# Chapter 2. End-to-End Machine Learning Project

1. Look at the big picture
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for Machine Learning algorithms.
5. Select a model and train it.
6. Fine-tuning your model.
7. Present you solution.
8. Launch, monitor, and maintain your system.

## Working the Real Data

## Look at the Big Picture

### Frame the Problem

Questions: 1 objective; 2 current solution; 3 type of problem: supervised, unsupervised, or Reinforcement Learning? classification/regression, batch/online learning?

Pipeline: a sequence of data processing components. Each component is self-contained (architecture robust, but uneasy to detect broken component)

### Select a Performance Measure

*Root Mean Square Error* (RMSE) ($l_2$, Euclidian norm) measures *Standard Deviation* ( $\sigma = \sqrt{Variance}$):

$$RMSE(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}^{(i)}) - y^{(i)})^2}$$

$h$ - hypothesis; $\mathbf{x}$ - matrix with $m$ rows of instances; each column is a feature; $i^{th}$ instance.

*Mean Absolute Error* (MAE) (Average Absolute Deviation, $l_1$, Manhattan norm)

$$MAE(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}|h(\mathbf{x}^{(i)}) - y^{(i)}|$$

$l_k$ norm $||\mathbf{v}||_k = (|v_0|^k + |v_1|^k + \cdots + |v_n|^k)^{1/k}$, $l_0$ - cardinality of the vector, $l_\infty$ - maximum absolute value in the vector. The higher the norm index, the more it focuses on large values and neglects small ones.

### Check the Assumptions

## Get the Data

#### Create the Workspace

#### Download the Data

Write scripts to fetch and load data.

#### Take a Quick Look at the Data Structure

DataFrame's head(), info(), value_counts(), describe() methods.

Histogram, hist() method in Matplotlib. Many histograms are *tail heavy*, transform them (e.g., by computing their logarithm).

#### Create a Test Set

Avoid *data snooping* bias.

When dataset is not large enough (relative to number of attributes), purely random sampling method could introduce a significant sampling bias.

*Stratified sampling*: divides the entire population into different homogeneous subgroups called *strata*, then randomly selects the final subjects **proportionally** from the different strata.

## Discover and Visualize the Data to Gain Insights

#### Visualizing Geographical Data

#### Looking for Correlations

*Standard correlation coefficient* (*Pearson's r*).

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

$cov$: covariance

Note: correlation coefficient only measures linear correlations.

corr() method or Pandas' scatter_matrix function

#### Experimenting with Attribute Combinations

It is an iterative process.

## Prepare the Data for Machine Learning Algorithms

Write functions to prepare data.

- Reproduce these transformations easily on any dataset.
- Gradually build a library of transformation functions that you can reuse in future projects.
- Use these functions in your live system to transform the new data.
- Easily try various transformations and see which combination of transformations works best.

### Data Cleaning

If some attribute has some missing values, 3 options to fix this:

- Get rid of the corresponding districts.
- Get rid of the whole attribute.
- Set the values to some value (zero, the mean, the median, etc.).

### Handling Text and Categorical Attributes

Convert text labels to numbers.

*One-hot encoding*: create one binary attribute per category.

### Custom Transformers

Scikit-Learn relies on duck typing (not inheritance), create a class and implement three methods: fit() (returning self), transform(), and fit_transform().

### Feature Scaling

Min-max scaling (normalization):

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \in [0, 1]$$

Standardization:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Stand deviation: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$

Unlike min-max scaling, standardization does not bound values to a specific range, which may be a problem for some algorithms (e.g., neural networks often expect an input value ranging from 0 to 1). However, standardization is much less affected by outliers.

### Transformation Pipelines

## Select and Train a Model

### Training and Evaluating on the Training Set

### Better Evaluation Using Cross-Validation

*K-fold cross-validation*: it randomly splits the training set into $K$ distinct subsets called *folds*, then it trains and evaluates the model $K$ times, picking a different fold for evaluation every time and training on the other $K - 1$ folds.

The goal is to shortlist a few (two to five) promising models, before dive much deeper in particular model.

*TIP*

> You should save every model you experiment with, so you can come back easily to any model you want. Make sure you save both the hyperparameters and the trained parameters, as well as the cross-validation scores and perhaps the actual predictions as well. This will allow you to easily compare scores across model types, and compare the types of errors they make.

## Fine-Tune Your Model

### Grid Search

Bootstrap: random sampling with replacement. (reason: test the stability of a solution)

### Randomized Search

When the hyperparameter search space is large, it is often preferable to use RandomizedSearchCV instead.

Benefits:

- If you let the randomized search run for, say, 1,000 iterations, this approach will explore 1,000 different values for each hyperparameter (instead of just a few values per hyperparameter with the grid search approach).
- You have more control over the computing budget you want to allocate to hyperparameter search, simply by setting the number of iterations.

### Ensemble Methods

Combine the models that perform best, especially if the individual models make very different types of errors.

### Analyze the Best Models and Their Errors

Random Forest Regression can indicate the relative importance of each attribute for making accurate predictions. With this information, you may want to try dropping some of the less useful features.

You should also look at the specific errors that your system makes, then try to understand why it makes them and what could fix the problem (adding extra features or, on the contrary, getting rid of uninformative ones, cleaning up outliers, etc.).

### Evaluate Your System on the Test Set

The performance will usually be slightly worse than what you measured using cross-validation if you did a lot of hyperparameter tuning. DON'T tweak the hyperparameters to make the numbers look good on the test set.

## Launch, Monitor, and Maintain Your System

Monitoring code to check your system’s live performance at regular intervals.

Plug human evaluation pipeline into your system.

Monitoring the inputs quality (particularly important for online learning systems).

Automatically train your models on a regular basis using fresh data. (online learning system, make sure you save snapshots of its state at regular intervals so you can easily roll back to a previously working state.)