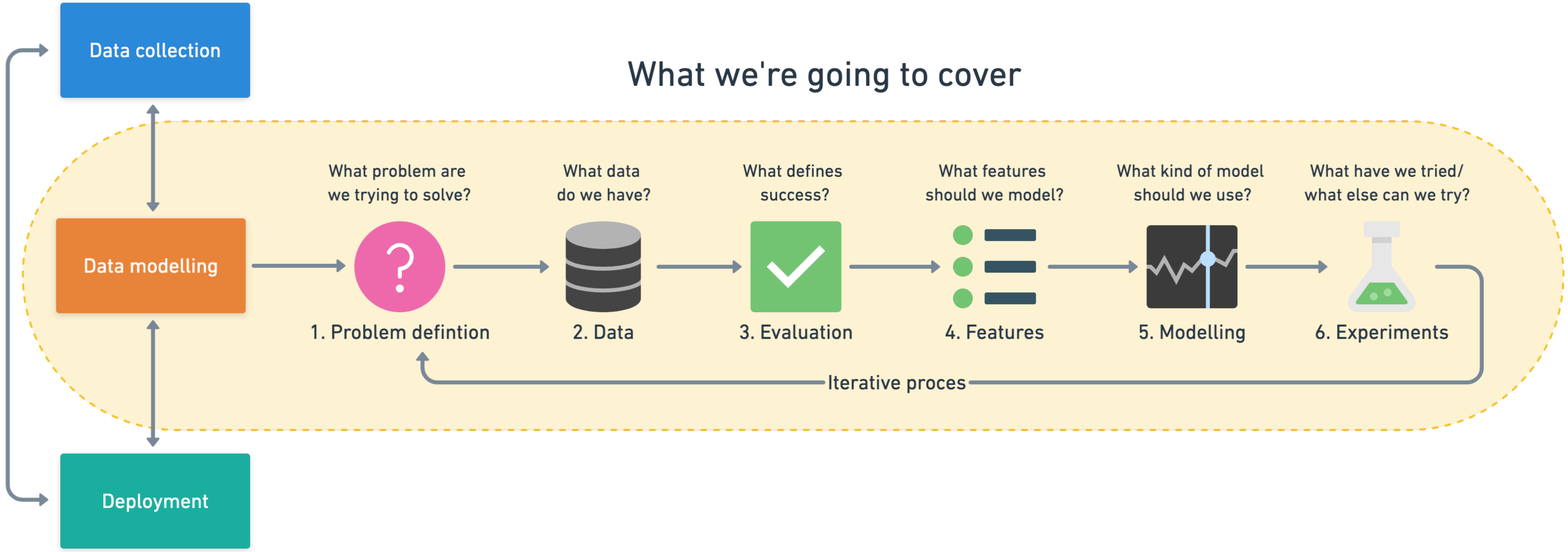
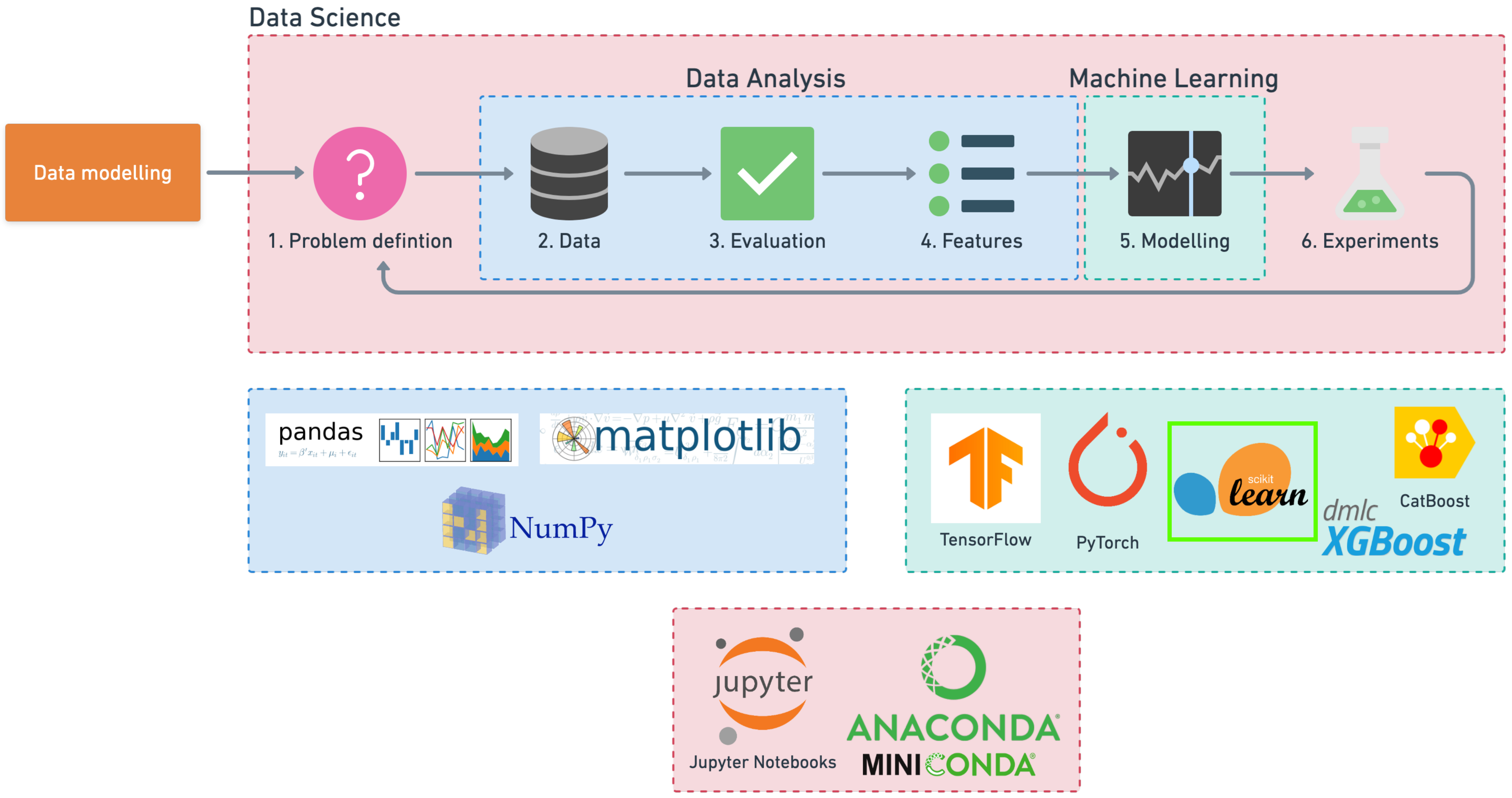
Steps in a full machine learning project

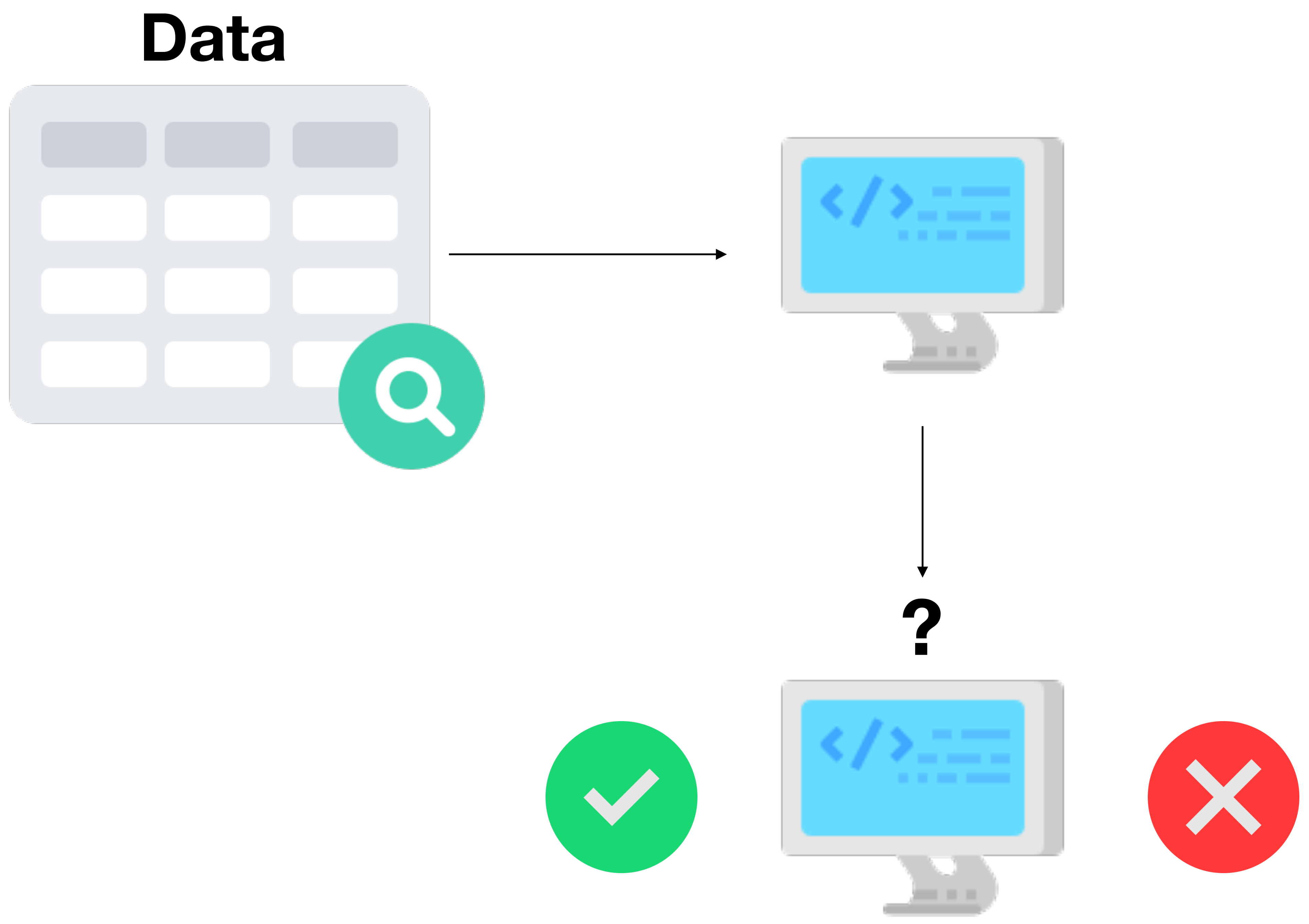
## What we're going to cover



# Tools you can use



# What is Scikit-Learn (sklearn)?

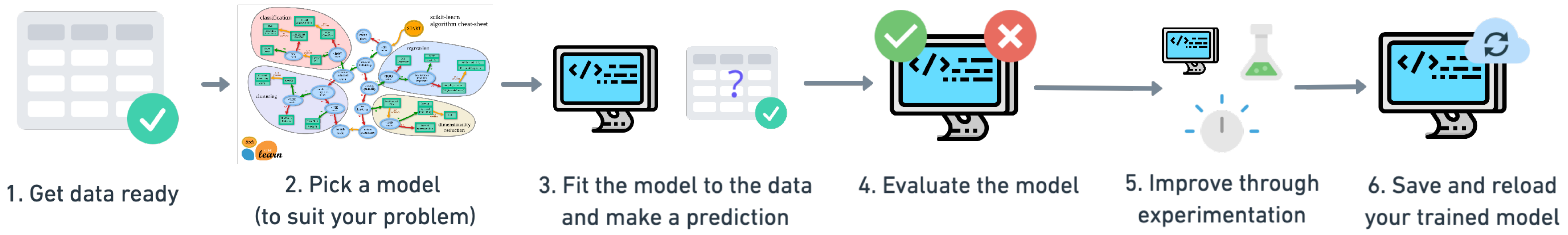


## Why Scikit-Learn?

- 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

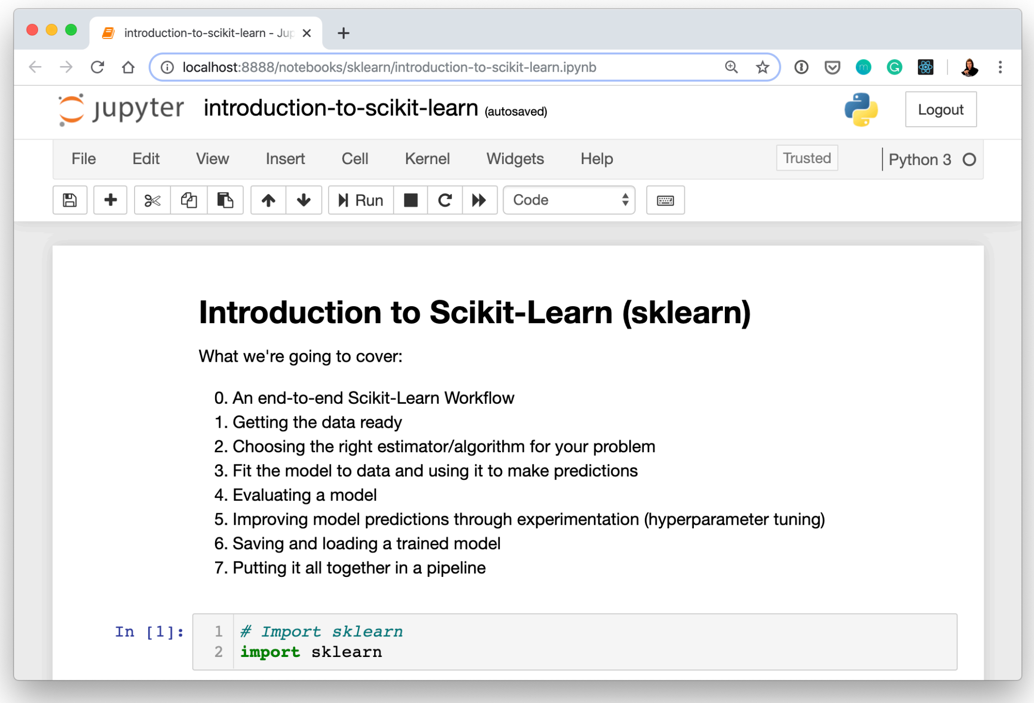

1. Get data ready
2. Pick a model (to suit your problem)
3. Fit the model to the data and make a prediction
4. Evaluate the model
5. Improve through experimentation
6. Save and reload your trained model
scikit-learn algorithm cheat-sheet
classification
regression
clustering
dimensionality reduction
START
learn

# 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
- Try it for yourself
- Press SHIFT + TAB to read the docstring
- Search for it
- Try again
- Ask

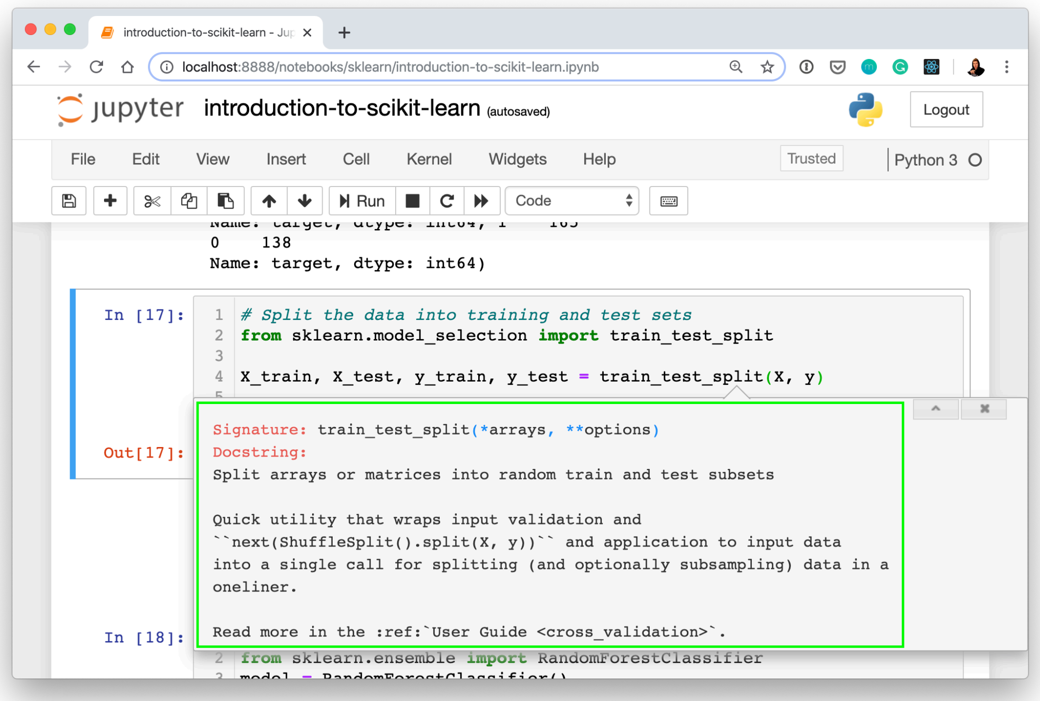

introduction-to-scikit-learn - Jup
localhost:8888/notebooks/sklearn/introduction-to-scikit-learn.ipynb
jupyter introduction-to-scikit-learn (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

## Introduction to Scikit-Learn (sklearn)

What we're going to cover:

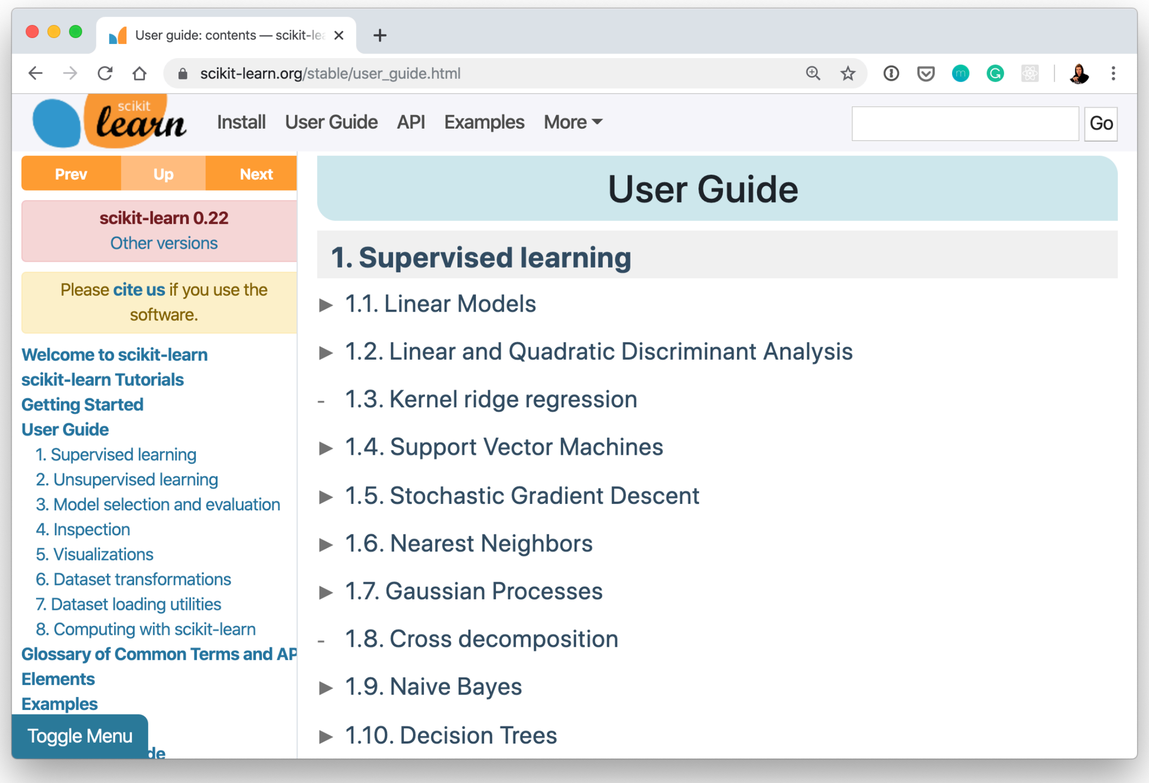
0. An end-to-end Scikit-Learn Workflow
1. Getting the data ready
2. Choosing the right estimator/algorithm for your problem
3. Fit the model to data and using it to make predictions
4. Evaluating a model
5. Improving model predictions through experimentation (hyperparameter tuning)
6. Saving and loading a trained model
7. Putting it all together in a pipeline

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


introduction-to-scikit-learn - Jup
localhost:8888/notebooks/sklearn/introduction-to-scikit-learn.ipynb
jupyter introduction-to-scikit-learn (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

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

In [17]: # Split the data into training and test sets
         from sklearn.model_selection import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(X, y)

Out[17]:

Signature: train_test_split(*arrays, **options)
Docstring:
Split arrays or matrices into random train and test subsets

Quick utility that wraps input validation and
``next(ShuffleSplit().split(X, y))`` and application to input data
into a single call for splitting (and optionally subsampling) data in a
oneliner.

Read more in the :ref:`User Guide <cross_validation>`.

In [18]:
         from sklearn.ensemble import RandomForestClassifier
```

stackoverflow


User guide: contents — scikit-le
scikit-learn.org/stable/user_guide.html
scikit learn
Install User Guide API Examples More
Go
Prev Up Next
Toggle Menu

scikit-learn 0.22
Other versions

Please cite us if you use the software.

Welcome to scikit-learn
scikit-learn Tutorials
Getting Started
User Guide
1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection
5. Visualizations
6. Dataset transformations
7. Dataset loading utilities
8. Computing with scikit-learn
Glossary of Common Terms and AP
Elements
Examples

User Guide

1. Supervised learning

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

## Let's model!

scikit learn

# 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 |
|---|---|
| 0 | Red |
| 1 | Green |
| 2 | Blue |
| 3 | Red |

→

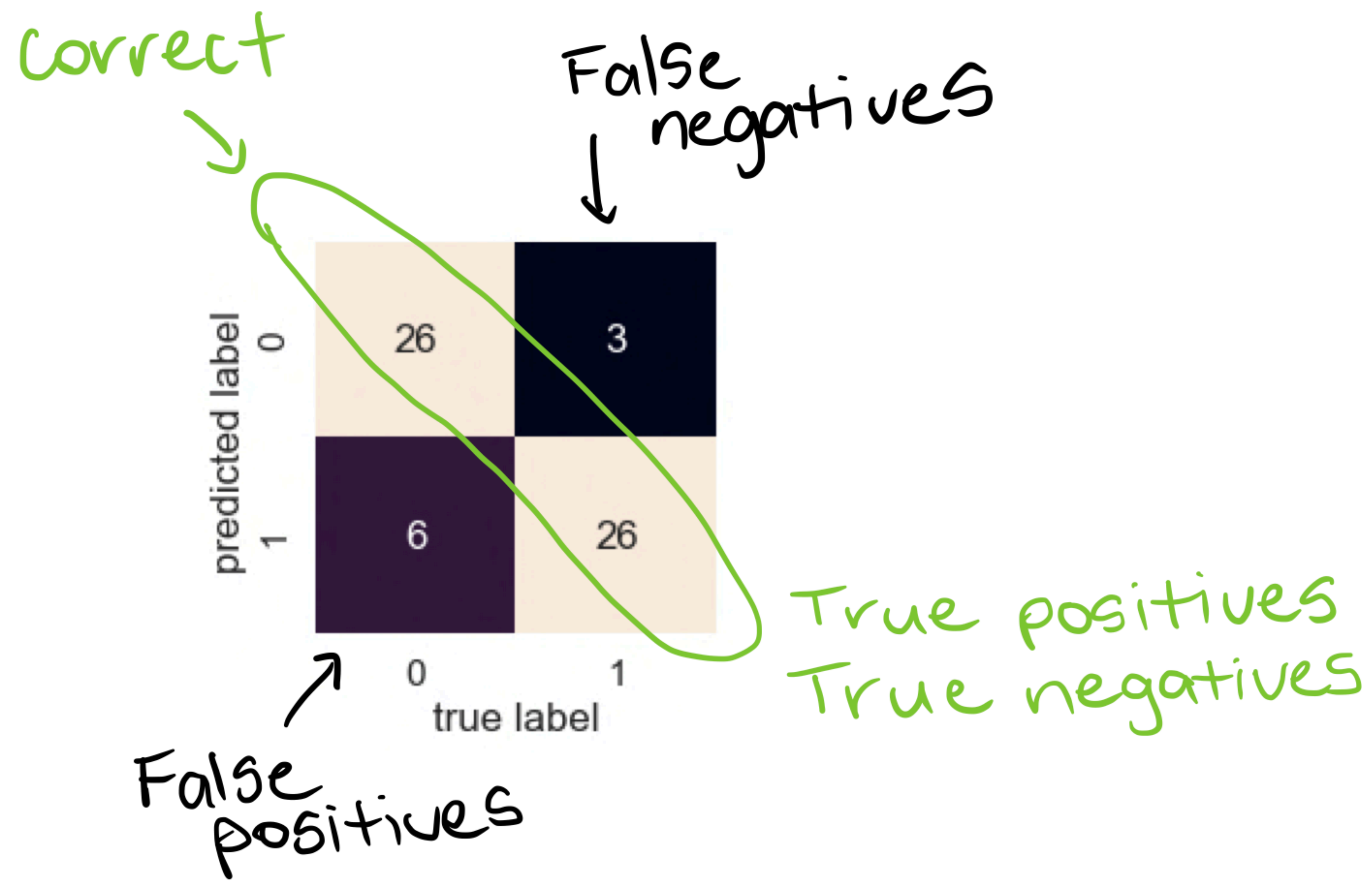
| Car | Red | Green | Blue |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 1 | 0 | 0 |

# Classification and Regression metrics

| Classification | Regression |
|---|---|
| **Accuracy** | **R² (r-squared)** |
| Precision | Mean absolute error (MAE) |
| Recall | Mean squared error (MSE) |
| F1 | Root mean squared error (RMSE) |

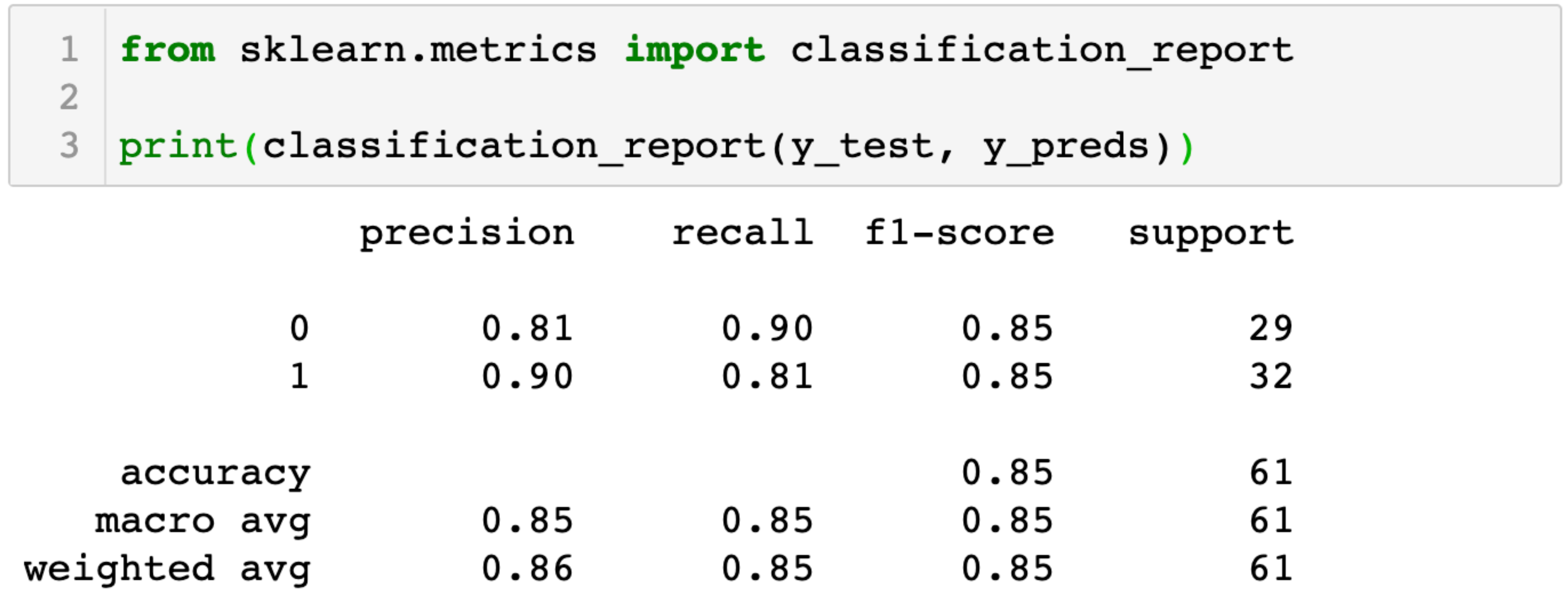
**Bold** = default evaluation in Scikit-Learn

# 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



```python
from sklearn.metrics import classification_report

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).

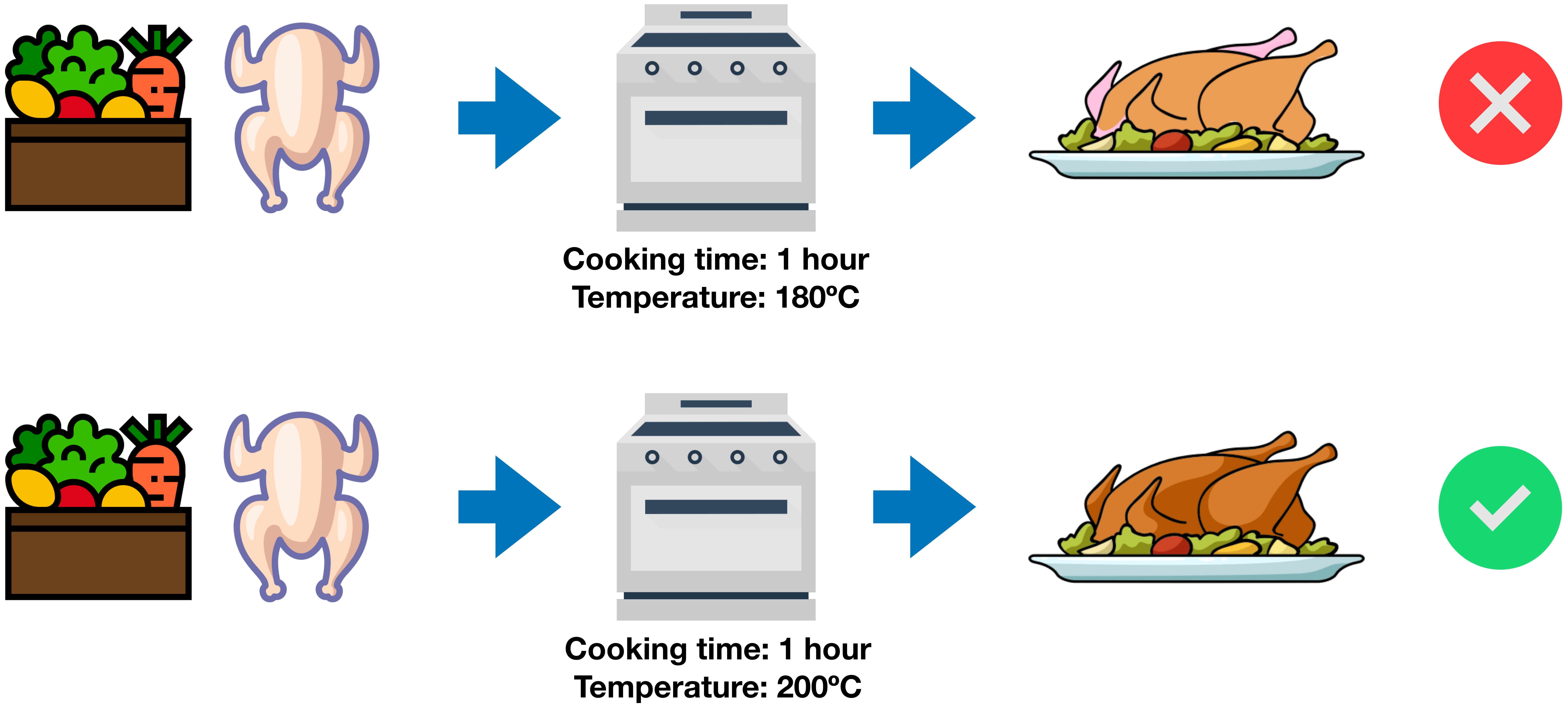
## Which classification metric should you use?

- **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.

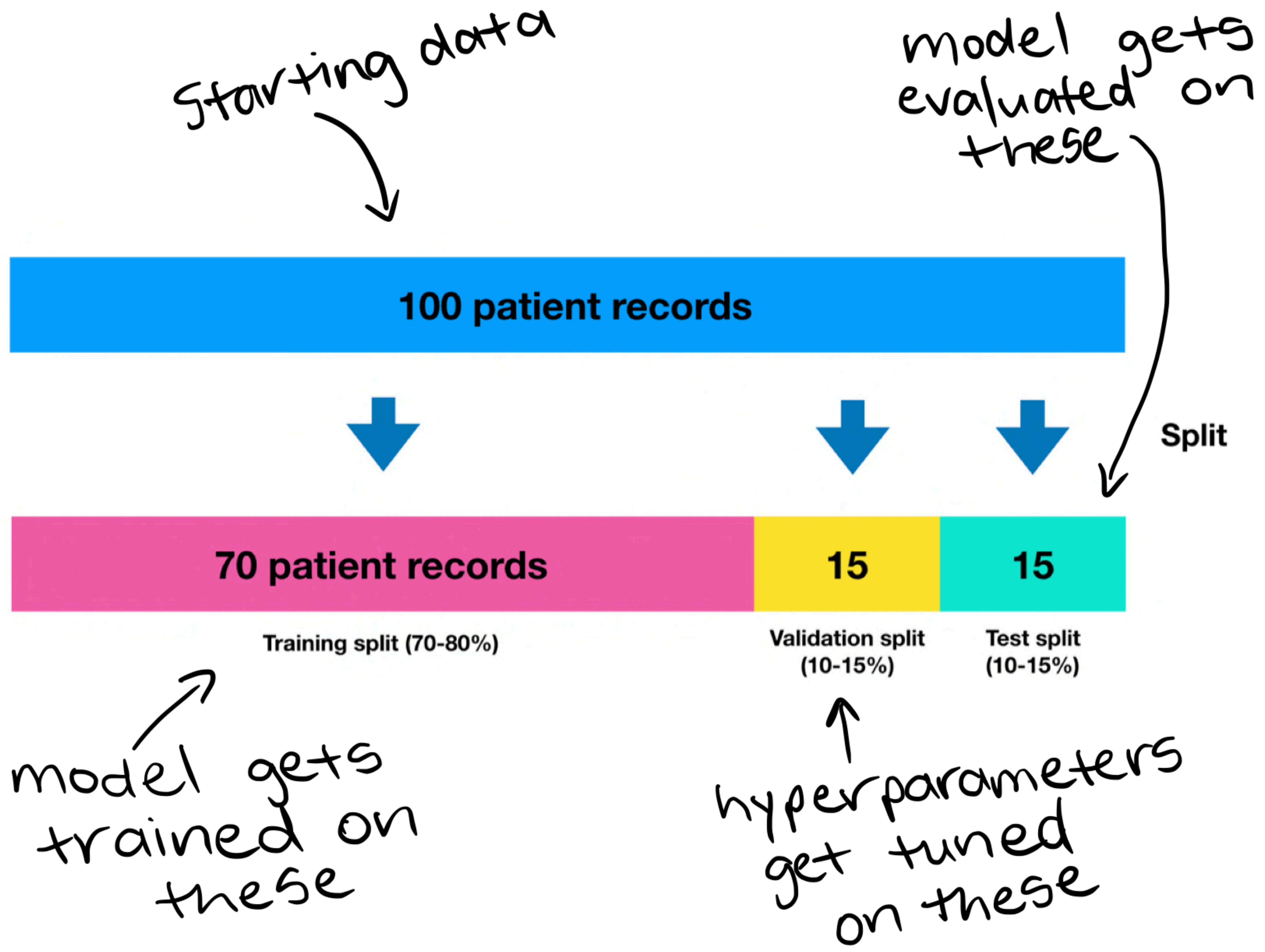
## Which regression metric should you use?

- **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)

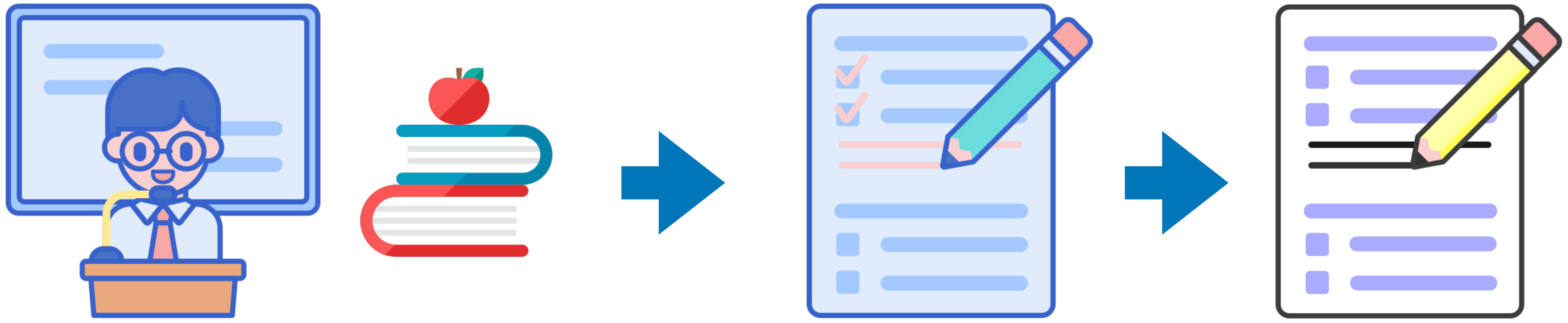


# Tuning Hyperparameters by Hand


Starting data
model gets evaluated on these
100 patient records
Split
70 patient records
15
15
Training split (70-80%)
Validation split (10-15%)
Test split (10-15%)
model gets trained on these
hyperparameters get tuned on these

# The most important concept in machine learning

(the 3 sets)



**Course materials (training set)**

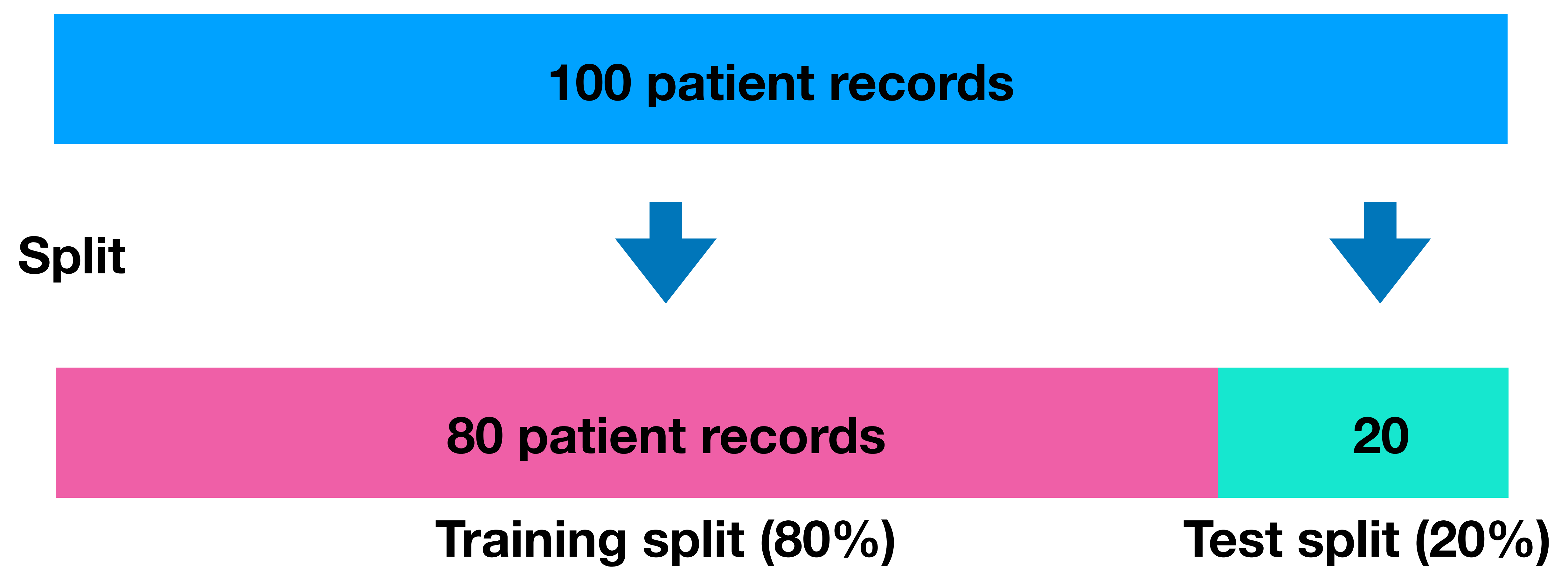
**Practice exam (validation set)**

**Final exam (test set)**

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

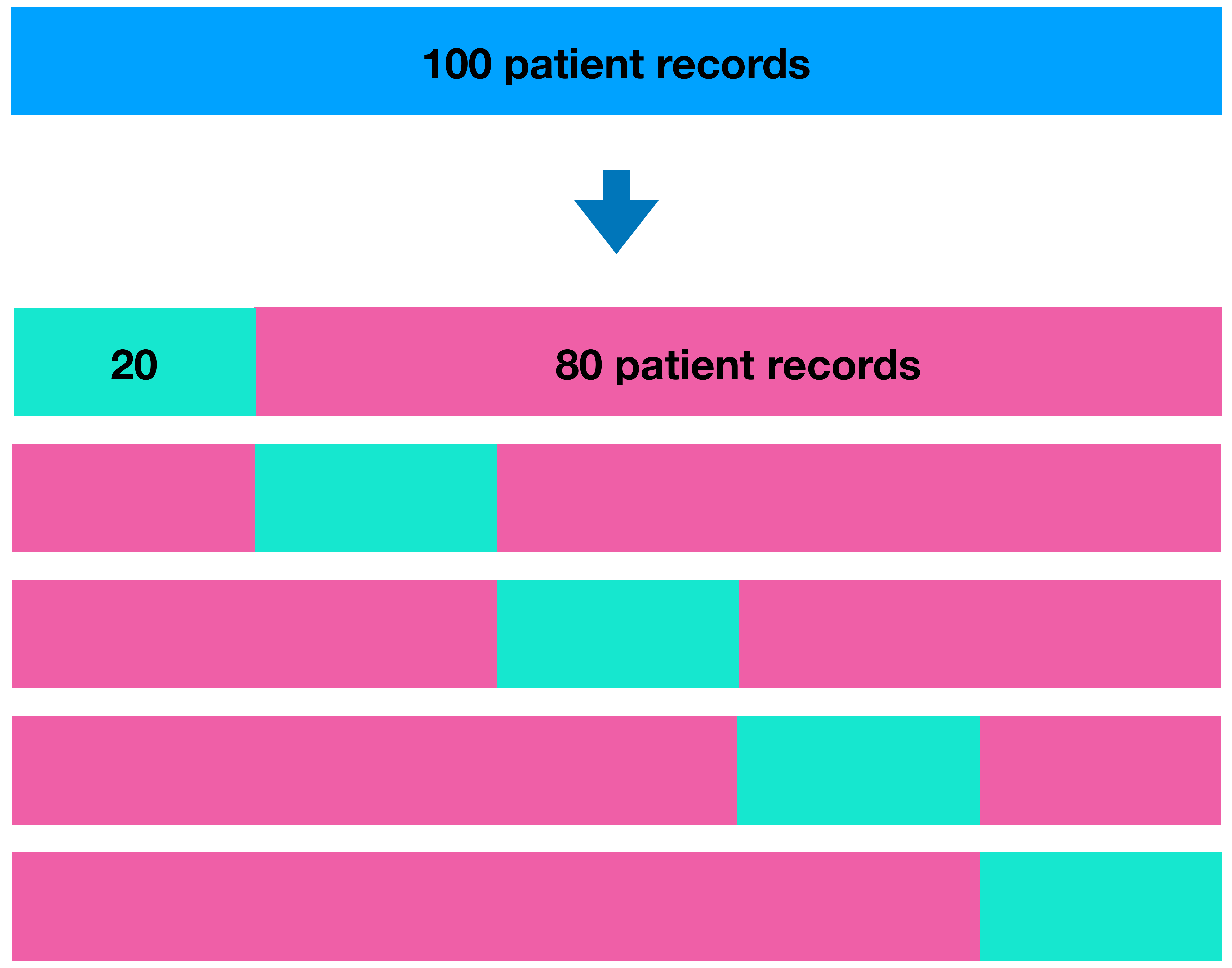
# Cross-validation

## Normal Train & Test Split



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

## 5-fold Cross-validation



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

## Normal Train & Test Split

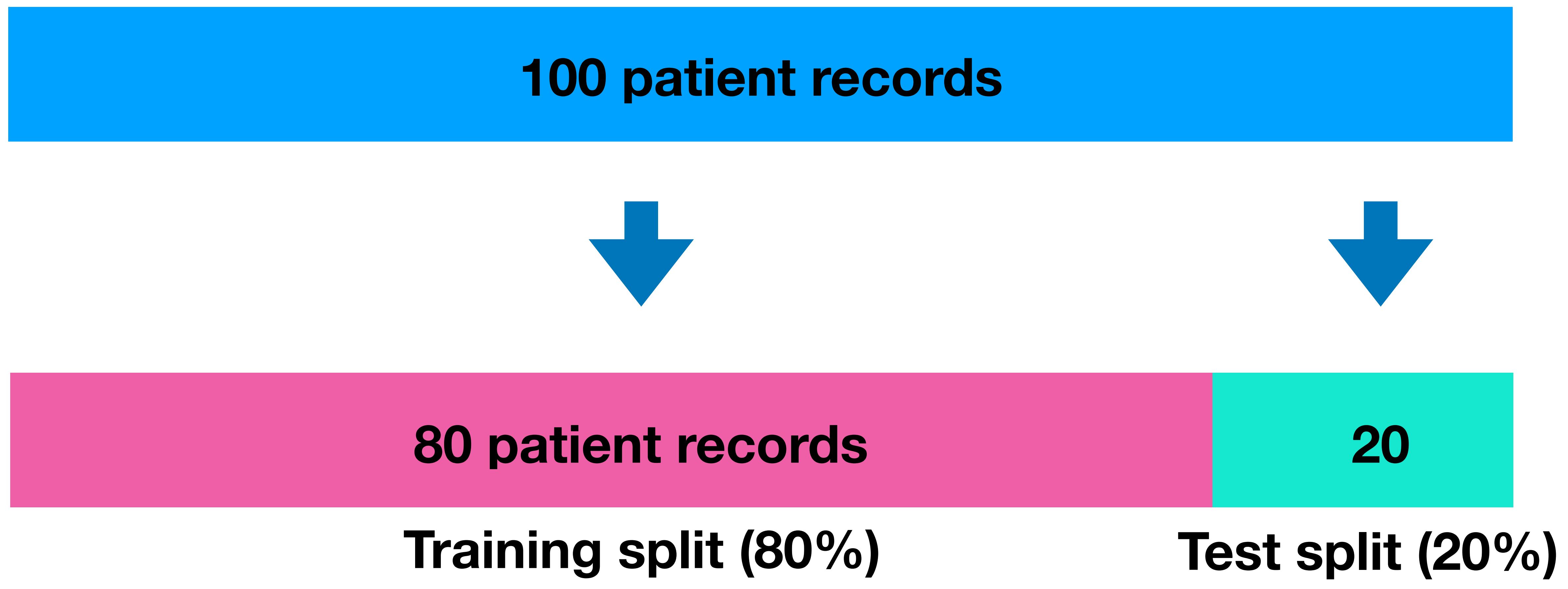


Figure 1.0: Model is trained on training data, and evaluated on the test data.

## 5-fold Cross-validation

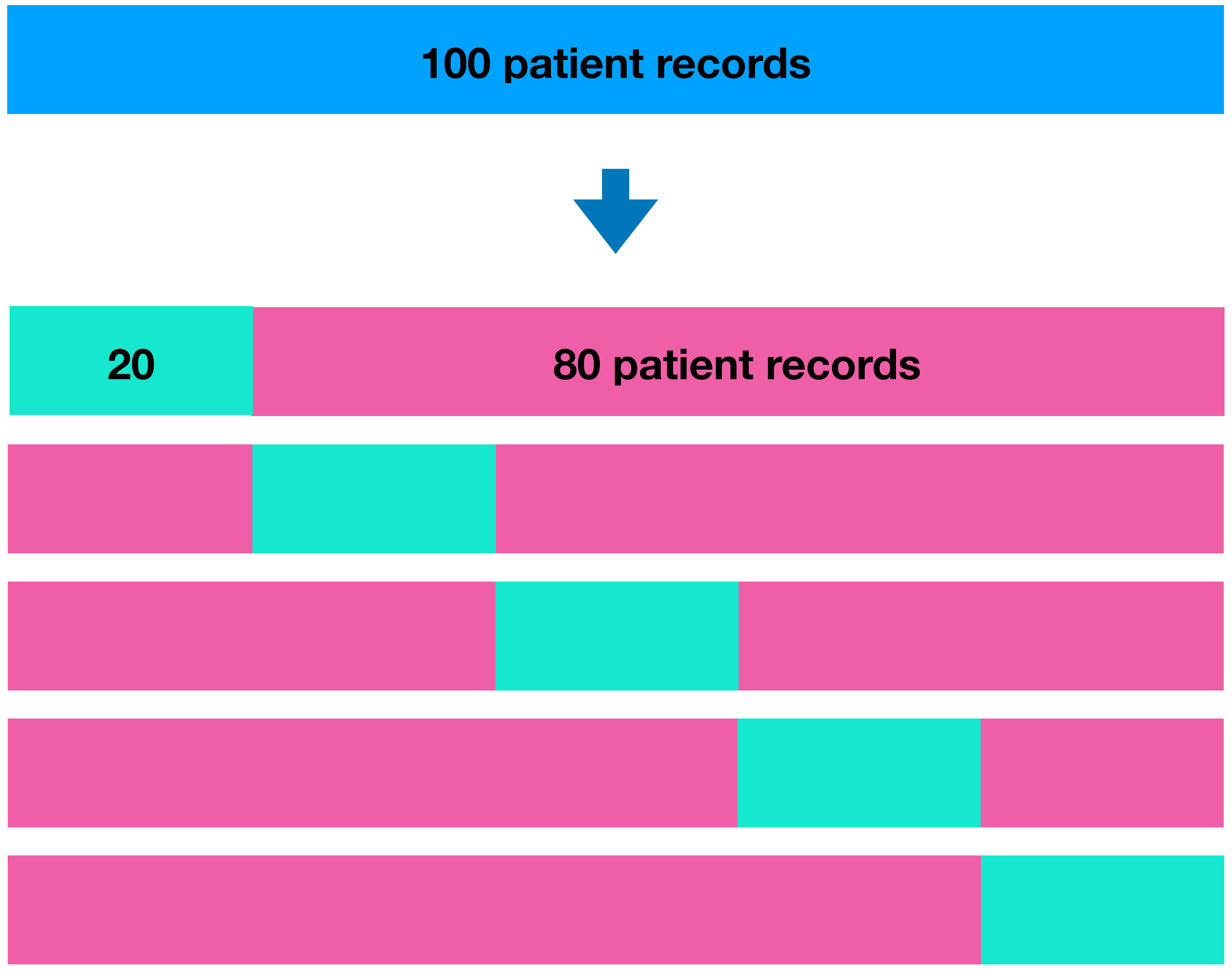


Figure 2.0: Model is trained on training data, and evaluated on the test data.

## 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