We will delve into the details of

- Classification task
- Dealing with Non-numeric data

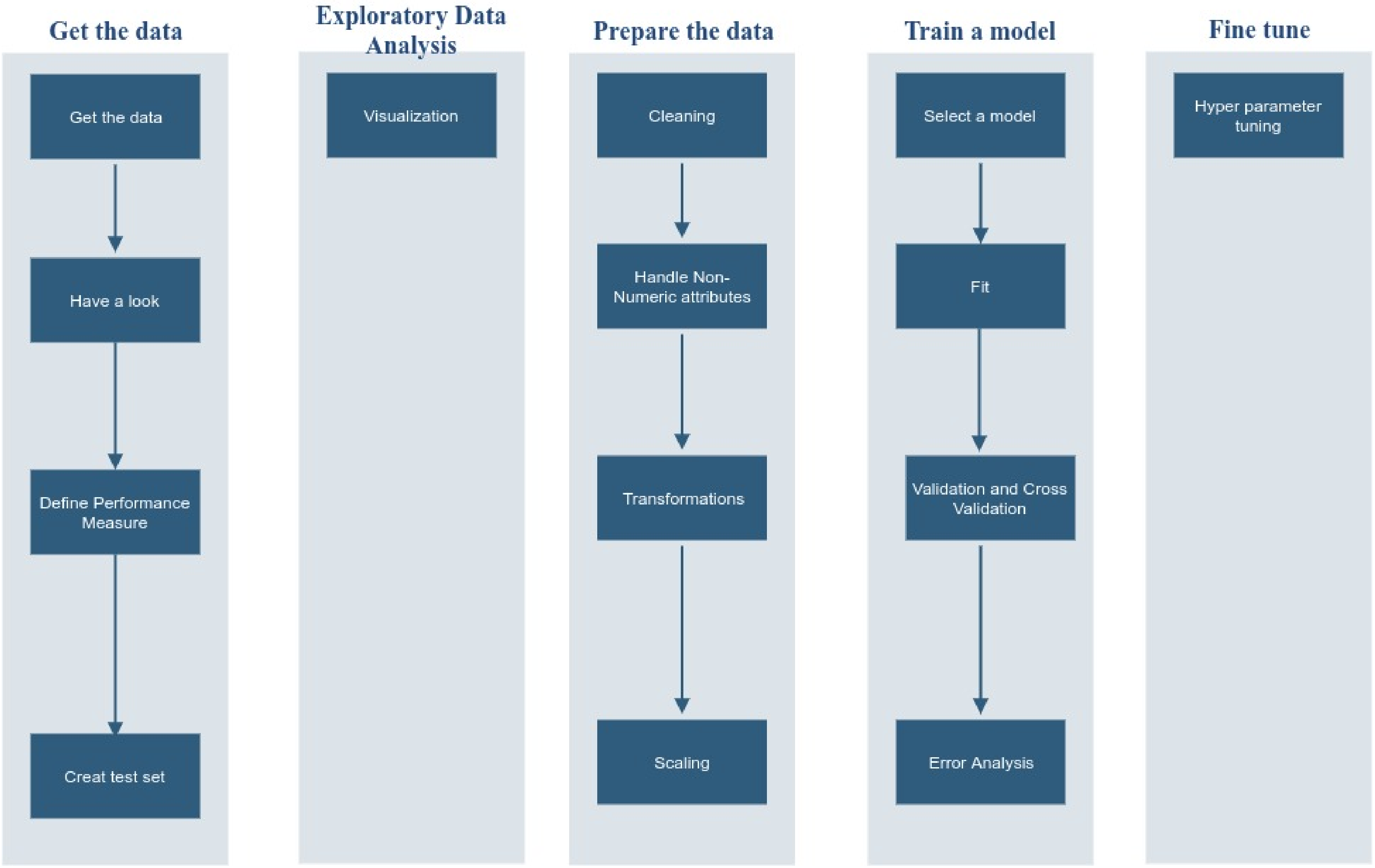
Recall: a Classification task has targets that are discrete values.

We will begin with the simple Binary Classification task in which the targets have only two possible values

- Positive
- Negative

We will explore an example in depth, following our Recipe for Machine Learning.

Recipe for Machine Learning



Get the data

Frame the problem

We will introduce the Classification task using an historical event: the sinking of the Titanic.

(Also the subject of a famous movies)

Our task: given attributes of a passenger on the ship, predict whether the passenger survived.

Let's visit the notebook section Get the data (Classification_and_Non_Numerical_Data.ipynb#Frame-the-problem) and perform the following steps

- Frame the problem
- Get the data
- Have a look at the data

Define a Performance Measure

Create a test set

For Regression (continous variable as target), RMSE was our Performance Measure.

What is an appropriate measure for a *discrete* target ?

Let's visit the notebook section Define a Performance Measure (Classification_and_Non_Numerical_Data.ipynb#Recipe-A.3:-Select-a-performance-measure) and perform the following steps

- Define a performance measure
- Create a test set

Recipe Step B: Exploratory Data Analysis (EDA)

There are quite a few features available.

Exploratory Data Analysis can help us

- Understand each feature in isolation (e.g., distribution)
    - May cause us to transform the feature, e.g., scaling
- Understand a possible relationship between target (survival) and each feature
    - Is this feature useful for predicting survival ?
- Understand possible relationships between features
- Possibly suggest creating new, synthetic features that alter or combine raw features

Visualize Data to gain insights

Let's visit the notebook section Visualize Data (Classification_and_Non_Numerical_Data.ipynb#Recipe-Step-B:-Exploratory-Data-Analysis-(EDA))

Prepare the data

Time to get our data ready. The steps we will perform include:

- Cleaning
- Handling non-numeric attributes
- Transformations

We will begin by discussing our plans for each step.

Let's visit the notebook section Prepare the data (Classification_and_Non_Numerical_Data.ipynb#Recipe-Step-C:-Prepare-the-data)

Code for Prepare the Data

We will use an `sklearn` Pipeline to implement the Prepare the Data Step

- One Pipeline devoted to numeric features
- One Pipeline devoted to categorical features
- a `FeatureUnion` Pipeline that combines the numeric and categorical pipelines

Let's return to the notebook section Using a Pipeline (Classification_and_Non_Numerical_Data.ipynb#Recipe-Step-C--using-a-sophisticated-pipeline) to see the code.

Train a model

We will perform the following steps

- Select a model
- Fit
- Cross Validation

The main classification model we will discuss is Logistic Regression.

But, it turns out:

- It is no harder to train *several* models than it is to just train one !
- So we will train a number of classification models, even before we formally introduce them
- This is the power of `sklearn` and similar toolkits for Machine Learning
    - All models have a consistent API: `fit`, `transform`, `predict`

So, after today's lecture: it will no longer be about the code

- The code to use most models is nearly identical, thanks to the consistent API
- My notebooks will be less "code-heavy"
    - The code will be isolated into separate modules, which you can examine
    - You just won't see the body in the notebook

If our goal was to learn an API, we'd be done.

But our goal is to pursue a systematic approach to problem solving in Machine Learning, with an emphasis

- On process
- Understanding concepts, loss functions, etc.
- Diagnosing problems with models and improving them

Let's return to the notebook section Train a model (Classification_and_Non_Numerical_Data.ipynb#Recipe-Step-D:-Train-a-model)

Error Analysis

Let's introduce some concepts relevant to analyzing errors for the Classification task.

We will be very brief, for now. We will explore this topic in depth in a dedicated module on Error Analysis.

Let's return to the notebook section Error Analysis (Classification_and_Non_Numerical_Data.ipynb#Recipe-D.4:--Error-analysis)

In [1]:

```
print("Done")
```

```
Done
```