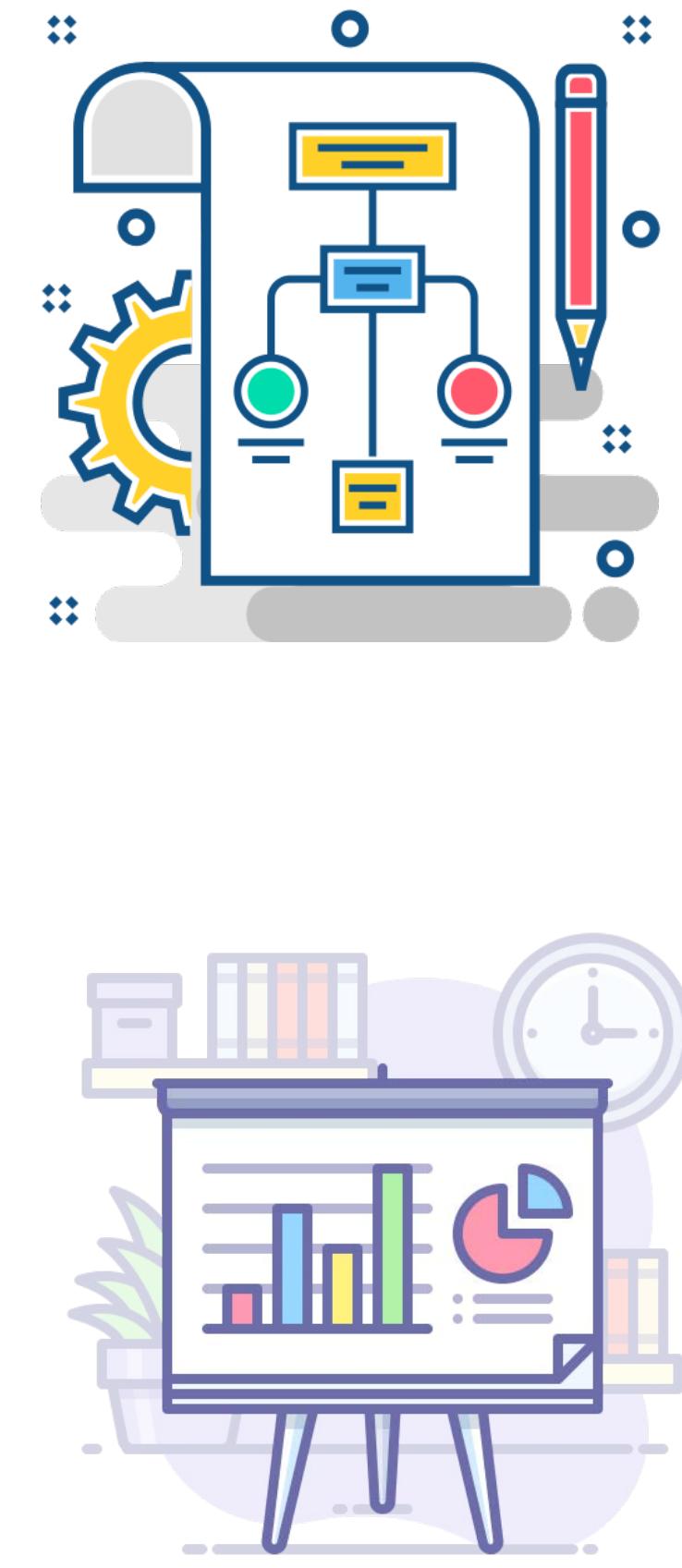
1. Create a framework



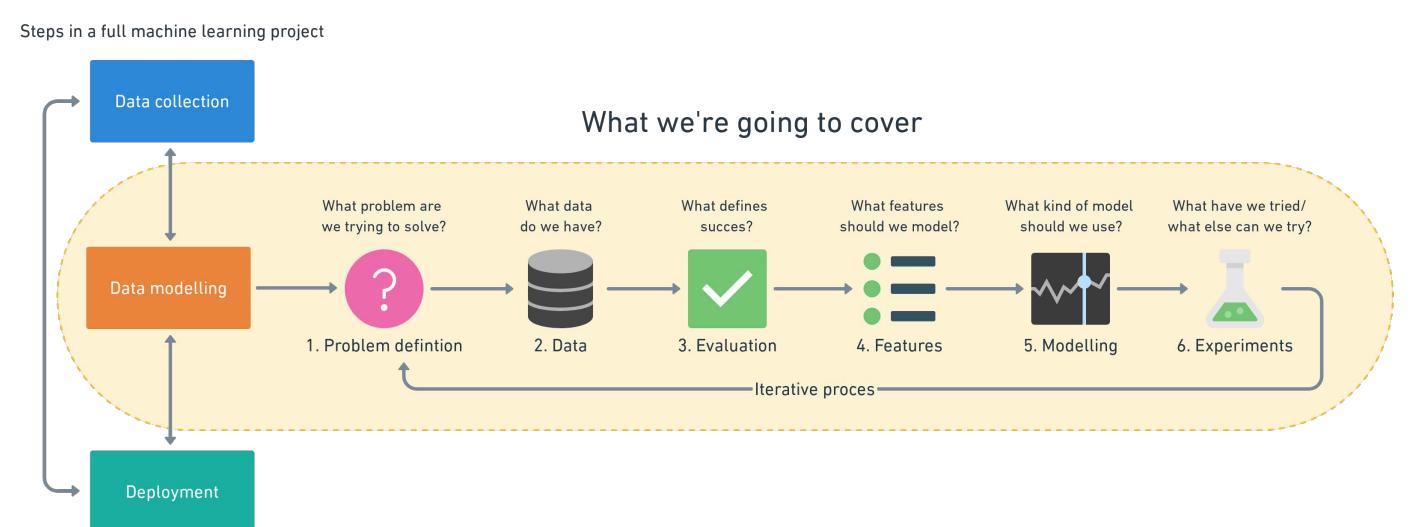
2. Match to data science and machine learning tools



3. Learn by doing



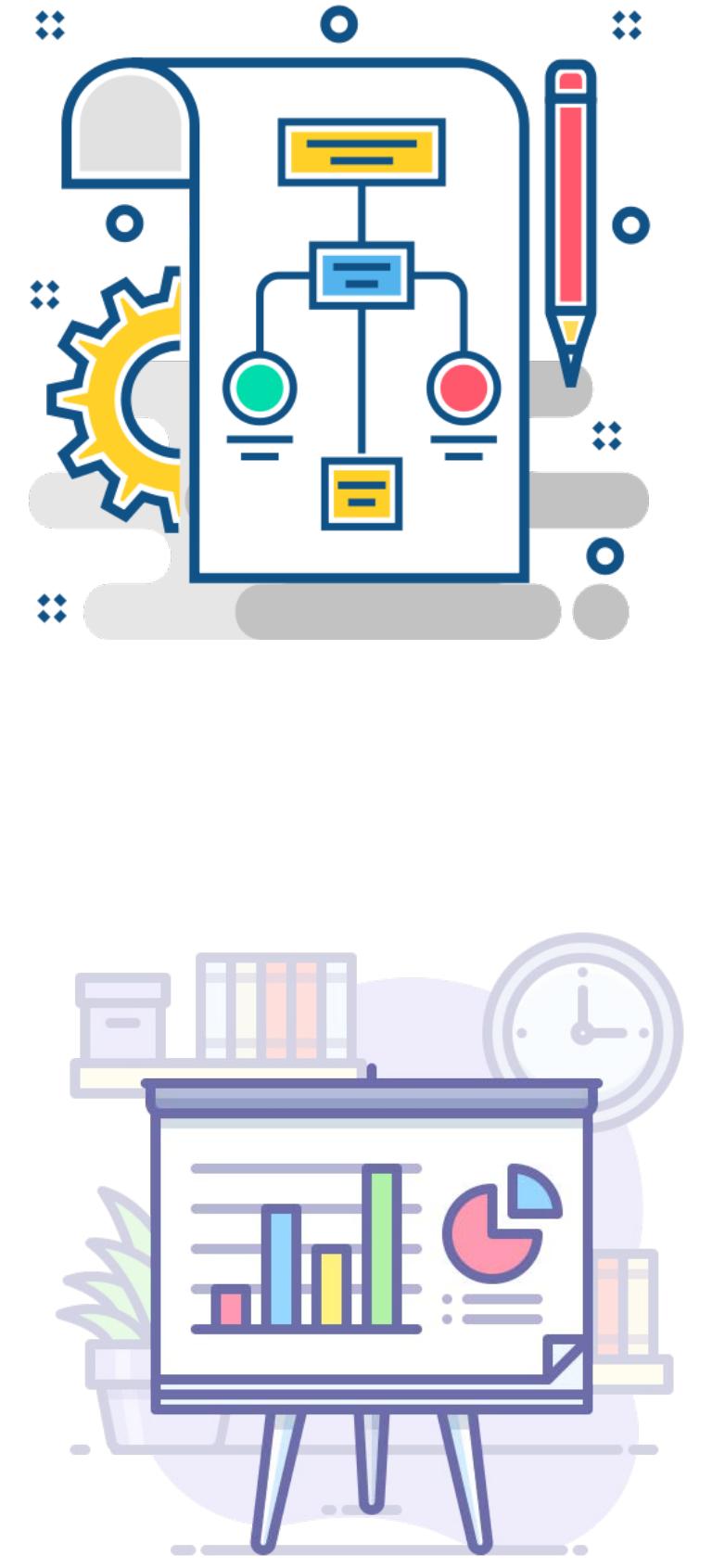
1. Create a framework



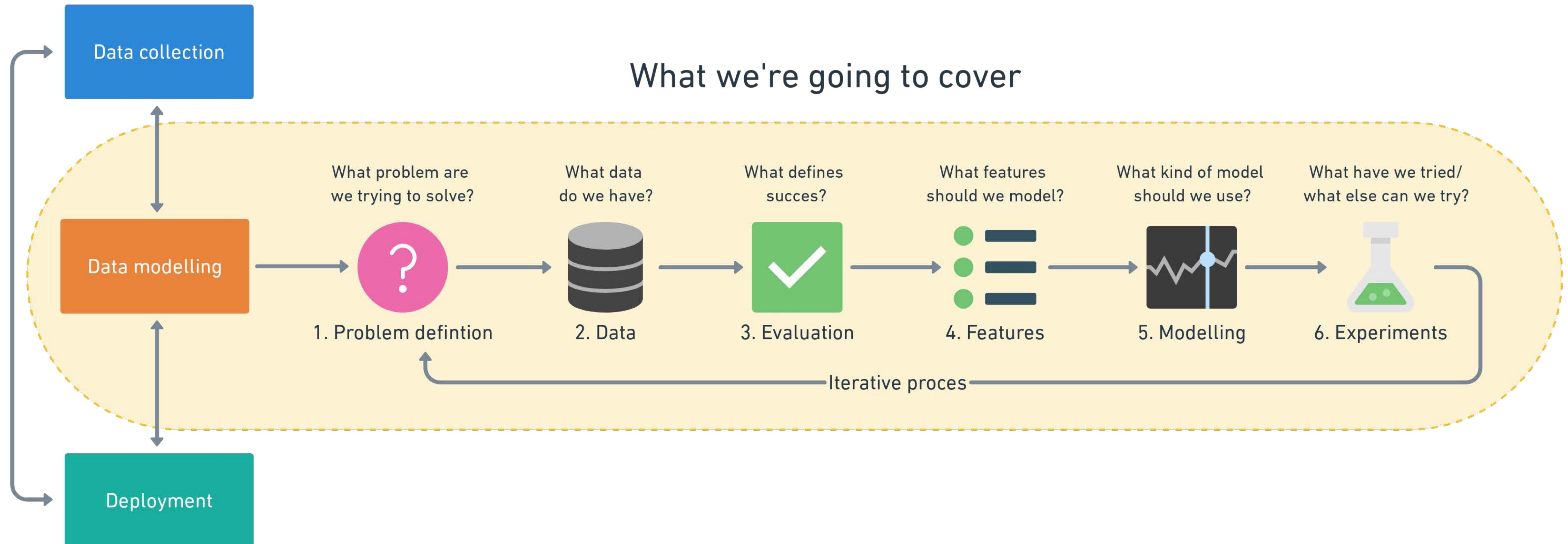
2. Match to data science and machine learning tools



3. Learn by doing

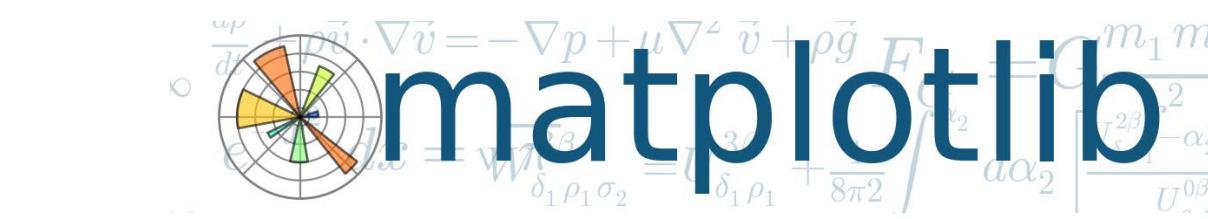
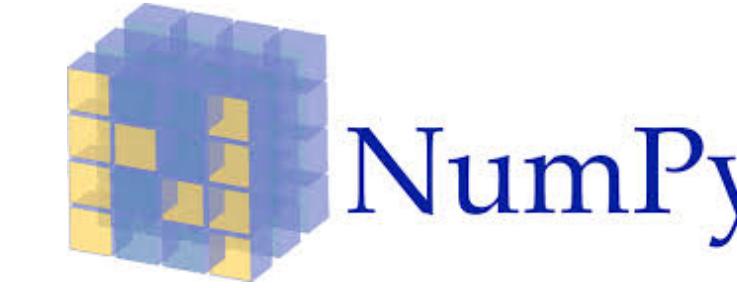
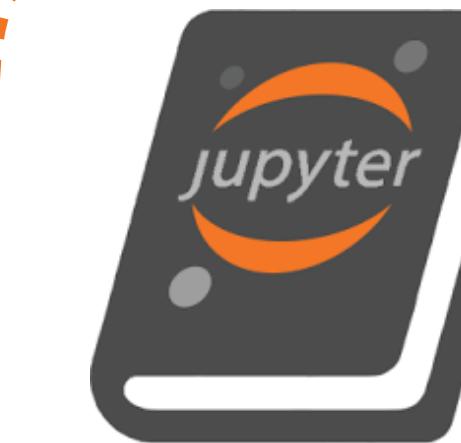


Steps in a full machine learning project

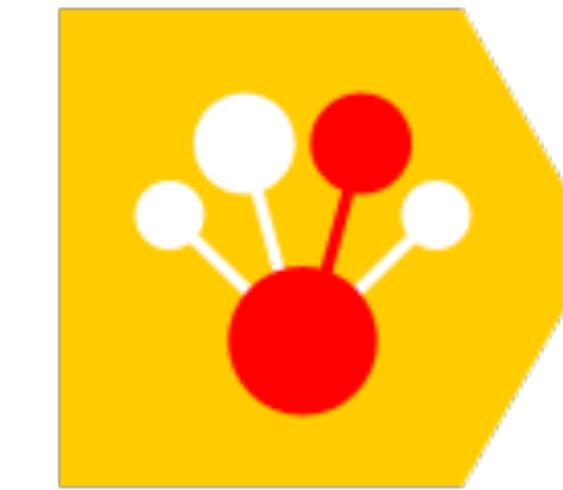
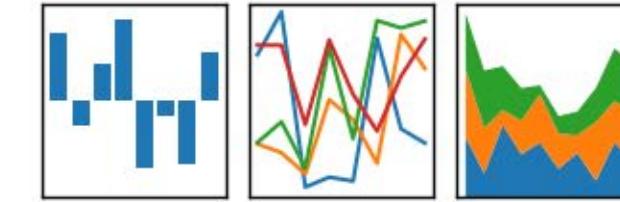




Your
computer



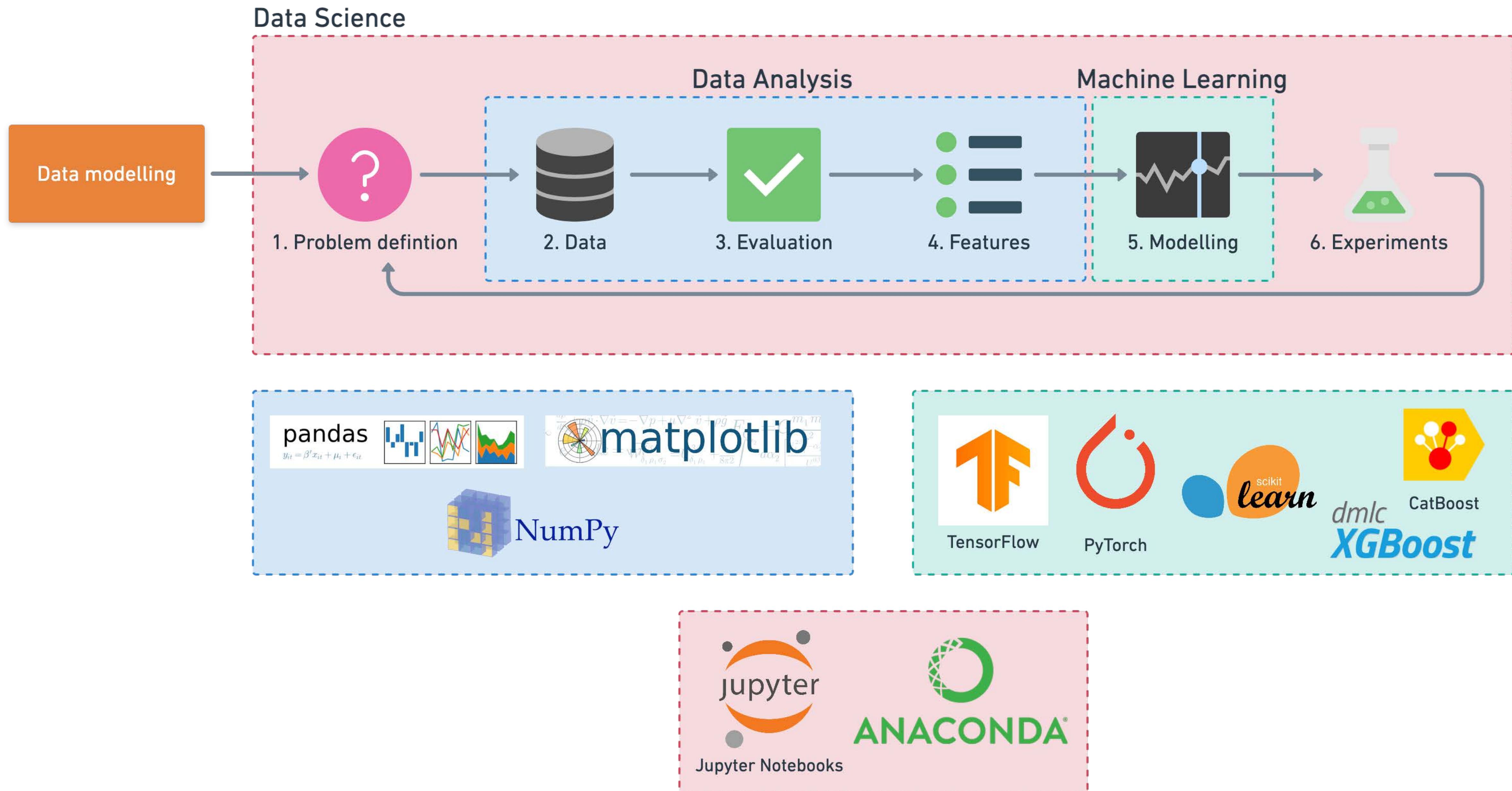
pandas
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



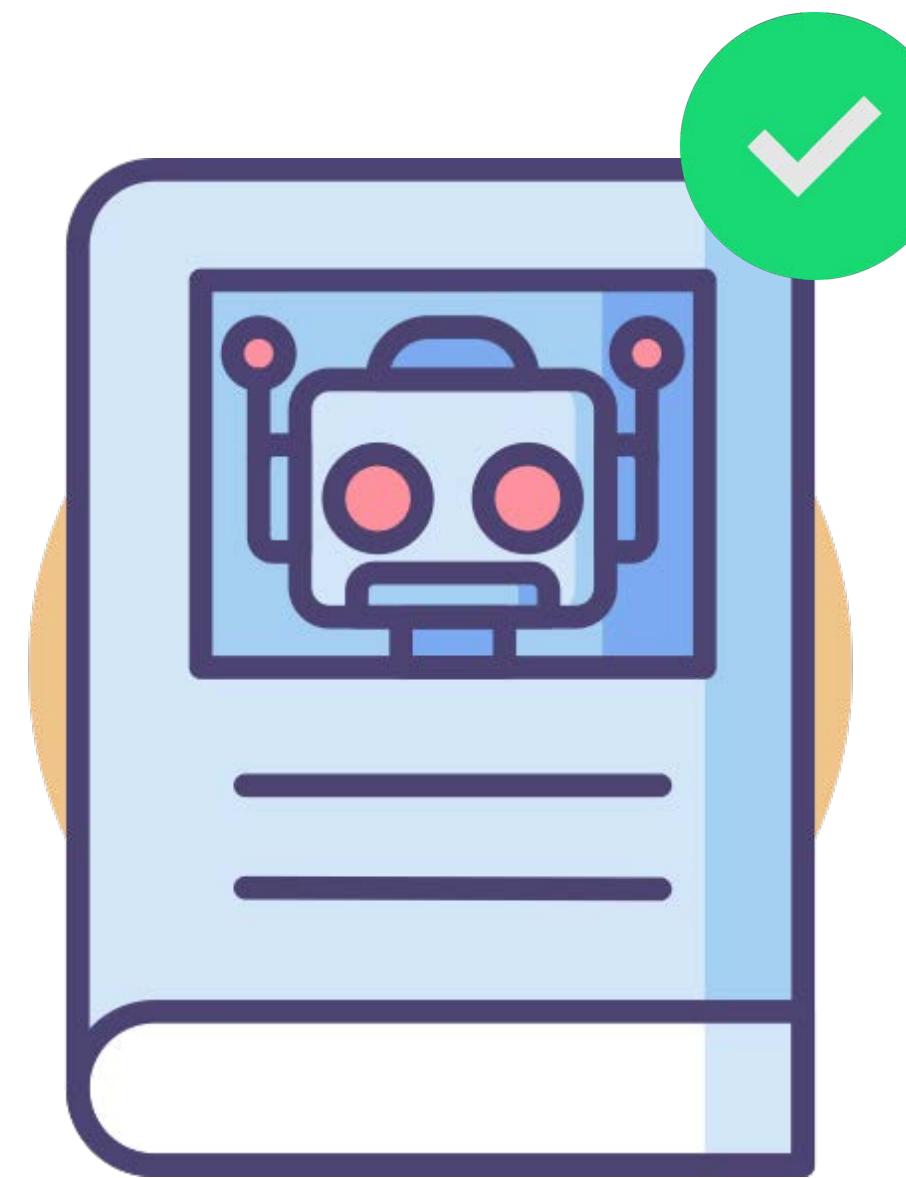
dmlc
XGBoost



Tools you can use



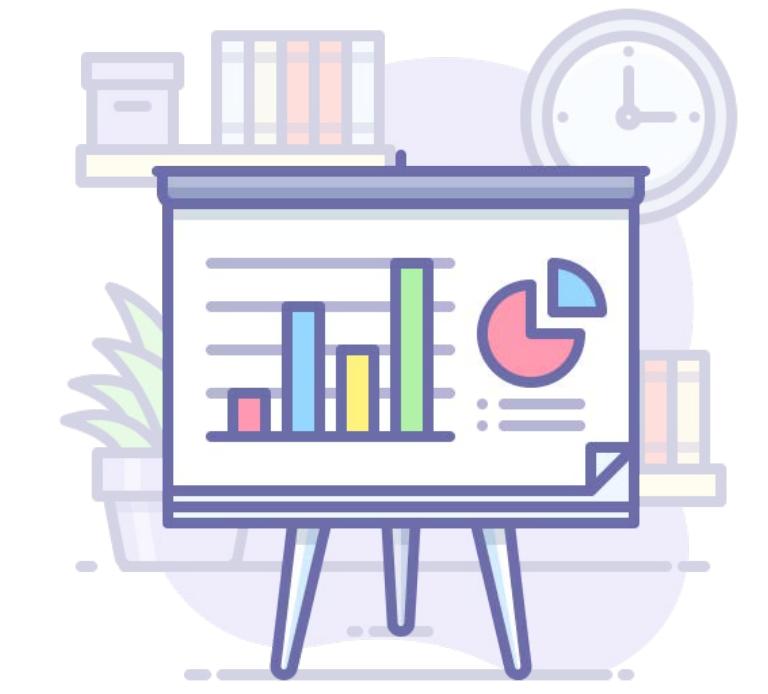
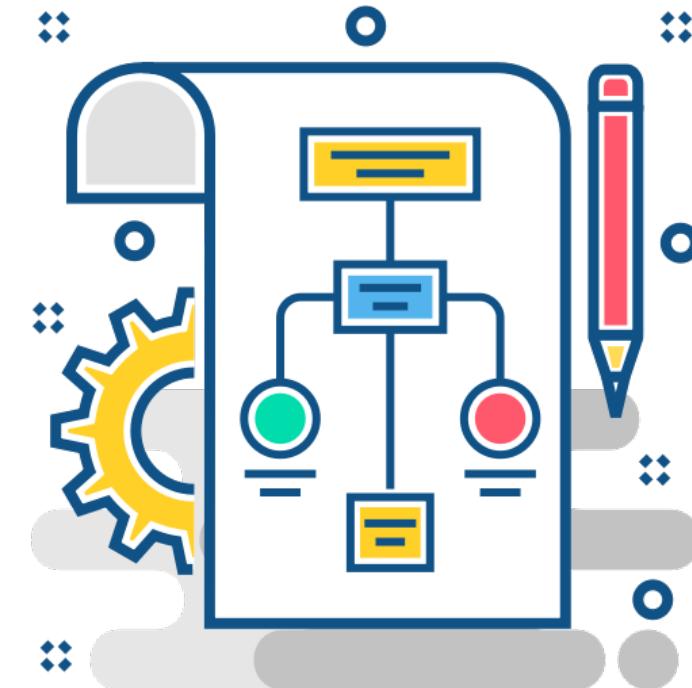
1. Create a framework



2. Match to data science and machine learning tools



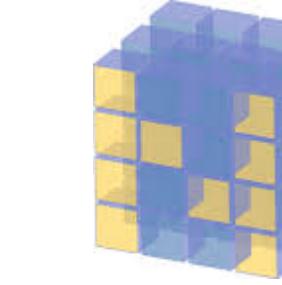
3. Learn by doing



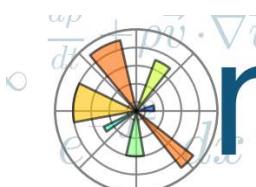


Your
computer

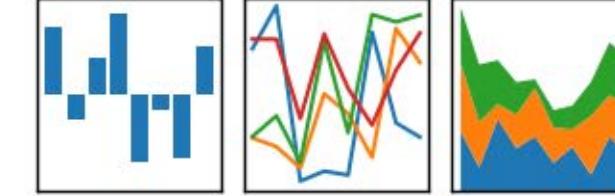

ANACONDA
MINICONDA
CONDA



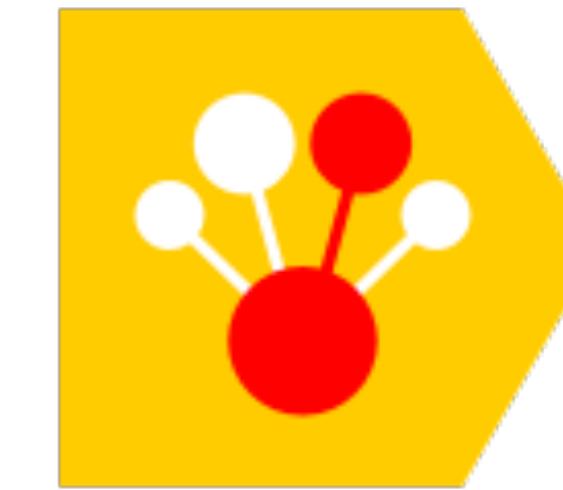
NumPy

 **matplotlib**

pandas




 scikit
learn



dmlc
XGBoost





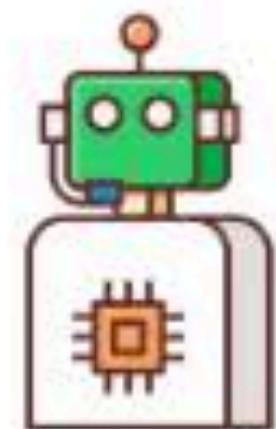
ANACONDA®



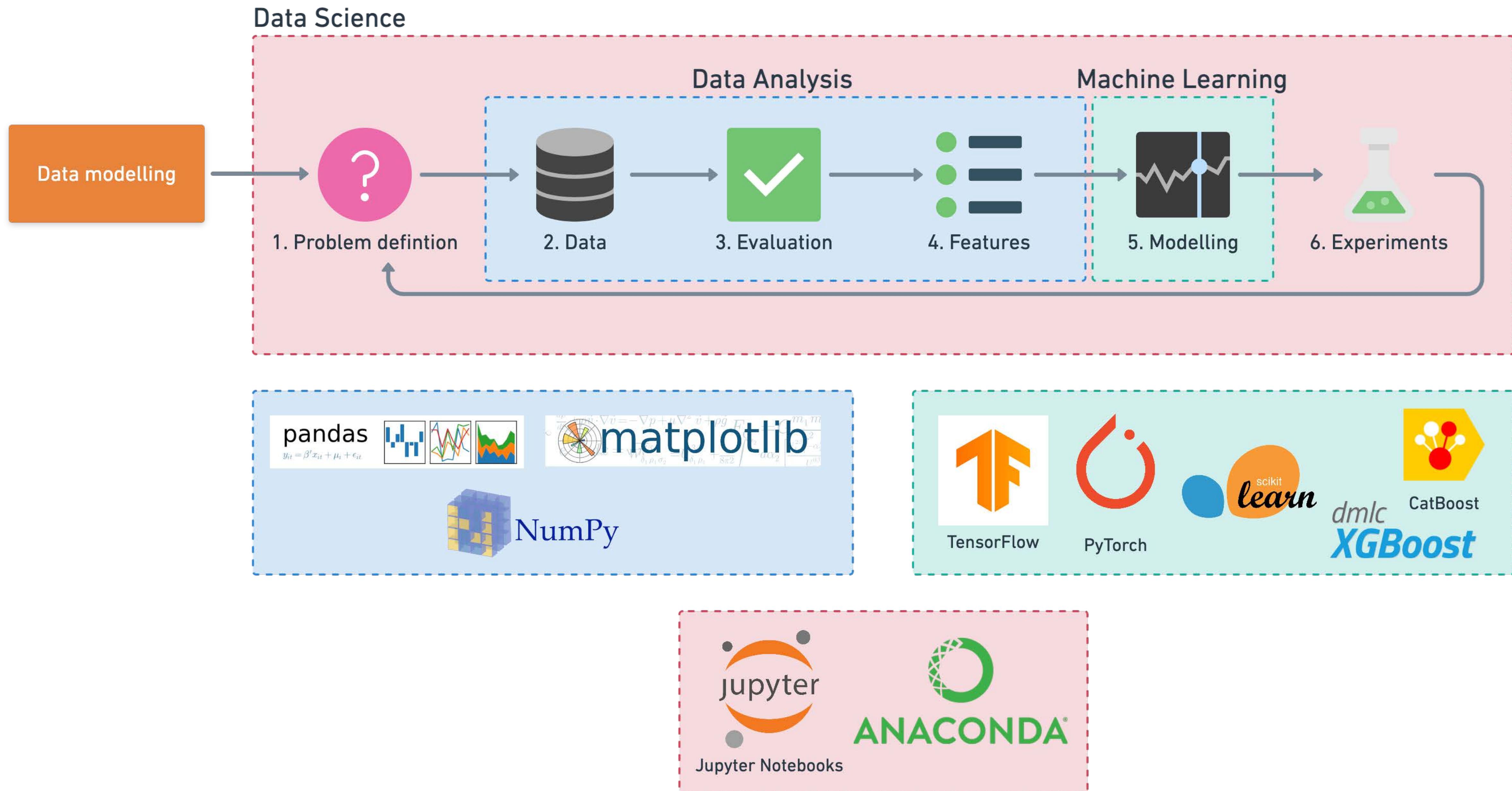
MINICONDA®



CONDA®



Tools you can use





ANACONDA®



Software Distributions

MINICONDA®



CONDA®



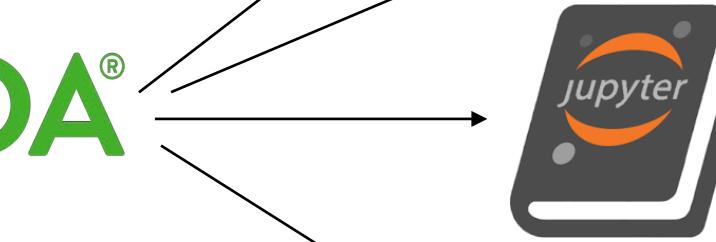
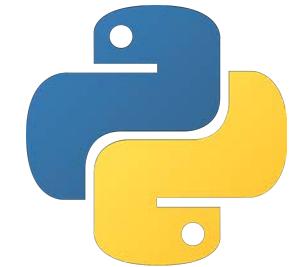
Package Manager



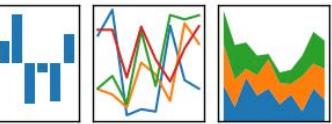
Your
computer

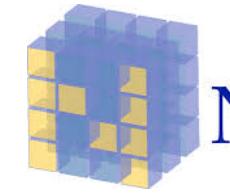


MINICONDA® + CONDA®

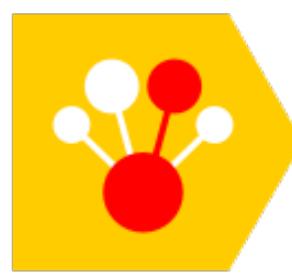


 **matplotlib**
$$\frac{m_1 m}{U_{\infty}^{\alpha_2}} \int_{\delta_1 \rho_1 \sigma_2}^{\delta_1 \rho_1} \int_{\delta_1 \rho_1}^{\delta_2 \rho_2} \frac{d\alpha_2}{d\alpha_1} \left[\frac{1}{U_{\infty}^{\alpha_2 - \alpha_1}} \right]$$

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

 **NumPy**

 **scikit-learn**



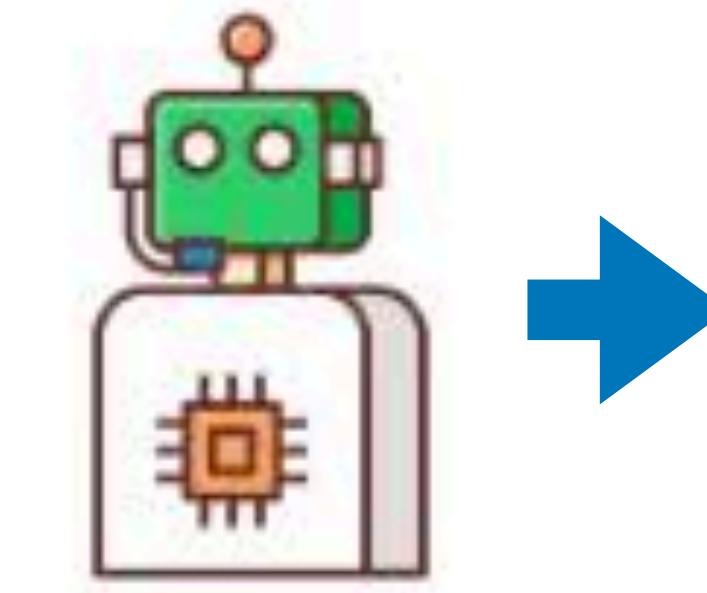
dmlc
XGBoost



Up next

Project 1

CONDA®

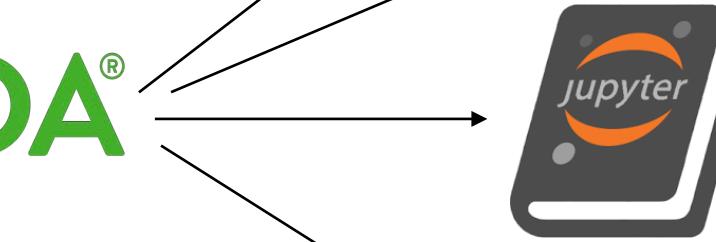
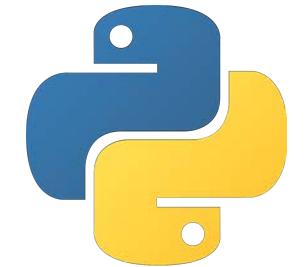




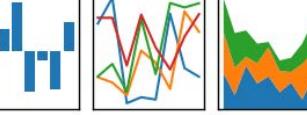
Your
computer

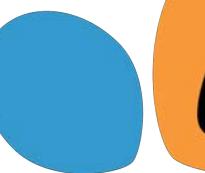


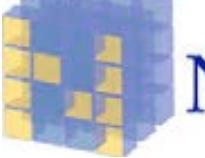
MINICONDA® + CONDA®

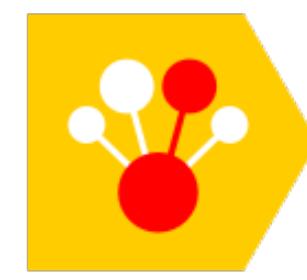


 **matplotlib**
$$\frac{m_1 m}{U_{\infty}^{\alpha_2}} \int_{\delta_1 \rho_1 \sigma_2}^{\delta_1 \rho_1} \int_{\delta_1 \rho_1}^{\delta_2 \rho_2} \frac{d\alpha_2}{d\alpha_1} \left[\frac{1}{U_{\infty}^{\alpha_2 - \alpha_1}} \right]$$

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

 **scikit-learn**

 **NumPy**

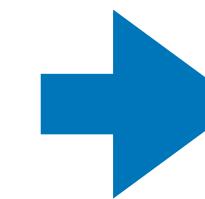


dmlc
XGBoost





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



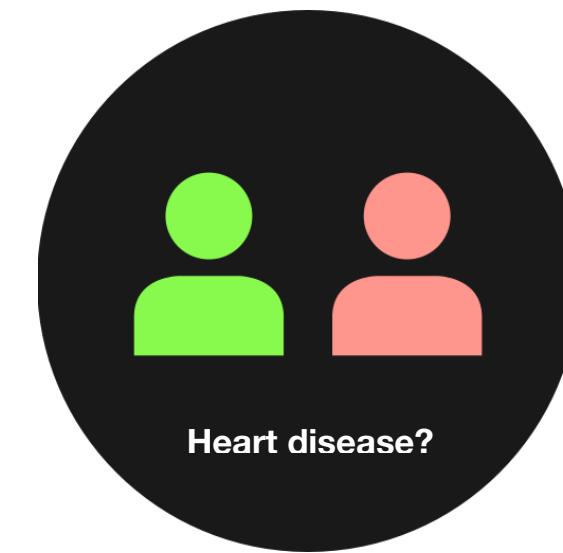
NumPy

pandas

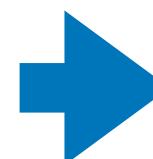


scikit
learn

Environment



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



NumPy

pandas



scikit
learn

Environment



Someone else's
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



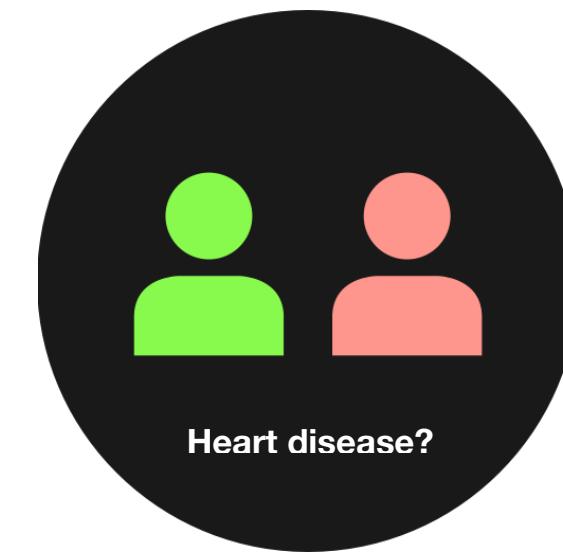
NumPy

pandas



scikit
learn

Environment



Your
computer

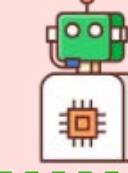
Project Folder

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

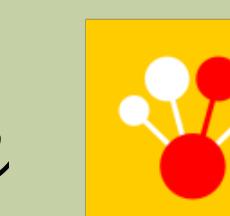


NumPy

pandas



scikit
learn



Environment

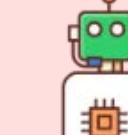
Project Folder

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®



matplotlib



NumPy

pandas



scikit
learn

Workspace

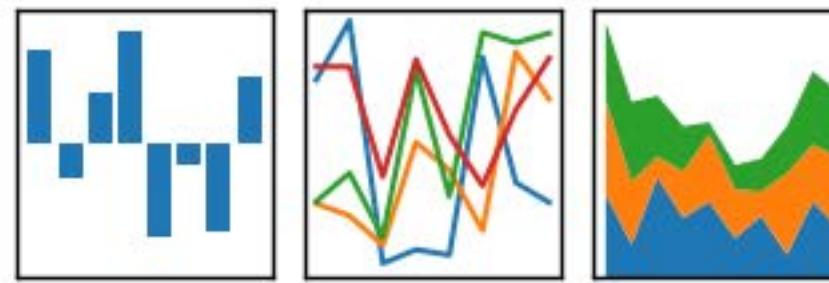
Environment



What is pandas?

pandas

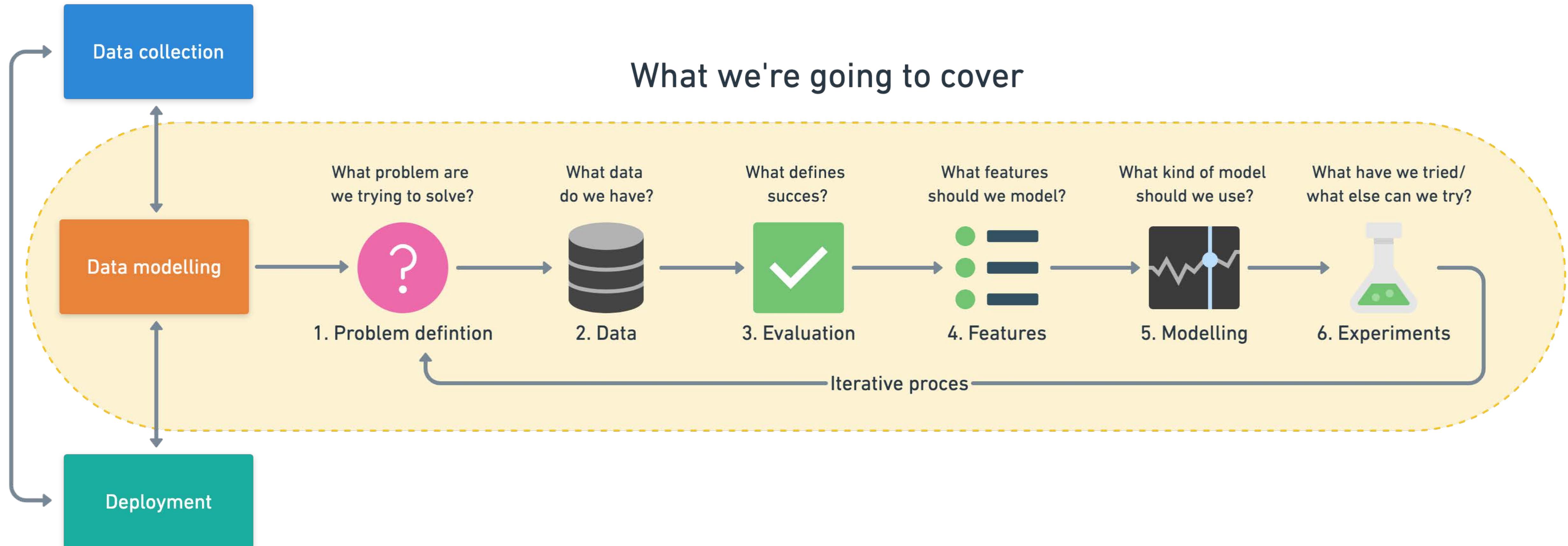
$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



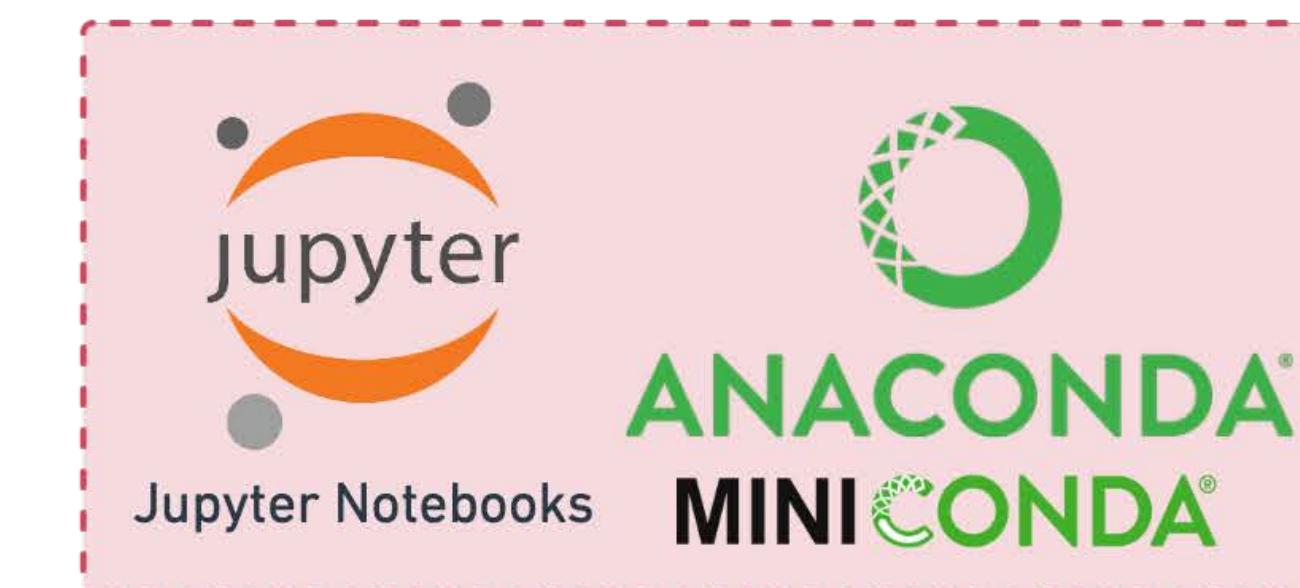
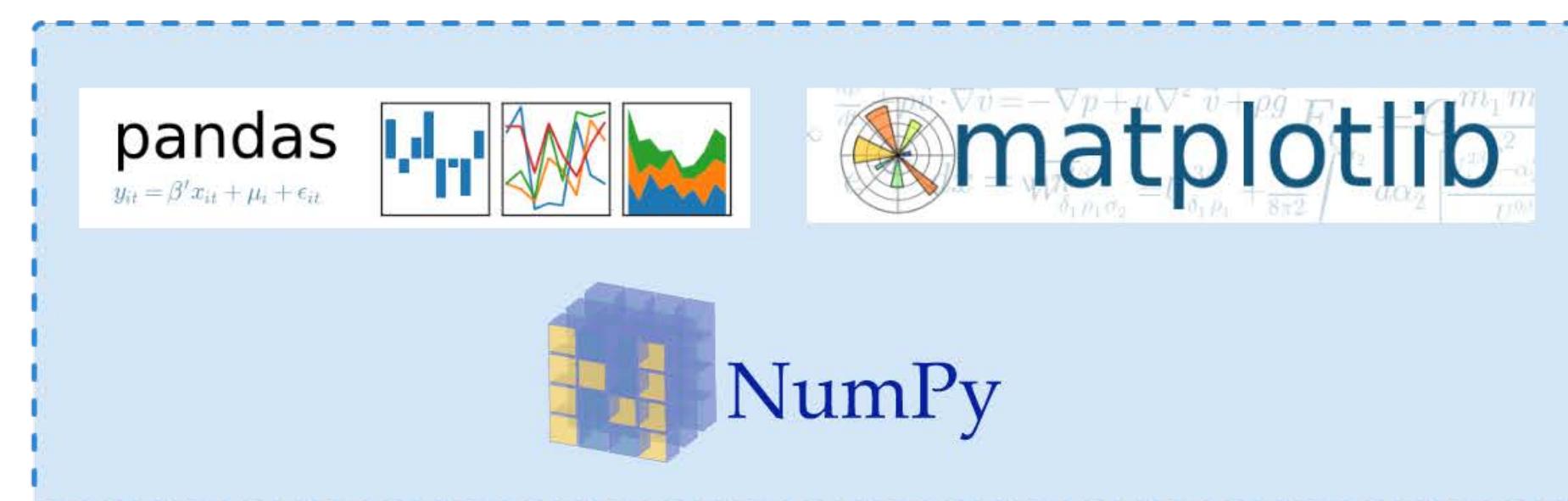
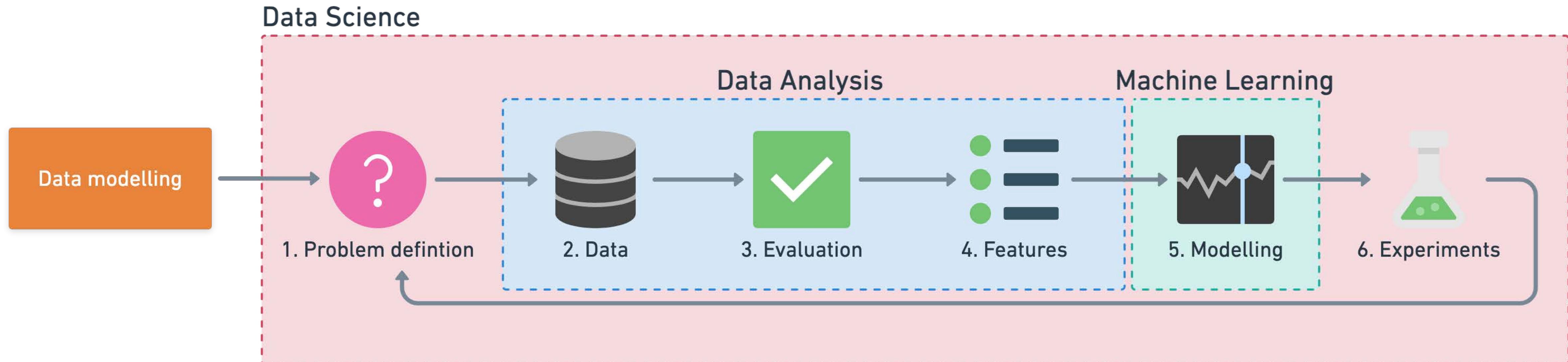
Data



Steps in a full machine learning project



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 notebook content shows a section titled "1. Datatypes" with a note about pandas having two main datatypes, `Series` and `DataFrame`. It defines `Series` as a 1-dimensional column of data and `DataFrame` as a 2-dimensional table of data with rows and columns. Below this, a code cell contains:
In [6]: `# Creating a series of car types
cars = pd.Series(["BMW", "Toyota", "Honda"])`
The output, Out[6], shows:
Out[6]:
0 BMW
1 Toyota
2 Honda
dtype: object

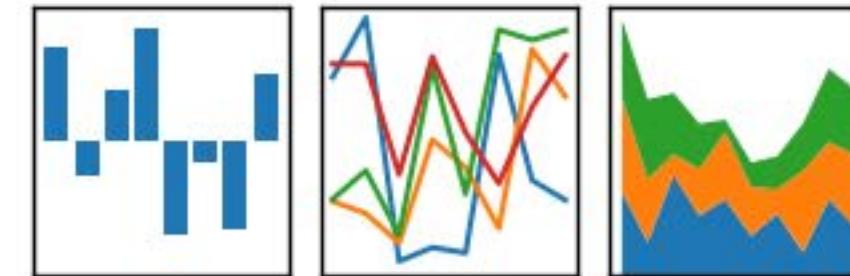


A screenshot of the Pandas documentation website at pandas.pydata.org/pandas-docs/stable/. The header says "pandas 0.25.2 documentation". The main content area features a large blue banner with the text "pandas: powerful Python data analysis toolkit". To the left is a sidebar with a "Table Of Contents" section containing links to "What's New in 0.25.2", "Installation", "Getting started", "User Guide", "Pandas ecosystem", "API reference", "Development", and "Release Notes". Below the sidebar is a "Search" input field. The main content area also includes a "Date: Oct 19, 2019 Version: 0.25.2", download links for "PDF Version" and "Zipped HTML", and useful links for "Binary Installers", "Source Repository", "Issues & Ideas", "Q&A Support", and "Mailing List". A brief description of Pandas as an open source library follows, along with a link to the "Package overview".

Let's code!

pandas

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



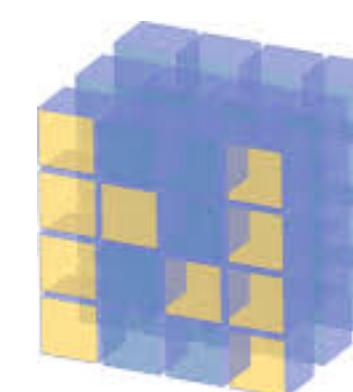
Anatomy of a DataFrame

The diagram illustrates the anatomy of a DataFrame with the following components:

- Column (axis = 1):** Labels the columns: Make, Colour, Odometer, Doors, and Price.
- Index number (starts at 0 by default):** Labels the rows with index numbers 0, 1, 2, 3, and 4.
- Row (axis = 0):** Labels the rows as individual data entries.
- Data:** Points to the value "150043" in the Odometer column of the first row.
- Column name:** Points to the header "Price" in the last column.

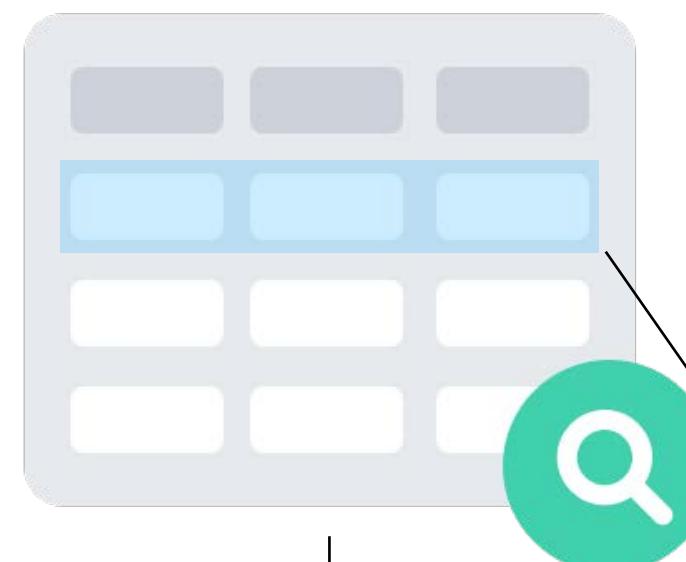
	Column (axis = 1)	Make	Colour	Odometer	Doors	Price	Column name
Index number (starts at 0 by default)	0	Toyota	White	150043	4	\$4,000	
Row (axis = 0)	1	Honda	Red	87899	4	\$5,000	
	2	Toyota	Blue	32549	3	\$7,000	
	3	BMW	Black	11179	5	\$22,000	
	4	Nissan	White	213095	4	\$3,500	

What is NumPy?

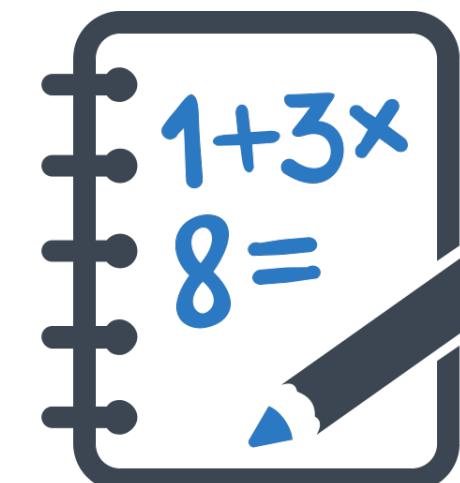


NumPy

Data

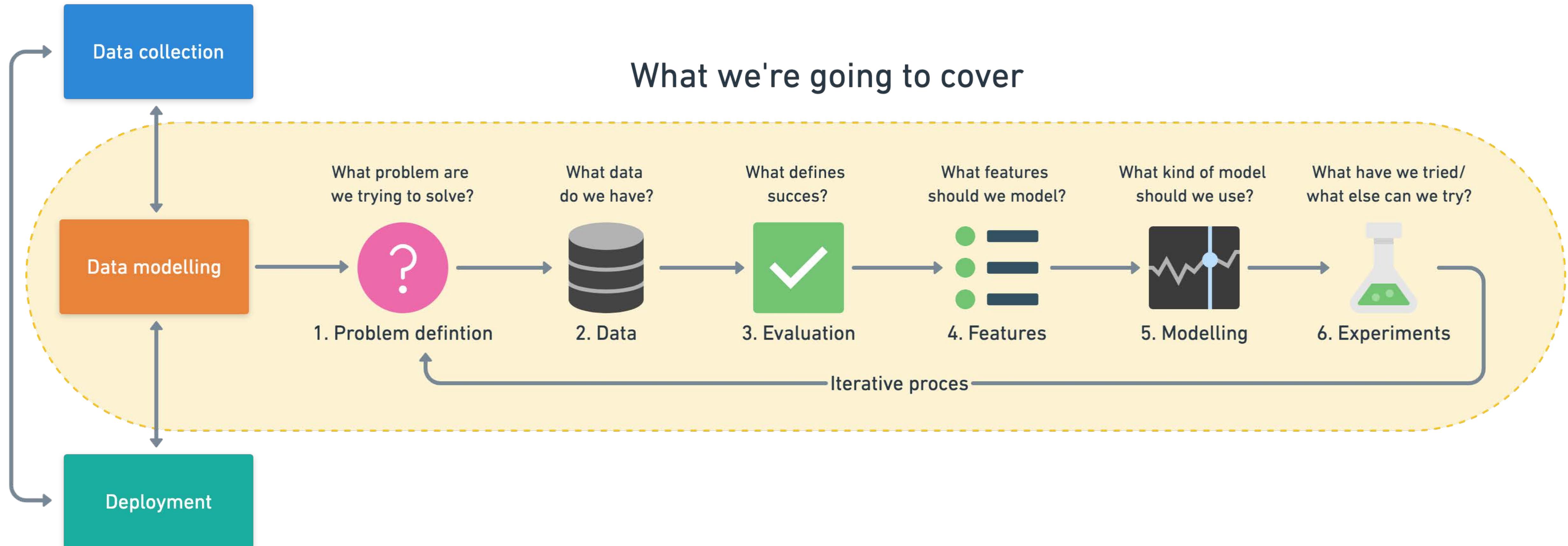


5	0	3
3	7	9
3	5	2

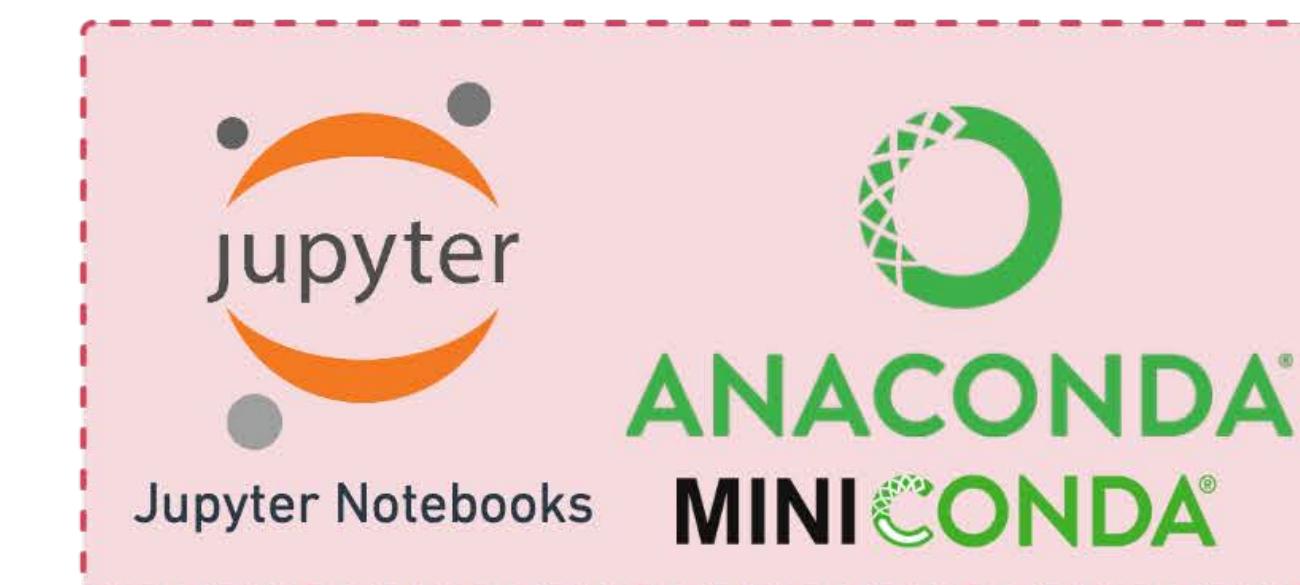
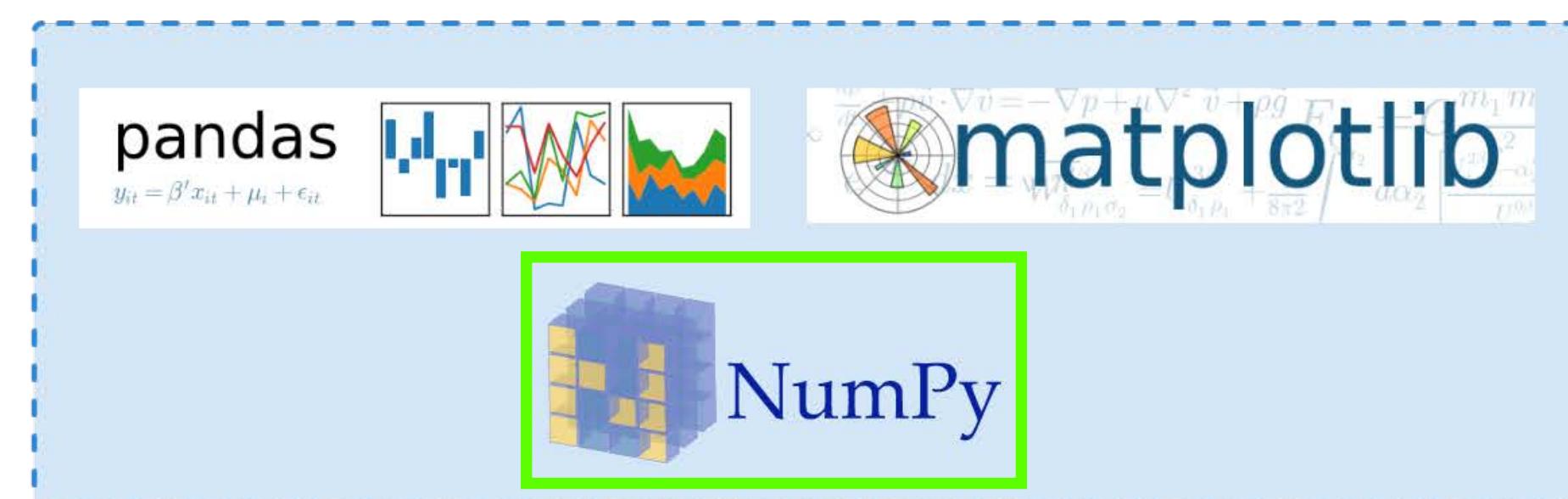
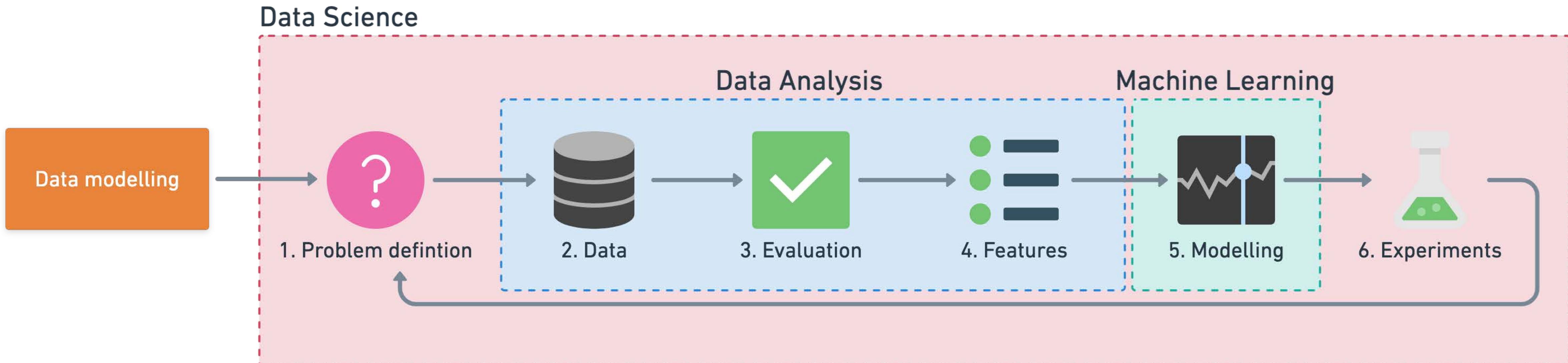


5	0	3
3	7	9
3	5	2

Steps in a full machine learning project



Tools you can use



Why NumPy?

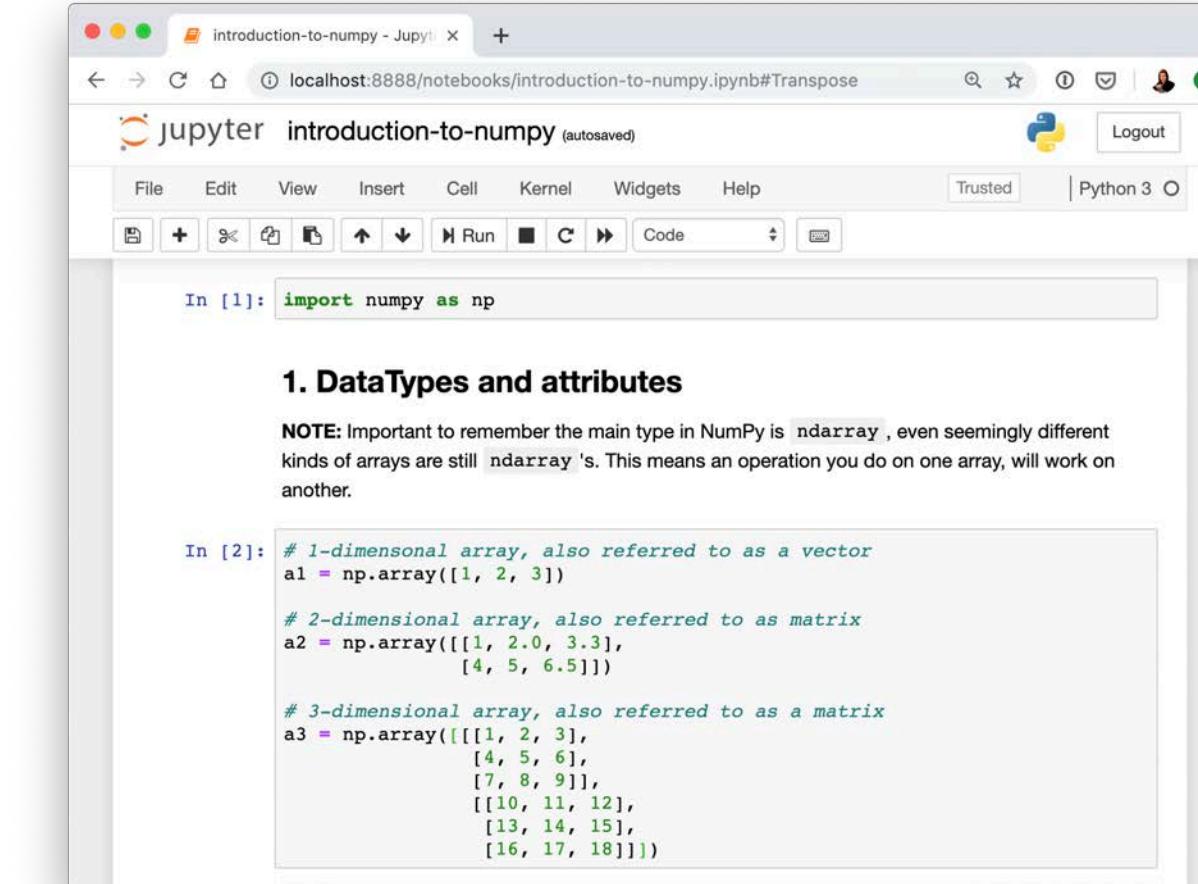
- It's fast
- Behind the scenes optimizations written in C
- Vectorization via broadcasting (avoiding loops)
- Backbone of other Python scientific packages

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.

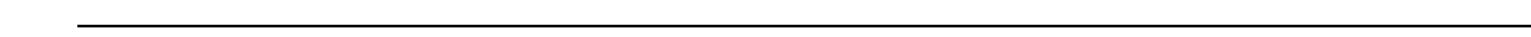
In [2]: `# 1-dimensional array, also referred to as a vector
a1 = np.array([1, 2, 3])

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]]]])`

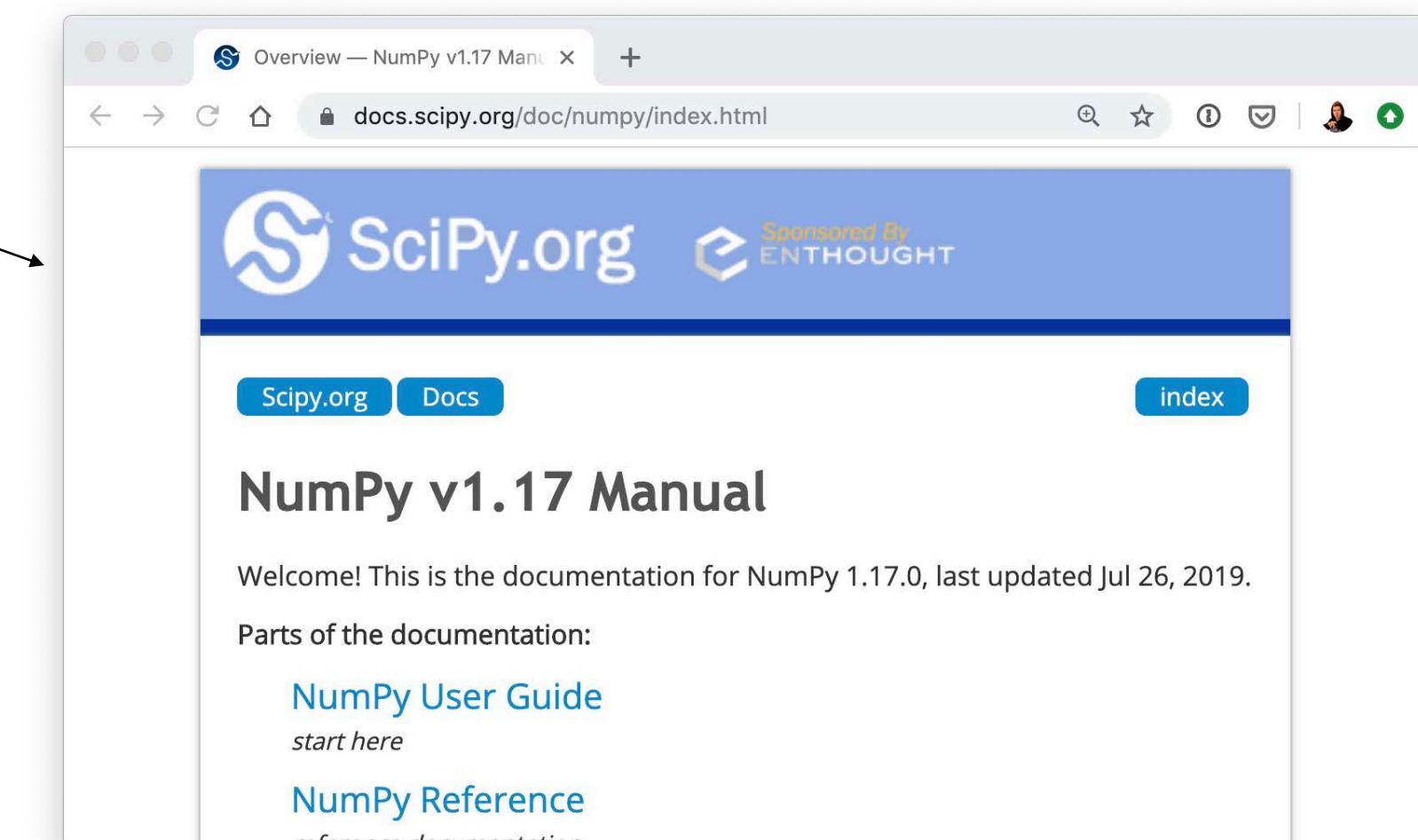
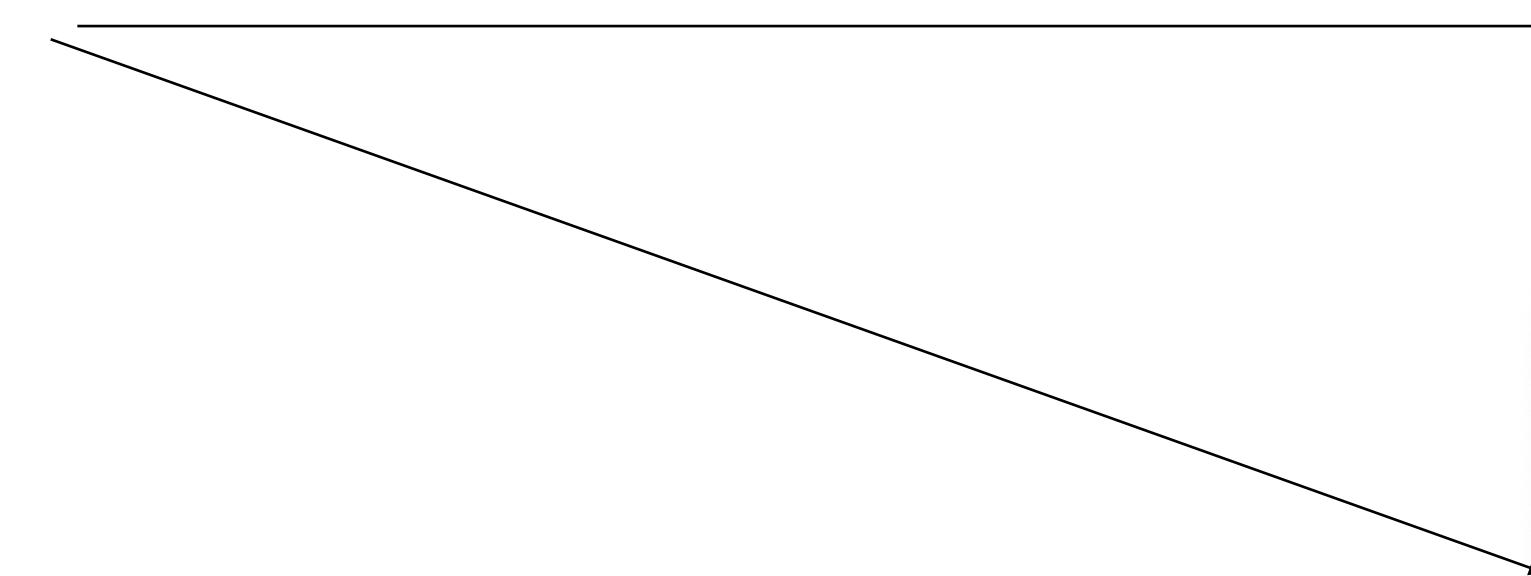
- Try it for yourself

- Search for it

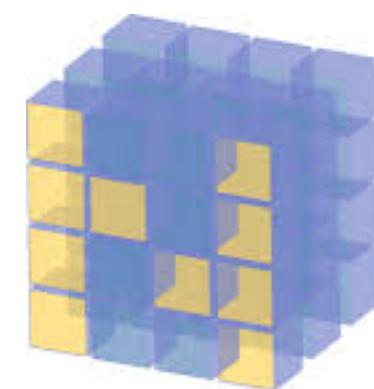


- Try again

- Ask

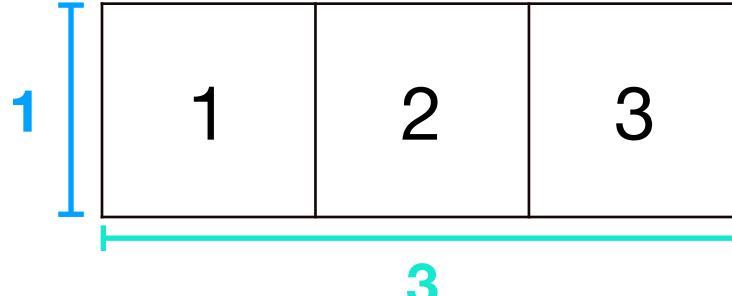
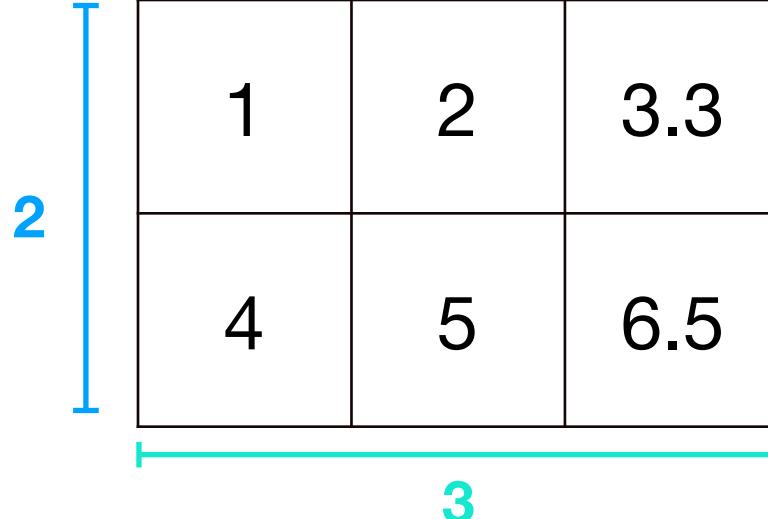
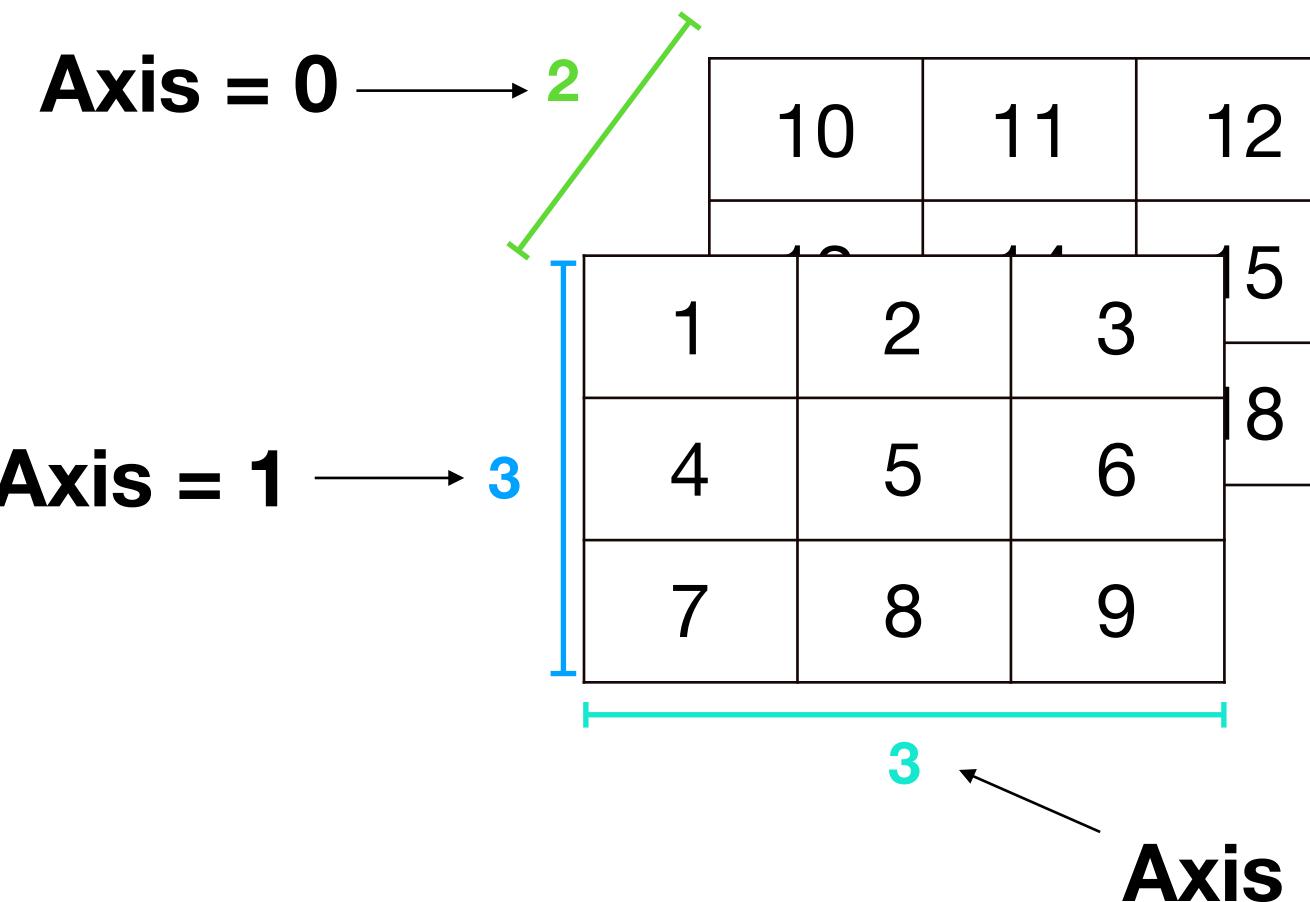


Let's code!



NumPy

Anatomy of a NumPy array

Data	NumPy	Details
	<code>array([1, 2, 3])</code>	<ul style="list-style-type: none">Names: Array, vector1-dimensionalShape = (1, 3)
	<code>array([[1., 2., 3.3], [4., 5., 6.5]])</code>	<ul style="list-style-type: none">Names: Array, matrixMore than 1-dimensionShape = (2, 3)
	<code>array([[[1, 2, 3], [4, 5, 6], [7, 8, 9]], [[10, 11, 12], [13, 14, 15], [16, 17, 18]]])</code>	<ul style="list-style-type: none">Names: Array, matrixMore than 1-dimensionShape = (2, 3, 3)

Dot product vs. element-wise

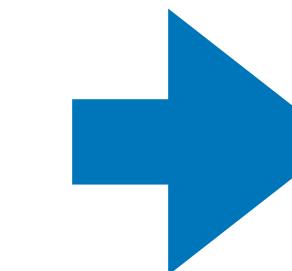
Element-wise

A	B
C	D

2x2

E	F
G	H

2x2



A*E	B*F
C*G	D*H

2x2

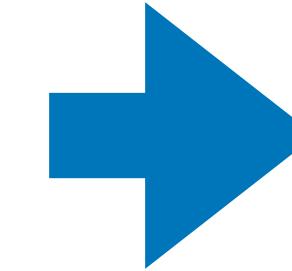
Dot product

A	B	C
D	E	F
G	H	I

3x3

J	K
L	M
N	O

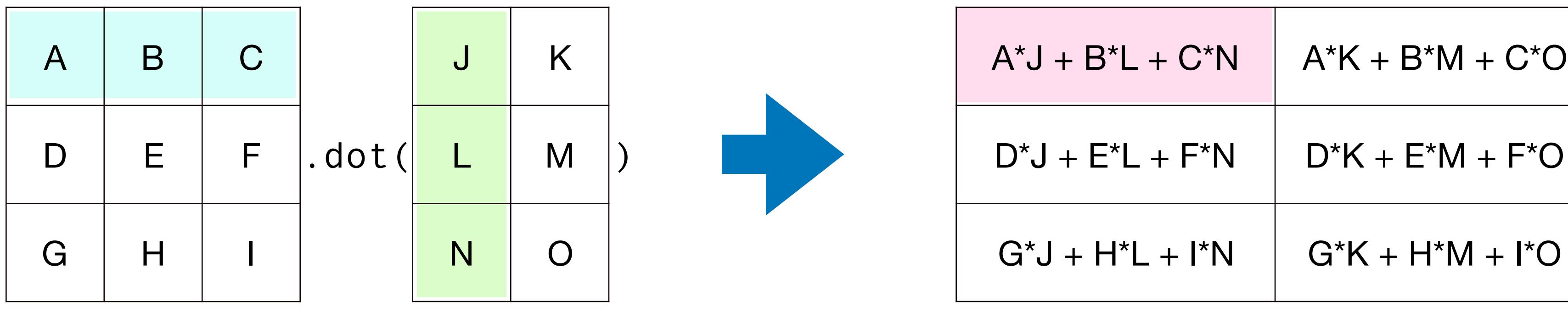
3x2



A*j + B*l + C*n	A*k + B*m + C*o
D*j + E*l + F*n	D*k + E*m + F*o
G*j + H*l + I*n	G*k + H*m + I*o

3x2

Dot product

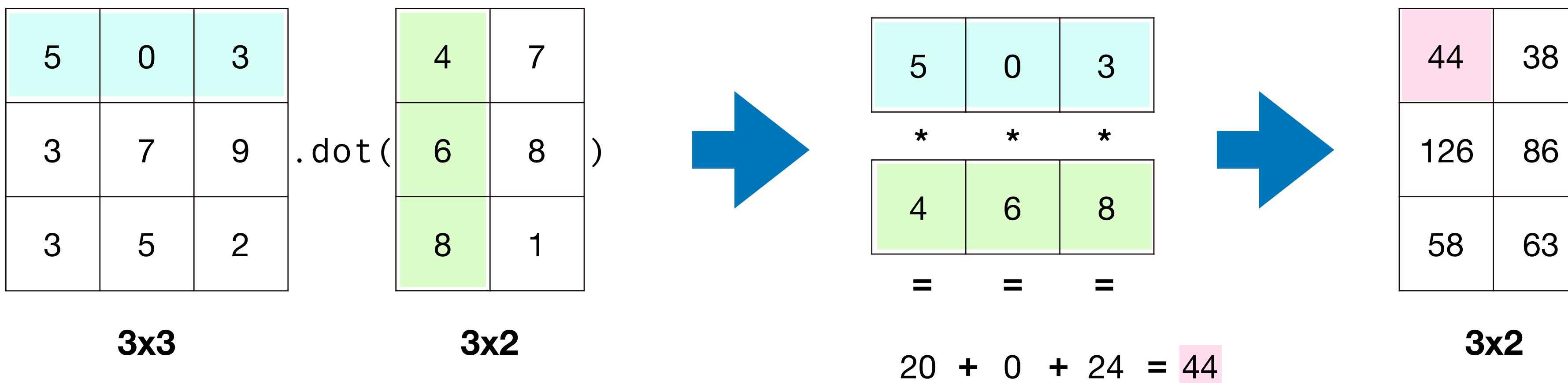


3x3 ← → **3x2**

Numbers on the inside must match

3x2

New size is same as outside numbers

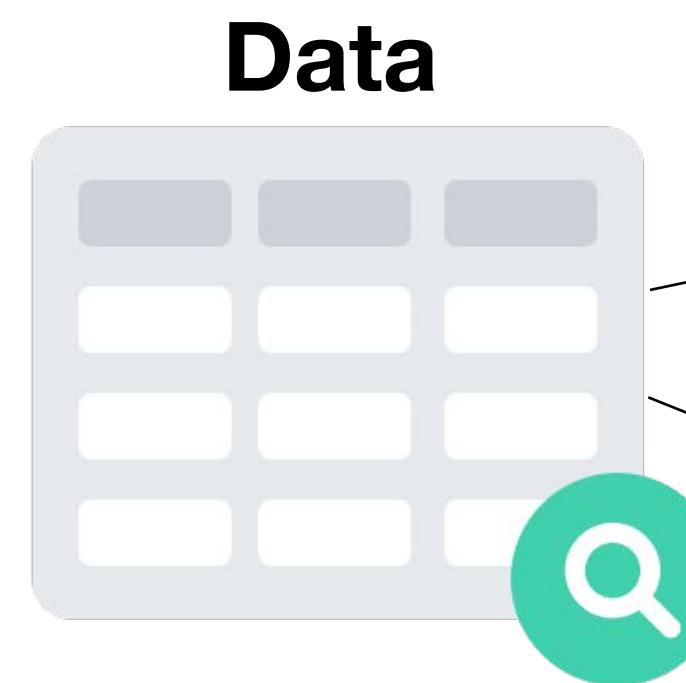


For a live demo, checkout www.matrixmultiplication.xyz

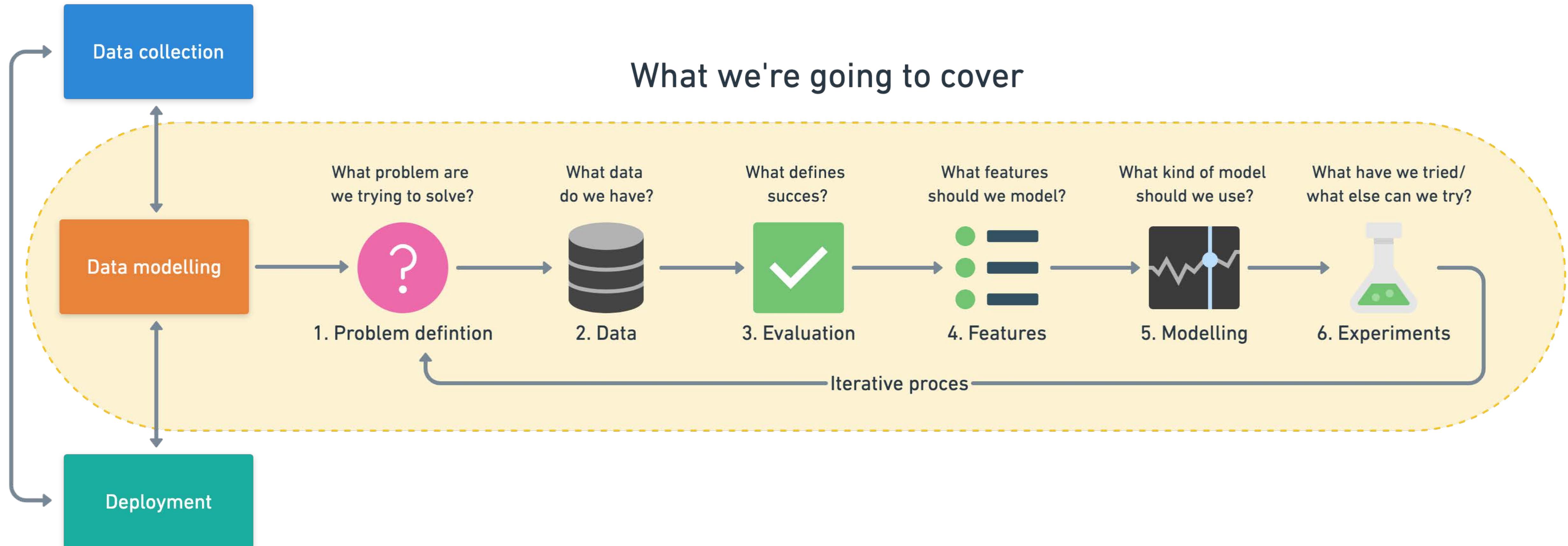
What is Matplotlib?



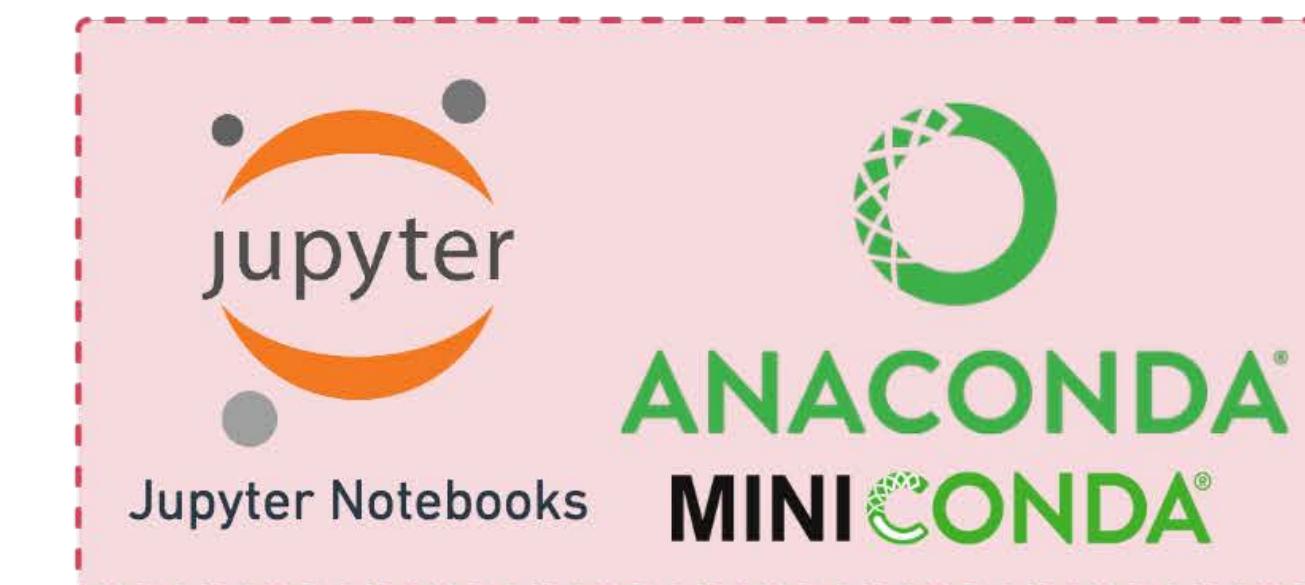
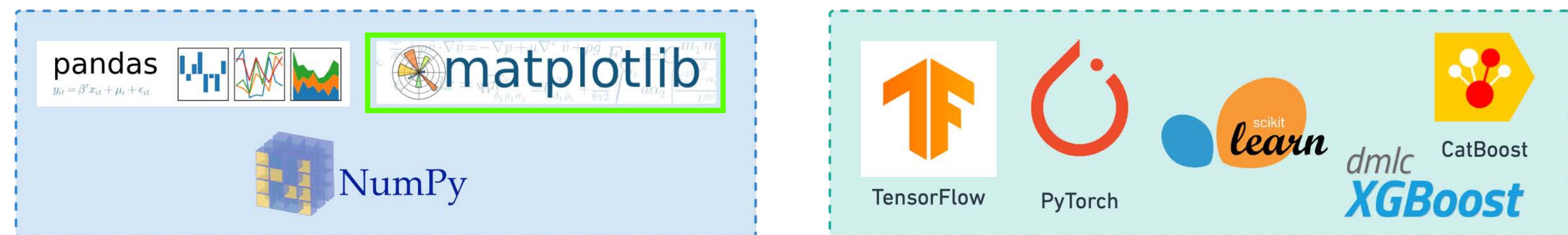
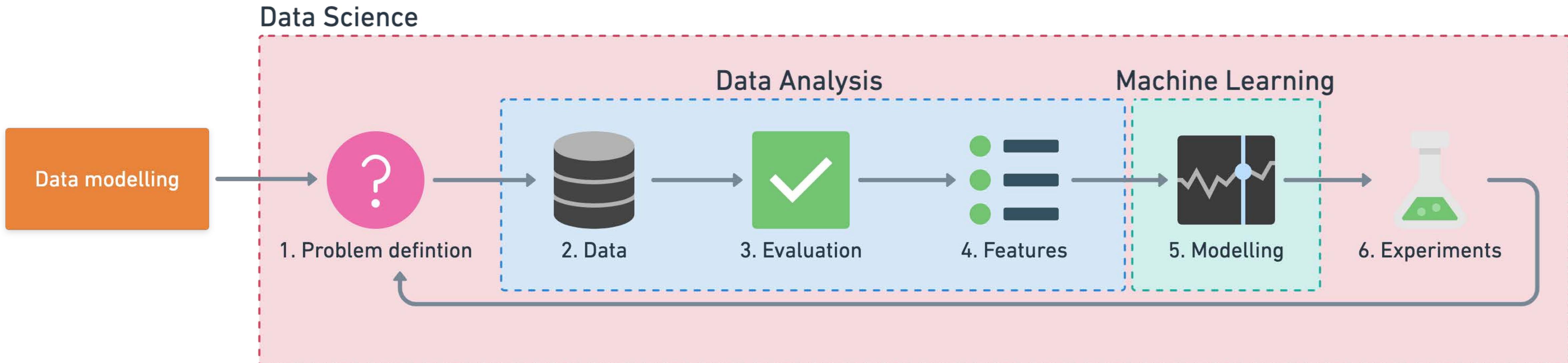
matplotlib



Steps in a full machine learning project



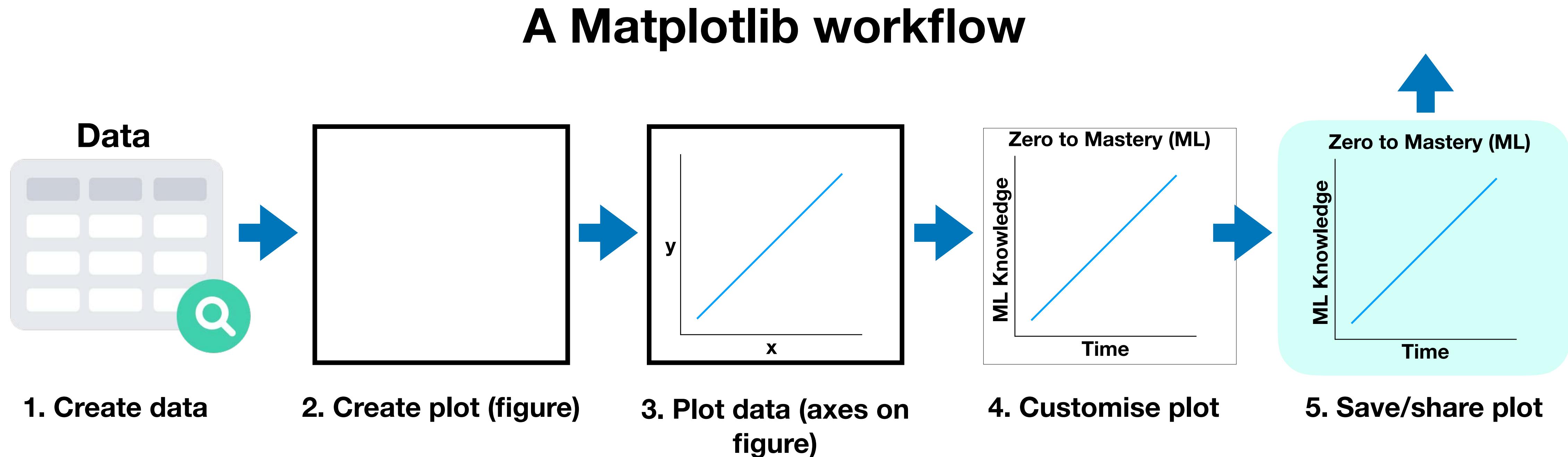
Tools you can use



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?

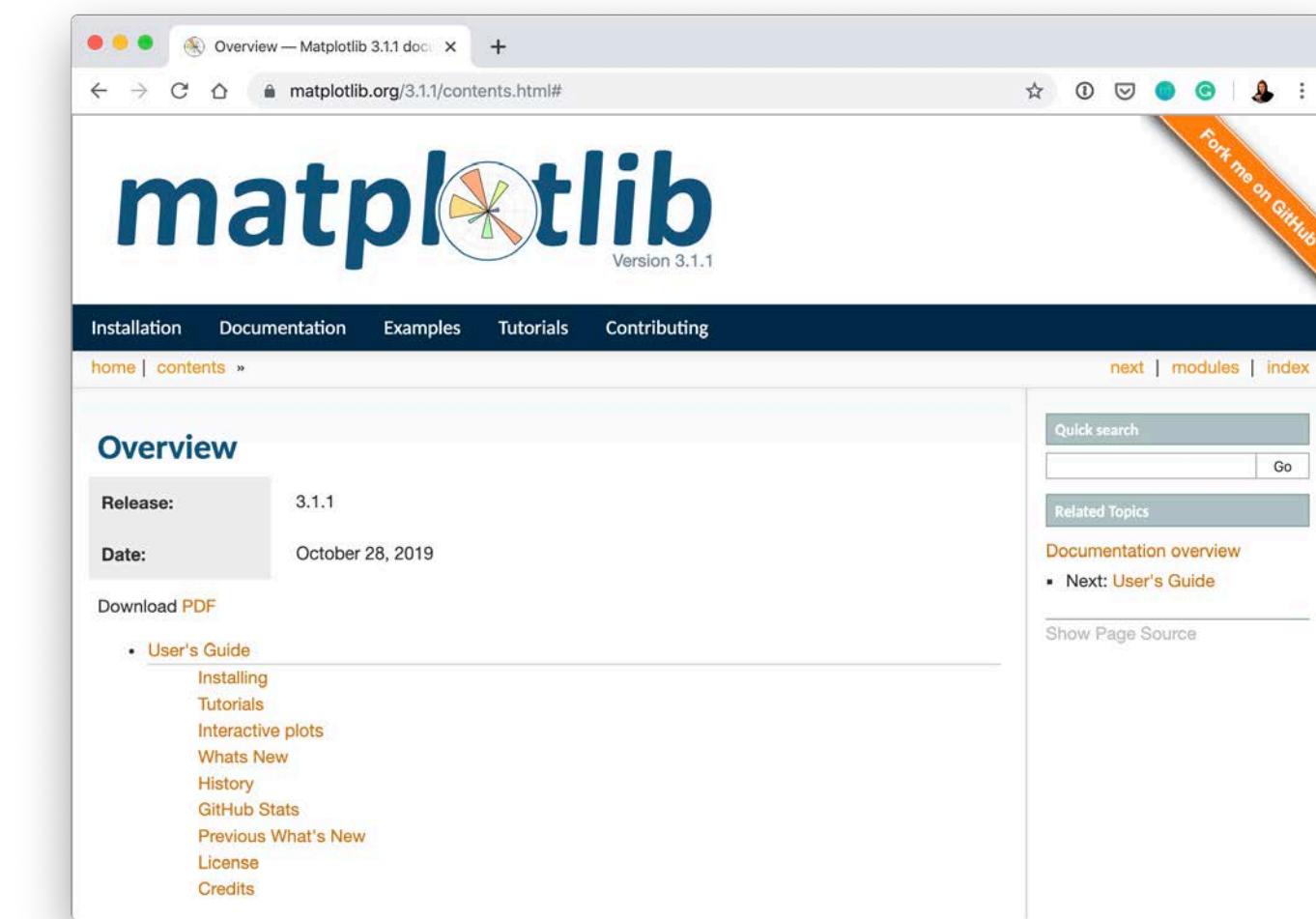
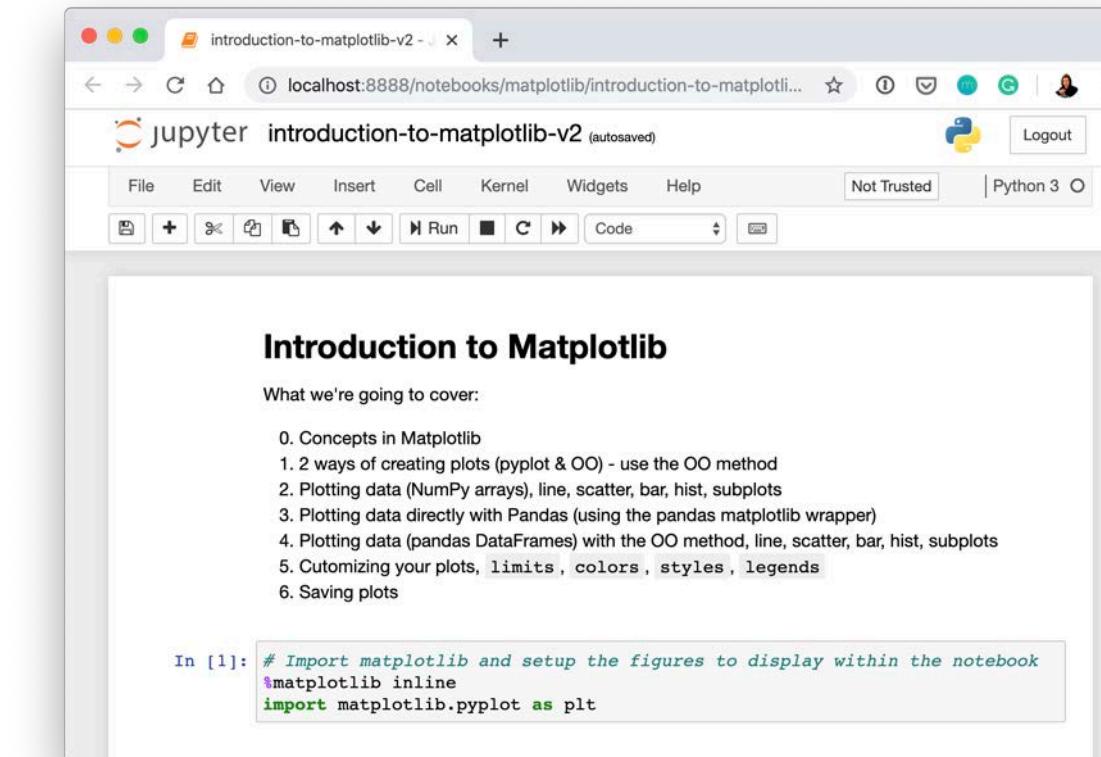


What are we going to cover?

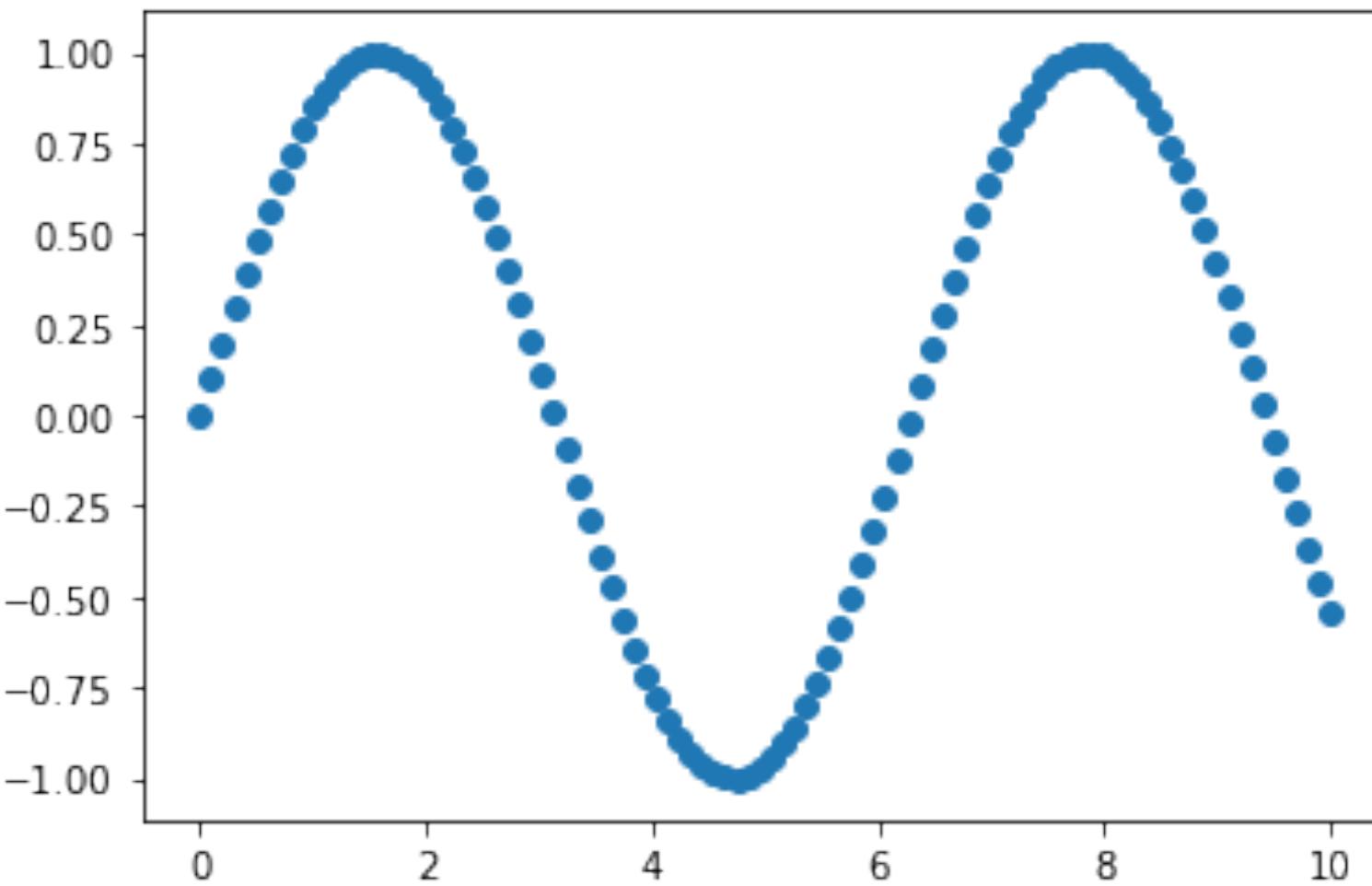
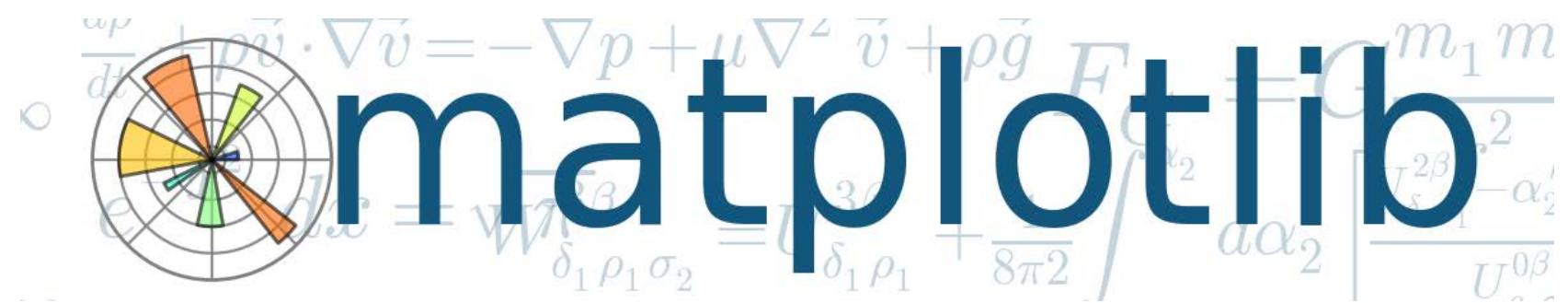
- Matplotlib workflow
- Importing Matplotlib and the 2 ways of plotting
- Plotting data from NumPy arrays
- Plotting data from pandas DataFrames
- Customizing plots
- Saving and sharing plots

Where can you get help?

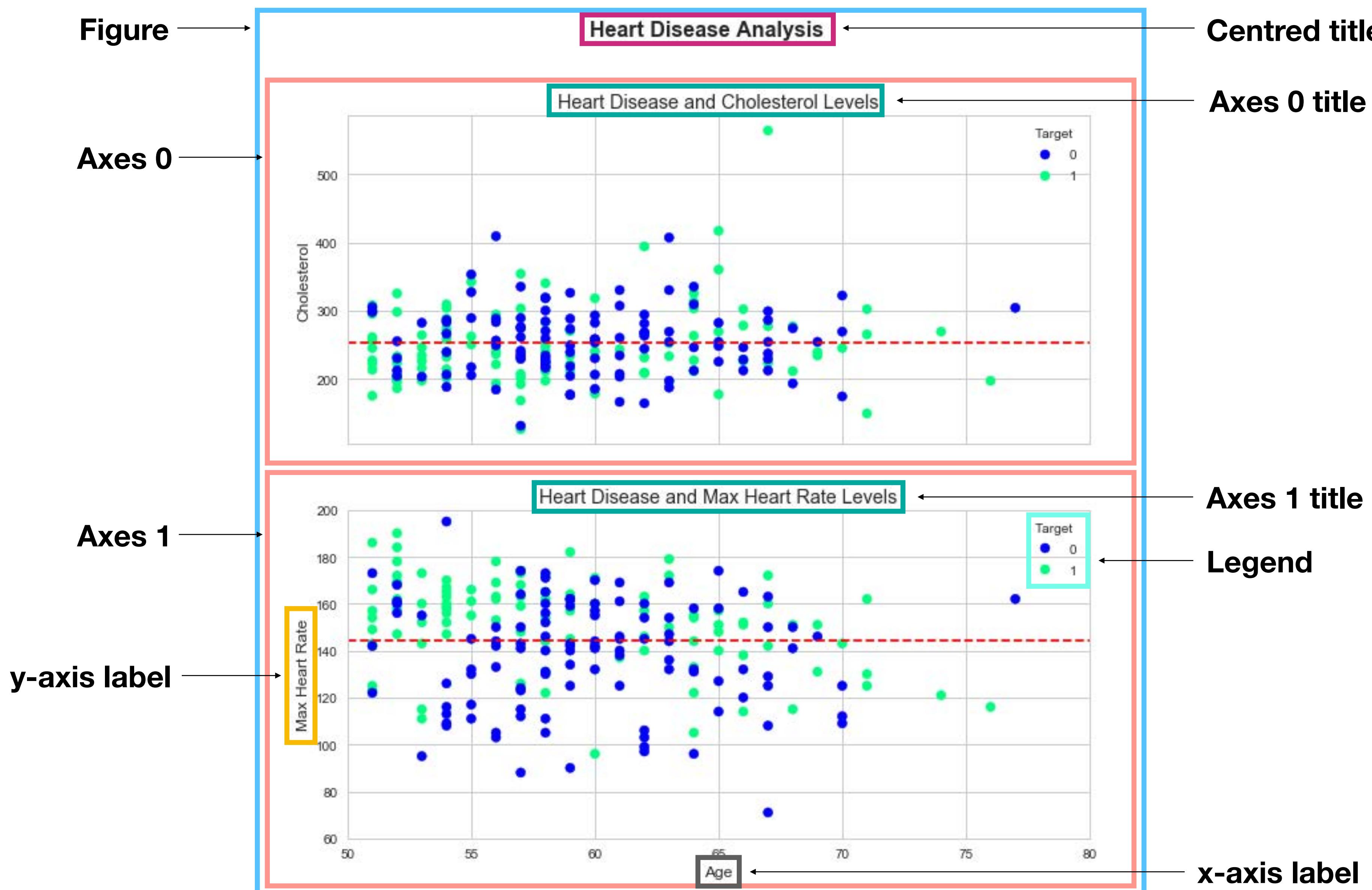
- Follow along with the code
- Try it for yourself
- Search for it
- Try again
- Ask



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

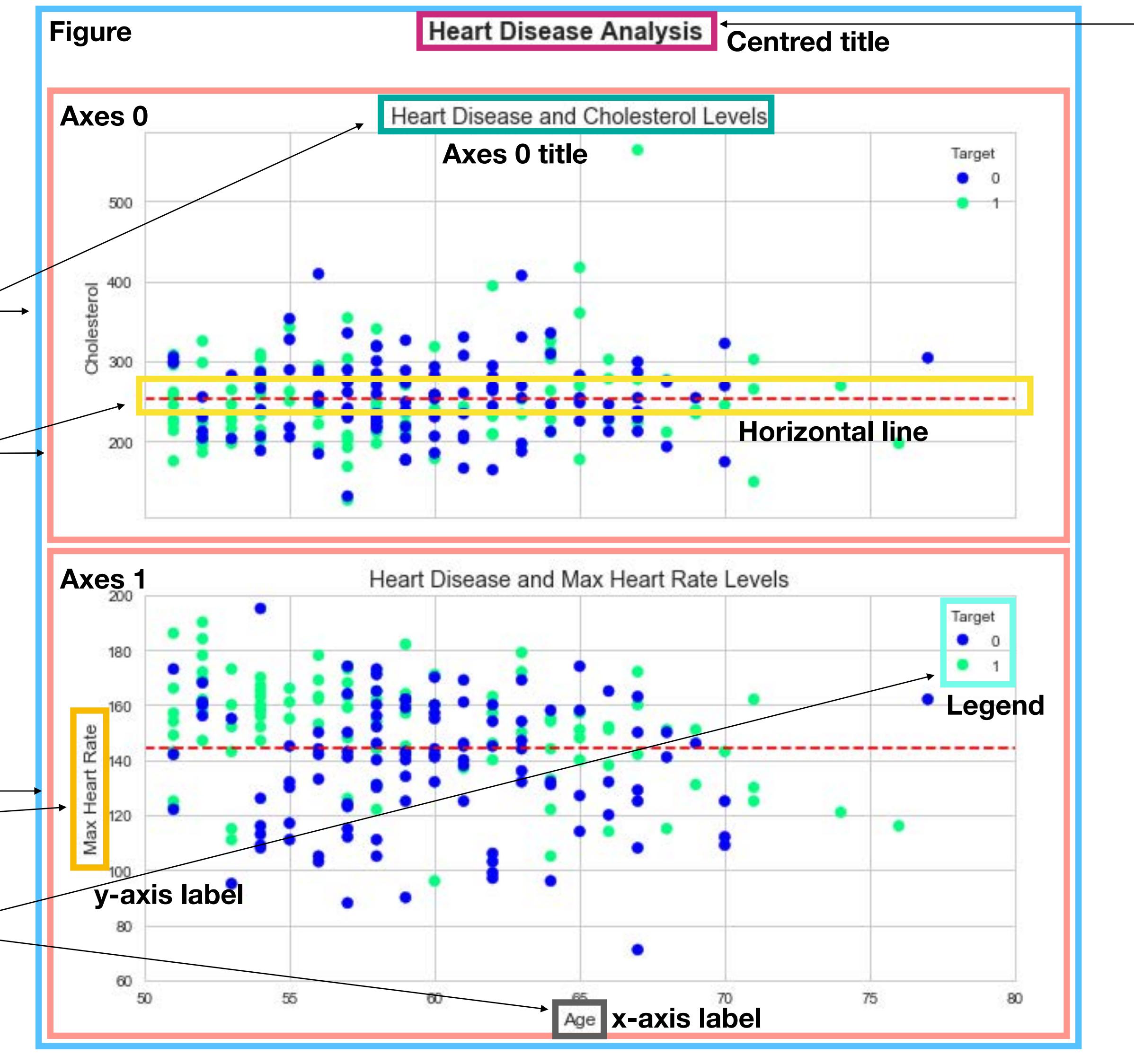
# Read in and manipulate data
heart_disease = pd.read_csv('../data/heart-disease.csv')
over_50 = heart_disease[heart_disease['age'] > 50]

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

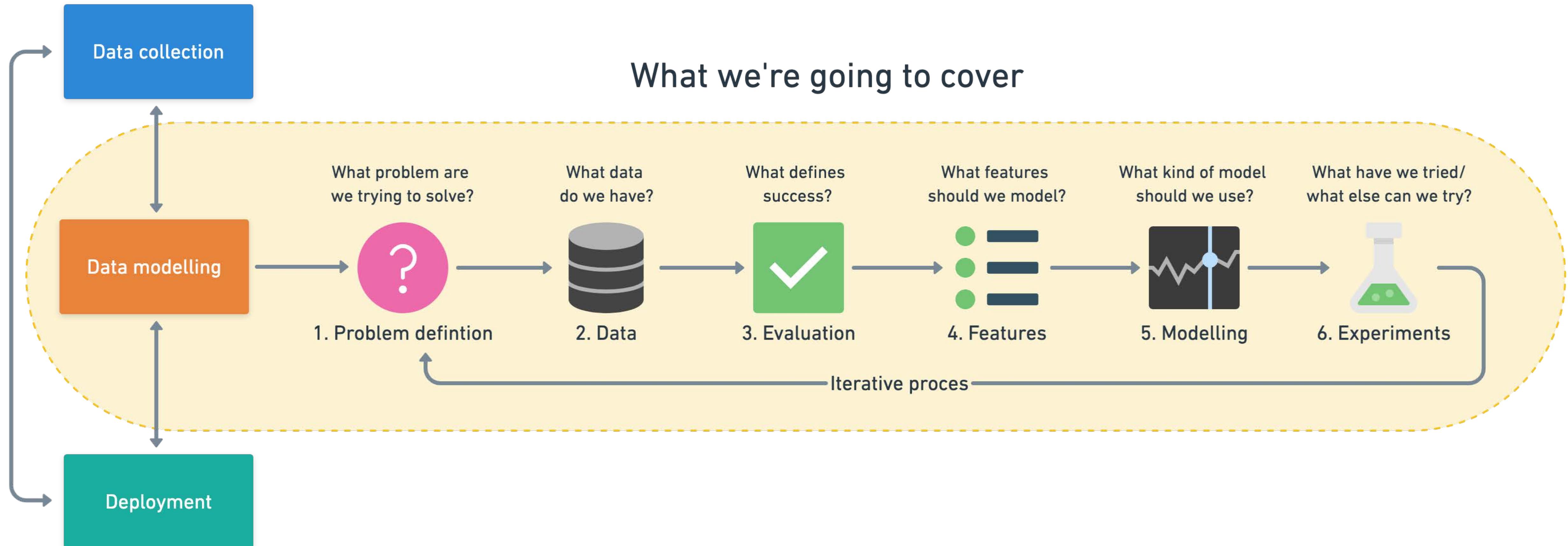
# Add data, titles, meanline (axhline) and legend to axes 0
scatter = ax0.scatter(over_50["age"],
                      over_50["chol"],
                      c=over_50["target"],
                      cmap='winter')
ax0.set(title="Heart Disease and Cholesterol Levels",
        ylabel="Cholesterol",
        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")

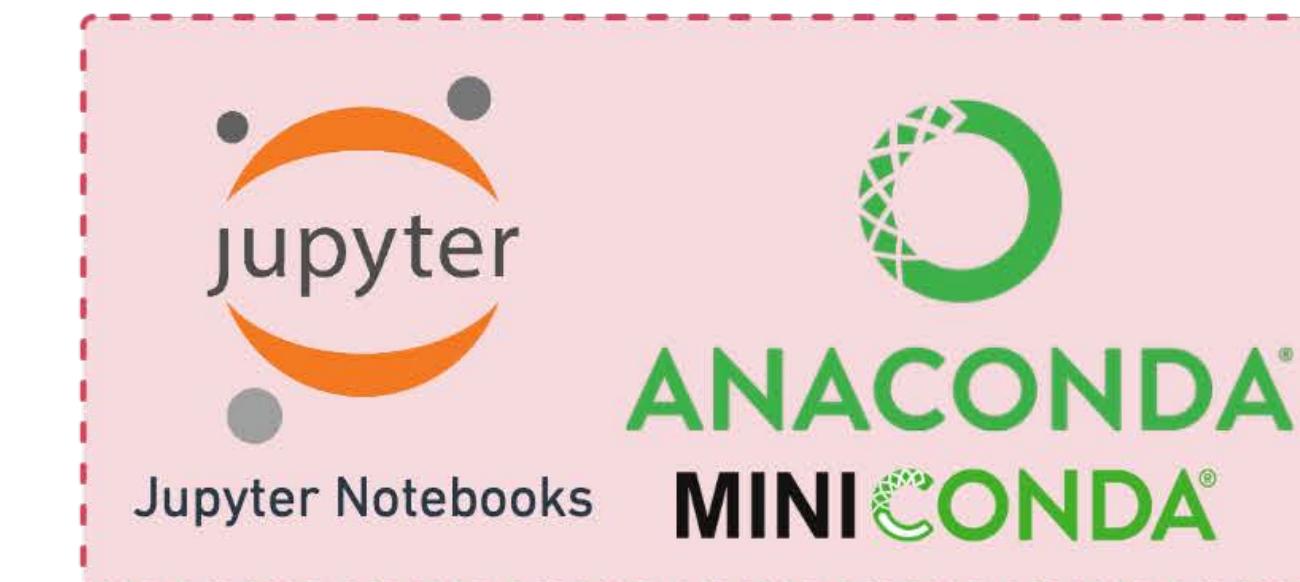
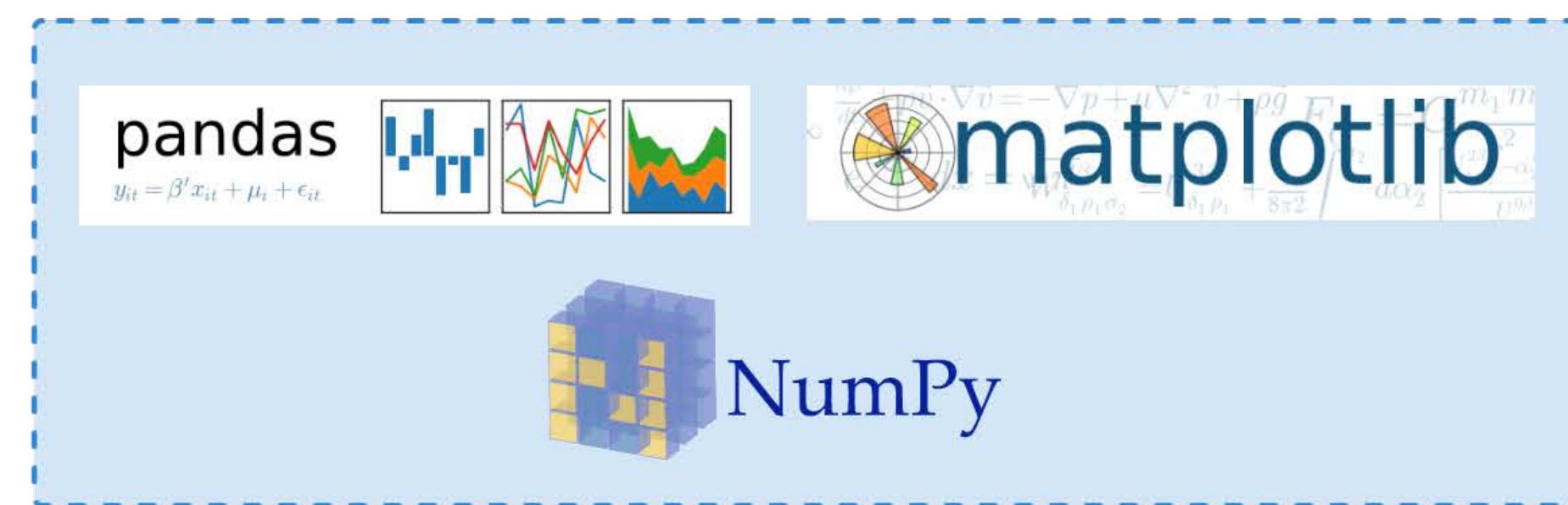
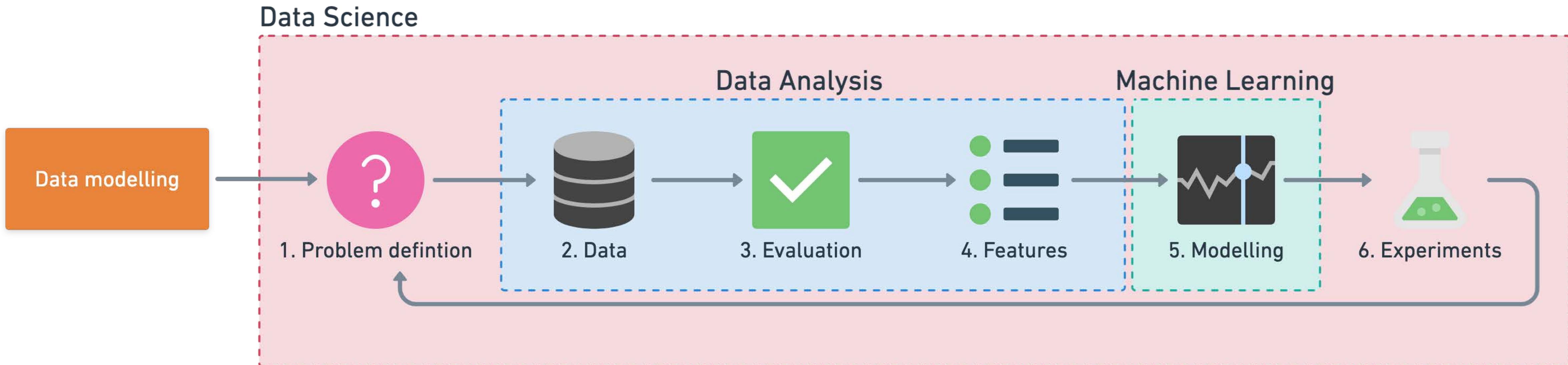
# Title the figure
fig.suptitle('Heart Disease Analysis', fontsize=16, fontweight='bold');
```



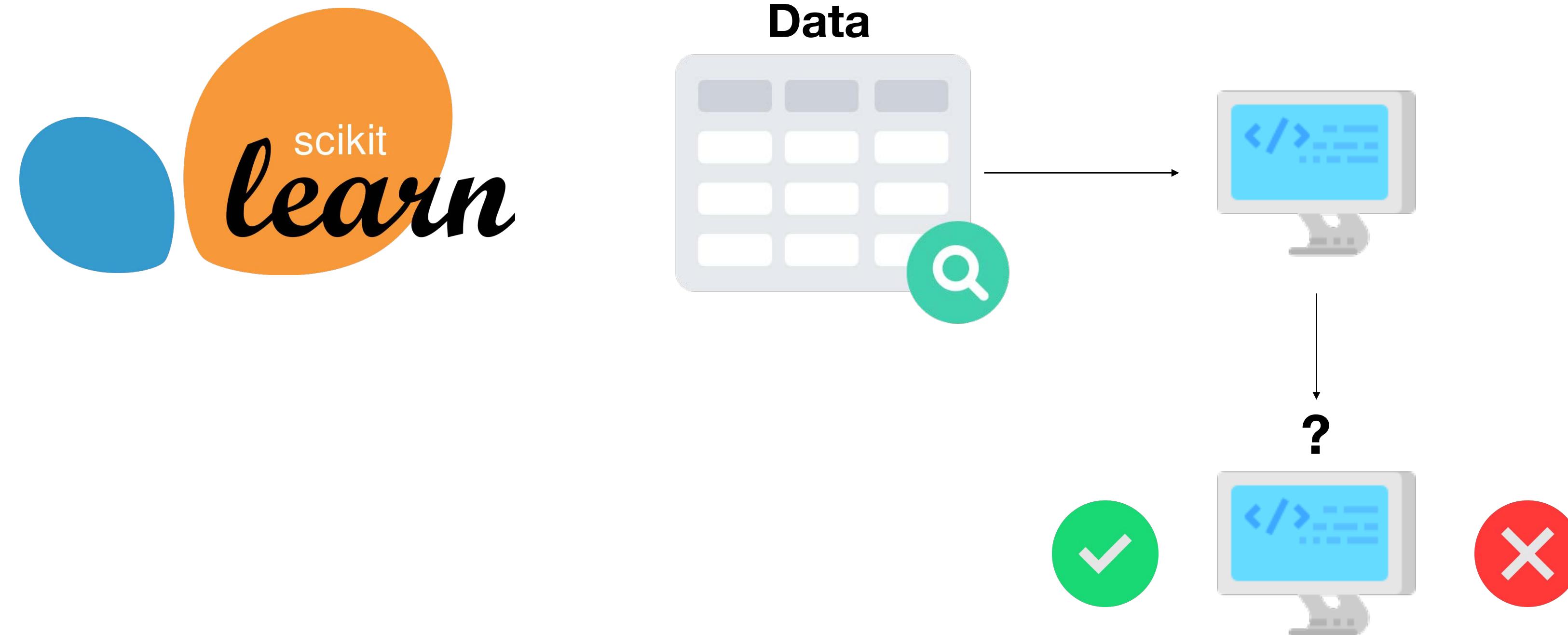
Steps in a full machine learning project



Tools you can use



What is Scikit-Learn (sklearn)?

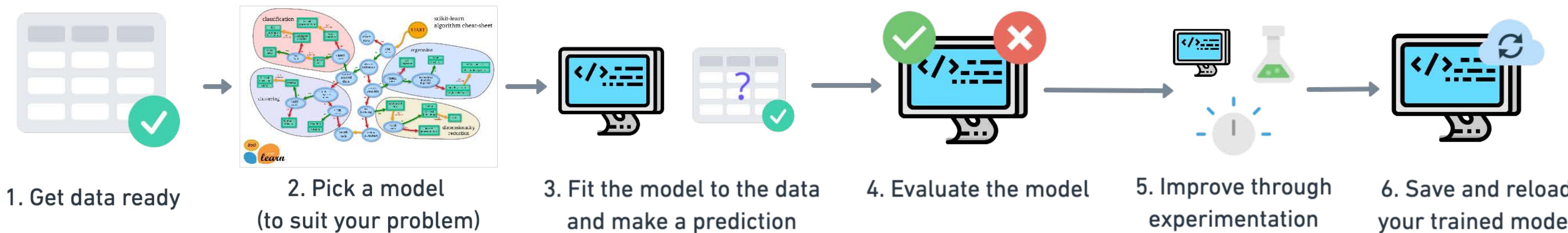


Why Scikit-Learn?

- 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

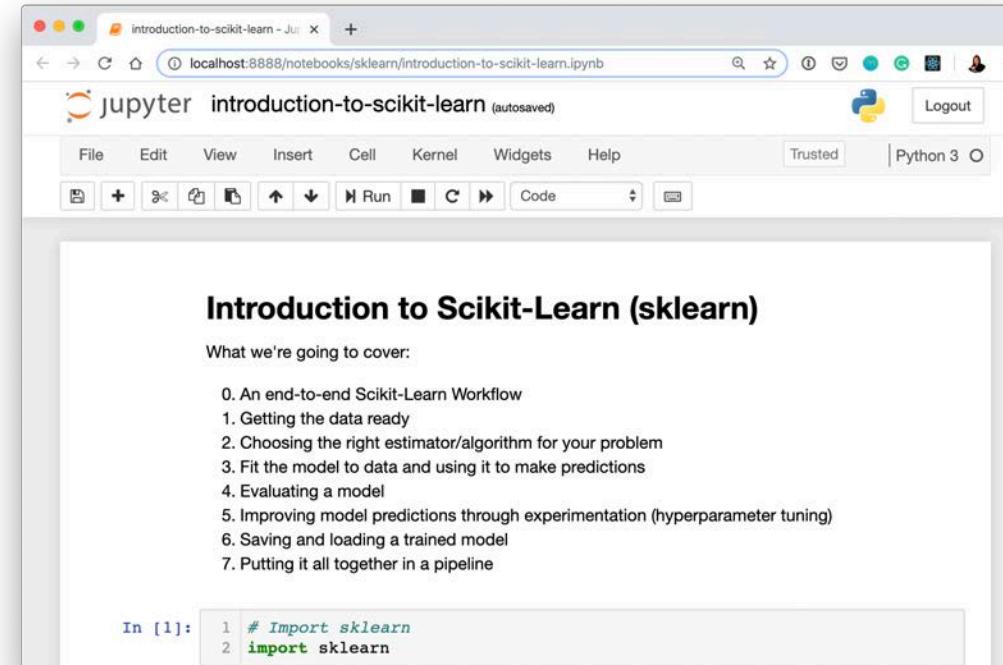


What are we going to cover?

- 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?

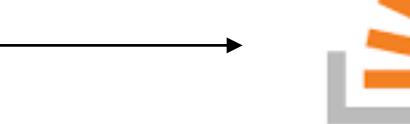
- Follow along with the code



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

- Try it for yourself

- Press SHIFT + TAB to read the docstring

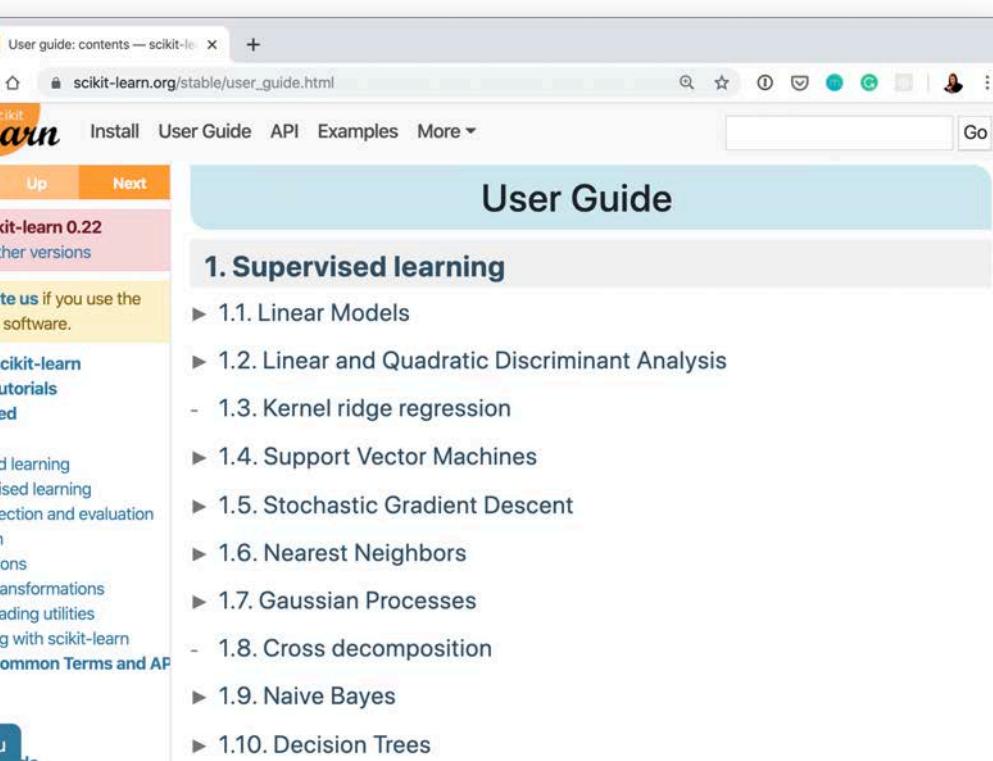


stackoverflow

- Search for it

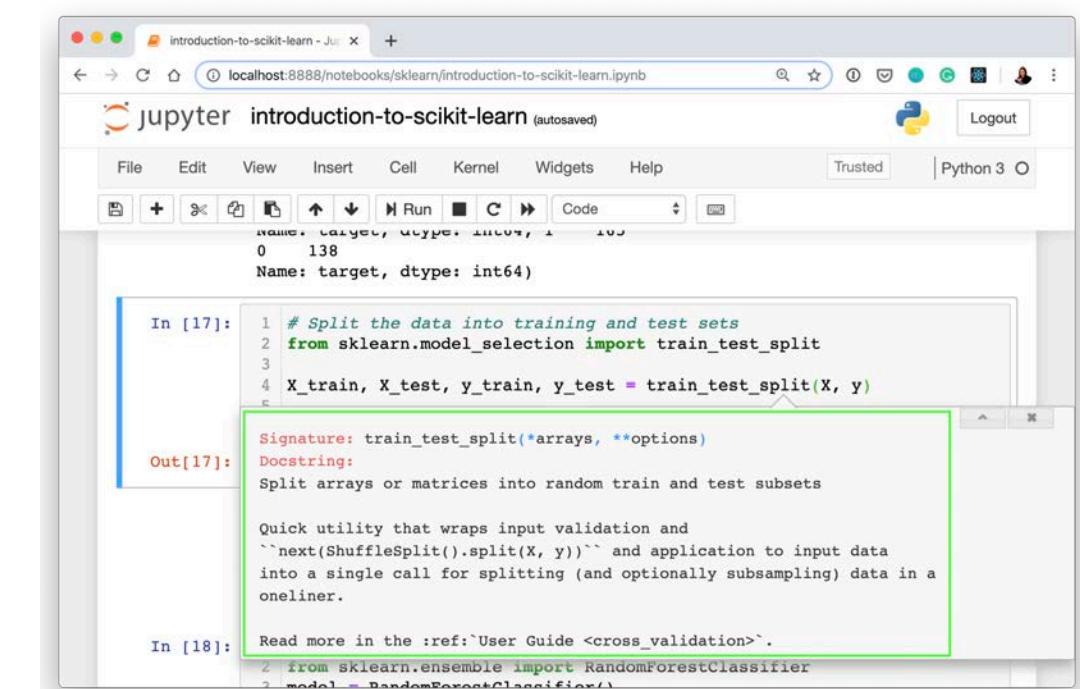
- Try again

- 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



```
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]: 0 138  
Name: target, dtype: int64
```

```
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  
3 model = RandomForestClassifier(n_estimators=10)
```

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

R² (r-squared)

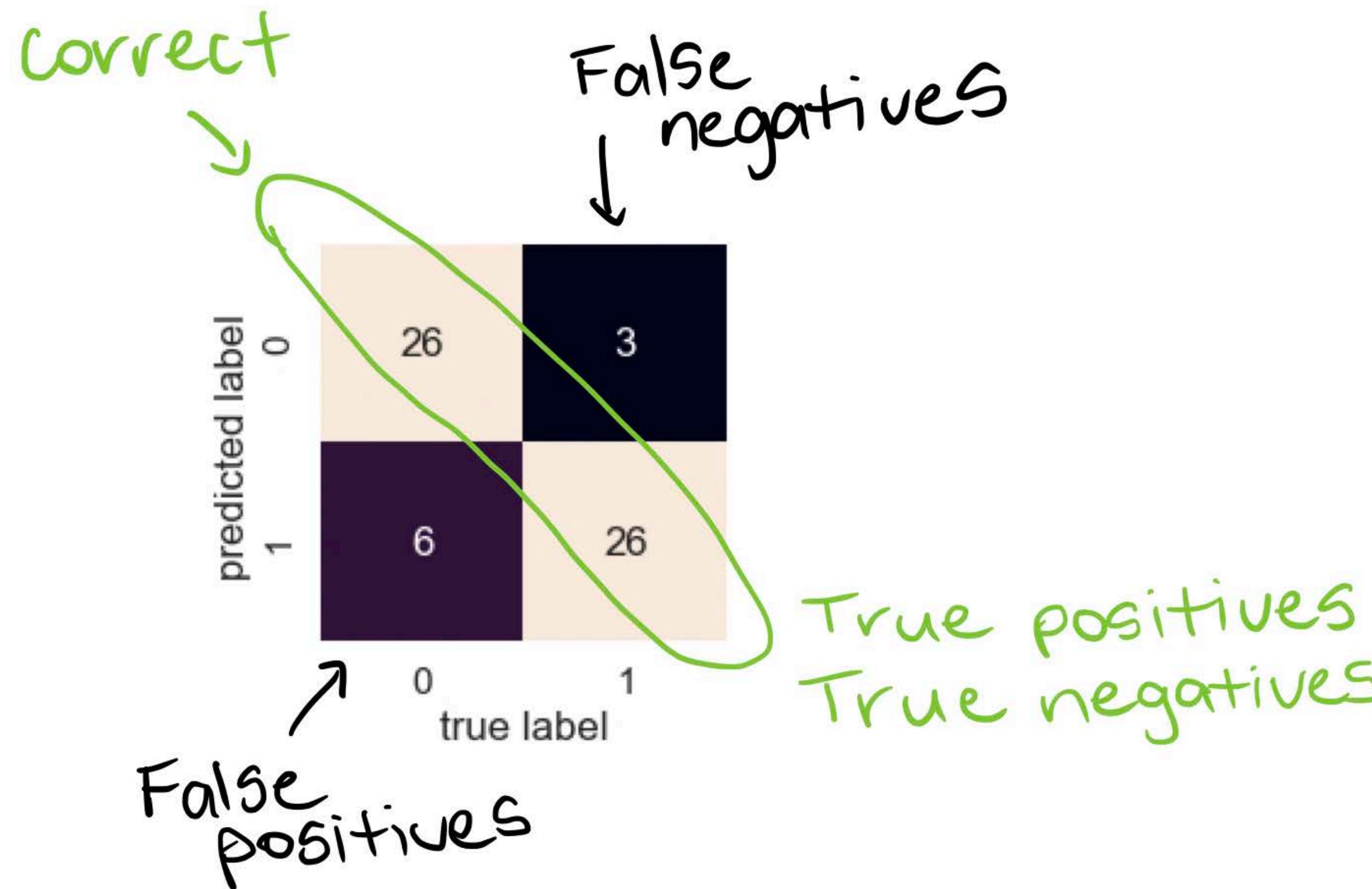
Mean absolute error (MAE)

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?

- **R²** is similar to accuracy. It gives you a quick indication of how well your model might be doing. Generally, the closer your **R²** 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)

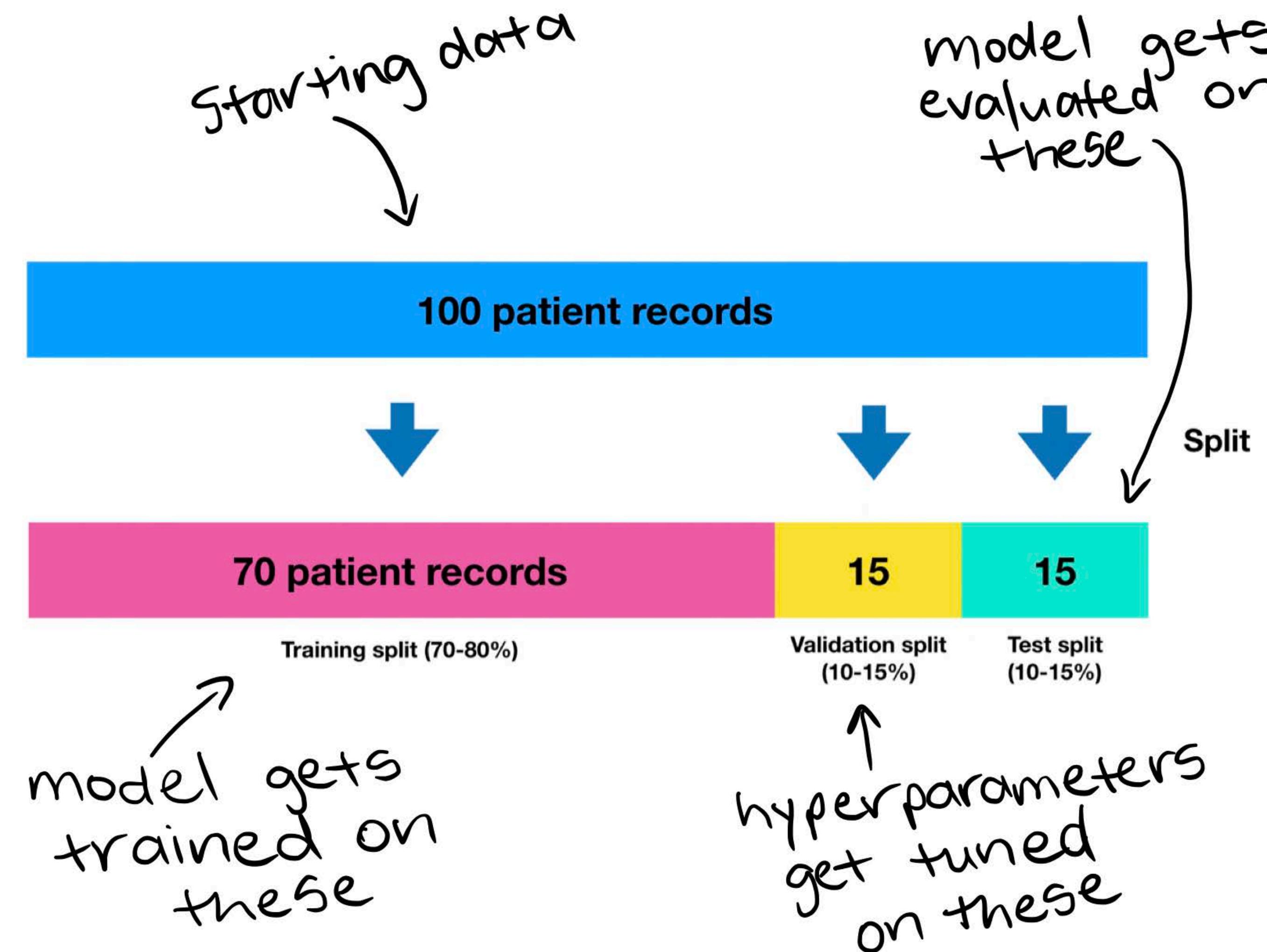


Cooking time: 1 hour
Temperature: 180°C



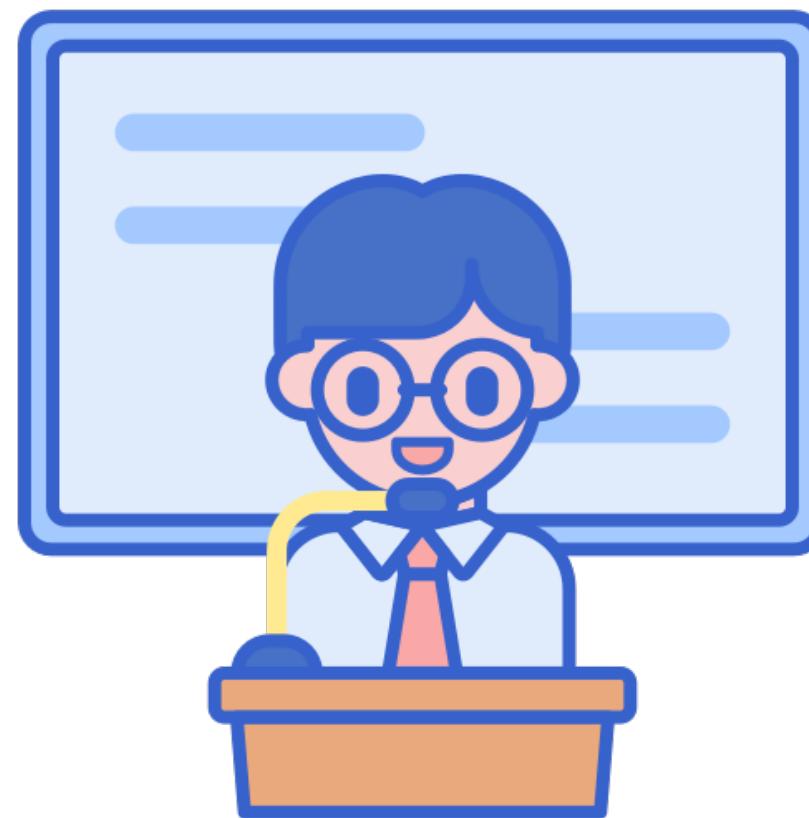
Cooking time: 1 hour
Temperature: 200°C

Tuning Hyperparameters by Hand

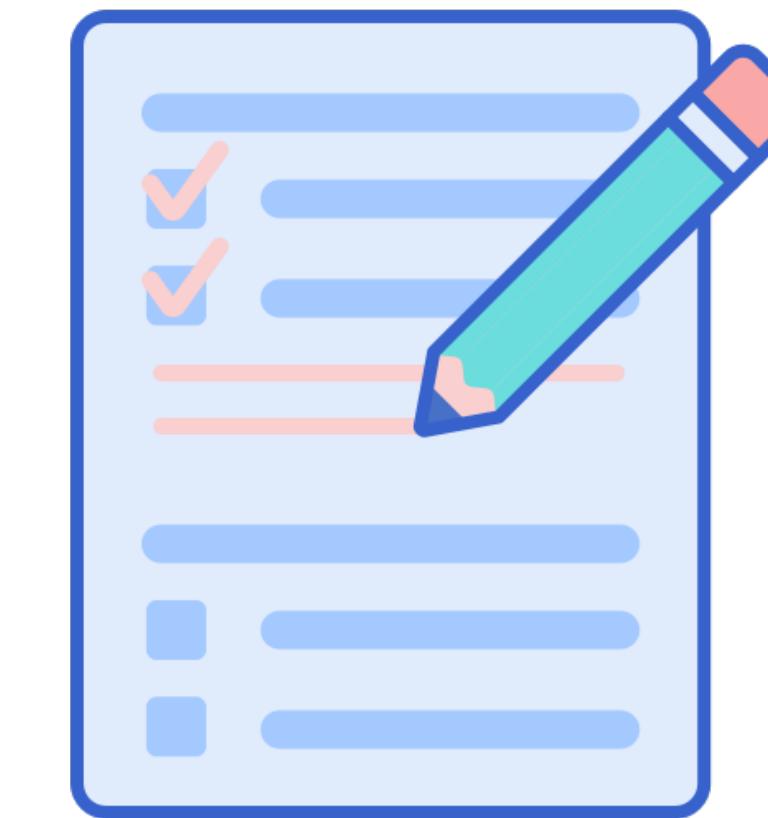
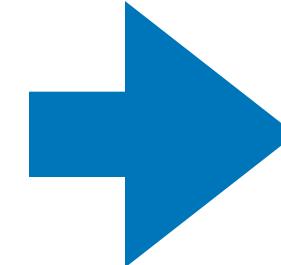


The most important concept in machine learning

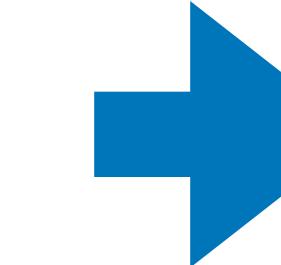
(the 3 sets)



Course materials
(training set)



Practice exam
(validation set)



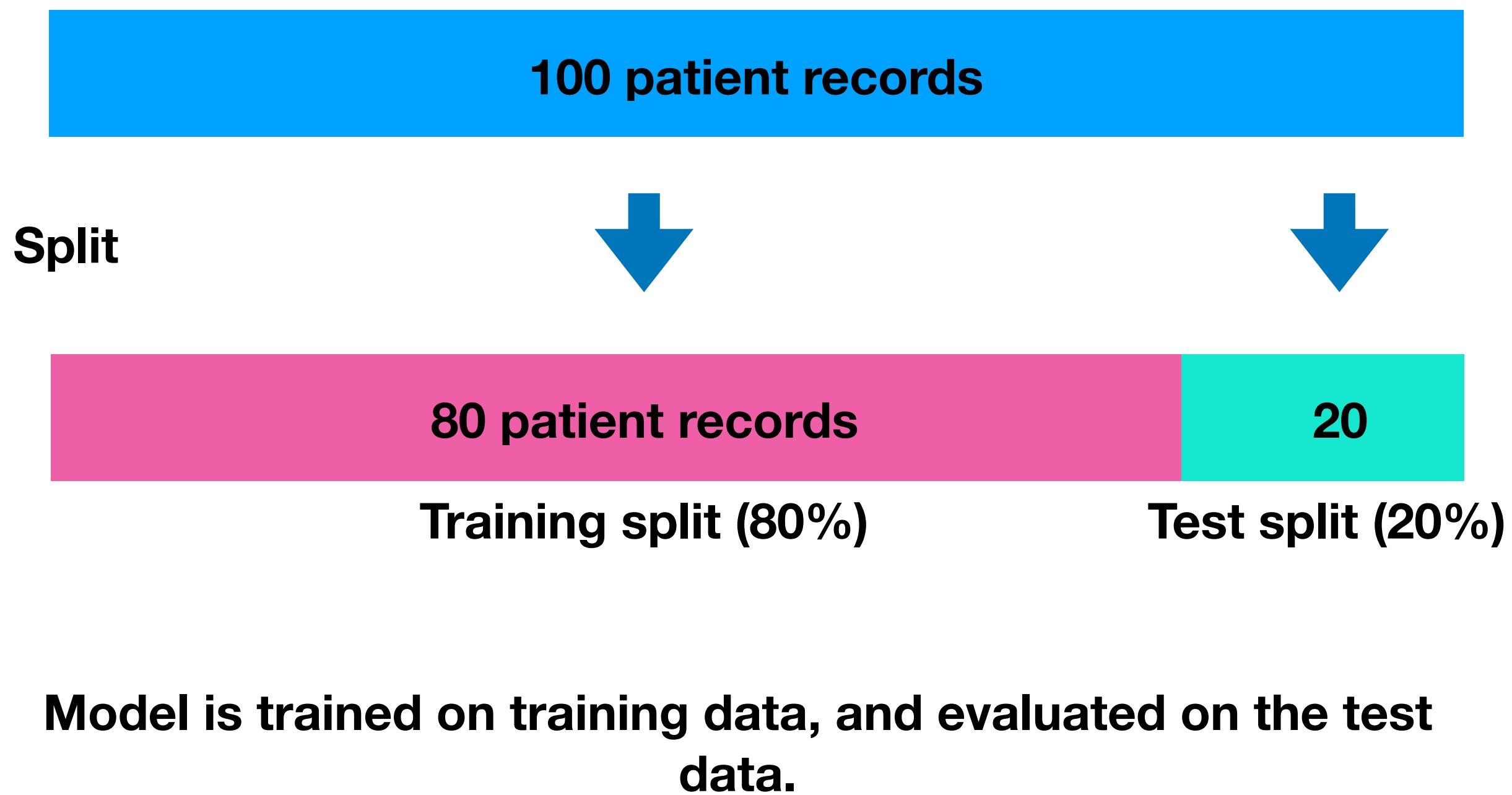
Final exam
(test set)

Generalization

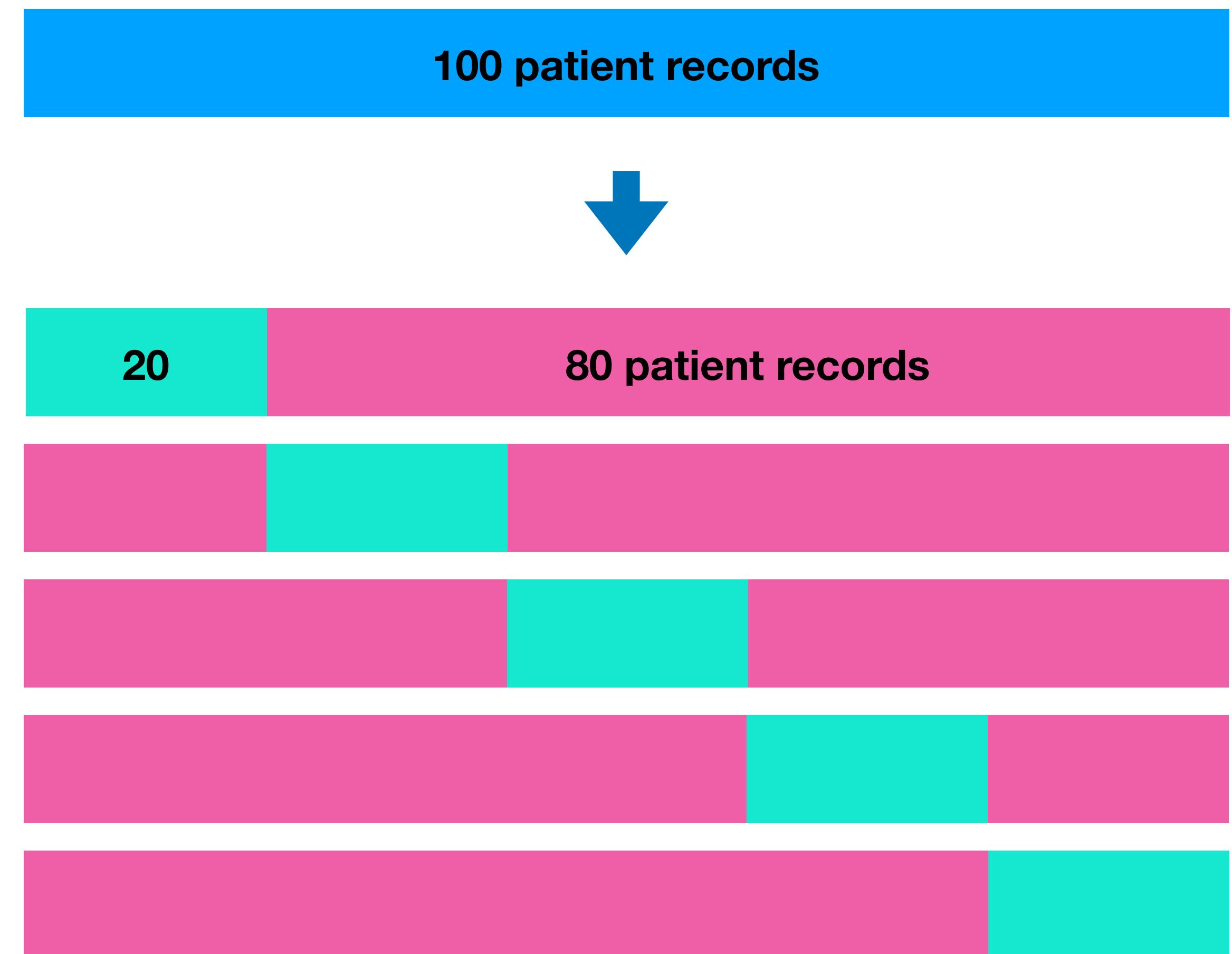
The ability for a machine learning model to perform well on data it hasn't seen before.

Cross-validation

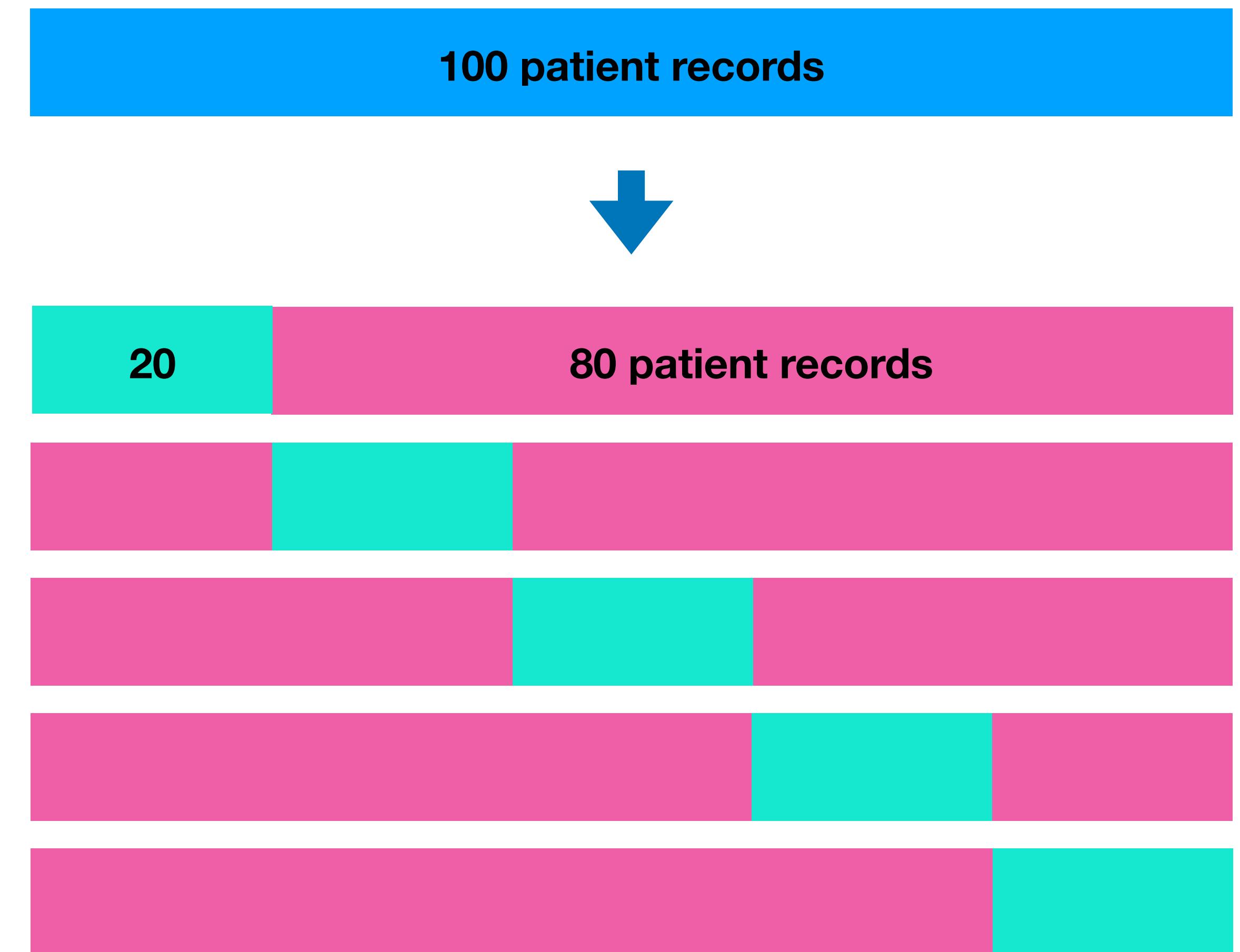
Normal Train & Test Split



5-fold Cross-validation



5-fold Cross-validation



Normal Train & Test Split

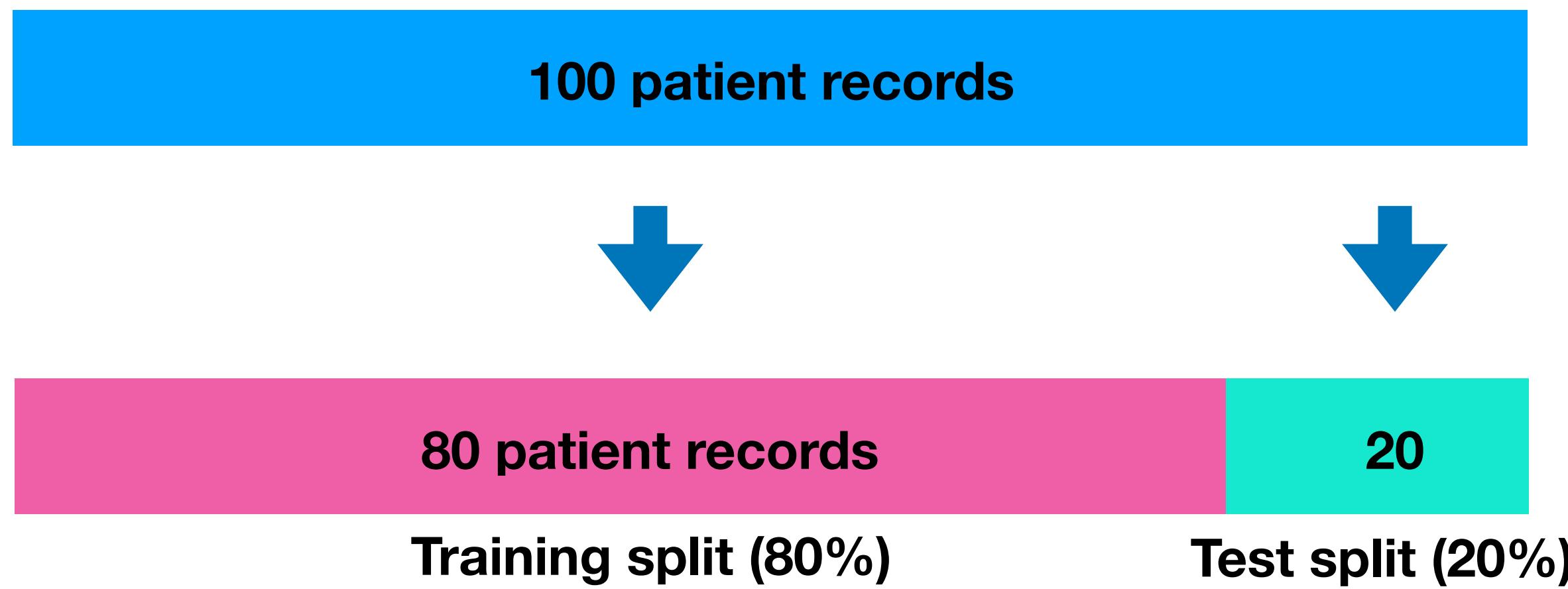


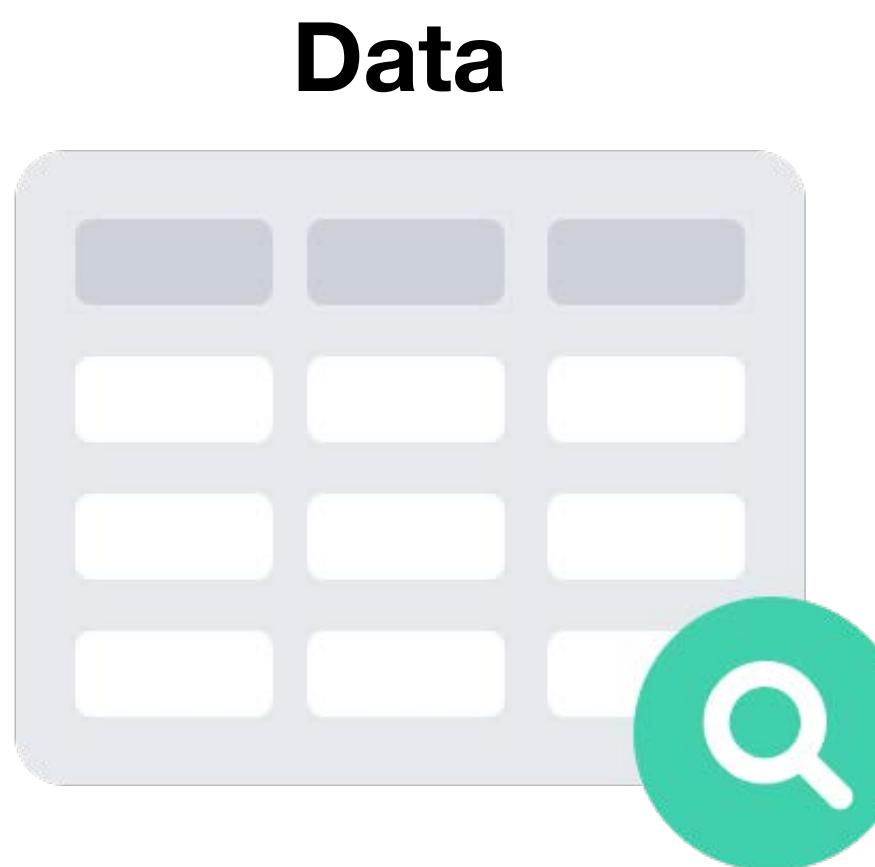
Figure 1.0: Model is trained on training data, and evaluated on the test data.

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

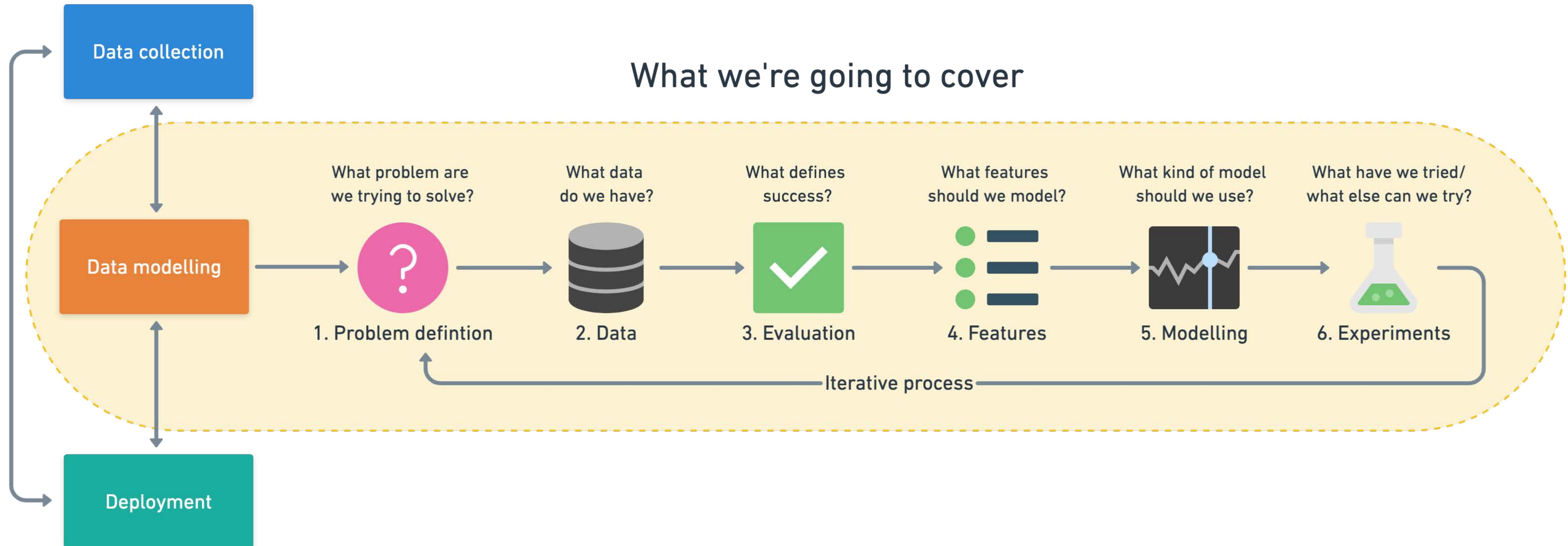
What is structured data?



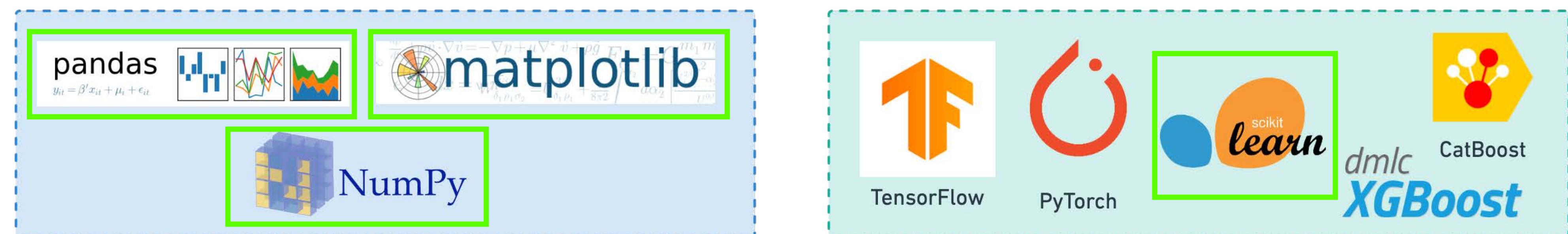
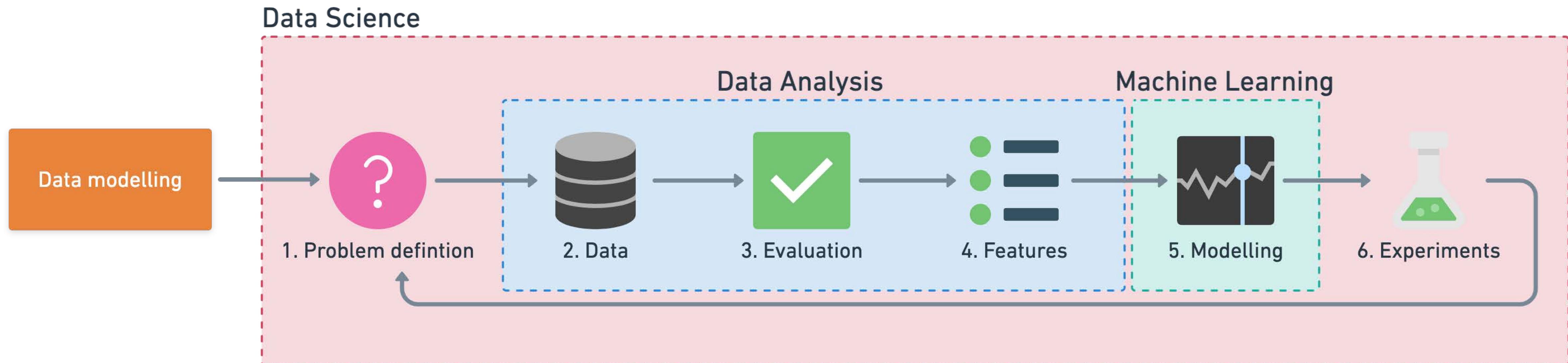
ID	Weight	Sex	Heart Rate	Chest pain	Heart disease?
4326	110Kg	M	81	4	Yes
5681	64Kg	F	61	1	No
7911	81Kg	M	57	0	No

Table 1.0 : Patient records

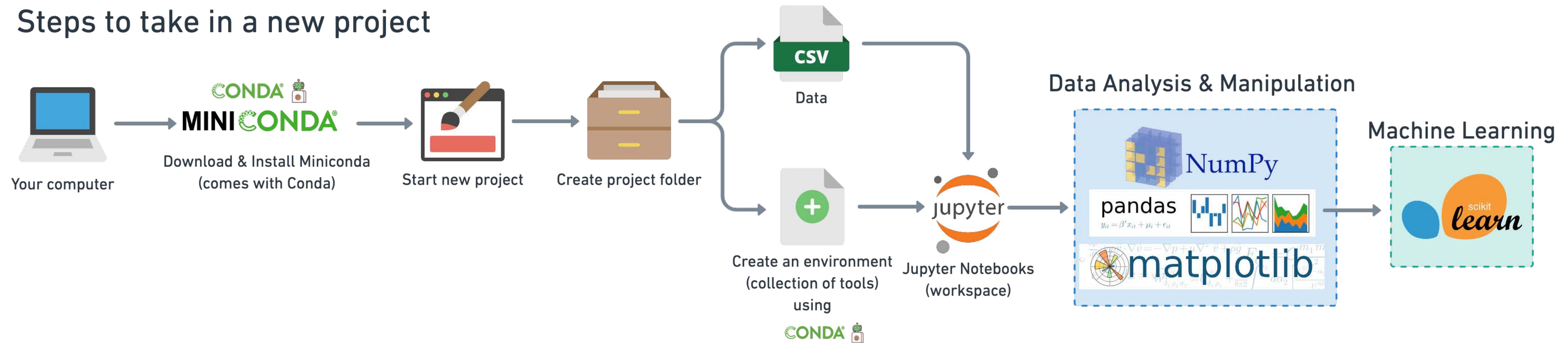
Steps in a full machine learning project



Tools you can use

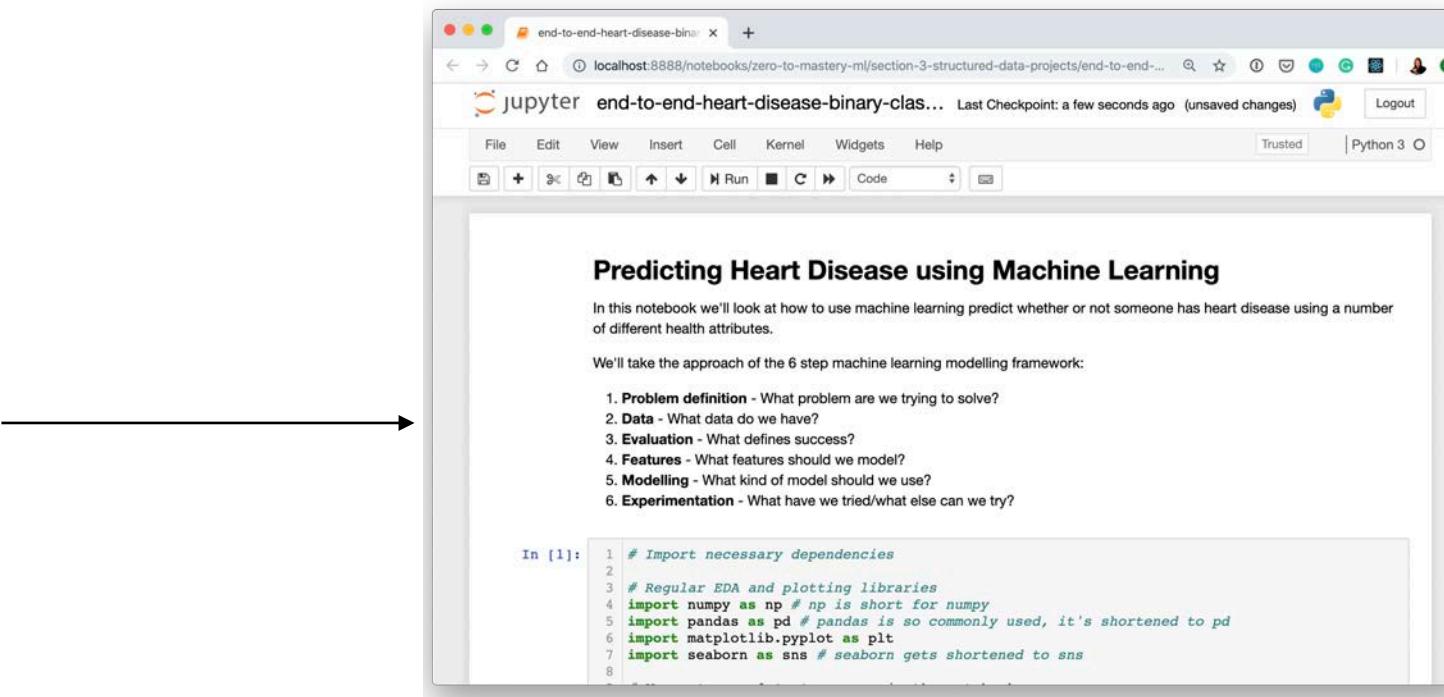


Steps to take in a new project



Where can you get help?

- Follow along with the code



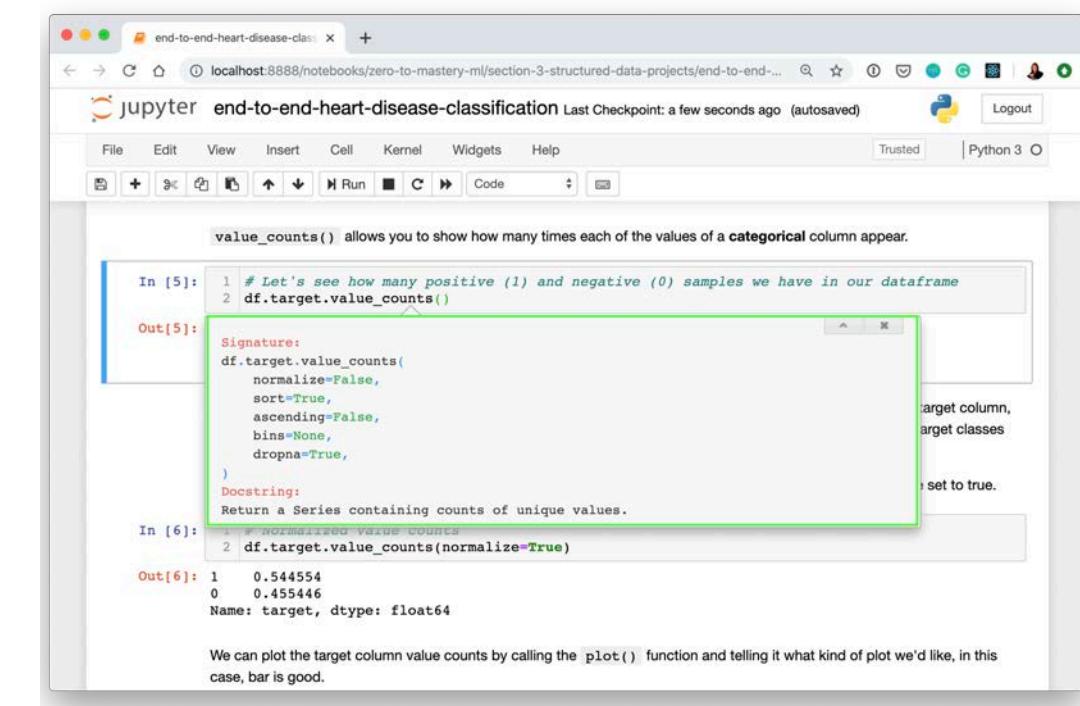
```
# Import necessary dependencies
# Regular EDA and plotting libraries
import numpy as np # np is short for numpy
import pandas as pd # pandas is so commonly used, it's shortened to pd
import matplotlib.pyplot as plt
import seaborn as sns # seaborn gets shortened to sns
```

In this notebook we'll look at how to use machine learning predict whether or not someone has heart disease using a number of different health attributes.

We'll take the approach of the 6 step machine learning modeling framework:

1. Problem definition - What problem are we trying to solve?
2. Data - What data do we have?
3. Evaluation - What defines success?
4. Features - What features should we model?
5. Modelling - What kind of model should we use?
6. Experimentation - What have we tried/what else can we try?

- Try it for yourself



```
# Let's see how many positive (1) and negative (0) samples we have in our dataframe
```

```
df.target.value_counts()
```

```
Signature: df.target.value_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True, ...)
```

```
Docstring:
```

```
Return a Series containing counts of unique values.
```

```
1   0.544554
```

```
0   0.455446
```

```
Name: target, dtype: float64
```

We can plot the target column value counts by calling the `plot()` function and telling it what kind of plot we'd like, in this case, bar is good.

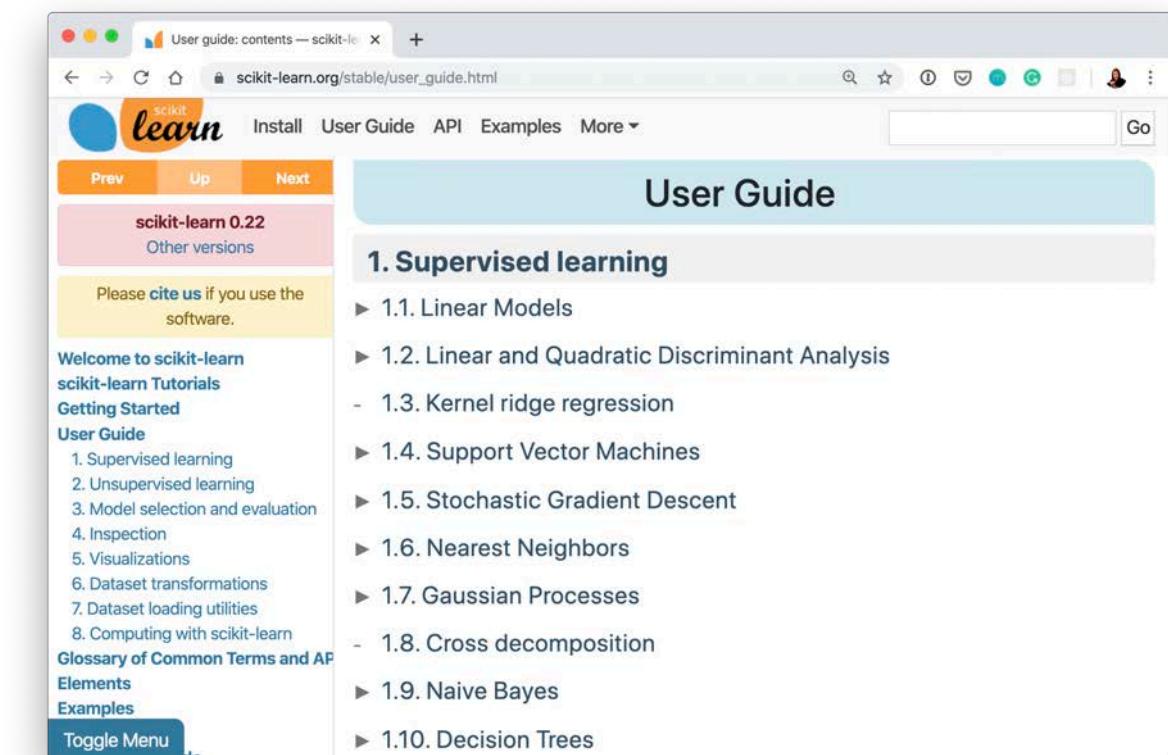
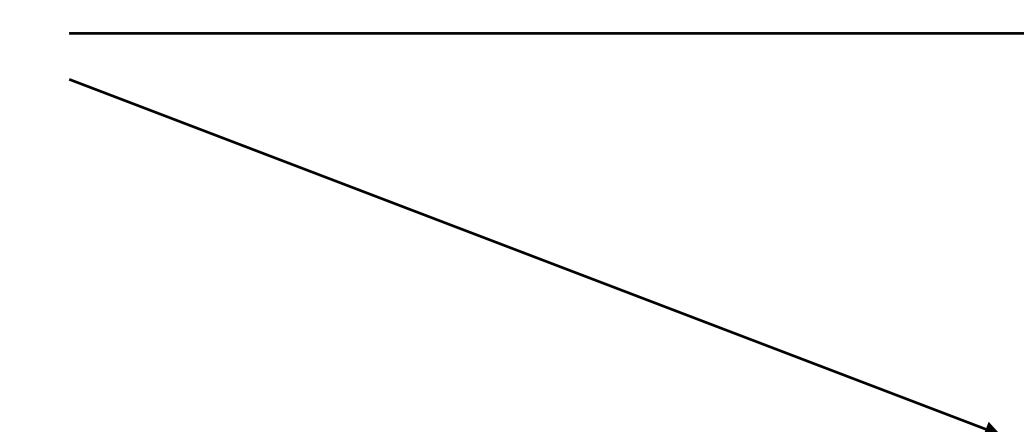
- Press SHIFT + TAB to read the docstring



- Search for it

- Try again

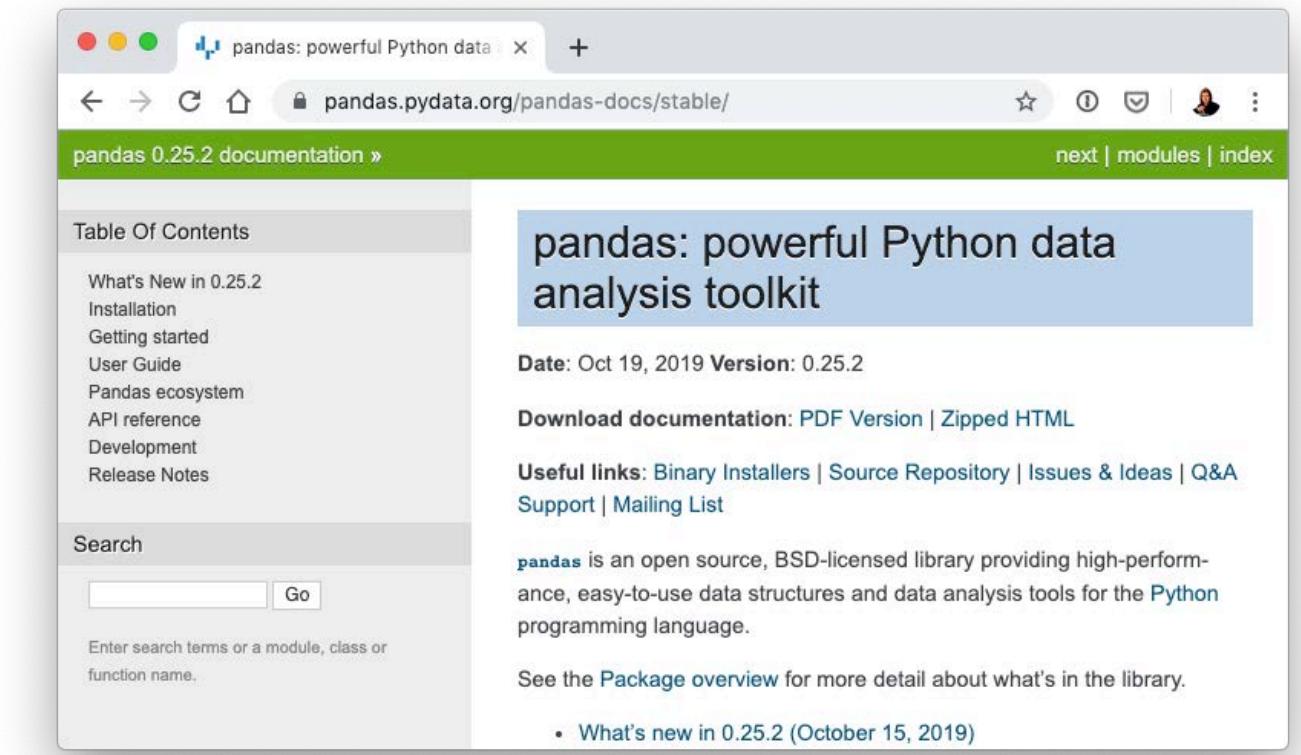
- Ask



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

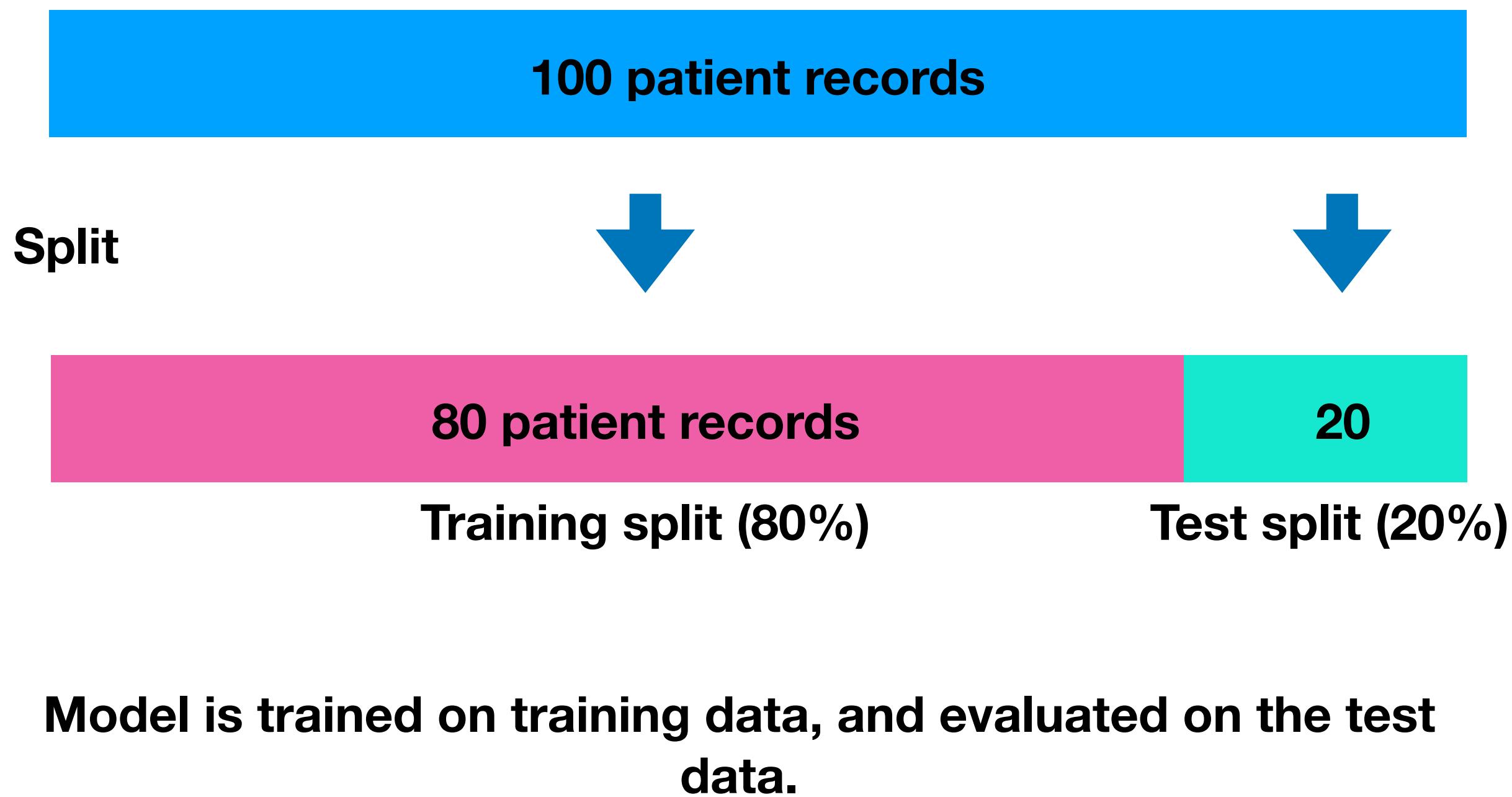
pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

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

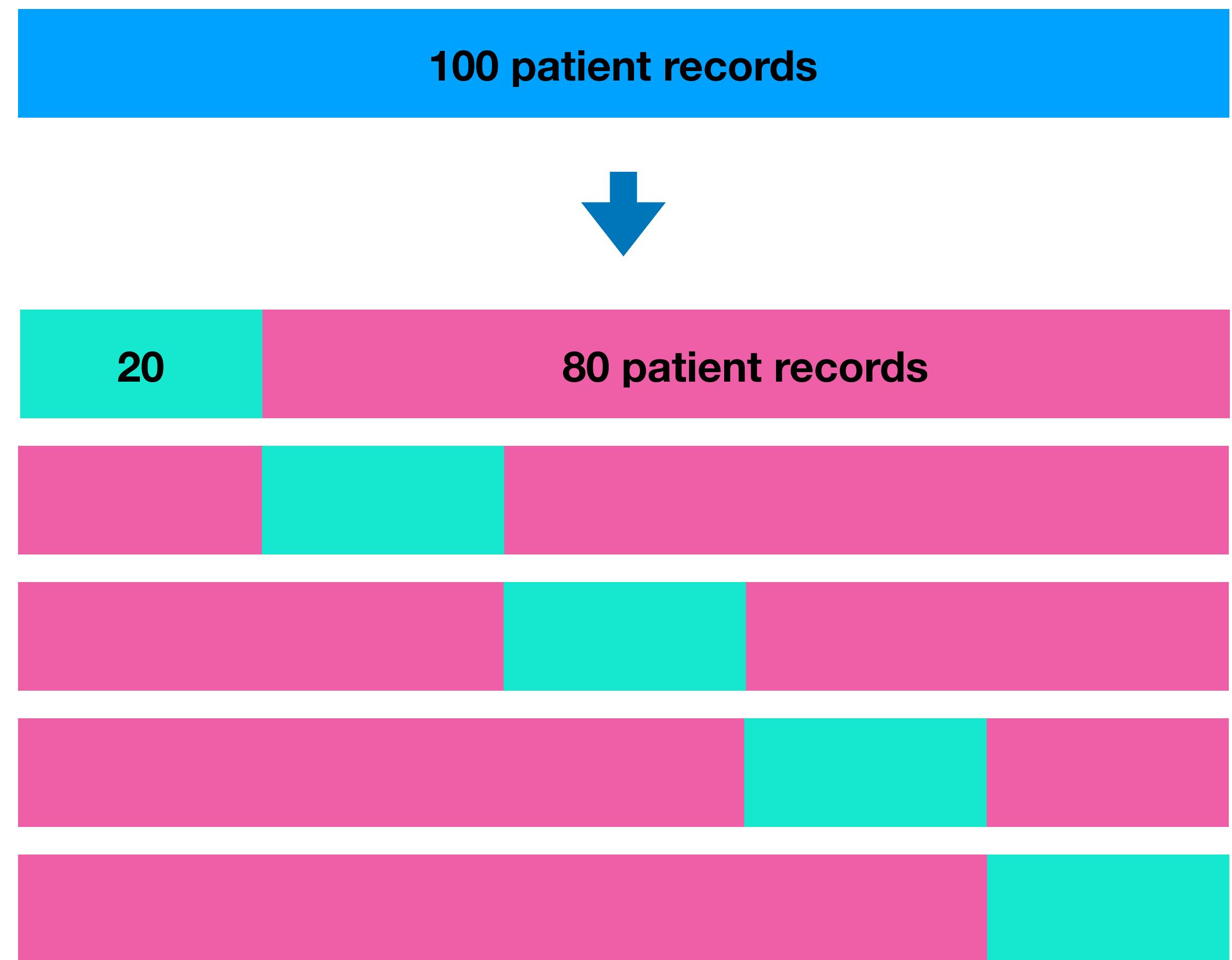
- What's new in 0.25.2 (October 15, 2019)

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

R² (r-squared)

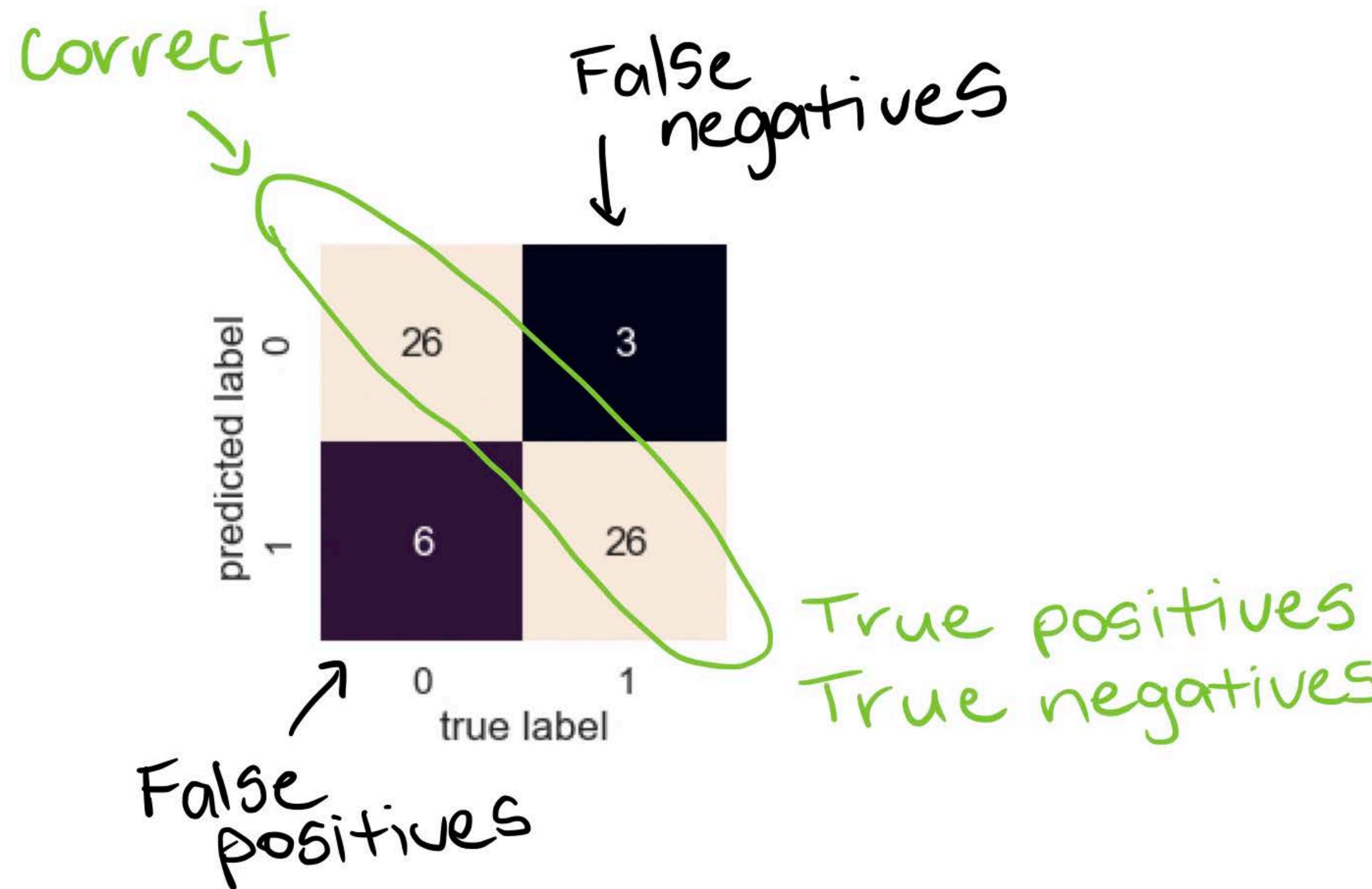
Mean absolute error (MAE)

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))
```

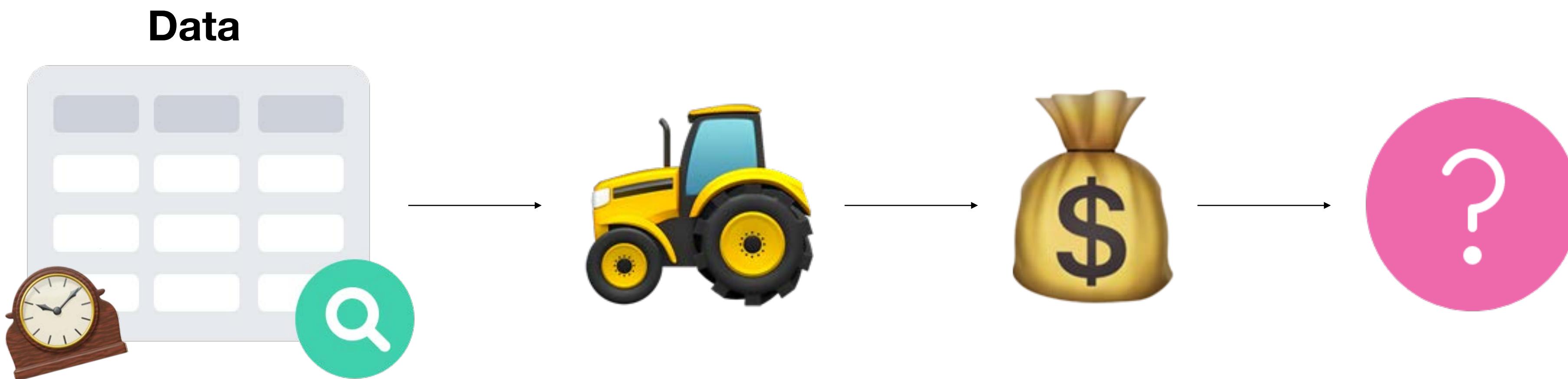
	precision	recall	f1-score	support
0	0.81	0.90	0.85	29
1	0.90	0.81	0.85	32
accuracy				
macro avg	0.85	0.85	0.85	61
weighted avg	0.86	0.85	0.85	61

- **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).

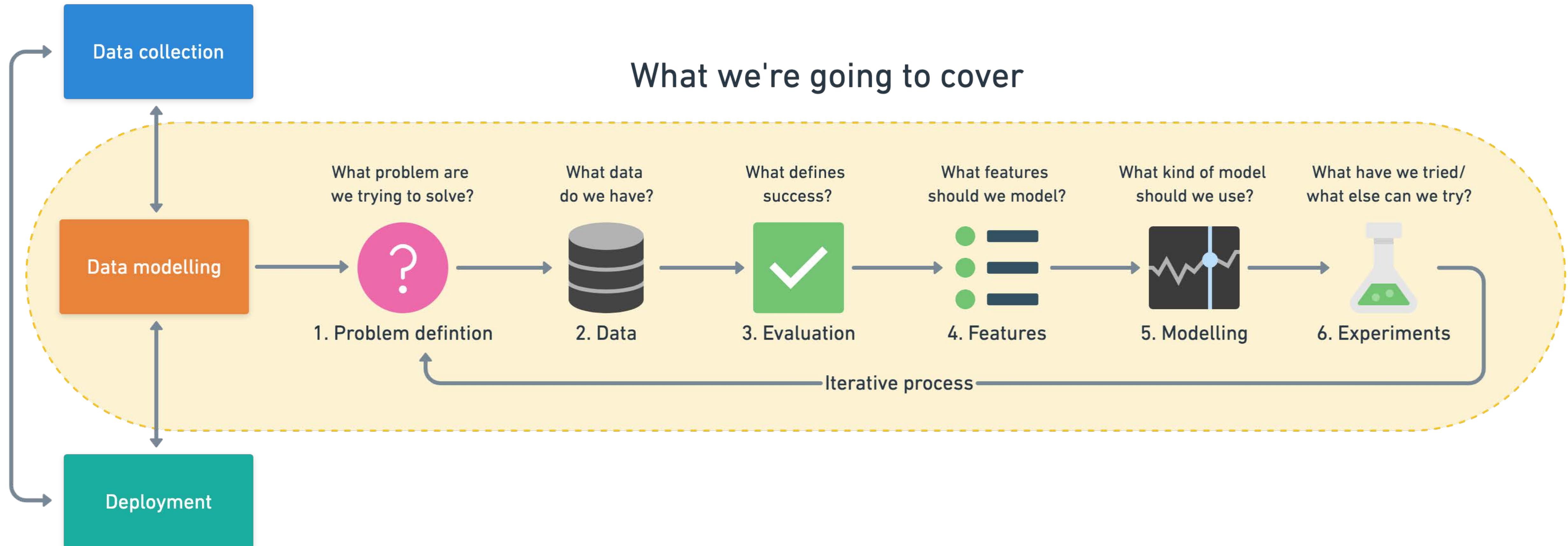
Which classification metric should you use?

- **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.

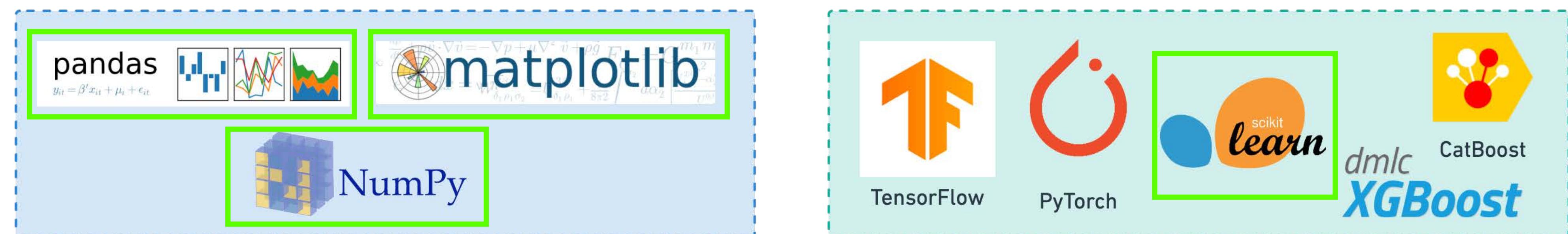
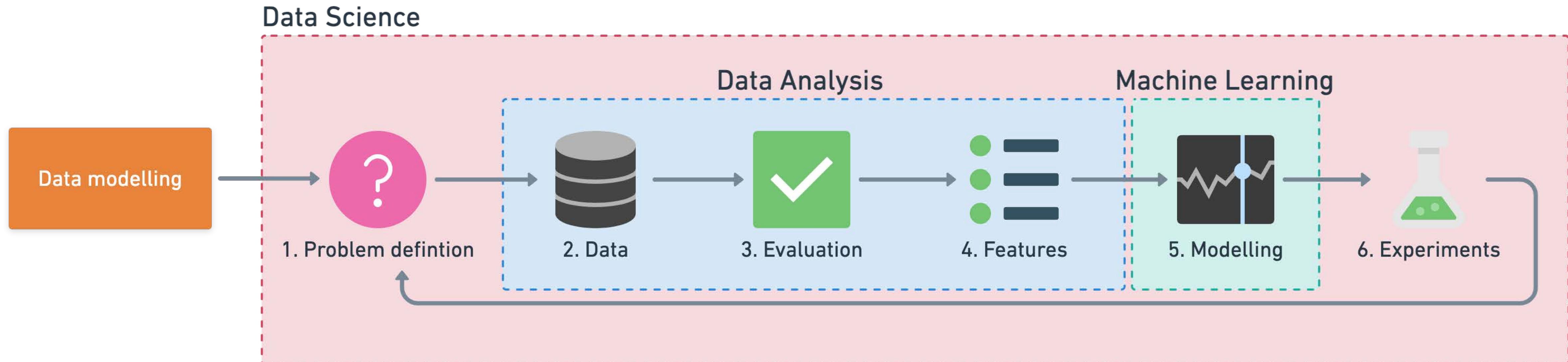
Structured Data Project 2: Predicting the sale price of Bulldozers (regression)



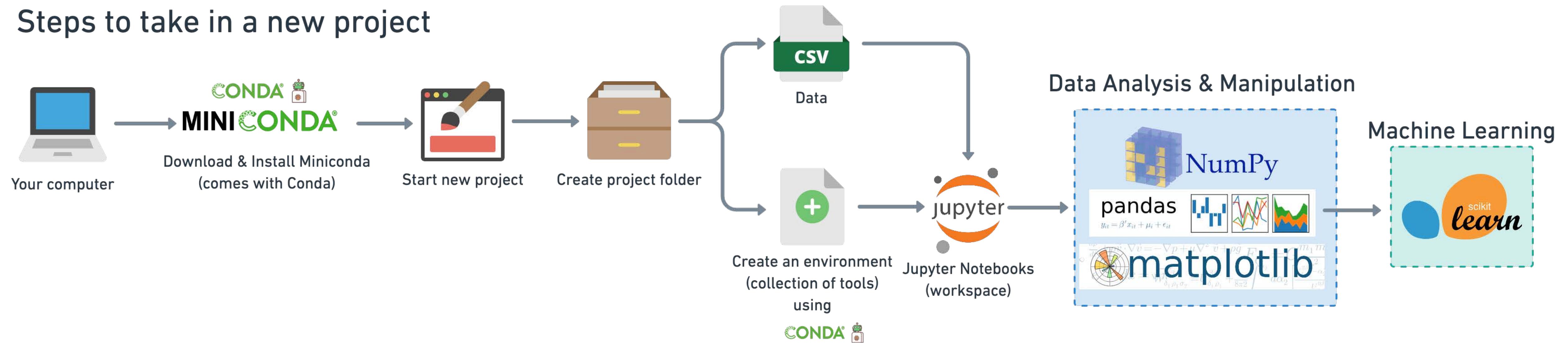
Steps in a full machine learning project



Tools you can use

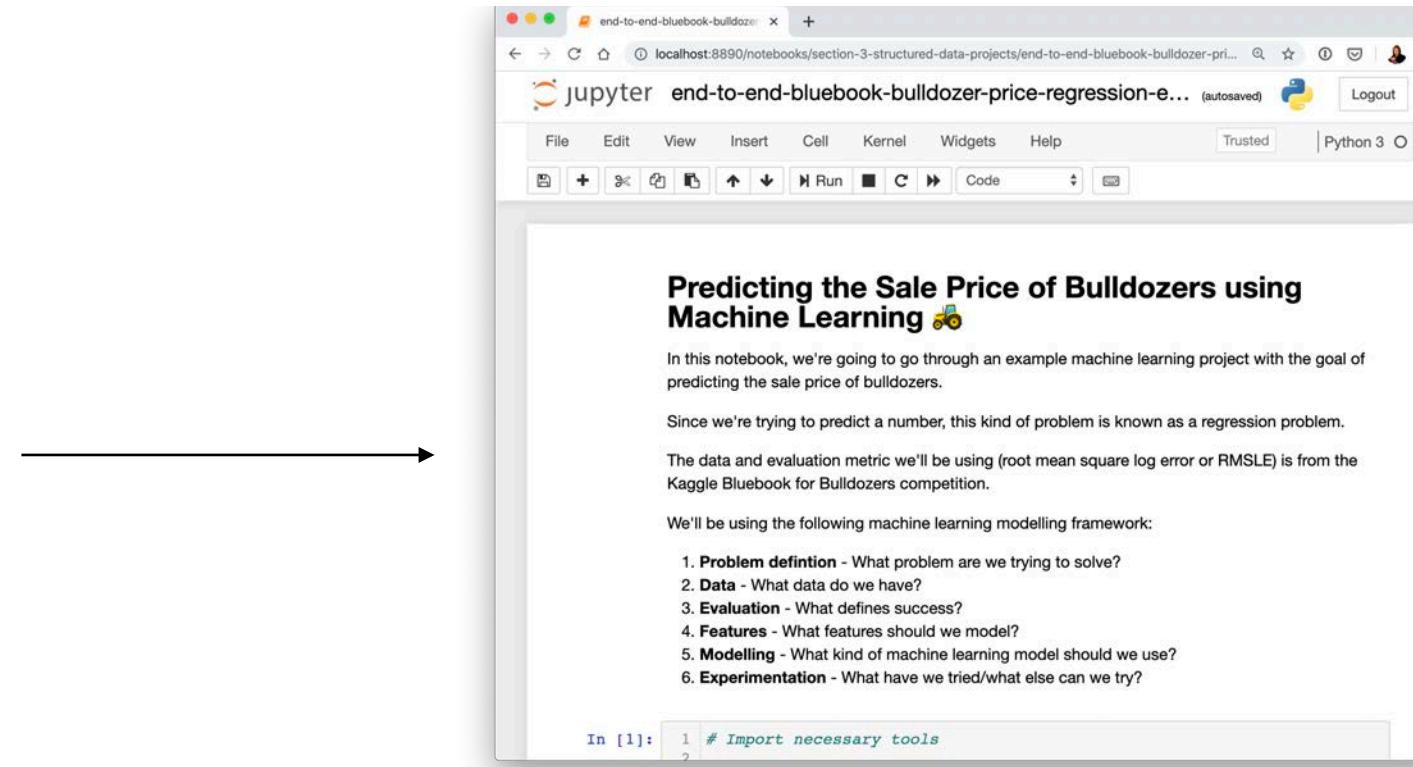


Steps to take in a new project



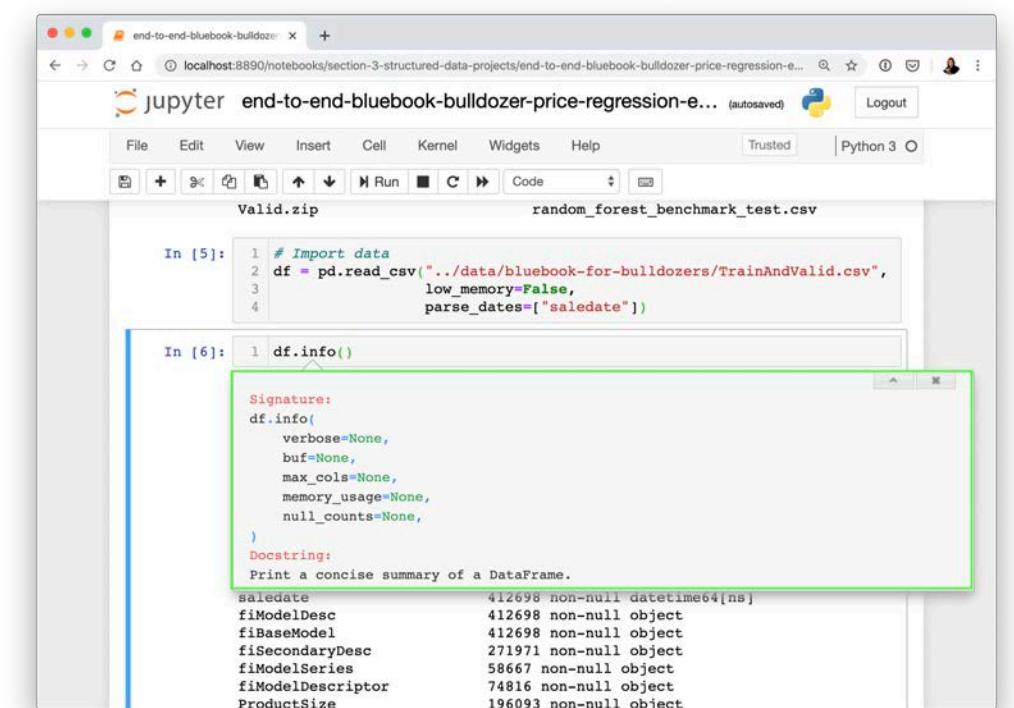
Where can you get help?

- 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,
buf=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
fimodelid          412698 non-null object
fissecondarydesc   271971 non-null object
fimodelseries      58667 non-null object
fimodeldescriptor  74816 non-null object
ProductSize        196093 non-null object
```

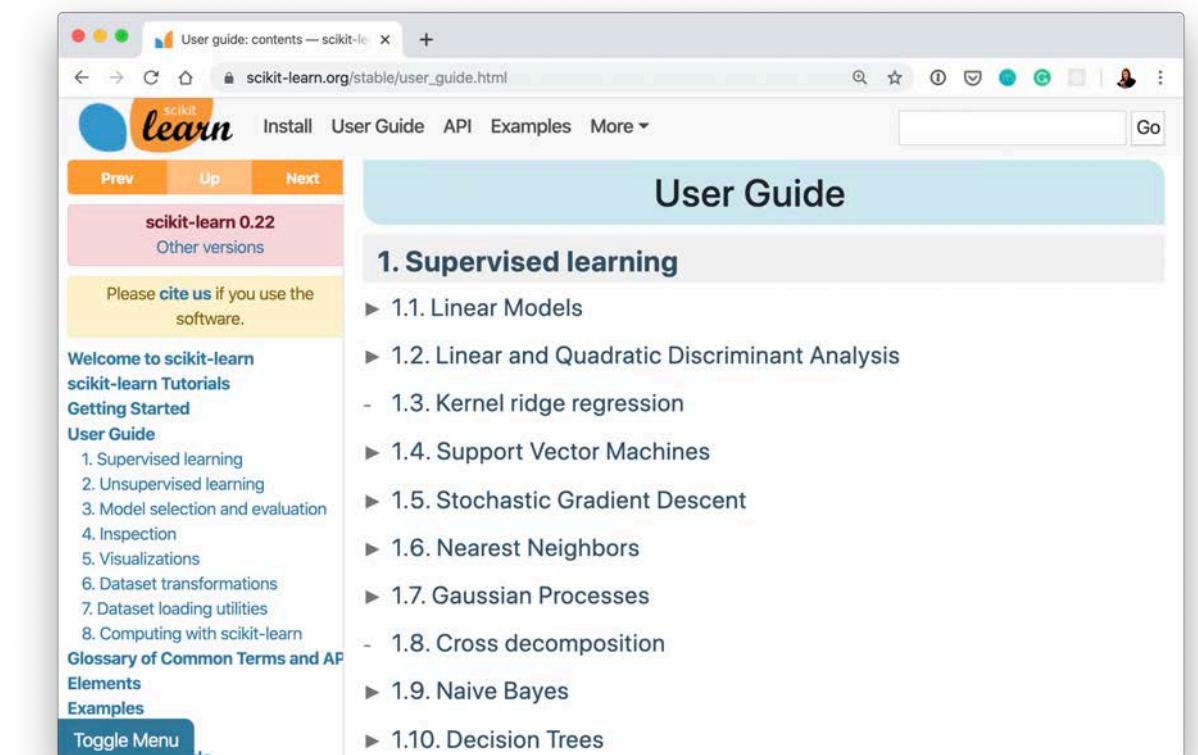
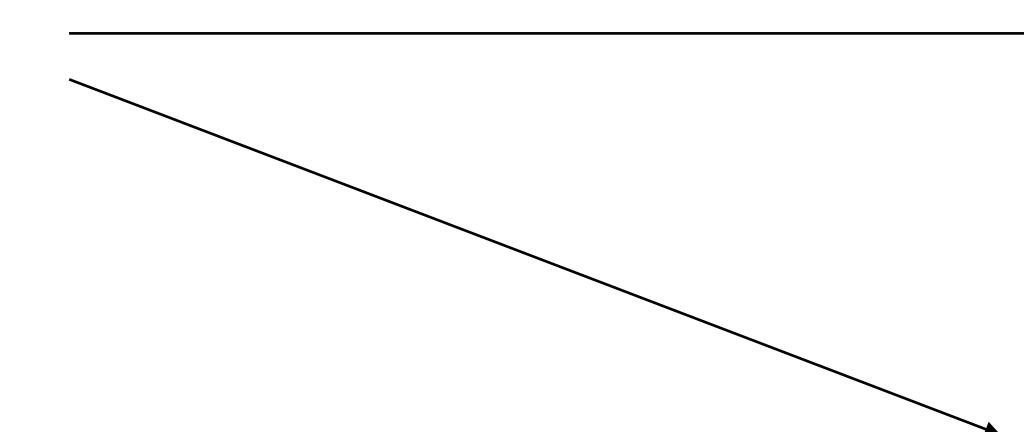
- Press SHIFT + TAB to read the docstring

- Search for it



- Try again

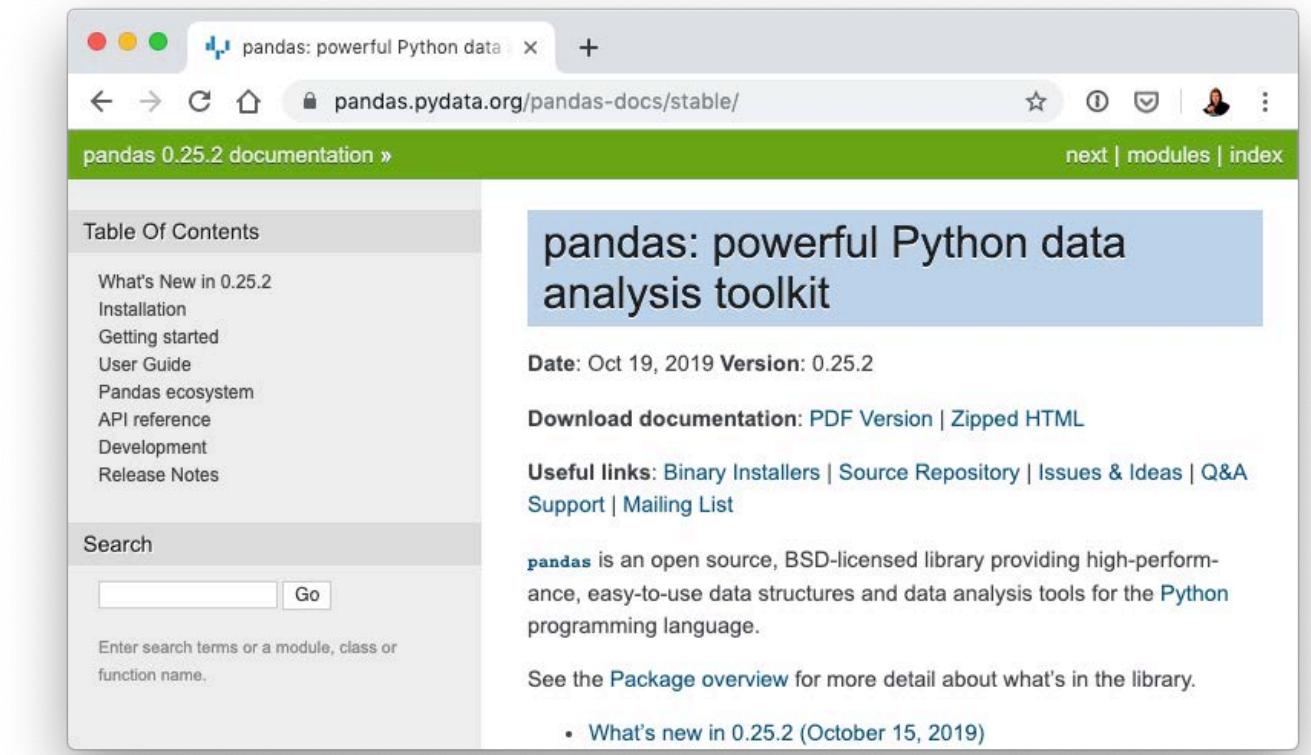
- Ask



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

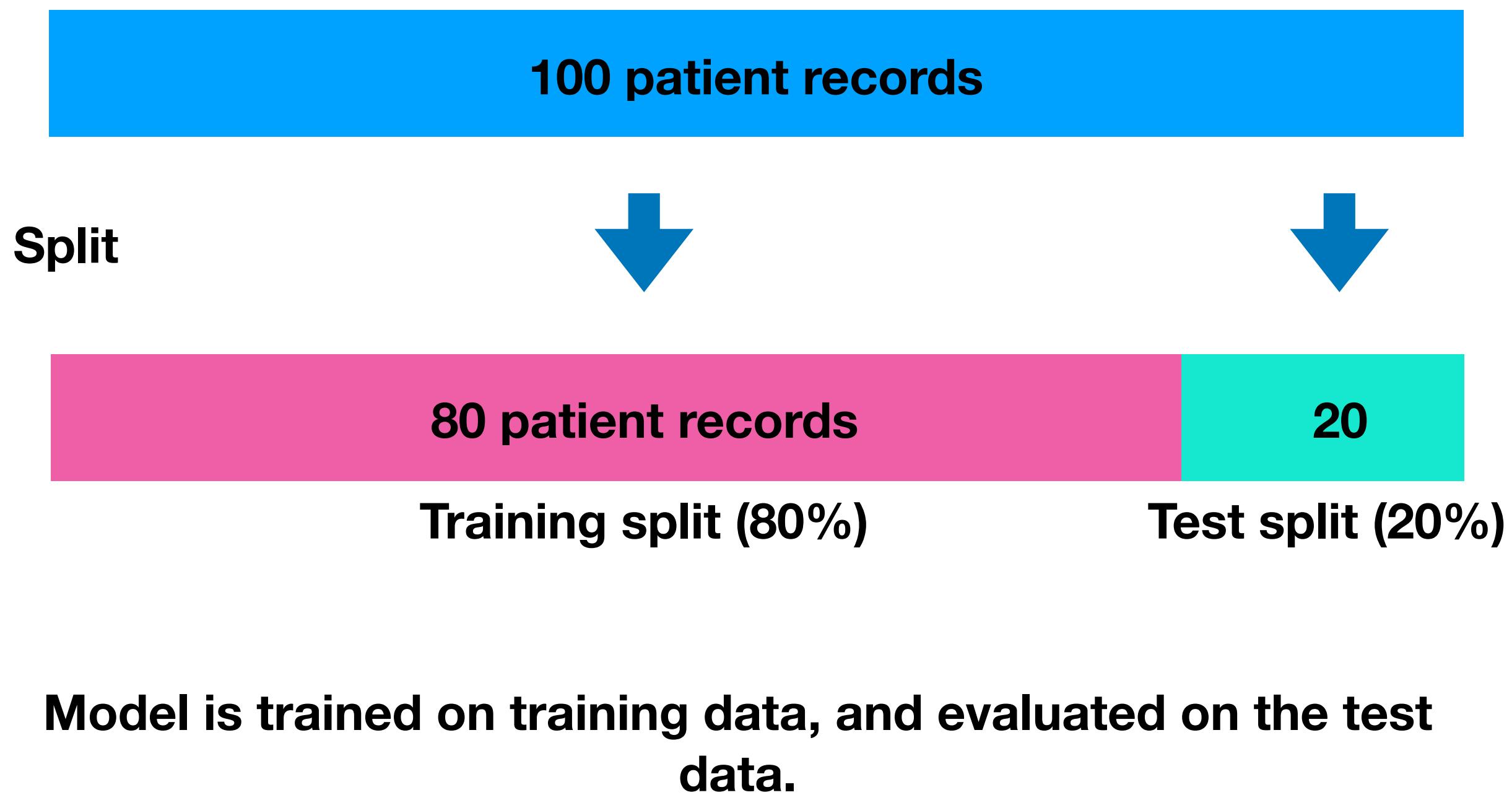
pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

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

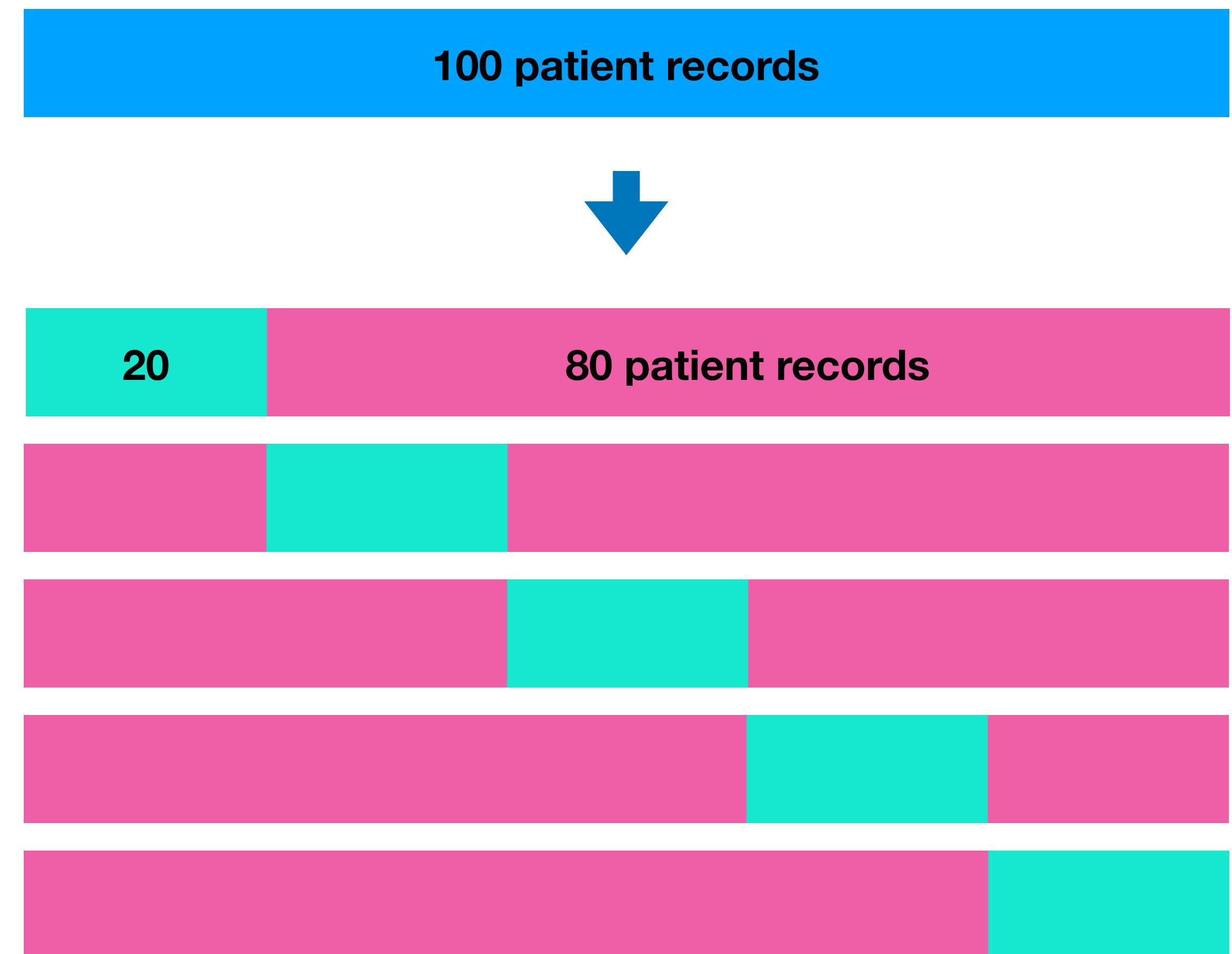
- What's new in 0.25.2 (October 15, 2019)

Cross-validation

Normal Train & Test Split



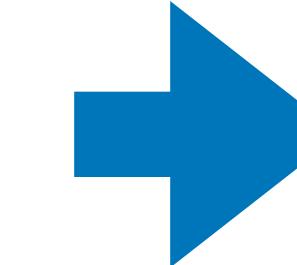
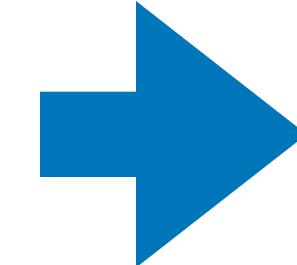
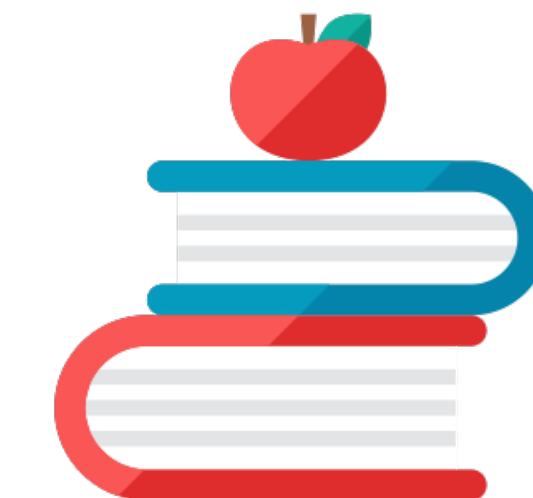
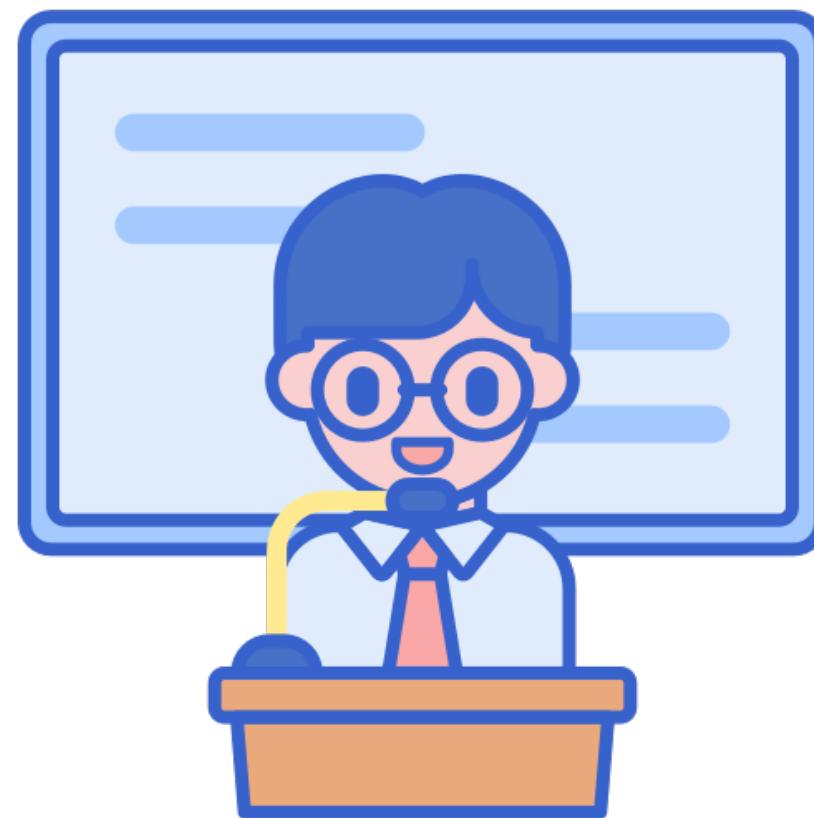
5-fold Cross-validation



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

R² (r-squared)

Mean absolute error (MAE)

Mean squared error (MSE)

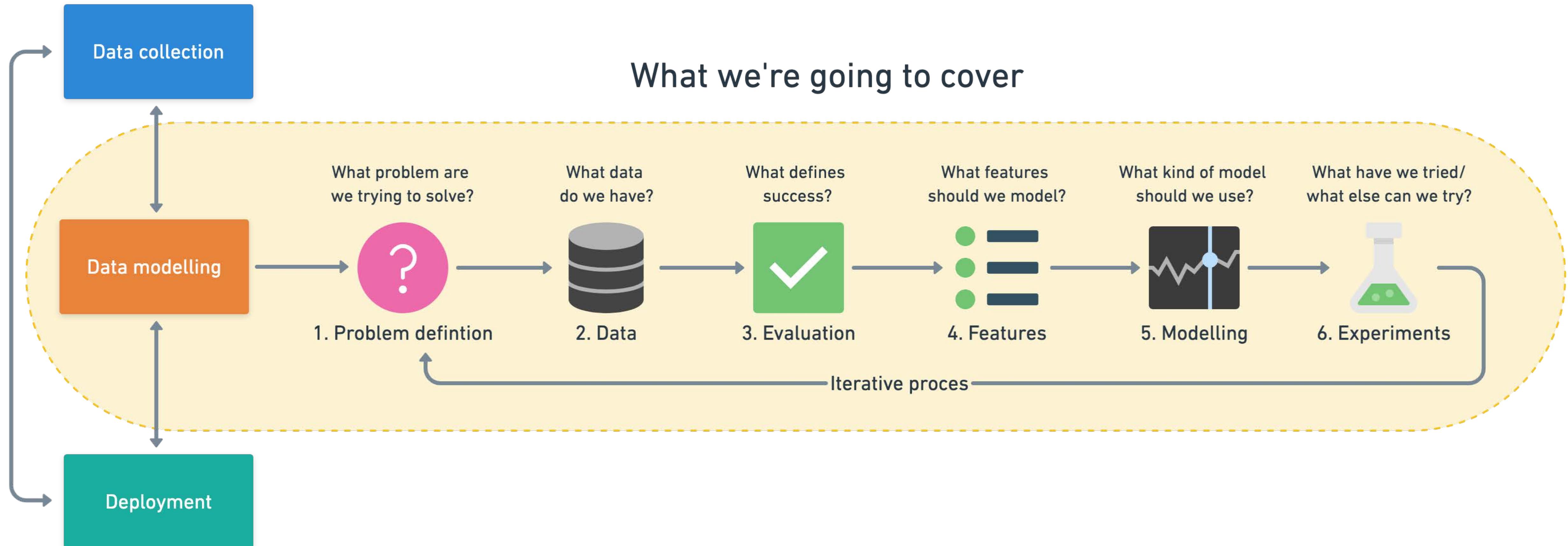
Root mean squared error (RMSE)

Bold = default evaluation in Scikit-Learn

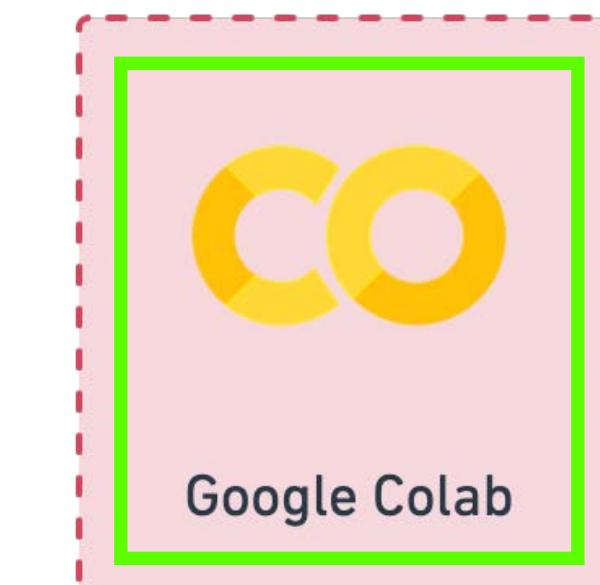
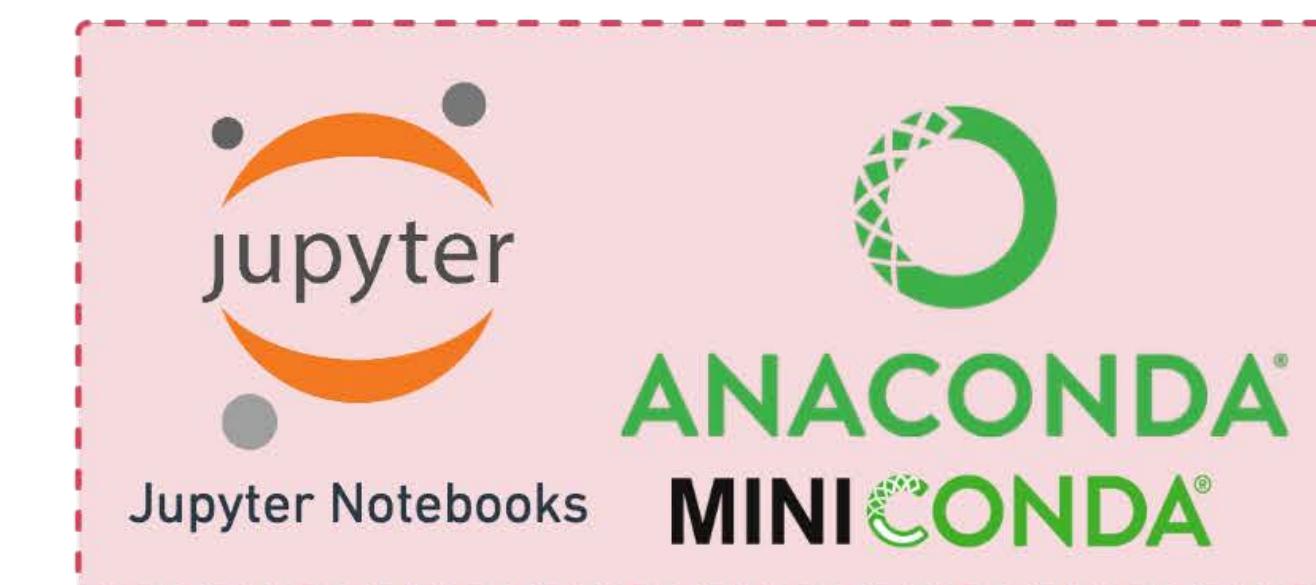
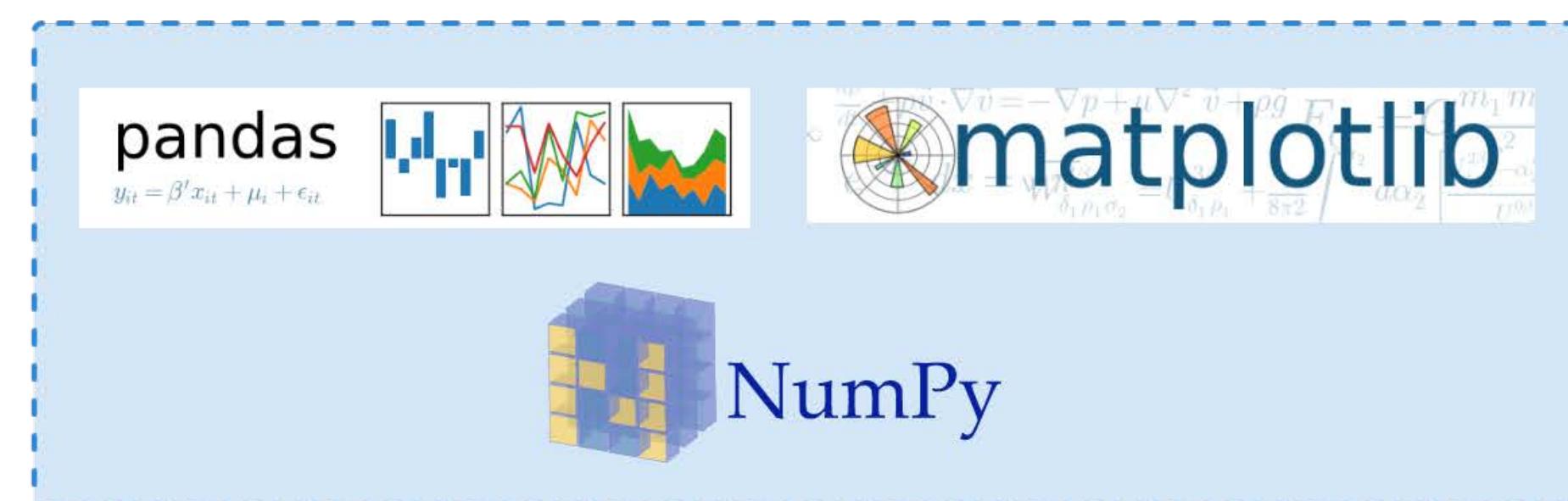
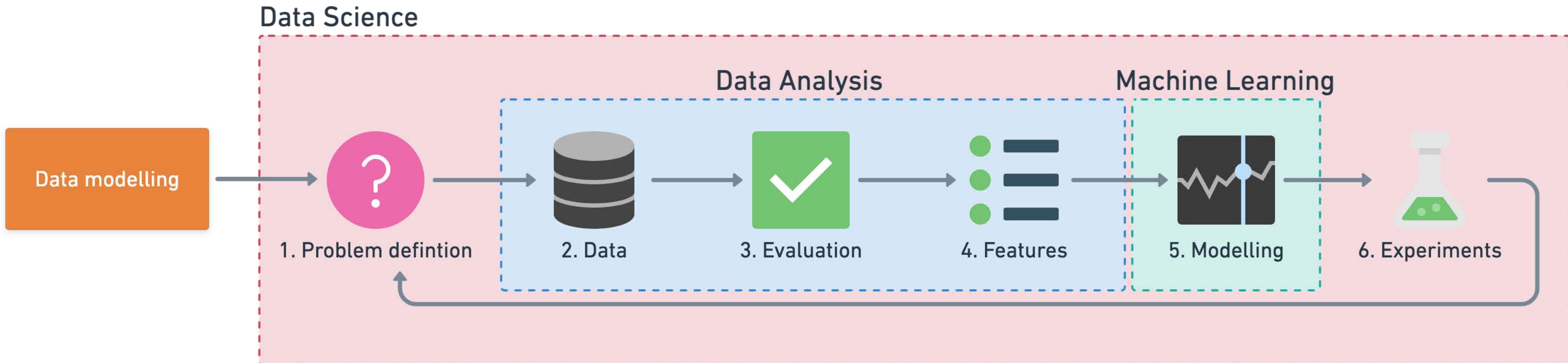
Which regression metric should you use?

- **R²** is similar to accuracy. It gives you a quick indication of how well your model might be doing. Generally, the closer your **R²** 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

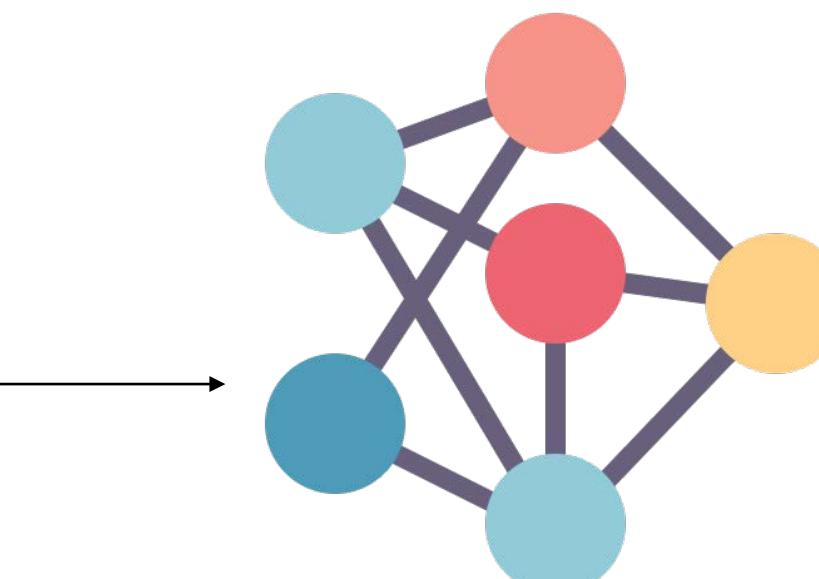
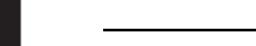
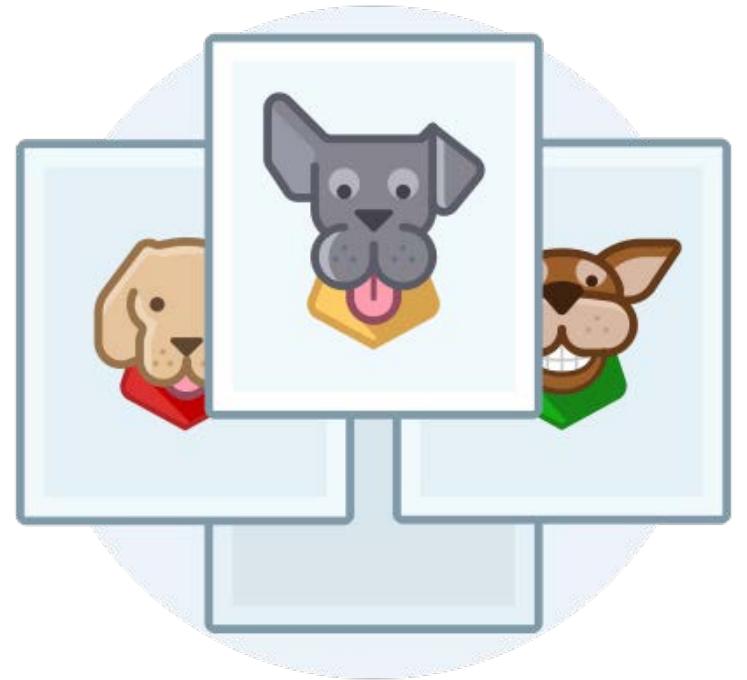
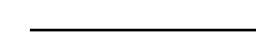


Tools you can use

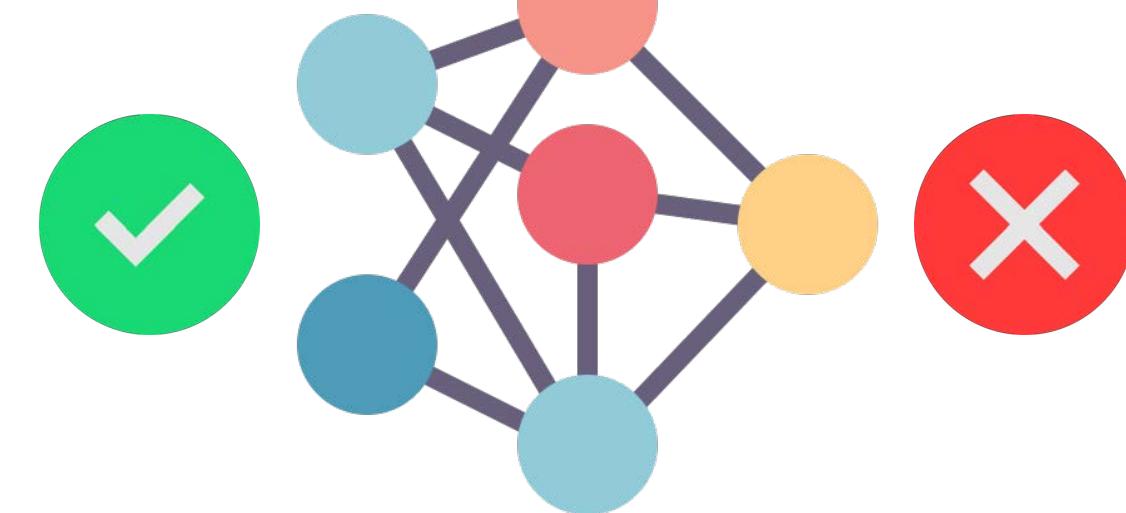


A new contender!

What is TensorFlow?



?



From: daniel@mrbourke.com
Hey Daniel,

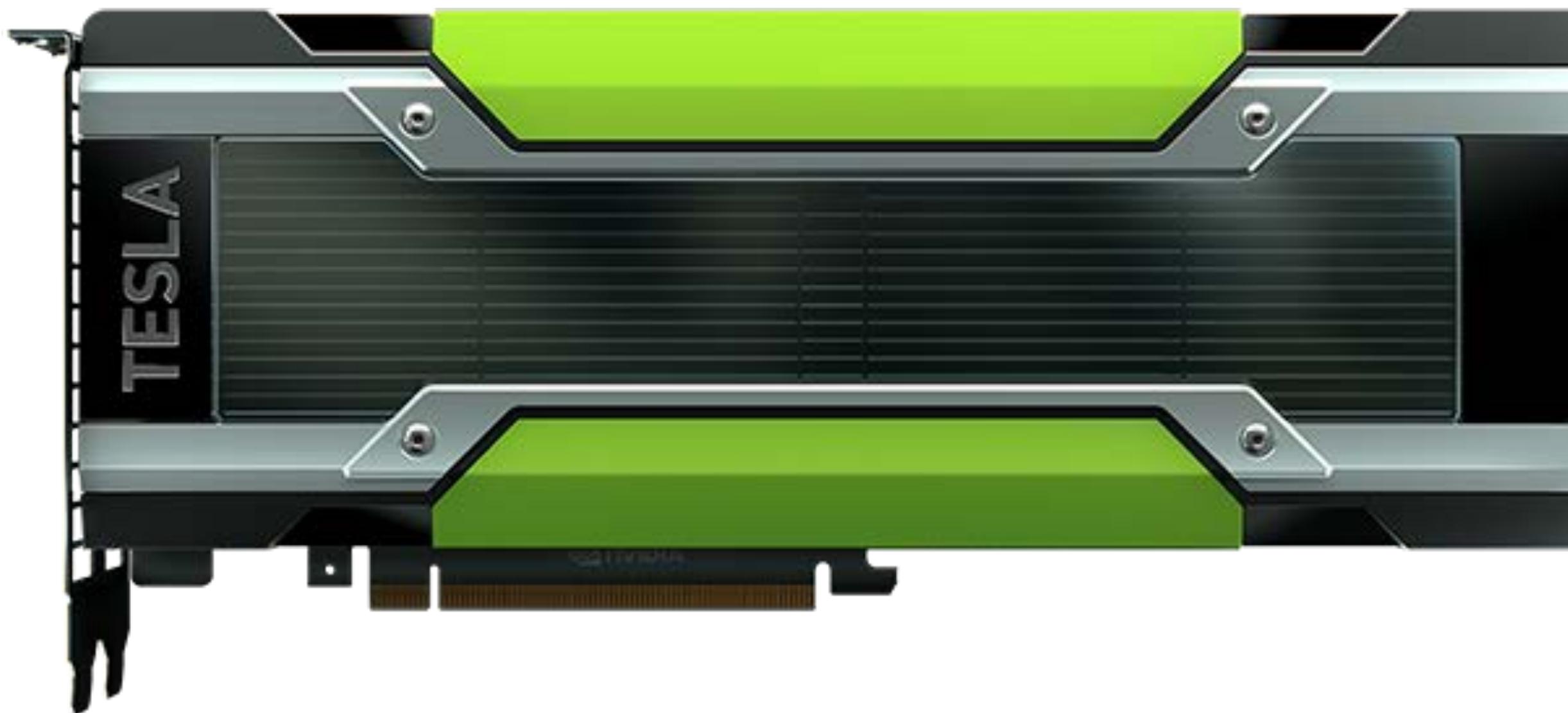
First of all, this machine learning course is incredible!
I can't wait to use what I've learned!

Unstructured

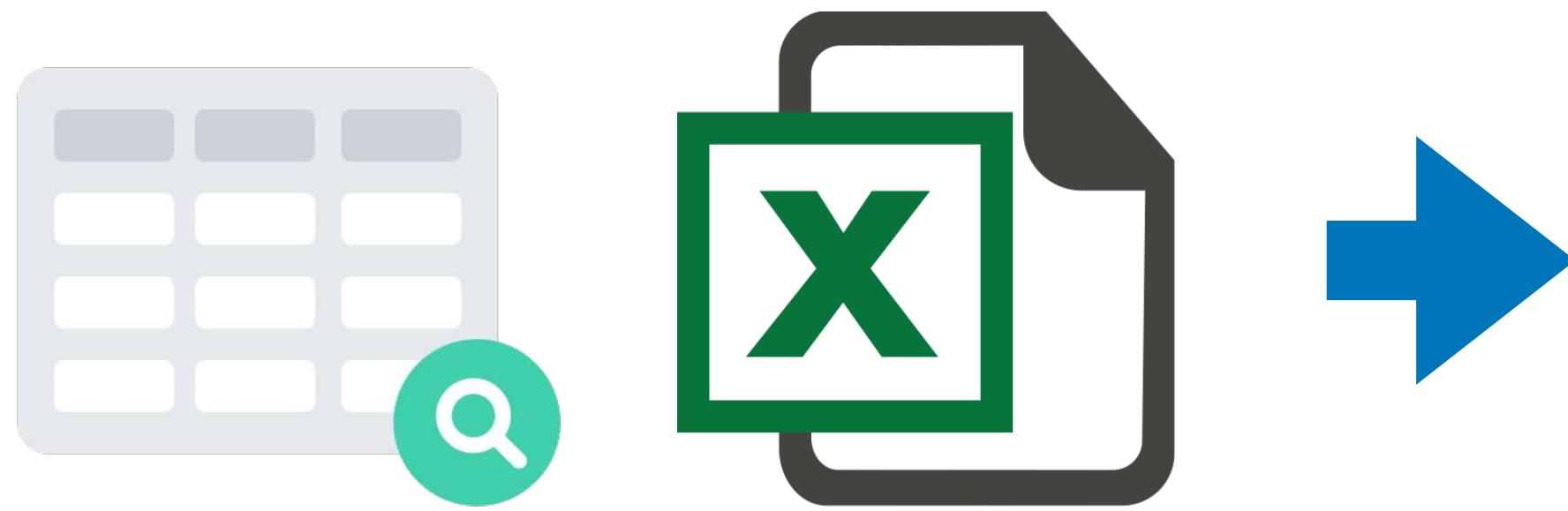
Why TensorFlow?

- 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)

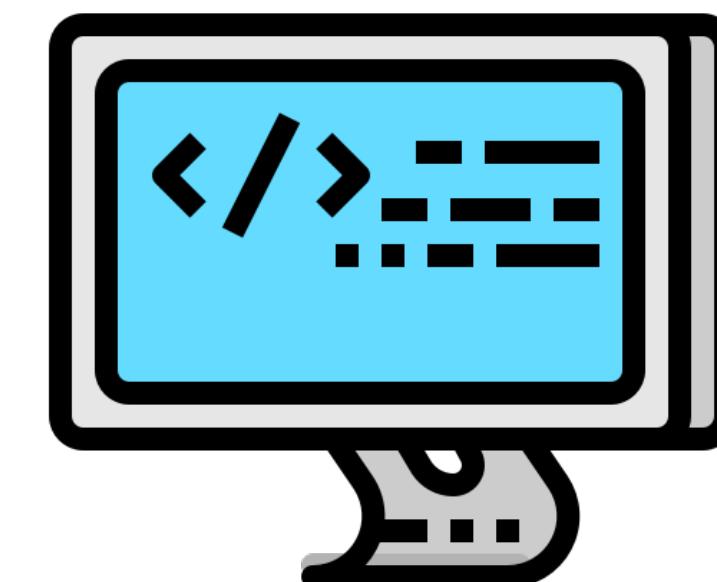
What is a GPU?



Choosing a model (throwback)



Problem 1 (structured)



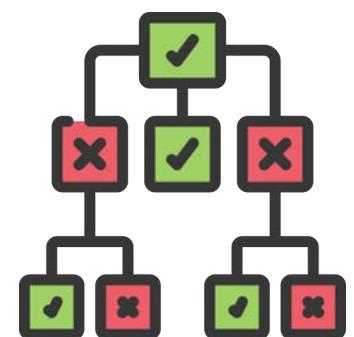
Model 1

Structured Data



CatBoost

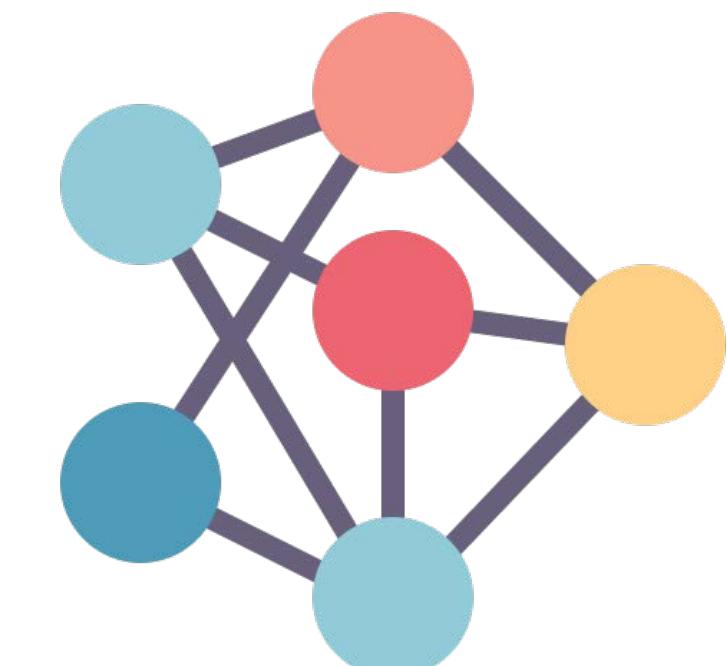
dmlc
XGBoost



Random Forest



Problem 2 (unstructured)



Model 2

Unstructured Data

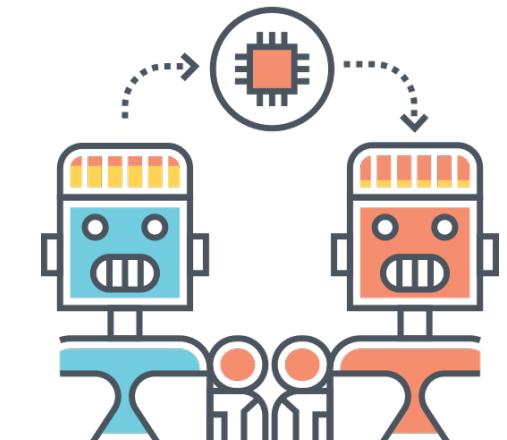


Deep Learning

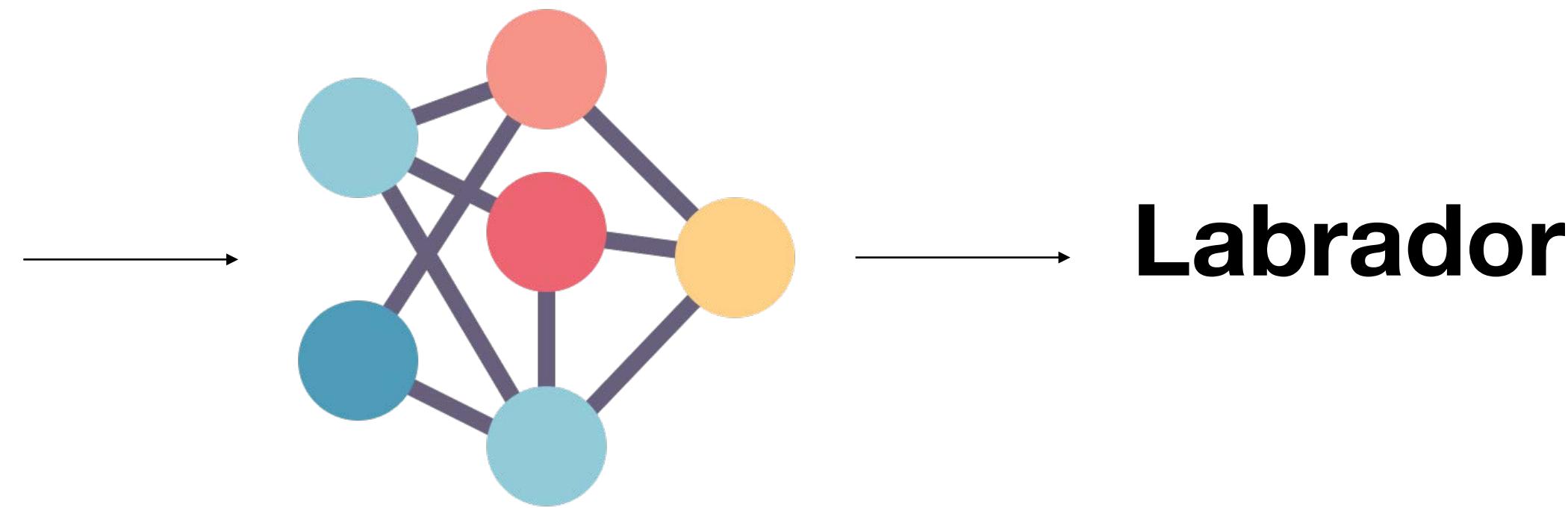
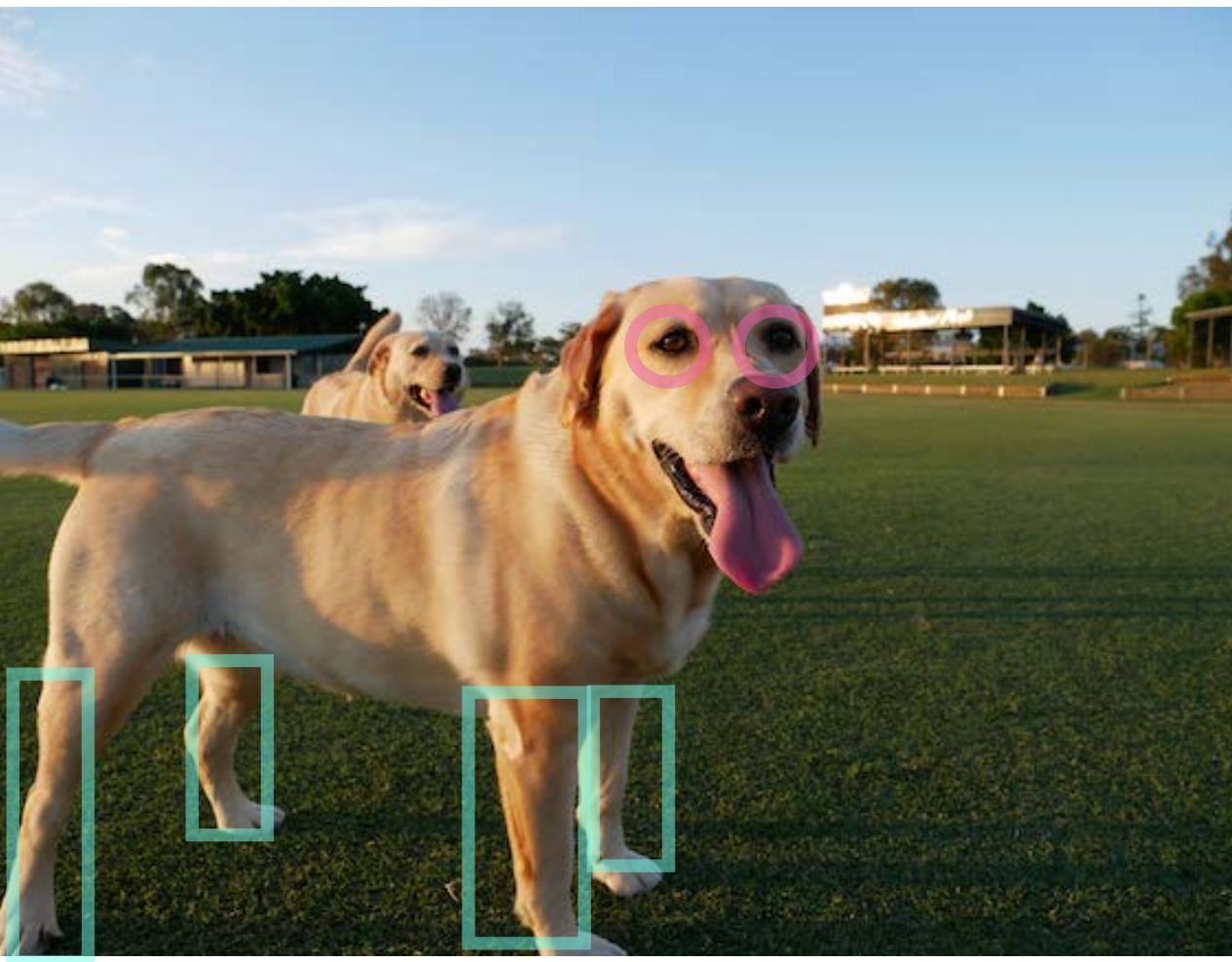


Transfer Learning

**TensorFlow
Hub**

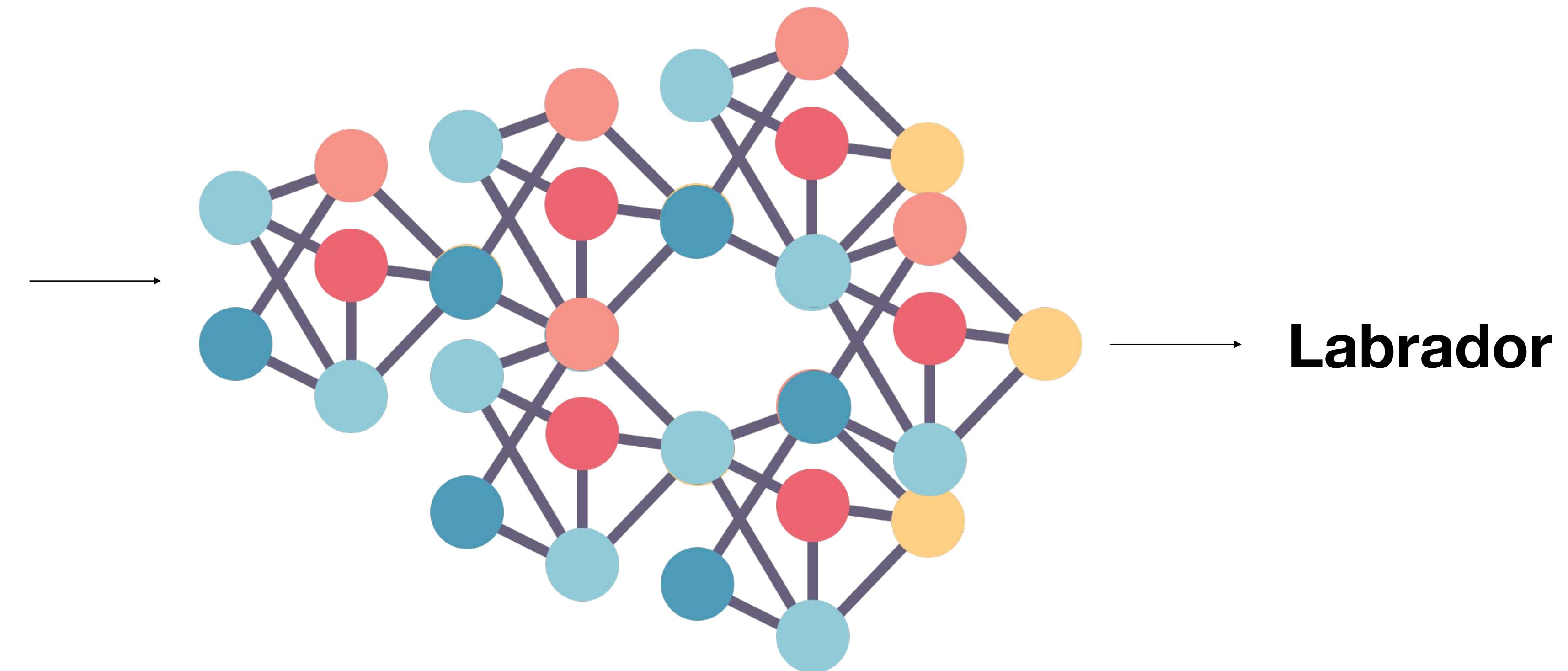
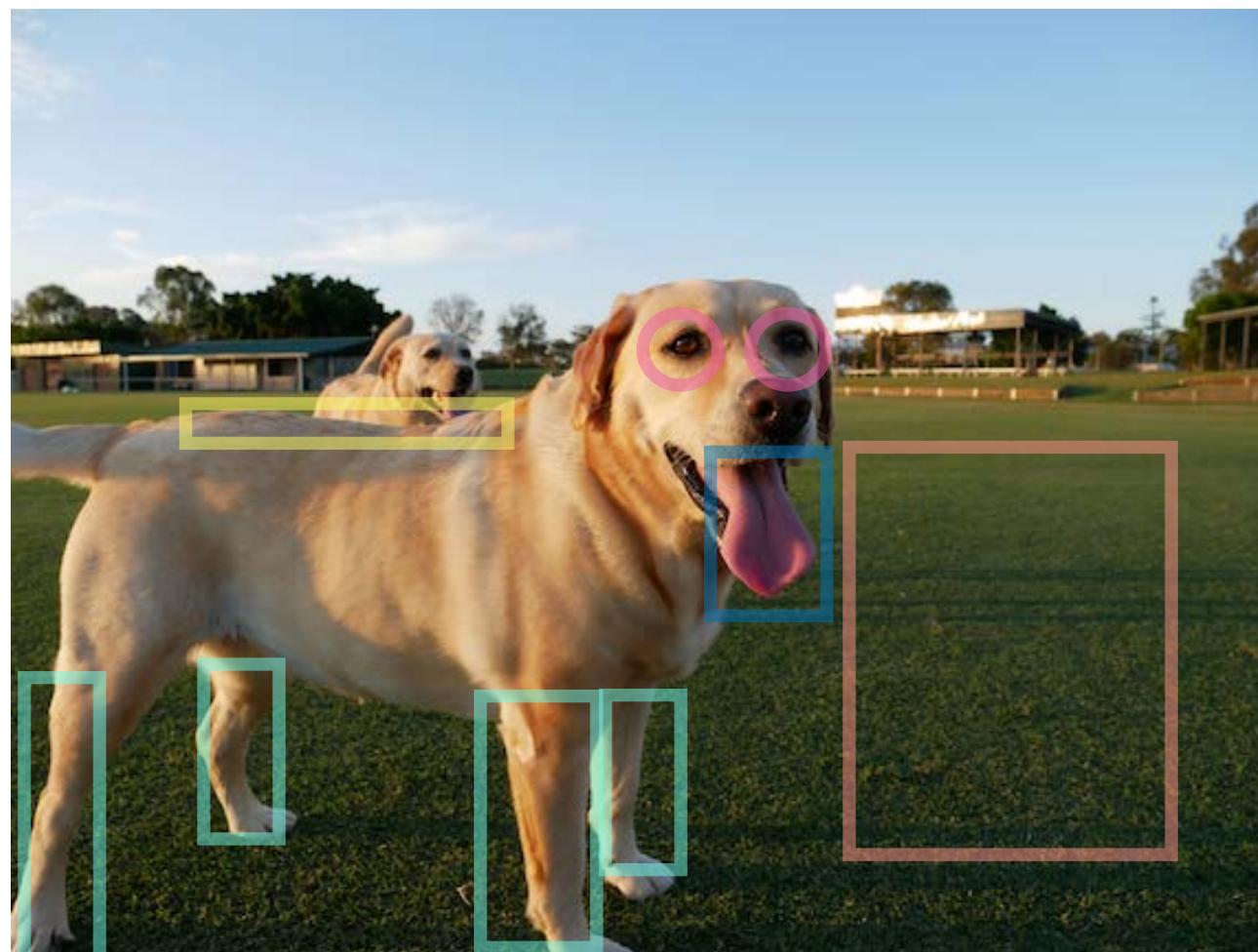


What is deep learning? What are neural networks?



Labrador

What is deep learning? What are neural networks?



What kind of deep learning problems are there?



From: daniel@mrbourke.com

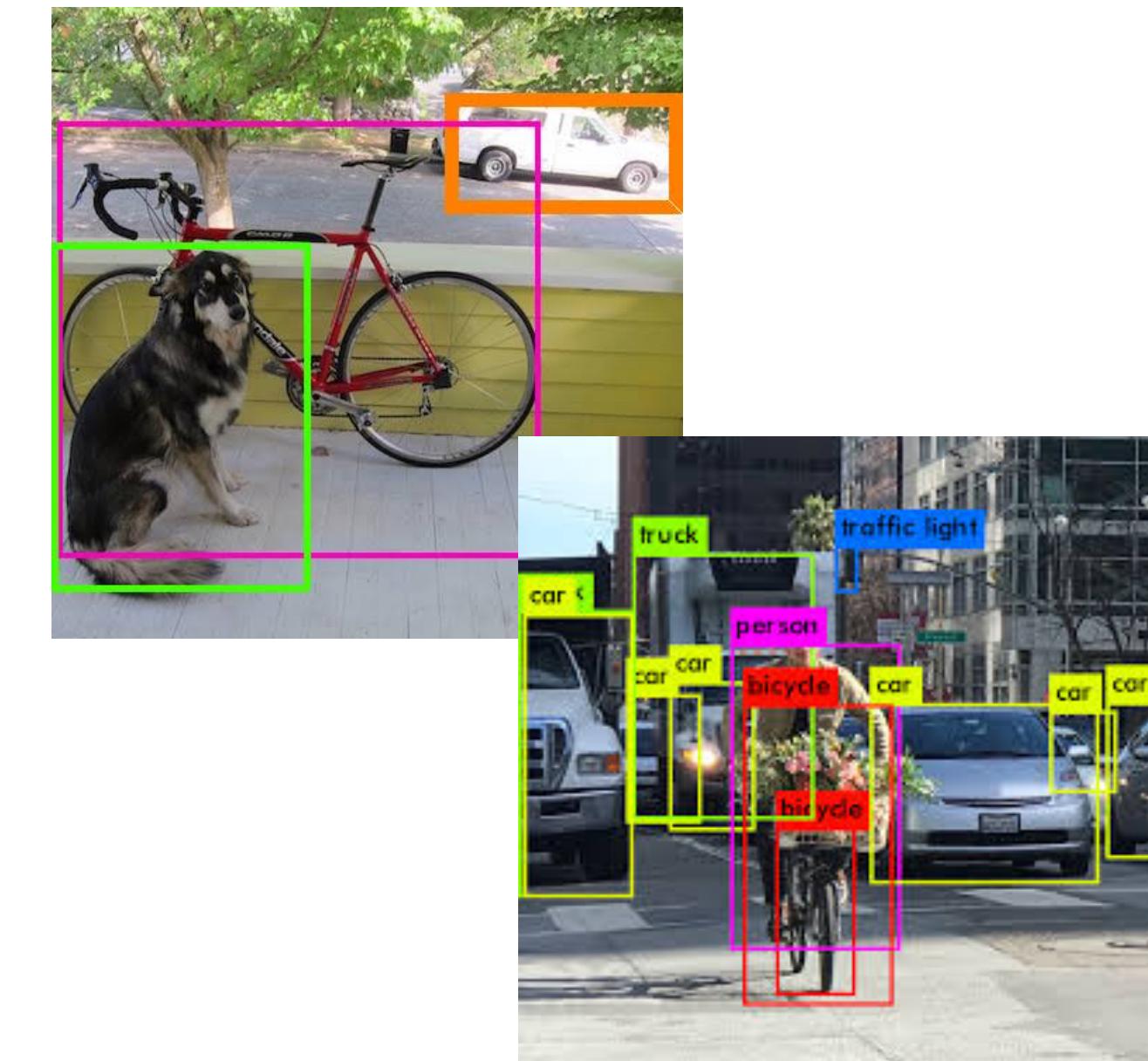
Hey Daniel,

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

Classification



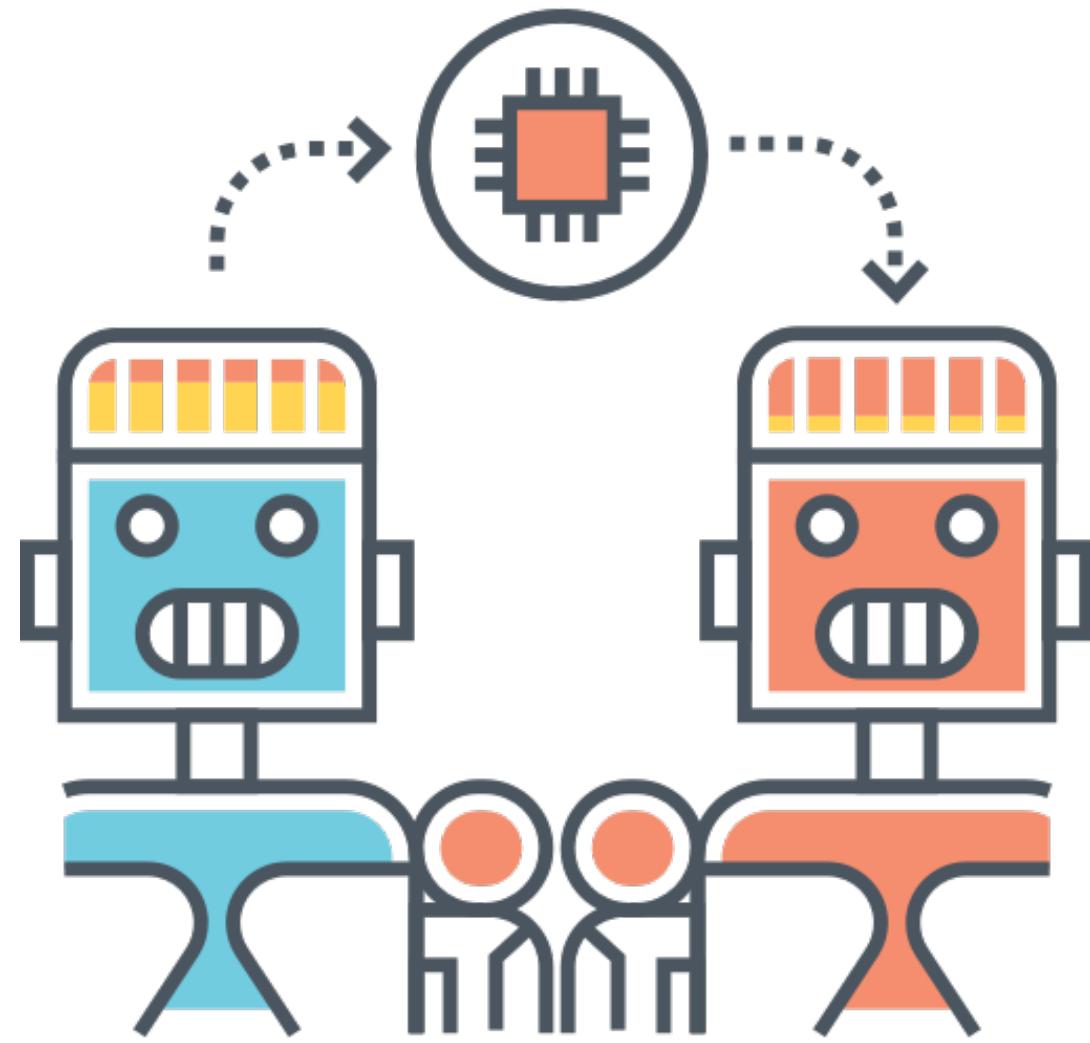
Hey Siri, where's the nearest cafe?



**Sequence to sequence
(seq2seq)**

Object detection

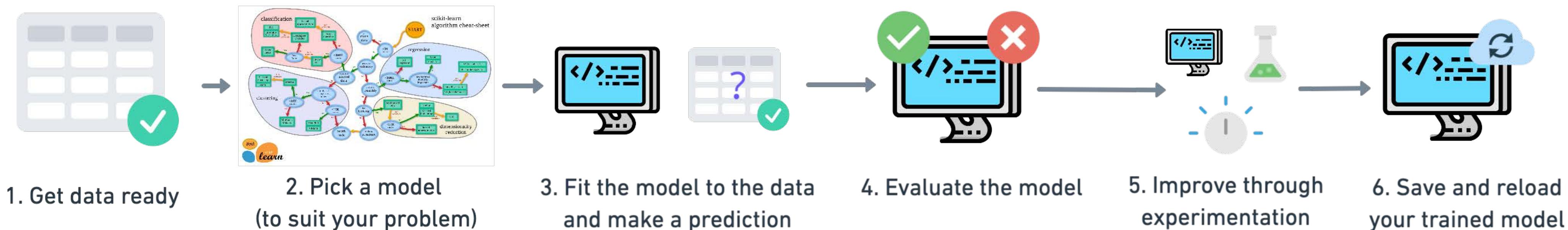
What is transfer learning? Why use transfer learning?



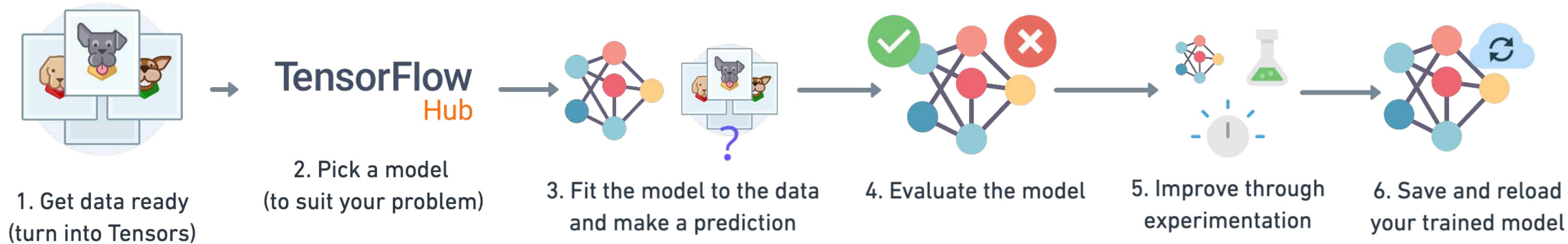
- Take what you know in one domain and apply it to another.
- Starting from scratch can be expensive and time consuming.
- Why not take advantage of what's already out there?

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



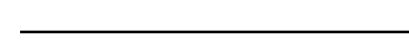
Classification

- “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 features a large image of a dog's head and the title "Dog Breed Identification: Determine the breed of a dog in an image". Below the title, it says "Kaggle · 1,282 teams · 2 years ago". The navigation bar includes "Overview", "Data", "Notebooks", "Discussion", "Leaderboard", "Rules", "Team", "My Submissions", and "Late Submission". The "Late Submission" button is highlighted in blue. The "Description" section contains the text: "Who's a good dog? Who likes ear scratches? Well, it seems those fancy deep neural networks don't have all the answers. However, maybe they can answer that ubiquitous question we all ask when meeting a four-legged stranger: what kind of good pup is that?". The "Evaluation" section contains the text: "In this playground competition, you are provided a strictly canine subset of ImageNet in".

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 Airbnb 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



```
# 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.Sequential(*args, **kwargs),
                      Linear stack of layers.
                      Arguments:
                        layers: list of layers to add to the model.
                        Example:
                          model = Sequential()
                          model.add(Dense(32, input_shape=(500,)))
                          # Optionally, the first layer can receive an `input_shape` argument:
                          model.add(Dense(32, input_shape=(500,)))
        )
    ])

    # Compile the model
    model.compile(
        loss=tf.keras.losses.categorical_crossentropy,
        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

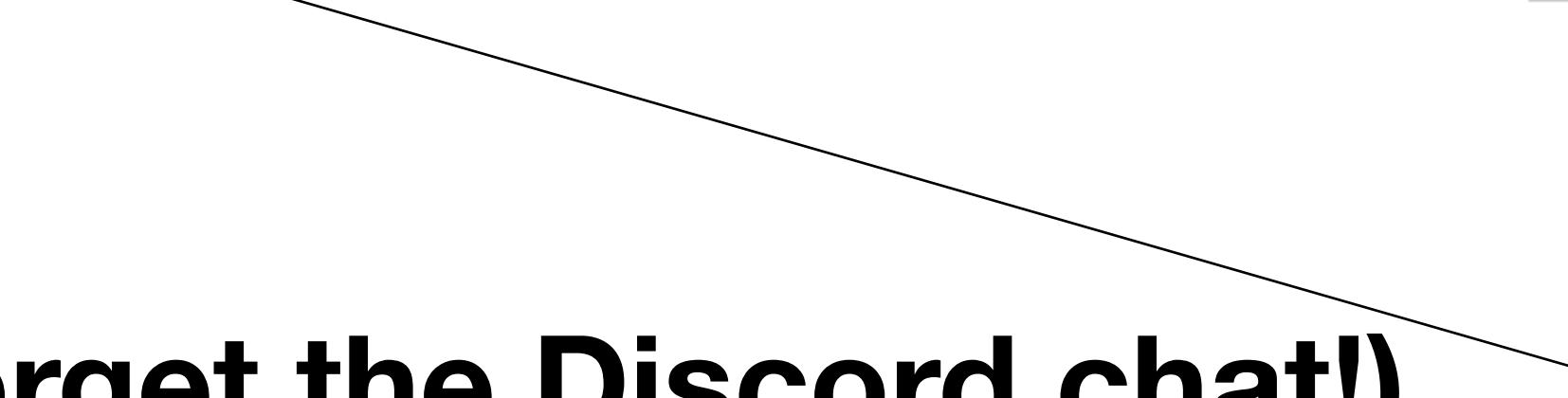
- Search for it



- Try again



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



TensorFlow Core

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

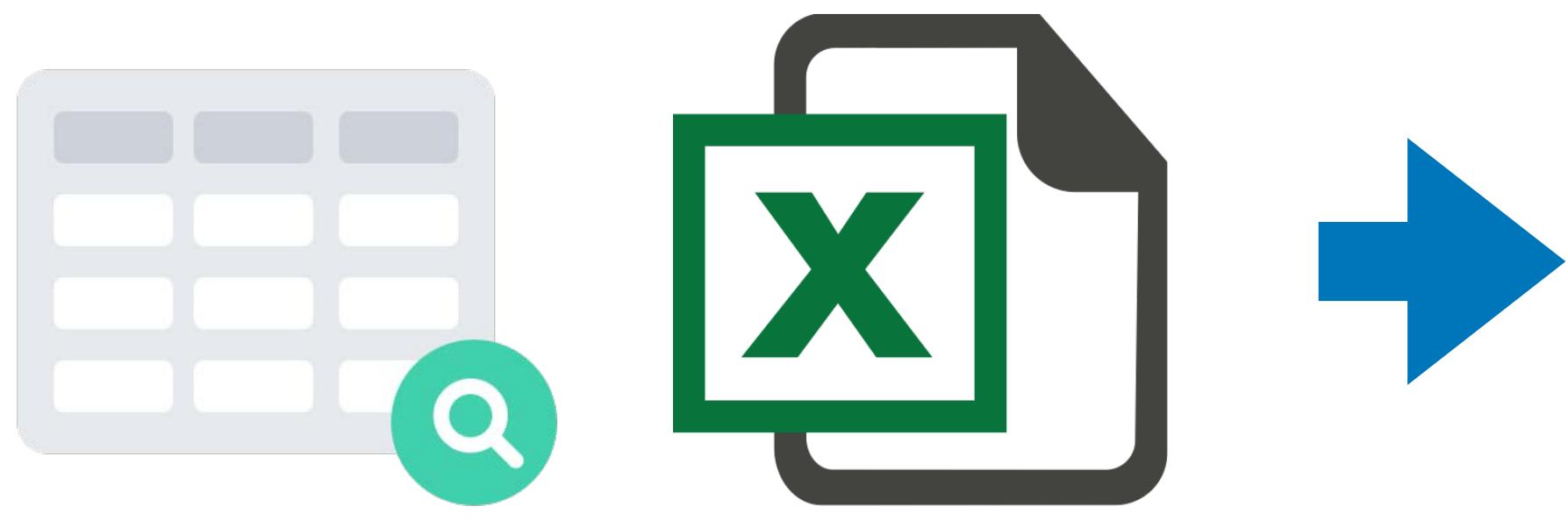
Install TensorFlow TensorFlow 2 Keras

TensorFlow 2 best practices and tools to Keras is a high-level API that's easier for

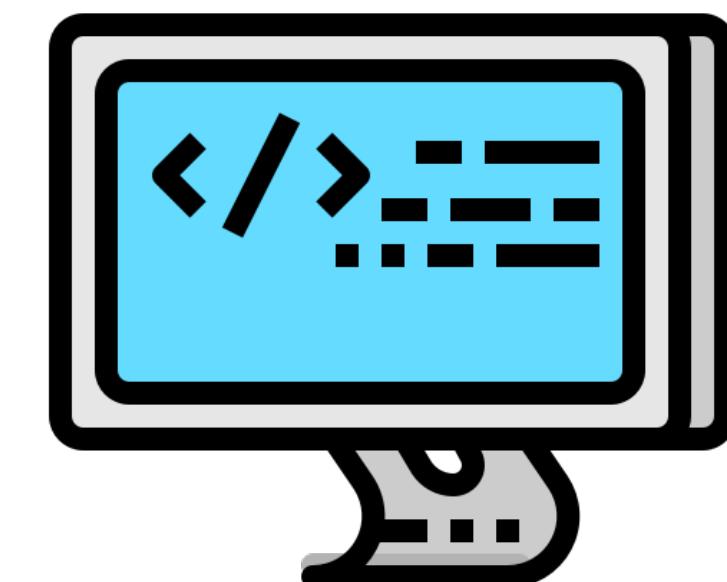
Let's find those doggos!



Choosing a model (throwback)



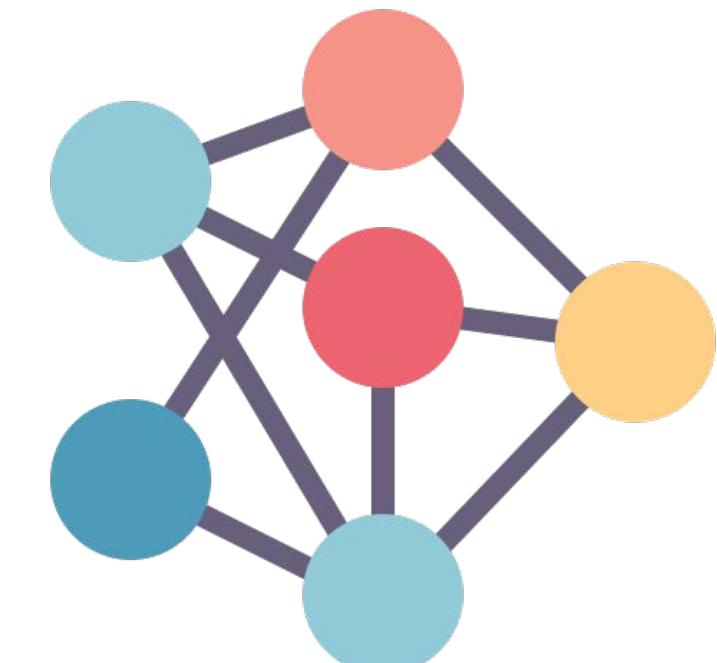
Problem 1 (structured)



Model 1

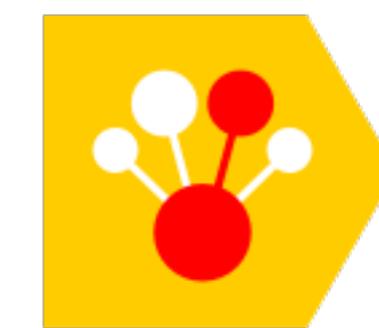


Problem 2 (unstructured)

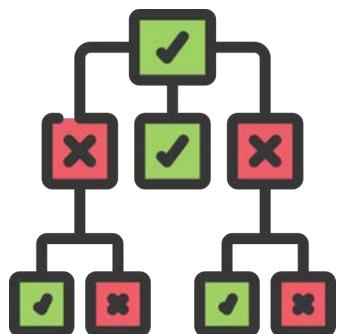


Model 2

Structured Data



dmlc
XGBoost



Random Forest

Unstructured Data

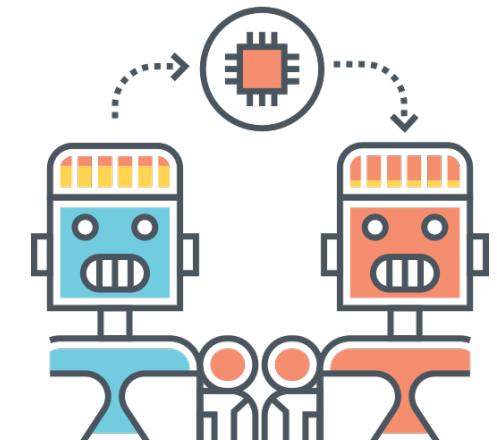


Deep Learning

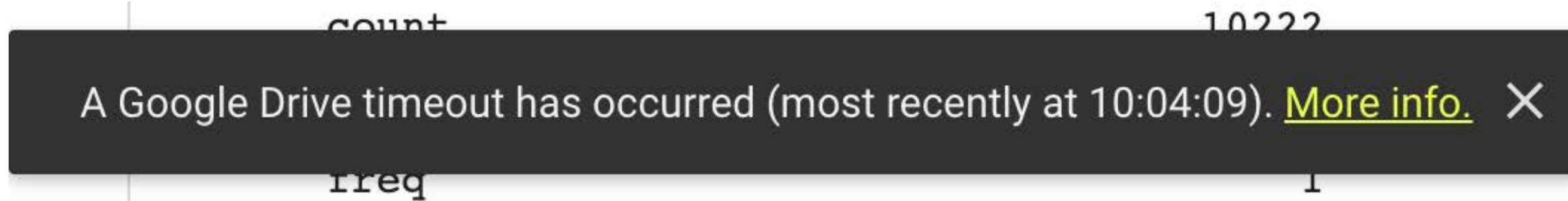


Transfer Learning

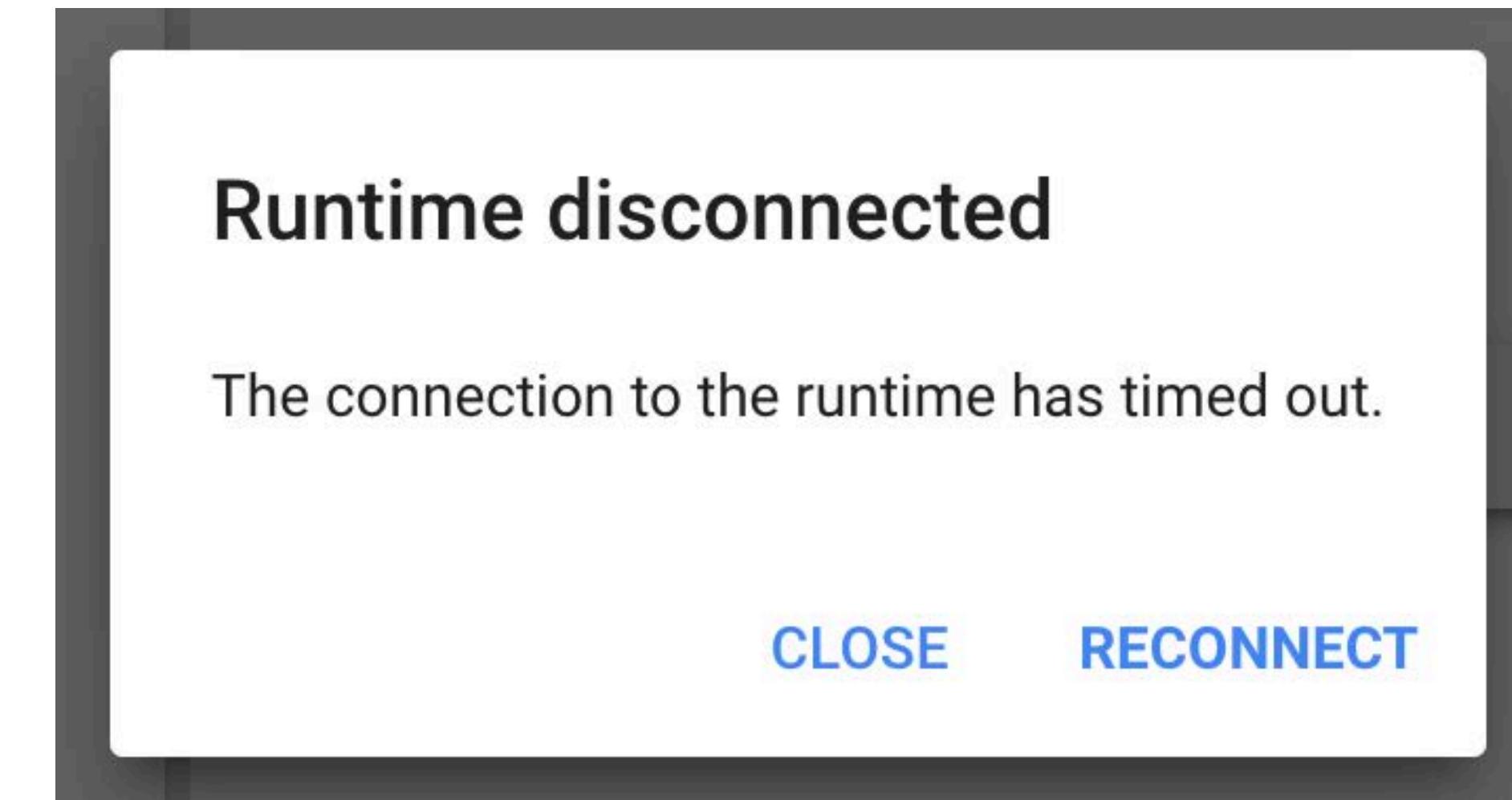
TensorFlow
Hub



Things you might see in Colab



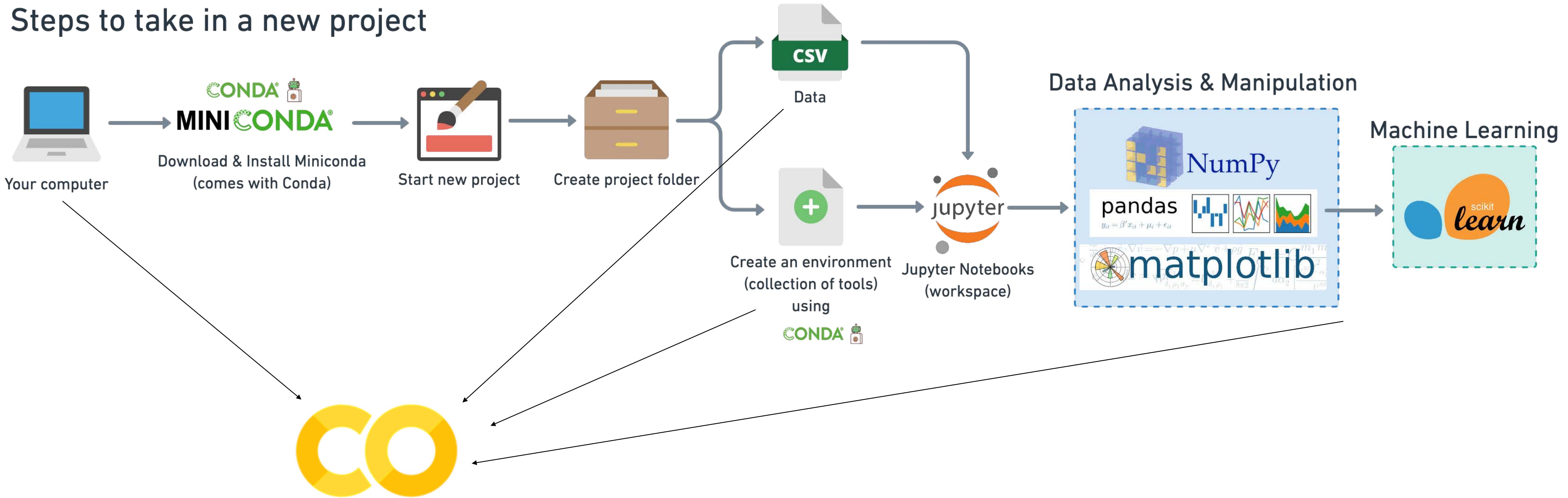
Usually fixes itself, may need to reconnect Google Drive to Colab.



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).

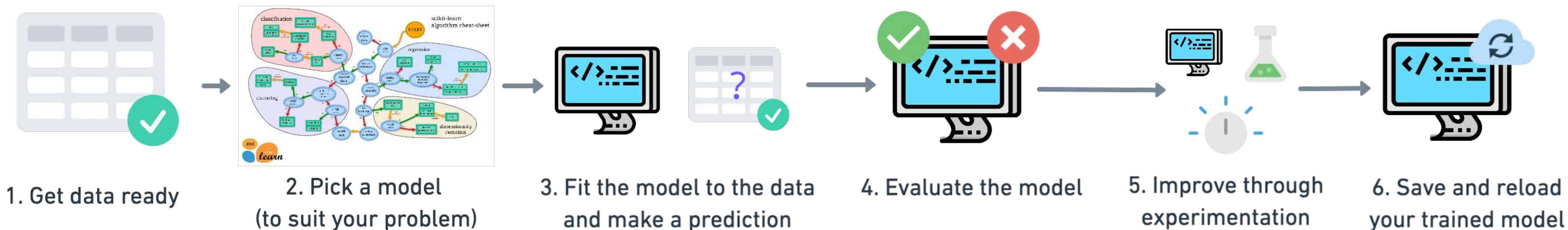
For more information and fixes, refer to the Google Colab FAQ: <https://research.google.com/colaboratory/faq.html>

Steps to take in a new project

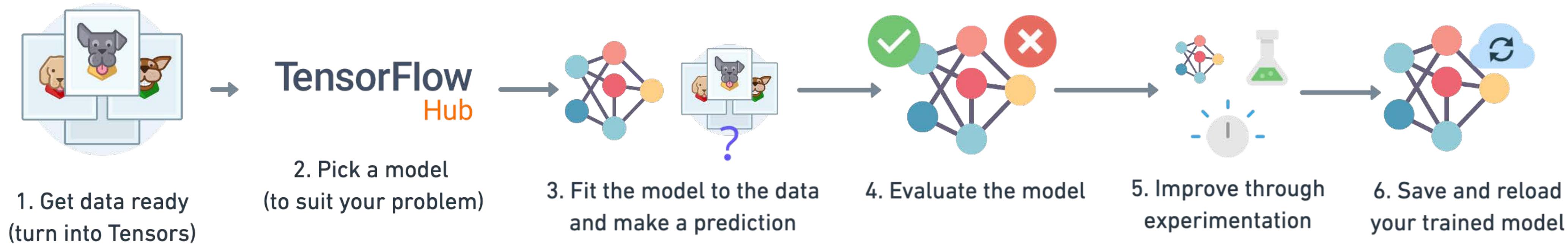


What are we going to cover?

A Scikit-Learn workflow

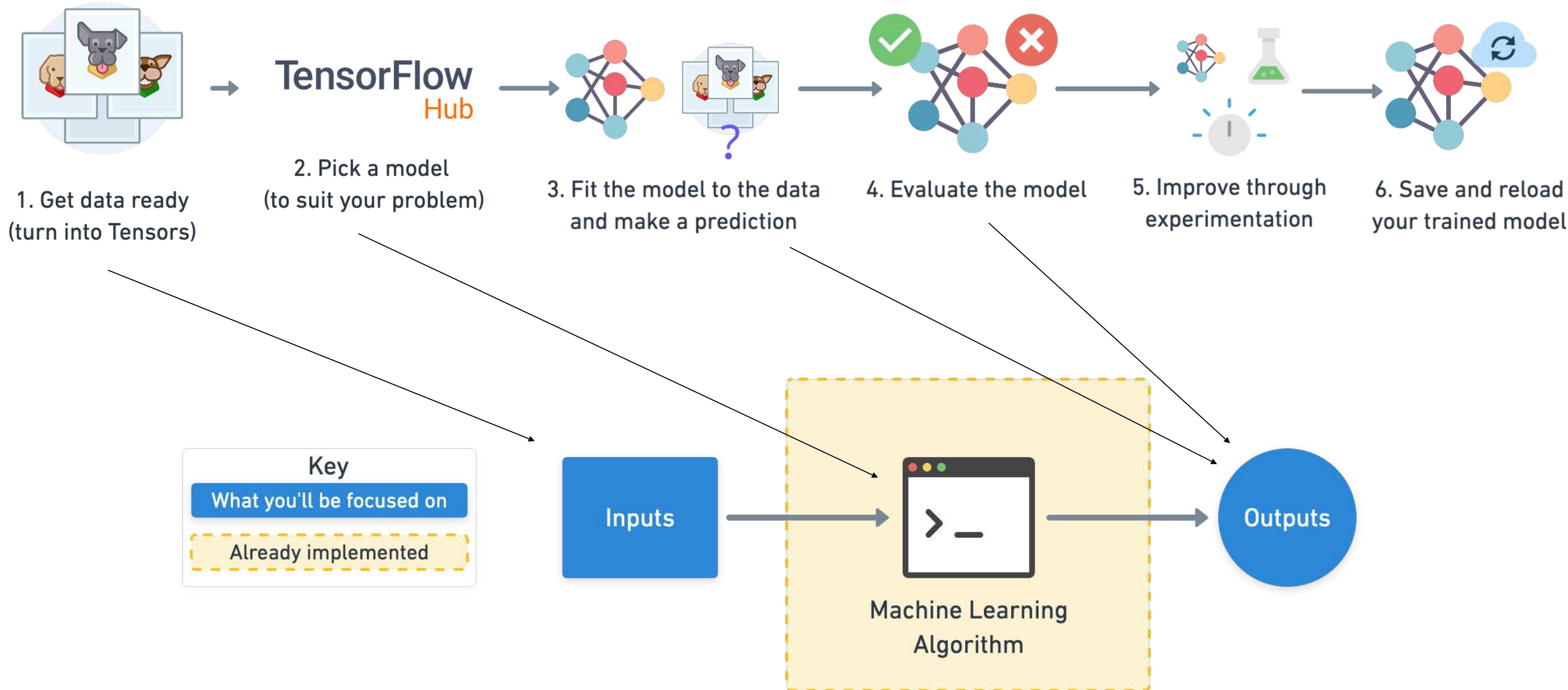


A TensorFlow workflow



What we're focused on

A TensorFlow workflow



Which activation? Which loss?

Binary classification

Activation: Sigmoid

Multi-class classification

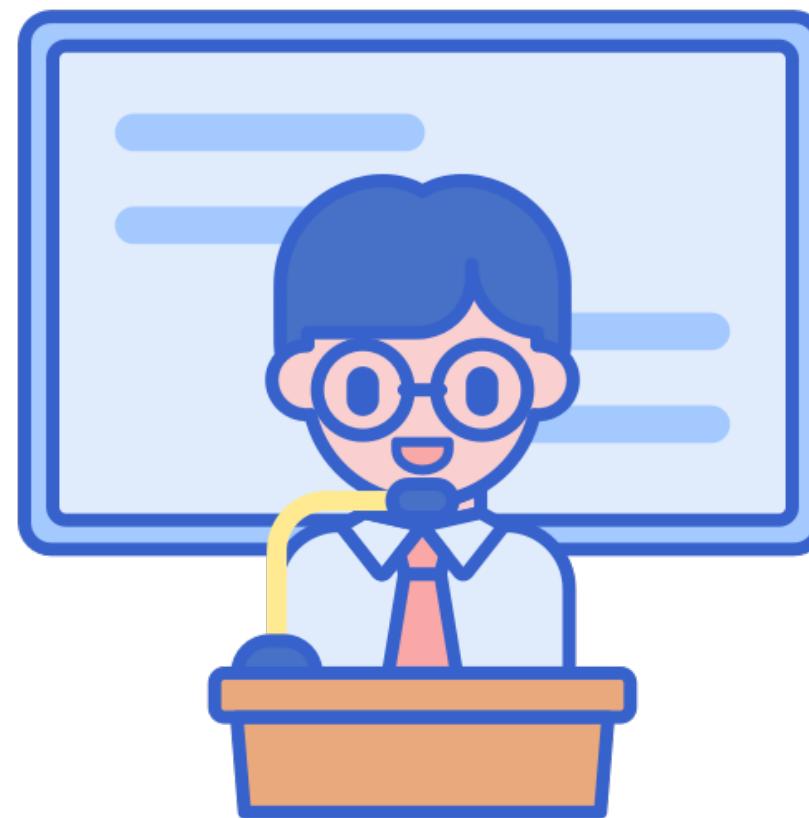
Activation: Softmax

Loss: Binary Crossentropy

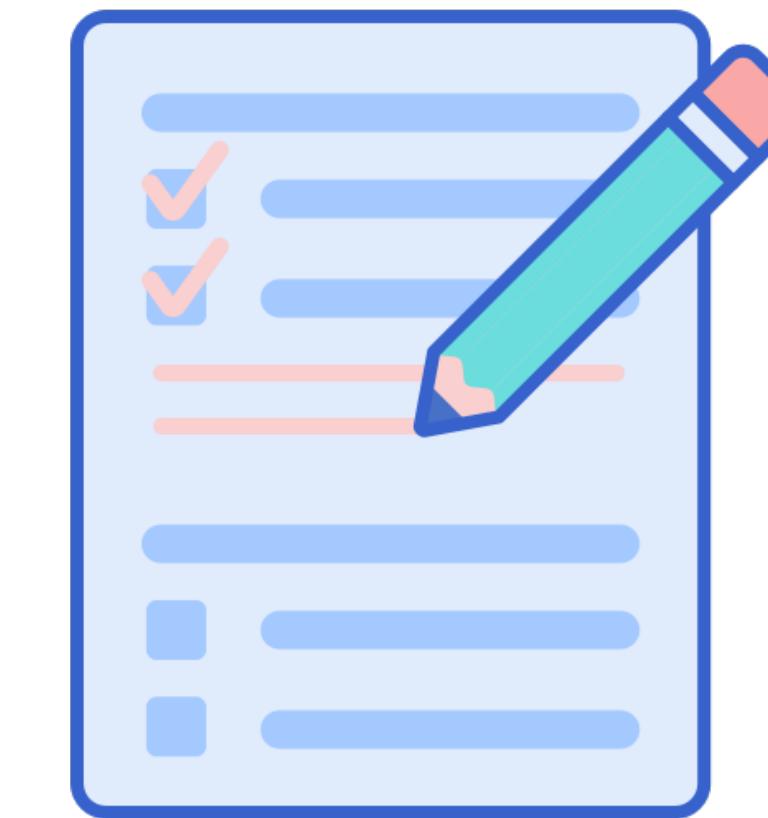
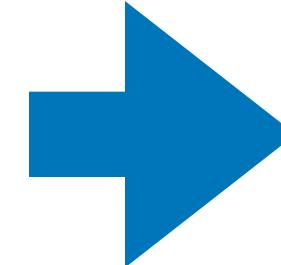
Loss: Categorical Crossentropy

The most important concept in machine learning

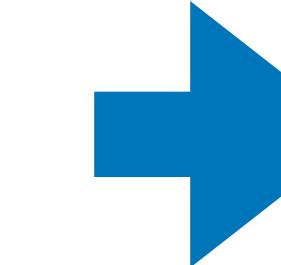
(the 3 sets)



Course materials
(training set)



Practice exam
(validation set)

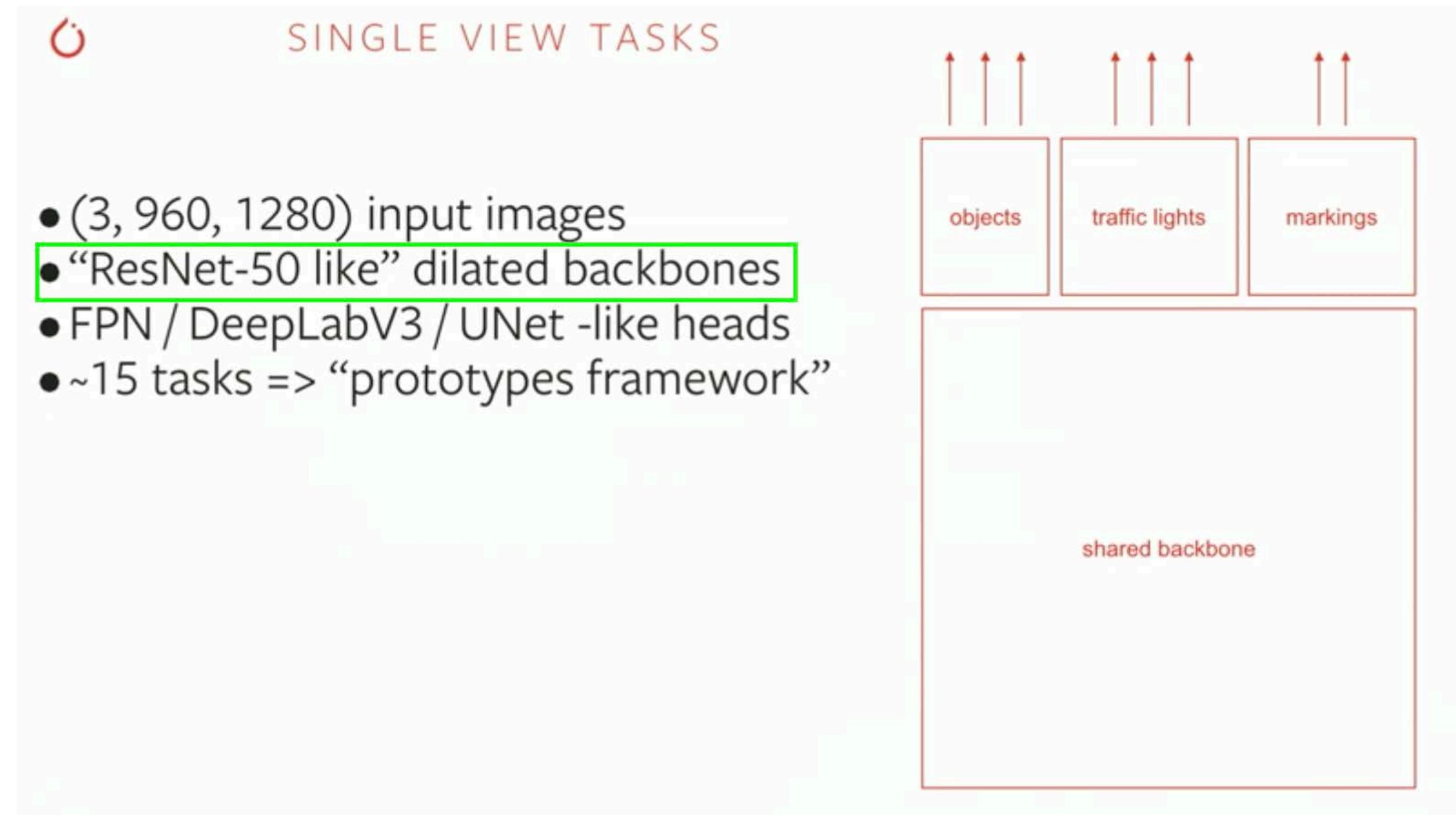


Final exam
(test set)

Generalization

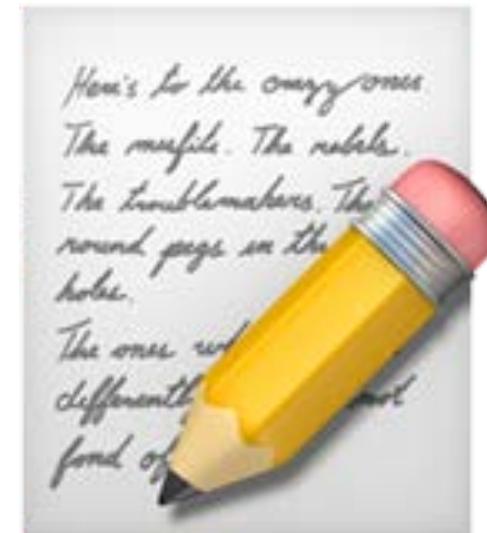
The ability for a machine learning model to perform well on data it hasn't seen before.

Tesla using ResNet50 backbones



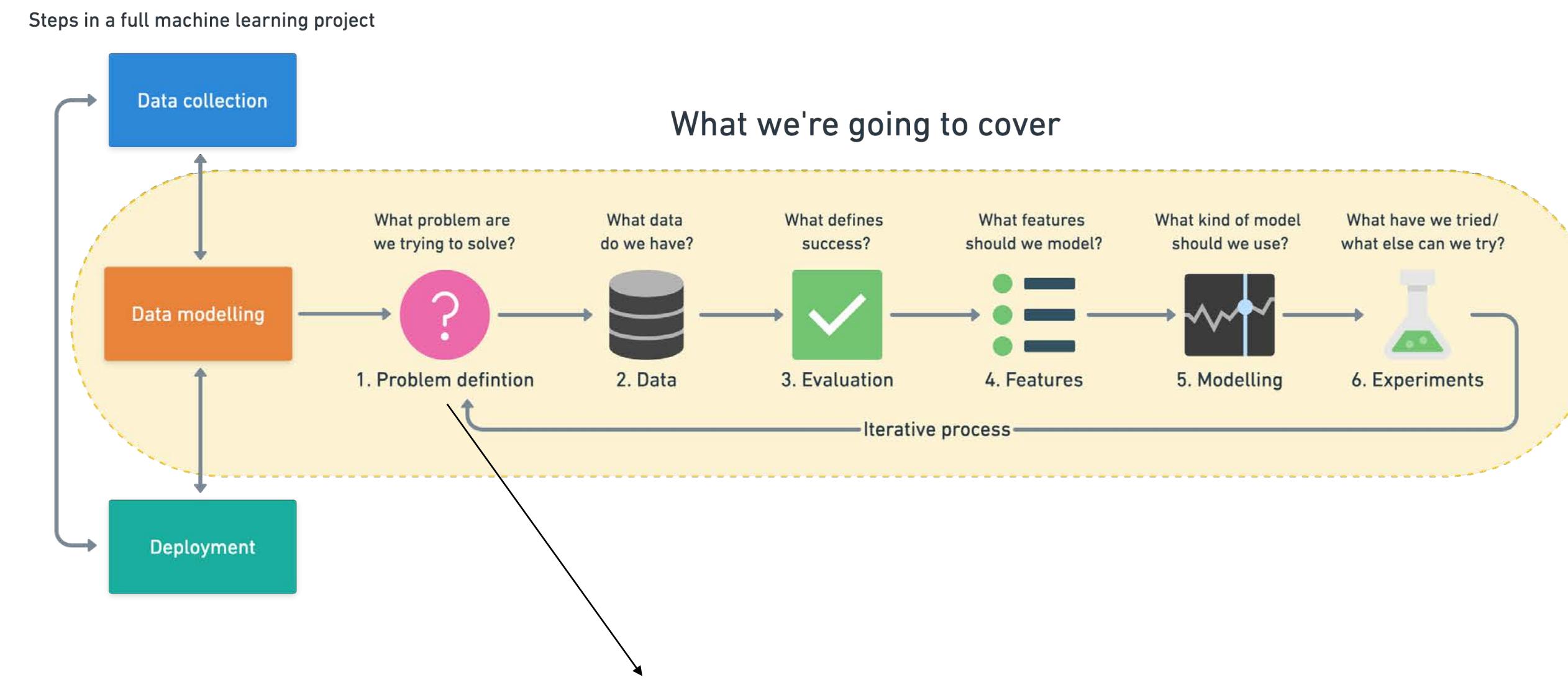
Source: “PyTorch at Tesla by Andrei Karpathy”
<https://youtu.be/oBklltKXtDE?t=173>

How to think about communicating and sharing your work



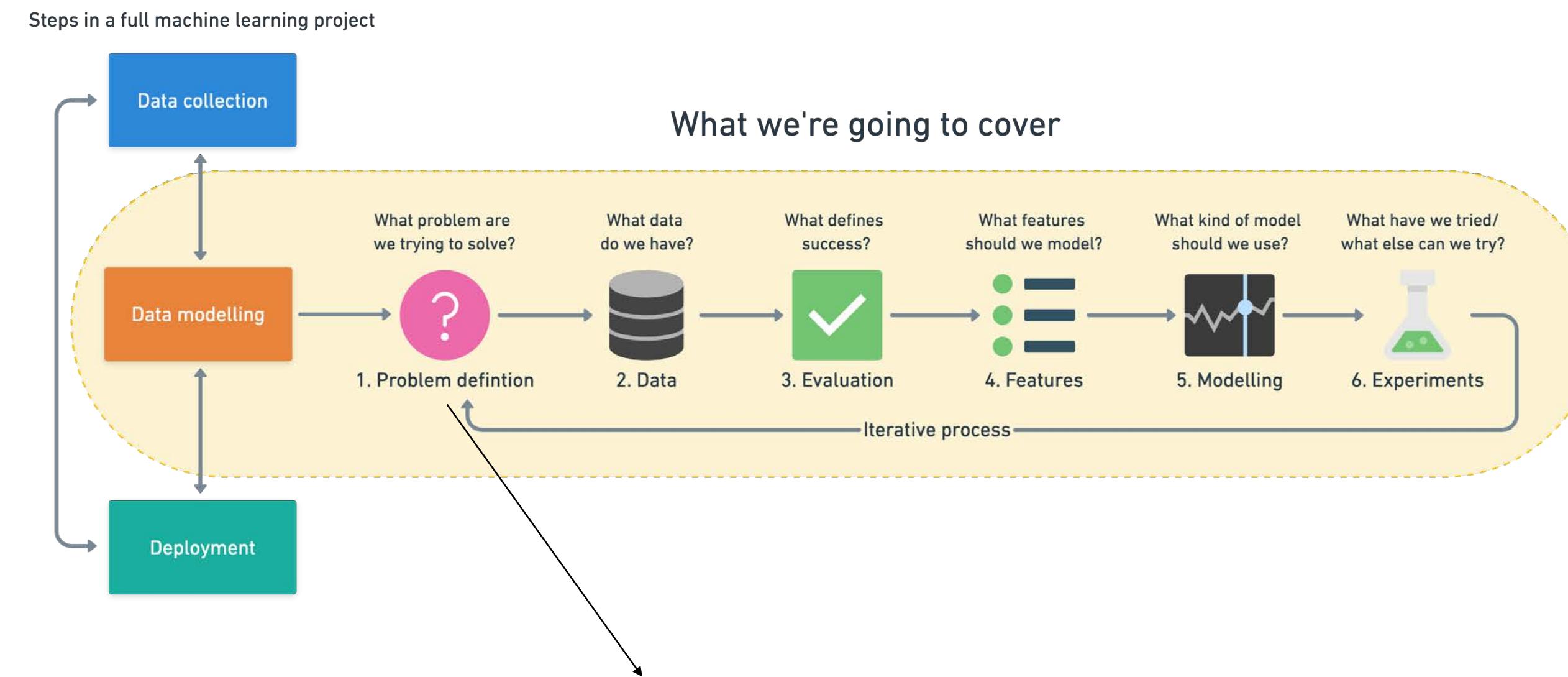
Now you've got skills, what do you do next?

The most important question you can ask



“Who’s it for?”

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

People on your team	People outside your team
Boss	Clients
Project manager	Customers
Teammates	Fans

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

- 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
- 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

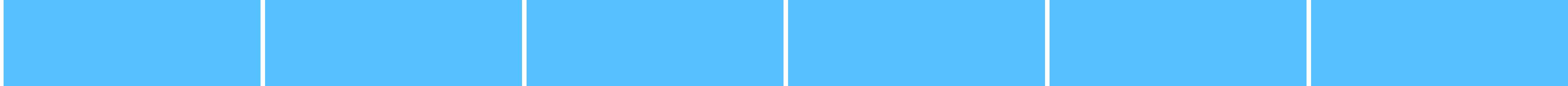


The People You're Working With, Sitting Next to, in the Group Chat

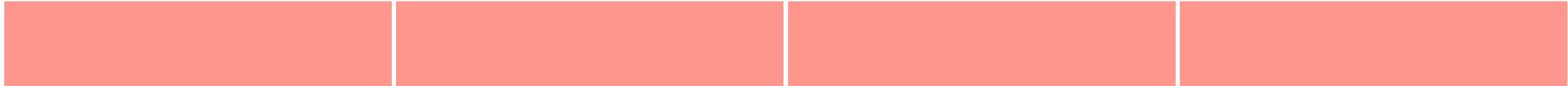


Break it down

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?

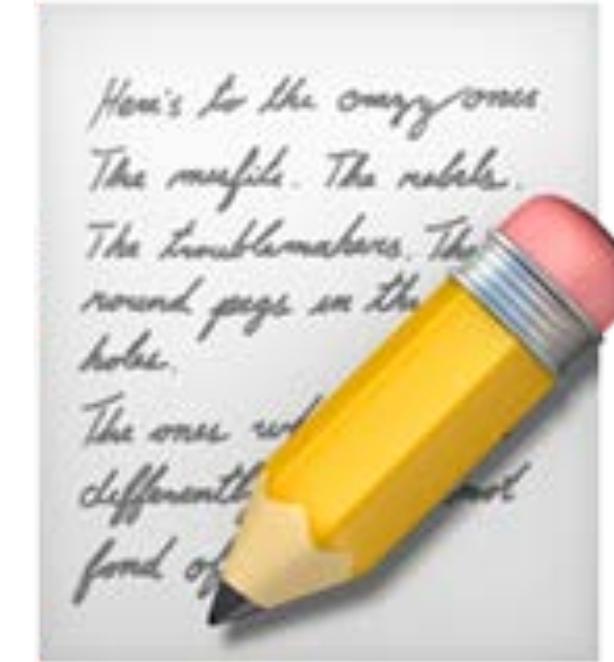
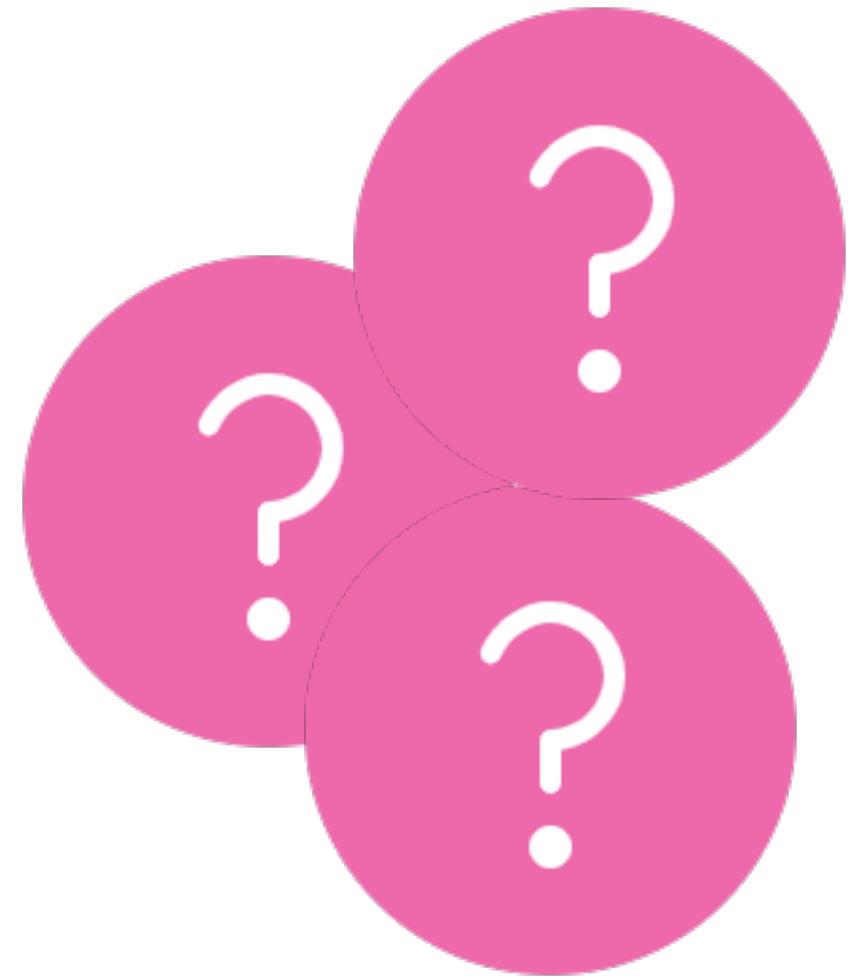
What I'm working on next:

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

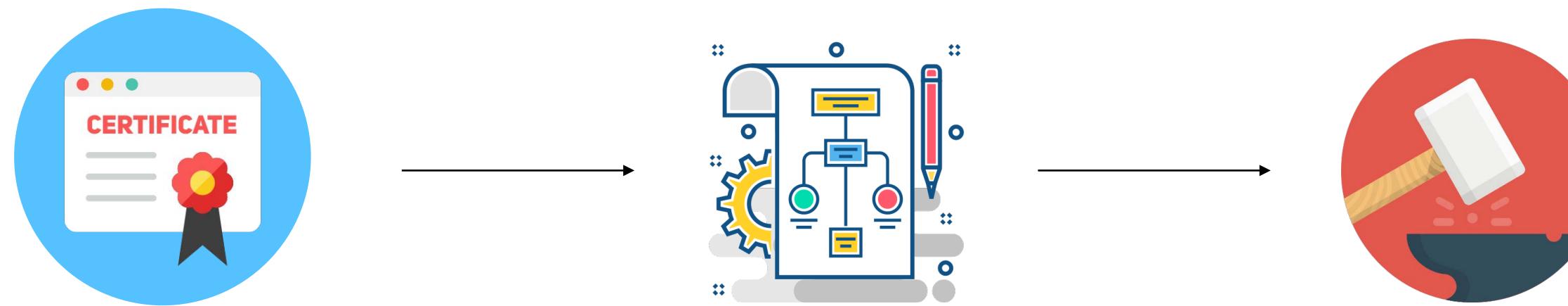


Relate back to overall project goal

Take note of overlaps



Progress, not perfection



“Here’s what I’ve done.”