

# Machine learning

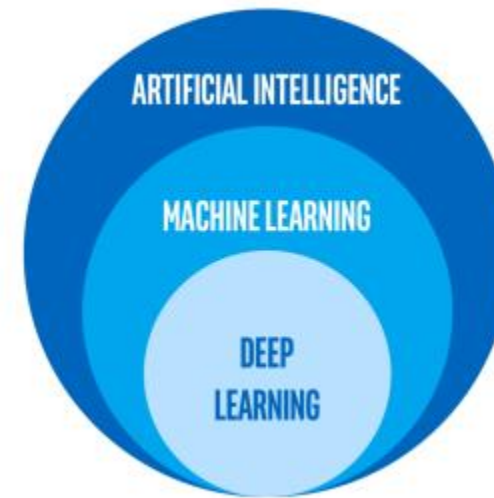
---

Hands on python and sklearn

# Machine learning

**Machine Learning** is the science (and art) of programming computers so they can learn from data

- Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed — Arthur Samuel, 1959
- A computer program is said to learn from experience  $E$  with respect to some task  $T$  and some performance measure  $P$ , if its performance on  $T$ , as measured by  $P$ , improves with experience  $E$  — Tom Mitchell, 1997



# Machine learning

There are many types of Machine Learning algorithms

Classify them in broad categories, based on the following criteria:

- Whether they are trained with human supervision
  - supervised, unsupervised, semi-supervised, and reinforcement learning
- Whether they can learn incrementally
  - online, batch learning
- Whether they compare new to known data points, or detect patterns/models in the training
  - instance-based, model-based learning

In this session, the focus is not on the different models of ML

- We stick to "classical" ML algorithms

# Machine learning

## Supervised learning tasks

- The training set you feed to the algorithm includes the desired solutions, called labels
- **Classification**
  - Approximating a mapping function ( $f$ ) from input variables ( $X$ ) to **discrete** output variables ( $y$ )
  - The output variables are called labels or categories
  - The mapping function predicts the class or category for a given observation
  - E.g., a spam filter is trained with many example emails along with their class (spam or ham)
- **Regression**
  - Approximating a mapping function ( $f$ ) from input variables ( $X$ ) to a **continuous** output variable ( $y$ )
  - A continuous output variable is a real-value, such as an integer or floating-point value
  - E.g., predict the price of a car given a set of features (mileage, age, brand, etc.) called predictors

# Sklearn

**Scikit-learn (Sklearn)** is a well-known library for ML in Python

- This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib
  - Open source and commercially usable
- Covers many algorithms
  - Supervised Learning algorithms: Linear Regression, Support Vector Machine, etc.
  - Unsupervised Learning algorithms: clustering, factor analysis, PCA, neural networks, etc.
  - Cross Validation: check the accuracy of supervised models on unseen data
  - Feature extraction: extract the features from data to define the attributes in image and text data

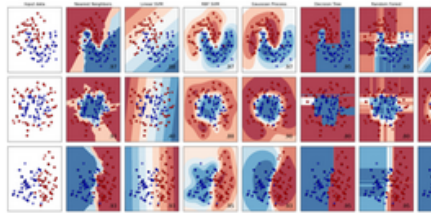
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

### Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, and more...



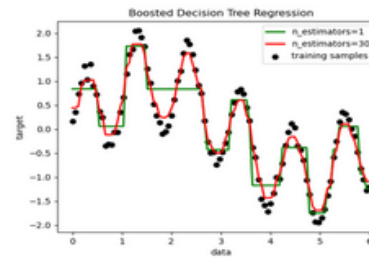
Examples

### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, nearest neighbors, random forest, and more...



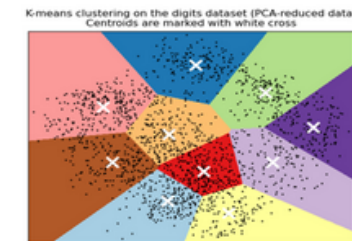
Examples

### Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, and more...



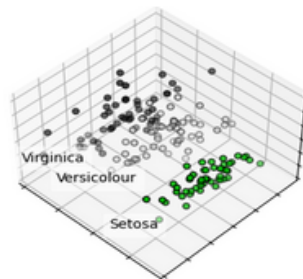
Examples

### Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** k-Means, feature selection, non-negative matrix factorization, and more...



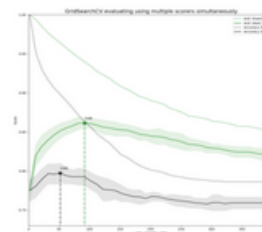
Examples

### Model selection

Comparing, validating and choosing parameters and models.

**Applications:** Improved accuracy via parameter tuning

**Algorithms:** grid search, cross validation, metrics, and more...



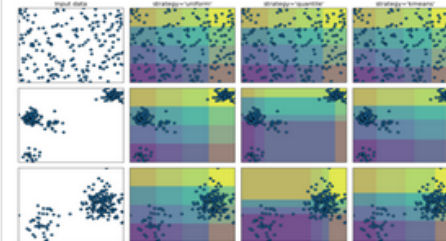
Examples

### Preprocessing

Feature extraction and normalization.

**Applications:** Transforming input data such as text for use with machine learning algorithms.

**Algorithms:** preprocessing, feature extraction, and more...



Examples

# Sklearn

## Scikit-learn uses data in the form of N-dimensional matrix

- Data as a **feature matrix** (e.g., a Pandas DataFrame)
  - The samples represent the individual objects described by the dataset (e.g., a person)
  - The features describe each sample in a quantitative manner (e.g., age and height)
  - It is usually denoted by **X**
- Data as **target array** (e.g., a Pandas Series)
  - Along with features matrix, we also have the target array (e.g., or label)
  - It is usually denoted by **y**
- How do we distinguish target and feature columns?

# Estimator

## Estimator

- A consistent interface for a wide range of ML applications
- The algorithm that learns from the data (fitting the data) is an estimator
- It can be used with any of the algorithms like classification, regression, and clustering

All the parameters can be set when creating the estimator

- `>>> estimator = Estimator(param1=1, param2=2)`
- `>>> estimator.param1`

All estimator objects expose a `fit` method that takes a dataset

- `>>> estimator.fit(X)`

Once data is fitted with an estimator, all the estimated parameters will be the attributes of the estimator object ending by an underscore

- `>>> estimator.estimated_param_`



# Estimator

1. Choose a class of model
  - Import the appropriate Estimator class from Scikit-learn (e.g., a decision tree)
2. Choose model hyperparameters
3. Arranging the data
  - Arrange the data into features matrix  $X$  and target vector  $y$
4. Model Fitting
  - Fit the model by calling `fit()` method of the model instance
5. Applying the model to new data
  - For supervised learning, use `predict()` method to predict the labels for unknown data.
  - For unsupervised learning, use `predict()` or `transform()` to infer properties of the data.

# Estimator

## 1. Choose a class of model

- `>>> from sklearn.linear_model import LinearRegression`

## 2. Choose model hyperparameters

- `>>> model = LinearRegression(fit_intercept = True)`

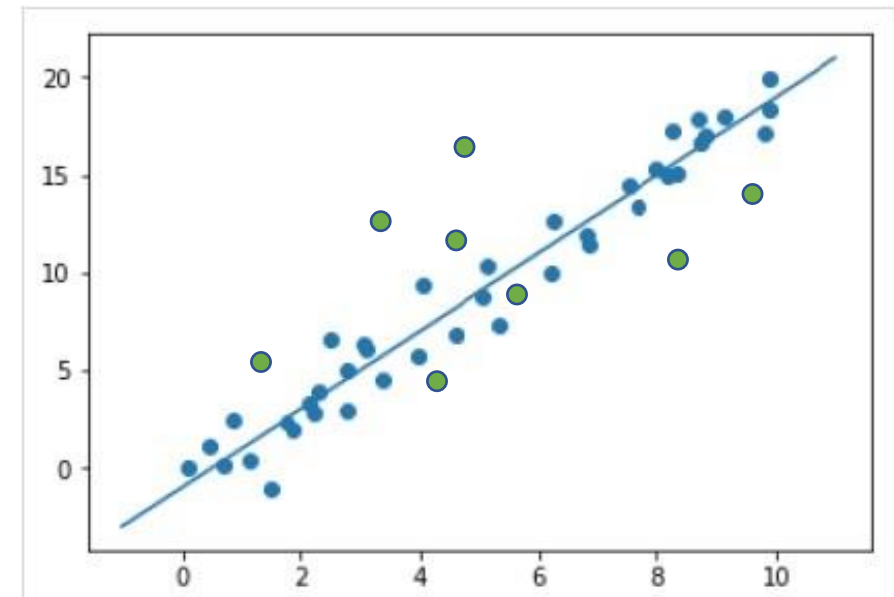
## 3. Arranging the data

## 4. Model fitting

- `>>> model.fit(X, y)`
- `>>> model.coef_`

## 5. Applying the model to new data

- `>>> model.predict(new_X)`



# Integrated analytics lab

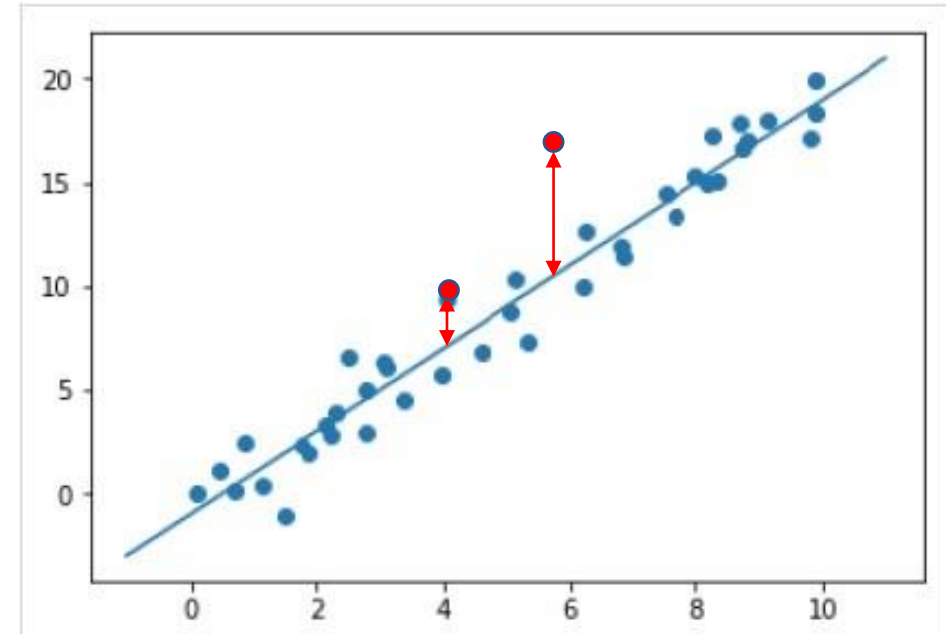
This checklist can help you while building your projects

- Frame the problem and look at the big picture
  - ✓ Define the objective in business terms
  - ✗ How should performance be measured? (let's do this!)

# Integrated analytics lab

We are facing a regression problem

- A typical performance measure for regression problems is the Root Mean Square Error (RMSE)
- RMSE is the standard deviation of the **residuals** (prediction errors)
- Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are

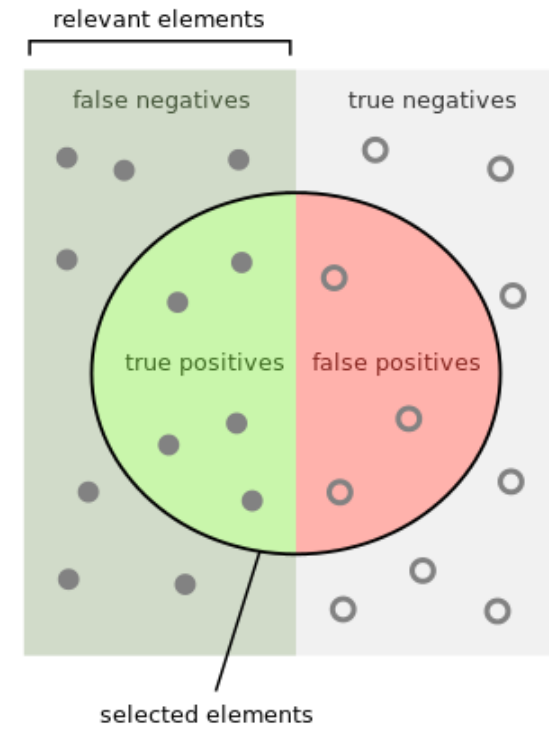


$$\text{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m} \sum_{i=1}^m \left( h(\mathbf{x}^{(i)}) - y^{(i)} \right)^2}$$

# Integrated analytics lab

(If) We are facing a classification problem

		Predicted condition	
Total population = P + N		Predicted condition positive (PP)	Predicted condition negative (PN)
Actual condition	Actual condition positive (P)	<b>True positive (TP), hit</b>	<b>False negative (FN), Type II error, miss, underestimation</b>
	Actual condition negative (N)	<b>False positive (FP), Type I error, false alarm, overestimation</b>	<b>True negative (TN), correct rejection</b>
Prevalence = $\frac{P}{P+N}$		Positive predictive value (PPV) = $\frac{TP}{PP}$ <b>precision</b> = 1-FDR	False omission rate (FOR) = $\frac{FN}{PN}$ = 1-NPV
<b>Accuracy (ACC) = <math>\frac{TP+TN}{P+N}</math></b>		False discovery rate (FDR) = $\frac{FP}{PP}$ = 1-PPV	Negative predictive value (NPV) = $\frac{TN}{PN}$ = 1-FOR
Balanced accuracy (BA) = $\frac{TPR+TNR}{2}$		<b>F<sub>1</sub> score = <math>\frac{2 \cdot PPV \cdot TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}</math></b>	<b>Fowlkes-Mallows index (FM) = <math>\sqrt{PPV \cdot TPR}</math></b>



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

# Integrated analytics lab

Precision

$$\text{Precision} = \frac{tp}{tp + fp}$$

Recall

$$\text{Recall} = \frac{tp}{tp + fn}$$

Accuracy

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

- Accuracy can be a misleading metric for imbalanced data sets

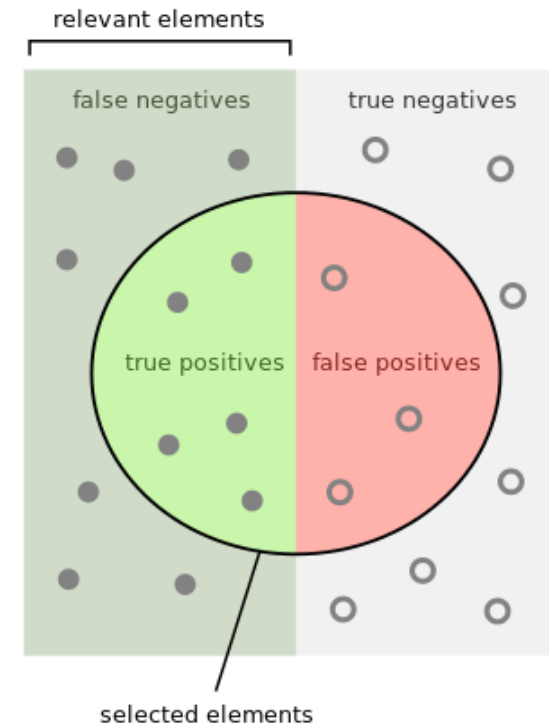
F1-score

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- Combines precision and recall

Summing up

- Accuracy is used when TP and TN are more important while F1-score is used when FN and FP are
- Accuracy can be used when the class distribution is similar, while F1-score is a better when there are imbalanced classes



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

# Hyper-parameter tuning

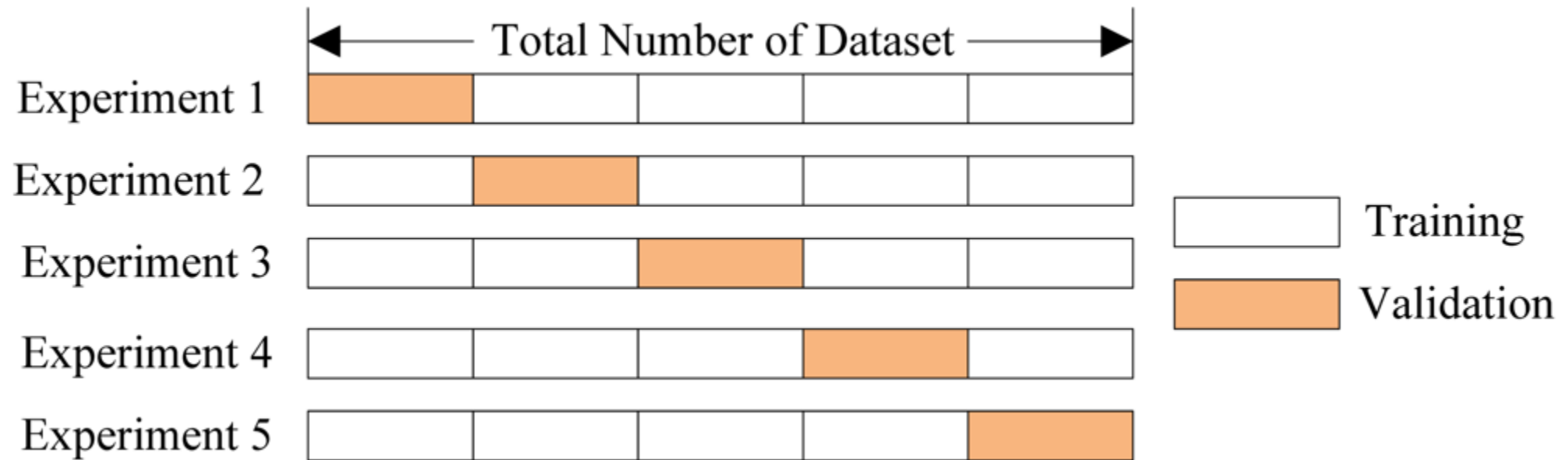
**Hyper-parameters:** parameters that are not directly learnt within estimators

- In scikit-learn they are passed as arguments to the constructor of the estimator classes
- Any parameter provided when constructing an estimator may be optimized
  - `>>> estimator.get_params()`

A search consists of:

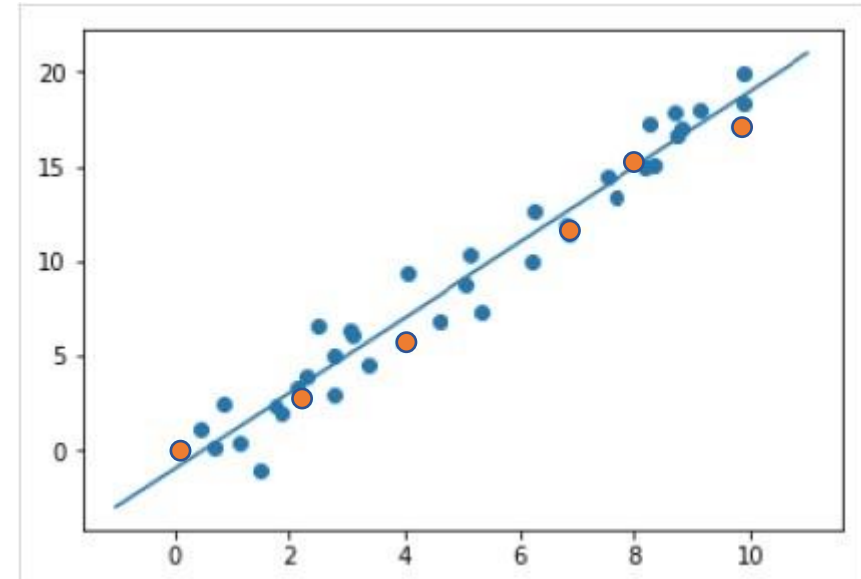
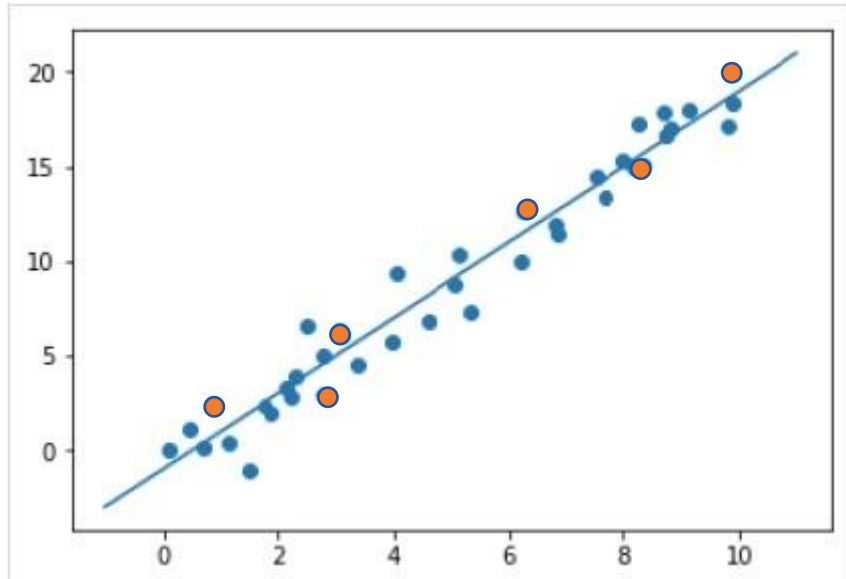
- an estimator
- a parameter space
- a method for searching or sampling candidates
- a cross-validation scheme
- a score function

# Cross validation





# Cross validation



# Integrated analytics lab

This checklist can help you while building your projects

- Frame the problem and look at the big picture
  - ✓ Define the objective in business terms
  - ✓ How should performance be measured?
- Get the data
  - ✓ List the data you need and how much you need
- Explore the data to gain insights
  - ✓ Create an environment to keep track of your data exploration
  - ✓ Study each attribute and its characteristics
- Prepare the data
  - ✓ Fix or remove outliers (optional)
  - ✓ Fill in missing values (e.g., with zero, mean, median...) or drop their rows (or columns)
  - ✓ Feature selection (optional): drop the attributes that provide no useful information for the task
  - ✓ Feature engineering, where appropriate: discretize continuous features
- Explore many different models and shortlist the best ones
  - ✓ Let's do this!

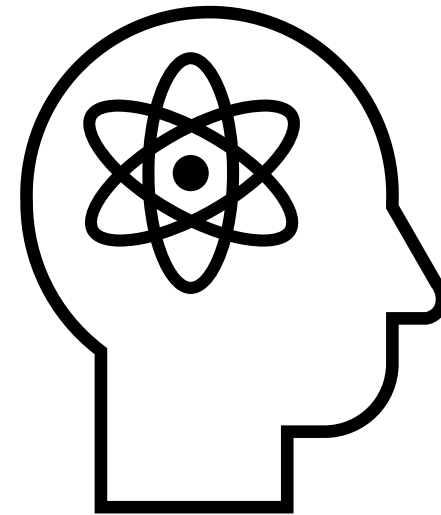
# In action!



Open the browser



Enter the notebook `03-Housing-MachineLearning`



<https://github.com/w4bo/2022-bbs-dm/blob/main/notebooks/03-Housing-MachineLearning.ipynb>

# Integrated analytics lab

This checklist can help you while building your projects

- ✓ Frame the problem and look at the big picture
- ✓ Get the data
- ✓ Explore the data to gain insights
- ✓ Prepare the data
- ✓ Explore many different models and shortlist the best ones
- ✓ Fine-tune your models and combine them into a great solution
- Present your solution
- Launch, monitor, and maintain your system